IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 12, 2020, accepted September 25, 2020, date of publication September 30, 2020, date of current version October 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028031

Enhancing Vendor Managed Inventory Supply Chain Operations Using Blockchain Smart Contracts

ILHAAM A. OMAR[1], RAJA JAYARAMAN[1], KHALED SALAH[2], (Senior Member, IEEE), MAZIN DEBE[2], AND MOHAMMED OMAR[1]

[1]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates
[2]Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, United Arab Emirates

Corresponding author: Raja Jayaraman (raja.jayaraman@ku.ac.ae)

This work was supported by the Khalifa University of Science and Technology–Research Center for Digital Supply Chain and Operations Management under Grant RCII-2019-002.

ABSTRACT Supply chain networks have grown in complexity and size due to increased globalization leading to a variety of challenges and opportunities for improvement. Optimizing inventory levels and adjusting replenishment policies have significant effects on the operational performance and profitability of supply chains. Vendor Managed Inventory (VMI) is a mutually beneficial arrangement between supplier and buyer, where the supplier is responsible for making inventory and replenishment decisions based on buyers' inventory status. Potential benefits of VMI include reducing inventories, enabling information sharing, eliminating safety stock, and reducing purchasing related costs across the supply chain. In today's supply chains, VMI operations face critical challenges related to data integrity, transparency, traceability, and single point of failure due to its centralized architecture. Blockchain technology is a distributed ledger that ensures a transparent, safe, and secure exchange of data among supply chain stakeholders. The advantages of adopting blockchain technology for VMI operations in a supply chain include decentralized control, security, traceability, and auditable time-stamped transactions. In this paper, we present a blockchain-based approach using smart contracts to transform VMI supply chain operations. We propose a generic framework using Ethereum smart contracts and decentralized storage systems to automate the processes and information exchange and detailed algorithms that capture the interactions among supply chain stakeholders. The smart contract code was developed and tested in Remix environment. We present cost and security analysis incurred by the stakeholders in the supply chain. Adopting a blockchain-based solution to VMI operations in supply chains is economically viable and provides a streamlined, secure, trusted, and transparent mode of communication among various stakeholders.

INDEX TERMS Blockchain applications, Ethereum, smart contracts, security, data integrity, supply chain management.

I. INTRODUCTION

Supply chain management is a core business function responsible for the movement of goods and services across several stakeholders. Supply chain broadly encompasses people, resources, activities, and organizations that are involved in transforming raw materials into finished products, fulfilling customer orders both directly and indirectly. The complexities of modern supply chains are exacerbated by many factors, including lack of transparency, disruptions, extra delays, information distortion, and uncertainties [1]. Effective inventory management acts as a critical lever in balancing demand and supply to achieve cost efficiency in the supply chain. Several strategies, such as demand-driven replenishment, collaborative planning, forecasting, and replenishment (CPFR), Just in Time (JIT), have been applied for inventory planning and replenishment. The exponential increase in data availability and implementation of technology-driven solutions results in effective inventory and replenishment policies leading to massive improvements in products and information

The associate editor coordinating the review of this manuscript and approving it for publication was SK. Hafizul Islam.

flow across the supply chain. The efficiency of the supply chain is enhanced by the integration of technology, leading to increased material production and shipment, and this very advantage causes distress to many businesses in the industry who are unable to keep up with these advancements. Vendor Managed Inventory (VMI) is a strategy that has been adopted by vendors to optimize supply chain processes as it offers many beneficial advantages to diverse stakeholders. Successful companies such as Wal-Mart, Hewlett Packard, and Ericsson Mobile Communication have benefitted immensely from using VMI arrangement to communicate with their suppliers [2].

VMI agreement between a vendor and a buyer essentially ensures restocking of supply autonomously with the maximum possible profit for both parties. The cost of inventory is one of the most expensive aspects of supply chain performance. Therefore, VMI effectively lowers inventory costs, eliminates safety stock, and enhances collaboration between suppliers and buyers in a way that improves accuracy and efficiency. In addition, VMI results in quicker restock, optimized allocation of inventory space, and increased returns for vendors while providing better service to buyers. Under VMI, the vendor assumes full responsibility for inventory strategy, execution, and delivery management of stock to the customer. This process eases customers the burden of constant tracking and replenishment of stock while allowing suppliers full freedom in devising the most cost-effective and efficient method of product supply. VMI practice is popular and commonly applied in multiple industries such as retail, restaurants, hospitality, construction, manufacturing, healthcare, and others.

VMI has a range of advantages when successfully implemented, and there would be an improvement in profit for all participating supply chain stakeholders. Such improved profits are generated by a combination of cost-cutting and increased sales and associated revenues [3]. To establish a VMI association, mutual benefit between both interested parties needs to be established. A customer benefits from alleviated responsibility along with a shift in planning and processing costs, thereby decreasing the costs for the customer. On the other hand, the vendor gains increased control over the inventory and ability to better predict demand since customers' data is readily available. Hence, inventory management efficiency is concentrated along with lean production, efficient transfers, and shipment transportation.

VMI success is centered on a high level of trust and efficient interaction between participating stakeholders [4]. The efficient flow of information is what makes it possible for VMI arrangements to succeed. Customers need to build a platform from which vendors can access and use the data from their company to maximize their revenue and inventory rates. They need to be able to trust that their vendors can make skilled and efficient use of this information. The data collection process or editing can be done by either the vendor or customers. The vendor can record and modify inventory data in person, or the customer can grant them electronic access. The customer continuously updates these data via an automated system, or personally if access to technology is unavailable. Accessing this information allows the vendor to monitor inventory status and replenish regularly, to prevent stockouts or overstock. Before implementing the VMI program, vendors and customers need to coordinate with each other to define baseline, peak, and reorder point rates as to when new stocks are to be ordered and how much is to be kept as on-hand inventory, this puts both parties in mutual consent.

However, the VMI vendor team faces several difficulties when collecting data. Some of these challenges include data integrity, accessibility, and information delay [2]. Other challenges include transparency of data, data centers centralization, and traceability of information within the stakeholders of the system. Blockchain technology can revolutionize the existing VMI process as this innovation has attracted various researchers and companies to investigate it in the field of systematization and supply network management [5], [6]. Blockchain has the capability to replace traditional CPFR practices across the supply chain and establish supply chain provenance [7], [8]. This technology establishes a platform in which all trading partners can coordinate and communicate with each other effectively [9]. In addition, information retrieval from various data sources such as retailer point of sale, inventories, and deliveries is made more accessible for vendors due to permissioned access to data on this platform. Moreover, blockchain synchronizes VMI operations between all stages of the supply chain as planning and execution processes are made transparent. Lastly, it can automatically generate alerts when stakeholders violate predefined conditions for replenishment stated in smart contracts.

In the area of supply chain collaboration, there is a scarcity of literature that addresses how blockchain technology can transform various supply chain operations. The primary objective of this paper is to propose a blockchain-based solution that helps to enhance the VMI model in supply chains. The main contributions of this work are summarized as follows:

- We propose an Ethereum blockchain-based solution which captures the key interactions between supply chain trading partners using an Ethereum smart contract and a decentralized storage system.
- We present and discuss the main smart contract code and its algorithms that define the underlying principles of the proposed blockchain approach. We also provide a detailed sequence diagram to explain the blockchain-based VMI solution.
- We validate and test various scenarios of the overall system functionalities of the proposed solution.
- We provide a cost and security discussion of the proposed system to show the feasibility of implementing our solution.

The remainder of this paper is organized as follows: Section II provides the background on the benefits of using VMI and explains the benefits of integrating

blockchain technology into the VMI process. Section III presents an overview of relevant literature. Section IV describes the proposed blockchain solution, while Section V defines the implementation approach. Section VI discusses the results for various test scenarios, and Section VII details the cost and security analysis along with the open research challenges facing our proposed solution. Finally, in Section VIII, we discuss the conclusions and future work.

II. BACKGROUND

In this section, we discuss the significance of VMI in supply chain management. We also explain the benefits of integrating blockchain technology with VMI.

A. IMPORTANCE OF VENDOR MANAGED INVENTORY (VMI)

VMI benefits both the front and back end of the supply chain as vendors and buyers realize synergistic advantages. One of the greatest benefits that vendors gain is obtaining precise demand information ahead of time, unlike in a traditional vendor-buyer relationship [3]. This enables vendors to respond quickly and efficiently to unexpected orders. Moreover, it allows the ordering process to take place in a shorter period as all decisions related to restocking products are made by the vendor. Furthermore, it allows them to stabilize purchase order frequency, and effectively manage inventory planning on-site, creating a consistent order schedule and reducing the number of rush orders [10]. Thus, building a VMI relationship aids in obtaining increased control over the vendor shipment schedule as their retailers provide sales and inventory data. This, in turn, helps mitigate bullwhip effect as large demand variance is reduced [11]. These variances are mainly due to actions driven by self-interest and miscommunication among supply chain stakeholders. Likewise, buyers benefit from increased sales and profitability due to fewer stockouts and administration costs incurred, quicker response time, and a better product mix [10]. VMI encourages better collaboration between vendors and buyers, allowing them to work cohesively. Therefore, VMI combines both companies' efforts efficiently to reduce carrying costs and the need for safety stock as this framework offers simplicity across the supply chain, facilitates improved communication among trading partners, and streamlines inventory management operations.

B. VMI THROUGH BLOCKCHAIN TECHNOLOGY

Effective replenishment strategy, such as VMI, creates an efficient platform to exchange information by eliminating the need for traditional orders. This process is usually carried out by VMI service providers that act as administrators in a centralized network [12]. However, using blockchain technology for VMI operations would eliminate the need for service providers resulting in little to no human intervention and reduced transactional costs [13]. This is possible through the use of second-generation blockchain platforms such as Ethereum, which uses smart contracts to execute verified transactions that meet predefined conditions [14]. Ethereum blockchain is a pseudo-anonymous blockchain which makes uses of Proof of Work (PoW) and Proof of Stake (PoS) consensus algorithms. PoW was the first consensus algorithm used in cryptocurrencies while PoS was developed as an alternative to PoW. The main difference among these two algorithms is that PoW utilizes miners and use their computational power (energy) to validate transactions in the blockchain, while PoS uses validators who must commit stake in order to validate blocks. Moreover, the inherent features of blockchain, such as time-stamped records and cryptographic techniques, aid in enhancing data security, data integrity, and immutability of stored transactions [15], [16]. Thus, blockchain empowers the VMI core principles, which include data sharing, transparency, and traceability.

Blockchain technology is a decentralized and distributed shared ledger that makes data sharing easy as each member of the network has a copy of all valid transactions in the ledger [17]. The transactions are added to the system in the forms of blocks, each of which holds multiple transactions within it. These blocks are appended to the global blockchain by miners. Mining encompasses validating transactions, grouping them, and forming them into a block format. Miners perform this task to gain rewards given by the blockchain. Choosing a miner to add the next block is done through a consensus protocol, which is different from one blockchain platform to another. Transactions performed through the blockchain are anonymous, permanent, and non-repudiable. This enables the blockchain to be a very suitable platform for various applications and use cases. Blockchain has been popular in recent years and has been used in various industries such as finance, food industry, and healthcare [18]. Blockchain can be used in the supply chain field for various activities, including the guarantee of product ownership to reduce the exchange of counterfeit products [19]. It also has the potential to enhance the information flow between supply chain partners [20]. Thereby, promoting data visibility and product traceability as the sale and inventory levels of the buyer are made transparent to all stakeholders in a private, permissioned network [21]. Consequently, making this information available to vendors on time allows inventory to accurately reflect product demand by monitoring the stock level frequently. This allows buyers to make smaller and frequent orders leading to a reduced number of stockouts and inventory carrying costs resulting in significant cost savings. Besides, vendors can improve their inventory planning as data is easily accessible; hence replenishment orders are made within a short period. Blockchain allows better communication among stakeholders with VMI arrangement as the system can be updated by vendors and buyers in real-time. Such bi-directional information exchange increases trust among these stakeholders and ensure consistency of data in the network. Furthermore, creating a blockchain network would eliminate the need for intermediaries, thereby allowing organizations to trade nationally or globally without any interferences leading

to the simplification of documents and communication flow between vendors and buyers.

Blockchain enables vendors to access data shared by retailers in real-time as it permits them to identify exactly when their customers have reached their reorder level. This allows them to process orders faster than using an intermediary such as a service provider. It minimizes the delay between recognizing the need for inventory and placing a new order. Therefore, using a blockchain solution integrated with decentralized applications (DApps) is capable of building a VMI system that meets its customer demand dynamically and, in response, creates an agile and responsive supply chain. It would also be able to offer real-time reporting in a transparent and timely manner, unlike using traditional methods where vendors are merely informed of the stock needed without knowing the actual inventory. Thus, the future industry of VMI supply chain operations would drastically transform with the integration of blockchain-based solutions. Some of the areas that would be enhanced by blockchain include improved order accuracy, increased sales and profitability, reduced inventory turnover, and the establishment of stronger customer relationships.

III. RELATED WORK

In this section, we discuss the relevant literature on the application of blockchain technology in the supply chain industry. Blockchain enhances supply chain management in various applications. For example, [22] proposes a blockchain-based scheme for securely sharing information in the pharmaceutical supply chain system with a consensus mechanism and smart contracts. Another example is presented in [23], which discusses an intelligent transportation system with a consensus mechanism to validate transactions and an authentication protocol for validating the user's vehicle. However, we would like to highlight that very few papers have been published in the VMI field. For instance, [24] discuss the critical success factors needed for implementing a blockchain solution for VMI. They present a use-case scenario that operates on a functional smart contract in which they consider a supply chain relationship with a one-to-one relationship between vendor and retailer. Thus, they propose an Ethereum based blockchain solution using a decentralized storage system, InterPlanetary File System (IPFS), alongside to store data. However, the proposed architecture in the paper does not discuss how privacy is achieved neither does it discuss the results of testing the smart contract. As an extension, [25] proposes a blockchain framework involving multiple vendors and buyers interacting via four smart contracts. Hence, this paper discusses several limitations addressed in [24], such as addressing the privacy among stakeholders by enforcing strict permissions in the contract.

Lastly, [26] provides a proof-of-concept that evaluates the practicality and usefulness of implementing a VMI based blockchain solution. Their approach is limited to a functional sequence diagram that covers the transactional information flow between VMI stakeholders using smart contracts. Nevertheless, the existing literature lacks a blockchain-based solution that attempts to implement an end-to-end, fully functioning VMI solution. Although these papers have implemented smart contracts using the Ethereum platform, none have discussed how the pricing agreements between different stakeholders are agreed upon within the contract. Therefore, this paper aims to provide a system architecture that captures the holistic view of blockchain-based VMI operations. The paper also discusses the cost and security analysis incurred with the implementation of such a system.

IV. PROPOSED BLOCKCHAIN-BASED SOLUTION

We propose a VMI blockchain solution that connects vendor, distributor, and retailers within the same decentralized Ethereum network, as illustrated in Fig. 1. These stakeholders interact with one another using both on-chain and off-chain transactions, as seen in Fig. 1. On-chain transactions are valid blockchain transactions that occur in the Ethereum platform, which demand an overall update of all the distributed ledgers in the network. Interactions within the blockchain network are made easy through the use of Ethereum smart contracts and decentralized storage systems. However, off-chain transaction agreements occur outside the blockchain, such as negotiations related to reorder point and transportation costs, pick up and distribution and distribution points, etc.

Vendor. The vendor manages inventory by placing necessary replenishment orders when needed by retailers. The role of the vendor in the proposed system is not limited to initiating the VMI contract and deploying it on the network, but it extends to registering trusted distributors and retailers to strengthen its customer relationship. Moreover, the vendor would upload the transportation cost that would correspond to delivering the new order schedule to its assigned retail stores. In the case where retailers or distributors do not agree with the price stated by the vendor, the price negotiations take place offline until an agreement is reached between both trading parties. The newly agreed price is then uploaded by the vendor, and accordingly, the new order is prepared for transportation. As a result, when retailers receive their orders, the vendor is responsible for paying the distributor.

Distributor. The registered distributor agrees to the delivery cost set in the contract by agreeing to pay a fixed deposit in return. The distributor is responsible for updating the status of the order and alerts the vendor when the orders are distributed to retail stores.

Retailer. The registered retailer uploads the sales report for a specific period and accordingly agrees to the new order price set by the vendor by agreeing to pay a partial payment in advance. Then the retailer completes the remaining payment upon receiving the order at the date and time scheduled.

Ethereum Smart Contract. The smart contract used in this framework was coded using Solidity 0.4.25 using an open-source tool, Remix IDE. This supports deploying, debugging, and testing contracts. The contract contains essential variables such as the stakeholder Ethereum addresses (EA), reorder quantity, and cost as well as the delivery cost.

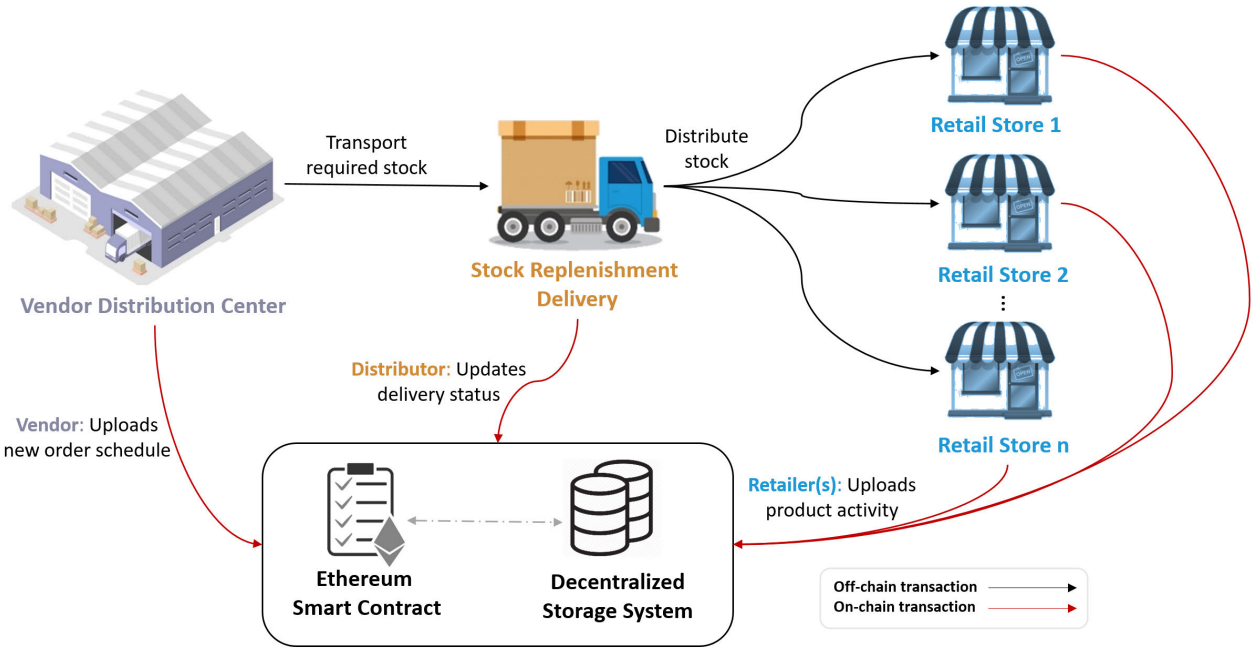


FIGURE 1. System overview of the VMI process using smart contract and decentralized storage system.

It also utilizes important elements such as modifiers and events to carry out key operations such as restrict stakeholders from updating sensitive information and alert stakeholders in cases where someone attempts to violate the rules that have been agreed upon. Modifiers are smart contract language constructs used to ensure only authorized blockchain actors or participants can execute, modify, or access certain attributes or functions within the smart contract. These functions are modified by conditional logic to check whether the client that triggered the function satisfies the requirements to access this functionality. The code in modifiers is executed before the code in the function to which it is attached. If the condition in the modifier is not met, the state of the smart contract does not change. Modifiers can be utilized to allow a specific group of users to view or modify the data in a smart contract. Moreover, real-time tracking of a new order is made possible by permitting stakeholders to access the contract.

Decentralized Storage System. The decentralized storage system, such as FileCoin [27] Swarm [28] or IPFS [29], can be used alongside blockchain to store data in a decentralized network. Files stored in such a system are difficult to tamper with as they are given unique cryptographic hashes. The proposed system utilizes a decentralized storage space to store documents such as the retailer sales activity report, delivery invoices, etc. To ensure privacy to content and files stored on IPFS, the solution in [30] can be employed. In [30], files stored on IPFS can be encrypted and the access to the encrypted files can be done by a single or multiple blockchain VMI actors by employing symmetric and asymmetric key encryption mechanisms and in a completely decentralized manner using re-encryption proxies or oracles. Readers can refer to [30] for further details on this. Therefore, the privacy as well as integrity of data stored within these decentralized entities is preserved.

V. IMPLEMENTATION

In this section, we present and discuss algorithms developed for implementing the VMI solution shown in Fig. 1. These algorithms capture the working principles of the proposed solution, and they form the foundation for developing the smart contract.

Algorithm 1 captures the initial steps taken by the stakeholders. Initially, the vendor registers its trusted retailers and distributors in the contract to grant permission to interact with the contract in its later stages. This registration is carried out by the vendor alone. After successfully registering the stakeholders, the retailers are allowed to upload the status of their periodic sales by uploading the sales report file hash and reporting the amount of quantity sold along with the current level of inventory.

Algorithm 2 illustrates that the vendor prepares a replenishment order after the reported sales data is analyzed. This process of making a new order schedule can be done periodically, such as daily, weekly, biweekly, or monthly depending on the product demand. So, the vendor uploads the reorder quantity along with its corresponding price for each retailer. Once the retailer agrees to the price made by the vendor, then the retailer accepts the price and simultaneously makes an advance partial payment that covers 50% of the total price. This acts as a guarantee that the full payment would be made when the new order arrives at their retail store.

Algorithm 1: Registration and Periodic Sales Reporting

Input: *Retailer EA, Distributor EA, Product sales report file hash, quantity sold and inventory level* are the list of submitted information

```
if EA = Vendor's EA then
  Register retailers and distributor
else
  Do not accept transaction from unauthorized EA
end
if EA = Registered Retailer EA then
  Allow the inputs to get added as valid transactions
else
  Do not accept transaction from unauthorized EA
end
```

Algorithm 2: Vendor's Replenishment Order and Retailer's Advance Payment

Input: *Reorder quantity and price, advance payment* are the list of submitted information

```
if EA = Vendor's EA & Retailers reported sales report then
  Upload reorder quantity and price for each registered retailer
else
  Do not accept transaction from unauthorized EA
end
if EA = Registered Retailer EA then
  Accept reorder price
  Pay half the reorder price
else
  Do not accept transaction from unauthorized EA
end
```

Likewise, Algorithm 3 presents the interaction between the vendor and distributor regarding transportation and delivery cost. The vendor uploads the cost that would cover the delivery of goods along with a fixed deposit amount. The distributor would accordingly accord with the stated delivery cost by agreeing to pay the requested deposit. This deposit would act as a reassurance to ensure orders will be delivered on schedule. Only when the distributor agrees, then the vendor pays an advance 50% payment to the distributor, which acts as a guarantee that the remaining payment will be transferred after delivery.

Algorithm 4 shows that when all stakeholders have made the necessary advance payments, then the distributor delivers orders according to the delivery schedule. During this stage, the distributor uploads information with regards to the status of the products such as time-stamp and location so that stakeholders in this network are granted permission to trace their products. Once the distributor delivers the order, then he uploads the quantity delivered at each retail store

Algorithm 3: Vendor's Transportation Cost Request and Distributor's Deposit

Input: *Transportation cost and deposit* are the list of submitted information

```
if EA = Vendor's EA  & new order is scheduled
  Upload transportation cost and deposit
else
  Do not accept transaction from unauthorized EA
end
if EA = Distributor's EA then
  Accept transportation cost
  Pay deposit to vendor
else
  Do not accept transaction from unauthorized EA
end
if distributor accepts then
  Vendor pays half the transportation cost
end
```

along with their invoice hash file. Thus, a retailer confirms that their order was received successfully when the quantity delivered equals the reorder quantity stated by the vendor in Algorithm 2.

Algorithm 4: Distributor Transports and Distributes Orders

Input*: Location, timestamp, retailer's EA, invoice file hash, quantity distributed* are the list of submitted information

```
if EA = Distributor's EA & vendor paid half the transportation cost then
  Upload location and timestamp
else
  Do not accept transaction from unauthorized EA
end
if EA = Distributor's EA  then
  Upload invoice file & quantity distributed
end
if quantity distributed = quantity reordered then
  Order is received successfully
else
  Order is incomplete
end
```

Finally, Algorithm 5 demonstrates that the vendor and retailers make the necessary payments, which conclude the VMI process. The retailers pay the remaining payment of the reorder cost to the vendor when the order is received. Simultaneously, the vendor pays the remaining transportation cost and returns the deposit to the distributor. The process then repeats itself by continuously tracking the level of inventory at all registered retail stores while reporting its periodic sales.

Algorithm 5: Vendor and Retailers Complete Payment

Input: *Vendor and retailer's payment* are the list of submitted information

```
if EA = Vendor's EA & order is received then
    Pay remaining half of transportation cost
    Return deposit to distributor
else
    Do not accept transaction from unauthorized EA
end
if EA = Registered Retailer EA & order is received then
    Pay remaining half of reorder price to vendor
else
    Do not accept transaction from unauthorized EA
end
```

Table 1 explains the functions that were used to implement the algorithms discussed earlier. The developed smart contract can be found in the GitHub repository[1]. Furthermore, these functions are illustrated as a series of functions and events in a sequence diagram, as shown in Fig. 2, in which the interactions between different stakeholders are also captured. Each stakeholder in the network holds an EA that allows them to interact with the contract by calling on the functions discussed in Table 1.

Initially, the vendor deploys the contract and registers retailers and distributors by calling the function *RegisterStakeholder(),* so they are granted permission to interact with the contract. Then registered retailers upload their sales report in a decentralized storage system and call the function *ReportPeriodicSale()* to upload their current sales, inventory level, etc. The vendor uses this information to establish a new order schedule which is suitable for each retailer by using appropriate forecasting tools. This new order is then uploaded via *OrderNewQuantity()* function.

The retailers then accept the new order quantity along with its price by agreeing to pay the vendor in advance. This payment transfer is made by retailers via *PayVendorPartly()* function in which they cover half of the vendor stated price. Simultaneously, the vendor uploads the transportation cost for delivering the orders via *AddTransportationCost()*. When the distributor agrees to the stated price, then a deposit is transferred to the vendor via *PayDeposit()* while the vendor pays the distributor partly in advance via *PayDistributorPartly()*. When the new order is loaded and ready for transportation, this triggers an event that notifies all stakeholders that the order has been successfully sent from the vendor distribution centre. During the transportation process, the distributor uploads details regarding the status of the order, such as time-stamp and location via *TraceNewOrder()*. The stakeholders can accordingly know the real-time status of the order by calling the function *GetTraceInfo()*. When the orders are distributed successfully to each retail store, then the distributor updates the status of the order by calling the function *DistributeNewOrder()*. Finally, the retailers and vendors complete their payments by calling the functions *PayVendorFully()* and *PayDistributorFully(),* respectively.

[1]https://github.com/Solidity-Contracts/Vendor-Managed-Inventory-Solution.git

VI. TEST SCENARIOS

In this section, we highlight various test scenarios when the smart contract was compiled successfully. It was observed that the functions were executed sequentially as expected.

The registration and reporting stages were tested, where two retailers were registered for testing the smart contract functionality. These retailers accordingly upload their sales report where the vendor analyzes the submitted data. Then the vendor issues a new order schedule accordingly. Fig. 3 shows that an event is triggered when the vendor attempts to make a new order for any retailer that has not been registered earlier.

This notification is necessary to reassure the retailers that the vendor carries out inventory management operations sincerely as every step is made transparent in the process. This strengthens the VMI relationship between the vendor and retailers.

The vendor then uploads a new order schedule for each retailer according to the data submitted. Each retailer can then obtain their new order details by calling the function GetNewOrderDetail(), as shown in Fig. 4a. However, it should be noted that when a distributor attempts to call this function, then an error is triggered, as shown in Fig. 4b as only vendors and retailers are granted permission to access this information.

Hence, after retailers agree to the vendor stated price, they are required to pay advance payment to guarantee the delivery of their orders. This was tested by allowing retailers A and B to transfer the requested amount to the vendor account. It is shown in Fig. 5 that the vendor balance increases by the total amount transferred from each retailer upon calling the *getBalance()* function.

The vendor then accordingly uploads the transportation details after obtaining the confirmation from all retailers. The distributor can access this information by calling the GetTransportationDetail() function, as shown in Fig. 6.

When the distributor agrees to the stated details, then a deposit payment is paid to the vendor. Only then the advance partial payment is transferred to the distributor. This was tested by checking the distributor account in which it was observed that the distributor account in Fig. 7 increases by the exact amount agreed between them. Moreover, if the amount is less than the stated price, then the transfer is not successful.

When all stakeholders make the necessary payments, the distributor uploads information concerning the new order, such as location and time-stamp. The vendor and retailers can obtain real-time information regarding the status of their order by calling the function *GetTraceInfo()*, as presented in Fig. 8.

TABLE 1. Description of functions used in the smart contract.

Stage	Function	Input	Output	Permissions	Description
Registration	`RegisterStakeholder()`	Retailers and distributor addresses	-	Vendor	Registers stakeholders such as retailers and distributor in a private network
Report periodic sale	`ReportPeriodicSale()`	Product sales report hash file, quantity sold, inventory level	-	Retailer(s)	Retailer uploads the product activity report after a fixed period
Reorder stock	`OrderNewQuantity()`	Reorder quantity and price	-	Vendor	Vendor inserts the reorder quantity required by each retailer
	`GetNewOrderDetail()`	Retailer address	Reorder quantity and price	Vendor, Retailer(s)	Returns the reorder quantity and price for a particular retailer
Part Payment on new order schedule agreement	`PayVendorPartly()`	Price agreement (Boolean), transfer partial advance payment	-	Retailer(s)	Retailer agrees to the reorder cost demanded by the vendor and transfers only half the amount to the vendor
Transportation agreement	`AddTransportationCost()`	Transportation cost, quantity to transport to each store, duration, deposit payment	-	Vendor	Vendor inserts the transportation details for the distributor to agree upon
	`GetTranportationDetail()`	Distributor address	Transportation cost	Vendor, Distributor	Returns the transportation cost
	`PayDeposit()`	Price agreement (Boolean), transfer deposit payment	-	Distributor	Distributor agrees on transportation and deposit amount stated by the vendor
Part Payment on new order transportation agreement	`PayDistributorPartly()`	Transfer partial advance payment	-	Vendor	Vendor transfers only half the amount to the distributor
Delivery	`TraceNewOrder()`	Status of the product such as timestamp and location	-	Distributor	Notifies stakeholders when reorder stock is collected from the vendor distribution center
	`GetTraceInfo()`	-	Status of the product such as time-stamp and location	Vendor, Distributor, Retailer(s)	Returns the location and time-stamp of the current status of the stock shipment
Order distribution	`DistributeNewOrder()`	Retailer address, Invoice hash file, quantity delivered to each retailer	Alert triggered when quantity distributed is incomplete	Distributor	Notifies stakeholders when reorder stock is distributed to all the registered retail stores
Payment Settlement	`PayVendorFully()`	Retailers transfer remaining price to the vendor	-	Retailer(s)	Stakeholders transfer the remaining price demanded
	`PayDistributorFully()`	Vendor transfers remaining price to the distributor	-	Vendor	

The distributor delivers the order to the registered retail stores. The contract declares that the order has been received successfully by the retailer when the quantity delivered by the distributor equals the reorder quantity that was forecasted by the vendor. Fig. 9 shows that the code triggers an error when this condition is not satisfied. Testing this requirement was

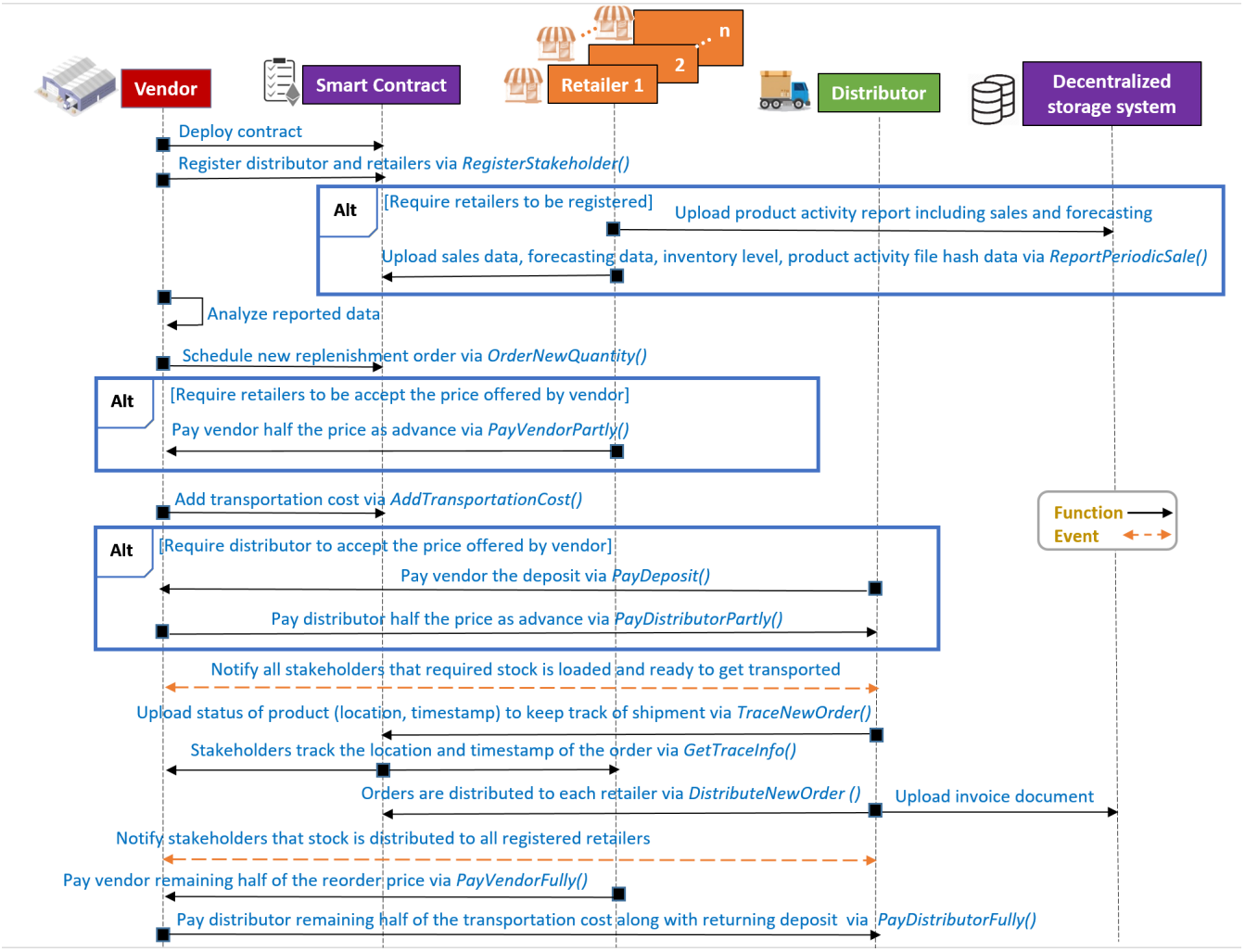


FIGURE 2. Sequence diagram showing the function calls and events in an automated VMI process.

FIGURE 3. Output showing the event triggered when unregistered retailers are used.

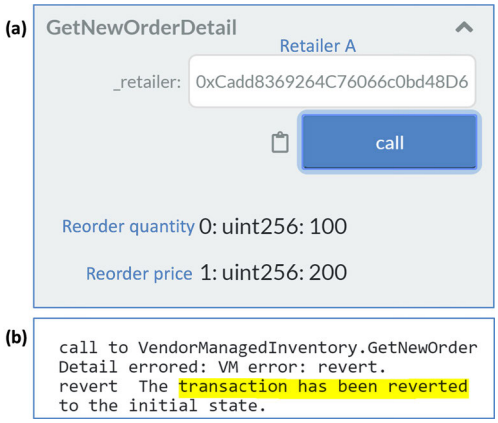


FIGURE 4. Output for testing the new order schedule function by using getter function permissioned to (a) vendors and retailers (b) excluding distributors.

necessary as stakeholders do not transfer the due payment in the case of incomplete deliveries.

As a result, retailers pay the vendor only when their orders are received, and accordingly, the vendor pays the distributor. The retailers transfer the balance payment that is the remaining half of the reorder price. This was tested by checking the vendor balance after the successful transfer was made. It was observed that the vendor balance increased as expected. Similarly, the vendor pays the outstanding amount and returns the deposit to the distributor. This transfer was tested in which

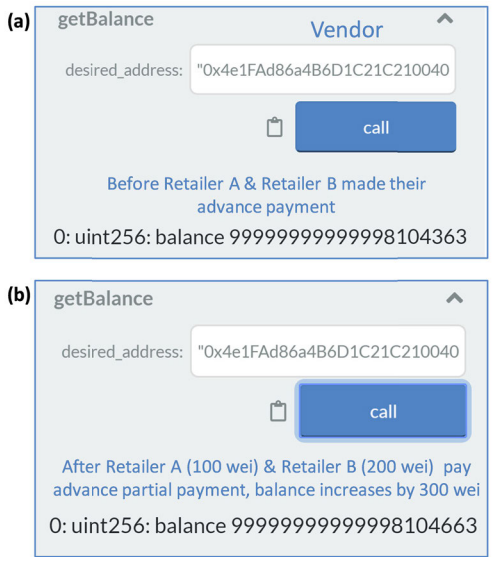


FIGURE 5. Checking the vendor balance after the successful advance payment transfer by retailer.

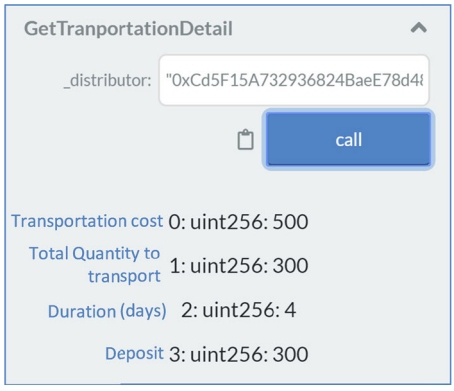


FIGURE 6. Testing the transportation details by viewing them using a getter function.

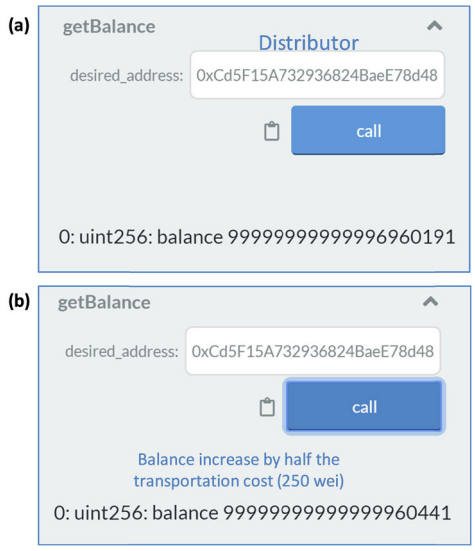


FIGURE 7. Checking the distributor account after the vendor made an advance payment.

FIGURE 8. Testing how the shipment details are obtained using *GetTraceInfo()* function.

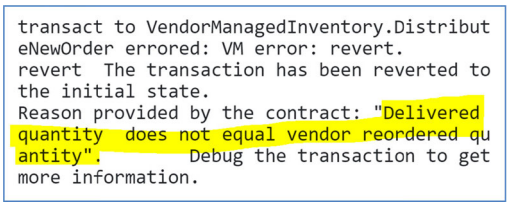


FIGURE 9. Output showing that alarm is triggered when the order delivered is not complete.

Fig. 10b shows that an error is triggered when the transferred amount is inaccurate, while Fig. 10c shows that the distributor balance increases from that shown in Fig. 10a when the vendor makes a transfer that equals the sum of the deposit and remaining transportation cost.

VII. DISCUSSION AND ANALYSIS

In our paper, we have implemented a blockchain-based solution for VMI in supply chain operations. The proposed solution captures the main operations in a VMI process. We discuss the cost and security analysis of the proposed system.

A. COST ANALYSIS

Every operation in an Ethereum network requires gas to get executed successfully. Hence, a certain amount of gas is required for every line of code that is written in Solidity, making it necessary to specify the adequate gas amount needed to execute smart contracts. There are mainly two costs incurred when executing Ethereum transactions on the blockchain. Execution cost takes into account all costs related to the internal storage and changes in the contract, while transaction cost includes the execution cost and cost of sending data such as transaction input costs and contract deployment [31].

The Ethereum gas is a unit used to measure the computational effort needed to execute transactions. Hence, this gas amount is specified by considering both the gas limit and price. The former is the total gallons of gas that is put in the smart contract gas tank while the latter is related to the gas consumed in a contract. Thus, the speed of executing transactions is directly affected by the gas price [31]. As the gas price increases, the rate of adding transactions to the block increases. It must be noted that gas price increases during high network traffic since a lot of transactions are competing to get added to the next block by miners. Correspondingly,

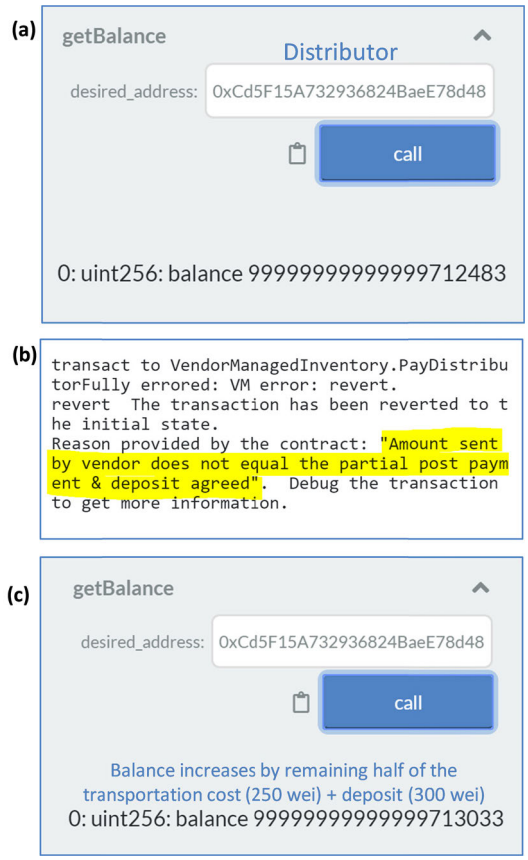


FIGURE 10. Checking the distributors account in (a) under two situations in (b) alarm triggered when payment is not complete while in (c) balance increased as complete payment is transferred.

```
INFO:root:contract vmi.sol:VendorManagedInventory:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                  16.2%
INFO:symExec:     Integer Underflow:                  False
INFO:symExec:     Integer Overflow:                   False
INFO:symExec:     Parity Multisig Bug 2:              False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:               False
INFO:symExec:     Re-Entrancy Vulnerability:          False
INFO:symExec:   ====== Analysis Completed ======
```

FIGURE 11. Smart contract vulnerabilities recorded by Oyente.

the miners are given a transaction fee; thus, these transactions are prioritized based on the order of gas price. As a result, the users willing to pay a higher gas price would have their transactions processed quickly.

Table 2 shows the gas costs of transactions and their corresponding prices in US Dollars at an Ether exchange rate of 1 ETH = 210.56 USD, for deploying the smart contract along with the major functions related to the vendor while, Table 3 and Table 4 correspond to the retailer and distributor respectively. The tables show the four different gas prices calculated to illustrate the change in transaction fees. These gas prices were obtained on May 8th, 2020, according to the ETH Gas Station [32]. Gas price equal to 19.6 Gwei corresponds to prompt processing; 165 Gwei would take several minutes, while a gas price of 13.2 Gwei would process at an average transaction time, and 10 Gwei would correspond to slower transaction time. Furthermore, we notice that the cost incurred by the vendor does not exceed 7 USD, while the retailer and distributor do not exceed 1 USD. This shows that implementing the presented solution encourages cost-savings and would result in increased profits to supply chain stakeholders.

B. SECURITY AND PRIVACY ANALYSIS

In this section, we discuss key properties of the proposed blockchain VMI solution in addressing major security and privacy concerns that include data integrity, authenticity, availability, accountability, and scalability. In addition, we will consider the robustness of the system against popular cyber-attacks such as DDoS attacks [1], [33].

1) SECURITY FEATURES

- **Data integrity and tamper-resistance.** Data integrity refers to the trustworthiness of data and ensuring unauthorized modifications or changes to the data. The source or origin of the data is vital to ensure that the information has not been jeopardized. Data integrity also refers to the resistance against the intentional tampering of data stored and recorded within the blockchain network itself.
  To maintain integrity, the data needs to be protected from tampering. In the blockchain, tamper-resistance means that transactions stored in the block cannot be jeopardized during or after the process of transaction validation by miners. There are several possible ways in which data could be tampered with. Miners may attempt to alter the data in the transactions received. For instance, they might try changing the transaction address of the payee to themselves or another entity. Such attempts cannot succeed as each transaction in the block is hashed by a secure hash function such as SHA-256 then signed using a secure digital signature algorithm. Hence, if a miner attempts to alter the information, then it will be automatically detected by other miners in the network. The second way that the data could be tampered with is by users. Users may attempt to manipulate and tamper with stored data. With a blockchain-based solution, such an attempt is impossible as the data is immutable using a hashing mechanism. The blocks are linked together using hash pointers, which are integral parts of the headers of the blocks.
- **Availability.** Refers to the probability that desired information or resources are accessible when needed at all times. For instance, inventory and demand data need to be made available for vendors so that they may calculate the forecasted data for the next period to avoid running into stockouts. Thus, the information generated and stored by retailers must be made accessible to vendors. This is necessary because systems with low availability are inefficient counterproductive. In certain situations,

TABLE 2. Transaction cost incurred by a vendor with different gas prices of Fastest (19.6 Gwei), Fast (16.5 Gwei), Average (13.2 Gwei) and Slow (10 Gwei) at an Exchange Rate of 1 Eth = 210.56 USD.

Function Name	Transaction Gas	Fastest Transaction Fee (USD)	Fast Transaction Fee (USD)	Average Transaction Fee (USD)	Slow Transaction Fee (USD)
`Deployment`	1533701	6.31271	5.31428	4.25143	3.22077
`RegisterStakeholder()`	58074	0.23904	0.20122	0.16099	0.12195
`OrderNewQuantity()`	43340	0.17839	0.15017	0.12014	0.09101
`AddTransportationCost()`	32187	0.13249	0.11153	0.08923	0.0676
`PayDistributorPartly()`	36029	0.1483	0.12484	0.09988	0.07566
`PayDistributorFully()`	35836	0.1475	0.12417	0.09933	0.07526
Total	**1739167**	**7.15843**	**6.02621**	**4.821**	**3.65225**

TABLE 3. Transaction cost incurred by a retailer with different gas prices of Fastest (19.6 Gwei), Fast (16.5 Gwei), Average (13.2 Gwei) and Slow (10 Gwei) at an Exchange Rate of 1 Eth = 210.56 USD.

Function Name	Transaction Gas	Fastest Transaction Fee (USD)	Fast Transaction Fee (USD)	Average Transaction Fee (USD)	Slow Transaction Fee (USD)
`ReportPeriodicSale()`	36001	0.14818	0.12474	0.09979	0.0756
`PayVendorPartly()`	47059	0.1937	0.16306	0.16306	0.09883
`PayVendorFully()`	45033	0.18535	0.15603	0.12482	0.09456
Total	**128093**	**0.52723**	**0.44383**	**0.38767**	**0.26899**

TABLE 4. Transaction cost incurred by a distributor with different gas prices of Fastest (19.6 Gwei), Fast (16.5 Gwei), Average (13.2 Gwei) and Slow (10 Gwei) at an Exchange Rate of 1 Eth = 210.56 USD.

Function Name	Transaction Gas	Fastest Transaction Fee (USD)	Fast Transaction Fee (USD)	Average Transaction Fee (USD)	Slow Transaction Fee (USD)
`PayDeposit()`	35309	0.14534	0.12235	0.09788	0.07415
`TraceNewOrder()`	33119	0.13631	0.11477	0.09181	0.06955
`DistributeNewOrder()`	28665	0.11798	0.09933	0.07946	0.06021
Total	**97093**	**0.39963**	**0.33645**	**0.26915**	**0.20391**

information has to be continuously updated, such as periodic sales and inventory, which demand quick and responsive interaction. One of the main threats that cause downtime are attempts to intentionally deny access to the services and data available. The blockchain mitigates this hazard by denying unauthorized users the ability to interact with smart contracts. The smart contract was designed to not allow unregistered users to add or modify information. This was enabled by leveraging the use of modifiers in creating smart contracts.

- **Authenticity.** An important element in information security is the confidence in that available data is trustworthy and genuine. In e-business and supply chain, it is vital to guarantee that the transactions, data, and communications are authentic. Moreover, it is essential to authenticate the data submitted by all parties who are involved in the VMI process. It is also necessary to verify that users are exactly who they claim to be and that unauthorized users are not able to steal their credentials. This is achievable through the use of blockchain, as data submitted by a particular user in the network is verified through their digital signature, which is directly linked to their private key. Also, no user can impersonate a stakeholder in the network as unauthorized users would not be granted permission since only registered participants are allowed to interact with the VMI contract.
- **Accountability.** This element in information security is crucial as any technology must be able to support nonrepudiation and uniquely trace the actions of each entity back to it. Nonrepudiation means that one party cannot deny the authenticity of their signature or the transactions originated by them. This forces stakeholders to fulfill their responsibilities as per the agreement. In our blockchain-based VMI solution, the vendor, retailers, and distributors cannot deny that they received a transaction or payment, nor can they deny a transaction that was sent by them. This is because, as specified by the blockchain, all entities sign and approve every transaction before it is added to the block. The contract alerts all stakeholders by triggering events whenever a stage from the VMI process has been completed successfully.

- **Resistance to DDoS Attacks.** A denial-of-service (DoS) attack is a type of cyber-attacks that targets a host machine, making it unavailable to its users due to deliberate overloading. A variation of DoS is DDoS, a ''distributed'' DoS attack, which attempts to create data traffic on the victim machine from multiple sources. These attackers take advantage of the security vulnerabilities in some computers that are connected to the network. However, blockchain networks are decentralized in nature, which allows transactions to get processed and executed despite few nodes going offline. Thus, for a DDoS to be successfully executed on a blockchain network, the attackers would need to possess machines with high computational power to cover infiltrate a significant number of members in the blockchain network. This makes it very difficult to achieve when attempting a large scale DDoS attack.

2) VULNERABILITY ANALYSIS

In addition to the general security and privacy risks that face blockchain smart contracts in general, we also analyzed the smart contract developed for our VMI supply chain solution for code vulnerabilities. This is done by analyzing the smart code using three security tools. SmartCheck code analyzer was used to discover potential risks in the code. In addition to all the syntax errors, run time errors, and warnings that Remix IDE notifies the developer about, SmartCheck provides further analysis of the smart code. SmartCheck detects errors such as costly loops, unusual gas consumption, and overflow and underflow. We found that SmartCheck tool showed that our final version of the smart contract code is fully free of any major or minor security and privacy vulnerabilities, as recorded by the SmartCheck log. Moreover, the code was also analyzed using Securify by chain security for code vulnerabilities. Securify only reported minor issues regarding the transferring of ethers, which are necessary to the algorithm implemented. To mitigate this, we validate the address that initiates the transaction and as well validating the data being sent within the transaction to ensure the amount of ether being sent is accurate. This was done via modifiers that restrict access of all methods to a specific group of pre-defined entities.

In addition, the smart contract code was analyzed using Oyente [34]. Oyente is a powerful tool that inspects Ethereum smart contract against its knowledge base of vulnerabilities. The tool produces a vulnerability report that lists crucial bugs and whether they exist in the provided smart contract code. These vulnerabilities include Integer overflow and underflow, Transaction Ordering Dependence (TOD), callstack depth attack, and re-entrancy attacks. Figure 11 below shows the analysis report generated by Oyente for the vendor managed inventory smart contract. As it can be seen, all vulnerabilities that the Oyente tool checks for were marked as ''False''. This proves that our smart contract is free of all of these bugs and thereby secure.

C. QUALITATIVE COMPARISON

The comparative analysis is given in Table 5. The table compares our proposed solution in this paper with existing solutions found in the literature. It can be observed that existing solutions presented in papers [24], [25] and [26] attempted to address VMI challenges such as data integrity, information accessibility, traceability, and transparency by providing smart contract solutions and suggesting consensus mechanisms. These papers provided a proof of concept and suggested how their solution could be implemented into the real world, with no or little technical details. For instance, [24], discusses the applicability of using Ethereum-based smart contract along with IPFS to store data while [25] utilizes multiple smart contracts to carry out VMI operations. Moreover, [26] suggested a framework that integrated blockchain and ERP systems to link the supplier to its customers. However, the implementation and testing of their proposed solution were not presented. Neither did they present any form of analysis, such as cost or security analysis, to indicate that their solution is potentially viable. On the contrary, our proposed solution implemented and tested the proposed framework using Remix IDE along with adequate analysis needed to show the feasibility of our solution.

TABLE 5. Comparison with Blockchain-Based VMI State-of-the-art Solutions.

Features	[24]	[25]	[26]	Our Work
Proof of Concept	✓	✓	✓	✓
Smart contract solution	✓	✓	✓	✓
Implementation and Testing	✗	✗	✗	✓
Cost Analysis	✗	✗	✗	✓
Security analysis	✗	✗	✗	✓
Decentralized Storage	✓	✓	✗	✓

D. OPEN RESEARCH CHALLENGES

Although blockchain has great potential in enhancing the VMI process, some challenges have to be considered and tackled in the future. In this section, we highlight some of these major challenges.

1) LIMITATIONS OF SMART CONTRACTS

Smart contracts offer tremendous opportunities for blockchain applications. Nevertheless, they suffer from major drawbacks that form obstacles facing developers as well as users of blockchain-based systems. There is a cost incurred for smart contract deployment, function calls in the smart contract, and each transaction initiated by the smart contract. These costs could result in a high cost of operation if the contract was not programmed to keep cost-effectiveness in mind. Programmers are required to develop low-complexity, cost-efficient code to minimize cost. In addition, buggy code can result in unnecessary loss of money, especially in cases such as infinite loops. Moreover, methods are restricted by a gas limit set by the user to avoid overspending of gas, which has to be taken into consideration when designing a smart

contract-based solution. Furthermore, the cost of execution in fiat currency is not constant amongst all users of the system. It depends on the gas cost set by the Ethereum client that is sending a transaction to the smart contract. Therefore, the cost of calling methods will vary between clients, and as a result, the time of execution will vary as well. Finally, the smart contracts are required to be written as to resist exploitation and attacks such as reentrancy attacks.

2) KEY MANAGEMENT

Managing keys in blockchain platforms and especially Ethereum, is different from other traditional systems. Users acquire a pair of keys, a public key and a private key, typically generated by a wallet application for the user. The client Ethereum address is derived from the public key. Ethereum clients need to maintain the secret private key in order to have access to the account balance. Managing the private key safely in Ethereum is crucial as there is no mechanism for retrieval of passwords from a central authority in a blockchain environment. Therefore, DApps such as cryptographic wallets often store the private key. This is either done through hexadecimal encoding, or more securely using a JSON file encrypted using a passcode, or more commonly using a mnemonic code backed up and linked to the user account.

3) SCALABILITY

The number of transactions increases as more users start utilizing blockchain platforms such as Bitcoin and Ethereum. A scalable system should be able to handle such increased load by allocating more resources. However, this is not applicable in blockchain networks as there is a limitation on the time interval and block size required to generate a new block. It should be noted that current blockchain platforms handle only a few transactions per second when compared to traditional centralized platforms [35]. As a result, it becomes problematic in a hectic industry, such as the supply chain, as it requires a lot of transactions to be processed in real-time. Moreover, miners in the network would rush to validate transactions with a higher fee to compensate for their computational effort as a reward since the block size is limited.

4) UNFORESEEN FUTURE EVENTS

The advancement of technologies today is not equipped with foreseeing risks such as employee strikes, natural disasters like hurricanes or earthquakes, etc. which would considerably lower the inventory levels required. Thus, the integration of blockchain technology with artificial intelligence, big data, machine learning is essential to enable the VMI process to operate vigorously. However, combining these various technologies to work together in a synchronized manner is still at an early stage.

5) RESISTANCE TOWARDS ADOPTING A DECENTRALIZED PLATFORM

Current stakeholders in the supply chain mostly deploy a centralized, industry-oriented platform for communication and information exchanges. Thus, it becomes very difficult to convince such stakeholders to adopt or integrate blockchain technology solutions into their daily operations. This is because some businesses in the supply chain industry view the adoption of blockchain applications to exchange transactions as an obstacle since it requires them to step out of their comfort zone and learn new skills or hire new talent. However, these businesses in the supply chain industry may need to adapt to these latest technologies if they wish to survive in a competitive industry.

VIII. CONCLUSION

In this paper, we have discussed the benefits of blockchain technology applied to VMI in supply chain operations. A blockchain-based VMI solution enables increased productivity, and improved supply chain performance as establishing such a network would enhance communication between different trading partners and promote transparency. It ensures a strong value chain partnership with an increase in trust among stakeholders. Moreover, it improves order accuracy as data integrity is maintained and potentially leads to increased profits to all participating supply chain stakeholders. VMI empowered with blockchain solution offers a cost-effective and efficient approach to inventory and replenishment strategies across various supply chains.

Our blockchain-based solution for implementing VMI operations framework focused on the use of Ethereum smart contracts to manage transactions in a decentralized manner without the need for intermediaries, such as VMI service providers. The system architecture, framework, algorithms, sequence diagram, and testing are suitable to work for any field in the realm of VMI applications. Our solution addresses the challenges in current VMI operations, such as single point of failure, data manipulation, and miscommunication. Hence, it ensures transparency, traceability, and data integrity enabled by the inherent features of blockchain technology. Furthermore, the smart-contract reassures stakeholders in the network that the operations are carried out without engaging in malicious activities, and it guarantees that only the stakeholders who are granted permission may communicate with the contract. The code is made public and available in GitHub. The functions related to the vendor, distributor, and retailers were tested to demonstrate the validity and operational aspects of the smart contract, as stated in the algorithms.

Furthermore, we discussed how the proposed blockchain solution ensures security features and issues related to data integrity, availability, authenticity, accountability, and cyber-attacks. In addition, we presented an extensive cost analysis to compute transaction costs that each stakeholder would incur when communicating with the contract. Our analysis has shown that our proposed solution is cost-efficient as the vendor would not have to pay more than 7 USD and that the distributor and retailers would pay an amount equal to almost less than a dollar. This demonstrates that the proposed solution is feasible as each stakeholder pays fractional amount when compared to using traditional methods in which

third party members must be paid for their services as well. For future work, we plan to design and build decentralized applications to fully automate the VMI process for all stakeholders. We also aim to expand the smart contract functions to address the multiple needs of the supply chain industry.

REFERENCES

[1] F. Xiong, R. Xiao, W. Ren, R. Zheng, and J. Jiang, "A key protection scheme based on secret sharing for blockchain-based construction supply chain system," *IEEE Access*, vol. 7, pp. 126773–126786, 2019.

[2] A. Angulo, H. Nachtmann, and M. A. Waller, "Supply chain information sharing in a vendor managed inventory partnership," *J. Bus. Logistics*, vol. 25, no. 1, pp. 101–115, 2004.

[3] Y. A. Hidayat, I. D. Anna, and A. Khrisnadewi, "The application of vendor managed inventory in the supply chain inventory model with probabilistic demand," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2011, pp. 252–256.

[4] Clear Spider. (May 26, 2015). *The Benefits of Vendor Managed Inventory (VMI)*. Accessed: Apr. 26, 2020. [Online]. Available: https://clearspider.net/wp-content/uploads/2019/12/Whitepaper-Vendor-Managed-Inventory.pdf

[5] K. Govindan, "Vendor-managed inventory: A review based on dimensions," *Int. J. Prod. Res.*, vol. 51, no. 13, pp. 3808–3835, Jul. 2013.

[6] P. Gonczol, P. Katsikouli, L. Herskind, and N. Dragoni, "Blockchain implementations and use cases for supply Chains–A survey," *IEEE Access*, vol. 8, pp. 11856–11871, 2020.

[7] P. Cui, J. Dixon, U. Guin, and D. Dimase, "A blockchain-based framework for supply chain provenance," *IEEE Access*, vol. 7, pp. 157113–157125, 2019.

[8] H. Juma, K. Shaalan, and I. Kamel, "A survey on using blockchain in trade supply chain solutions," *IEEE Access*, vol. 7, pp. 184115–184132, 2019.

[9] T. Guggenberger, A. Schweizer, and N. Urbach, "Improving interorganizational information sharing for vendor managed inventory: Toward a decentralized information hub using blockchain technology," *IEEE Trans. Eng. Manag.*, early access, Apr. 22, 2020, doi: 10.1109/TEM.2020.2978628.

[10] K. Radzuan, M. F. Omar, M. N. M. Nawi, M. K. I. A. Rahim, and M. Yaakob, "Vendor managed inventory practices: A case in manufacturing companies," *Int. J. Supply Chain Manage.*, vol. 7, no. 4, pp. 196–201, 2018.

[11] S. M. Disney and D. R. Towill, "Vendor-managed inventory(VMI) and bullwhip reduction in a two level supply chain," *Int. J. Oper. Prod. Manage.*, vol. 23, no. 6, pp. 625–651, 2003.

[12] M. Murray. The Balance Small Business. (Dec. 5, 2018). *Small Business Supply Chain: Vendor Managed Inventory (VMI)*. Accessed: Apr. 26, 2020. [Online]. Available: https://www.thebalancesmb.com/vendor-managed-inventory-vmi-2221270

[13] G. Perboli, S. Musso, and M. Rosano, "Blockchain in logistics and supply chain: A lean approach for designing real-world use cases," *IEEE Access*, vol. 6, pp. 62018–62028, 2018.

[14] R. M. Parizi, A. Singh, and A. Dehghantanha, "Smart contract programming languages on blockchains: An empirical evaluation of usability and security," in *Proc. Int. Conf. Blockchain*, 2018, pp. 75–91.

[15] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 653–659, Sep. 2017.

[16] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[17] P. K. Wan, L. Huang, and H. Holtskog, "Blockchain-enabled information sharing within a supply chain: A systematic literature review," *IEEE Access*, vol. 8, pp. 49645–49656, 2020.

[18] A. Saha, R. Amin, S. Kunal, S. Vollala, and S. K. Dwivedi, "Review on 'Blockchain technology based medical healthcare system with privacy issues,"' *Secur. Privacy*, vol. 2, no. 5, 2019, doi: 10.1002/spy2.83.

[19] K. Toyoda, P. T. Mathiopoulos, I. Sasase, and T. Ohtsuki, "A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain," *IEEE Access*, vol. 5, pp. 17465–17477, 2017.

[20] N. Hackius and M. Petersen, "Translating high hopes into tangible benefits: How incumbents in supply chain and logistics approach blockchain," *IEEE Access*, vol. 8, pp. 34993–35003, 2020.

[21] S. Wang, D. Li, Y. Zhang, and J. Chen, "Smart contract-based product traceability system in the supply chain scenario," *IEEE Access*, vol. 7, pp. 115122–115133, 2019.

[22] S. K. Dwivedi, R. Amin, and S. Vollala, "Blockchain based secured information sharing protocol in supply chain management system with key distribution mechanism," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102554.

[23] S. K. Dwivedi, R. Amin, S. Vollala, and R. Chaudhry, "Blockchain-based secured event-information sharing protocol in Internet of vehicles for smart cities," *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106719.

[24] T. Dasaklis and F. Casino, "Improving vendor-managed inventory strategy based on Internet of Things (IoT) applications and blockchain technology," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, Seoul, South Korea, May 2019, pp. 50–55.

[25] F. Casino, T. K. Dasaklis, and C. Patsakis, "Enhanced vendor-managed inventory through blockchain," in *Proc. 4th South-East Eur. Design Autom., Comput. Eng., Comput. Netw. Social Media Conf. (SEEDA-CECNSM)*, Piraeus, Greece, Sep. 2019, pp. 1–8.

[26] J. Kolb, D. J. Hornung, F. Kraft, and A. Winkelmann, "Industrial application of blockchain technology: Erasing the weaknesses of vendor managed inventory," in *Proc. ECIS Res. Papers*, 2018. [Online]. Available: https://aisel.aisnet.org/ecis2018_rp/189

[27] Protocol Labs. (Jul. 2017). *Filecoin: A Decentralized Storage Network*. Accessed: Dec. 18, 2018. [Online]. Available: https://filecoin.io/filecoin.pdf

[28] J. H. Hartman, I. Murdock, and T. Spalink, "The swarm scalable storage system," in *Proc. 19th IEEE Int. Conf. Distrib. Comput. Syst.*, Austin, TX, USA, Jun. 1999, pp. 74–81.

[29] J. Benet. (2014). *IPFS-Content Addressed, Versioned, P2P File System*. Accessed: Dec. 18, 2018. [Online]. Available: https://arxiv.org/pdf/1407.3561.pdf

[30] A. Battah, M. Madine, H. Alzaabi, I. Yaqoob, K. Salah, and R. Jayaraman. (Jul. 2020). *Blockchain-Based Multi-Party Authorization for Accessing IPFS Encrypted Data*. Accessed: Aug. 12, 2020. [Online]. Available: https://www.techrxiv.org/articles/preprint/Blockchain-based_Multi-Party_Authorization_for_Accessing_IPFS_Encrypted_Data/12788306

[31] Medium. (Apr. 5, 2018). *Optimizing Your Solidity Contract's Gas Usage*. Accessed: Mar. 9, 2020. [Online]. Available: https://medium.com/coinmonks/optimizing-your-solidity-contracts-gas-usage-9d65334db6c7

[32] *ETH Gas Station*. Accessed: Mar. 9, 2020. [Online]. Available: https://ethgasstation.info/calculatorTxV.php

[33] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surv.*, vol. 1, no. 1, 2019, doi: 10.1145/3316481.

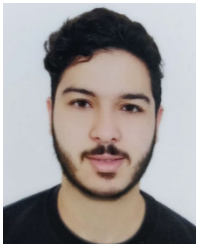
[34] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2016, pp. 254–269.

[35] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE 6th Int. Congr. Big Data*, Honolulu, HI, USA, Jun. 2017, pp. 557–564.

ILHAAM A. OMAR received the bachelor's degree in electrical and electronic engineering from Khalifa University, in 2017, where she is currently pursuing the master's degree in engineering systems and management. She and her team built a talking digital clock for the visually impaired along with her team in 2014, and switch mode power supply for her graduation project. She focused on the fabrication of microscale memristors in which she participated in the undergraduate research competition in 2016. Her research interests include blockchain technology and how it impacts the healthcare sector. She found this research area intriguing as it combines both a cutting-edge technology and healthcare both of which she finds captivating.

RAJA JAYARAMAN received the bachelor's and master's degrees in mathematics from India, the M.Sc. degree in industrial engineering from New Mexico State University, and the Ph.D. degree in industrial engineering from Texas Tech University. He is currently an Associate Professor with the Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, UAE. His expertise is in multi-criteria optimization techniques applied to supply chain and logistics, Healthcare, energy, environment, and sustainability. His research interests include applying technology, systems engineering, and process optimization techniques to analyze complex systems. His Postdoctorial Research was centered on technology adoption and implementation of innovative practices in the Healthcare supply chain logistics and service delivery. He has led several successful research projects and pilot implementations in the area of supply chain data standards adoption in the U.S. Healthcare system. His research has appeared in top rated journals, including *Annals of Operations Research*, *IISE Transactions*, *Computers & Industrial Engineering*, *Energy Policy*, *Applied Energy*, *Knowledge Based Systems*, IEEE Access, *Journal of Theoretical Biology*, *Engineering Management Journal*, and others.

KHALED SALAH (Senior Member, IEEE) received the B.S. degree in computer engineering with a minor in computer science from Iowa State University, USA, in 1990, and the M.S. degree in computer systems engineering and the Ph.D. degree in computer science from the Illinois Institute of Technology, USA, in 1994 and 2000, respectively. He joined Khalifa University, UAE, in August 2010, and is teaching graduate and undergraduate courses in the areas of cloud computing, computer and network security, computer networks, operating systems, and performance modeling and analysis. Prior to joining Khalifa University, he worked for ten years at the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia. He is currently a Full Professor with the Department of Electrical and Computer Engineering, Khalifa University. He has over 190 publications and three patents, has been giving a number of international keynote speeches, invited talks, tutorials, and research seminars on the subjects of blockchain, the IoT, fog and cloud computing, and cybersecurity. He is also a member of IEEE Blockchain Education Committee. He was a recipient of Khalifa University Outstanding Research Award from 2014 to 2015, KFUPM University Excellence in Research Award from 2008 to 2009, and KFUPM Best Research Project Award from 2009 to 2010. He was also a recipient of the departmental awards for Distinguished Research and Teaching in prior years. He also serves on the Editorial Board of many WOS-listed journals, including *IET Communications*, *IET Networks*, *Journal of Network and Computer Applications* (JNCA) Elsevier, *Security and Communication Networks* (SCN) Wiley, *International Journal of Nanomanufacturing* (IJNM) Wiley, *Journal of Universal Computer Science* (J.UCS), and *Arabian Journal for Science and Engineering* (AJSE). He is also the Track Chair of IEEE Globecom 2018 on Cloud Computing. He is also an Associate Editor of IEEE Blockchain Newsletter.



MAZIN DEBE received the B.Sc. degree in computer engineering and the M.Sc. degree in electrical and computer engineering from the Khalifa University of Science and Technology, Abu Dhabi, UAE. He is currently working as a Research Associate with the Center for Cyber-Physical Systems, Khalifa University of Science and Technology. He has published four research articles in highly ranked IEEE conferences and journals. His research interests include blockchain technology, the Internet of Things (IoT), fog computing, and supply chain applications.

MOHAMMED OMAR is currently a Full Professor and a Founding Chair of the Department of Engineering Systems and Management (currently renamed Industrial and Systems Engineering); prior to joining the Masdar-Institute/KUST, he has served as an Associate Professor and a Graduate Coordinator at the Clemson University, Clemson, SC, USA, and was a part of the Founding Faculty Cohort for the Clemson University research park in Greenville, SC, USA. He has over 100 publications in the area of product lifecycle management and knowledge based manufacturing and automated testing systems, in addition to authoring several books and book chapters; he has been granted four U.S. and international patents. He was named a Tennessee Valley Authority Fellow for two consecutive years during his Ph.D. studies, in addition to being a Toyota Manufacturing Fellow. His professional career includes a Postdoctoral Service at the Center for Robotics and Manufacturing Systems (CRMS), and a Visiting Scholar appointment at the Toyota Instrumentation and Engineering Division, Toyota Motor Company, Japan. His group graduated seven Ph.D. Dissertations and over 35 M.Sc. theses; four of his Ph.D. Students are also on academic ranks in U.S. universities. His work has been recognized by the U.S. Society of Manufacturing Engineers (SME) through its Richard L. Kegg Award, and was also awarded the SAE Foundation Award for Manufacturing Leadership. Additionally, the Clemson University, College of Engineering awarded him the Murray Stokely Award. He also led an NSF I/UCRC center and was part of a DoE GATE center of excellence in Sustainable Mobility Systems. His current laboratories at the Masdar City campus include capabilities in composite fabrication and manufacturing analytics. His current research group did support two Postdoctoral scholar's career planning to become an Assistant Professor at the Texas A&M (TAMU) in 2013, and the University of Sharjah in 2015. He also serves as an Editor-in-Chief for the *Journal of Material Science Research* (Part of the Canadian Research Center), and as an Associate Editor for the *Journal of Soft Computing* (a Springer Publication), handling the area of decision science and knowledge based systems; in addition to his membership on several editorial boards and conference organizations. Furthermore, he serves on the advisory board of the Strata PJSC (part of Mubadala Aerospace).