applied sciences

MDPI

Article

PetroBlock: A Blockchain-Based Payment Mechanism for Fueling Smart Vehicles

Faisal Jamil [1,†], Omar Cheikhrouhou [2,*], Harun Jamil [3,†], Anis Koubaa [4], Abdelouahid Derhab [5] and Mohamed Amine Ferrag [6]

[1] Department of Computer Engineering, Jeju National University, Jejusi 63243, Korea; faisal@jejunu.ac.kr
[2] College of CIT, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia
[3] Department of Electronic Engineering, Jeju National University, Jejusi 63243, Korea; harunjamil@hotmail.com
[4] Computer Science, Robotics and Internet of Things Lab of Prince Sultan University, Riyadh 12435, Saudi Arabia; akoubaa@psu.edu.sa
[5] Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 11451, Saudi Arabia; abderhab@ksu.edu.sa
[6] Department of Computer Science, Guelma University, BP 401, Guelma 24000, Algeria; ferrag.mohamedamine@univ-guelma.dz
* Correspondence:o.cheikhrouhou@tu.edu.sa
† These authors contributed equally to this work.

Abstract: Current developments in information technology and increased inclination towards smart cities have led to the initiation of a plethora of features by technology-oriented companies (i.e., car manufacturers) to improve users' privacy and comfort. The invention of smart vehicle technology paved the way for the excessive use of machine-to-machine technologies. Moreover, third-party sharing of financial services are also introduced that support machine-to-machine (M2M) communication. These monetary systems' prime focus is on improving reliability and security; however, they overlook aspects like behaviors and users' need. For instance, people often hand over their bank cards or share their credentials with their colleagues to withdraw money on their behalf. Such behaviors may originate issues about privacy and security that can have severe losses for the card owner. This paper presents a novel blockchain-based strategy for payment of fueling of smart cars without any human interaction while maintaining transparency, privacy, and trust. The proposed system is capable of data sharing among the users of the system while securing sensitive information. Moreover, we also provide a blockchain-based secure privacy-preserving strategy for payment of fueling among the fuel seller and buyer without human intervention. Furthermore, we have also analytically evaluated several experiments to determine the proposed blockchain platform's usability and efficiency. Lastly, we harness Hyperledger Caliper to assess the proposed system's performance in terms of transaction latency, transactions per second, and resource consumption.

Keywords: vehicle refueling; Hyperledger Fabric; distributed ledger; secure payment; blockchain

check for updates

Citation: Jamil, F.; Cheikhrouhou, O.; Jamil, H.; Koubaa, A.; Derhab, A; Ferrag, M.A. PetroBlock: A Blockchain-Based Payment Mechanism for Fueling Smart Vehicles. *Appl. Sci.* **2021**, *11*, 3055. https://doi.org/10.3390/app11073055

Academic Editor: Gianluca Lax, Roberto Tonelli, Marco Ortu and Andrea Pinna

Received: 16 December 2020
Accepted: 24 March 2021
Published: 29 March 2021

1. Introduction

The emergence of Internet of Things (IoT) technologies plays a vital role in various walks of human life, including business, agriculture, healthcare, transport, etc. According to Cisco and Ericsson, the quantity of IoT devices in daily usage will exceed over 100 billion by 2020 [1]. IoT devices, i.e., smart mirrors, smart cameras, smart cars, smartphones, etc., connect a user with a diverse range of services like traveling, manufacturing, transactions, etc., by harnessing the Internet. Most of these IoT devices depend upon machine-to-machine (M2M) communication, which involves communication without humans' intervention. The M2M communication-based systems are capable of automating the process transaction through online payment systems [2]. Although M2M communication transactions are broadly being harnessed across the globe, the context data in M2M communication have a severe concern regarding security and privacy. The contemporary

M2M-based applications face three significant challenges: (i) transparency, (ii) interoperability, and (iii) trust [3]. Transparency is one of the most significant issues faced by IoT and M2M-based applications. The current IoT and M2M devices are highly insecure in nature; therefore, communication among them must be secured by employing different encryption techniques. Moreover, data ownership is also non-persistent in these devices. Encrypting IoT and M2M infrastructure involves huge complexity. Interconnection among peripherals enhances productivity in industrial settings as well as residential area. However, when the quantity of devices increases and network connections expand due to increased population, the issue of interoperability arises. The third challenge involves the trust of users. Figure 1 illustrates an example of M2M communication wherein a smart vehicle uses limited-range wireless communication (e.g., Bluetooth and NFC) to make payment for refueling while the user of the smart car trusts vendor by providing credit card information. The smart car and gas station are equipped with IoT applications that hold the digital wallet of a user.

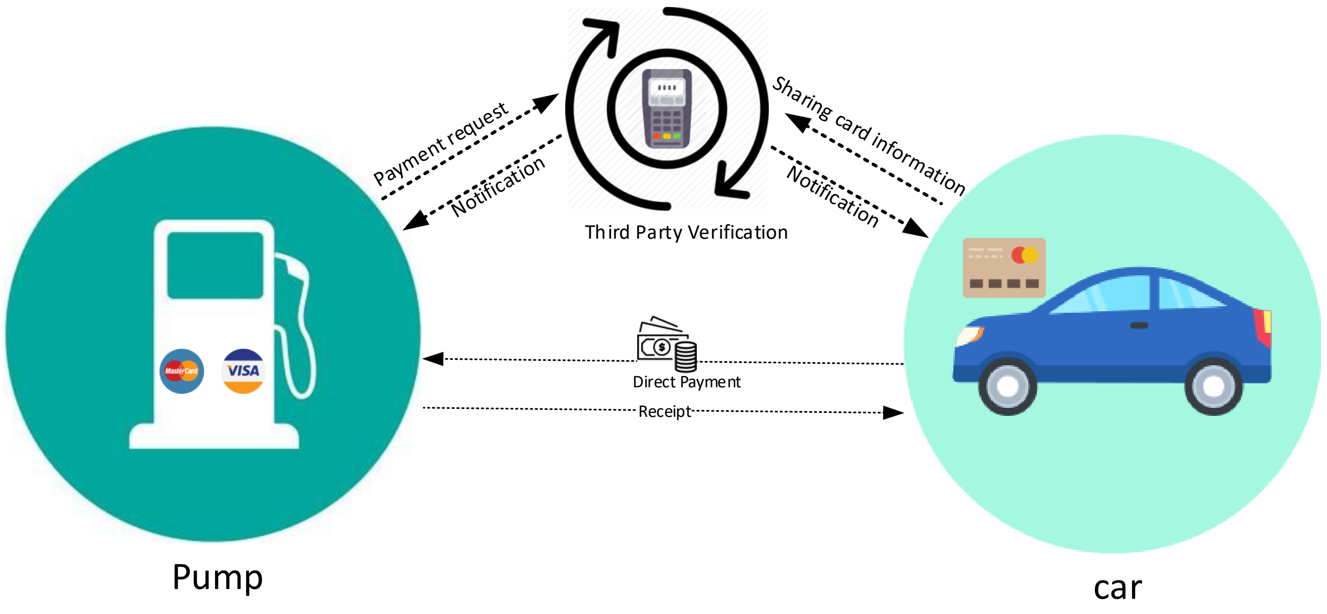


Figure 1. A typical smart pump IoT application which persist user information (e.g., credit card).

The M2M and IoT applications that exchange physical or digital assets of users should be reliable and provide risk-free interfaces during transactions. However, the current IoT applications do not hold a consistent structure for the security, access control, and privacy-related models [4]. This raises serious concerns related to the safety and reliability of transaction processing. Recently, there has been a boom of the latest Information technology Blockchain [5]. The concept of blockchain is deemed as a decentralization ledger shared among all connected networks. Since it is not feasible to mathematically update the ledger, cryptographic-based methods are employed for the said purpose. The data structure of a blockchain comprises immutable, ordered, and timestamped data blocks. Each block in a design holds a connection with the preceding block, which forms a strict order. In a blockchain, the first block is known as the genesis block used to create the blockchain blocks, also known as block zero. These blocks consist of multiple ordered transactions. Such data structure allows provenance wherein a transaction has a single origin. All users involved in the blockchain network authenticate and approve the transaction. It is a decentralized technology wherein all records are maintained at different locations. This decentralized behavior enables blockchain to govern under multiple entities, indicating that governments or other authorities do not have any access to that. Therefore, the data in the blockchain system could not be altered.

Blockchain has significant properties that directly address the challenges of Intelligent Transportation System (ITS). Therefore, the companies dealing with financial services emphasize on using blockchain because it ensures highly secured transactions of assets [6]. Furthermore, it provides transparency like public-audibility, an append-only ledger of transactions involved in the chain. The recent emerging of blockchain technology supports the shifting of the traditional payment system at a gasoline pump into a secure automated system [7]. Smart contracts and Hyperledger Fabric can be employed for allowing a software platform to run without a trusted third party. This behavior leads to opening up opportunities for harnessing product-centric enterprise systems. In this study, a secure payment method is implemented to refuel smart vehicles at smart pumps. The designed model leverages the blockchain platform (i.e., Hyperledger Fabric [8,9]) to secure the transfer of payment and share information among different gasoline pumps. Firstly, we propose a novel blockchain payment method to fuel smart vehicles in which a user can share sensitive information like credit card credentials or others in a secured, authentic chain of connected networks of the associated franchises of gasoline pumps. This blockchain model is extended towards a web-driven system by designing web front-end technologies like HTML5 and JavaScript for enhancing resource management in a network. Furthermore, we also configure composer-rest-server to expose REST API to visualize services related to the designed system. Afterward, smart contracts' functionality is embedded to enable data uniformity of card and bank information and other users' assets and other participants involved in the network. Afterward, an Access Control List (ACL) is defined, which provides access to only valid and authentic users. To enhance the model's efficiency and performance, we employ DBCouch [10] to manage a large number of records and evade form data redundancy problems within the blockchain File System. The performance of the proposed system is evaluated using Hyperledger Caliper [11,12], a benchmarking tool for blockchain-based system [11,13].

More precisely, the contributions of this paper are as following:

- We present PetroBlock: a secure payment mechanism for refueling smart vehicles without human interaction, providing transparency, trust, and privacy.
- The proposed model can share data among the system's users while securing sensitive information.
- We evaluate the designed system using Hyperledger Caliper in terms of transaction latency (average, percentile, minimum and maximum), transaction throughput, resource utilization (e.g., CPU, memory, I.O.).

The rest of the paper is structured as follows: Section 2 explains the state-of-the-art related work on payment mechanisms for fueling smart vehicles. Section 3 presents the designed methodology, smart contract modeling, implementation, and storage design. Section 4 gives the details of the system results, and Section 5 concludes the paper.

2. Related Work

This section encompasses preliminary studies related to the secure payment mechanism and refueling of the smart vehicle by harnessing smart contract and blockchain technologies. Many studies have been presented to interact blockchain with machines [14–23].

In [24], authors proposed an automated scheme to charge the electric vehicle by taking dynamic tariff decisions in a privacy-preserved manner by exploiting distance and price information. The presented system is based on a blockchain platform where electric vehicles send a signal with a charging station's demand, similar to an auction bid. The owner of the electric vehicle opts for a specific charging station using the received supply-side offers. The proposed system is comprised of three parts: (i) the provided system finds the optimal charging station based on public bidding as a query response; (ii) the geographical position of the person is not revealed during protocol execution; (iii) the Distributed Ledger Technology (DLT) of blockchain for immutable storing of data related to the bidding of an electric vehicle.

The authors in [25] developed a model for an automatic fueling system for automobiles. The presented model comprises a fuel dispenser, which is movable, including a nozzle connected with the vehicle's fuel inlet. The automatic fuel dispenses movement through a programmable unit helps it attach nozzle with a fuel inlet. The sensor implant in a car indicates the fuel dispenser's direction to the fuel inlet through signals.

Smart contract to enable communication among machines is introduced in [3,26]. The authors presented the AGasP, an IoT application for machine-to-machine gasoline payment by using a smart contract. The proposed system uses a smart contract for the payment transaction. The developed method is based on Ethereum, and allows the user to audit all interactions between the design and the blockchain platform as long as the network of blockchain exists. The third parties that are part of the payment systems are removed via AGasP, and transaction charges are waived off up to 79% in comparison to the conventional method.

In [27], authors introduced a blockchain-based autonomous system that selects the most suitable charging stations for electric vehicle. The design system provides automated secure machine-to-machine payment, auction, and bidding. The system is developed on a Raspberry Pi and raspbian operating system. The proposed method is divided into three actors: a charging station, a vehicle, and a driver. The driver's responsibility is to launch the selection procedure and confirm the bid suggested by the car. The car has two entities, the infotainment system provided by the electric vehicle and the blockchain-based smart contract contract [27]. A mobile application based on Ethereum was developed to provide the appropriate charging bid with automated payment.

In [28], authors presented the case study of an automatic payment mechanism based on the smart contract. When entering his/her car, the user's smartphone automatically synchronizes with the AutoPay service. The autonomous payment service provides trust and security through smart contracts implemented as an adapter on its blockchain interface. The AutoPay service detects current fuel status and finds the route passed by a petrol station if fuel status is low. AutoPay automatically pays the fuel bill after refueling through smart contract features. AutoPay also pays for user's car parking automatically through a smart contract when he/she goes to work.

In [29], ChargeItUp case study is presented to analyze the refueling scenario for an autonomous vehicle. The proposed system is an Ethereum-based system that sends transactions through smart contracts, mainly machine-to-machine (M2M). The proposed electric car charging company architecture consists of three parts: the client (the car user), the server (the charging station), and the blockchain smart contract. Firstly, the user went to the charging station to charge his car. The charging station and user agree on the price per unit of charging and other aspects to open the channel. The user sends an open channel notification to a charging station verified by the station, and they start the charging for the time specified by the user. In the end, the car user pays for the charging by sending the final transaction to the station.

In [30], a novel decentralized approach of electric vehicle charging and discharging is presented using a blockchain-enabled smart grid. Adaptive Blockchain-based Electric Vehicle Participation (AdBEV) scheme is proposed to reduce the power fluctuation level. Iceberg order algorithm is used to implement the beast order mechanism for matching the system designed for smart grid electricity charging and discharging. The ethereum-based platform can achieve a low consumption cost as compared to other ethereum-based E.V. systems.

Kim et al. in [31] proposed a blockchain-based mobile charger billing system. The design system is used to send online transactions securely in a peer-to-peer distributed network. The mobile charger may be grouped by utilizing the groupID. Moreover, using blockchain technology reduces the size of the data block. As explained earlier, the blockchain platform is permissionless or not open-source; therefore, a general user cannot modify it.

Nevertheless, none of these models have addressed the payment mechanism for refueling employing Hyperledger Fabric platforms. There is no functional payment mechanism model for refueling smart cars using Hyperledger technology to the best of our knowledge.

3. PetroBlock Framework

3.1. Scenario of Adversary Model

In our threat model, global external attacker A is considered in opposite to blockchain-based payment platform that performs the following categories of attacks, such as replay attack, impersonation attack, key attack, Sybil attack, false data injection attack, tampering attack, modification attack, man-in-the-middle attack, and hiding blocks attack [32].

- Key attack: This attack exploits the private key leakage vulnerability over a smart car network.
- Replay attack: An adversary tries to intercepts and then replay a valid data transmission over a smart car network. This attack can be used to deceive financial institutions by duplicating transactions, allowing the adversary to withdraw money directly from their victims' accounts.
- Impersonation attack: Under this attack, an adversary tries to masquerade as a legal user for doing some unauthorized transactions in the blockchain.
- Sybil attack: By frequently interacting in the smart car network, an adversary can form various fake identities using some cars' reputation.
- False data injection attack: For making smart cars take wrong control decisions, an adversary injects false data wherein data integrity associated with the control system may get compromised.
- Tampering attack: Under this attack, an adversary uses the leakage of the private key or 51% vulnerability to tamper with the amounts, transactions, and other information in the ledger.
- Modification attack: This type of attack consists of modifying the blockchain operation, such as the consensus process.
- Hiding blocks attack: In this attack, only those transactions are displayed, which he holds a positive image and disguises those that have a negative reputation.
- Man-in-the-middle attack: Under this attack, an adversary exploits 51% vulnerability, or the private key leakage and identities of two parties are spoofed in the network of a smart car then modify the transmitted data.

3.2. Scenario of Proposed Blockchain Platform

Figure 2 represents the conceptual model of the proposed privacy-preserving blockchain-based payment strategy for fueling smart vehicles in a smart pump. The design platform will manage and update the entire payment mechanism of a smart car's fueling, storing data related to car users, smart pumps, and a smart car. The PetroBlock data lake serves as an individual property, which is also known as a stored-off blockchain. It is an essential tool employed to conduct assessment analysis like visualization, analytics, and data reporting related to a smart pump and smart car. Furthermore, the pump manager can also access a car user's data by taking a car user's permission within the network. In these permissions, access control policies are defined in the smart contract so that integrity and privacy related to user's data can be maintained. The proposed system comprises nodes that manage the consensus protocol's execution to maintain the distributed ledger's consistency. Initially, the smart car user initiates the request of refueling to the smart pump. In response, the smart pump validates the request. On successful validation, the refueling process starts. Once the refueling is completed, the smart car user receives the payment notification. The designed system automatically pays for the refueling once the user grants permission. The successful payment notification is sent to both smart pumps and smart car users.

Mutual authentication is executed when the car user sends a refueling request to ensure end-to-end secure communication between the car user and the smart pump. Both parties employ the elliptic curve encryption to generate their temporary private and public keys (i.e., keys that are valid for one session), as in [33]. Then, each party executes the Diffie–Hellman key exchange to compute the shared session key, and hence a session key is established. The session is terminated when they receive a successful payment notification.

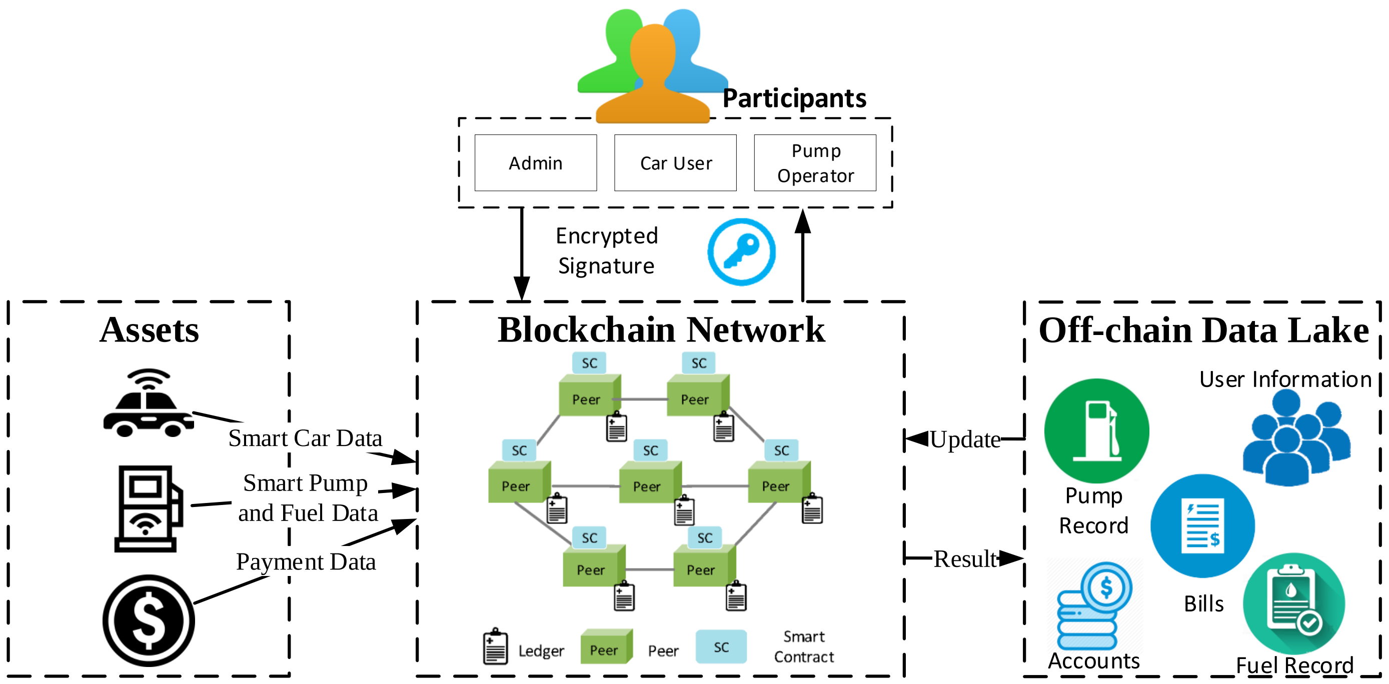


Figure 2. Conceptual model of the privacy-preserving payment service with smart contract.

Similarly, in Figure 3, the submitting clients reside in the blockchain network to invoke a smart contract and receive a notification on adding a new transaction in the blockchain ledger. The participating networks belong to the business network having permission to submit transactions and maintains assets. The assets can be used in the smart contract to acquired data from sensors. The sensing data are then utilized in various methods for different tasks like paying refueling charges and car maintenance. The newly generated instance of participants and assets are stored in the respective registries. This study considers mapping the identities of participants by employing an enrollment certificate stored by the identity registry. New mappings between identity registries are created when the admin bounds an identity to a participant.

3.3. *PetroBlock System Architecture*

The distributed ledger technology of blockchain enables the storage of information while mainlining data privacy and integrity. The system architecture of the proposed PetroBlock is illustrated in Figure 4. The proposed application contains different services like data sharing, smart pump management, secure payment transaction, smart car management, and user identity management. These services mentioned above use smart contracts and distributed ledger as an adapter to the application. The car user and pump manager are responsible for sending the transnational proposal through application to use the proposed blockchain platform's services like payment of refueling, smart pump management, data sharing, smart car management, and user profile management.

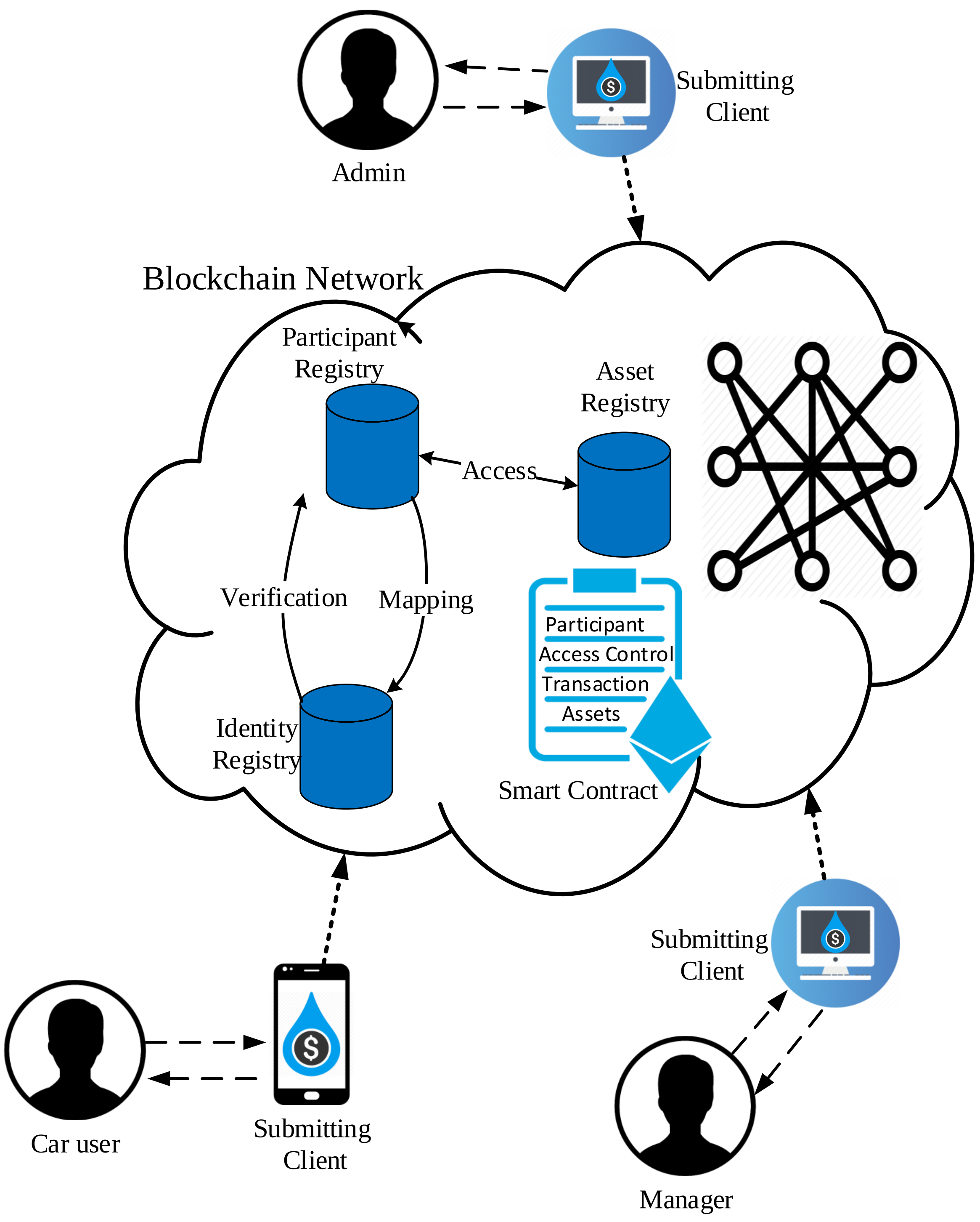


Figure 3. Proposed refueling service interaction scenario.

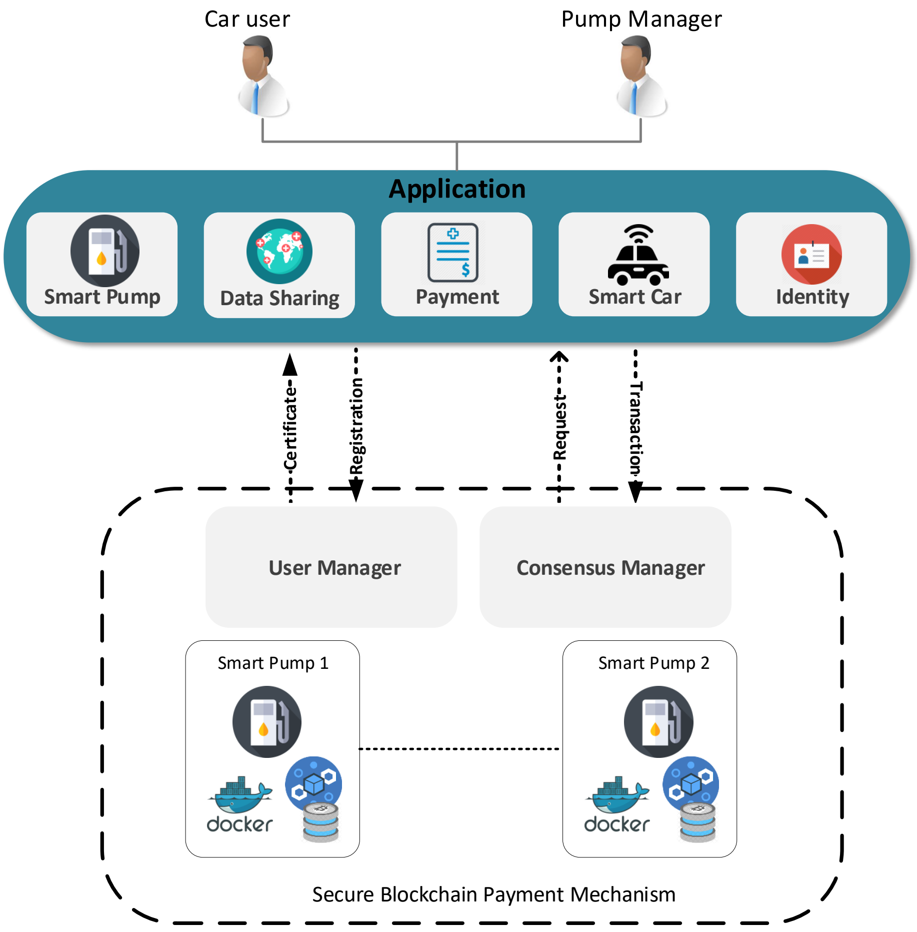


Figure 4. System architecture of proposed blockchain platform.

The proposed system provides a secure way to transfer payments and offers the complete Create, Read, Update, and Delete (CRUD) operation on the involved assets and participants that modified the connected nodes' data. The user manager is responsible for enrolling the valid participant and providing with enrollment certificate. The consensus manager enables the different participants to connect through the proposed blockchain application. The entity involved in the proposed system is comprised of a distributed ledger and nodes. The node's responsibility is to endorse transaction proposals through smart contracts and store data. The data consistency is managed by a consensus algorithm, which also ensures the integrity of data. The DLT [34] can store the immutable transaction and keep the consistency of every ledger copy by using a different cryptographic algorithm.

Sustaining the copy of a blockchain and transaction processing is the responsibility of nodes in a network. An example of such a blockchain node is shown in Figure 5. The characteristics of the node include blocks, policies, state databases, and smart contracts. The ledger state at a given time is maintained in a state database. The association among block and state databases is represented via sample white box illustration. The policies decide which transaction is recorded to the ledger in terms of node agreement. Each blockchain block contains the data related to payment, user profile, and other entities involved in the blockchain platform. Each block is assigned a sequence number, aiming to identify a block's order in the blockchain and resist hiding block attack, as in [35].

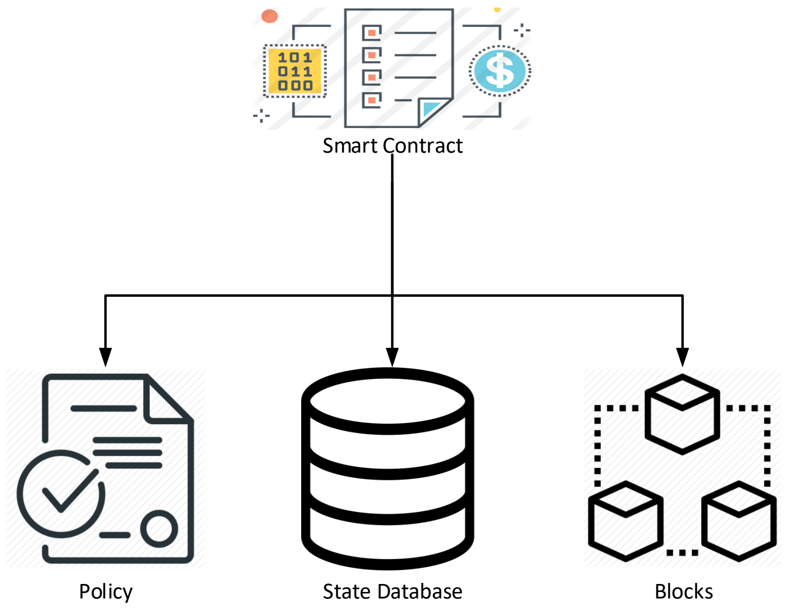


Figure 5. Representation of the node in the proposed blockchain platform.

A smart contract adds functionality to a blockchain because it is a computer program that performs transactions and views the blockchain history and blocks. A distributed database is used to store this computer program. Smart contract features include validations, adding constraints, and applying business logic to the transactions. A blockchain developer may suffer from various challenges while preparing smart contracts. Smart contracts are made using new programming languages such as solidity, making the learning curve steep and hard to maintain. Moreover, the transaction's execution has minimal performance as nodes in a network follow order during transactions. Furthermore, network concurrency and high parallelism is also attained. We employ Node.js and Java programming languages to implement the smart contract. The presented system has various functions for interaction among users and ledger. For instance, details regarding payment history, fuel storage can be analyzed by the pump manager. A smart car user can record previous refueling visits, payment history, and car consumption. A user holds complete access to CRUD operations and can query the permitted information. The smart contract-based application accepts the transaction, runs different queries on it, and then modifies the ledger state by combining the transaction in a block and giving back the amended result in the form of a response. The illustration about the ledger queries and ledger updates harnessing a smart contract is shown in Figure 6.

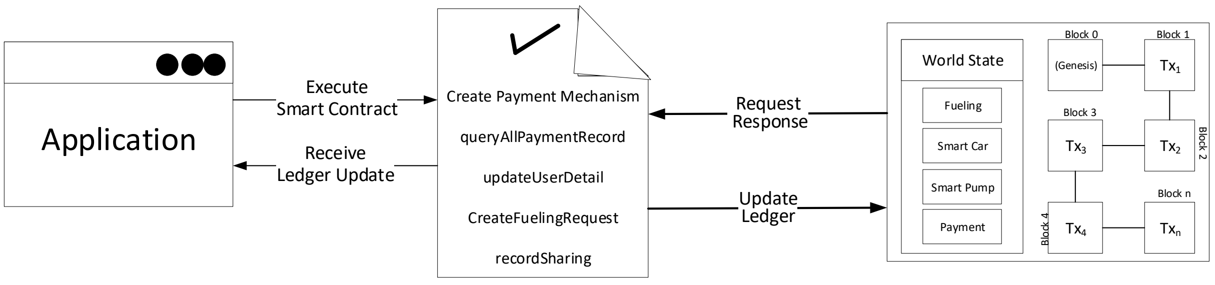


Figure 6. Ledger modification using a smart contract.

3.4. PetroBlock Network Structure

Figure 7 presents the network structure of the proposed PetroBlock blockchain network. The main participants are the pump manager and the car user. They can access the secure payment blockchain network with the user application by invoking the exposed REST API's. The Membership Service Provider (MSP) is used for user authentication, signature generation, credential validation, credential issuance, and verification. The orderer service provides the interface through which peers can connect to the channel and is responsible for transaction orders. The proposed blockchain-based payment mechanism for fueling smart vehicles comprises multiple peer nodes that can endorse the proposal for a transaction or write a block of a transaction to the ledger. Each peer node consists of chaincode, endorsement policies, and storage space.

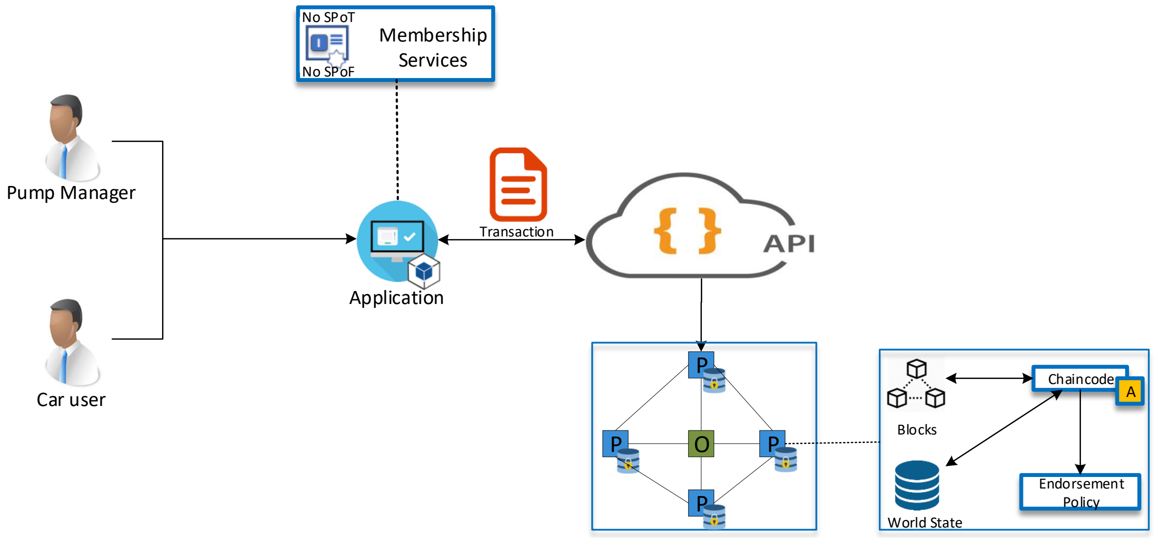


Figure 7. PetroBlock network structure.

3.5. Workflow for Channel Creation and Setting Chaincode in Proposed Blockchain Network

Figure 8 presents a workflow of channel creation in the proposed PetroBlock platform. The user-like administrator submits a request to the application, which creates a fabric channel to isolate chaincode to the participating member only. This allows the administrator to add the car user and pump manager as a participant to a channel. The channel creation event requests the selected participants through a chain messaging mechanism so that the participating member's parties can join the channel. The receiving member application accepts the channel message, then invites its peer to join the channel. Once the channel is joined, the application installs the chaincode on the peer file system. This process then allows multiple organization peers to send transactions or query the Chaincode with its state secured in the channel.

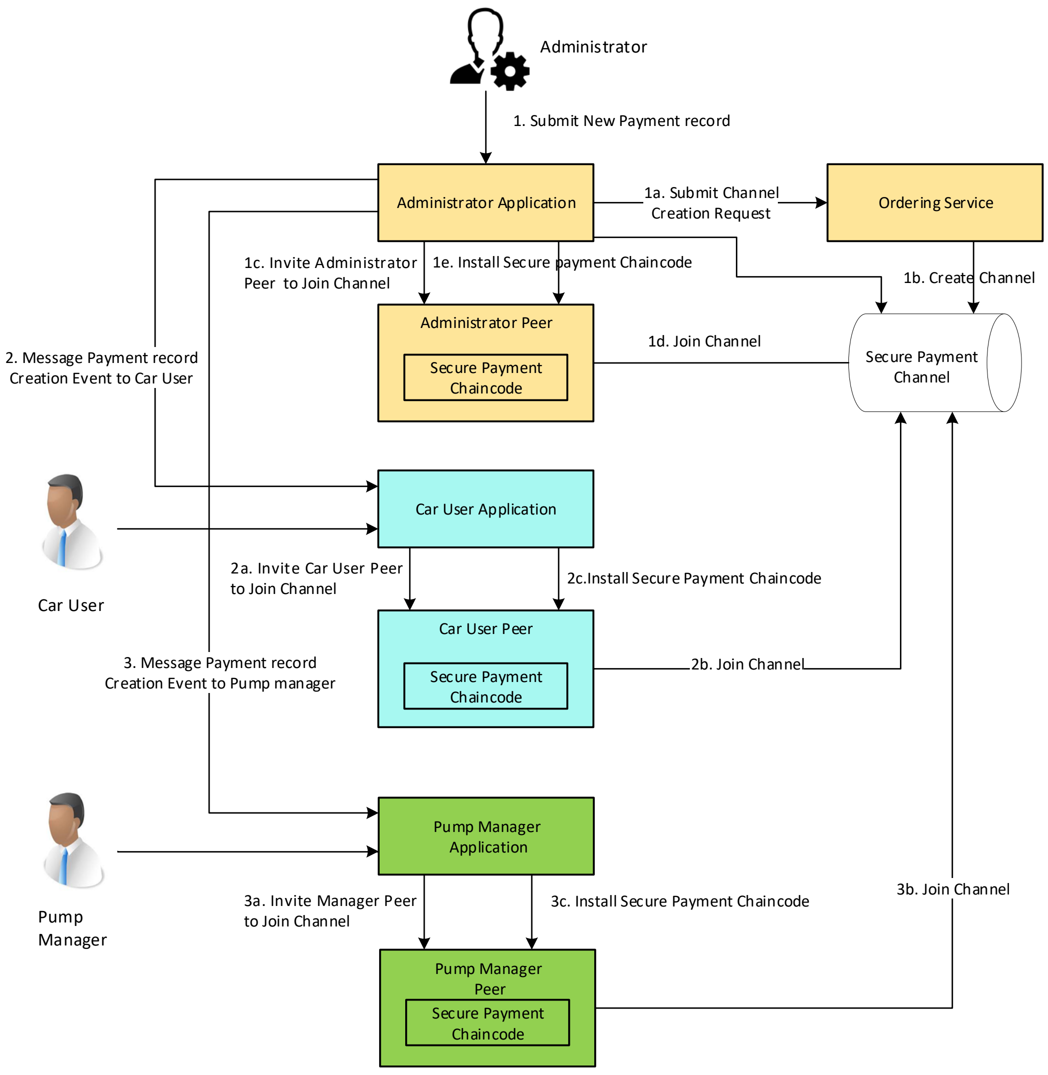

Administrator
1. Submit New Payment record
Administrator Application
1a. Submit Channel Creation Request
Ordering Service
1b. Create Channel
1c. Invite Administrator Peer to Join Channel
1e. Install Secure payment Chaincode
Administrator Peer
Secure Payment Chaincode
1d. Join Channel
Secure Payment Channel
2. Message Payment record Creation Event to Car User
Car User
Car User Application
2a. Invite Car User Peer to Join Channel
2c.Install Secure Payment Chaincode
Car User Peer
Secure Payment Chaincode
2b. Join Channel
3. Message Payment record Creation Event to Pump manager
Pump Manager
Pump Manager Application
3a. Invite Manager Peer to Join Channel
3c. Install Secure Payment Chaincode
3b. Join Channel
Pump Manager Peer
Secure Payment Chaincode

Figure 8. Workflow for channel creation and setting chaincode in proposed blockchain network.

4. PetroBlock Implementation

4.1. Development Environment

The designed blockchain platform is developed using Hyperledger Fabric [8,9], Hyperledger Composer [11,13], Docker Composer, Docker Engine, and CLI tool. The configuration of the proposed system is shown in Table 1. The blockchain platform is deployed on a Ubuntu platform that involves the Hyperledger Composer, composer-rest-server, and node. We use Ubuntu Linux 18.04 LTS on Intel Core i-5-8500 @3.2 GHz processor with 8-gigabyte memory. Additionally, we use the docker engine for configuring the docker image and container. We also use docker composer for configuring the composer-rest-server. We use Hyperledger Fabric, an open-source platform hosted by the Linux foundation for developing blockchain applications. Moreover, we also use composer-playground to develop and design the business network definition. We use composer-rest-server to create the REST API for the entities involved in the proposed blockchain platform to expose the web services.

Table 1. Development Environment.

Tools	Description
Hyperledger Fabric	v1.2
Docker Engine	Version 18.06.1-ce
Docker Composer	1.13.0
IDE	Composer-playground
CLI	Composer-rest-server
Operating System	Ubuntu Linux 18.04.1 LTS
Programming Language	HTML, CSS, JavaScript,Node.js
Browser	Microsoft Edge, Google Chrome, Firefox
External Libraries	jQuery, Bootstrap

4.2. Business Model

The Hyperledger Fabric is divided into three component, i.e., transaction, assets, and participant as shown in Figure 9. In the proposed business model, participants are the *caruser* and *pumpManager*. Assets includes *PumpMoneyPool*, *PaymentRecord*, *Bills*, *SmartCar*, *fuel*, and *smartpump*. Finally transaction are ShareFuelBillwithCarUser, PayFuelBill, UpdateCarRemainingFuel, UpdateFuelPrice, UpdatePumpStorage, UpdateCarVisit, UpdateFuelConsumption, and UpdatePaymentInformation

4.3. Smart Contract Modeling

In Hyperledger Composer, we use the composer-rest-server to create the REST APIs services to expose the blockchain platform's services. In the proposed blockchain platform, we use the following services mentioned in Table 2. These services consist of three parts: resource, verb, and action. The resource is the request path through which we call the services. Verb contains the type of action applied like PUT, POST, GET, DELETE on a particular verb. The action is the service modules of the proposed system.

Table 2. HTTP request in the proposed system.

Resource	Verb	Action
/api/CarUser	POST,GET,PUT,DELETE	Smart Car User Management
/api/PumpManager	POST,GET,PUT,DELETE	Smart Pump Management
/api/PumpMoneyPool	POST,GET,PUT,DELETE	Pump Money Pool Management
/api/PayFuelBill	POST	Pay Fuel Bill to Pump
/api/Bills	POST,GET,PUT,DELETE	Fuel Bill Management
/api/SmartCar	POST,GET,PUT,DELETE	Smart Car Management
/api/PaymentRecord	GET,POST,PUT,DELETE	Payment Record Management
/api/ShareFuelBill	POST	Share Fuel Bill with Car User
/api/UpdateCarFuel	POST	Update Smart Car
/api/UpdateFuelPrice	POST	Fuel Management
/api/UpdatePumpFuelStorage	POST	Pump Management
/api/UpdateFuelPaymentInformation	POST	Bill Management

The transaction is used to interact with the entities involved in blockchain technologies like assets, participants. Table 3 mentions the transactions and events defined in the proposed blockchain-based payment mechanism for fueling smart vehicles. Transactions are defined to provide extra functionality to the user of the system through REST APIs. Events are also part of a transaction and notify the system user if the user application triggers a specific transaction.

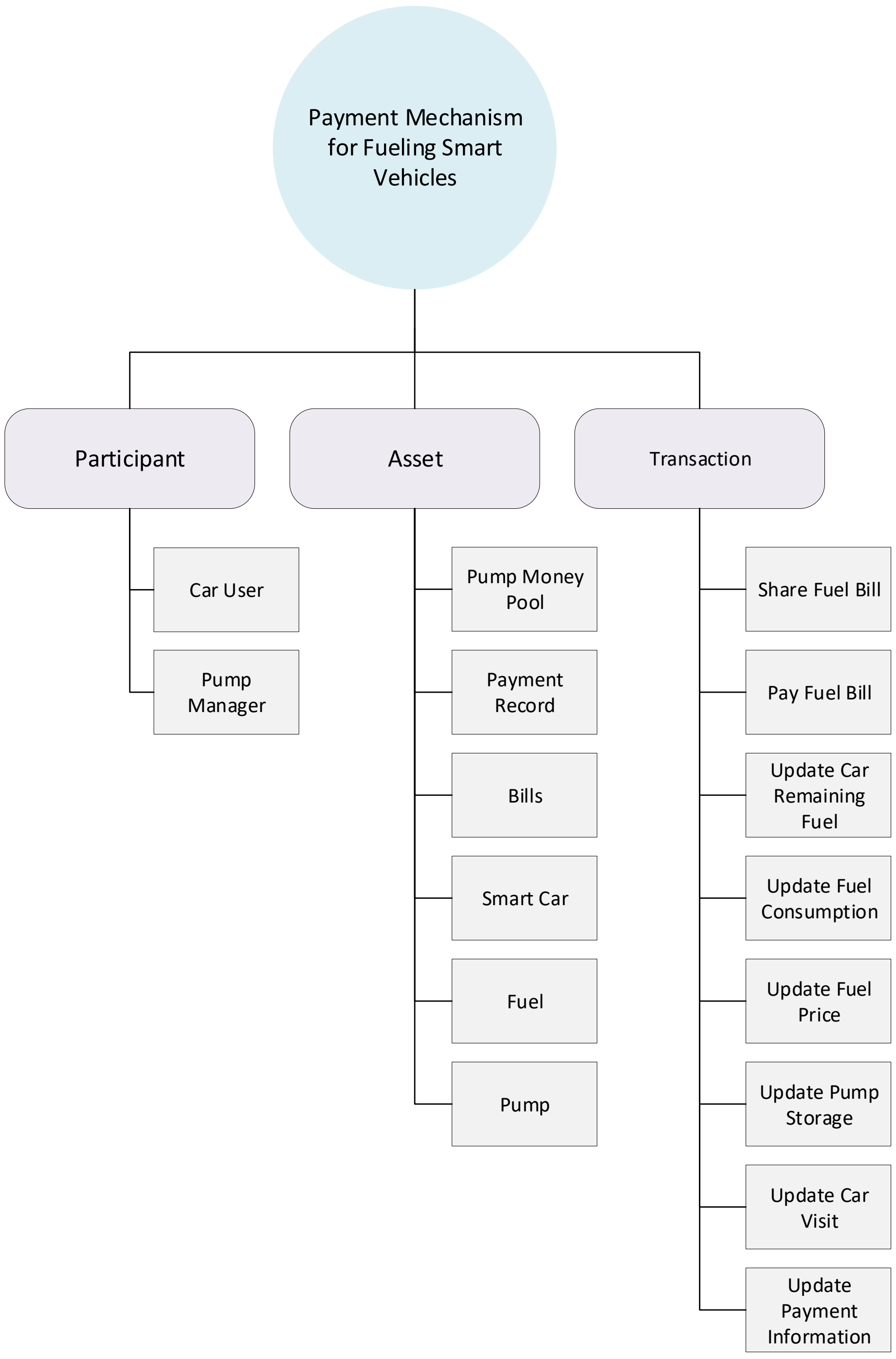


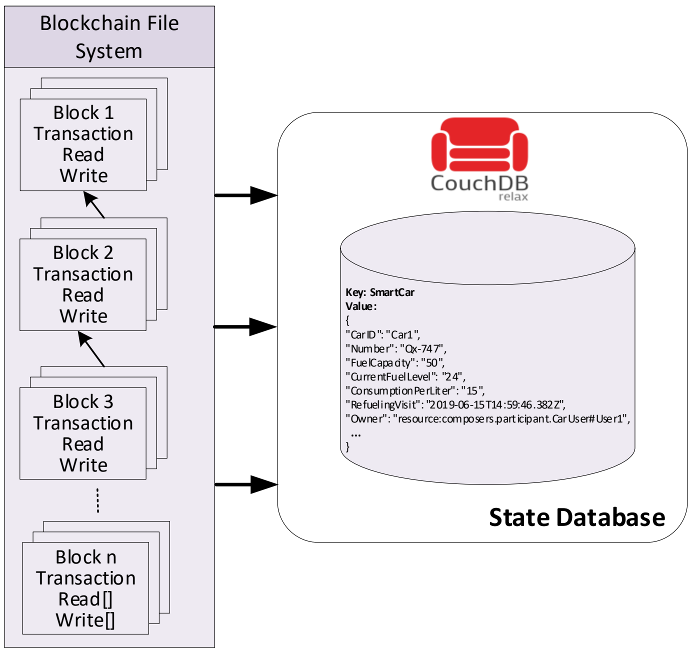
Figure 9. Business model of the proposed blockchain platform.

4.4. Distributed Ledger Storage Structure

This section encompasses details about the structure of the ledger from the presented blockchain platform. The system is split into two parts: (1) blockchain and (2) world state. The current state is another name of the world state in which a ledger state's present values are kept in a database. The proposed system improves the transaction processing performance as traversing of the transaction log is not required. Apache CouchDB state database supports rich queries for smart contract data modeling in the form of JavaScript Object Notation (JSON). To execute content-based JSON queries in CouchDB, data must be modeled in JSON format. The query methods like getting, delete put in combination with a state key are supported by the format. This allows the invoking of smart contracts by the application by using simple APIs. As shown in Figure 10, one record is stated in the ledger, i.e., record1 of CouchDB having value for a key. A simple state value is supported by the CouchDB having only a complex state value and key-value pair having various other key-value teams.

Table 3. Transaction and event definition of the proposed blockchain platform.

Component	Type	Role
Update Fuel Payment Information	Transaction	Update fuel payment Information (e.g., price, date, car_no)
Update Pump Fuel Storage	Transaction	Update fuel storage information (e.g., currentFuel, remainingFuel)
Update Fuel Price	Transaction	Update fuel Price
Update Car Fuel	Transaction	Update car fuel tank level after refueling
Share Fuel Bill	Transaction	Grant access permission
Share Fuel Bill	Transaction	Grant access permission
Pay Fuel Bill	Transaction	Pay fuel bill after refueling
Update Fuel Payment Information Notification	Event	Notified that the payment information has been modified
Update Pump Fuel Storage Notification	Event	Fuel level notification has been sent to pump manager if fuel drop or increased from required level
Update Fuel Price Notification	Event	Price notification has been sent to car user
Update Car Fuel Notification	Event	Notification has been sent to car user after refueling
Share Fuel Bill Notification	Event	Notified that the specific record permission has been successfully granted to car user
Pay Fuel Bill Notification	Event	Notification has be sent after paying refueling bill

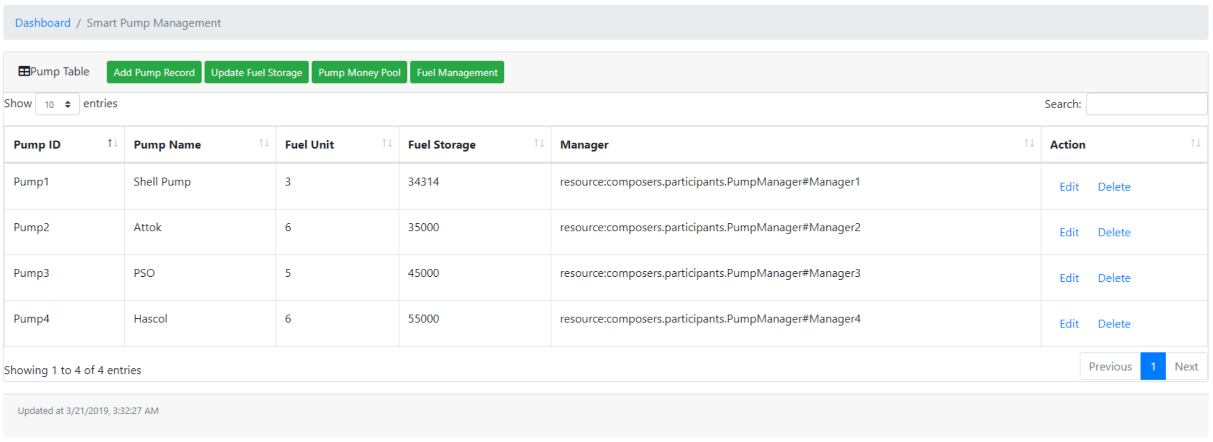

Blockchain File System
Block 1 Transaction Read Write
Block 2 Transaction Read Write
Block 3 Transaction Read Write
Block n Transaction Read[] Write[]
CouchDB relax
State Database

```
Key: SmartCar
Value:
{
"CarID": "Car1",
"Number": "Qx-747",
"FuelCapacity": "50",
"CurrentFuelLevel": "24",
"ConsumptionPerLiter": "15",
"RefuelingVisit": "2019-06-15T14:59:46.382Z",
"Owner": "resource:composers.participant.CarUser#User1",
  ...
}
```

Figure 10. Distributed ledger storage structure of the proposed blockchain platform.

4.5. Execution Results

This section provides some snapshots to describe the functionality of the proposed system. In Figure 11, the pump management dashboard is illustrated, which contains the pump's information. The web form allows the user to add a new pump in the system, update pump information, update fuel price, share fuel bill, and pump money pool. The $AddNewPump$ contains information like PumpID, PumpName, NoOfUnit, FuelCapacity, and ManagerID. The $updatepumpinformation$ updates the information regarding pump manager and fuel capacity. The $PumpMoneyPool$ manage the related monetary transaction in the proposed system like total sales of the day, and payment paid by the card for refueling. The proposed pump management dashboard offers the complete CRUD operation related to the pump services, as mentioned in Figure 12.



Pump ID	Pump Name	Fuel Unit	Fuel Storage	Manager	Action
Pump1	Shell Pump	3	34314	resource:composers.participants.PumpManager#Manager1	Edit Delete
Pump2	Attok	6	35000	resource:composers.participants.PumpManager#Manager2	Edit Delete
Pump3	PSO	5	45000	resource:composers.participants.PumpManager#Manager3	Edit Delete
Pump4	Hascol	6	55000	resource:composers.participants.PumpManager#Manager4	Edit Delete

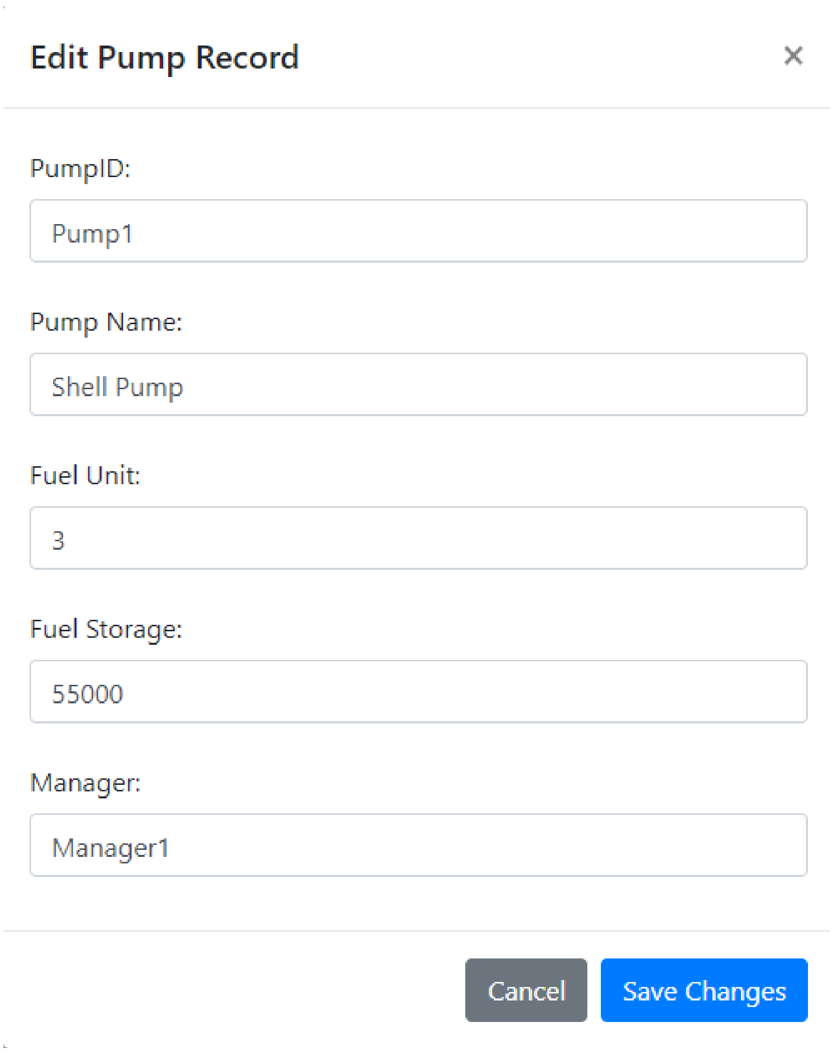
Showing 1 to 4 of 4 entries

Updated at 3/21/2019, 3:32:27 AM

Figure 11. Pump management dashboard.

The $updatepumpinformation$ in Figure 12 is used to modify the current pump information via invoking a request to the blockchain platform. The accurate request to the system repopulates the updated information on the user information. The information like pump manager and fuel capacity will be updated from composer-rest-server after a successful response.

Figure 13 shows the dashboard of smart car management contains the web form related to car management. The system allows the portal to add new cars, update car visits for refueling, update fuel consumption, update car remaining fuel, pay fuel bills, and update fuel payments. The $AddCar$ feature allows the system to add a new vehicle in the system after a successful transaction. The $updateCarvisitforRefueling$ is used to update the time and date of car visit for refueling. The $UpdateFuelConsumption$ is used to update the smart car's fuel consumption by updating parameters like No_of_KM_PerLit(Number of km per liter). The $UpdateCarRemainingfuel$ allows the user to update the car fuel after refueling. The user may modify the current information by sending a request to the blockchain. The client interface will repopulate the updated information. $PayFuelbill$ transaction is used to send fuel bills after refueling. The fuel bill will be calculated by multiplying the price per liter by the number of liters refueled in a car. The $UpdateFuelPayment$ updates the existing information by sending a request to the proposed blockchain system, and after a successful response, the data are repopulated on the user interface.



Edit Pump Record

PumpID: Pump1

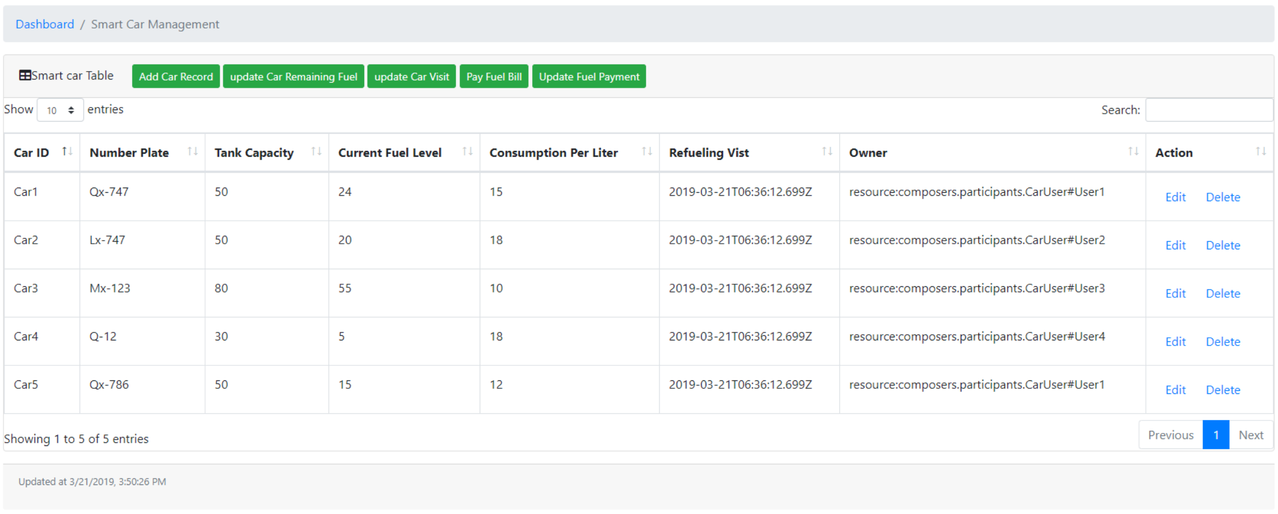
Pump Name: Shell Pump

Fuel Unit: 3

Fuel Storage: 55000

Manager: Manager1

Figure 12. Pump management dashboard: update pump record.



Car ID	Number Plate	Tank Capacity	Current Fuel Level	Consumption Per Liter	Refueling Vist	Owner	Action
Car1	Qx-747	50	24	15	2019-03-21T06:36:12.699Z	resource:composers.participants.CarUser#User1	Edit Delete
Car2	Lx-747	50	20	18	2019-03-21T06:36:12.699Z	resource:composers.participants.CarUser#User2	Edit Delete
Car3	Mx-123	80	55	10	2019-03-21T06:36:12.699Z	resource:composers.participants.CarUser#User3	Edit Delete
Car4	Q-12	30	5	18	2019-03-21T06:36:12.699Z	resource:composers.participants.CarUser#User4	Edit Delete
Car5	Qx-786	50	15	12	2019-03-21T06:36:12.699Z	resource:composers.participants.CarUser#User1	Edit Delete

Showing 1 to 5 of 5 entries

Updated at 3/21/2019, 3:50:26 PM

Figure 13. Smart car management dashboard.

5. Performance Evaluation

The designed platform is evaluated through Hyperledger Caliper [13], which is an open-source benchmark tool available for blockchain platforms. To evaluate the performance of the blockchain-based system, the Linux Foundation has developed Hyperledger Caliper. The designed model's performance is evaluated for transactions per second, resource utilization, and transaction history. Like transaction latency and transaction throughput, the performance parameters also indicate resource allocation (memory utilization, CPU consumption, I.O., etc.). Table 4 illustrates the environmental setup of Hyperledger Caliper.

Table 4. Environmental setup of Hyperledger Caliper.

Component	Description
Docker Engine	Version 18.06 -ce
CLI Tool	Node-gyp
Docker-Composer	Version 1.130
Node	v8.11.4

The latency and throughput are the two parameters used to analyze the performance of the designed clinical trial service platform. The throughput, also known as transaction per second, is segregated into two sub-groups, such as transaction throughput and read throughput. The transaction throughput is a valid transaction executed during a defined time period, also known as transaction per second (tps), as shown in Equation (1).

$$\text{Transaction per second} = \frac{\text{Successful Transaction}}{\text{Time (Sec)}} \tag{1}$$

Furthermore, the tps is measured using all nodes across the entire blockchain. Similarly, the read throughput is calculated as a total count of read operations in a specifically defined time known as read per second (rps) as shown in Equation (2).

$$\text{Read Per Second} = \frac{\text{Read Transaction}}{\text{Time (Sec)}} \tag{2}$$

The proposed clinical trial service framework is evaluated in terms of latency, such as read and transaction latency. The read latency is the total round trip time between the send request and receives a response as computed in Equation (3).

$$\text{Read Latency} = \text{Request response time} - \text{Request time} \tag{3}$$

Similarly, the transaction latency is the total time to authenticate a transaction, including the consensus algorithm's processing. In the developed system, we also defined the network threshold used to define the time to commit the transaction. This paper has used Practical Byzantine Fault Tolerance (PBFT); therefore, the network threshold is set to a hundred. The mathematical formulation of transaction latency is a calculation using Equation (4).

$$\text{Transaction Latency} = \text{Transaction commit time} \times \text{network threshold} - \text{Request time} \tag{4}$$

5.1. Simulation Results

Figure 14 shows information regarding transactions per second (tps) for the proposed blockchain platform. Here tps is deemed like throughput. The proposed model is evaluated by considering different groups. The group is split into three classes, including the number of students as 300, 500, and 1000. The first phase is investigated for the throughput from 300 users. Similarly, 500 users are considered for the second round and 1000 users for the third round. As we have seen the average throughput, the more the number of users, the better the system will perform. As demonstrated in Figure 14 , the user-group with 300 users, the average transactions are 30 for the elapsed time of 100ms. However, the value for tps increases by 20 transactions per second if the number of users increases to 500. Finally, with the user-group with 1000 users, the average number of transactions are 56.

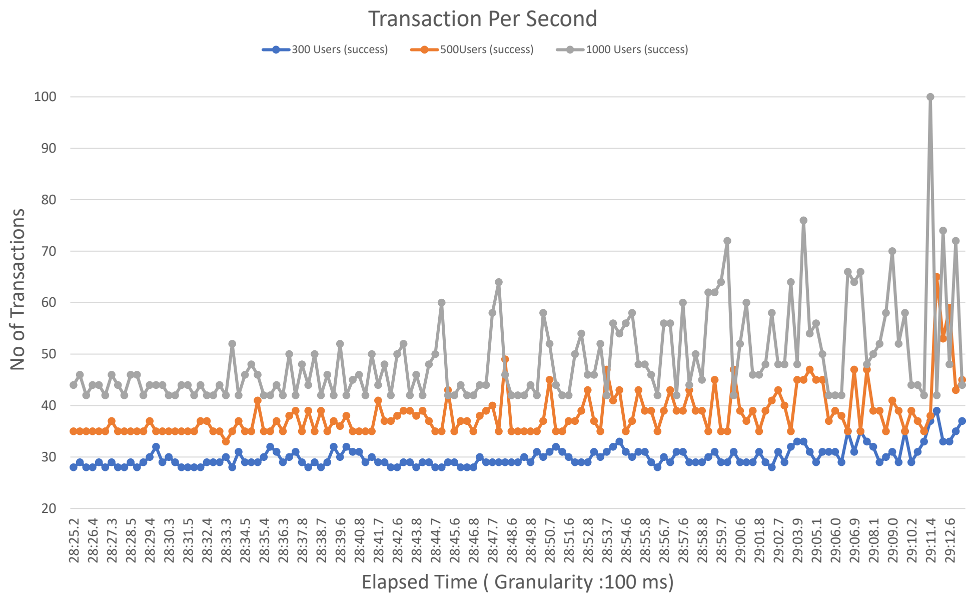


Figure 14. Transactions per second.

The latency of executing the proposed system's invoke transaction having three different sub-classes for minimum and maximum, and average latency is investigated and shown in Figure 15. The categories for user-group are comprised of 300, 500, and 100 users. The average latency for the group of 300 users is reported as 2709 ms. Similarly, the group of 500 users attained an increased value of latency, i.e., 2820 ms. The latency value reaches 2984 for 1000 users. This behavior illustrates that the average latency increases as the number of users increases; however, there is a minor difference between the three groups' latency values. Furthermore, the group with 300 users has the minimum latency compared to the 500 and 1000 users group. It is observed from the graph that the latency of the group with 1000 users increases linearly until the elapsed time of 29.11.4 ms after the proposed model converges into a stable state.

The latency value, obtained by executing a query function for three different user-groups, is shown in Figure 16. The average latency value for executing query transactions for a group with 300 users is 256. Similarly, the average latency values are 327 and 450 for user groups with 500 and 1000, respectively. We have observed that minimum latency values for 300, 500, and 1000 users are 68, 71, and 97, respectively, and the maximum latency values for 300, 500, and 1000 users are 378, 455, and 850, respectively.

Hyperledger Caliper is employed to assess the resource utilization of the blockchain network in five iterations. The resource utilization results are expressed in terms of maximum and average memory and central processing unit (CPU) usage rate as illustrated in Table 5. The average CPU utility was captured as 5.77% and memory usage rate at 97.37 MB for the peer. The average CPU utility was captured as 1.25% and memory usage rate at 26.5 MB to the peer for the ordering node. Similarly, the average CPU utility was captured as 0% and memory usage rate at 5.5 MB for CA node. The outcomes of resource utilization reveal that the blockchain network holds a declining rate of resource occupation, increased reliability, and a promising user experience.

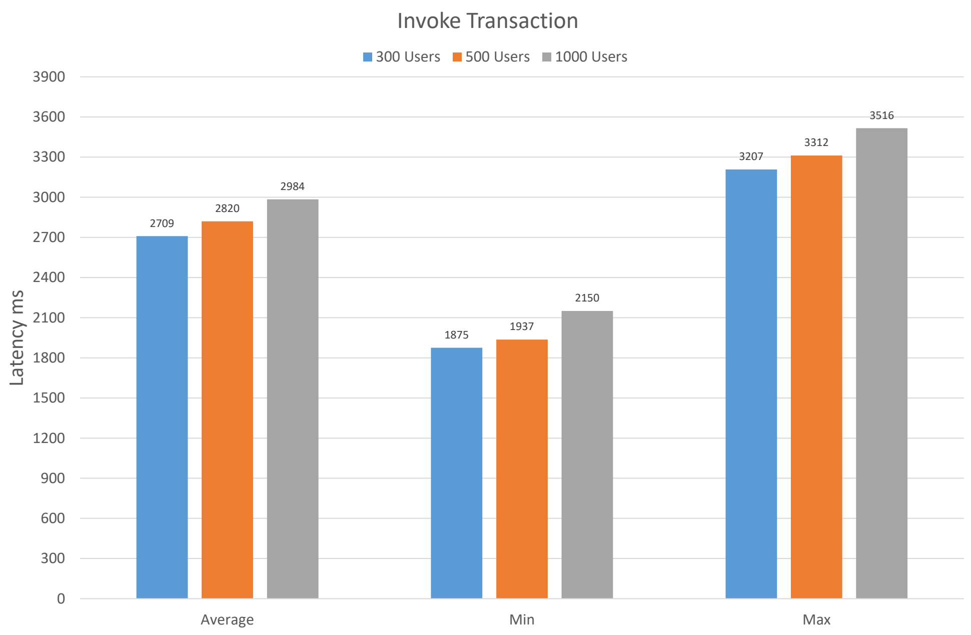


Figure 15. Latency in invoke transaction.

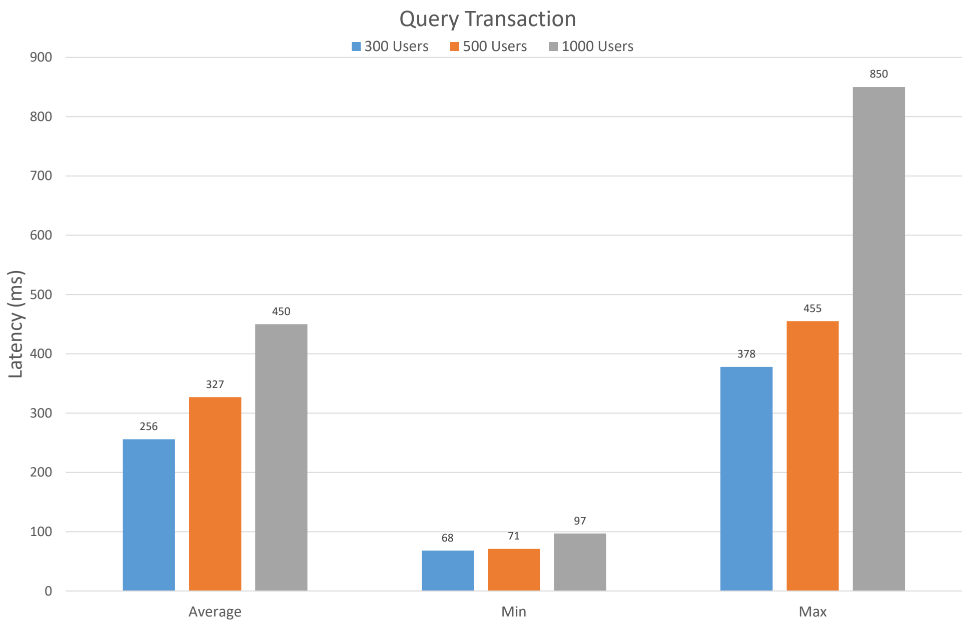


Figure 16. Latency in query transaction.

Table 5. Resource utilization analysis of the proposed system.

Type	Name	CPU (max)	CPU (avg)	Memory (max)	Memory (avg)	Traffic In	Traffic Out
Process	local-client.js	14.64%	8.76%	105.2 MB	90.5 MB	-	-
Docker	peer1.pump1.com	12.44%	5.59%	106.6 MB	98.5 MB	1 MB	421.3 KB
Docker	peer0.pump2.com	17.09%	6.24%	105.7 MB	96.7 MB	1.6 MB	666.7 KB
Docker	peer0.pump1.com	15.02%	4.56%	89.5 MB	82.3 MB	697 KB	287.6 KB
Docker	peer1.pump2.com	0.00%	6.54%	112.8 MB	105.8 MB	819 B	0 B
Docker	orderer.com	14.95%	6.75%	93.6 MB	88.7 MB	5 MB	5.6 MB
Docker	ca_nodeDepartment1	0.00%	0.00%	5.5 MB	5.5 MB	546 B	0 B
Docker	ca_nodeDepartment2	0.00%	0.00%	0 B	0 B	0 B	-

5.2. Security Analysis

We analyze the security of PetroBlock against the following attacks:

- *Key attack:* PetroBlock uses elliptic curve encryption to generate the key pair, and the attacker cannot compute the private key as solving the elliptic curve logarithm problem is difficult which ensures the key security. In addition, a different temporary private key is generated for each session agreement between nodes. In this way, the leakage of one private key will not affect: (a) during the session, the attacker cannot compute a session key of an already established session between two nodes, and (b) after the session is terminated, the leaked private key becomes useless.
- *Replay attack:* PetroBlock employs a separate temporary private key for each session agreement between nodes. As private keys have bounded lifetime, the replay attack cannot succeed and can be easily detected.
- *Impersonation attack:* This attack could only be successful if the attack can obtain the private key. As PetroBlock uses elliptic curve encryption and a separate temporary private key for each session agreement, this attack cannot succeed.
- *Sybil attack:* There are various ways to mitigate the effect of Sybil attack on PetroBlock: (a) raise the cost of creating a new identity. This resource requirement limits the number of attackers that attempt to create fake identities, (b) adoption of two-factor authentication mechanism, or (c) collecting the I.P. and MAC addresses of participants, which allows detecting those who have different identities that map to the same address.
- *False data injection attack:* In blockchains, consensus algorithms are executed before validating the records. Each node can verify the integrity of the received record when a positive consensus is reached.
- *Tampering attack:* Public key cryptosystem is used to encrypt and sign the transactions. This means that a tampering node cannot tamper with the transaction as it does not have the private key of the signer node. In addition, as shown above, PetroBlock is resilient against key attacks, and hence, the private key cannot be exploited by adversaries.
- *Modification attack:* As shown above, this attack is not possible as adversaries cannot exploit the private keys.
- *Hiding blocks attack:* In PetroBlock, each record has its sequence number. The requested vehicles must provide their records in case others request them. If any node refuses to provide records, it will be isolated, and no one will interact with others.
- *Man-in-the-middle attack:* PetroBlock ensures mutual authentication among nodes, as it uses temporary private keys for each session agreement, and hence it can prevent man-in-the-middle attacks.

6. Comparison and Significance

This section encompasses details regarding the comparison of the proposed system with contemporary state-of-the-art studies. To ensure the validity and efficiency of the proposed model, we carried out a benchmark study. The results obtained via evaluation are illustrated in Table 6.

Table 6. Comparative analysis of proposed PetroBlock framework with existing platforms.

Approach	Native Cryptocurrency	Client Supported	Required Mining	Smart Contract	Device as Node	Access Policy	Consensus Mechanism
[36]	✓	✓	✓	✗	✓	Permissionless	Every Nodes
[37]	✓	✗	✓	✓	✗	Permissionless	Every Nodes
[38]	✓	✓	✓	✓	✓	Permissionless	Every Nodes
[39]	✗	✓	✗	✓	✗	Permissionless/ Permissioned	Arbitrary Nodes
[40]	✓	✓	✓	✓	✓	Permissionless	Every Nodes
[41]	✓	✓	✓	✓	✓	Permissionless	Every Nodes
PetroBlock Framework	✗	✓	✗	✓	✗	**Permissioned**	**Arbitrary Nodes**

Following are characteristics that have been considered to compare the proposed study with existing studies. It can be seen in the table that the study [39] holds very identical characteristics to our proposed approach; therefore, we have drawn a comparison of this study with the proposed study. A similar simulation environment to [39] is chosen for comparative analysis. During a simulation, a network of 50 peers was selected, and the simulation ran for 60 seconds that executed 960 transactions. The time cost of verifying a new block in the network forms a processing time metric. Figure 17 shows the results of the simulation for the evaluation of processing overhead. It can be seen in the graph that our study has resulted in less overhead than the other approach when a number of blocks varied from 10 to 60. Overall, the proposed study reduced 23% of the processing time.

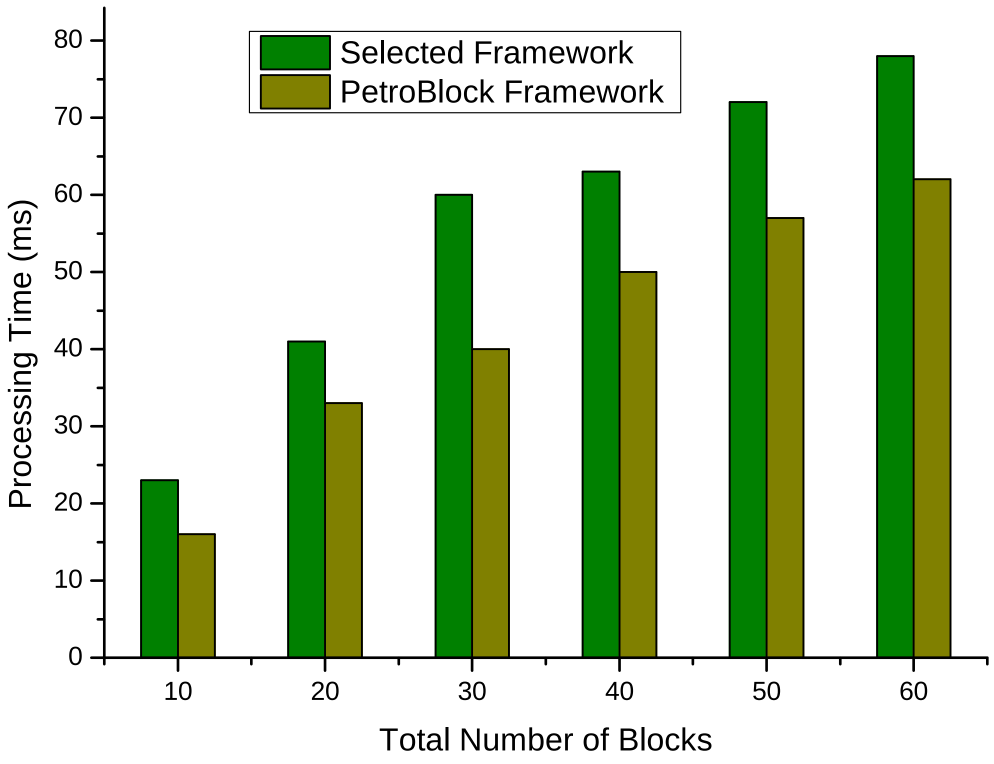


Figure 17. Processing overhead comparison analysis.

Most of the systems have been formed on a permissionless blockchain network that allows participants to participate anonymously. This indicates that these systems do not have the confidentiality of contracts and transaction data. These systems address this non-confidentiality problem by issuing their own token to fuel smart contract execution. Adverse association with cryptocurrencies can have a major impact on transaction cost and speed. Additionally, it prevents interaction with other distributed systems since then tokens should be unified. On the other hand, our proposed approach is designed on a permissioned network due to which participants are unable to introduce malicious code via smart contract. All participants are aware about each other and all actions are captured in the blockchain as per endorsement policy that was formed for transaction type and network. Moreover, various contemporary systems do not have resource-constrained IoT devices because they have time-consuming mining due to full node deployment on these devices. However, the integration of IoT with blockchain has always suffered from resource-constrained architecture as consensus algorithms are limited to function within these constraints. Some of the existing studies deploy heavy consensus algorithms on the devices that are components of the IoT system, like gateways. However, these IoT gateways have limited storage space. Many existing platforms do not have a facility for lightweight nodes, and full nodes' deployment must be done on gateways for verification of blocks and transactions. Moreover, this results in making gateways target as they serve as a bridge among the internet and devices. Unlike these systems, our proposed model provides a lightweight solution that mitigates the need to integrate blockchain technology and IoT devices. In addition, it does not require modification of these devices.

We have incorporated blockchain as an external entity in order to have secure and reliable storage. Moreover, there is no need to download the entire blockchain network to validate transactions of IoT devices. The proposed system can play an effective role in various IoT scenarios having the limited capability. Additionally, our service APIs are used as a cross-platform to enable communication among blockchain networks and IoT devices. This study also provides the feasibility of integration with other contemporary systems.

This paper has considered a real-life case study for smart space executed during the experimentation phase to evaluate the proposed system's effectiveness. The proposed system has specifically considered that the model can easily be extended to fulfill other domains like data marketplace, energy trading, and supply chain. For instance, the proposed model may be extended in the food supply chain network as blockchain provides reliable and traceable functionality. IoT sensors can play an effective role hereby being placed on a food item such as fish relegated for transport and remotely sensed data like humidity, temperature, and location. All food processing phases, such as digital compliance information, test results, and audit certificates, may be captured and stored in a blockchain network by enabling access to a shared ledger among all the supply chain parties. The dire need of having an IoT blockchain application with a permissioned network, user-friendly interface, and architecture, no currency exchange, high transactional throughput has been focused on in this study.

7. Conclusions

This paper has presented PetroBlock: a novel blockchain-based payment platform that allows to refuel smart vehicles in a smart pump in a privacy preserved manner. The system has employed Hyperledger Fabric for the said purpose. We have harnessed a smart contract for providing a secure payment transaction for refueling. The model presents a proof-of-concept application using blockchain technology, which maintains a record of all the refueling payments in a decentralized manner. We have employed Hyperledger Fabric, which is permissioned blockchain architecture which provides modular, scalable, and secure foundation for PetroBlock framework. Moreover, we have carried out several experiments in order to evaluate the Petroblock framework performance by using Hyperledger Calliper in terms of resource utilization, latency, and throughput. The result yields that harnessing blockchain technology enhances the performance of the proposed platform. In future, we have aimed to test the interoperability of the designed Petroblock framework with the other IoT-blockchain platform. Moreover, in order to enhance the performance, we are intended to utilize other consensus algorithms for improving the transaction processing rate for fast query execution.

Author Contributions: Data curation, F.J.; Formal analysis, F.J.; Funding acquisition, O.C.; Investigation, F.J.; Methodology, H.J.; Software, A.K.; Supervision, A.D.; Validation, M.A.F.; Visualization, F.J.; Writing—original draft, H.J.; Writing—review & editing, O.C., A.K., A.D., M.A.F. and H.J. All authors have read and agreed to the published version of the manuscript.

Funding: Dr. Omar Cheikhrouhou thanks Taif university for its support under the project Taif University Researchers supporting project number (TURSP-2020/55), Taif university, Taif, Saudi Arabia.

Acknowledgments: The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

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

References

1. Evans, D. The internet of things: How the next evolution of the internet is changing everything. *CISCO White Paper* **2011**, *1*, 1–11.
2. Jamil, F.; Iqbal, M.A.; Amin, R.; Kim, D. Adaptive thermal-aware routing protocol for wireless body area network. *Electronics* **2019**, *8*, 47. [CrossRef]
3. Hanada, Y.; Hsiao, L.; Levis, P. Smart Contracts for Machine-to-Machine Communication: Possibilities and Limitations. In Proceedings of the 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), Bali, Indonesia, 1–3 November 2018; IEEE: New York, NY, USA, 2018; pp. 130–136.

4. Jing, Q.; Vasilakos, A.V.; Wan, J.; Lu, J.; Qiu, D. Security of the Internet of Things: perspectives and challenges. *Wirel. Netw.* **2014**, *20*, 2481–2501. [CrossRef]
5. Maroufi, M.; Abdolee, R.; Tazekand, B.M. On the Convergence of Blockchain and Internet of Things (IoT) Technologies. *arXiv* **2019**, arXiv:1904.01936.
6. Nguyen, Q.K. Blockchain—A financial technology for future sustainable development. In Proceedings of the 2016 3rd International Conference on Green Technology and Sustainable Development (GTSD), Kaohsiung, Taiwan, 24–25 November 2016; IEEE: New York, NY, USA, 2016; pp. 51–54.
7. Zhaoyang, D.; Fengji, L.; Liang, G. Blockchain: a secure, decentralized, trusted cyber infrastructure solution for future energy systems. *J. Mod. Power Syst. Clean Energy* **2018**, *6*, 958–967.
8. Cachin, C. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*; IBM: Ruschlikon, Switzerland, 2016; Volume 310.
9. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Porto Portugal April, 2018*; ACM: New York, NY, USA, 2018; p. 30.
10. Thakkar, P.; Nathan, S.; Viswanathan, B. Performance benchmarking and optimizing hyperledger fabric blockchain platform. In Proceedings of the 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Milwaukee, WI, USA, 25–28 September 2018; IEEE: New York, NY, USA, 2018; pp. 264–276.
11. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B. Performance analysis of hyperledger fabric platforms. *Secur. Commun. Netw.* **2018**, *2018*, 3976093 . [CrossRef]
12. Baliga, A.; Solanki, N.; Verekar, S.; Pednekar, A.; Kamat, P.; Chatterjee, S. Performance Characterization of Hyperledger Fabric. In Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, Switzerland, 20–22 June 2018; IEEE: New York, NY, USA, 2018; pp. 65–74.
13. Sukhwani, H.; Wang, N.; Trivedi, K.S.; Rindos, A. Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network). In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; IEEE: New York, NY, USA, 2018; pp. 1–8.
14. Jamil, F.; Hang, L.; Kim, K.; Kim, D. A novel medical blockchain model for drug supply chain integrity management in a smart hospital. *Electronics* **2019**, *8*, 505. [CrossRef]
15. Jamil, F.; Iqbal, N.; Ahmad, S.; Kim, D. Peer-to-Peer Energy Trading Mechanism based on Blockchain and Machine Learning for Sustainable Electrical Power Supply in Smart Grid. *IEEE Access* **2021**, *9*, 39193–39217. [CrossRef]
16. Jamil, F.; Kahng, H.K.; Kim, S.; Kim, D.H. Towards Secure Fitness Framework Based on IoT-Enabled Blockchain Network Integrated with Machine Learning Algorithms. *Sensors* **2021**, *21*, 1640. [CrossRef]
17. Shahbazi, Z.; Byun, Y.C. Towards a secure thermal-energy aware routing protocol in Wireless Body Area Network based on blockchain technology. *Sensors* **2020**, *20*, 3604. [CrossRef]
18. Shahbazi, Z.; Byun, Y.C. Integration of Blockchain, IoT and Machine Learning for Multistage Quality Control and Enhancing Security in Smart Manufacturing. *Sensors* **2021**, *21*, 1467. [CrossRef] [PubMed]
19. Shahbazi, Z.; Byun, Y.C. Improving Transactional Data System Based on an Edge Computing–Blockchain–Machine Learning Integrated Framework. *Processes* **2021**, *9*, 92. [CrossRef]
20. Shahbazi, Z.; Byun, Y.C. A Procedure for Tracing Supply Chains for Perishable Food Based on Blockchain, Machine Learning and Fuzzy Logic. *Electronics* **2021**, *10*, 41. [CrossRef]
21. Jamil, F.; Ahmad, S.; Iqbal, N.; Kim, D.H. Towards a remote monitoring of patient vital signs based on IoT-based blockchain integrity management platforms in smart hospitals. *Sensors* **2020**, *20*, 2195. [CrossRef]
22. Jamil, F.; Kim, D. Payment Mechanism for Electronic Charging using Blockchain in Smart Vehicle. *Korea* **2019**, *30*, 31.
23. Iqbal, N.; Jamil, F.; Ahmad, S.; Kim, D. A Novel Blockchain-Based Integrity and Reliable Veterinary Clinic Information Management System Using Predictive Analytics for Provisioning of Quality Health Services. *IEEE Access* **2021**, *9*, 8069–8098. [CrossRef]
24. Knirsch, F.; Unterweger, A.; Engel, D. Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. *Comput. Sci. Res. Dev.* **2018**, *33*, 71–79. [CrossRef]
25. Ginsburgh, I.; Runes, E. Automatic Fueling System for Automobiles. U.S. Patent 3,642,036, 30 April 1972.
26. Jamil, F.; Kim, D.H. Improving Accuracy of the Alpha–Beta Filter Algorithm Using an ANN-Based Learning Mechanism in Indoor Navigation System. *Sensors* **2019**, *19*, 3946. [CrossRef]
27. Pustišek, M.; Kos, A.; Sedlar, U. Blockchain based autonomous selection of electric vehicle charging station. In Proceedings of the 2016 international conference on identification, information and knowledge in the Internet of Things (IIKI), Beijing, China, 20–21 October 2016; IEEE: New York, NY, USA, 2016; pp. 217–222.
28. Huckle, S.; Bhattacharya, R.; White, M.; Beloff, N. Internet of things, blockchain and shared economy applications. *Procedia Comput. Sci.* **2016**, *98*, 461–466. [CrossRef]
29. Pedrosa, A.R.; Pau, G. ChargeltUp: On blockchain-based technologies for autonomous vehicles. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*; ACM: New York, NY, USA, 2018; pp. 87–92.

30. Liu, C.; Chai, K.K.; Zhang, X.; Lau, E.T.; Chen, Y. Adaptive blockchain-based electric vehicle participation scheme in smart grid platform. *IEEE Access* **2018**, *6*, 25657–25665. [CrossRef]
31. Kim, N.H.; Kang, S.M.; Hong, C.S. Mobile charger billing system using lightweight Blockchain. In Proceedings of the 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, Korea, 27–29 September 2017; IEEE: New York, NY, USA, 2017; pp. 374–377.
32. Ferrag, M.A.; Derdour, M.; Mukherjee, M.; Derhab, A.; Maglaras, L.; Janicke, H. Blockchain technologies for the internet of things: Research issues and challenges. *IEEE Internet Things J.* **2019**, *6*, 2188–2204. [CrossRef]
33. Huang, X.; Xu, C.; Wang, P.; Liu, H. LNSC: A security model for electric vehicle and charging pile management based on blockchain ecosystem. *IEEE Access* **2018**, *6*, 13565–13574. [CrossRef]
34. Kuo, T.T.; Kim, H.E.; Ohno-Machado, L. Blockchain distributed ledger technologies for biomedical and health care applications. *J. Am. Med. Informatics Assoc.* **2017**, *24*, 1211–1220. [CrossRef] [PubMed]
35. Otte, P.; de Vos, M.; Pouwelse, J. TrustChain: A Sybil-resistant scalable blockchain. *Future Gener. Comput. Syst.* **2020**, *107*, 770–780. [CrossRef]
36. Lin, J.; Shen, Z.; Miao, C. Using blockchain technology to build trust in sharing LoRaWAN IoT. In Proceedings of the 2nd International Conference on Crowd Science and Engineering, Beijing, China, 6–7 July 2017; pp. 38–43.
37. Özyılmaz, K.R.; Yurdakul, A. Work-in-progress: Integrating low-power IoT devices to a blockchain-based infrastructure. In Proceedings of the 2017 International Conference on Embedded Software (EMSOFT), Seoul, Korea, 15–20 October 2017; IEEE: New York, NY, USA, 2017; pp. 1–2.
38. Huh, S.; Cho, S.; Kim, S. Managing IoT devices using blockchain platform. In Proceedings of the 2017 19th international conference on advanced communication technology (ICACT), PyeongChang, Korea, 19–22 February 2017; IEEE: New York, NY, USA, 2017; pp. 464–467.
39. Dorri, A.; Kanhere, S.S.; Jurdak, R. Towards an optimized blockchain for IoT. In Proceedings of the 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI), Pittsburgh, PA, USA, 18–21 April 2017; IEEE: New York, NY, USA, 2017; pp. 173–178.
40. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: a new Blockchain-based access control framework for the Internet of Things. *Secur. Commun. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
41. Yu, C.; Zhang, L.; Zhao, W.; Zhang, S. A blockchain-based service composition architecture in cloud manufacturing. *Int. J. Comput. Integr. Manuf.* **2020**, *33*, 701–715. [CrossRef]