Received: 3 March 2021 | Revised: 30 March 2021 | Accepted: 2 May 2021

DOI: 10.1002/spy2.172

RESEARCH ARTICLE

Smart contract-based land registry system to reduce frauds and time delay

Sandeep Kumar Panda[1] | Gouse Baig Mohammad[2] | Sachi Nandan Mohanty[3,4] | Sipra Sahoo[5]

[1]Department of Data Science and Artificial Intelligence, IcfaiTech, Faculty of Science and Technology, ICFAI Foundation of Higher Education, Hyderabad, India

[2]Department of Computer Science and Engineering, Vardhman College of Engineering, Hyderabad, India

[3]Department of Computer Engineering, College of Engineering Pune, Pune, India

[4]Department of Informatics and Computing, Singidunum University, Serbia

[5]Department of Computer Science and Engineering, SoA Deemed to be University, Bhubaneswar, Odisha, India

Correspondence
Sachi Nandan Mohanty, Department of Computer Engineering, College of Engineering Pune, Pune, India.
Email: sachinandan09@gmail.com; snandan.mohanty@singidunum.ac.rs

Abstract

In today's scenario, many news related to counterfeit land titles, fraud land registry, delay in ownership transfer, the involvement of government officers in fraudulent activities is frequently being heard. However, this depicts that the existing land registry system is not efficient to provide security and timely settlement of transactions between the seller and buyer. To solve this problem, we proposed a blockchain-based land registry system in this article. The specialty and popularity of blockchain technology is its transparency and security. Blockchain is being inculcated with the trait of persistence, immutability, decentralization. Its ascent to new opportunity of efficiency and cost saving. It can provide right framing for digital asset, online payment, and transfer of remittance. Additional to this it can check upon black money laundering. Enterprise that can use blockchain technology can gain faith of consumer. In this article, we proposed a decentralized application. In particular, for creating and deploying the smart contract, we used Ethereum network. The deployed contracts are interacted through frontend web pages. React is used for the development of web page. For server and routing purposes, Next.js is used. Finally, the results and analysis show that our proposed model is efficient and viable.

KEYWORDS

blockchain, DApp, Ethereum, immutability, Next.js, react

1 | INTRODUCTION

A common man is worried about the security of their personal data and loss of money because every day such threats are published in newspapers and social media which breaks and damages of cyber security.[1] Some criminally minded hackers are manipulating, stealing, scam, and web-based spying of traditional currency if chances are given to them. Such fraudulent activities are also associated with existing land registry system.

The land registry system is a chain structure consisting of large number of intermediaries linked together to carry out overall process.[2] The land registry system connects multiple entities such as seller, buyer, broker, government authorities. Existing land registry system goes through number of stages to transfer ownership from seller to buyer. This results in delay of overall process.[3,4] In the traditional land registry system, the data are recorded by each entity in a centralized ledger, which is maintained locally. This may lead to tampering of land record data and thus records may be forged easily.[5,6] However, emergence of blockchain technology provides us a suitable approach to solve the above stated problem.

Blockchain has become a buzzword in academic research and industry.[7] The sole inventor of the cryptocurrency bitcoin was Nakamoto.[8] No one having more knowledge of Nakamoto. This may be a person or group of many people. The blockchain approach is a cryptographic-based distributed ledger[9] which empowers trusted trades amid untrusted members in the network. In recent years, blockchain technology is famous throw-out the world and this is increasing substantial attention amongst academicians, investigators, developers, and business experts because of inimitable trust and security features. A big retail business giant Walmart has filed a patent for a stable coin via blockchain which is backed by USD on August 1, 2019.[10]

Company has gained a huge success by the use of blockchain recently. Every industry wants to adopt blockchain technology to secure their personal data and economic property by difficult-to-trace encrypted transmissions. Blockchain transaction can be finished without any intermediary control.[11] Most of the industries are marching toward block chain because of its immutability. The next generation internet systems are springing up toward block chain technology.

To reduce above stated threats on the land registry system,[12] confident unconventional practices should be suggested to reduce the chances of breaches in inflated security. In this article, we proposed a decentralized land registry system based on blockchain technology. The processes involved in land registry system such as land registration by the owner, request made by buyer to view land details, ownership transfer after both the parties agrees for the deal is controlled by smart contracts.

Security, trust, and privacy are always be the most important things people look whenever they start using an application or platform. In the world, there are many platforms[13] which are pruned to fraud, manipulation of content, hacking. All these platforms are centralized and they are always prone to hacking, manipulation and other security, privacy, and trust issues.[14] However, with the use of blockchain technology, the security, privacy can be enhanced exponentially because of its properties like distributed, decentralized, and immutability.[15] This can be solved by using smart contract which is one of the keys in Ethereum platform and thus developing a decentralized platform.[16]

In seeking to explore the blockchain technology and its platforms and how they can be disrupting the current commerce and economy systems. We mainly focused on the issues in the existing land registry system and how they can be solved using blockchain.

In this article, we tend to explain the importance and the benefits of the blockchain technology and its platform especially on the Ethereum, addresses the issues and problems in the current land registry system,[2] point out the importance of smart contracts and how efficiently they can be used to change the economic system and demonstrate, describe and interpret the whole implementation of the land registry system using Ethereum blockchain platform. Finally, a comparative study is done to ensure that the proposed model can eliminates the problems prevailing in the existing land registry system.

So, in this article, we proposed a blockchain-based land registry system. This article is organized as follows. Section 2 surveys about the issue in the existing land registry system, and fundamentals of blockchain. Section 3 deals with preliminaries. In Section 4, the proposed system model is described lucidly. The concrete implementation of proposed model is manifested in Section 5, and the result and experimental evaluation is done in Section 6. Finally, possible future extensions to our work were discussed in Section 7.

2 | RELATED WORK

As it is well known, the existing land registry system involves lot of mediators to put faith in the system.[17] Land registry refers to the process in which ownership and land-related information are recorded by a third-party agency, that is, government. These records provide information related to ownership, facilitate transactions and help to prevent fraud. Existing traditional land registry system delays the process of ownership verification and ownership transfer, slow down transactions, and could even lead to land pilfering. According to the World Bank, 70% of the world's population lacks access to land titles.[17]

All the land records at present are maintained manually by the department which leads to difficulty in access and retrieval of records. This primitive method is expensive and time taking process. At the same time, the process of tracking the owner of the property is challenging and tedious task when there are large numbers of land records to maintain.

Other problems such as counterfeit titles, forged documents are also associated with existing land registry system. Nearly 1700 complaints[18] of fraud land registry have been filed in Tamil Nadu in the past 8 months and even few government officials were involved in this fraudulent activity.

In the past, many research works were done to solve the above stated problem. With the advancement of science and technology such problem can be solved in more efficient way. A paper by Dobhal et al. "Immutability and

Auditability"[19] tells how blockchain feature of data immutability benefits in reducing the number of forged documents in the ecosystem.

Similarly, a paper by Mizrahi 2017 titled "A Blockchain Based Property Ownership Recording System"[20] tells how owner's details can be stored in distribute ledger using blockchain. The system model proposed in this article ensures that data is not tampered by any node present in the network thus helps in achieving security for the system.

Business network today are often inefficient[21] because each participant in the network keeps records or a ledger of all transactions between all the parties that the business interacts with.

The process is expensive because of duplication of effort and intermediaries adding costs for their services. Blockchain can solve this problem as it provides shared ledger technology that allows any participant in the network to see the one system of record.[22] Ledger is defined as the system of records for a business transaction.[23]

Those who are working with cryptocurrency across the borders, by purchasing and vending goods and services, may be gaining a huge profit. This cryptocurrency and blockchain technology remain dignified to disturb occupation finance includes deficiency of exchange rate.[24]

In the recent times, the word "blockchain"[25,26] has become a buzzword and is hailed as one of the most disruptive technology in decades. Blockchain is poised to change IT in much the same way open-source software did a quarter of a century ago. And in the same way that Linux took more than a decade to become a cornerstone in modern application development, blockchain will take years to become a lower cost, more efficient way to share information between open and private networks.

In the white paper published in November 2008, Nakamoto proposed Bitcoin[8] as the first electronic payment system based on a decentralized peer-to-peer network,[27] without the need for a trusted third party. The core technology of this protocol, the blockchain, is widely acknowledged as a major breakthrough in fault-tolerant distributed computing, after decades of research in this field.[14]

The blockchain[8,25] is distributed, decentralized, immutable,[26] and peer-to-peer network[8,27] system where multiple peers can exchange assets securely from all over the world. Reliability and availability are provided by blockchain.[28] A Blockchain is managed by all nodes in the network. The blockchain technology is truly a likely outcome of that the ledger technology advanced in addition to dispersed situations.

After the boom of Bitcoin, many platforms came into existence for the development of blockchain technology and the creation of decentralized applications (DApp). Some of the popular platforms are Ethereum, Hyperledger-fabric, R3 Corda, Ripple, Quorum, IBM Bluemix, and so forth. Ethereum is the most popular and widely used for the development DApp. For the proposed, Ethereum platform is used to build the DApp.

This article is outcome of the literature survey of the previous research work. We discussed the advantages of blockchain-based model over the existing system, proposed a DApp for land registry system based on blockchain technology. Our proposed model solves the problem in the existing land registry system such as counterfeit titles, involvement of brokers, time delays, human error ensure and ensure security.

3 | PRELIMINARIES

In this section, we will discuss the preliminaries that reader should go through and understood well in advance to understand the proposed model and implementation easily. This also helps the readers to understand and analyze that blockchain-based solution is the best approach to solution the stated problem.

3.1 | BLOCKCHAIN

The blockchain is distributed, decentralized, immutable, and peer-to-peer network system where multiple peers can exchange assets securely. The blockchain technology is a likely outcome of that the ledger technology advanced in addition to dispersed situations. The blockchain technology resolves the problem of multiparty trust in disseminated bookkeeping. The blockchain technology is a series of data linked together. This technology is bounding every single transaction by linking with the chain by principles of cryptography in batches by building different blocks. Each block consists of block header and the list of transactions. These blocks are mutually connected with a unique identifier code (also called as hashes) which is connected through previous and the succeeding blocks through some mechanism. This association is greatly similar to the data structure linked list. All the transactions in a block are hashed together to form

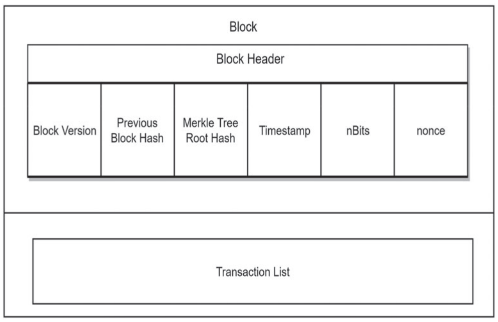


FIGURE 1 Structure of block header

Merkle root hash[26] which is placed in the block header along with some other elements. Figure 1 shows the contents in the block header.

All the contents in the block header[25,26] are hashed to form the block hash which satisfies the constraints set by the network and will be inserted in the block header of next block. If there is a slightest change in any of the transactions, the hash of the block will be changed and the chain will be broken. So, if anyone tries to tamper the data, he has to recalculate the hash of every block which takes a lot of time and computational power, before being noticed by a peer on the network which is almost impossible. This mechanism of tamper-proof makes the blockchain trusted, reliable and secure.

The forthcoming of blockchain technology is full of perspective and opportunities. Due to the advantages and enhanced security, privacy and trust many platforms came into existence for the development of applications based on blockchain. Some of the popular platforms are Ethereum, Hyperledger-fabric, R3 Corda, Ripple, Quorum, IBM Bluemix, and so forth.

3.2 | ETHEREUM

Ethereum was invented by Buterin in late 2013. Although it's working came in existence in 2015. Ethereum[29,30] is an opensource platform built on top of blockchain. Ethereum allows DApps to be deployed and run inside it. It is the most famous Ethereum provides both public and Permissioned configurations. Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of time delay, bowdlerization and fraud or third-party interference. Ethereum networks are used to transfer money and store data.[31] Networks are formed by one or more nodes. Each node may contain a full copy of the blockchain.

Key features of Ethereum are its flexibility, scalability, lower transaction fees. It supports a modified version of Nakamoto consensus via transaction-based[8] state transitions. Ether is a cryptocurrency[32] whose blockchain is generated by the Ethereum platform. Ether can be transferred between accounts and used to compensate participant mining nodes for computations performed. Ethereum provides a decentralized Turing-complete virtual machine, the Ethereum virtual machine (EVM),[29,30,33] which can execute scripts using an international network of public nodes. Every application that is built on Ethereum comprises of a piece of code known as "smart contract." Every node in the network runs the EVM for the execution of the smart contract. EVM is a runtime environment for smart contract, which activates testing. These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.[29,33] These key features make business network more reliable without the involvement of intermediaries.

3.3 | SMART CONTRACT

The term smart contract[11,34] was coined by Nick Sczabo in 1997. Smart contracts are comparable to general contracts in the real world but they are in digital form and are self-executing. The modern economy is driven by contracts. A contract between a shareholder and a company, a contract between tenant and landlord, and so forth. In all the above cases, both the parties trust a third party which is controlled by government, banks, or centralized authorities. Smart contracts are built with ideology of trust in digital world /logic /code rather than relying on trust of a third party.

Smart contract[35] can be considered as a way to automate some of the conditions and obligations described in the contract. In other words, smart contract is a piece of code with embedded business logic. Smart contract define contractual

TABLE 1 Properties of smart contract

Field	Description
Balance	Either owned by any account
Storage	Data storage for the contract
Code	Raw machine code for the contract

conditions under which corporate bond transfer occurs. The sole purpose of the smart contract is to move assets/currency, represent ownership that are stored in it based on some conditions. The developer of the smart contract assimilates a set of rules/logic to handle the state of the asset stored in it. Smart contracts are stored inside the blocks in the blockchain network,[36] due to which they inherit the properties of the blockchain technology, that is, immutable and distributed. So, once a smart contract is written it is almost impossible to tamper it.

Every smart contract has its own address and is useful while sending or receiving the ether.[37] Smart contract can identify the person who calls it, by this privileged access to the contract can be imposed. Every smart contract account has four properties associated with it.[38,39] These four properties are field, balance, storage, and code. Table 1 gives the description of each of these properties.

Smart contracts reside in Ethereum blockchain. Besides tracking the transactions made, smart contracts programs the transactions too.[40] Smart contracts are written by Solidity programming language. It is a statically typed language. Solidity[41] is compiled to bytecode that is executable on the EVM. It is to be noted that execution of mathematical computations of smart contracts incurs "Gas-Cost."

Smart contract file is saved with the extension .sol file. Most of the editors like Atom, VSCode have the support for the solidity language. Smart contract, when compiled gives two objects namely bytecode and ABI.[11,42] On deploying a Smart contract to Ethereum network, the bytecode of smart contract will be stored in the blockchain and it returns the address of the smart contract called contract address.[43] Just with the help of bytecode, it is highly impossible to know the functionality of a smart contract, hence ABI plays a crucial role.

ABI is JavaScript[44] interpretation layer needed to access the bytecode. It is required to interact with the deployed smart contract using address. If any account wants to invoke a function of smart contract it utilizes ABI to hash the function and build EVM bytecode to invoke the function.

3.4 | Web3 library

For creating and deploying the contract on to Ethereum network, Node.js and its modules are used. For compiling a solidity file in node.js a node module "solc" has to be installed on the machine. Using this module, the node.js will compile the solidity file which gives the bytecode and the ABI. To interact with contract on the blockchain, web3[40,45] library is used.To utilize Web3, an object of Web3 class must be instantiated.

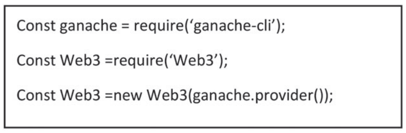
```
const Web3 = require('web3');
const web3 = new Web3(); //Instance
```

Multiple instances of web3 library can be created and each instance is used to connect to a different Ethereum networks. To connect or communicate with an Ethereum network, the instance of web3 requires a communication layer known as provider. There are different types of provider and each can be used to reach out different Ethereum Networks.[46]

Every provider has a set of identical methods to send or receive a request from Ethereum network. These methods allow the web3 library to essentially send a request over to a local network and to receive a response to that request. The provider acts as a medium between the web3 library and the Ethereum network, just like when two persons are talking to each other, the air acts as a medium to transfer the sound waves from one person to another person.

Where the provider = ganache. provider(); //Local Test Network

```
const web3 = new Web3(provider);
```



```
Const ganache = require('ganache-cli');
Const Web3 =require('Web3');
Const Web3 =new Web3(ganache.provider());
```

FIGURE 2 Ganache module

3.5 | GANACHE MODULE

The ganache is a local Ethereum test network which can be installed through the "ganache-cli" npm module. It has to be required or imported into the project to create a local Ethereum test network. Figure 2 shows the code required to import ganache module.

When the contract is deployed in a local network there is no need for security and public and private keys to unlock the accounts that are provided by test network (ganache). The good thing about these accounts is that they are created in an unlocked state and there is no requirement for ether. However, in case of a Ethereum main network or Ethereum public test networks, the public and private keys are required to unlock the accounts and the accounts should have some ether (wei) to deploy the contract.

The main network uses real ether for deployment, manipulation and interaction with contract. On the other hand, the Ethereum test networks (Rinkeby, Kovan, Ropstan) uses fake ether for development and testing purpose.

3.6 | REACT

Once the smart contracts are deployed, they need to be interacted. This interaction is done through web pages. React is used for the development of web pages. For server and routing purposes, Next.js is used. React is JavaScript library developed by Facebook. React is used for fabricating user interface (UI) components.

React provides various functionalities that makes task easy and efficient. React knows how to render multiple components on the screen at one time. We always make one component per file.

3.7 | JSX

JSX is a subset or dialect of JavaScript that allows us to write, what looks like HTML inside of our JavaScript but really behind the scenes just JavaScript. JSX can also be nested inside of itself just like normal HTML.

4 | SYSTEM MODEL

In this section, we proposed a decentralized land registry system based on blockchain technology. The proposed system leverages blockchain benefits by embarking decentralization, data immutability, trust, reducing risks, removing overheads, and cost intermediaries.

The blockchain-based land registry system designed in this article guarantee the overall security of the network. Privacy of the stored data is maintained by ensuring appropriate visibility and authenticating transactions. Blockchain supports data immutability that is once a transaction or a data is recorded in the blockchain, it is nearly impossible to modify it. The system proposed in this article has tamper-proof functionality. Blockchain-based land registry platform eliminates the involvement of intermediaries and append only distributed database system of land record that is shared across the network.

Since, the existing land registry system involves a large number of intermediaries makes the entire process hefty and inefficient. Therefore, the proposed model in this article considers dividing intermediaries involved into manager, owner and buyer to embed all the nodes involved in the land registry system. The system overview is shown in Figure 3.

Each node involved in the network has specific attributes and functionality. Therefore, by understanding the relationship between the nodes present in the chain structure of land registry system, we deployed the smart contract that defines contractual conditions under which transactions occur. Furthermore, in the process of transaction, we fabricate an event

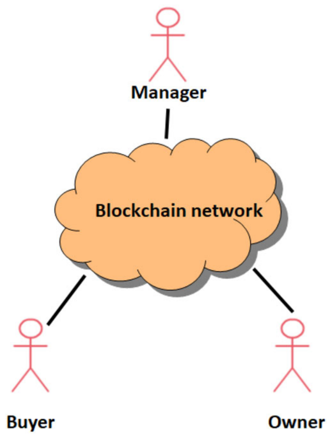


FIGURE 3 System overview

response mechanism to make sure that both the parties of the transaction agree on the terms and conditions for selling and buying land, and finally the transaction along with ownership transfer will be invariably stockpiled in the blockchain.

4.1 | SYSTEM OVERVIEW

As shown in Figure 3, the model proposed in this article mainly consists of manager, owner, and buyer as entities. These entities are connected through a blockchain network. The identity of each entity is recognized through a unique Ethereum address.[24] These entities can be called as nodes of the blockchain network. The functions of these nodes are discussed below.

4.1.1 | Manager

The manager is the person who deploys the smart contract. The manager corresponds to a particular geographic location. The validation of the transaction is done by manager. Smart contract is designed in such a fashion that it ensures manager does not perform any illegal or fraudulent activity.

4.1.2 | Owner

The owner is the person who owns the land. The owner is responsible to complete registration of his/her land on the DApp by providing required land details.

When the buyer requests the owner to show land details, it depends on owner whether to accept or reject request. Once the owner provides all the land details, the land gets registered on the DApp. Using these details and SHA3 algorithm a unique NUMBER is generated for the registered land. The methodology used to generate this NUMBER is discussed in later section of this article.

4.1.3 | Buyer

The buyer is the person who is interested to buy particular land. The buyer makes a authenticate request to owner to show the land records.

4.2 | Smart contract design

In order to understand the land registry process, this article presents two smart contracts, namely land registration contract and transfer contract. In order to link both the contracts we designed, the smart contracts in such a manner that both the contracts contain the addresses of each other. These smart contracts contain embedded business logic, rules of an agreement and stores asset (anything of important value)/currency. Finally, a decentralized application is developed.

The proposed model focuses on development of a simple land registry DApp. This is an alternative to the traditional existing land registry system. In the DApp, the landowner registers the land details and price for which he/she is interested to sell the land. Only the government officer who is marked as the manger validates the registration process. Although the registration process involves a government officer, the entire process is tamper-proof and increases trust in the system by supporting verifiable audit trails.

The smart contract has been deployed in such a way that all the functionalities of the nodes present in the network are being carried out without any inconvenience. The functionality of few function created in "LandRegistration" is discussed below.

Functions ():

1. addManager(): This function is used for adding the manager. This function can only be called by the person who deployed the smart contract. The Ethereum address of the manager and the area allocated to the manager is given as input.
2. register (): This function is used for registering the land on the DApp. This function takes land's details such as plot No., location, district, state, landmark, and owner's Ethereum address along with the price at which owner is interested to sale the land as the input. The concept of mapping is used for sharing the stored details.
3. calculateNumber(): Used to compute a unique NUMBER for a property. Plot No., location, district, and state are provided as input and a unique NUMBER is returned as the output. This function uses keccak256 standards to generate this NUMBER.
4. Owner (): This function is used to show the details of land to the owner. The input is land unique NUMBER and land details along with the Ethereum address of the person who is interested to buy the land. It completely depends on owner whether he/she wants to accept or reject the request sent by the buyer.
5. Buyer (): This function is used to view the details of land for any interested buyer. Input is land unique NUMBER and output is the current owner of the property, market value, availability of the land to buy and request status.
6. requstToLandOwner(): This function is used to send a request to the owner to buy the land. Land unique NUMBER is provided as the input and buyer's details is sent to the owner. This function has "restricted" attached to it as a modifier which indicates that the control will first go to the restricted modifier we have defined and execute. So, it will verify whether the request is being created by the buyer or not.
7. viewRequest(): Function to view the address of the buyer for a particular land, if there are any. Input is the property unique NUMBER and the output is the address of the buyer who is interested to buy the land.
8. purchaseLand(): In case the owner agrees to sell land, then transaction need to occur between both the parties. The land unique NUMBER is provided as the input and the transaction of ether corresponding to the market value of the land is transferred to the owner and finally transfer of land ownership occurs.

The transaction is secured, verified, and authenticated by the manager of the smart contract. Finally, the buyer and seller sign an agreement and this agreement is stored in blockchain. Government officer marked as manager also captures and uploads the photograph of seller and buyer on DApp to ensure authenticity.

Figure 4 depicts the process of event response mechanism, that is, how each event in the land registry system is triggered by the deployed smart contract and finally in case both the parties agree for the deal, the transaction is made. It is described below.

1. First, the owner registers the land on the DApp. Smart contract triggers the **`"Request ()"`** event. The owner's Ethereum account public key along with the signature and account private key is used to validate the authenticity of the registration.
2. The buyer queries the land details using unique NUMBER generated for registered land and smart contract triggers **`"createRequestToOwner()"`** event. The owner gets notification along with buyer's Ethereum address.
3. Next, in case the owner agrees to sell the land to the buyer, an event **`"purchaseLand()"`** is triggered.
4. Finally, **`"ValidateTransaction"`** event is triggered and ownership is transferred from owner to buyer. Manager of the deployed smart contract validate the transaction. All these transactions are stored as immutable records in blockchain-based decentralized ledger.

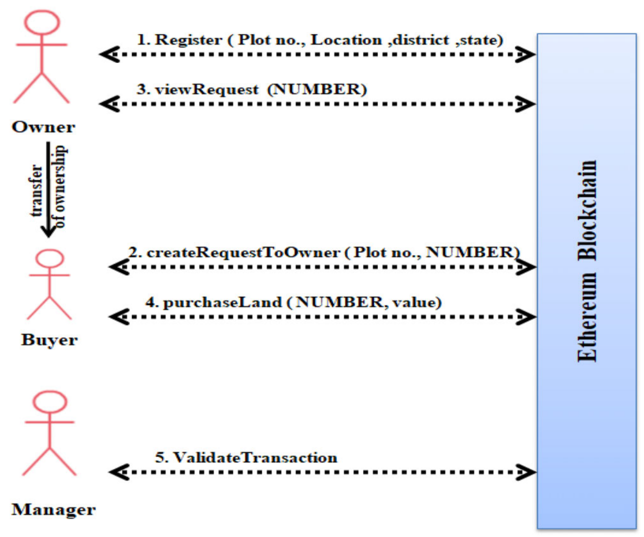

Owner
Buyer
Manager
transfer of ownership
1. Register (Plot no., Location ,district ,state)
3. viewRequest (NUMBER)
2. createRequestToOwner (Plot no., NUMBER)
4. purchaseLand (NUMBER, value)
5. ValidateTransaction
Ethereum Blockchain

FIGURE 4 Event response mechanism

5 | IMPLEMENTATIONS

In this section, the implementation of the proposed model is explained lucidly. Design and Analyses were carried out by utilizing the smart contracts, web3 library, node.js and infura api in the Rinkeby test network of Ethereum.

In this article, for creating and deploying the contract on to Ethereum network, Node.js and its modules are used. For compiling a solidity file in node.js, a node module “solc” has to be installed on the machine. Using this module, the node.js will compile the solidity file which gives the bytecode and the ABI. To interact with contract on the blockchain, web3 library is used. To utilize Web3, an object of Web3 class must be instantiated.

We deployed the smart contracts in Ethereum Rinkeby test network. Smart contract file is saved with the extension .sol file. Most of the editors like Atom, VSCode have the support for the solidity language. We developed our smart contract on VSCode IDE by installing suitable plugins. Smart contract, when compiled gives two objects namely bytecode and ABI. On deploying a Smart contract to Ethereum network, the bytecode of smart contract will be stored in the blockchain and it returns the address of the smart contract called contract address. Just with the help of bytecode, it is highly impossible to know the functionality of a smart contract; hence, ABI plays a crucial role. It’s only the bytecode that is feed to Rinkeby test network for deploying and testing purpose of the smart contract.

The focus of this work is to design and deploy of smart contract in the Ethereum test network using web3 library and Metamask wallet which is Ethereum client on the system. The web3 library[47,48] is used to connect to the Ethereum network from an application.

To deploy a smart contract, an account in Ethereum client like Metamask[49] or Mist Browser is required with some amount of ether. In the project, Metamask is used as Ethereum client. Metamask is a web browser extension. When an account is created in the Metamask, a 12-word mnemonic is displayed. A 12-word mnemonic is 12 random words which is used to generate (uses BIP39 algorithm) a series of accounts, each with a public key, private key and an account address. The Rinkeby faucet (https://faucet.Rinkeby.io) can be used to get some ether into the Metamask account. Figure 5 shows a Metamask account with some Ethers, which we get from above mentioned Rinkeby faucet.

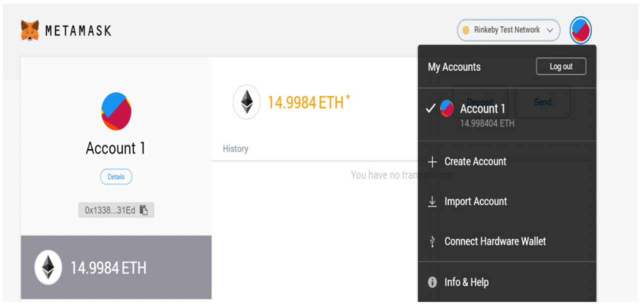
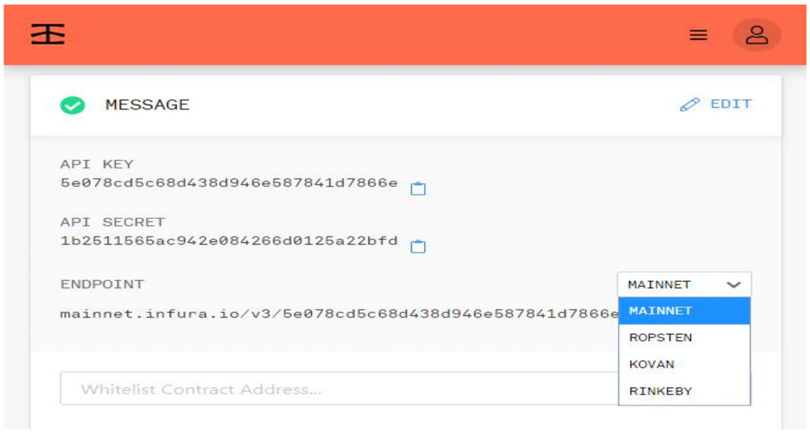


FIGURE 5 Metamask


MESSAGE

API KEY
5e078cd5c68d438d946e587841d7866e

API SECRET
1b2511565ac942e084266d0125a22bfd

ENDPOINT
mainnet.infura.io/v3/5e078cd5c68d438d946e587841d7866e

MAINNET

FIGURE 6 Infura interface

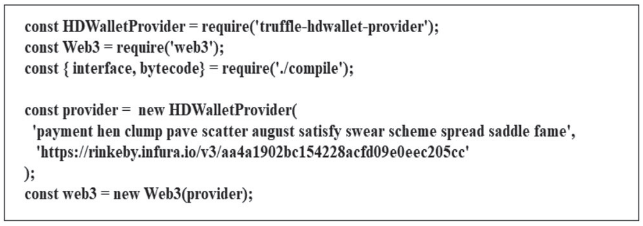

```
const HDWalletProvider = require('truffle-hdwallet-provider');
const Web3 = require('web3');
const { interface, bytecode} = require('./compile');

const provider =  new HDWalletProvider(
  'payment hen clump pave scatter august satisfy swear scheme spread saddle fame',
   'https://rinkeby.infura.io/v3/aa4a1902bc154228acfd09e0eec205cc'
);
const web3 = new Web3(provider);
```

FIGURE 7 Truffle-hdwallet-provider

The mnemonic is passed as an argument to the provider to unlock the accounts in the Metamask. To deploy the contract in public networks, a connection with a node on the public network is required.

To become a node in Ethereum network, one needs to configure their machine, which is a little tedious job and requires a lot of time and effort. So, instead a service called Infura[50,51] is used to connect to a node already configured and existing on the network. Infura is a public API which gives us the ability to access a node on the network.

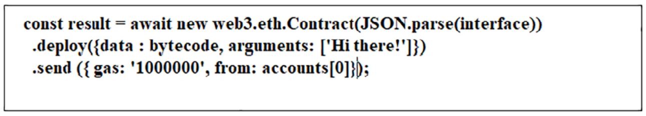
The Infura has the nodes on all the Ethereum networks including the Main network as shown in Figure 6. Get the api key and the link to connect to the Rinkeby network from https://infura.io.

As said above, a provider is needed to communicate with the Ethereum test network. To connect or communicate with an Ethereum network, the instance of web3 requires a communication layer known as provider. The provider acts as a medium between the web3 library and the Ethereum network. The provider has to be setup manually specifying the network to connect and the account mnemonic to unlock the accounts. For this purpose, in our implementation we use a node module "truffle-hdwallet-provider." Truffle is a command line tool for contract creation. It is kind of one shot for development of Ethereum contract. Truffle is going to allow us to connect the Rinkeby network hosted through Infura and more importantly allow us to unlock and use accounts very easily.

The provider has to be setup manually specifying the network to connect and the account mnemonic to unlock the accounts. For this purpose, a node module "truffle-hdwallet-provider" should be installed and imported into the project, shown in Figure 7.

As mentioned, when a smart contract is compiled by a solidity compiler, it generates a bytecode and ABI, which are required in the deployment of the contract on to the Rinkeby network. To interact with the Ethereum network, the web3 instance is used.

The web3 library has many modules inside it which are used to work with different types of networks. Here, "eth" is an Ethereum module in web3 which has a method "getAccounts()." The await key word is used since it is an asynchronous method.After getting the accounts, the contract can be deployed using one of the accounts. The contract deployment script,


```
const result = await new web3.eth.Contract(JSON.parse(interface))
  .deploy({data : bytecode, arguments: ['Hi there!']})
  .send ({ gas: '1000000', from: accounts[0]});
```

The “Contract” in the above code is a constructor in the Ethereum module for creating a new contract, which has “interface (ABI)” as an argument. The “deploy ()” is a method in Contract which has “bytecode and arguments: []” as the arguments. The arguments array contains all initial values or messages required for the contract. The “send ()” method takes two arguments. One the gas limit and the other is the account address from which the contract is deployed. The send method will actually send this transaction to the network and the returns the address with many other parameters to the result variable. The contract is deployed to Rinkeby Test network and the address value can be known from result.options.address.

Now the contract is deployed, a front-end app or a webpage should be created which can interact with the deployed contract. For this, the contract address and the ABI should be imported to the app. Whenever the Metamask is running inside the browser, it will automatically inject an instance of web3 library into the current active page. This web3 also has a provider which is connected to the public networks available in the Metamask.

After deployment of the smart contracts namely “land registration” and “transfer,” a front-end app or a webpage was created which can interact with the deployed contracts. For this, the contract address and the ABI should be imported to the app. Whenever the Metamask is running inside the browser, it will automatically inject an instance of web3 library into the current active page.

5.1 | DESIGNING SMART CONTRACT

Smart contracts are designed in such a manner that they manage all the tasks related to land registry system. In the proposed model, we assumed there are only three kinds of people on the network, the manager, the owner who owns the property and the buyer who is interested to buy/purchase a particular land.

The proposed land registry system consists of two smart contracts namely LandRegistration and Transfer.

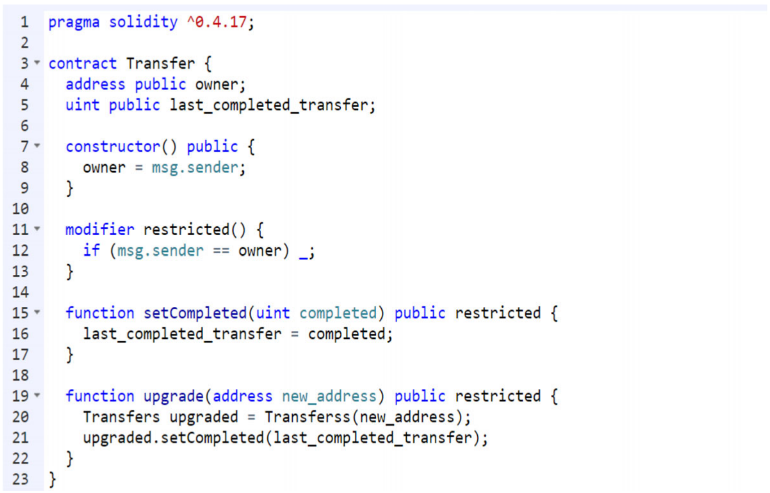
5.2 | Transfer contract

Figure 8 shows the code involved in transfer contract.

The method setCompleted has an argument completed of type “uint.” “public” is a key word which indicates the access modifier or who can access this function. The variable “msg” is struct type which is automatically invoked when a transaction occurs or when a function is called and stores the details like the address of the person who invoked the transaction, the amount of ether he is sending with it, and so forth.

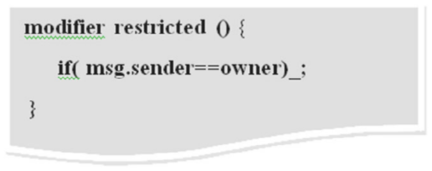
In solidity, there two types of functions:

1. The function which modifies the values in the contract or changes its state which is also known as transaction.
2. The function which does not change any value in the contract or changes its state which is also known as “call” to a function. To represent this type of function a keyword “view” or “pure” is attached in the function declaration.



```
pragma solidity ^0.4.17;

contract Transfer {
  address public owner;
  uint public last_completed_transfer;

  constructor() public {
    owner = msg.sender;
  }

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_transfer = completed;
  }

  function upgrade(address new_address) public restricted {
    Transfers upgraded = Transferss(new_address);
    upgraded.setCompleted(last_completed_transfer);
  }
}
```

FIGURE 8 Transfer contract



```
modifier restricted () {
    if( msg.sender==owner)_;
}
```

FIGURE 9 Modifier restricted()

Modifier restricted (): Some functions like creating a request and finalizing request should only be done by the manager. So, these kinds of functions will have a similar code of checking whether the function is invoked by the manager or not.

This verification can be kept in a modifier restricted function as shown in Figure 9. The restricted should be attached to the functions which will use this modifier. The “require” will check whether the condition given is true or false. If it is true the function will execute or else it will not be executed. The symbol “_,” represent that there is a below it that still needs to be executed.

The functions with access modifier as “public” indicate that it can be accessed by anyone on the platform.

5.3 | LandRegistration contract

The LandRegistration contract consists of various functions. Each function is assigned for a particular process/task, which is necessary in the land registry system.

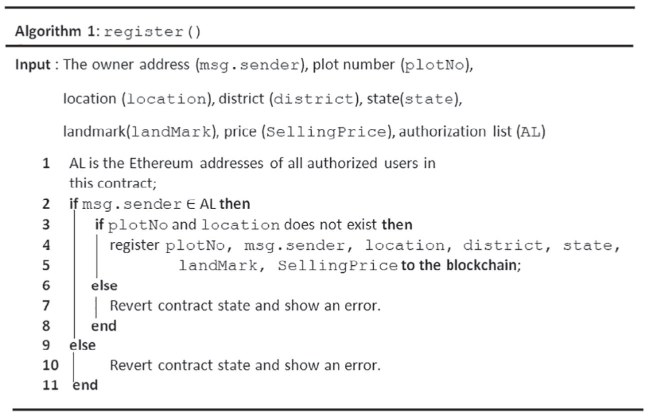
The very first step is registration of land by the owner on the DApp. The LandRegistration smart contract take cares of this process very effectively. Figure 10 gives the pseudo-code implementation of the **`register ()`** function. This function is used for registering the land on the DApp. This function takes land’s details such as plotNo, location, district, state, landmark, and owner’s Ethereum address along with the price at which owner is interested to sale the land as the input. The concept of mapping is used for sharing the stored details.

The next task is to generate a unique number for the land registered on the DApp. Later in the process land registry system, this unique number will be used to complete many other transactions. Figure 11 gives the pseudo-code implementation of the **`computerNumber()`** function. The functions keccak256 standards to complete the task.

5.3.1 | Methodology used to generate unique number to land register on DApp

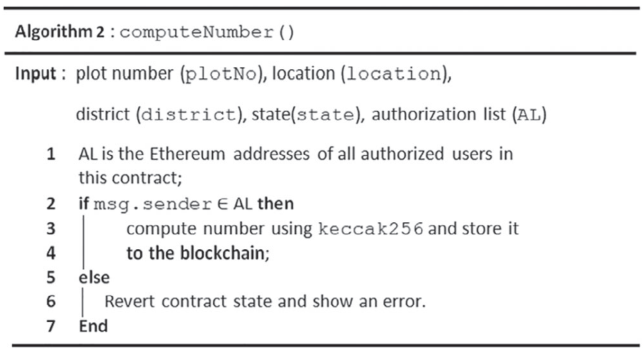
We are going to take the current block difficulty, current time, `plot No., location, district`, and `state`. Current block difficulty takes some amount of time to process an actual transaction, the amount of time that it takes to pick or solve a block or close out a block is referred to as the block difficulty. This is represented as an integer. These inputs are feed them into the SHA3 algorithm, and it spits out a really big hexadecimal number, illustrated in Figure 12. The generated big hexadecimal number will be used to compute unique number for the registered land.

The code for `computeNumber()` function is shown in Figure 13.



```
Algorithm 1: register()
Input : The owner address (msg.sender), plot number (plotNo),
        location (location), district (district), state(state),
        landmark(landMark), price (SellingPrice), authorization list (AL)
AL is the Ethereum addresses of all authorized users in
this contract;
if msg.sender ∈ AL then
    if plotNo and location does not exist then
        register plotNo, msg.sender, location, district, state,
            landMark, SellingPrice to the blockchain;
    else
        Revert contract state and show an error.
    end
else
    Revert contract state and show an error.
end
```

FIGURE 10 Register() function algorithm



```
Algorithm 2 : computeNumber()
Input : plot number (plotNo), location (location),
        district (district), state(state), authorization list (AL)
1  AL is the Ethereum addresses of all authorized users in
   this contract;
2  if msg.sender ∈ AL then
3      compute number using keccak256 and store it
4      to the blockchain;
5  else
6      Revert contract state and show an error.
7  End
```

FIGURE 11 ComputeNumber() function algorithm

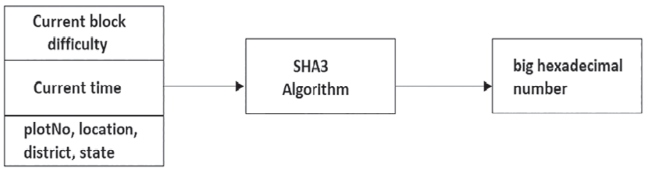
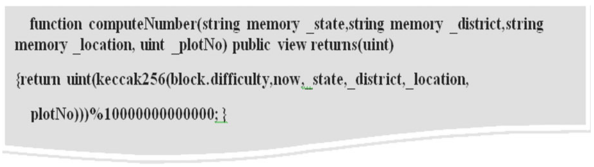


FIGURE 12 Method for unique NUMBER generation



```
function computeNumber(string memory _state,string memory _district,string
memory _location, uint _plotNo) public view returns(uint)
{return uint(keccak256(block.difficulty,now,_state,_district,_location,
plotNo)))%10000000000000;}
```

FIGURE 13 ComputeNumber function

The function **computeNumber()** is a “public” function because anyone can view the content of this function. The function is mark as a “view” type because we are not modifying any state or any data in the contract. The only goal of this function is to return a random number.

The block variable is a global variable that we have access to at any given time. Difficulty will be a number that indicates how challenging it is going to be to seal or solve the current block. Now inside the random function we do have to make sure that we return an unsigned int but hashing algorithm and it returns a hash. So, we can convert this hash to an integer.

Now, let us have a look at the entire LandRegistration smart contract for our better understanding.

There are various functions in LandRegistration smart contracts that takes cares of tasks like sending request by the buyer to the owner to show land details, viewing land details, processing request for the land by accepting or rejecting, availing land for sale, buying the approved property, removing the ownership of various owner once the deal is finalized, validating of the transaction by the manager.

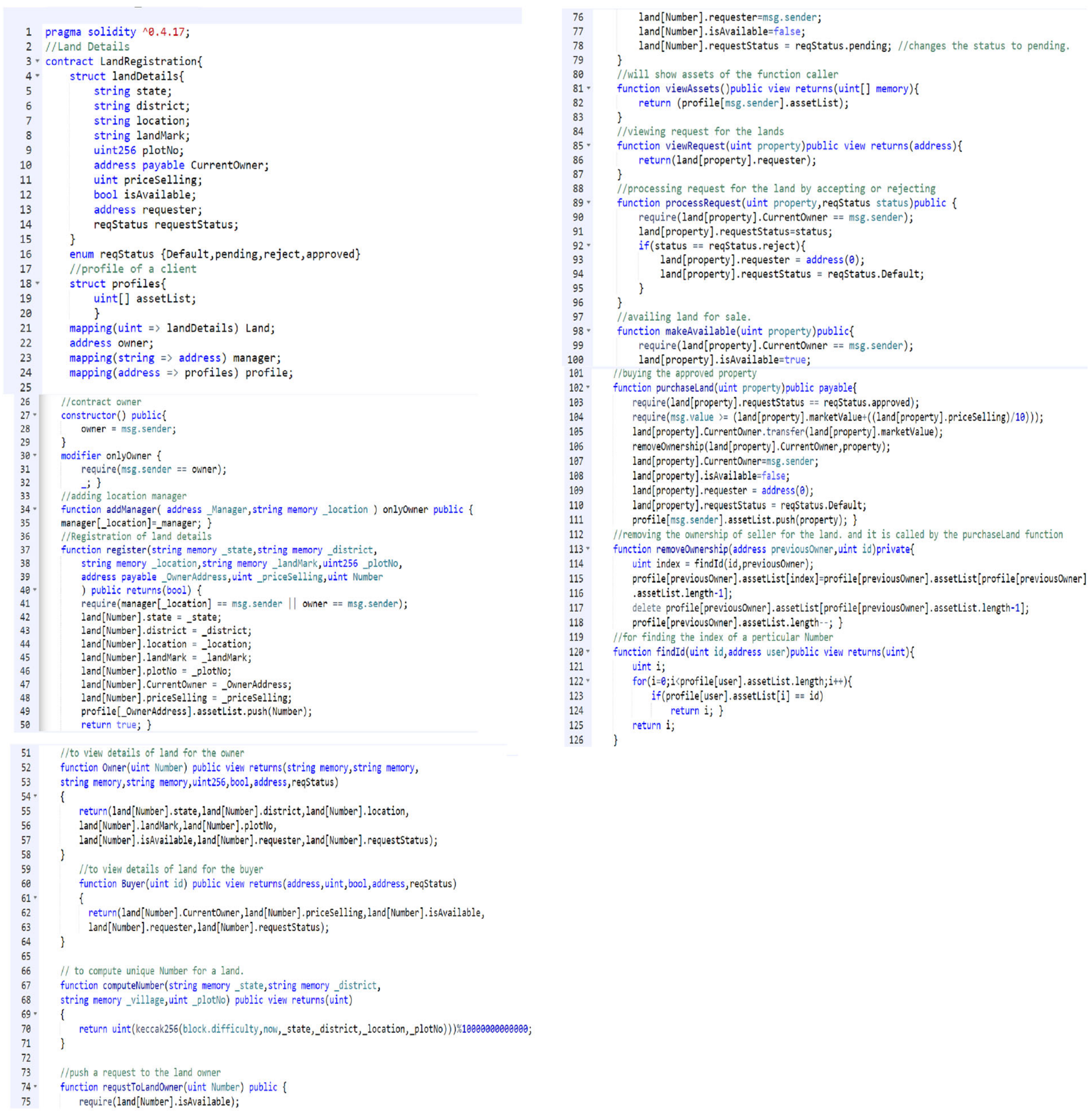
The sequence how smart contracts interacts with three entities present in the network is shown in Figure 14.

5.4 | Creating the front-end using react

The essential concept in DApp development is writing the smart contract and then interacting with deployed smart contracts through frontend web pages.

After deploying the contract, the address and the ABI of the contract will be stored in a file. The ABI and the address are used to call or invoke a function and sent a transaction. For the model proposed in this article, React is used for the development of web page. For server and routing purposes, Next.js is used. Next.js is a module used for server and routing purposes. By using, the JavaScript will render in the server side and also on the client side, that is, web browser. It wraps all the react and renders it in the browser. Semantic-ui-react is used for the styling of the web page.

In ReactJS, the “pages” and “components” folder consist of all the files for the creation of website. The components folder consists of component files which are required by the files in pages folder. This also requires a server.js and routes.js files for server and routing purposes. A component is a function or an object that returns some amount of HTML. So, we can have different purposes within our application.



```
pragma solidity ^0.4.17;
//Land Details
contract LandRegistration{
    struct landDetails{
        string state;
        string district;
        string location;
        string landMark;
        uint256 plotNo;
        address payable CurrentOwner;
        uint priceSelling;
        bool isAvailable;
        address requester;
        reqStatus requestStatus;
    }
    enum reqStatus {Default,pending,reject,approved}
    //profile of a client
    struct profiles{
        uint[] assetList;
        }
    mapping(uint => landDetails) Land;
    address owner;
    mapping(string => address) manager;
    mapping(address => profiles) profile;

    //contract owner
    constructor() public{
        owner = msg.sender;
    }
    modifier onlyOwner {
        require(msg.sender == owner);
        _; }
    //adding location manager
    function addManager( address _Manager,string memory _location ) onlyOwner public {
    manager[_location]=_manager; }
    //Registration of land details
    function register(string memory _state,string memory _district,
        string memory _location,string memory _landMark,uint256 _plotNo,
        address payable _OwnerAddress,uint _priceSelling,uint Number
        ) public returns(bool) {
        require(manager[_location] == msg.sender || owner == msg.sender);
        land[Number].state = _state;
        land[Number].district = _district;
        land[Number].location = _location;
        land[Number].landMark = _landMark;
        land[Number].plotNo = _plotNo;
        land[Number].CurrentOwner = _OwnerAddress;
        land[Number].priceSelling = _priceSelling;
        profile[_OwnerAddress].assetList.push(Number);
        return true; }

    //to view details of land for the owner
    function Owner(uint Number) public view returns(string memory,string memory,
    string memory,string memory,uint256,bool,address,reqStatus)
    {
        return(land[Number].state,land[Number].district,land[Number].location,
        land[Number].landMark,land[Number].plotNo,
        land[Number].isAvailable,land[Number].requester,land[Number].requestStatus);
    }
        //to view details of land for the buyer
        function Buyer(uint id) public view returns(address,uint,bool,address,reqStatus)
        {
          return(land[Number].CurrentOwner,land[Number].priceSelling,land[Number].isAvailable,
          land[Number].requester,land[Number].requestStatus);
    }

    // to compute unique Number for a land.
    function computeNumber(string memory _state,string memory _district,
    string memory _village,uint _plotNo) public view returns(uint)
    {
        return uint(keccak256(block.difficulty,now,_state,_district,_location,_plotNo))%10000000000000;
    }

    //push a request to the land owner
    function requstToLandOwner(uint Number) public {
        require(land[Number].isAvailable);
        land[Number].requester=msg.sender;
        land[Number].isAvailable=false;
        land[Number].requestStatus = reqStatus.pending; //changes the status to pending.
    }
    //will show assets of the function caller
    function viewAssets()public view returns(uint[] memory){
        return (profile[msg.sender].assetList);
    }
    //viewing request for the lands
    function viewRequest(uint property)public view returns(address){
        return(land[property].requester);
    }
    //processing request for the land by accepting or rejecting
    function processRequest(uint property,reqStatus status)public {
        require(land[property].CurrentOwner == msg.sender);
        land[property].requestStatus=status;
        if(status == reqStatus.reject){
            land[property].requester = address(0);
            land[property].requestStatus = reqStatus.Default;
        }
    }
    //availing land for sale.
    function makeAvailable(uint property)public{
        require(land[property].CurrentOwner == msg.sender);
        land[property].isAvailable=true;
    //buying the approved property
    function purchaseLand(uint property)public payable{
        require(land[property].requestStatus == reqStatus.approved);
        require(msg.value >= (land[property].marketValue+((land[property].priceSelling)/10)));
        land[property].CurrentOwner.transfer(land[property].marketValue);
        removeOwnership(land[property].CurrentOwner,property);
        land[property].CurrentOwner=msg.sender;
        land[property].isAvailable=false;
        land[property].requester = address(0);
        land[property].requestStatus = reqStatus.Default;
        profile[msg.sender].assetList.push(property); }
    //removing the ownership of seller for the land. and it is called by the purchaseLand function
    function removeOwnership(address previousOwner,uint id)private{
        uint index = findId(id,previousOwner);
        profile[previousOwner].assetList[index]=profile[previousOwner].assetList[profile[previousOwner]
        .assetList.length-1];
        delete profile[previousOwner].assetList[profile[previousOwner].assetList.length-1];
        profile[previousOwner].assetList.length--; }
    //for finding the index of a perticular Number
    function findId(uint id,address user)public view returns(uint){
        uint i;
        for(i=0;i<profile[user].assetList.length;i++){
            if(profile[user].assetList[i] == id)
                return i; }
        return i;
    }
```

FIGURE 14 continued

Although the deployed smart contract contains many functions, the UI is developed in such a way that it makes entire land registry process easier and efficient without any time delay.

5.5 | Architecture of Next.js

Next.js is a module used for server and routing purposes.[52,53] By using, the JavaScript will render in the server side and also on the client side, that is, web browser. It wraps all the react and renders it in the browser. Figure 15, depicts the features and advantages of next.

When we use next, the default port of server is 3000. The next will renders all the js files in the folder pages. When the server starts, the next will search for the index.js file in the pages folder. So, in the pages folder, we are going to create all the required JavaScript files for the interaction with the contract.

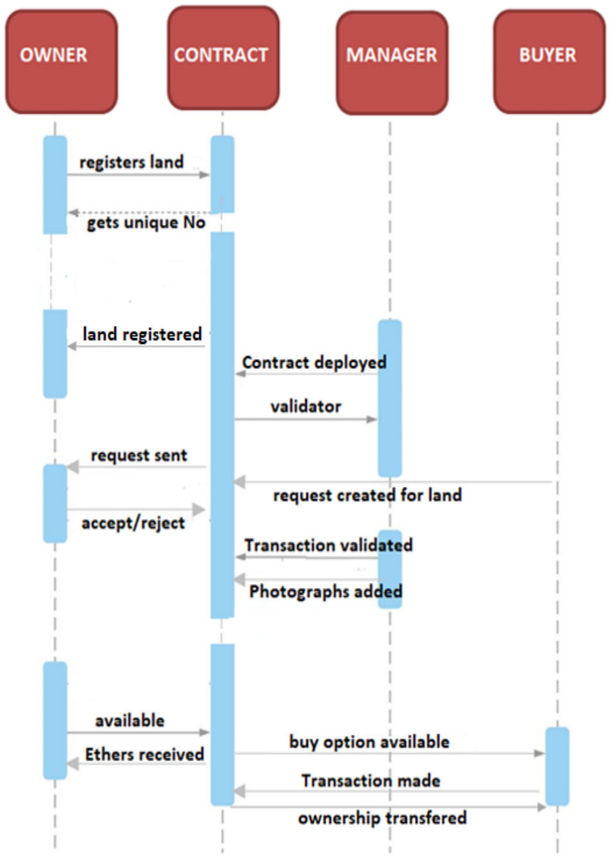


FIGURE 14 Sequence diagram

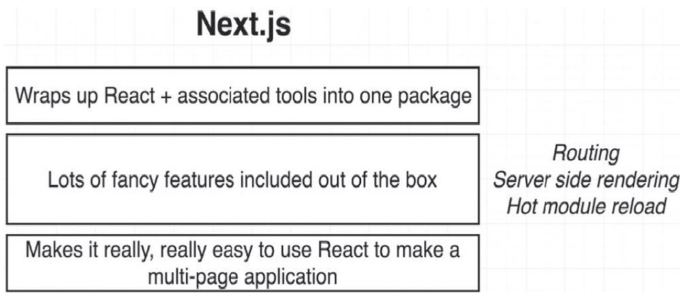

Next.js
Wraps up React + associated tools into one package
Lots of fancy features included out of the box
Routing
Server side rendering
Hot module reload
Makes it really, really easy to use React to make a multi-page application

FIGURE 15 Features of Next.js

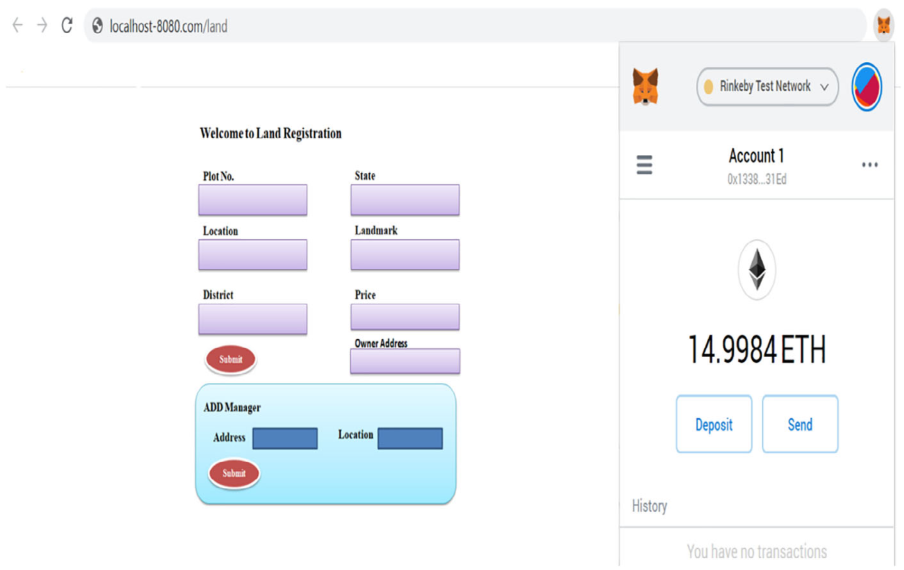
6 | RESULT AND EXPERIMENTAL EVALUATION

In order to interact with deployed smart contracts, we designed webpage for UI. This section deals with outcome of smart contract and how users interact with it and finally experimental evaluation is done by comparing existing land registry system with the proposed model.

When the user runs the DApp, the registration page is made accessible, as shown in Figure 16. The webpage gives options to register the land and to register the manager. The browser must have Metamask wallet installed, it is a necessary for the website to work as it meant to be.

As shown in Figure 16, the owner needs to enter the land details in order to register the land on the DApp. The smart contract designed ensures to store data in immutable record. So, it ensures that no one can tamper the records. "Owner address," that is, 0x13380Ec58e617BFF46C0207D99d9D2C8138C31Ed is the Ethereum address of the owner. Figure shows a Metamask account with some Ethers. For all the transactions some amount of Ethers are needed. For testing purpose, we incurred these Ethers from https://faucet.Rinkeby.io.

The Rinbeby faucet can be used to get some ether into the Metamask account.


localhost-8080.com/land

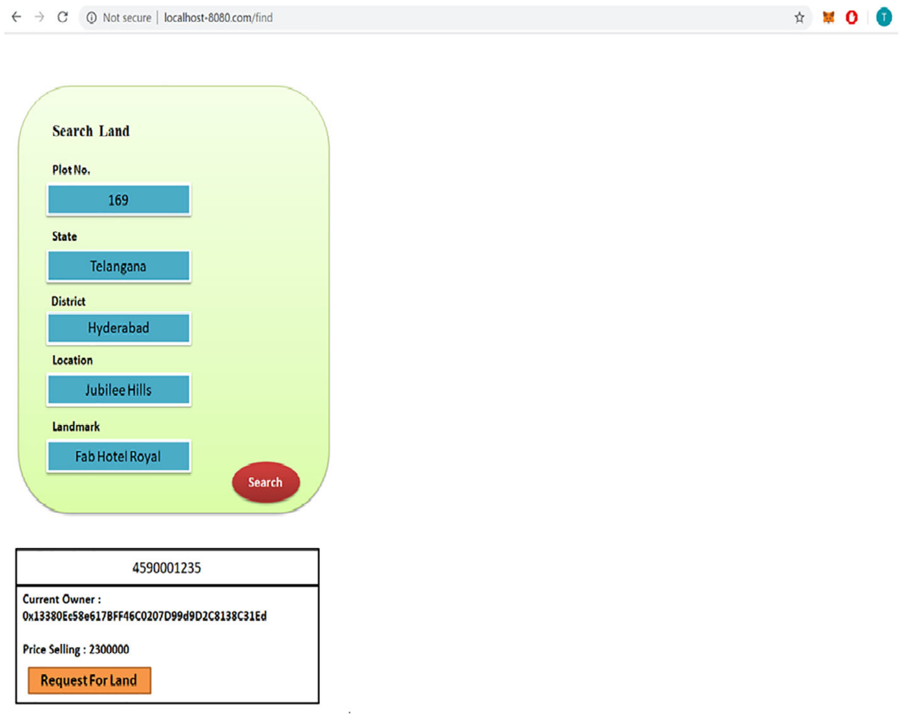
Welcome to Land Registration

Plot No. State
Location Landmark
District Price
Owner Address
Submit

ADD Manager
Address Location
Submit

Rinkeby Test Network
Account 1
0x1338...31Ed
14.9984 ETH
Deposit Send
History
You have no transactions

FIGURE 16 Home page


Not secure | localhost-8080.com/find

Search Land

Plot No.	169
State	Telangana
District	Hyderabad
Location	Jubilee Hills
Landmark	Fab Hotel Royal

Search

4590001235

Current Owner : 0x13380Ec58e617BFF46C0207D99d9D2C8138C31Ed

Price Selling : 2300000

Request For Land

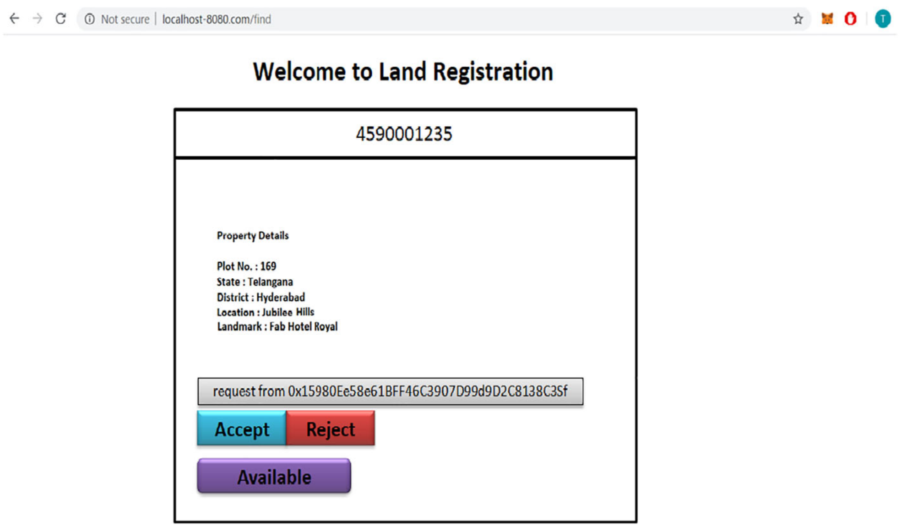
FIGURE 17 Search land/request for land

Once the land is registered on the DApp, any buyer can search the land by entering details and can send request to the owner of the land, if he is interested to purchase it. Figure 17 shows the webpage to search the land by entering land details.

Based upon the details entered, an output table is generated as shown in the figure. This table consists of a unique number, that is, “4 590 001 235” generated using SHA3 algorithm. The methodology used to generate this number is discussed in Section 5 of this article. Ethereum address of the current owner and price of the land is also displayed in the output.

If the buyer is interested to purchase the land, he/she can click on “Request for Land” option to send request to owner.

After the request is send by the buyer. The owner can view the request as shown in Figure 18. The request contains the Ethereum address of the interested buyer who makes the request. Here the buyer Ethereum address is “0x15980Ee58e61BFF46C3907D99d9D2C8138C3Sf.” The webpage gives the option to either “Accept” or “Reject” the request. In case the owner accepts the request, he/she also clicks on “Available” option to show land record to the buyer.



Welcome to Land Registration

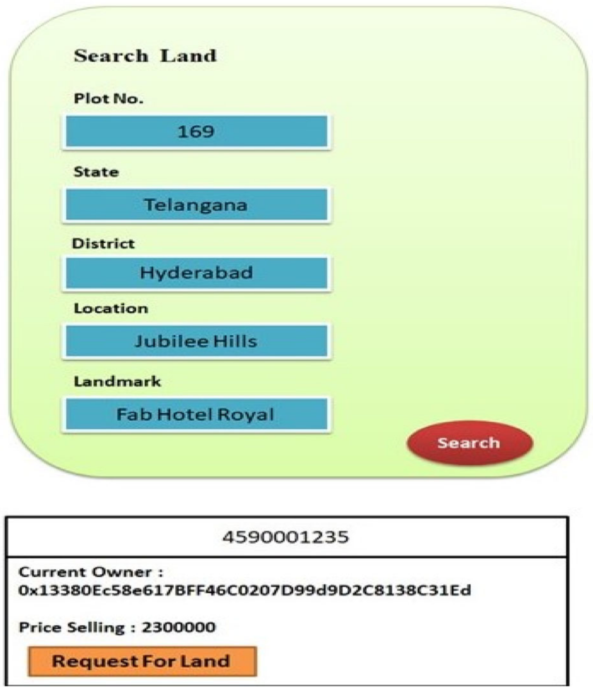
4590001235

Property Details

Plot No. : 169
State : Telangana
District : Hyderabad
Location : Jubilee Hills
Landmark : Fab Hotel Royal

request from 0x15980Ee58e61BFF46C3907D99d9D2C8138C3Sf

FIGURE 18 Owner notified


Search Land

Plot No.: 169
State: Telangana
District: Hyderabad
Location: Jubilee Hills
Landmark: Fab Hotel Royal

4590001235

Current Owner :
0x13380Ec58e617BFF46C0207D99d9D2C8138C31Ed

Price Selling : 2300000

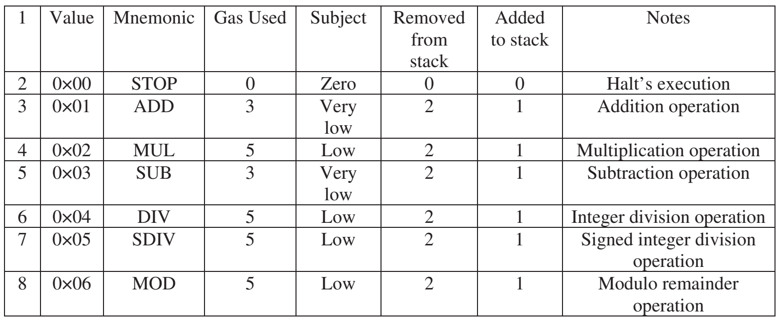
FIGURE 19 Option to buy land

After the owner accepts the request of the buyer. The smart contract is triggered and buyer is notified regarding this. Now, the buyer gets the option to buy the land, as shown in Figure 19. The buyer can click on "Buy" option to purchase the land and corresponding ethers are deducted from the wallet the transaction. Once the transaction is complete, the smart contract automatically transfers the ownership from current owner to the new buyer.

The designed UI clearly shows that the proposed model is efficient and easy to interact in order to carry out all steps involved in land registry system.

6.1 | Experimental evaluation

Solidity programming language is used to write smart contracts. Solidity is compiled to bytecode that is executable on the EVM. Syntax of solidity is similar to ECMAScript. From Figure 20, it is inferred that execution of mathematical computations of smart contracts incurs "Gas-Cost."



1	Value	Mnemonic	Gas Used	Subject	Removed from stack	Added to stack	Notes
2	0×00	STOP	0	Zero	0	0	Halt's execution
3	0×01	ADD	3	Very low	2	1	Addition operation
4	0×02	MUL	5	Low	2	1	Multiplication operation
5	0×03	SUB	3	Very low	2	1	Subtraction operation
6	0×04	DIV	5	Low	2	1	Integer division operation
7	0×05	SDIV	5	Low	2	1	Signed integer division operation
8	0×06	MOD	5	Low	2	1	Modulo remainder operation

FIGURE 20 Gas cost for computation of smart contract

TABLE 2 Comparison between existing and proposed model

Parameter	Existing model	Proposed model
Decentralization	No	Yes
Intermediary cost	High	Low
Scalability	Low	High
Security	Low	High
Tile involved in ownership transfer	High	Low
Accountability	Low	High

The author of smart contract must specify the gas limit so that the validators can decide to mine it before validating the smart contract. If the reward of the smart contract is satisfactory then miner chooses to invoke smart contract function call in their next block. Every smart contract has its own address and is useful while sending or receiving the ether. Smart contracts can identify the person who calls it; by this, we can impose the privileged access to contract.

Finally, we choose certain parameters to compare our proposed model with the existing land registry system. These parameters are decentralization, intermediary cost, scalability, security, time involved in ownership transfer and accountability. The comparison results are shown in Table 2.

The above listed parameters are discussed below.

6.1.1 | Decentralization

We instigated the conviction of blockchain into the existing land registry system to annihilate the chances of tampering with records by entities. Also, since blockchain is shared ledger technology it allows any participant in the network to see the one system of records. This degree of decentralization is quite low in existing land registry system.

6.1.2 | Intermediary cost

Existing land registry system involves lot of mediators to put faith into the system. Blockchain technology reduces the number of intermediaries involved in transferring of asset from one user to another. This results in reducing the intermediary cost. The proposed model eliminates the role of brokers in selling and buying of land and thus commission involved. Also cost of registering property is high, while registering a property transaction, the buyer has to pay a registration fee along with stamp duty. Stamp duty rates across states vary between 4% and 10%. This percentage varies from state to state. So, the intermediary cost in the proposed model is low.

6.1.3 | Security

Entire land records are maintained through distributed ledger. The land records are secured and not any can access it easily. One needs to seek permission of the owner to view land record. In this sense, blockchain-based model ensures high security.

6.1.4 | Scalability

The system is designed in such a fashion that it can easily handle more number of land registrations on the DApp. The functions in the smart contracts are efficient enough to handle scalable issues. The scalability of the proposed model is high.

6.1.5 | Time involved in ownership transfer

Each and every step involved in the land registry system is automated through smart contracts. There is no time delay in transfer of ownership from current owner to new buyer of the land. This task is completed within seconds by just one click. So, time involved in ownership transfer is low.

6.1.6 | Accountability

The proposed model lacks centralized authority to settle any disputes. However, the chances of disputes are negligible. The transaction is validated by the manager. So the accountability of the proposed model is high enough.

We analyzed that our proposed model is efficient enough to solve existing problem in a user-friendly way.

7 | CONCLUSION AND FUTURE SCOPE

The main goal of this article is to research the blockchain technology and how it can be applied to solve the problems in the existing land registry system. In the paper, a brief description about the blockchain has been given lucidly. Further, the paper focuses on the blockchain-based model for land registry system and emulates on the Ethereum platform for developing the DApp. Then, the paper talks about how these problems can be solved in an efficient way using Ethereum platform and discussed the whole development, deployment and interaction part in a profound manner.

Such blockchain-based model can be used in different sectors like healthcare, supply chain management, trading, e-commerce, and so forth, to eradicate existing problem through technical path.

Using the blockchain technology, the security, privacy and the trust can be enhanced exponentially. Blockchain is a disrupting technology and in the very future it will change whole economic and commerce systems. Blockchain technology is having the perspective of being the next major disruption. The blockchain has the latent to reform how worth is traded.

The future of blockchain technology is full of perspective and opportunities. Many researchers believe that blockchain accompanied with artificial intelligence can solve existing problems in economy and commerce in more effective manner.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author.

ORCID

Sandeep Kumar Panda https://orcid.org/0000-0002-0752-4267
Sachi Nandan Mohanty https://orcid.org/0000-0002-4939-0797

REFERENCES

1. Krishnan A. *Blockchain Projects that Will Change the Real Estate Industry*. India: Invest in Blockchain; 2018.
2. Updating the Land Registration Act 2002, @ Crown Copyright; 2016.
3. Shin L. Republic of Georgia to pilot land titling on blockchain with economist Hernando De Soto, Bitfury; 2016.
4. AlBar AM, Mohsen A, Tsaramirsis G. Business process modeling notation for MEASUR's Semantic Analysis. 3rd International Conference on Computing for Sustainable Global Development (INDIACom); 2015.
5. Shin L. The first government to secure land titles on the Bitcoin blockchain expands project; 2017.
6. Yamin M, Tsaramirsis G, Nistazakis M, Zhang N. Transformation of semantic analysis to com+ business requirements using MDA approach. *J Serv Sci Manag*. 2010;3:67.
7. Du M, Ma X, Zhang Z, Wang X, Chen Q. *A Review on Consensus Algorithm of Blockchain*. China: IEEE; 2017.

8. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system; 2008.
9. Registering property the paths of digitization, Doing Business; 2016.
10. Crosby M, Nachiappan K, Pattanayak P, Verma S, Kalyanaraman V. *BlockChain Technology Beyond Bitcoin*. Berkeley, CA: Fairchild, University of California; 2019.
11. Buterin V. A next-generation smart contract and decentralized application platform. https://github.com/Ethereum/wiki/wiki/White-Chapter.
12. Land Titling and Credit Access Understanding the Reality, USAID (United States Agency International Development).
13. Bach LM, Mihaljevic B, Zagar M. Comparative analysis of blockchain consensus algorithms. *MIPRO*. 2018;6:45-56.
14. Lian H, Yu Y. Reference of land registration system from countries in Asia Pacific region: comparison of the United States, Canada, Russia, South Korea, Japan and Australia. *IOSR J Bus Manag*. 2016;11:45-56.
15. Mulholland G, Pontesthe L. Technology behind the Integrated Cadastre/Land Registry Solutions in the Canadian Provinces of New Brunswick and Nova Scotia. In: Symposium on Innovative Technologies for Land Administration Madison WI; 2005
16. Porat A, Pratap A, Shah P, Adkar V. Blockchain consensus: an analysis of proof-of-work and its applications.
17. Domeher D, Abdulai R. Access to credit in the developing world: does land registration matter? *Third World Q*. 2012;33(1):161-175.
18. Xxxx. https://timesofindia.indiatimes.com/city/chennai/572-of-1700-tn-land-deal-frauds-in-8-months-in-chennai-zone/articleshow/64938941.cms
19. Dobhal A, Regan M. Immutability and auditability: the critical elements in property rights registries. Annual World Bank Conference on Land and Property: Annual World Bank Conference on Land and Property; 2016.
20. Mizrahi A. *A blockchain based property ownership recording system*. Stockholm, Sweden: ChromaWay; 2017.
21. Akingbade AO. Improvement of availability of land registration and cadastral information in Ondo state Nigeria; 2005.
22. Shalikashvili I. How can Georgian economy benefit from Bitcoin?; 2016.
23. Nicholsland S. *Registratıon: Managing Information for Land Administratıon*. New Brunswick, NJ: University of New Brunswick; 2017.
24. Yavuz E, Koç AK, Çabuk UC, Dalkılıç G. Towards secure e-voting using Ethereum blockchain. 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya; 2018.
25. Zhang K, Jacobsen H. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna; 2018
26. Hofmann F, Wurster S, Ron E, Böhmecke-Schwafert M. The immutability concept of blockchains and benefits of early standardization. 2017 ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K), Nanjing; 2017
27. Lua EK, Crowcroft J. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun Surv Tutor*. 2005;7(2):72-93.
28. Shelkovnikov A. Blockchain applications in the public sector; 2016.
29. Ethereum Docs. https://www.Ethereum.org/
30. Chen T, et al. Understanding Ethereum via graph analysis. IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI; 2018.
31. Mougayar W. A decision tree for Blockchain applications: problems, opportunities or capabilities? *Startup Manag*. 2015;3:23-43.
32. Lemieux VL. Trusting records: is blockchain technology the answer? *Rec Manag J*. 2016;26(2):110-139.
33. Aldweesh A, Alharby M, Solaiman E, van Moorsel A. Performance benchmarking of smart contracts to assess miner incentives in Ethereum. 14th European Dependable Computing Conference (EDCC), Iaşi, Romania; 2018.
34. Niya SR, Shüpfer F, Bocek T, Stiller B. Setting up flexible and light weight trading with enhanced user privacy using smart contracts; NOMS 2018–2018 IEEE/IFIP Network Operations and Management Symposium, Taipei; 2018.
35. Eisenhardt K. Building theories from case study research. *Acad Manag Rev*. 1989;14(4):532-550.
36. Guha S, Grover V, Kettinger WJ, Teng JT. Business process change and organizational performance: exploring an antecedent model. *J Manag Inform Syst*. 1997;14(1):119-154.
37. Graglia JM, Mellon C. Blockchain and property in 2018: at the end of the beginning. *Innov Technol Gov Glob*. 2018;12(1–2):90-116.
38. Glaser F. *Pervasive Decentralisation of Digital Infrastructures: A Framework for Blockchain Enabled System and Use Case Analysis*. Hawaii: HICSS; 2017.
39. Haug A, Graungaard Pedersen S, Stentoft Arlbjørn J. It readiness in small and medium sized enterprises. *Ind Manag Data Syst*. 2011;111(4):490-508.
40. Chen Y, Chen S, Lin I. Blockchain based smart contract for bidding system. 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba; (2019).
41. Grimsley M, Meehan A. E-government information systems: evaluation-led design for public value and client trust. *Eur J Inform Syst*. 2007;16(2):134-148.
42. Morabito V. Smart contracts and licensing. *Business Innovation through Blockchain*. New York, NY: Springer; 2017:101-124.
43. Kirby P. *A humble update on the Honduras title project*; 2015.
44. Heider C, Connelly A. *Why Land Administration Matters for Development*. Washington, DC: World Bank Group; 2016.
45. Mohanta BK, Panda SS, Jena D. An Overview of Smart Contract and Use Cases in Blockchain Technology. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bangalore; 2018.
46. Jardine B. Georgia stakes place on wild frontier of blockchain governance; 2018.
47. Cui-Hong H. Research on Web3.0 application in the resources integration portal. 2012 Second International Conference on Business Computing and Global Informatization, Shanghai; 2012.

48. Manzalini A, Stavdas A. A service and knowledge ecosystem for Telco3.0-Web3.0 applications. 2008 Third International Conference on Internet and Web Applications and Services, Athens; 2008.
49. Metamask, https://metamask.io/.
50. Xxxx. https://infura.io
51. Tsaramirsis G, Karamitsos I. Charalampos Apostolopoulos “Smart parking: An IoT application for smart city”. 3rd International Conference on Computing for Sustainable Global Development (INDIACom); 2016.
52. Xxxx. https://nextjs.org
53. Coron JS. What is cryptography? *IEEE Secur Priv J*. 2006;12(8):70-73.

How to cite this article: Panda SK, Mohammad GB, Nandan Mohanty S, Sahoo S. Smart contract-based land registry system to reduce frauds and time delay. *Security and Privacy*. 2021;e172. https://doi.org/10.1002/spy2.172