



Decentralizing Commercial Property Registration and Bidding Process with Smart Contracts

Susmit Lavania¹ · Aditi Mishra¹ · Abhishek Sharma¹ · Prateek Jain²

Received: 23 September 2021 / Accepted: 26 June 2023
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2023

Abstract

Blockchain technology has brought up new opportunities in the area of business processes with security perspective. It is a decentralized technology which operates on a consensus algorithm in a multiple environment. The recent online bidding system involves the extra transaction cost along with the several intermediaries as the third party plays a crucial part in trading done between the buyers and sellers during the auction. The trust of third party is always doubtful. To do so, smart contracts are developed using ERC721 standard with remix as development platform for compiling the smart contracts. Meta-mask wallet is used to get the ether. This wallet is connected to the blockchain system using the Ropsten network. This is in line to the IPFS system which stores all the data. It generates as well as updates the mutable links to the content. The buyers and sellers are interacting using the front-end framework. The smart contract is running at back-end which executes as well as traces the transactions. The involvement of third parties is eliminated which enables the transactions directly between the buyer and seller. Smart contracts are autonomous in nature due to which they do not execute if any condition is required to violate. Hence, to make the system secure from the frauds, land properties were tokenized with ERC721 token to retain non-fungible and unique properties. Then, these tokens can be auctioned, transferred, queried, bought, and sold on ethereum smart contract. Front-end interacts with smart contract via web3js deployed on IPFS.

Keywords Blockchain · Smart contract · IPFS · Remix · ERC721 · Digital tokens · Metamask · Truffle

Abbreviations

ERC721	Ethereum request for comment
ERC20	Ethereum request for comment
IPFS	Interplanetary file system
DHT	Distributed hash tables
EVM	Ethereum virtual machine
ETH	Ether
IDE	Integrated development environment
Abi	Application binary interface

IPNS	Inter-planetary name system
DApp	Decentralized application

Introduction

The various transactions are executed in the blockchain. Those are required to be recorded. These are required to verify on the basis of the consensus algorithm. If it is validated, then the block is added to the blockchain. Many crypto-currencies run on this infrastructure [1, 2]. The process through which transactions are validated, it is called mining. The reward for correcting the mining is bitcoin [3]. The lack of central authority along with distributed public ledger allows this technology to grow beyond being a decentralized database of crypto-currency transactions. It utilizes a peer-to-peer infrastructure where all the decisions are taken on the basis of consensus algorithm crypto-graphic methods, which are used for security perspective. Smart contract is a computerized transaction protocol that processes the terms of a contract while eliminating the need for third party [5, 6]. It is a

✉ Abhishek Sharma
abhisheksharma@lnmiit.ac.in

Susmit Lavania
15ucs148@lnmiit.ac.in

Aditi Mishra
15ucs010@lnmiit.ac.in

Prateek Jain
prateek.jain@nirmauni.ac.in

¹ The LNM Institute of Information Technology,
Jaipur 302031, India

² Nirma University, Ahmedabad, Gujarat, India

script-based program on the distributed ledger allowing an immutable, verifiable, secure record of all contracts and transaction. Features of smart contract are autonomous, auto-sufficient, decentralized. Smart contracts are developed in network-specific languages like viper, solidity, Ivyland, and rust. Some of its advantages are that it is faster, simpler, offers hassle-free processes, which reduces settlement times, lowers operational overheads leading to economic financial products. It reduced administration costs owing to automation. Integrated several business processes occurring throughout a supply chain needs a well-functioning integrated collaboration of several parties, which are issues of data redundancy, trust, and security. This becomes a major concern for inter-organizational processes involving entrusted parties [8, 9]. Online bidding system involves the extra transaction cost along with several intermediaries as the third party played a crucial part in trading, thus trace-ability is required. Online auctions have always been an important as well as influential aspect of the world economy [10, 11]. In an auction, sellers get the opportunity to advertise their assets. After this, bidding is done to sell the asset at the highest possible price. Traditionally, there are four main types of auctions [7]:

1. *First-price sealed-bid auctions* Bidders submit their bids in sealed envelopes and hand them to the auctioneer. Subsequently, the auctioneer opens the envelopes to determine the bidder with the highest bid.
2. *Second-price sealed-bid auctions (Vickrey auctions)* It is similar to first-price sealed-bid auctions with the exception that the winner pays the second highest bid instead.
3. *Open ascending-bid auctions (English auctions)* Bidders increasingly submit higher bids and stop bidding when they are not willing to pay more than the current highest bid.
4. *Open descending-bid auctions (Dutch auctions)* Auctioneer initially sets a high price, which is gradually decreased until a bidder decides to pay at the current price.

This paper explores the application of blockchain in tracing the several transactions occurring between the several parties; hence, ensuring the trust and security was the major problem in the previous online bidding system. This paper also focuses on methods of developing online assets that could be sold in the online bidding system which is requires to deploy on blockchain infrastructure. The paper aims to extend the concept towards the full trace-ability of transactions through smart contracts. The proposed work represented the implementation of method through a software prototype.

Literature Review

The Ethereum provides decentralized software enabled through blockchain and smart contract functionality and is used for various applications such as crypto-currencies like Ether, e-auction platforms, and communication for distributed platforms. The Ethereum Virtual Machine (EVM) provides the decentralized work environment for deploying smart contracts and required functionality in the application with the possibility of self-execution. Gas is the fees or a portion of crypto currency required to perform the transaction; it is also used for resource allocation in EVM. As fees is involved in the process, it need to be kept somewhere. MetaMask is a wallet used for the same purpose; it also provides the accessibility and interaction with application through browser and blockchain.

Many research works have been focused on how the blockchain is enhancing the supply chains along with different business processes. It has significantly affected our economy as well. This technology enables a group of connecting networks, cryptography, consensus-making and market mechanisms. Research based on real-time predictive delivery performance metric, identity management system, identity authentication model, the potential to give different direction to the world of banking through feature of global money remittance, smart contracts, automated banking ledgers, along with digital assets have already been conducted [8].

Several types of research have been done on implementations of a formal, agent-based simulation model on the ethereum for smart contracts and blockchain as a service which is helping to users for a smart contract automatic integration testing platform online [9]. Surveys of Bitcoin, Ethereum, Kindly check and confirm the edit in the sentence “Surveys of Bitcoin, Ethereum...” and Hyperledger have already been done which show how these relate to each other in their mechanisms. Their operations as well as the relations between the participants are analyzed which are ruled by several consensus algorithms. Many researchers have also focused on how the payment can be done in third-party absence along with receiving money from one or more parties with mutual transactions only if certain conditions are met for contracts. The work has been done for implementation as well as evaluation of a management plane for both smart contracts and decentralized applications [11, 12].

This paper discusses the methods to develop a digital asset and to bid this digital asset on the ethereum network so that trust, security, along with the traceability of the system could be maintained. Since the transactions are both asynchronous and transparent to outside observers, a vulnerability is created such that outside observers can

intentionally affect the outcomes of pending transactions. In blockchain-based bidding system, this can manifest as front-running: it is the practice of intercepting transactions from the network mempool by monitoring that will influence the market price of a tokenized property asset along with one step. This is in front of transactions to gain a competitive price advantage over the expense of others. In this section of the paper, several aspects involved related to the proposed work which are discussed in detail.

In Fig. 1, the workflow for the proposed auction system is represented. The user interacts with the browser after creating a metamask wallet to gain ether. After this, the browser interacts with the smart contract to execute the further transactions.

The blockchain is the publicly distributed encrypted ledgers, in which blocks record the various transactions verified on the basis of consensus algorithm. If verified transactions are added to the ethereum network, many crypto-currencies run on this infrastructure. It is a database that is stored not only on our computer but also on thousands of other computers across the world. The block not only stores the transactions but also stores the order where those transactions are executed. The process through transactions are validated is called bitcoin [13].

Smart contract is written in solidity, which is a contract-oriented programming language. It is used on various platforms, by defining methods as well as properties. To maintain peer-to-peer network, the concept of the digital signature is used. The digital signature ensures that the person is doing the real transaction along with the correct amount of transaction. The public key of sender together with receiver is taken along with the amount. These data are fed to a hash algorithm where it gets encrypted by the private key. It is then added as the digital signature to our document. Now, this document is transmitted to the network after the document which is fed to a hash function and again hashes the

amount data. The signature of the document is decrypted using the public key. Now, both the hashes are checked. If they are the same, then it is understood that the transaction is valid as well as ensures that the data are not manipulated and propagated [14].

Smart contract executes as listing property by changing the states in ethereum with one of the functions written with access modifier in it. Once the function executes a transaction, it is sent to the transaction pool. Miners are incentivized for the gas associated to mine the transaction along with including in the block. Once the transaction is broadcasted and mined property is successfully registered on the ethereum, property is tokenized with the help of ERC721. Many functions are available in smart contract to modify the price of property. Any user to pay the amount becomes the owner of the property after which it can be sold to its next owner. There are functions that are private and public. Private functions are used for internal consumption within the smart contract, while the public function is available for all users in the world. Properties to be auction need to be tokenized with the help of ERC721. ERC721 defines a few functions that give it some compliance with the ERC20 token standard. It makes it easier for existing wallets to display simple information about the token. The functions for the smart contract that fits this standard act like a common cryptocurrency such as Bitcoin or Ethereum by defining functions. These also facilitates to users to perform actions such as sending tokens to others as well as checking balances of accounts.

To do the further transactions along with recording them, metamask wallet is required. It allows the user to run Ethereum dApps right in the browser without running a full Ethereum node. MetaMask includes a secure identity vault that provides a user interface to manage the identities on different sites as well as to sign the transactions. Metamask extension converts chrome into ethereum browser with websites retrieving data

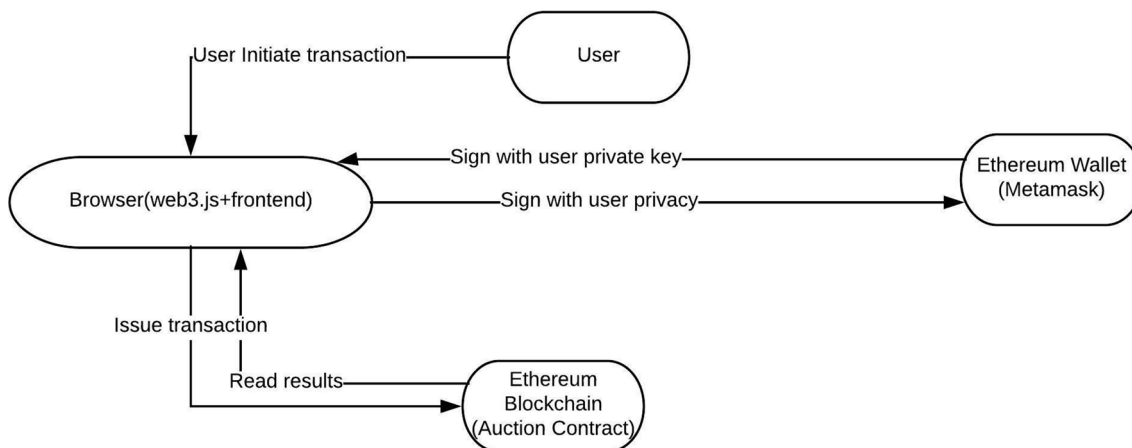


Fig. 1 Execution of the proposed system using Blockchain

from the blockchain. Users securely manage identities and sign transactions. When the metamask is started, a seed phrase is provided that can restore all of the metamask accounts. Account vault is encrypted as well as stored in the browser which means no account information ever touches servers. Its main strength is to enable the browser to access ethereum enabled websites. Through this, the connections with the main network are controlled. Metamask interfaces the interaction [11, 18] between ordinary websites and ethereum providers all while letting the user to store as well as manage the private keys. Metamask wallet is used to store the ethers which are used to pay for gas consumptions for further transactions. The limit is the amount of gas one is willing to spend on a transaction. Many software are utilized to broadcast Ethereum transactions that are capable to auto-estimate the amount of gas that will be necessary to implement a function. It will usually suggest a figure right off the bat. For example, simple monetary A->B transactions usually need only 21,000 gas. More complex ones which call-specific smart contract functions might run into hundreds of thousands or even millions of gas. The spent amount on gas is called gas cost. One can modify the amount of gas the user wants to spend on a transaction but if the transaction runs out of gas during execution, the user loses the gas sent in. It has been spent with the transaction being rejected. On the other hand, if the user provides more gas than is needed, the rest is refunded to the user. Hence, it is always better to send more gas than needed to execute a transaction. To store as well as maintain the dynamic links to the content, the interplanetary file system is used which is a protocol where each node stores the files in the hashed form. To access a file, only its hash value is required. IPFS basically generates hash values for all the encrypted files that are uploaded on the ethereum network. A hash table is a structure of data that contains information as key/value pairs. In distributed hash tables (DHT) [17, 27], the data are shared across a network of computers. It is coordinated to enable efficient access. The main advantages of DHTs are in decentralization, scalability, and fault tolerance. Nodes do not need central coordination. The system can work reliably even when nodes fail or leave the network. DHTs can be dynamic to accommodate millions of nodes. These features give the result in a system together that is generally more resilient than client-server structures. The IPFS protocol consists of identities, routing, exchange, objects, along with files. Several advantages of IPFS are deduplication, integrity, incentivized hosting, cheaper hosting, high performance, peer-to-peer scales better.

The Novel Contributions in Current Paper

Blockchain-based smart contract for bidding system is reported with significant contribution. The significant contributions in the present paper are as follows:

1. Implementation of consensus application in a distributed environment.
2. Inclusion of ERC721, Meta-mask wallet, and Ropsten network-based blockchain system.
3. Application development to mitigate fraud activities, more secure bidding platform.
4. Implementation of IPFS and web3js for front-end interaction.

Proposed Methodology

In this section, the approach used to develop the non-fungible digital asset using ERC721 is discussed and then how the smart contracts (written in solidity) are deployed on the blockchain network is also discussed. The approach used in the paper has four phases. In the first phase, the digital token is developed by writing smart contracts in solidity using the ERC721 standard; in the second phase, a metamask wallet is created; in the third phase, frontend for the smart contracts is developed, connected to the backend; and in the final phase, this whole system is deployed to interplanetary file system [16, 17].

Phase 1: Building up the Property

A repository called property is made. The project is initialized with truffle init. Several contracts like migrations (Truffle utilizes a migration system to handle smart contract deployments. A migration is an additional and special smart contract that holds track of changes.), test, truffle-config.js, truffle.js directories, and files are being added to repository. After this, the property.sol, math.sol, and ERC721.sol along with owner.sol contracts are added in the contracts directory as well as compiled. property.sol (because written in solidity) contains business logic. According to this contract, several properties can be created. It is a non-fungible as well as unique property/digital asset as it implements ERC721 standards. To prevent overflow in the arithmetic operations, safemath.sol library is used. To maintain the ownership privacy, a smart contract called owner.sol is developed. Ganache is installed as well as run in the background. After this, navigation to the repository migrations is made and a file named 2_deploy_contracts.js is created. Then, truffle migrate is hit. The status of blockchain changes in ganache. The code is deployed on the ganache.

Phase 2: Deploying Smart Contracts

Ropsten network is used for deploying the smart contracts but before that, a wallet is needed so the chrome extension called Metamask [11, 19] is used in which certain terms and conditions are acceptable to use it. Metamask helps the user to gain the ether which is a unit of payment for the gas used in the transactions. Every available low-level operation in the EVM is known as an OPCODE. These include the operations like ADD, BALANCE, CREATE—create a new contract. Gas is an abstract number that shows the relative complexity of operations. All kind of transactions cost is 21,000 gas as a base. Hence, transferring the funds from one account to another without using a smart contract will definitely cost 21,000 gas. Gas price is fixed for each operation, the amount paid by user per gas—gas price is dynamic and dictated by market conditions. Gas price is a value which represents the Ether is required to pay by the user per gas. When particular user sends a transaction, they mention the gas price in Gwei/Gas (1 Gwei equals 0.000000001 ETH), the total paying fee that is equal to $\text{gas_price} * \text{gas_used}$. Miners are paid out the fee so they use to give priority to the transactions with a higher gas price. The higher gas price you are willing to pay, the faster your transaction will be processed. Operations in Ethereum cost $\text{gas_price} * \text{gas_used}$, but it does this translate in both ether and dollars.

To connect this wallet to the ethereum network, the system is connected to the Ropsten network where an account is created and using the account address, ether is gained to do the further transactions.

The contracts are then copied to remix ide as well as are executed. Injected Web 3 Ropsten under environment is selected. The account in Metamask is shown under Account with balance ether as well. Then, the system is deployed by clicking on the Deploy button. The metamask popup comes up with transaction details the gas price is filled. Once this is completed, an etherscan link is generated. The account address and the abi are copied because these details would be required in connecting the frontend together with backend.

Phase 3: Frontend for Smart Contract and Interaction

A repository called frontend in the directory is created. In that repository, a file called index.html is added in which the code for the front end is written. Now, to make this frontend interact with the smart contracts, web3js is used. If web3 is not undefined, then it is used as the provider. If it is undefined, then the provider is specified manually. Now, the code is compiled in Remix IDE under the Details section, Interface ABI data are copied. In the index.html file, contract address and Abi are pasted.

Phase 4: Deployment to IPFS

The interplanetary file system is a protocol where each node stores the files in the hashed form and whenever access to some file is required, its hash value should be known [21, 23, 24]. IPFS basically generates hash values for all the encrypted files that are uploaded on the blockchain network. IPFS is installed as well as run. From here, the hash gives the root of the site. After all this process, the next step is publishing to Inter-Planetary Name System (IPNS) (Data in IPFS is content-addressable so it keeps on changing with dynamic data [30]. So, IPNS is used for creating along with updating mutable links to IPFS content. The name in IPNS is the hash of a public key. It is from a record which contains the information about the hash. It links to that which is signed through the corresponding private key.). This returns a peerID and the hash that is published. The distributed application is finally deployed.

In Fig. 2, the working of the proposed system has been shown. The smart contract is developed using ERC721 protocol. Ganache is started in the background. Metamask wallet is created so that ether can be obtained. This wallet is connected to the blockchain using the ropsten network from where application binary interface and address is returned. Using these two, the frontend is connected to the smart contract. This whole system is then published to the IPFS.

Results

Smart contract once deployed is written forever. Its functions are communicated with the help of web3js. Web3js consumes abi along with contract address to interact with smart contract and pops up metamask to fill gas price to send transactions that modify the states. The front end that interacts with smart contract are hosted ipfs to be truly decentralized as if uploaded on the server gives a honey spot along with becoming centralized. Smart contract can be developed on remix or truffle environment. Truffle provides an ecosystem for smart contract development and deployment. Smart contract can be tested on local blockchain or testnet. Ganache works in hand with truffle providing us ten default account funded with 100 eth. Ropsten and kovan testnet are also be used in conjunction for a function to be executed as opposed to real ethers.

Interacting with smart contracts is basically interacting with the functions written in it. These functions need to have access modifiers defined and visibility also. Access modifier is a way to restrict the user to some predefined community of users. Visibility of function means it needs to be public as well as open for all or required for internal communication within the smart contract to be consumed by other functions. Figure 3 defines functions property along with

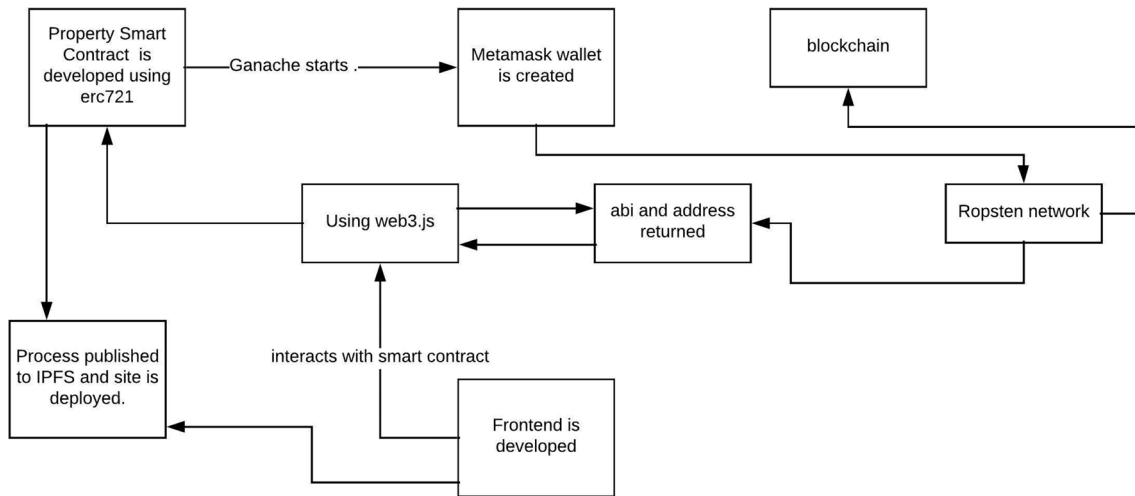


Fig. 2 The experimental steps involved in building the system

their ability to modify states or not. Only one function in the smart contract is payable which means only that function can receive ethers.

Figure 4 shows the basic functioning of the proposed system under which deployer would set the mint tokens and various other functions using the smart contract. Buyers together with sellers use that smart contract for carrying out the further auction. Each time any transaction occurs, it is recorded in the block. Then each block is verified by miners as well as added to the blocks which are added to the blockchain. This way the new infrastructure eliminates the chances of frauds, increases security and transparency.

In Fig. 5, Contract Creation Transaction is illustrated on the left side of the dotted line. Here, the smart contract is written in.sol file which is compiled by the solidity compiler. After compilation, it is sent back to the distributed application from where it pushes the smart contract to the ethereum network. Web3 establishes smart contract interaction with frontend. At this point, nonce remains zero. The right side of the dotted line shows how the function call transaction occurs. After the deployment of smart contract on the network, address and application binary interface(abi) are returned which would be used for connecting frontend to backend. Now, interface can make a call (this call contains the address, abi, nonce) to the smart contract for validating as well as executing the requested transaction. Nonce becomes 1 and this nonce prevents any duplicate transaction which increases security.

The operations in Ethereum cost $\text{gas_price} * \text{gas_used}$, but it does translate to in both ether as well as dollars. The transaction fee depends on the transaction that one is doing. For a simple transfer of ETH from one address to another address would be done in about 21,000 wei of gas whereas if one is interacting with a contract it might be 150,000 or








more. The actual transaction fees are: Gas amount multiply by Gas price and the gas price is variable as it depends on the number of transactions pending in the network. ETH Gas Station is a good source to estimate the average gas price while doing your transaction.

A lower gas price is considered by most of consumers since it permits for inexpensive transactions on the Ethereum. When the gas prices increase, it puts a strain on the entire network and hinders transaction activity, thus causing DApp usage to suffer. Another effect of high gas prices is that it increases the ability to improve the profit of being a miner. In theory, this creates further incentives for more miners to join the network, which will increase throughput. When the amount of newly generated transactions in ethereum exceeds maximum throughput (for Ethereum, throughput is dictated by the block gas limit), the backlog of pending transactions starts to grow, it will increment the average length of time that a transaction will remain in the mempool before being mined into a block. Due to this time delay race conditions come into picture making these transactions vulnerable to race conditions. In the ethereum-based transactions, race conditions manifest as trade collisions(accidental) when same bid order is attempted by both parties to be filled simultaneously.



The smart contract in our ecosystem inherits two smart contract ERC721 as well as ownable. ERC721 standard is required to make non-fungible assets. ERC721 allow properties to be unique tokens that can be managed, owned, as well as traded. The ERC721 standard defines the following functions: name, symbol, balanceOf, ownerOf, totalSupply, approve, takeOwnership, transfer, tokenOfOwnerByIndex, tokenMetadata. It also defines two events: Transfer and Approval. Inheriting ownable smart contract allows us to add modifiers to prevent certain functions from being used.

Fig. 3 Functions access modifier type, visibility, and mutability

Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Property	Implementation	ERC721, Ownable		
L	mintToken	Public !		onlyOwner
L	buyProperty	Public !		
L	bidPropertyPrice	Public !		
L	withdraw	External !		onlyOwner
L	balanceOf	Public !		
L	ownerOf	Public !		
L	transfer	Public !		
L	approve	Public !		
L	takeOwnership	Public !		

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

In Fig. 6, the property smart contract inherits two smart contract ERC721 and ownable smart contract for having non-fungible properties. It access control over its functions. Property smart contract consists various function that serves the purpose of business logic that smart contract needs to carry out trustlessly. Some of these functions modify public state variables or private state variable of smart contract and consume gas. Some function not only need to query the state but also do not modify state on the blockchain. Some function might need access control which is enforced by ownable smart contract or require in solidity.

Figure 7 shows all functions of smart contract that encodes the business logic in an auction of property through smart contract. Smart contract ecosystem has three types of roles that a party can play. An interested party can be deployer of smart contract on ethereum and simultaneously be a bidder as well as a buyer. Another interested party can be either a bidder or buyer. The bidder can either raise or lower the property prices in ethereum.

Table 1 illustrates the role of different ethereum account plays while interacting with smart contract. Account 0x0aa.. deploys the smart contracts on ropsten network, it can also bid as well as buy through smart contract. Account 0x319.. and 0x27e.. play the role of bidding as well as can buy property through smart contract.

Table 2 shows the transaction where ethereum account 0x0aa.. has deployed the contract on ropsten network. The contract address on Ropsten ethereum network is 0x35c75.. and gas used by transaction was 1,833,370 which costed 0.00183 ethers. Smart contract can be deployed through various ways on ethereum that is with truffle or remix with metamask. It can also be deployed on various networks like Ropsten, Rinkby with ether that has no value and used only for testing purpose. The ethereum account or wallet that deploy the smart contract becomes the owner of smart contract. Smart contract once deployed are immutable.

Once smart contract is on chain interaction with its function can be done. To start auction, the user need to register

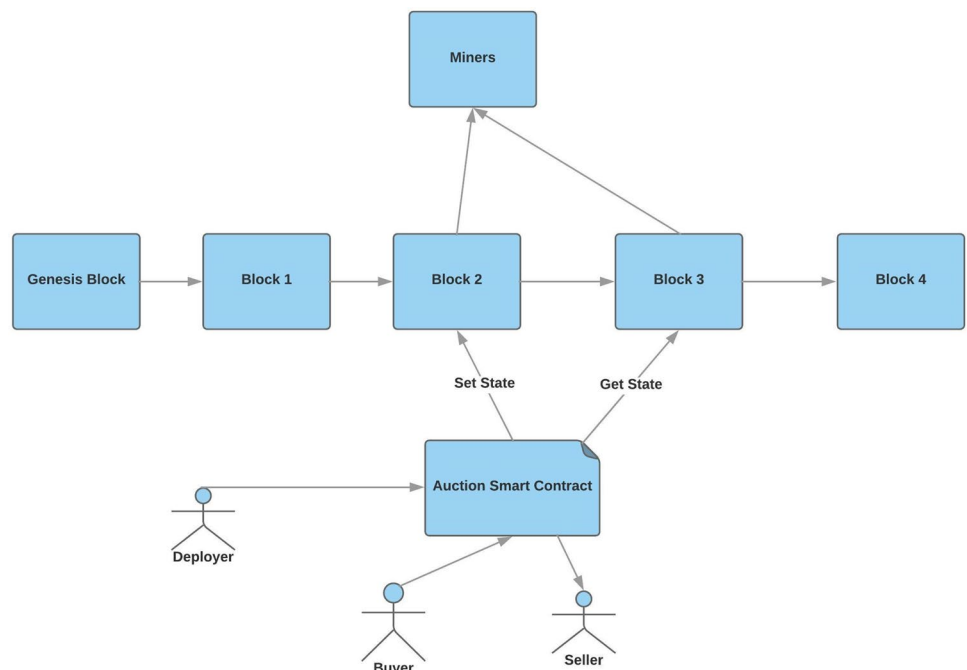
the property. The mintToken function handle this for the user. mintToken has three arguments and has onlyOwner as its access modifier, which means only the owner of smart contract can register the property. This function will require transaction fees as it has gas associated with it because it modifies a blockchain state.

Table 3 and Algorithm 1 show transaction where the deployer of smart contract 0x0aa.. Mint an ERC721 token with three parameters name, location and price. This ERC721 token represents a property on ropsten ethereum. Only 0.000170262 ether was the transaction fees for minting a token. The sealed-bid electronic auction platform is

Algorithm 1 Showing the function input, output, and logs

decoded input	{ "string _name": "Inmiit", "string _location": "jaipur", "uint256 _price": "2" }
decoded output	-
logs	[{ "from": "0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1", "topic": "0x1288a9be9cf41889b5dbfb4b2b742f7db98a8389aa3027315b45c23791dfca27", "event": "PropertyRegistered", "args": { "0": "Inmiit", "1": "jaipur", "2": "2", "_name": "Inmiit", "_location": "jaipur", "_price": "2", "length": 3 } }]

Fig. 4 Interactions shown in the multi-user environment



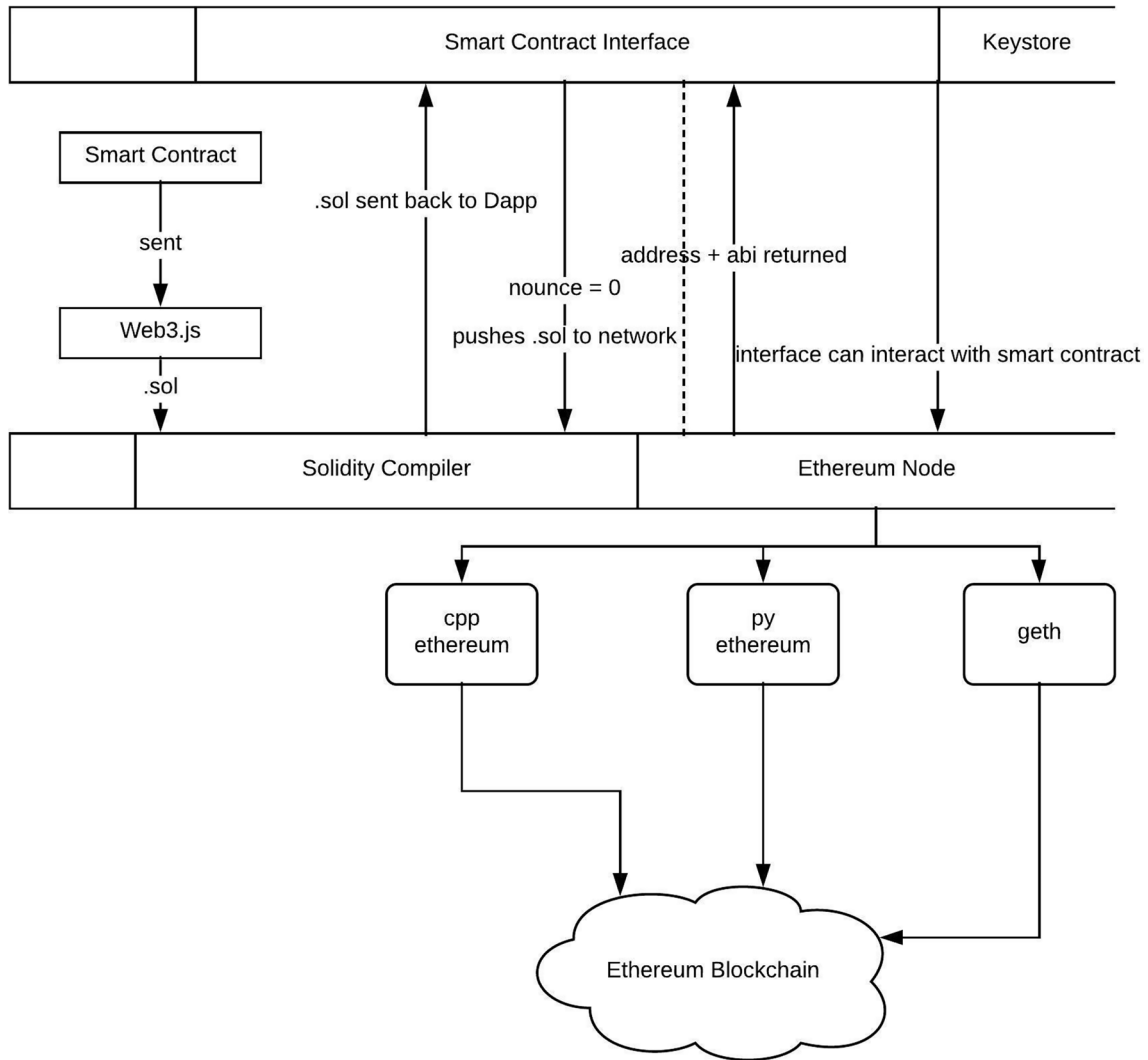


Fig. 5 Ecosystem of ethereum smart contract from sending transactions to the deployment of smart contracts

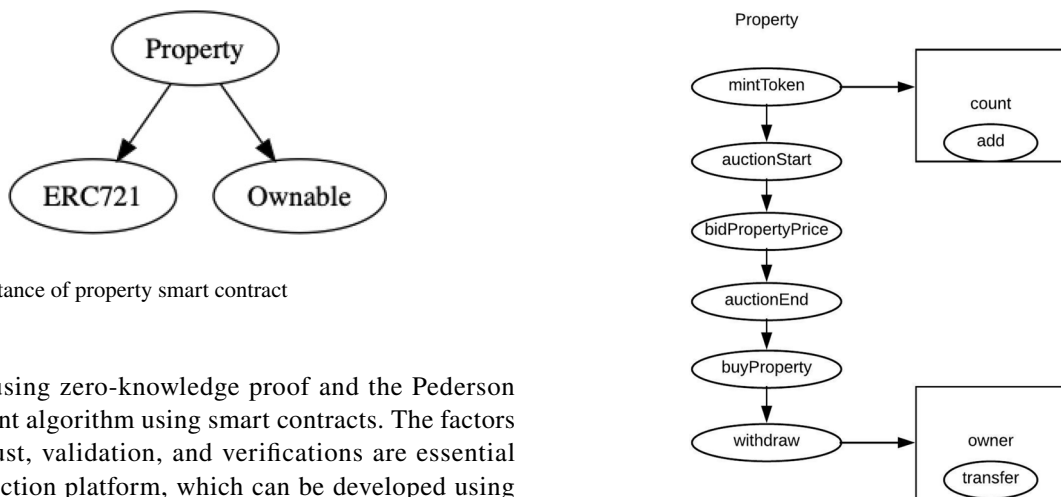


Fig. 6 Inheritance of property smart contract

designed using zero-knowledge proof and the Pederson commitment algorithm using smart contracts. The factors such as trust, validation, and verifications are essential in the e-auction platform, which can be developed using blockchain and smart contracts with ERC721 tokens [32].

Fig. 7 Showing smart contract functions of property

Table 1 Interacting parties roles in smart contract on the blockchain

Deployer,Bidder1,Buyer1	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Bidder 2 and Buyer2	0x3192324af2fc309dbbb4a98b5b7f04a95e819ea8
Bidder 3 and Buyer3	0x27e64eb92f1b338984c599b8120cf464fc7406a8

Table 2 Deployment of smart contract (Ropsten network)

Block height	4,728,663
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd-94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether (\$0.00)
Gas limit	1,833,370
Gas used by transaction	1,833,370 (100%)
Gas price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.00183337 Ether
Nounce & (Position)	10 — {35}

Table 3 Minting of property ERC721 token

Block height	4,728,695
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd-94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether (\$0.00)
Gas Limit	170,262
Gas used by transaction	170,262 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000170262 Ether
Nounce & (Position)	11 — {51}

After registering property in the etherium, the auction of the property can be started. Since property is public state anyone around the world can see this property details. Now to start the auction auctionStart function in the smart

contract need to be called with token id as parameter. Only the owner of that token id can start the auction which is enforced by require function (see Tables 4, 5, 6, and 7).

Algorithm 2 Showing the function input, output, and logs

decoded input	{ "uint256 _id": "0" }
decoded output	-
Logs	[]

Algorithm 3 Showing the function input,output and logs

decoded input	{ "uint256 _prc": "3", "uint256 _id": "0" }
decoded output	-
logs	[]

Table 4 Auction start transaction

Block height	4,728,726
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether (\$0.00)
Gas limit	42,997
Gas used by transaction	42,997 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000042997 Ether
Nounce & (Position)	12 — {40}

Table 5 Bid 1 by deployer ethereum account

Block height	4,728,736
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether
Gas limit	27,814
Gas used by transaction	27,814 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Max Tx Cost/Fee	0.000027814 Ether
Nounce & (Position)	13 — {Pending}

Table 6 Bid 2 another ethereum random account

Block height	4,728,773
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x3192324af2fc309dbbb4a98b5b7f04a95e819ea8
Value	0 Ether (\$0.00)
Gas limit	27,814
Gas used by transaction	27,814 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000027814 Ether
Nounce & (Position)	0 — {52}

Table 7 Bid 3 by random ethereum account

Block height	4,728,786
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x27e64eb92f1b338984c599b8120cf464fc7406a8
Value	0 Ether (\$0.00)
Gas limit	27,814
Gas used by transaction	27,814 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000027814 Ether
Nounce & (Position)	18 — {25}

Table 8 Auction end transaction

Block height	4,728,811
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether (\$0.00)
Gas limit	22,933
Gas used by transaction	22,933 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000022933 Ether
Nounce & (Position)	14 — {45}

Algorithm 4 Showing the respective code

decoded input	{ "uint256 _prc": "4", "uint256 _id": "0" }
decoded output	-
Logs	[]

Algorithm 5 Showing the respective code

decoded input	{ "uint256 _prc": "5", "uint256 _id": "0" }
decoded output	-
Logs	[]

Table 9 Mint a new ERC721 token

Block height	4,728,904
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
Value	0 Ether (\$0.00)
Gas limit	125,006
Gas used by transaction	125,006 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000125006 Ether
Nounce & (Position)	15 — {60}

Presented Tables and Algorithms 1–7 demonstrate the ethereum account bidding on a property registered within on which auction has started. To place the bid, an account has to provide a price in ether and token id no to place the bid. All the above transaction has used only 27,814 gas for bidding. Account 0x0aa...,0x27...,0x391.. are bidding here.

Now, after the market has saturated and the property has reached its conclusive price it should be available for sale so that interested party can buy this property. A property can only be bought when it is out of auction which can be initiated by auctionEnd function.

Table 10 Buy a property

Block height	4,728,932
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
From	0x27e64eb92f1b338984c599b8120cf464fc7406a8
Value	3 Ether (\$0.00)
Gas Limit	54,351
Gas used by transaction	54,351 (100%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000054351 Ether
Nounce & (Position)	19 — {108}

Table 11 Withdraw funds from contract

Block Height	4,728,970
From	0x0aa7e2509416a9bb8166898cbfbb8a22fc7328ca
To	0x35c7f08c63b7de8f79f5a8bb4ad83fd94fefa4d1
Value	0 Ether (\$0.00)
Gas Limit	32,191
Gas Used By Transaction	29,936 (92.99%)
Gas Price	0.000000001 Ether (1 Gwei)
Actual Tx Cost/Fee	0.000029936Ether
Nonce & {Position}	16 — {57}

Txhash	Blockno	From	To	TxnFee(ETH)
0xf0e26bfa9a2b1c69e a1499bc6b4afe6d78e 20e61f6cf94dd9ae98a 826018ada7	4728663	0x0aa7e2509416a 9bb8166898cbfbb 8a22fc7328ca	0x35c7f08c63b7de8f79f 5a8bb4ad83fd94fefa4d1	0.00183337
0xdf276d42693187ae 8bdd4899f8b3041480 a7a45ea5f4368e1d2f1 a1575fd6237	4728695	0x0aa7e2509416a 9bb8166898cbfbb 8a22fc7328ca	0x35c7f08c63b7de8f79f 5a8bb4ad83fd94fefa4d1	0.000170262
0x7e0195fc87f9800a 5b3e92b08fcc177942 21320bfc956accfe635 bb4b0eb6857	4728726	0x0aa7e2509416a 9bb8166898cbfbb 8a22fc7328ca	0x35c7f08c63b7de8f79f 5a8bb4ad83fd94fefa4d1	0.000042997

Fig. 8 Transaction list of smart contract interaction and their fee

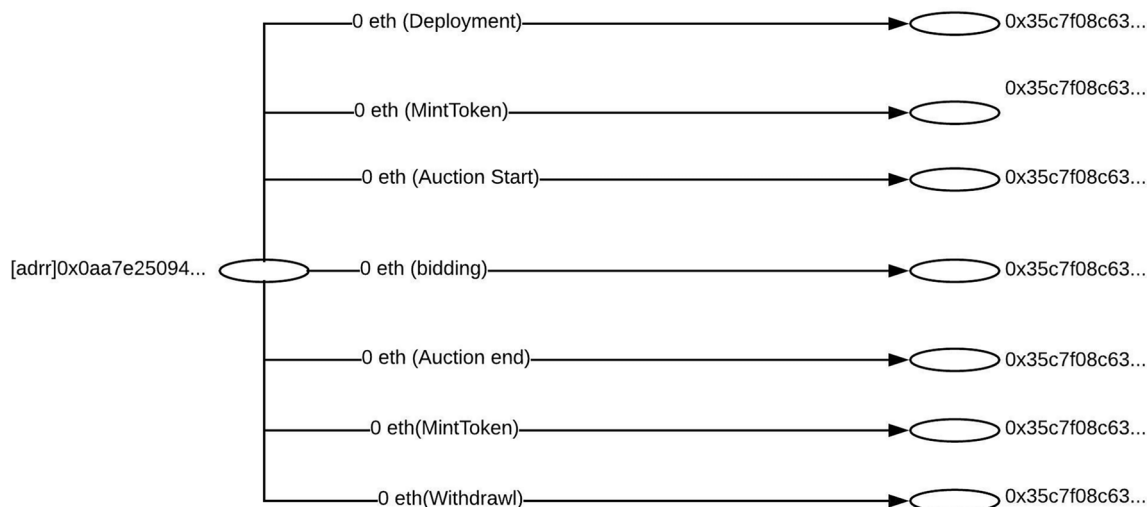


Fig. 9 Interaction of Eth account 0x0aa.. that plays the role of deployer, mint token, auction start, auction end, bidding, and withdrawal

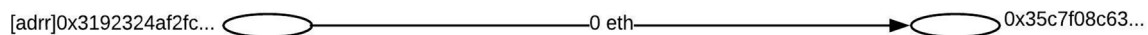


Fig. 10 Interaction of eth account 2 that played the only role of bidding in smart contract

Algorithm 6 Showing the respective code	
decoded input	{ "uint256 _id": "0" }
decoded output	-
logs	[]

Table 8 and Algorithm 6 represent the transaction sent by ethereum account 0x0aa.. to end the auction. By putting the end to auction, ERC721 token will be available for sale and others cannot bid on it.

Smart contracts can handle registration of property and can start its auction. This auction can then decide the price of property in consensus with other people around the world. New property can also be registered with the same contract with mintToken function.

Algorithm 7 Showing the function input,output and logs

decoded input	{ "string _name": "mnit", "string _location": "mngb", "uint256 _price": "3" }
decoded output	- [{ "from": "0x35c7f08c63b7de8f79f5a8bb4ad83fd94fe4d1", "topic": "0x1288a9be9cf41889b5dbfb4b2b742f7db98a8389aa3027315b45c23791dfca27", "event": "PropertyRegistered", "args": { "0": "mnit", "1": "mngb", "2": "3", "_name": "mnit", "_location": "mngb", "_price": "3", "length": 3 } }]
logs	{ "0": "mnit", "1": "mngb", "2": "3", "_name": "mnit", "_location": "mngb", "_price": "3", "length": 3 }

Table 9 and Algorithm 7 show a transaction where 0x0aa.. registered another property ie ERC721 token. To register this property, it requires mining of ERC721 token and emits an event called Property Registered. These events

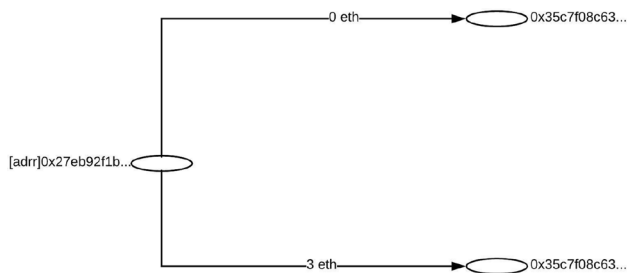


Fig. 11 Interaction of eth account 3 that played the role of highest bidder and subsequently buying property for 3 eth

are available for monitoring the activities going inside smart contract.

Property smart contract is able to handle multiple property registration and their auction simultaneously. To buy a property from the current owner, it needs special payable function purchase property that can receive ether to change the owner of the property. This function can only be triggered when the property is out of auction and price paid is equal to property value.

Table 10 explains a transaction sent by account 0x27.. to smart contracts for purchase the registered property. Since the property cost was 3 ethereum account 0x27.. had to give 3 eth to smart contract to purchase the property.

Since the smart contract handles ether transaction, ether can get deposited to contract ethereum account due to various reasons. Withdraw function is external and known by the owner to transfer the contracts funds to deployer account to resolve disputes.

Table 12 Analysis of resource utilization for various activation function

Scheme	[33]	[34]	[35]	[36]	[32]	Proposed work
Decentralization	No	Yes	No	Partial	Yes	Yes
Anonymity	Yes	Yes	Yes	Yes	Yes	Yes
Authentication	Yes	Yes	Yes	Yes	Yes	Yes
Unforgeability	Yes	Yes	Yes	Partial	Yes	Partial
Nonrepudiation	Yes	Yes	Yes	Partial	Yes	Yes
Validation	Yes	No	Yes	Yes	yes	Yes
IPFS	No	No	No	no	No	Yes

Table 11 elaborates the transaction for withdrawing the funds to 0x0aa.. ethereum accounts. These were the funds that payable function in our smart contract received which refers to smart contract account. This transaction needed 29,936 gas and it was provided 32191, i.e., utilized 93% gas and rest were refunded to the user.

Figure 8 represents the summary report of some transactions by the property smart contract on ropsten network using faucet ethereum. These transactions are irreplaceable and can be marked by any blockchain explorer. They consist of the transaction fee that one has to pay for the transaction.

All ethereum account in the world can interact with property smart contract functions. These functions are access improved which means not everyone can execute all functions. They can only execute functions according to the functions played. 0x0aa account implemented the property smart contract on the Robsten network. It could also originate the auction start and auction end which can only be performed by sending the transaction to ethereum. The same account since was the deployer of the smart contract which had access to the mint token and withdraw functions of the smart contract.

In Fig. 9, 0x0aa.. sends 7 transactions to the ropsten through smart contracts. 0x0aa.. performed the role of smart contract deployer, property initiation, and termination of the auction and withdrawing funds from the smart contract.

Since all functions in the smart contract can interact through all ethereum accounts to the world. Accounts that are neither deployers nor any owner of the registered property with a smart contract can only bid on the property once the auction has started.

An ethereum account has the highest bid. It can purchase the property once the owner of the property eliminates it from the auction. Then, the buyer can send his ether to the payable function of smart contract that changes the owner state of ethereum.

Interaction of eth accounts 2 and 3 that played the role of bidding in smart contract is reported in Figs. 10 and 11. Smart contracts can effortlessly eliminate the requirement of middlemen from any business logic. After the arrangements, users require to interact with it in a simple way so it requires to be executed with the front-end. To integrate the smart contract with front-end abi and the contract, an address on ethereum is needed. These parameters are needed by web3js so that metamask can interact with the property smart contract underhood. To make a truly decentralized system, deploying the front end on the server is not directed towards decentralization. For decentralization, the front end can be hosted on IPFS. Since everything in IPFS is content-addressed and their address updates every time with the corresponding content. An IPNS is the hash of a public key. It

is corresponding to a record containing data about the hash it links to that is signed by its private key. New records can be signed and published at any time. Table 12 represents the analysis of resource utilization for various activation functions.

Conclusion

In the proposed work, an advance contract is presented for a justified blockchain-based auction system on Ethereum. ERC721, which delivers a standard for the advancement of non-fungible assets, is used to evolve the auction's digital asset. Here, the smart contract elaborates all the restrictions for assuring security. Whenever the non-relevant user attempts to violate the conditions of the smart contract, warnings are originated as well as after each attempt, acknowledgement of each transaction is originated. The smart contract gets executed (as it is an autonomous unit, it executes by itself). If all the defined conditions are satisfied, transactions get executed using the functions specified in the smart contract and all these transactions are tracked as well as recorded. The gas consumption, transaction fee, and the time required for the transactions are also recorded and displayed. For future work, other approaches can be analyzed which will apply to the ethereum so that the privacy of bids can be protected from all parties including the auctioneer.

Funding Not applicable.

Data availability The code and all other information will be available on request.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical Approval This article does not contain any studies with animals performed by any of the authors.

References

1. ethtrade.org. Understanding Ethereum. <https://ethtrade.org/Understanding Ethereum.pdf>, 2017-06-22. 2016.
2. Bentov I, Kumaresan R. How to use bitcoin to design fair protocols. In: International Cryptology Conference, 2014; p. 421–439, Springer.
3. Bonabeau E. Agent-based modeling: methods and techniques for simulating human systems. *Proc Natl Acad Sci.* 2002;99(suppl 3):7280–7.
4. Burgess M, Hagerud H, Straumnes S, Reitan T. Measuring system normality. *ACM Trans Comput Syst (TOCS).* 2002;20(2):125–60.
5. Bussmann S, Jennings N, Wooldridge M. Multiagent systems for manufacturing control. Berlin: Springer-Verlag; 2004.

6. Eriksson J, Finne N, Janson S. Sics marketplace: an agent-based market infrastructure. In: First International Workshop on AgentMediated Electronic Trading, AMET-98, Selected Papers. Springer. 1998.
7. Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L. Secure multiparty computations on bitcoin. In: 2014 IEEE Symposium on Security and Privacy (SP), pages 443–458. IEEE, 2014.
8. Kroll J, Davey IC, Felten E. The economics of bitcoin mining, or Bitcoin in the presence of adversaries. In Proceedings of WEIS, volume 2013. 2013.
9. Khan N, Lahmadi A, Francois J, State R. Towards a management plane for smart contracts: Ethereum case study. In: NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium. 2018. <https://doi.org/10.1109/noms.2018.8406326>.
10. Hou Y, Chen Y, Jiao Y, Zhao J, Ouyang H, Zhu P, Liu Y. A resolution of sharing private charging piles based on smart contract. In: 2017 13th International Conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD). 2017. <https://doi.org/10.1109/fskd.2017.8393262>.
11. Di Ciccio C, Cecconi A, Mendling J, Felix D, Haas D, Lilek D, Riel F, Rumpl A, Uhlig P. Blockchain-based Traceability of inter-organisational business processes. 2018. https://doi.org/10.1007/978-3-319-94214-8_4.
12. Dumas M, Rosa ML, Mendling J, Reijers HA. Fundamentals of business process management. 2nd ed. Heidelberg: Springer; 2018. <https://doi.org/10.1007/978-3-662-56509-4>.
13. Liao C-F, Cheng C-J, Chen K, Lai C-H, Chiu T, Wu-Lee C. Toward a service platform for developing smart contracts on blockchain in BDD and TDD styles. In: 2017 IEEE 10th Conference on service-oriented computing and applications (SOCA). 2017. <https://doi.org/10.1109/soca.2017.26>.
14. García-Bañuelos L, Ponomarev A, Dumas M, Weber I. Optimized execution of business processes on the Blockchain. In: Carmona J, Engels G, Kumar A, editors. BPM 2017, vol. 10445. LNCS. Cham: Springer; 2017. p. 130–46. <https://doi.org/10.1007/978-3-319-65000-5>.
15. Weber I, Xu X, Riveret R, Governatori G, Ponomarev A, Mendling J. Untrusted business process monitoring and execution using Blockchain. In: La Rosa M, Loos P, Pastor O, editors. BPM 2016, vol. 9850. LNCS. Cham: Springer; 2016. p. 329–47. <https://doi.org/10.1007/978-3-319-45348-4>.
16. Cha S-C, Peng W-C, Huang Z-J, Hsu T-Y, Chen J-F, Tsai T-Y. On design and implementation a smart contract-based investigation report management framework for smartphone applications. Smart Innov Syst Technol. 2017. https://doi.org/10.1007/978-3-319-63859-1_35.
17. Saberi S, Kouhizadeh M, Sarkis J, Shen L. Blockchain technology and its relationships to sustainable supply chain management. Int J Prod Res. 2018. <https://doi.org/10.1080/00207543.2018.1533261>.
18. Egelund-Müller B, Elsmann M, Henglein F, Ross O. Automated execution of financial contracts on blockchains. Bus Inf Syst Eng. 2017;59(6):457–67.
19. Huckle S, Bhattacharya R, White M, Beloff N. Internet of things, blockchain and shared economy applications. Proc Comput Sci. 2016;98:461–6.
20. Yin C. Energy blockchain and energy internet. Review. 2016;05:14–5.
21. Moergestel L, Berg Mvd, Knol M, Paauw Rvd, van Voorst K, Puik E, Telgen D, Meyer J-J. Internet of Smart Things—a study on embedding agents and information in a device. 2017.
22. Lansiti M, Lakhani KR. The Truth About Blockchain, Harvard business review. 2017. <https://hbr.org/2017/01/the-truth-about-blockchain>. Accessed 4 Jan 2018.
23. Bartoletti M, Pompianu L. An empirical analysis of smart contracts: platforms, applications, and design patterns. 2017. [arXiv: abs/1703.06322](https://arxiv.org/abs/1703.06322) [CoRR].
24. Thomas Cook AL, Lee JH. DappGuard: active monitoring and defense for solidity smart contracts. MIT, Tech. Rep., 2017.
25. Niya SR, Shupfer F, Bocek T, Stiller B. Setting up flexible and lightweight trading with enhanced user privacy using smart contracts. In: NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium. 2018. <https://doi.org/10.1109/noms.2018.8406112>
26. Bogner A, Chanson M, Meeuw A. A decentralized sharing app running a smart contract on the Ethereum blockchain. In: Proceedings of the 6th International Conference on the Internet of Things, ser. IoT'16. New York, NY, USA: ACM, 2016; pp. 177–178.
27. Bocek T, Stiller B. Smart contracts-blockchains in the wings. Heidelberg: Springer; 2017. p. 1–16.
28. Buterin V, et al. 'What is Ethereum?' <http://www.ethdocs.org/en/latest/introduction/what-is-Ethereum.html>, Last Seen: 14 Jan 2018.
29. Niya SR, Schupfer F, Bocek T, Stiller B. A peer-to-peer purchase and rental smartcontract-based application. IFI-TecReport No. 2017.04, Zurich, Switzerland, Tech. Rep., Nov 2017.
30. Light Client Protocol. <https://github.com/Ethereum/wiki/wiki/Light-client-protocol>, Last Seen: Dec 25, 2017.
31. Ethereum.io. Ethereum average gas price chart. <https://etherscan.io/chart/gasprice>. Last seen 14 Jan 2019.
32. Li H, Xue W. A blockchain-based sealed-bid e-auction scheme with smart contract and zero-knowledge proof. Secur Commun Netw. 2021. <https://doi.org/10.1155/2021/5523394>.
33. Galal HS, Youssef AM. Verifiable sealed-bid auction on the Ethereum blockchain. In: Proceedings of the 2018 Financial Cryptography, pp. 265–278, Springer, Nieuwpoort, Curaçao, March 2018.
34. Peng Y, Gao Y, Wu JX. A privacy preserving sealedbid auction scheme based on block chains. Cybersp Secur. 2018;9(8):1–7.
35. Xiong J. Research on anonymous electronic auction protocol based on blockchain. Guangzhou: Jinan University; 2019.
36. Yu R. Research on the sealed-bid auction scheme for blockchain based on secure comparison protocols. Xianyang: Northwest A & F University; 2019.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.