The Journal of Supercomputing (2023) 79:13442–13461
https://doi.org/10.1007/s11227-023-05196-1

Collaborative Smart Contracts (CoSC): example of real estate purchase and sale(s)

Tunahan Timuçin[1] · **Serdar Biroğul**[1]

Accepted: 13 March 2023 / Published online: 25 March 2023

Abstract

The emergence of blockchain technology has opened up opportunities for innovative solutions in various fields. One of these solutions is the concept of smart contracts(SCs), which can automate the execution of contract terms and conditions. In this paper, we introduce Collaborative Smart Contracts (CoSC), a novel approach that combines the benefits of smart contracts and collaborative decision-making. CoSC allows multiple parties to participate in the contract execution process, enabling a more transparent and fair system. We present an example of the application of CoSC in the real estate industry, specifically in the purchase and sale of properties. Our proposed approach utilizes a consortium blockchain network to create a secure and decentralized environment for CoSC. The CoSC platform facilitates communication, collaboration, and decision-making between the buyer, seller, and other relevant parties involved in the transaction. We evaluate the performance of the CoSC platform in terms of execution time, benefits, and security. With the CoSC platform, an average of 20+% success is achieved in terms of execution time. Our results demonstrate that CoSC is a promising solution for complex, multi-party transactions and can improve the efficiency, transparency, and trustworthiness of the real estate industry.

Keywords Smart contracts · Collaborative structure · Real estate sales

✉ Tunahan Timuçin
tunahantimucin@duzce.edu.tr

Serdar Biroğul
serdarbirogul@duzce.edu.tr

[1] Computer Engineering, Duzce University, Düzce, Türkiye

1 Introduction

Bitcoin emerged in 2008 as a decentralized digital currency invented by an unknown person or group of people using the pseudonym Satoshi Nakamoto [1]. It used blockchain technology as a distributed ledger to record all transactions on the network. Smart contracts were first introduced by computer scientist Nick Szabo [2] in the mid-1990s and became widely used with the emergence of blockchain technology.

Smart contracts have emerged as a revolutionary technology that enables the automation of contractual agreements in a decentralized and secure manner. They are self-executing digital contracts that enable the automation of transactions and the enforcement of rules and regulations, without the need for intermediaries or third-party services. Smart contracts are being explored and implemented in a wide range of areas, including the Internet of Things (IoT), health care, security, finance, cloud computing, and artificial intelligence (AI).

In the area of IoT, smart contracts are being used to enable devices to autonomously carry out transactions and interact with other devices, without human intervention [3, 4]. Liu et al., who provided a solution to the problem that existing Ethereum contracts do not have the ability to communicate with the external IOT network, proposed an architecture for accessing off-chain data [5].Lone and Naaz, who said that IOT security should be provided by the combination of smart contracts and blockchain technologies, carried their research to the field of security [6]. Moreover, smart contracts can ensure the security and privacy of data exchanged between devices in IoT networks, by enabling the creation of secure and tamper-proof digital identities for devices. Smart contracts have the potential to enhance security in a wide range of applications, by enabling the creation of tamper-proof digital identities and the verification of transactions [7]. Studies on security of smart contracts are ongoing [8–14].

Since the number of data in finance and all other fields will reach the status of "big data" after a while, there is a need for cloud technologies. In cloud computing, smart contracts are being used to automate the management and allocation of cloud resources and enable the creation of decentralized cloud platforms. Working on systems where computations are verified, Reddy et al. have proposed protocols for different verifiable computing applications for smart contracts [15]. Smart contracts can enable users to automatically allocate resources based on their needs, and pay only for what they use. Furthermore, smart contracts can enable the creation of decentralized cloud platforms, which can increase the privacy and security of cloud computing services, by reducing the need for centralized servers.

AI is another area where smart contracts are being explored as a way to enable the automation of decision-making processes and the execution of actions based on predefined rules. Gupta et al. proposed to provide privacy protection with a combination of smart contracts and artificial intelligence [16]. This proposal also shows itself in the literature study by Raja et al., in their work on artificial intelligence-supported blockchain [17] and in Vacca et al. Additionally, smart contracts can enable the creation of decentralized AI platforms, which can increase the privacy and

security of AI applications, by reducing the need for centralized data storage and processing.

In the finance industry, smart contracts are being used to automate the execution of financial agreements, such as loans and insurance policies, and reduce the need for intermediaries. Smart contracts enable the creation of decentralized platforms for financial transactions, which can increase the speed and efficiency of financial transactions, while reducing transaction costs. Furthermore, smart contracts can enhance the security of financial transactions, by enabling the creation of tamper-proof digital identities for individuals and organizations, and the verification of transactions using cryptography. The credit management system for the finance field proposed by Wang et al. becomes very important when security is provided [18]. Additionally, smart contracts can enable the automation of compliance and regulatory requirements in the finance industry. One of the areas where the number of data is very high and confidentiality should be equally high is the health-care field. In health care, smart contracts are being explored as a way to securely store and share medical records and enable automated insurance claims processing [19]. Smart contracts can enable patients to have full control over their medical data, and securely share it with health-care providers and insurance companies. Furthermore, smart contracts can ensure that patients are automatically compensated for any medical procedures or treatments that they receive, without the need for lengthy and complicated claims processes. In the field of health, Kıyak et al. contributed by presenting the field of health in Turkish in smart contracts due to the scarcity of Turkish resources as a regional language [20]. Again, one of the many studies in the field of health is the studies of Staifi and Belguidoum [21]. They have implemented an application on smart home systems and home elderly care using smart contracts.

There are also studies conducted in many different areas on smart contracts [22–28]. Technological progress has been made with literature studies [29–34] which examines smart contracts and blockchain technologies extensively. Overall, smart contracts have the potential to transform a wide range of industries by enabling the automation of contractual agreements in a secure and decentralized manner. As such, they are an area of active research and development, with many potential applications yet to be explored. In the literature, there is a lack of collaborative contracts that work by interacting with existing smart contracts in the blockchain network. In this study, Collaborative Smart Contract Model is proposed to fill the gap in this area. Our propose platform in this study—Collaborative Smart Contracts—is a type of smart contract that allows multiple parties to participate in a transaction or agreement. In contrast, regular smart contracts are designed for two parties to execute an agreement.

In Section 2, the definition of the problem that has been solved, and the reasons for needing it are explained. Section 3 is on the introduction of the sample real-world problem from which the presented sample model is adapted. The model presented in Section 4 is introduced in general terms, and in Section 5, the application of the model on the problem is given. Finally, the study was completed by giving the results.

2 Problem definition

In smart contracts, a new process has started after the blockchain integration. This technology, which was not preferred due to the possibilities such as third-party intervention, theft, and alteration, has managed to become the new face of technology, with security being largely provided by blockchain. Research and development activities for smart contract technology continue rapidly. Smart contracts technology is mostly coded with the solidity programming language developed by the Ethereum founders. Solidity language has many shortcomings as it is still a developing language. The development of the solidity language also increases the number of new features that are planned to be added to the contracts. When a literature review on smart contracts technologies is made, it is seen that existing systems continue in a certain direction. A smart contract is coded in all its aspects in the designated area and published on the blockchain network. The biggest development for smart contracts is thought to be provided by the integration of artificial intelligence. However, before the integration of artificial intelligence, there are very important points that need to be developed for smart contracts. Figure 1 shows the working structure of current smart contracts. Each smart contract developer distributes their own smart contract to the blockchain network. However, none of them checks the existence of smart contract with the same coding as their own code in the blockchain network. Instead of interacting directly with the same existing contract, one more contract from the same is distributed to the blockchain network. In this case, the blockchain network turns into a garbage dump. This problem can be overcome with the help of a different smart contract that can interact with this contract already in the network. This requires a new collaborative model of smart contracts. In this study, a Collaborative Smart Contracts Model is proposed that enables smart contracts

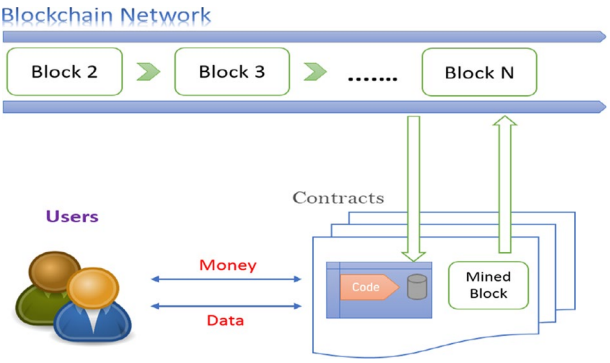


Fig. 1 How smart contracts work?

to work collaboratively in order to prevent unnecessary publication of contracts with the same content on the network, to facilitate adaptation to new technologies

by partitioning, to prevent code complexity, and to facilitate manageability. In Chapter 3, the introduction of the Collaborative Smart Contracts Model, which has been presented to the literature, its differences from other contracts, and its advantages are explained in detail.

3 Collaborative Smart Contracts model (CoSC)

Smart contracts have been developing rapidly since their integration with blockchain technology. Contracts, which quickly and continuously show themselves in new areas, are frequently encountered in the fields of IoT networks, finance, health, security, gaming, and logistics. In the literature review, it is seen that all the processes and methods of the developed smart contracts are created within the same contract. These created contracts are deployed in the blockchain network. The model proposed in this section enables the deployed contracts to interact with each other through the collaborative structure. In order to fully understand the proposed model, first of all, how a simple smart contract structure is, and the methods of accessing these contracts should be understood.

3.1 Simple smart contract structure

There are many parameters that build a smart contract. A general structure of a contract is given in Fig. 2. Every contract begins with the Pragma command. "Pragma solidity 0.4.8," in the first line code, determines which compiler version the contract will run on. Because solidity is a constantly evolving and changing language, it is important to choose the right compiler. After the version is selected, the contract is started to be written with the keyword "Contract." The content of the contract is developed according to the dynamics of the application to be made. The "Struct" command works as in other objectoriented languages. In the example in the figure, Clients with the same parameters can be developed with different features by keeping the information of the Client and then preventing the code repetition. Some of

```
1  pragma solidity ^0.4.8;                                    //Compiler Releases
2
3  contract Municipality{                                     //Contract command
4
5      struct Client {                                        //Struct Example
6      }
7
8      mapping(address => Client) public clientstructs;       //Mapping Example
9      address[] public propertyList;                         //Array Example
10
11
12     function Municipality() public payable{                //Constructor Example
13     }
14
15     function pay() payable external returns(bool success){ //Payable function Example
16     }
17
18 }
```

Fig. 2 Smart contract code architecture example

the features that distinguish smart contracts from others are the keywords "Payable" in the function on line 14, "mapping" in line 8, and "address" in line 9. The keyword "payable" indicates that the function is suitable for digital currency exchange. Digital money transfer can be made using the contract to the Pay function. "Mapping" is used to create a layout. In the example given, addresses are kept together by mapping them to the client. In other words, after the mapping process, the information of which client a called address belongs to is also provided. The keyword "address" is used to keep the addresses of the contracts in the blockchain network or the accounts that transfer digital money to the contracts and to access the information. At the same time, the constructor structure of the contracts can be created as shown in line 11. In solidity programming language, there are four access specifiers, internal, external, private, and public. Public, access type that can be seen and called both internally and externally; private, the type of access available only by contract; internal, the type of access that can only be accessed by internal or inherited contracts; and external, it is an access type that can be seen and accessed from the outside, but not from the inside. It is very important to determine the access types correctly in terms of security. If it is desired to reach a contract with an address in the Ethereum network, the required function or variable must be determined externally.

The developed model and dApp application were developed using solidity programming language. Ethereum's Remix IDE was used as the development and compilation environment.

3.2 Access methods to smart contracts

Access methods are a way for external parties to interact with a smart contract on a blockchain network. Access methods allow external parties to interact with these contracts by sending requests to the blockchain network, which then executes the code in the contract and returns the appropriate response.

There are several access methods that can be used to interact with smart contracts.

3.2.1 Upgradeable smart contracts

One of the biggest problems when creating Ethereum smart contracts is that the distributed contracts and transactions are immutable. This means that once a contract has been deployed on a network, its source code cannot be changed. This situation also puts the developers under pressure because this situation does not accept any mistakes. Smart contracts are vulnerable to zero-day exploits, as there will be no problem if the code is written completely healthy. For the same reason, smart contracts do not have updates such as software lifecycle. It is not possible to upgrade the contract code distributed on the blockchain network. However, when there is such a need, it is possible to set up a Proxy Contract Architecture to enable the use of distributed contracts. With the Proxy Contract, the address of the last distributed contract can be stored. This feature is a technology used to update the address of the new contract when the address of the contract changes. However, it is insufficient

since it does not have the feature of obtaining information from the content of the contract.

3.2.2 External function call by function signature

One of the basic components of the Ethereum network is the EVM (Ethereum Virtual Machine). The code of smart contract languages must be compiled via EVM bytecode in order to run. The EVM bytecode, which is a code executed on the EVM, interacts with the ABI (Application Binary Interface). For example, when coding a contract with the javascript language, the ABI executes an interface function that communicates between the EVM bytecode and javascript. ABI is an interface that defines a standard for how to call functions and retrieve data in a contract. EVM cannot directly execute code of high-level smart contract languages. These codes are compiled with the help of opcodes (low-level machine languages) and encoded in bytecode. In order to deploy the contract on the blockchain, the code must be compiled into ABI and bytecode (bin). Having the function signatures or actual code (ABI) of the contract in order to interact with a contract that is already distributed on the network is beneficial for a healthier interaction. However, if the ABI of these contracts is unknown, four methods can be applied.

Callcode Deprecated

Call It is used to execute the code of another contract.

Staticcall It is the same as a normal call. The only difference is that the called functions return when they are changed in any way.

Delegatecall In this function, another contract code is executed, but the difference is that the storage property is used in the called contract.

3.2.3 Interface

Interface also functions in the solidity programming language, similar to the tasks it takes in other object-oriented programming languages. The task of an Interface is to keep a description of the functions that one object uses to operate and execute the functions specified in another object. They are usually found on the top line of a contract with the "Interface" command. To call functions contained in another contract, it holds a signature of the required function or variable. Interface's function signatures end with semicolons and are in the form of a list. Interface reduces both duplacition and load by avoiding code complexity. There are also some restrictions on the Interface. They can inherit from other interfaces, but not from other contracts. An Interface function can only be of type "external." An interface cannot take a constructor and cannot have state variables. Interface and abstract contracts are similar in many ways and can be used interchangeably. In this study, an abstract contract structure with the same features is used instead of Interface to keep function signatures from another contract. Figure 3 shows the abstract contract (Interface) structure used in the application.

```
1   pragma solidity ^0.4.8;
2
3
4   contract Municipality
5   {
6       function isSellable() external returns (uint);
7   }
8
9   contract Bank
10  {
11      function GetBankResults() external returns (uint);
12  }
```

Fig. 3 Abstract contract (Interface) structure

3.3 Collaborative Smart Contract (CoSC) Model Structure

Collaborative Smart Contract (CoSC) is a model structure that enables multiple parties to participate in a smart contract, typically for complex or multi-party transactions. The CoSC model aims to provide a more collaborative, efficient, and transparent approach to executing these transactions.

3.3.1 CoSC framework overwiew

As open-source platforms develop themselves, smart contracts also need to be renewed and developed. A new model is needed in order to communicate with the contracts whose addresses are known in the network that has been deployed from a contract, and to access the functions and variables of these contracts and get the results.

In Fig. 4, the proposed model for smart contracts to work collaboratively is given.

In the example in Fig. 3, the signatures of the functions and variables to be called from two separate contracts (municipality and bank) are kept. The external structure

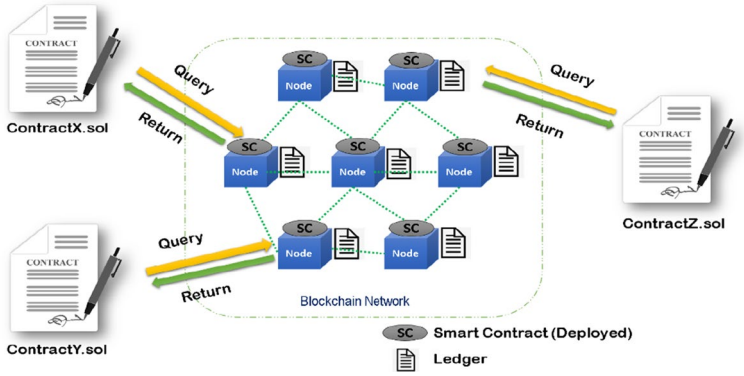


Fig. 4 Collaborative Smart Contracts

```
21
22  function getMunifrom () public returns (uint result){
23
24      address client = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
25      Municipality mp = Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138);
26
27      result=mp.isSellable();
28      setMuni(result);
29      return result;
30  }
31
32
```

Fig. 5 Calling deployed contract

provides access to the contract from outside. In the study, the Collaborative Smart Contract Model was created by using the Interface structure, one of the methods of accessing the contracts.

In the model in Fig. 4, contracts deployed to the blockchain network have an address in the network. Each contract in the form of "0x6cd... " has a unique address in the blockchain network. The functions and variables in these contracts can be accessed through their unique addresses. In Fig. 5, the code for reaching the contract with a unique address and calling the function of the contract is given. As shown in Fig. 5, the sample address of the Deployed Municipality contract in the network is kept in the mp object. With this object, the isSellable variable of the contract is called and the result is requested. The data received as a result of the call is returned. In the dApp application, a sample address of the client is kept. This address can be sent to the function in the municipality contract as a parameter, and the information of the client with this address can be obtained.

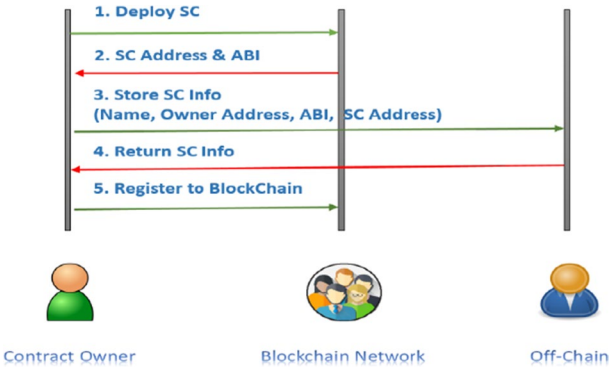


Fig. 6 SC registration

3.3.2 Smart contract registration procedure

A smart contract registration procedure provides a mechanism for smart contract holders. Thanks to this mechanism, smart contract owners can distribute their smart contracts in the blockchain network and keep all information about this contract off-chain.

Figure 6 shows the smart contract procedure and the stages of saving contract information. The owner of the contract writes the necessary code for the smart contract, compiles this code in a compilation environment, and distributes it to the blockchain network. When deploying this contract, it receives the contract address and ABI (other contract information). ABI information may include information such as identities and access information that have access to this agreement. The received information is stored off-chain for later transportation. As a final step, this contract is registered in the blockchain network by the owner. By keeping the version information of this contract, it is possible to update the contract address information, which will change with the update to be made in the contract code later on.

3.3.3 Interaction with deployed smart contracts

It is described as a procedure for interacting with a deployed contract in the blockchain network. Through this procedure, it enables the reuse of a contract that already exists in the blockchain network. This prevents the same code from being redeployed to the network. In this way, more effective use of the network is ensured. However, instead of writing all the code in a single contract, different contracts are enabled to interact, making it easier to act independently and manage the contracts. A change to a contract only affects that contract. Figure 7 shows the procedure for interacting with the deployed contract. The owners of the newly created contract, with their own identities, request the information of the contract with which they will interact. With this request reaching the blockchain network, the blockchain

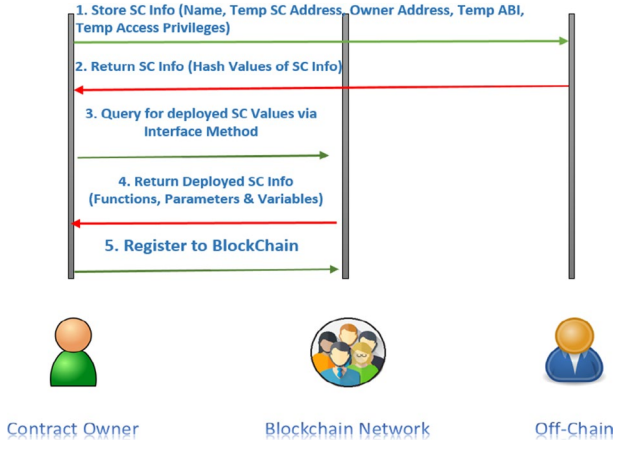


Fig. 7 SCs interaction procedure

records the access status of this identity from off-chain records. With the access privilege of the relevant identity, the requested information will be forwarded to the relevant user. After obtaining information about the contract with which it will interact, contract owners use this obtained contract information to call functions and variables of the relevant contract when coding their contract.

3.3.4 The advantages of the model

The proposed model is important for the development of smart contracts. Existing contracts are only suitable for uploading to the network and transferring digital money. In this study, there are many benefits such as calling the functions and variables of the contracts uploaded to the network and tracking their changes. Among the prominent advantages of the model; -Accessing a contract deployed on the blockchain network and using its functions -Preventing the code complexity -Reducing the cost by deploying contracts as a chapters (As a contract is deployed to the network, the cost of the fee increases as the line of code increases) -Providing modularity -Simplification of manageability -Facilitating integration with current technological developments. It is expected to provide many benefits as specified above.

Table 1 shows the comparison of Traditional Smart Contracts and Collaborative Smart Contracts in many different aspects such as security, cost, application, upgradeability, and complexity. These values show that the proposed system in this study, Collaborative Smart Contracts, is promising and ready to provide benefits in many ways.

The application of the Collaborative Smart Contracts model presented in Chapter 4 to a real-world problem (real estate sales process) is explained.

4 Real estate purchase and sales: literature review

Real estate purchase and sales is the process of handover of a house or immovable property. It is necessary to know exactly the transactions performed while performing a real estate purchase–sale process. During these processes, problems such as excessive paperwork, involvement of third parties, reliability problems, and inability to prevent fraud are encountered. For these reasons, it is necessary to real estate purchase and sale transactions to the digital environment within the scope of the conditions agreed by the parties. It is expected that this system will be transferred to a platform where the parties will not doubt security. It seems that the most suitable method for this system is to be created with smart contracts, which are built on blockchain and remove third parties and do not need a central server. Smart contracts, which have a place in many fields, have also attracted the attention of researchers in real estate purchase and sale transactions. Figure 8 describes a purchase and sale agreement.

There are studies in the literature on real estate purchase and sale. Sahai and Pandey have worked to integrate the land registry system into smart contracts in order to prevent fraud in India [35]. Panda et al. have transferred the real estate to the web with next.js and react tools, using smart contracts, in order to prevent fraud and

Table 1 Comparison of Traditional and Collaboratic Smart Contracts

Feature	Traditional smart contracts	Collaborative Smart Contracts
Definition	Self-executing contract with the terms of the agreement between buyer and seller being directly written into code and executed on a blockchain network	Contract that involves multiple parties who agree to work together toward a common goal, and the terms of the agreement are written into code and executed on a blockchain network
Security	Security vulnerabilities may exist in the code or platform	Security is enhanced through the collaborative nature and use of blockchain technology
Cost	Gas fees may be higher due to the need for more computational resources	Gas fees may be lower due to the ability to share resources and distribute costs
Application	Often used in simple transactions and financial applications	Can be used in a wide range of applications, including complex supply chains and multi-party agreements
Upgradeability	Difficult	Easy
Complexity	Limited in terms of complexity and scope	Can handle more complex use cases and transactions

Fig. 8 Purchase and sale agreement

money laundering incidents [36]. Soner et al. revealed the necessity of automating the land registry system in order to eliminate the problem of insecurity in the network and to contribute to the e-government system [37]. In their proposed model, Shinde et al. have developed a smart contract-based system that ensures that, when a land is purchased, a hard copy of the land purchase is given to the buyer by the government official [38]. There are also different studies on the need to carry out land registry transactions with smart contracts [39–41]. All these studies are very useful in terms of the development of smart contracts and their dissemination in the literature. However, smart contracts need to be developed with different models. The current studies in the literature are carried out entirely on a contract basis. The model proposed in this study was developed to enable the interaction of smart contracts that were coded and deployed in the blockchain network. In this study, this structure is named as collaborative smart contract structure in accordance with its functionality. In Chapter 3, this structure is introduced in detail.

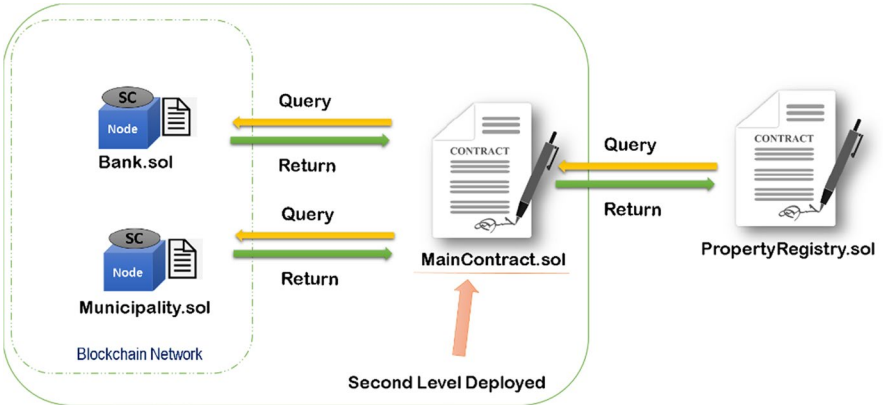


Fig. 9 Real estate purchase and sale—dApp

5 Application of CoSC model to real estate purchase and sale(s) agreement

Real estate is property consisting of land and the buildings on it, along with its natural resources such as minerals, water, or crops. Transfer of deed is process of handover a real estate or immovable property is called. At least three institutions must be visited for title deed transfer. In the example in this study, while a title deed is being transferred, smart contracts with different functions are coded for the municipality, bank, and land registry directorate.

The adaptation of the Collaborative Smart Contract Model to real estate purchase and sale transactions is presented in Fig. 9.

The application consists of four contracts acting in cooperation with each other. First, "Bank.sol" and "Municipality.sol" contracts were created and deployed to the blockchain network. In the second stage, the "MainContract.sol" contract is deployed in a way that calls the necessary functions from the two contracts deployed in the network and assigns the information it receives to the variable.

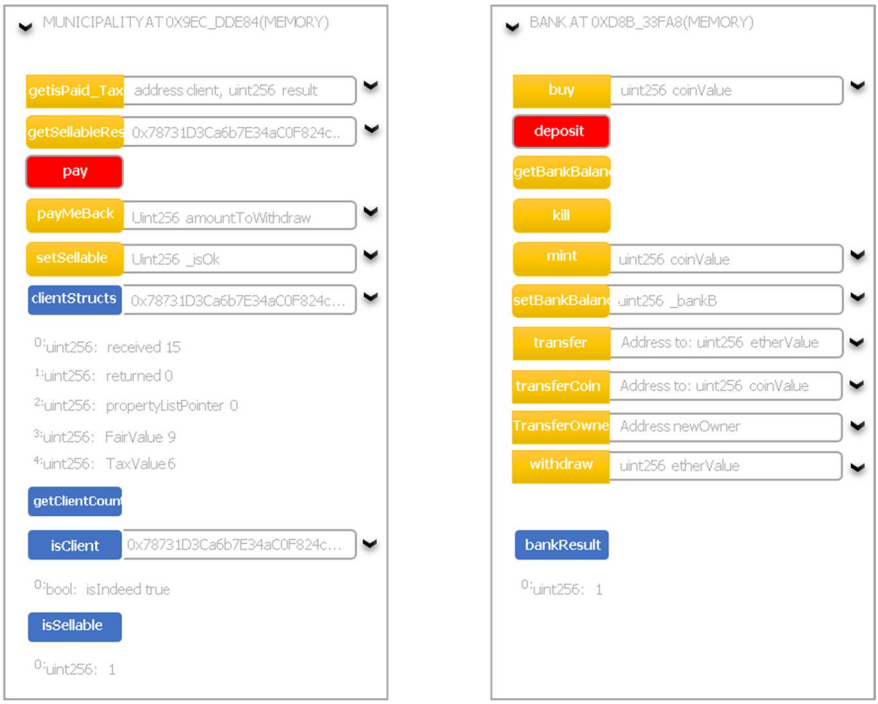
Finally, the "PropertyRegistry.sol" contract controls the data it receives from the "MainContrat.sol" contract and the conditions contained within it. If the result is suitable for the sale of the real estate, the deed transfer is allowed by the owner. If the result is not suitable, it is expected that the terms of the agreement will be fulfilled by informing the parties why it is not suitable for sale.

In Fig. 10, the contract code "MainContract.sol" interacting with the "Bank.sol" and "Municipality.sol" smart contracts in the blockchain network is given. While coding is done, first of all, it is done in a certain order as explained in Chapter 3;

```
pragma solidity ^0.4.8;

contract Municipality{ function isSellable() external returns (uint);} // variable signature of 'Municipality.sol'

contract Bank{function bankResults() external returns (uint);}         // function signature of 'Bank.sol'

contract calltheDeployedContracts
{
    uint public ok1;
    uint public ok2;
    address public owner;

    modifier OnlyOwner() {
        require(owner == msg.sender, "you are not owner");
        _;
    }
    function getMunifrom () public returns (uint resultMuni){

        //address client = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
        Municipality mp = Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138); // address of Municipality contract
        resultMuni=mp.isSellable();
        setMuni(resultMuni);
        return resultMuni;
    }

    function getBankfrom () public returns (uint resultBank){

        //address client2 = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
        Bank b = Bank(0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8);   //address of Bank contract
        resultBank=b.bankResults();
        setBankResult(resultBank);
        return resultBank;
    }

    function setMuni(uint _ok1) OnlyOwner {ok1=_ok1;}

    function setBankResult(uint _ok2) OnlyOwner {ok2=_ok2;}

}
```

Fig. 10 MainContract.sol

1. Function signatures of the information requested from municipality and bank smart contracts are encoded with abstract contract (Interface).
2. Then, the Modifier named OnlyOwner is coded to adapt the changes in the functions to the determined functions that only the contract owner can do.
3. Functions are created on the contract to be able to interact with the other two contracts.
4. The address of the contracts to be interacted with on the blockchain network is assigned to a variable of type ''address.''
5. Municipality mp= Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138) with the code, the data are transferred to the mp object via the address of the contract.
6. Finally, the functions or variables that are desired to be reached with the mp object are accessed with or without parameters.
7. Setter functions, on the other hand, are created so that the contract owner can decide that the conditions are met.



(a) Municipality Contract Interface. (b) Bank Contract Interface.

Fig. 11 Contract interfaces

5.1 dApp application results

The real estate purchase aggreement, implemented with the Collaborative Smart Contracts Model, was developed, compiled, and tested on the Ethereum network in the Remix IDE environment developed by the Ethereum founders for contract developers. In this section, the introductions of the interfaces of the developed application are given. A screenshot of the content of the "Municipality.sol" contract is given in Fig. 11a. Payment information of each customer is kept in the contract. In the figure, as an example, a customer's payment of 15 wei made on the contract has been made. This payment is taken as the sum of current market value and tax value. In the code, the price received

from the customer is recorded in the information of the customer. The isSellable variable is a variable that depends on the current market value and tax value within the contract. When the result of the isSellable value is queried with the collaborative smart contract logic from other contracts, the value of 1 is returned if the current, and tax values of a house have been paid by the customer, and 0 if not. A screenshot of the content of the "Bank.sol" contract is given in Fig. 11b.

In the bank smart contract, there are many transactions. Among them, there are features such as depositing, withdrawing, transferring money, buying coins with the money available in the bank, and viewing the bank balance. The bankResult variable in this contract can be called from outside using collaborative smart contracts. When querying the result of bankResult variable from other contracts, if there is a sufficient amount of credit withdrawal in the bank belonging to the customer and there is no foreclosure, etc., related to the relevant house, the result of this variable is returned as 1. If any of these values does not meet the required value, the bankResult value is 0. Screenshot of the content of the "MainContract.sol" contract is given in Fig. 12. The contract whose interface is given in Fig. 7 was created to work in cooperation with the "Bank.sol" and "Municipality.sol" smart contracts. In this contract, the signatures of other contracts are

Fig. 12 Main contract interface (Call the Contracts)

Table 2 Comparison of Traditional and Collaborative Smart Contracts in terms of execution time

#	Traditional smart contracts	Collaborative Smart Contracts
Average execution time	15 s	5 s
Maximum execution time	30 s	10 s
Minimum execution time	5 s	1 s
Standard deviation	4 s	2 s
Deployment time	50 s	30 s
Number of test cases run	100	100

kept with the interface method. In this way, the desired functions and variables of these contracts are queried through the objects of other contracts. In Fig. 12, the isSellable variable of the contract "Municipality.sol" has been queried and assigned to the variable isMuniOk, the bankResult variable of the contract "Bank.sol" has been queried and assigned to the variable isBankOk. If these two variables return 1, the house is in a sellable condition. If even one of these two variables is 0, an information message is returned to the user to correct the 0 value(s), that is, to make the payment. In the contracts shown in Fig. 11a and b, it is seen that the desired criteria are met, and the results return 1 accordingly. In this case, thanks to the collaborative structure in Fig. 12, inquiries were made from the other two contracts, and the same results were obtained. These results show that the Collaborative Smart Contract Model works well.

Table 2 shows the average execution times of Traditional Smart Contracts and Collaborative Smart Contracts as a result of 100 experiments on real estate application. These values also show that Collaborative Smart Contracts have about 20%+ better value in terms of execution time.

6 Conclusions

Smart contracts have revolutionized the way we conduct business transactions. These digital contracts use blockchain technology to automate and self-execute the terms and conditions of an agreement, providing a more secure, efficient, and transparent process for conducting transactions.

Collaborative Smart Contracts (CoSC) represent the next step in the evolution of smart contracts. By enabling multiple parties to access a shared ledger and collaborate in real-time, CoSC creates a more streamlined and efficient process for conducting complex transactions, such as buying and selling real estate. CoSC offers increased transparency and security, reducing the risk of fraud and misrepresentation, and eliminating the need for intermediaries, such as lawyers or notaries.

In conclusion, Collaborative Smart Contracts represent a significant innovation in the real estate industry, offering a more efficient, secure, and transparent process for conducting transactions. With continued research and development, CoSC has the

potential to transform the way we buy and sell real estate, streamlining the process and increasing trust and efficiency. As blockchain technology continues to evolve, the potential for Collaborative Smart Contracts in real estate transactions will only continue to grow, creating exciting opportunities for buyers, sellers, and the industry as a whole.

Authors' contributions TT wrote the manuscript with support from SB. TT developed the theory and performed the computations, planned and carried out the simulations. SB verified the analytical methods. SB was in charge of overall direction and planning. All authors contributed to the interpretation of the results, provided critical feedback, and helped shape the research, analysis, and manuscript.

Funding The authors have no relevant financial or non-financial interests to disclose.

Availability of data and materials Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical Approval The submitted work is original and is not published elsewhere in any form or language (partially or in full).

References

1. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. Decent Bus Rev. 31:21260
2. Szabo N, et al (1994) Smart contracts
3. Carreño R, Aguilar V, Pacheco D et al (2019) An iot expert system shell in block-chain technology with elm as inference engine. Int J InformTechnol Decision Making 18(01):87–104
4. Aguilera RC, Ortiz MP, Ortiz JP et al (2021) Internet of things expert system for smart cities using the blockchain technology. Fractals. 29(01):2150
5. Liu X, Muhammad K, Lloret J et al (2019) Elastic and cost-effective data carrier architecture for smart contract in blockchain. Future Generat Comput Syst 100:590–599
6. Lone AH, Naaz R (2021) Applicability of blockchain smart contracts in securing internet and iot: a systematic literature review. Comput Sci Rev 39(100):360
7. Demertzis K, Iliadis L, Tziritas N et al (2020) Anomaly detection via blockchained deep learning smart contracts in industry 4.0. Neural Comput Appl 32(23):17361–17378
8. Vacca A, Di Sorbo A, Visaggio CA et al (2021) A systematic literature review of blockchain and smart contract development: techniques, tools, and open challenges. J Syst Softw 174(110):891
9. Abou El Houda Z, Hafid AS, Khoukhi L (2019) Cochain-sc: An intra-and inter-domain ddos mitigation scheme based on blockchain using sdn and smart contract. IEEE Access 7:98893–98907
10. Huang Y, Bian Y, Li R et al (2019) Smart contract security: a software lifecycle perspective. IEEE Access 7:150184–150202
11. Moubarak J, Chamoun M, Filiol E (2020) On distributed ledgers security and illegal uses. Future Generat Comput Syst 113:183–195
12. Qian P, Liu Z, He Q et al (2020) Towards automated reentrancy detection for smart contracts based on sequential models. IEEE Access 8:19685–19695
13. Liao JW, Tsai TT, He CK et al (2019) Soliaudit: smart contract vulnerability assessment based on machine learning and fuzz testing. 2019 Sixth International Conference on Internet of Things: Systems. Management and Security (IOTSMS), IEEE, pp 458–465

14. Momeni P, Wang Y, Samavi R (2019) Machine learning model for smart contracts security analysis. 2019 17th International Conference on Privacy. Security and Trust (PST), IEEE, pp 1–6
15. Dorsala MR, Sastry V, Chapram S (2020) Fair payments for verifiable cloud services using smart contracts. Comput Secur 90(101):712
16. Gupta R, Tanwar S, Al-Turjman F et al (2020) Smart contract privacy protection using ai in cyber-physical systems: tools, techniques and challenges. IEEE access 8:24746–24772
17. Raja G, Manaswini Y, Vivekanandan GD, et al (2020) Ai-powered blockchain-a decentralized secure multiparty computation protocol for iov. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, pp 865–870
18. Wang H, Guo C, Cheng S (2019) Loc-a new financial loan management system based on smart contracts. Future Generat Comput Syst 100:648–655
19. Aguilera RC, Ortiz MP, Banda AA et al (2021) Blockchain cnn deep learning expert system for healthcare emergency. Fractals 29(06):2150227
20. Kiyak YS, Coskun O, Budakoglu İİ (2019) Blokzinciri, akilli kontratlar ve sağlik alanindaki üç uygulama örneği. Hacettepe Saglik Idaresi Dergisi 22(2):457–466
21. Staifi N, Belguidoum M (2021) Adapted smart home services based on smart contracts and service level agreements. Concurrency Comput: Practice Exper 33(23):e6208
22. Hasan HR, Salah K (2019) Combating deepfake videos using blockchain and smart contracts. IEEE Access 7:41596–41606
23. Kim Y, Pak D, Lee J (2019) Scanat: identification of bytecode-only smart contracts with multiple attribute tags. IEEE Access 7:98669–98683
24. Zheng W, Zheng Z, Chen X et al (2019) Nutbaas: A blockchain-as-a-service platform. IEEE Access 7:134422–134433
25. Xiong W, Xiong L (2019) Smart contract based data trading mode using blockchain and machine learning. IEEE Access 7:102331–102344
26. Zhang S, Lee JH (2019) Smart contract-based secure model for miner registration and block validation. IEEE Access 7:132087–132094
27. Lee SM, Park S, Park YB (2019) Formal specification technique in smart contract verification. In: 2019 International Conference on Platform Technology and Service (PlatCon), IEEE, pp 1–4
28. Agrawal TK, Kumar V, Pal R et al (2021) Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry. Comput Indust Eng 154(107):130
29. Marwala T, Xing B (2018) Blockchain and artificial intelligence. arXiv preprint arXiv:1802.04451
30. Du WD, Pan SL, Leidner DE et al (2019) Affordances, experimentation and actualization of fintech: A blockchain implementation study. J Strat Inform Syst 28(1):50–65
31. Wang X, Zha X, Ni W et al (2019) Survey on blockchain for internet of things. Comput Commun 136:10–29
32. Lu Y (2019) The blockchain: state-of-the-art and research challenges. J Ind Inform Integrat 15:80–90
33. Casino F, Dasaklis TK, Patsakis C (2019) A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telemat Inform 36:55–81
34. Timucin T, BİRO*heckG*UL S, (2021) A survey: Making smart contracts really smart. Transact Emerg Telecommun Technol 32(11):4338
35. Sahai A, Pandey R (2020) Smart contract definition for land registry in blockchain. In: 2020 IEEE 9th International conference on communication systems and network technologies (CSNT), IEEE, pp 230–235
36. Panda SK, Mohammad GB, Nandan Mohanty S et al (2021) Smart contract-based land registry system to reduce frauds and time delay. Secur Privacy 4(5):e172
37. Soner S, Litoriya R, Pandey P (2021) Exploring blockchain and smart contract technology for reliable and secure land registration and record management. Wireless Personal Commun 121(4):2495–2509
38. Shinde D, Padekar S, Raut S et al (2019) Land registry using blockchain-a survey of existing systems and proposing a feasible solution. 2019 5th International Conference On Computing. Communication, Control And Automation (ICCUBEA), IEEE, pp 1–6
39. Kumar P, Dhanush G, Srivatsa D, et al (2019) A buyer and seller's protocol via utilization of smart contracts using blockchain technology. In: International Conference on Advanced Informatics for Computing Research, Springer, pp 464–474

40. Joshi SM, Rajeswari K (2019) Efficient and accurate property title retrieval using ethereum blockchain. In: International Conference on Sustainable Communication Networks and Application, Springer, pp 424–438
41. Desic J, Lenac K (2020) Is blockchain technology the future of land registry digitalization? Zb Prav Fak Sveuc Rij 41:609