Blockchain based Secured Transactions in Rental Applications

[1]Sivaramakrishnan Rajendar, [2]Dharani T, [3]Rajasekaran Thangaraj, [4]Gowtham Kumar N, [5]Kavin A, [6]Lavanya M

[1,3]Assistant Professor, [2,4,5,6]UG Scholar

[1,2,3,4,5,6]SIG-CPS, Department of CSE, KPR Institute of Engineering and Technology, Coimbatore.

[1]sivaraamakrishnan2010@gmail.com, [2]dharaniarcuturus@gmail.com, [3]rajasekaran30@gmail.com, [4]16cs035@kpriet.ac.in, [5]a.kavin24@gmail.com, [6]lavanyamuthurayappan98@gmail.com

Abstract

The real estate industry is experiencing digital transformation in recent years. The historical paper-pencil business need to preserve archaic records and is inefficient. Also, the information sharing business processes are burdened with intermediaries, inefficiencies, and concerns about protection. Nowadays, the technology has already started to further reshape the growing global economy. In particular, blockchain technology is leading into this transition, similar to how other new technologies disrupt the established industries. This paper introduces a distributed-record keeping methodology using blockchain for housing rental system between tenants and property-owners. It ensures the authenticity of digital transactions and achieves peer-to-peer intermediate-free exchange of information. The proposed system is implemented using Hyperledger Sawtooth. During lease process, the smart contract retains all transactions, and the records of tenant and property-owners, thereby ensuring security for both.

Keywords – Hyperledger Fabric, Blockchain, Smart Contract

1. INTRODUCTION

1.1. Real estate/ Property booking

Property management is a difficult task. It involves property-owners, property managers, tenants, and vendors. The properties are handled through manual paper-pencil work or through a software. But, generally they aren't well integrated with each other. As the rate of real estate transactions are also increasing rapidly, it is essential than ever to have a common database of leases and purchases. By using a decentralized framework based on blockchain-smart contracts, the entire property management process can be carried out in a secure and transparent manner. This helps property-owner's and manager's to process payments, complete the credit background checks and handle ticketing for services. Distributed Ledger Technology (DLT) helps owners get a more transparent, comprehensive view of payment history and tenant background.

The property-owner and the tenant digitally sign a smart contract which comprises rental value, frequency of payment, and property details. Depending on the terms negotiated, the smart contract will automatically trigger the tenant's lease payments to the property-owner, as well as to any contractors conducting periodic maintenance. The smart contract can also be configured to automatically send the security deposit payment back to the tenant upon termination of the lease. This enables mortgage and tenancy contracts to be put into a blockchain to create agreements history and financial transactions that can be tracked and audited.

Hyperledger Fabric is an open source enterprise-grade permissioned DLT platform, developed for enterprise contexts to deliver some key capabilities over other popular platforms of the same nature. It acts as a neutral home for diverse distributed ledger architectures like Hyperledger Fabric, Indy, Sawtooth, as well as Hyperledger Caliper and Ursa libraries. It is clearly meant for the development of modular-architecture applications. The modular-architecture facilitates the diversity of business use cases through plug-in components like privacy, consensus, and membership services. This paper effectively uses Hyperledger Fabric Sawtooth for implementation purpose.

This paper is organized as follows. Section 2 introduces the background principles related to blockchain and proposed system. Section 3 describes the environmental setup. Section 4 explains the module implementation details of the entire system. Finally, Section 5 presents the conclusion.

2. Background

2.1. Ledger

A ledger consists of two distinct (but related) parts: a “blockchain” and the “state database”, also called “world state”. Unlike other ledgers, blockchains are immutable; that's, once a block has been added to the chain, it cannot be changed. In contrast, the world state may be a database containing this value of the set of key-value pairs that are added, modified or deleted by the set of validated and committed transactions within the blockchain. It is helpful to consider there being one logical ledger for every channel within the network. Figure 1 depicts a ledger.

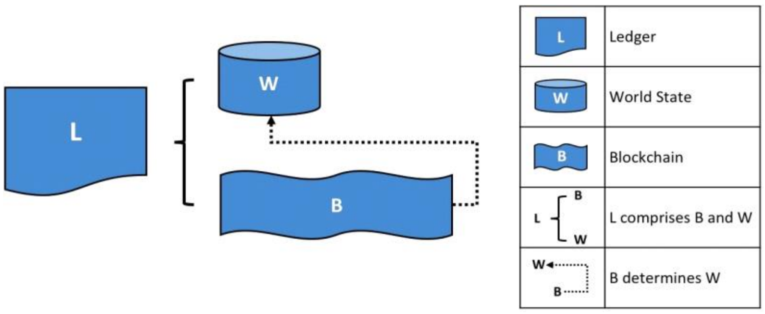


Figure 1. Ledger diagram

Peer node is a network entity that maintains a ledger and runs chaincode containers in order to perform read/write operations to the ledger. Peers are owned and maintained by members. Certificate Authority (CA) is a default Certificate Authority component, which issues PKI-based certificates to network member organizations and their users. In addition, a root certificate (rootCert) is issued to all members and an enrolment certificate (ECert) is issued to all authorized users. Ordering service is a laid out group of hubs that orders exchanges into a square so appropriates squares to associated peers for approval and submit. The requesting administration exists autonomous of the companion procedures and requests exchanges on a first-start things out serve reason for all channels on the system. It is intended to assist pluggable executions past the out-of-the-crate Kafka and Raft assortments. It is a regular authoritative for the overall system and contains the cryptographic character material attached to every member.

2.2. Smart contracts:

Smart contract (aka chaincode) is a wise contract code invoked by a client application external to the blockchain network that manages access and modifications to a group of key-value pairs within the world state via transaction. In Hyperledger Fabric, brilliant agreements are bundled as chaincode. Chaincode is introduced on peers so characterized and utilized on a minimum of one channel. To assist the steady update of data and to empower a whole host of record capacities (executing, questioning, then forth), a blockchain organize utilizes keen agreements to deliver controlled access to the record Savvy contracts are just a key system for embodying data and keeping it basic over the system, they're going to even be composed to permit members to execute certain parts of exchanges consequently.

2.3. Consensus:

The way toward keeping the record exchanges synchronized over the system — to confirm that records update just exchanges are affirmed by the reasonable members, which when records do refresh, they update with similar exchanges within the indistinguishable request is called consensus. Present day innovation has taken this procedure from stone tablets and paper organizers to hard drives and cloud stages, however the essential structure is that the equivalent. Bound together frameworks for coping with the personality of system members don't exist, increase provenance is so relentless it takes days to clear protections exchanges (the world volume of which is numbered inside the an enormous number of dollars), contracts must be marked and executed physically, and each database inside the framework contains one among a sort data thus speaks to 1 purpose of disappointment. It's outlandish with this cracked thanks to cope with data and procedure sharing to fabricate a rendezvous of record that traverses a business organize, while the requirements of perceivability and trust are clear.

2.4. Hyperledger Fabric:

Fabric is likewise a secluded and extensible open-source framework for sending and dealing permissioned blockchains and one on the whole the Hyperledger ventures facilitated by the Linux Foundation. Fabric is that the primary truly extensible blockchain system for running distributed applications. It underpins secluded agreement conventions, which allows the framework to be custom fitted to specific use cases and trust models. Fabric is furthermore the first blockchain framework that runs appropriated applications written in standard, universally useful programming dialects, without fundamental reliance on a region cryptographic money. Fabric realizes the permissioned model employing a conveyable notion of membership, which can be integrated with industry-standard identity management. To assist such adaptability, fabric presents an absolutely novel blockchain structure and patches up the way blockchains oblige non-determinism, asset weariness, and execution assaults. Table 1 presents the characteristics of Hyperledger Fabric.

Table 1. Characteristics of Hyperledger Fabric

S. No.	ATTRIBUTES	HYPERLEDGER FABRIC
1	Mode of Operation	Permissioned
2	Transaction Architecture	Execute-Order-Validate Model
3	Description of Platform	Highly Modular and Configurable
4	Governance	Private consortiums
5	Consensus	Multiple approaches but not limited to CFT, BFT.
6	Confidentiality & Privacy	Channels and Private Data Collections
7	Smart Contracts	Multi-language – Go, Java, NodeJs
8	Ledger DBMS	LevelDB, CouchDB
9	Performance	20,000 TPS

3. EXPERIMENTAL SETUP:

3.1. Software specifications:

Binaries	Hyperledger Fabric Binaries >= 2.1.0 Hyperledger Fabric CA Binaries >= 1.4.6
Github Repository	<https://github.com/hyperledger/fabric-samples/tr ee/v2.1.0>
Operating System	Ubuntu 18.04.3 LTS
Libraries	Git >=2.25.2 curl >=7.69.1 Docker >=19.03.5 Docker-Compose >=1.25.4 Go >=go1.14.1 NodeJS >= v12.16.1 Python == 2.7

Docker Images	hyperledger/fabric-peer:latest hyperledger/fabric-orderer:latest hyperledger/fabric-ca:latest hyperledger/fabric-baseos:2.1 hyperledger/fabric-ccenv:2.1 hyperledger/fabric-tools:2.1 hyperledger/fabric-nodeenv:2.1 hyperledger/fabric-couchdb:0.4.18
Network	wi-fi
Speed	1.1GHz

3.2. Hardware specifications:

RAM	4 GB
Keyboard	Standard windows keyboard
Mouse	Two or three button mouse
Hard disk	20 GB

3.3. Setting up environmental variables:

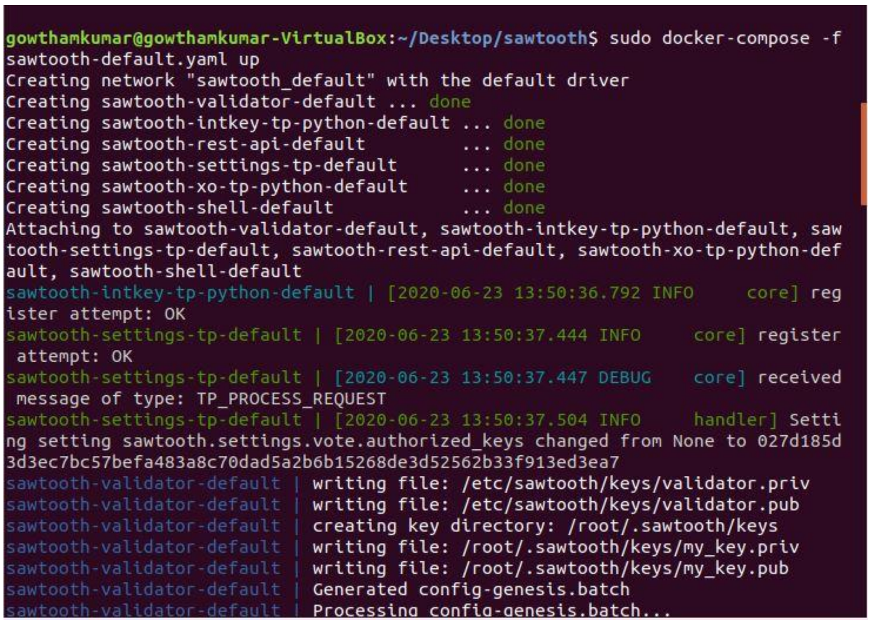
Property Booking Blockchain uses the Go Programming Language for many of its components. There are two environment variables user will need to set properly. User can make these settings permanent by placing them in the appropriate startup file.

First, user must set the environment variable GOPATH to point at the Go workspace containing the downloaded Fabric code base.

export GOPATH=$HOME/go

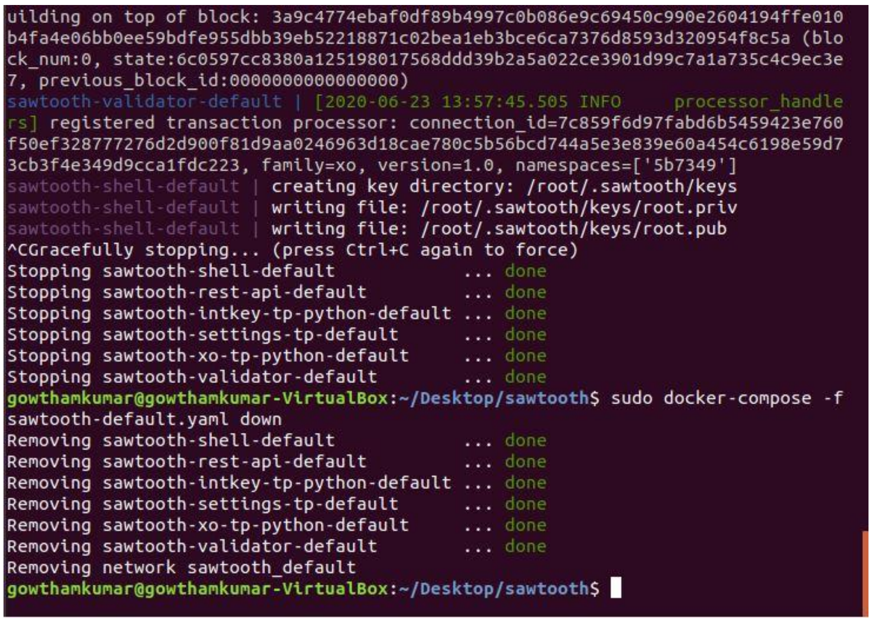
Second, user should (again, in the appropriate startup file) extend your Shell search path to include the Go bin directory.

export PATH=$PATH:$GOPATH/bin



```
gowthamkumar@gowthamkumar-VirtualBox:~/Desktop/sawtooth$ sudo docker-compose -f
sawtooth-default.yaml up
Creating network "sawtooth_default" with the default driver
Creating sawtooth-validator-default ... done
Creating sawtooth-intkey-tp-python-default ... done
Creating sawtooth-rest-api-default         ... done
Creating sawtooth-settings-tp-default      ... done
Creating sawtooth-xo-tp-python-default     ... done
Creating sawtooth-shell-default            ... done
Attaching to sawtooth-validator-default, sawtooth-intkey-tp-python-default, saw
tooth-settings-tp-default, sawtooth-rest-api-default, sawtooth-xo-tp-python-def
ault, sawtooth-shell-default
sawtooth-intkey-tp-python-default | [2020-06-23 13:50:36.792 INFO     core] reg
ister attempt: OK
sawtooth-settings-tp-default | [2020-06-23 13:50:37.444 INFO     core] register
 attempt: OK
sawtooth-settings-tp-default | [2020-06-23 13:50:37.447 DEBUG    core] received
 message of type: TP_PROCESS_REQUEST
sawtooth-settings-tp-default | [2020-06-23 13:50:37.504 INFO     handler] Setti
ng setting sawtooth.settings.vote.authorized_keys changed from None to 027d185d
3d3ec7bc57befa483a8c70dad5a2b6b15268de3d52562b33f913ed3ea7
sawtooth-validator-default | writing file: /etc/sawtooth/keys/validator.priv
sawtooth-validator-default | writing file: /etc/sawtooth/keys/validator.pub
sawtooth-validator-default | creating key directory: /root/.sawtooth/keys
sawtooth-validator-default | writing file: /root/.sawtooth/keys/my_key.priv
sawtooth-validator-default | writing file: /root/.sawtooth/keys/my_key.pub
sawtooth-validator-default | Generated config-genesis.batch
sawtooth-validator-default | Processing config-genesis.batch...
```

Figure 2. Environment variable GOPATH



```
uilding on top of block: 3a9c4774ebaf0df89b4997c0b086e9c69450c990e2604194ffe010
b4fa4e06bb0ee59bdfe955dbb39eb52218871c02bea1eb3bce6ca7376d8593d320954f8c5a (blo
ck_num:0, state:6c0597cc8380a125198017568ddd39b2a5a022ce3901d99c7a1a735c4c9ec3e
7, previous_block_id:0000000000000000)
sawtooth-validator-default | [2020-06-23 13:57:45.505 INFO     processor_handle
rs] registered transaction processor: connection_id=7c859f6d97fabd6b5459423e760
f50ef328777276d2d900f81d9aa0246963d18cae780c5b56bcd744a5e3e839e60a454c6198e59d7
3cb3f4e349d9cca1fdc223, family=xo, version=1.0, namespaces=['5b7349']
sawtooth-shell-default | creating key directory: /root/.sawtooth/keys
sawtooth-shell-default | writing file: /root/.sawtooth/keys/root.priv
sawtooth-shell-default | writing file: /root/.sawtooth/keys/root.pub
^CGracefully stopping... (press Ctrl+C again to force)
Stopping sawtooth-shell-default            ... done
Stopping sawtooth-rest-api-default         ... done
Stopping sawtooth-intkey-tp-python-default ... done
Stopping sawtooth-settings-tp-default      ... done
Stopping sawtooth-xo-tp-python-default     ... done
Stopping sawtooth-validator-default        ... done
gowthamkumar@gowthamkumar-VirtualBox:~/Desktop/sawtooth$ sudo docker-compose -f
sawtooth-default.yaml down
Removing sawtooth-shell-default            ... done
Removing sawtooth-rest-api-default         ... done
Removing sawtooth-intkey-tp-python-default ... done
Removing sawtooth-settings-tp-default      ... done
Removing sawtooth-xo-tp-python-default     ... done
Removing sawtooth-validator-default        ... done
Removing network sawtooth_default
gowthamkumar@gowthamkumar-VirtualBox:~/Desktop/sawtooth$
```

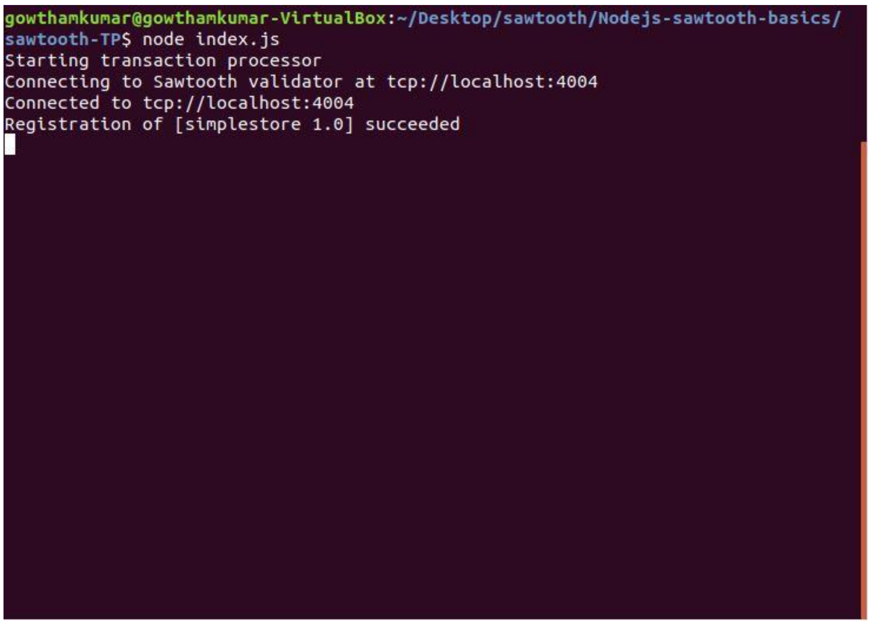
Figure 3. Shell search path to include the Go bin directory

Third, If the Chaincode and applications for property booking blockchain leveraging the Hyperledger Fabric SDK for Node.js are developed in NodeJS, then their path must also be included.

export PATH=$PATH:$HOME/nodejs/bin

Finally, User may want to add that to PATH environment variable so that platform-specific can be picked up without fully qualifying the path to each binary.

export PATH=<path to download location>/bin:$PATH



```
gowthamkumar@gowthamkumar-VirtualBox:~/Desktop/sawtooth/Nodejs-sawtooth-basics/
sawtooth-TP$ node index.js
Starting transaction processor
Connecting to Sawtooth validator at tcp://localhost:4004
Connected to tcp://localhost:4004
Registration of [simplestore 1.0] succeeded
```

Figure 4. Registration diagram

4. SYSTEM IMPLEMENTATION

The first step in the implementation process is to define and startup the Property Booking network. The second step is creating and joining a channel to the network followed by creating Certificate Authorities and creating World State Database. Subsequently, the third step defines the business logic within the variety of Smart Contracts which is to be packaged, installed, approved and committed by respective organizations within the network. Finally, Client applications have to be developed which is employed to interact with the Property Booking Network to book the property in rental solutions.

4.1. MODULE-1

Setting Up Certificate Authorities:

To participate within the blockchain network, Identity is critical for each Participant of the Network. CA provides identity within the sort of X.509 Digital Certificates. Administrator of CA signs certificates & issue to the requesting user in a very network. Member Organizations similarly as Orderer Organizations must have a separate CA.

CA Configuration is defined in a very docker compose file and given below. Image & Container name of CA Environmental Variables for CA Specific Port to CA Includes name of network & command to start out. Finally, mount volume for CA.

Setting Up World State Database:

Rental Applications Network uses the CouchDB because the state database. It models data as JSON. It supports to issue rich queries against data values instead of the keys. Supports indexes to form queries more efficient & enables to question large datasets. CouchDB Configuration is defined in a very docker compose file and given below. CouchDB Username & Password Image & Container name of CouchDB Environmental Variables for CouchDB Specific Port to CouchDB Includes name of network & command to begin. Finally, specifying instances for each peer.

4.2. MODULE-2

Defining & Startup the Rental Applications Network:

- **Defining the Network:**

The Network consists of the subsequent components: Ledger. One per channel. Smart contract (aka chaincode) ,Peer nodes, Ordering service, Channel ,Certificate Authority. The number of Peer Nodes & Orderers together with all configurations must be specified. Ledger & Certificate Authorities must be configured before remarking the network.

- **Startup the Network:**

Following is that the command to startup the network:

```
./network down

./network up -ca -s couchdb
```

The startup command must include flags for Certificate Authorities and therefore the State Database to be employed by the network. Note: Best practice is usually bring down the network before starting because it removes all containers created at previous runs if it exists.

4.3. MODULE-3

Creating & Joining a Channel to Network:

One key a part of the Rental Solution Network – a channel. It is a primary communications mechanism by which the members of a consortium can communicate with one another. There will be multiple channels during a network.

Command to make & join a channel:

```
./network up createChannel <<channelName>>
```

```
Peer channel join -b ./channel-artifacts/$CHANNEL_NAME.block.
```

4.4. MODULE-4

Developing Smart Contracts:

- **Smart Contracts:**

It defines the various states of a business object. It governs the processes that move the article between these different states. It permit modelers and keen agreement designers to characterize the key business procedures and data.

It is shared by the various network participants, like property owners and guest users. The same version of the smart contract must be utilized by all applications connected to the network.

- **Lifecycle of Rental Solution:**

States and Transactions are two important concepts in rental solutions. Below are conceptual objects useful, modeled as states, whose lifecycle transitions are described by transactions.

- **Transactions:**

Transactions defines the business logic of the network. Every transactions defined in an exceedingly network decides the lifecycle of rental solution.

Once the Smart Contract is developed, it must be deployed to the channel to be availed by the client applications. More number of smart contracts are combined into one chaincode file. A chaincode is deployed to a channel employing a process referred to as the Chaincode Lifecycle. End users interact with the blockchain ledger by invoking the deployed smart contracts.

- **Lifecycle of Chaincode:**

Deploying the chaincode to channel falls under the subsequent four steps:

Step one: Package the smart contract

Step two: Install the chaincode package

Step three: Approve a chaincode definition

Step four: Submitting the chaincode definition to the channel.

Only the administrator of the network have a permission to deploy a chaincode a channel.

4.5. MODULE-5

Developing Client Applications:

An application can interact with a blockchain network by submitting transactions to a ledger or querying ledger content using the material SDK. An application should follow six basic steps to submit a transaction, Select an identity from a wallet hook up with a gateway Access the required network. Construct a transaction request for a sensible contract Submit the transaction to the network Process the response.

5. CONCLUSION

Blockchain technology is widely adopted due to its decentralized, robust, and glassy nature. It works on a distributed ledger, yet secured to record and trace information in a decentralized network.

Blockchain-based rental solution is a paramount in the applications of blockchain. The proposed system uses the blockchain distributed ledger to list property information and eliminates the intermediary fees. The system experiences the real peer-to-peer listings. It effectively uses Sawtooth during the rental process between the tenant and the property-owner. The best part is that the contract can run only in the way it was initially set. Moreover, the contract retains transaction history and records between both parties to ensure security.

REFERENCES

1. Fanning, Kurt, and David P. Centers. "Blockchain and its coming impact on financial services." *Journal of Corporate Accounting & Finance.* 27.5 (2016): 53-57.
2. Eyal, Ittay. "Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities." *Computer* 50.9 (2017): 38-49.
3. Schaub, Alexander, et al. "A trustless privacy-preserving reputation system." *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, Cham, 2016.
4. Lo, Yuen C., and Francesca Medda. "Bitcoin mining: converting computing power into cash flow." *Applied Economics Letters* 26.14 (2019): 1171-1176.
5. Tabane, Elias, Seleman M. Ngwira, and Tranos Zuva. "Survey of smart city initiatives towards urbanization." *2016 International Conference on Advances in Computing and Communication Engineering (ICACCE)*. IEEE, 2016.
6. Guizani, Mohsen, et al. "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies." *IEEE Communications Surveys & Tutorials* (2018).
7. Stanciu, Alexandru. "Blockchain based distributed control system for edge computing." *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2017.
8. Niranjanamurthy, M., B. N. Nithya, and S. Jagannatha. "Analysis of blockchain technology: pros, cons and SWOT." *Cluster Computing* 22.6 (2019
9. Li, Xiaoqi, et al. "A survey on the security of blockchain systems." *Future Generation Computer Systems* (2017).
10. Tapscott, Don, and Alex Tapscott. "How blockchain will change organizations." *MIT Sloan Management Review* 58.2 (2017).