Securing Farm Insurance Using a Private-Permissioned Blockchain Driven by Hyperledger Fabric and IPFS

Nishat Tasnim Haque, Zerin Tasnim, Ananya Roy Chowdhury, and Saha Reno

1 Introduction

The majority of the population in Bangladesh depends on farming. Livestock farming is an integral element of our country's agriculture production system, which provides multiple sources of food, employment opportunities, agricultural development, and other services. However, it is concerning that a larger percentage of them are not protected by a safe insurance system that will offer financial security or compensation for damages. As a result, they are unable to recover from the immense loss they experience in the event of incidents like fires or natural disasters. An insurance management system that records crucial data about a farm, its owner, loss history, etc., is needed to ensure that livestock farmers can claim refunds in case of mishaps or unforeseen events. And on account of this, it will be effortless to provide the insurance they require based on their circumstances and to identify fraudulent insurance claimants.

Various authors have provided numerous ways to secure insurance management; however, the constraints in their systems prevent real-world use of these approaches. Raikwar and his co-authors suggested a permissioned blockchain-based approach [8] for insurance processes using Hyperledger Fabric v1.0.0-beta. Golanf v1.8 was used to create the chaincode. However, their database is not encrypted, and each transaction cannot have its own set of endorsing peers since smart contracts are not implemented at the transactional level. Additionally, the confirmation time will increase with node count, making the network slower. The authors of "NEO smart contract for drought-based insurance" [7] developed a system for South-East Asian (SEA) countries where small households experienced insufficient crop production due to heavy drought. The system was built on the NEO blockchain technology and runs on NeoVM. Given that this system's consensus protocol lacks a fallback mechanism, the

N. T. Haque · Z. Tasnim · A. R. Chowdhury · S. Reno (✉)
Bangladesh Army International University of Science and Technology, Cumilla, Bangladesh
e-mail: reno.saha39@gmail.com

J. K. Mandal et al. (eds.), *Proceedings of International Conference on Network Security and Blockchain Technology*, Lecture Notes in Networks and Systems 738,
https://doi.org/10.1007/978-981-99-4433-0_29

consensus must be forcibly reset when the code cannot reach consensus. NEO consensus relies on bookkeeper nodes that stop trying to come to an understanding when the time difference between the timestamp of the previous block and the present time exceeds 10 min. "Application of smart contracts based on the Ethereum blockchain for the purpose of insurance services" [1] proposed an Ethereum blockchain-based solution for insurance processes. They created a decentralized crypto-token based on the ECR20 smart-token standard. The problem with their approach is that they automated the handling of insurance claims using a public, permission-free blockchain, which could cause problems with real-time confirmation. On top of that, Ethereum employs a computationally intensive consensus process known as proof-of-work (POW), which has a significant scalability issue that restricts it to about 20 transactions per second.

In this study, we developed a system that handles applicants' information for insurance companies in the safest and most practical way possible using the hyperledger fabric blockchain. It's a special form of a private blockchain that enables the transaction of an asset or its state to be consented upon, maintained, or observed by only the members of a permissioned group [5]. Despite the fact that it is built on the idea of a distributed or shared ledger, each participant must be verified. This assists in concealing sensitive data and prevents data manipulation (i.e., creating new data, updating old data, or deleting existing data). Hyperledger fabric does not require consensus techniques like the computationally demanding proof of work (POW) or proof of stake (POS) used in blockchain networks because it is a private and permissioned network [2]. Scalability, transaction speed, and overall network performance are all boosted by doing this [9]. Last but not least, hyperledger fabric gives users the option to alter the entire blockchain's underlying infrastructure. Our proposed system offers

(i) Information access control within a permissioned group. Outsiders are not permitted to change the data; only the allowed participants can add new assets, along with updating, deleting, and searching the existing information.
(ii) An insured is only permitted to read his/her transaction information in order to prevent data manipulation.
(iii) Insurance companies can look up information about an insured by the asset ID, which is a unique identification number that prevents them from claiming insurance to which they are not entitled.

2 Literature Review

Over the past several years, blockchain research has gained popularity. Numerous people have introduced their outstanding efforts to address significant issues regarding insurance with a blockchain-based solution. Loukil and others proposed CioSy, a collaborative blockchain-based insurance system for monitoring and processing insurance transactions [6]. Through the use of three smart contracts, they created

a community of insurers, acquired insurance from the insurer, submitted claims in accordance with the insurance contract, and automated the processing of claims and refund payments. This system, though, works best for goods with extremely low risk expectations. Premiums for claims in the event of sudden catastrophes, like natural disasters, may be woefully inadequate.

The MIStore is a medical insurance storage system [12] by Lijing and his co-authors that achieves several unique features, including decentralization, secure data storage, threshold, efficient verification, and efficient homomorphic computation. The tamper-resistance of the blockchain provides users with great trust; and because it is decentralized, users may connect with one another without the use of intermediaries. The system's drawbacks include that it utilized the Ethereum blockchain approach, which presented challenges with scalability due to its multipurpose ledger and complex programming language.

In [11], by Iman and others, a new framework is presented for securing a cyber-product using blockchain technology. To share the risk of insurance, they have applied for crowdfunding through a sealed-bid auction process. The binding property ensures that a commitment cannot be opened to another value, whereas hiding demands that a commitment not reveal any information about the committed value. They used the Ethereum-SHA3 hash function (Keccak256) as the commitment scheme for their implementation. The drawback of this work is that the price of Ether has fluctuated dramatically in the past, so beginners may find it risky to invest in this currency.

A framework for using smart contracts and storing them in blockchain for insurance contracts is presented in [3] by Abid and his co-authors. The proof of authority (PoA) consensus algorithm is used to verify the transactions, and it rejects any invalid transaction requests. Any incorrect endorsement will be obvious and detectable; the algorithm operates on both private and public networks. A drawback of smart contracts is that they significantly limit the volume of transactions that the network can handle in a given second.

A solution [10] by Rui and others proposed that prevents specific sorts of fraud in the field of vehicle insurance was created using blockchain and smart contracts. The double dipping fraud arrangement is the use case for the created Ethereum-based blockchain system to prevent insurance fraud. Drawback is that they are now in the prototyping stage of their suggested solution. It has been tested using data from hypothetical insurance companies, clients, and vehicles, and it has shown to be effective. However, expanding the solution's capabilities and developing a finished product are required.

Nishant and others have developed a blockchain-based crop insurance for Indian farmers [4]. Due to the fact that latest blockchain-based frameworks are built on the idea of smart contracts, they offer a tamper-proof environment where a transaction is only carried out after receiving validated data, and no bad user is able to alter the system. When the instances were upgraded, the throughput rose to 220. When the CPU load was 70%, RAM consumption reached 80%. Drawback is, since this system doesn't directly employ any cryptocurrencies like Bitcoin, system failure in unstable cryptocurrency markets may be more likely.

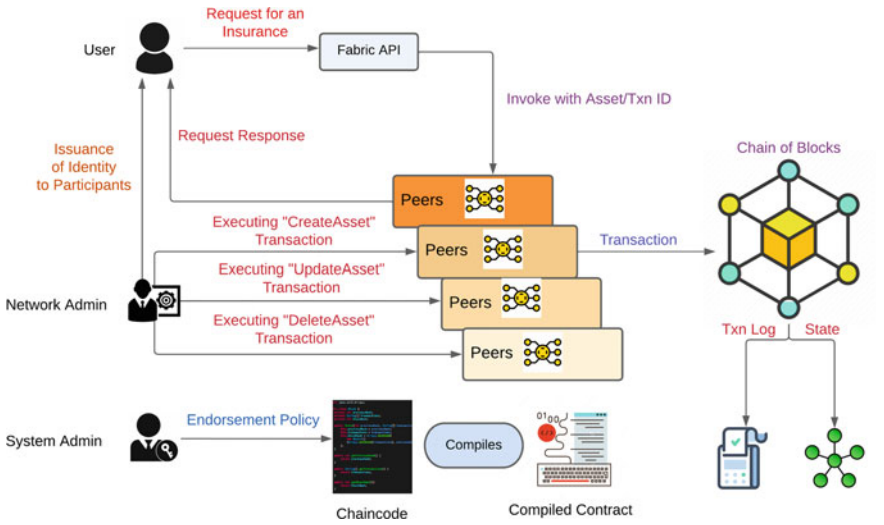


Fig. 1 Basic architecture of the proposed system

Our system overcomes the aforementioned shortcomings as: (i) it does not require any transaction fee, (ii) has better throughput rate (0.021 s per transaction), (iii) supports querying the ledger using SQL and (iv) has a robust access control module.

3 Methodology

Our chaincodes, written in JavaScript, have been installed in Hyperledger Fabric 2.3.1. Only those users who are identified in the system's asset identification field can access the document information. There must be an electronic copy of every document that is connected to the information provided by the insured. So, IPFS stores the images of the documents by cryptographically hashing them, creating a unique content identifier (CID) for each image. Additionally, they are linked to the appropriate assets. By doing this, we can instantly detect any altered or destroyed images, given that the accompanying asset's hash will likewise change. Any time an asset is created, added to, updated, or removed from a block, a transaction involving that asset is recorded. Nobody can effectively alter or modify the information provided in the system in this way without notifying others. The main concept of our system is illustrated in Fig. 1.

3.1 Resources of the Network

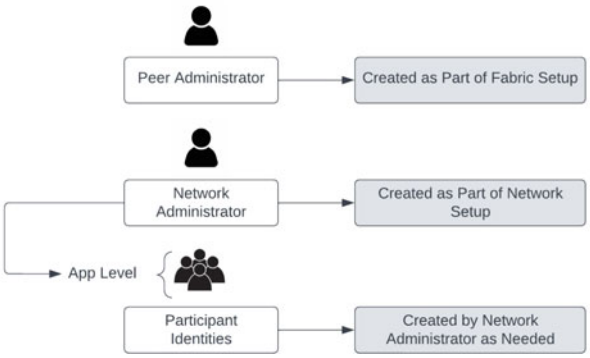
Any object that has information is a resource in the Hyperledger Fabric and is called an asset. Details included within an asset can be seen as key value pairs, which signify the value of anything, from the tangible to the intangible. With hyperledger fabric, we can manipulate these assets via chaincode-dependent transactions. In our work, each farm is considered an asset, and for each of these assets, there are 11 attributes which correspond to different relevant and unique information about the farms.

An asset of our private blockchain, which is built on hyperledger fabric, is characterized by these attributes. A new asset ID is constructed each time information regarding a new farm is provided, permitting the ledger to be queried for that person's farm information. Only by performing transactions is it possible to modify and update this information in future. Participants in our system fall into one of three categories: (i) system admin, (ii) network admin, and (iii) user. The users in this scenario are farm owners who may only request information updates and read information about their insurance transactions. Admin has broader access and control over system resources than users have, which is covered in more detail in the following subsections. The configuration setup is done by Peer Admin; similarly identities to new participants are appointed by Network Admin. The relation between assets and participants is depicted in Fig. 2.

3.2 Initiating and Processing Transactions

A private blockchain is a distributed ledger that operates as a closed database secured with cryptographic concepts and the organization's needs. Only those with permission can run a full node, make transactions, or validate/authenticate the blockchain changes. The changes needed to be made in our blockchain are related to the creation of insurance, which will require reshaping of the assets. After each transaction is recorded, in order to change any record, 51% of the participants must reach consensus. The farm's hash from IPFS is stored so that the insurer can access this information at any time and update as needed. These updates can also be verified by the asset manager to make sure all the alterations are correct. Transactions are additionally employed to store the farm's IPFS hash, permitting the farm owners to access their farm's data. When a user performs a transaction, the asset ID is associated with the transaction's transaction ID. Owing to the immutability of blockchain transactions, asset managers are given access to an unalterable log that they can audit at any time in future. Asset managers utilize this transactional data to securely look at information, including earlier iterations of asset records and what modifications have been made to the assets.

(a) Assets and Participants



(b) Administrators

Fig. 2 Resources required for the proposed system

3.3 Deployment of Transactions Inside Historian Registry

The registry carrying the records of transactions is known as the Historian Registry. The Historian Registry is updated after each transaction is submitted and verified to be correct. And thus, a complete history of every transaction made within the network is created. In this record, along with the transaction history, information about who made the transaction and their identities are also preserved. A report of this Historian Registry is created in hyperledger fabric that includes transaction information such as timestamp, initiator ID, details of the changes made and a list of successful transactions that occurred. The Historian Registry cannot be viewed by anyone; only people with permissioned access and proper roles can inspect it. The declaration and definition of the queries are contained in the query document. Both untampered information and assets are subjected to queries. The Historian Registry and Asset Registry demonstrate the modifications recorded about the assets when a transaction is reported, along with the participant information and their identities

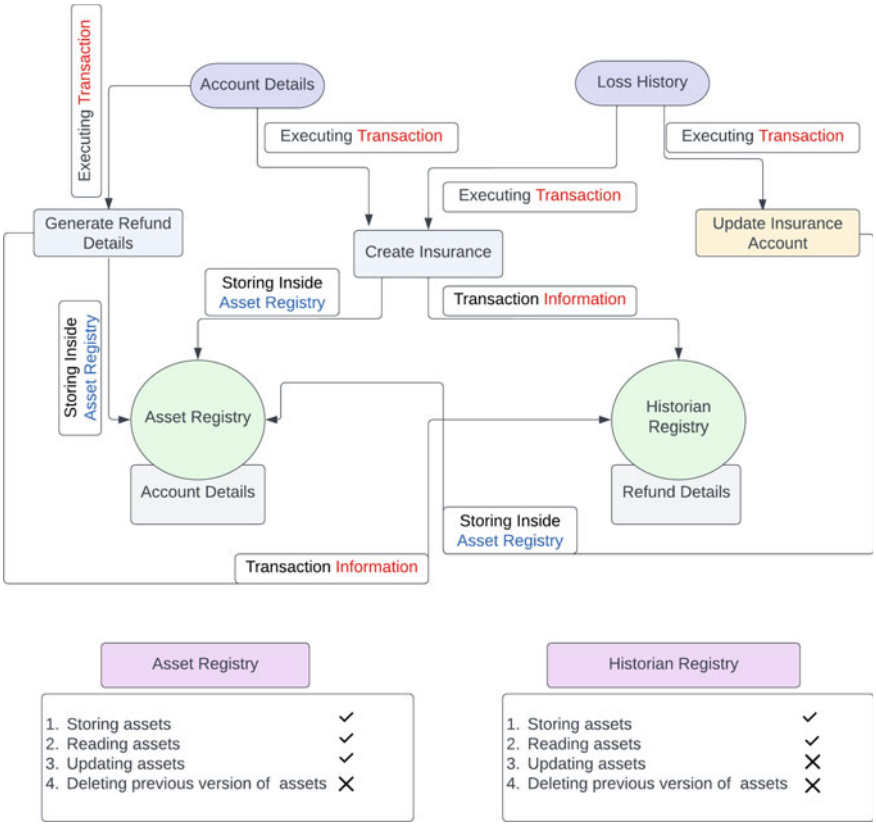


Fig. 3 Functionalities of asset and historian registries

in reporting such transactions. The Historian Registry's listing does not include the rejected transactions. Correspondingly, the Fabric runtime also refers to a number of configuration-related tasks it performs as transactions. The transactions which we identified and sought to conduct in our system can be divided into the following, (i) CreateAsset, (ii) ReadAsset, (iii) UpdateAsset, (iv) DeleteAsset, (v) AssetExists, (vi) TransferAsset, (vii) GetAllAssets, (viii) SearchAssets. Additionally, IPFS hash is linked to the assets. We may access the Historian Registry to obtain all of our private blockchain transactions, which are all preserved in blocks. In conjunction with asset ID, the Historian Registry is also categorized by asset ID, which is responsible for carrying out the transaction, and transaction genre. Like traditional databases, hyperledger fabric facilitates queries, by utilizing this feature inside the Historian Registry, any transaction's status may be confirmed. Figure 3 shows the implementation of queries to acquire asset, transaction, and heavyweight data from IPFS.

3.4 Regulating Access Using Access Control Module

The hyperledger fabric is a type of private blockchain that has access control as a unique feature. It refers to a script that lists the policies of a specific company. This feature allows our system to restrict a participant's access to the system's resources according to their designation and assigned function. We can limit the use of diverse systems utilizing create, read, update and delete (CRUD) procedures because of the extent of customizability. For instance, an insured can retrieve an insurance cover note from Asset Registry and view what alterations were made to the document from Historian Registry. They cannot, however, alter the documents directly. They are permitted to use the "RequestModification" transaction to update any inaccurate information in the document. The peer and network administrators, on the other hand, are eligible to update the assets since they are granted permissions in the access control script. The Historian Registry cannot be destroyed, hence accountability is preserved.

3.5 IPFS Hosting Electronic Copies of the Documents

In Hyperledger Fabric, a constraint of the blockchain is that only textual data may be recorded inside the blocks. Images and other large pieces of data cannot be directly uploaded to the ledger. For our system, we need to keep image files like each applicant's NID photo, a GD copy, etc. Therefore, we require a data storage system where the files' electronic copies will be kept. For this, we took advantage of the decentralized, peer-to-peer interplanetary file system (IPFS). However, before we can connect our system to the IPFS network, we must first install and initialize the IPFS client. The client is ready to upload an image file to IPFS when the daemon has been started. Following that, we must launch a new Ubuntu terminal from the picture file location. To upload the document's pdf, perform the bash command "ipfs add [nameoftheimagefile]". As a result, a hash is produced. Following that, this hash is added to the asset it belongs to using chaincode and transactions. Any changes made to the uploaded soft copy will significantly alter the hash..The soft copy of the document is downloaded and saved where the terminal is open when the command "ipfs get [hash]" is issued.

4 Result Analysis

We had to create a real implementation using the Ubuntu 20.04 LTS operating system and a Docker container because hyperledger fabric lacks a testing environment similar to composers (Hyperledger Composer Playground). For carrying out the necessary duties, such as producing endorsement responses, creating blocks, and broadcasting

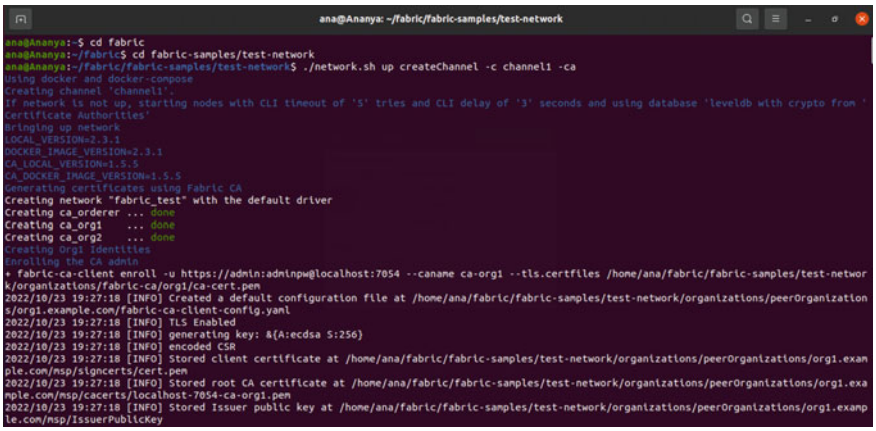


Fig. 4 Setting up hyperledger fabric

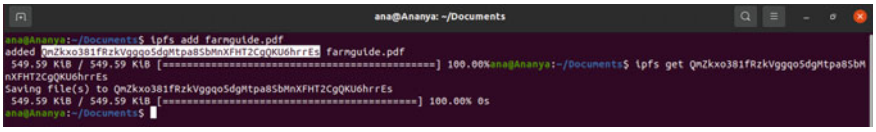


Fig. 5 Uploading and retrieving insurance information from IPFS

them throughout the network; the endorsers, peers, and orderers were created. The chaincodes are written in Javascript. In Fig. 4, the installation of prerequisites is shown, along with the fabric setup and activation.

After uploading the relevant electronic copy of the farm insurance, the IPFS generated a content identifier, which is what the hash in each specific asset for the farm insurance refers to. The IPFS daemon is started, followed by the initialization of IPFS, which is covered in the "methodology" section, in order to upload and receive the CID of farm insurance. The farm insurance's soft copy is uploaded using the command "ipfs add [fileName]," and it is retrieved using "ipfs get [hash]", as seen in Fig. 5.

Using the "CreateAsset" transaction listed in the chaincode, we generated numerous assets. The name of the specific transaction, together with any required parameters, must be supplied to the Fabric network in order for it to be executed. The process to initiate and execute the "CreateAsset" transaction has been shown in Fig. 6. We programmed this specific transaction, so that, after generating the asset, the user is given the success message through the asset details in the Ubuntu bash. Additionally, as shown in Fig. 6, the "AssetExists" transaction allows users to see if a certain asset is present in the Fabric network by providing an asset ID as an input. The output displays "true" if it exists and "false" otherwise.

Any existing asset may be updated using the "UpdateAsset" transaction, but in order to do so, the fabric network must receive the name of the specific transaction

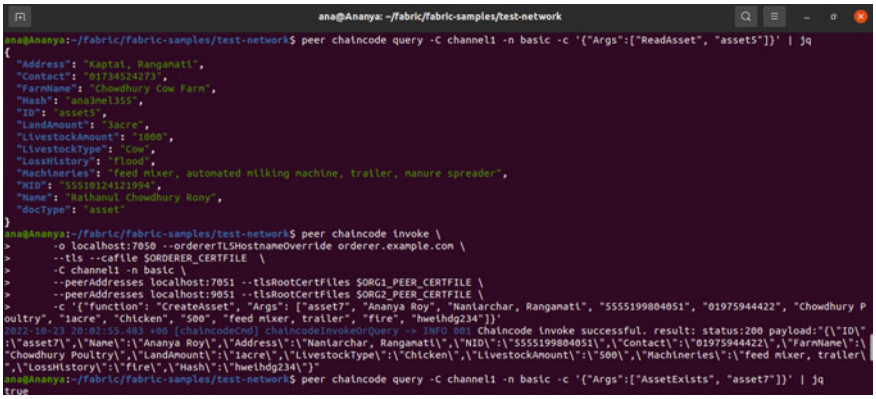


Fig. 6 Execution of "CreateAsset" and "AssetExists" transactions

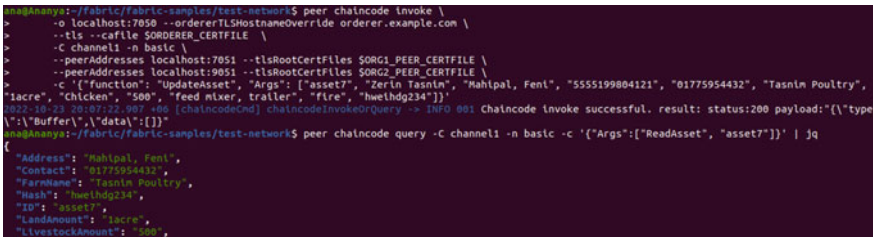
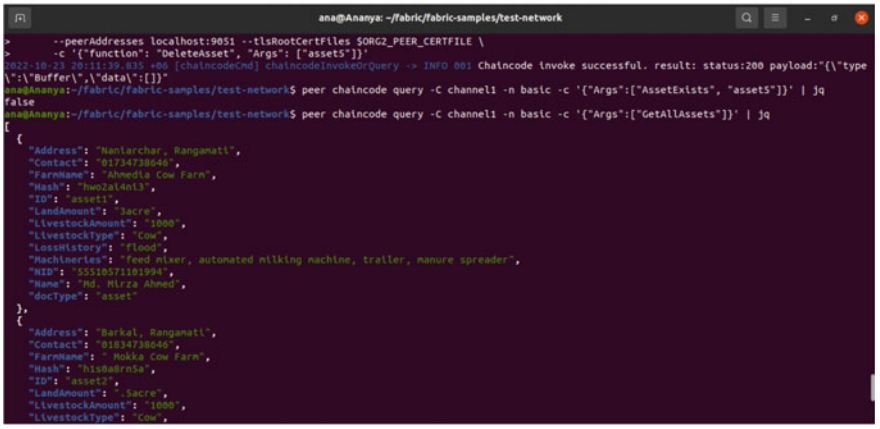


Fig. 7 Execution of "UpdateAsset" and "ReadAsset" transactions

as well as any necessary parameters as shown in Fig. 7. The "ReadAsset" transaction (Fig. 7), which accepts the asset ID as an input, and the "SearchAssets" transaction (Fig. 8a), allow users to get specific assets from our system. In "SearchAssets", any asset parameter from the assets can be used, instead of the NID. The asset ID must be entered when using the "DeleteAsset" operation to remove any asset, as shown using Fig. 8a and b. A demonstration of using the "GetAllAssets" transaction to get all the data associated with the farm insurance is displayed in Fig. 8b.

In addition, we developed our system using both Ethereum and Bitcoin. We used the Remix IDE and the solidity programming language to build our Ethereum-based solution. From GitHub, we downloaded an example Bitcoin project that uses the "Python-bitcoinlib" package to implement Bitcoin. All three solutions, including hyperledger fabric, had their throughput evaluated. For Ethereum, the "block.number" and "block.timestamp" functions from the solidity programming language were utilized; for Bitcoin implementation, the "time.time" and "time.now" functions from the "time" library of Python were used. Table 1 shows that fabric appears to be even better suited for getting farm insurance than the other two systems since it has the lowest throughput of the three blockchain-based systems.

(a) Initiating "SearchAssets" Transaction



(b) Initiating "DeleteAsset" and "GetAllAssets" Transactions

Fig. 8 Searching and deleting assets from asset registry

Table 1 Throughput comparison among the fabric-based system, Ethereum and Bitcoin

Amount of transactions (Txns)	Required time in hyperledger (s)	Required time in Ethereum (s)	Required time in Bitcoin (s)
620	18.6	40.4	89.3
550	16.7	39.7	86.4
470	13.4	37.6	84.2
365	12.5	35.3	83.5
240	10.8	33.8	81.1
170	8.9	32.5	79.6

Required time to execute per transaction on average

Hyperledger	Ethereum	Bitcoin
0.021	0.183	0.642

5 Conclusion and Future Recommendations

In our research, we developed a system that allows a person who owns a farm to secure farm insurance using the hyperledger fabric private blockchain, which is intended to provide protection to farm owners in the case of a catastrophe or loss that farm owners cannot afford. It also addresses all essential blockchain elements including tracking, verification, and validation. To verify that our chaincodes are accurate, a few contractual tests have been carried out. IPFS makes sure that all information saved on the blockchain network is encrypted, allowing only the insurance buyer to read and distribute it as they see fit.

We experienced a few complexities while developing the system. For example, each fabric version causes a change in the chaincode commands. As we used version 2.3.1, its chaincodes may not work in other versions. Also, there is no testing platform for fabric; the entire system can only be tested after implementation.

Moreover, we did not create any front end for our system which makes it hard to utilize the system. In future, we hope to develop a Web client for our system to make the system user-friendly. In addition, we hope to utilize the OCR functionality which recognizes texts of the digital image. By using such technology to scan images, users would be able to automatically add data or information from any image, eliminating the need for human entry.

References

1. Aleksieva V, Valchanov H, Huliyan A (2019) Application of smart contracts based on Ethereum blockchain for the purpose of insurance services. In: 2019 international conference on biomedical innovations and applications (BIA). IEEE, pp 1–4
2. Gatteschi V, Lamberti F, Demartini C, Pranteda C, Santamaría V (2018) Blockchain and smart contracts for insurance: is the technology mature enough? Future Internet 10(2):20
3. Hassan A, Ali M, Ahammed R, Khan MM, Alsufyani N, Alsufyani A et al (2021) Secured insurance framework using blockchain and smart contract. Sci Program 2021
4. Jha N, Prashar D, Khalaf OI, Alotaibi Y, Alsufyani A, Alghamdi S (2021) Blockchain based crop insurance: a decentralized insurance system for modernization of Indian farmers. Sustainability 13(16):8921
5. Lepoint T, Ciocarlie G, Eldefrawy K (2018) Blockcis—a blockchain-based cyber insurance system. In: 2018 IEEE international conference on cloud engineering (IC2E). IEEE, pp 378–384
6. Loukil F, Boukadi K, Hussain R, Abed M (2021) Ciosy: a collaborative blockchain-based insurance system. Electronics 10(11):1343
7. Nguyen T, Das A, Tran L (2019) Neo smart contract for drought-based insurance. In: 2019 IEEE Canadian conference of electrical and computer engineering (CCECE). IEEE, pp 1–4
8. Raikwar M, Mazumdar S, Ruj S, Gupta SS, Chattopadhyay A, Lam KY (2018) A blockchain framework for insurance processes. In: 2018 9th IFIP international conference on new technologies, mobility and security (NTMS). IEEE, pp 1–4
9. Reno S, Haque M (2023) Utilizing off-chain storage protocol for solving the trilemma issue of blockchain. In: Emerging technologies in data mining and information security. Springer, pp 169–179

10. Roriz R, Pereira JL (2019) Avoiding insurance fraud: a blockchain-based solution for the vehicle sector. Procedia Comput Sci 164:211–218
11. Vakilinia I, Badsha S, Sengupta S (2018) Crowdfunding the insurance of a cyber-product using blockchain. In: 2018 9th IEEE annual ubiquitous computing, electronics & mobile communication conference (UEMCON). IEEE, pp 964–970
12. Zhou L, Wang L, Sun Y (2018) Mistore: a blockchain-based medical insurance storage system. J Med Syst 42(8):1–17