Journal of Theoretical and Applied Information Technology
31st December 2024. Vol.102. No. 24
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INSURECHAIN: A BLOCKCHAIN-BASED SYSTEM FOR SECURE, EFFICIENT AND INTEROPERABLE INSURTECH

EMAN YASER DARAGHMI[1], HAZIM HARB[2], YOUSEF AWWAD DARAGHMI[3*]

[1]Computer Science Department, Palestine Technical University-Kadoorie, Tulkarm

[2]Ministry of Finance, Ramallah, Palestine

[3]Computer Systems Engineering Department, Palestine Technical University-Kadoorie, Tulkarm

E-mail: [1]e.daraghmi@ptuk.edu.ps , [2] hhazem@pmof.ps, [3]y.awwad@ptuk.edu.ps

ABSTRACT

Insurance Technology (InsurTech) solutions aim to transform traditional insurance models to more personalized services, streamlined claims processing and faster delivery. However, records in InsurTech are fragmented and isolated, rather than interoperable and cohesive. The need for multiple access to insurance records had raised the interoperability challenges between clients and providers, which pose additional barriers to effective data sharing. Additionally, insurance data face increasing threats since several advanced techniques have been developed to violate digital privacy and security. Therefore, Blockchain, which is a distributed trusted immutable database solution and ledger technology, can solve these problems. Although several studies were proposed to employ the Blockchain for managing insurance records, there is still a need for more research to better understand, characterize and evaluate its utility in InsurTech systems. This paper proposes a Blockchain-based InsureTec system called InsureChain that provides interoperable, secure, and efficient access to the records by providers, clients and third parties while maintaining privacy. The InsureChain employs smart contracts for effectively managing the claims of clients, governing transactions, and monitoring computations through the enforcement of acceptable usage policies. For better efficiency, the InsureChain uses Proof of Authority (PoA) and an incentive mechanism that leverages the degree of provider's nodes from the perspective of InsurTech systems by measuring their efforts regarding maintaining records and creating new blocks. The system was evaluated by reviewing privacy, integrity communication channels, and security and comparing consensus algorithms. The results show that the system achieves high privacy, integrity, and secure communication. The PoA outperformers the proof of work and proof of stake.

Keywords: *Insurance, InsurTech, Privacy, Confidentiality, Cybercrime, Blockchain, smart contracts*

1. INTRODUCTION

The development in the InsurTech promotes possibilities of new services and opportunities for data collection [1]. InsurTech includes different process started from buying insurance policy until settling claims [2]. An individual may have more than one insurance from the same or different providers for various needs, such as health, car, and house insurances. The client's records are stored in the provider's database who issued the insurance and will only be the eligible provider for managing and editing it. Clients with access rights could query their records from different providers, and providers with access rights could query records of common clients from other providers. This makes records fragmented and isolated, rather than cohesive and interoperable causing a lack of coordinated data management and exchange [3]. Additionally, several advanced techniques are developed to violate digital privacy and security. Unfortunately, personal records are considered as major targets for information theft as they contain sensitive information, such as names, identity numbers, contacts info and addresses.

The immutability and traceability of the Blockchain technology can solve these problems well. Essentially, a Blockchain is a distributed trusted immutable database solution and ledger, which stores a continually increasing set of data verified and confirmed by participants [4][5]. Blockchain technology provides a stable, secure, auditable, transparent, and effective way to data information and record transactions. Thus, preserving the information of insurance records on the Blockchain can ensure both the originality of the data and solve the security problem of the central authorization mechanism. In this way, a fair and just insurance business can be achieved between the insurance company and the client [6][7].

Although the Blockchain technology was first introduced through Bitcoin, the industry of InsurTech is one of the fields where the Blockchain technology is believed to have considerable impact. Researchers propose scalable permissioned Blockchain based frameworks for insurance records where records are stored in the existing databases of providers and the Blockchain can be integrated as an access control layer to allow the interaction between participants [6][7][8]. Researchers also propose varying techniques for establishing secure access controls for the Blockchain [9][10]. Although a number of studies is proposed to employ the Blockchain for managing insurance records, there is still a need for more research to better understand, characterize and evaluate its utility in InsurTech systems. More importantly, research is needed to make InsurTech based Blockchain systems applicable for real-life scenarios with full support of security, privacy and interoperability.

This paper proposes the InsureChain which is an InsurTech Blockchain based system that aims at providing interoperable, secure, and efficient access to the insurance claim records by providers, clients and third parties while maintaining privacy. Smart contracts whose design meet the demands of InsurTech are employed for governing the transactions, monitoring the computations and managing the use of data after transmission. Each task in the insurance process will be recorded and stored on Blockchain by sending an interactive transaction to the smart contract. The InsureChain adopts advanced cryptographic techniques for supporting security, and specific portions of the data, particularly sensitive business-related information, are encrypted to ensure privacy. In addition, since an insurance record is a client's asset and not a cryptocurrency or a digital currency to be exchanged, we propose a new incentive mechanism that leverages the degree of provider's nodes from the perspective of InsurTech systems by measuring their efforts regarding maintaining records and creating new blocks. Providers' nodes with less degree are more likely to be selected for creating the new block.

The InsureChain was evaluated by reviewing privacy, integrity, secure communication and security. The results show that the InsureChain achieved high privacy, security, integrity and secure communication. The InsureChain implement the RSA algorithms for privacy, the AES for confidentiality, SHA256 for integrity and HTTPS for secure web communication. The results also show that the system is efficient as the used PoA and incentive mechanism enables small number of nodes to participate in mining and creating new blocks. The primary contributions of this work are threefold:

- We provide a complete analysis regarding how the proposed system and the smart contracts can interact with the various demands of insurance providers, clients and third parties.

- We demonstrate how the proposed system would address the longstanding issues of privacy and security in the InsurTech industry.

- We propose an incentive mechanism that aims at evaluating the degree of providers regarding their work in maintaining insurance records, which in turn will enhance data quality.

2. RELATED WORK

Blockchain uses a distributed, peer-to-peer network to make a continuous, growing list of ordered records called blocks to form a digital ledger [11]. Each transaction, represented in a cryptographically signed block, is then automatically validated by the network itself. So, Blockchain is a decentralized ledger that records all transactions on the network in a safe, verifiable, and transparent manner. The ledger is shared across the network's dispersed computers, and data are encrypted with a cryptographic method before being checked by miners to determine if the transaction is genuine [12][13]. A new block is added to the chain if the majority of the miners agree to the transaction. The main advantage of blockchain over traditional technology is that it enables two parties to conduct encrypted transactions via the Internet without the need for a third-party intermediary [14]. These features support security, interoperability, efficiency and privacy making Blockchain to solve real-world problems [5][12][14][15][16].

Smart contracts have greatly accelerated the use of blockchain technology in various sectors [17]. A smart contract is a computer program that is established on the Ethereum blockchain and represents a digital agreement between two or more parties, typically in the context of a business process, similar to how a regular paper contract works. The blockchain network nodes execute the smart contract, and the execution results are recorded on the blockchain [5][18]. All blockchain network nodes must agree on the new state of the smart contract after execution using the adopted consensus method [19][20]. The use of blockchain in conjunction with smart contract technology eliminates the need for transaction parties to rely on a central system.

Blockchain and smart contracts have been introduced to the InsuTec domain [3][21]. In [10], researchers used fog nodes to encrypt insurance records for more security. The authors in [9] proposed a decentralized Blockchain-based framework for the auto insurance sector that regulates the activities in terms of insurance claims for automobiles and automates payments. This article also discussed how Blockchain technology can be useful for the decentralized autonomous vehicle's ecosystem. In [22], authors tried to change insurance process and make it easier and more efficient. The authors proposed a new Blockchain system called "LifeBlocks" in order to solve different issues related to insurance sector.

The authors in [23] proposed a system called CAIPY. This system contains different smart contracts. The main idea of this system is to reduce cost and simplified process of claim insurance mainly in car crashed. The authors highlighted the usefulness of Blockchain and smart contracts in financial sector to avoid fraud. In [24], the authors demonstrated that there are many problems in insurance policy and manual process in handling claims between company and clients. So they suggested a framework based on blockchain and Ethereum smart contract for providing a secure procedure to execute the whole process from claim registration to refund. In [25], researchers presented a new framework called CioSy to assist insurance company in competitive tracks, and to automate the process of insurance and claiming from client. In [26], the authors suggested a Blockchain based framework that allows customers to register, obtain a policy, make a claim, and receive a refund in cooperation with the corresponding agent.

3. THE PROPOSED SYSTEM: InsurChain

This chapter presents the proposed Blockchain-based system, namely InsureChain that is developed to support the insurance sector. The aim of InsureChain is preventing fraud and manipulation while providing privacy, interoperability, security, efficient access by all stakeholders, ensuring the integrity of the insurance process and speeding the claim process.

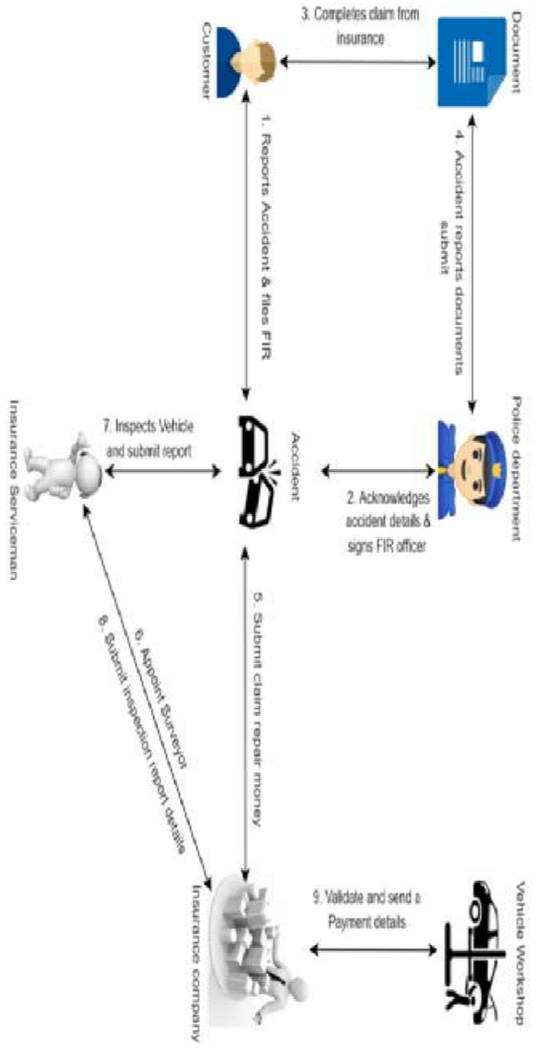

Customer
3. Completes claim from insurance
Document
1. Reports Accident & files FIR
4. Accident reports documents submit
Insurance Serviceman
7. Inspects Vehicle and submit report
Accident
2. Acknowledges accident details & signs FIR officer
Police department
6. Appoint Surveyor
8. Submit inspection report details
5. Submit claim repair money
Insurance company
9. Validate and send a Payment details
Vehicle Workshop

Figure 1: The InsurTech workflow showing the main components of the system and the communication among all components

3.1 InsurTech Workflow

The proposed system workflow is proposed based on the existing insurance claim workflow, and the InsurTech flow is shown in Figure 1.

As seen in Figure 1: the system has the following attributes:

- Insurance Company which serves as Header.

- Insurance Serviceman who serves as employee.
- Customer who is the Client.
- Police Station represented by Police Officer.
- Garage Man representing the Car Work Shop.

In the real-life system, the customer should have a valid insurance policy before submitting the claim. In the case of accident, the claim passes through specific procedures which are:

1. The customer reports an accident with all needed document, such as license, ID, car registration, policy number to prepare the First Information Report (FIR).
2. The FIR will be examined by a police officer who checks all documents and signed the FIR.
3. The customer completes the Claim Form and submits it to the insurance company and at the same time, the Police Officer sends the FIR to insurance company.
4. The customer submits claim repair money to the insurance company.
5. The insurance company sends the insurance serviceman to review the car and the accident.
6. The insurance serviceman prepares the inspection report and sends it back to the insurance company
7. The repair workshop sends payment request to the insurance company with full detail report about the accident.
8. The insurance company prepares payment based on the previous reports.

If any medical services is needed, the system can include a third party called Medical Services. Thus, medical reports and payment requests could be managed through the system too.

3.2 Privacy

One of the key issues for blockchain applications is privacy. Two forms of privacy must be examined in terms of blockchain to secure sensitive information and satisfy the standards outlined above [5], [12]. The first is the privacy of identification: the genuine identity of people interacting with the blockchain must be protected. The second is transaction privacy, which means that only authorized users have access to the contents of a transaction. When discussing blockchain technology, transaction privacy is the most challenging task to solve.

To improve and protect identity privacy, a mechanism based on the public and private keys were utilized. The RSA encryption algorithm is used to generate public and private keys for the public user at the registration phase, in addition to a secret phrase with 6 words, each word is combined from 4 numbers. The user used the private key or the secret phrase with a password to log in to the system. Also, if a user forgets his password, the private key with the secret phrase will be used in combination to reset the password. By this mechanism, there is no personal information for the user entered or stored in the system. By utilizing this mechanism, the customer identity privacy and anonymity will be achieved.

On the other hand, dealing with transaction privacy is more difficult and has to be properly handled. To address transaction privacy, the hash value of the claim or the feedback is stored on-chain, while the information itself is stored off-chain in a database in encrypted form. In addition, a smart contract can be written to allow authorized users only to access the transactions.

The proposed system deals with sensitive data, and the proposed system used several means to achieve confidentiality for protecting these data from revealing by unauthorized users. To improve the confidentiality of the claim data, no personal information is required during the registration or filing of a claim. On the other hand, the hash value only for the user registration in the information is stored. In addition, these data are allowed to be accessed by a smart contract for authorized users only. Moreover, to improve the confidentiality of the transaction data, Asymmetric Encryption Algorithm (AES) is used to encrypt the claim and the related transactions in the database, while the hash values only for these data are stored on-chain. In addition, to protect the data in transit, the secure socket layer protocol is utilized to protect data while it is moved between the clients and the nodes.

The transaction hash value that has been stored on the blockchain will be used to verify the integrity of the transaction data by recalculating the hash for the retrieved data and comparing it with the hash stored in the blockchain. On the other hand, the proposed system utilized the temporal table mechanism in the database engine, so in case of detecting any modification or deletion of data, the historical table will be used as a corrective phase to restore the original data.

3.3 Proof Of Authority (PoA)

To ensure security and efficiency, we use Proof of Authority which is a consensus mechanism that is either controlled or permissioned [5][27]. It is a family of Byzantine Fault Tolerance (BFT) algorithms to achieve a consortium by way of authorized nodes or validators. The validators are limited to a predetermined small number of authorized nodes to ensure efficiency. In PoA, the validating machines are responsible for generating new transaction blocks, which are then needed to be added to the network. The setup of any blockchain network determines the process through which it accepts new blocks and mines them. This mechanism includes the mining of new blocks. Because the reputation of the authorized node is at risk within the context of this protocol, malicious behavior might inflict significant harm on the entity. As a result, the network-validating nodes are responsible for maintaining network security.

3.4 Incentive Mechanism

As records are clients' assets and not a cryptocurrency or digital currency to be exchanged, this work proposes a new incentive mechanism integrated with the PoA for mining. It leverages the degree or significance of insurance company participating in the InsurChain regarding their efforts on maintaining records and creating new blocks. Our mechanism rewards the "block's creator" and incentive to be added to its degree and accordingly decreasing its probability of re-creating the next block. Thus, achieving the fairness and the equality among companies and ensuring the sustainability of the system.

3.5 Cryptography

Cryptography is utilized in the proposed system to preserve the privacy, confidentiality, and integrity of the data and users, and also to preserve the anonymity of the claim. To preserve the privacy and anonymity of the users, the asymmetric encryption algorithm RSA 2048 bit is utilized to generate public and private keys for the users. These keys are generated at the registration phase with a secret phrase. The user can use the private key or the secret phrase with a complex password to sign in to the system. When the user forgets his password, he can reset it using both the private key and secret phrase. On the other hand, the private key is used to digitally sign the transaction.

To ensure the integrity of the data, the hash algorithm SHA 256 is used in the proposed system to preserve the integrity of the user data and the transactions data. Confidentiality is ensured by utilizing a symmetric encryption algorithm AES to encrypt the data stored off-chain. Moreover, HTTPS is utilized to ensure the confidentiality of the data while it is transmitted between the client and the nodes.

3.6 InsureChain Architecture

The proposed InsureChain system focuses on anonymity, privacy, confidentiality, integrity, secure communication channel, and protection of insurance data from cybercrime. Based on these conditions and on the insurance workflow, the proposed system components shown in Figure 2 consists of user interface, database manager, Encryption/ Decryption component, and Ethereum Client.

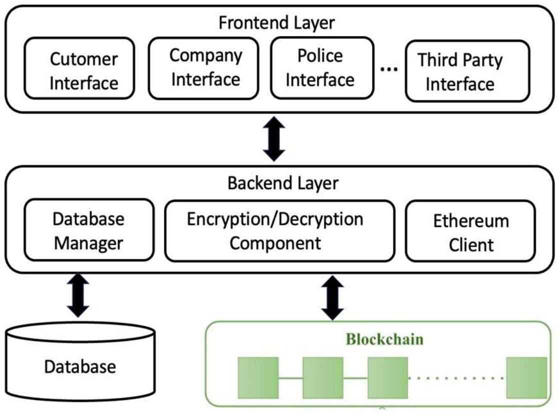


Figure 2: The proposed InsureChain architecture where the backend layer is responsible for communicating data with the interface, database via DB manager or the Blockchain via Ethereum client

The architecture is distinguished by other insurance digital system by employing blockchain for storing on-chain all relevant information associated with a claim. Also, the system tracks all transactions by employing a smart contract, which is software that executes contracts and cannot be altered after it has been deployed. Therefore, all procedures are recorded and documentation of every step in the process is always accessible. All parties will be able to keep track of the proof at any time, and they will be secure from manipulation since the information is recorded in the blockchain.

3.6.1 InsureChain Interface

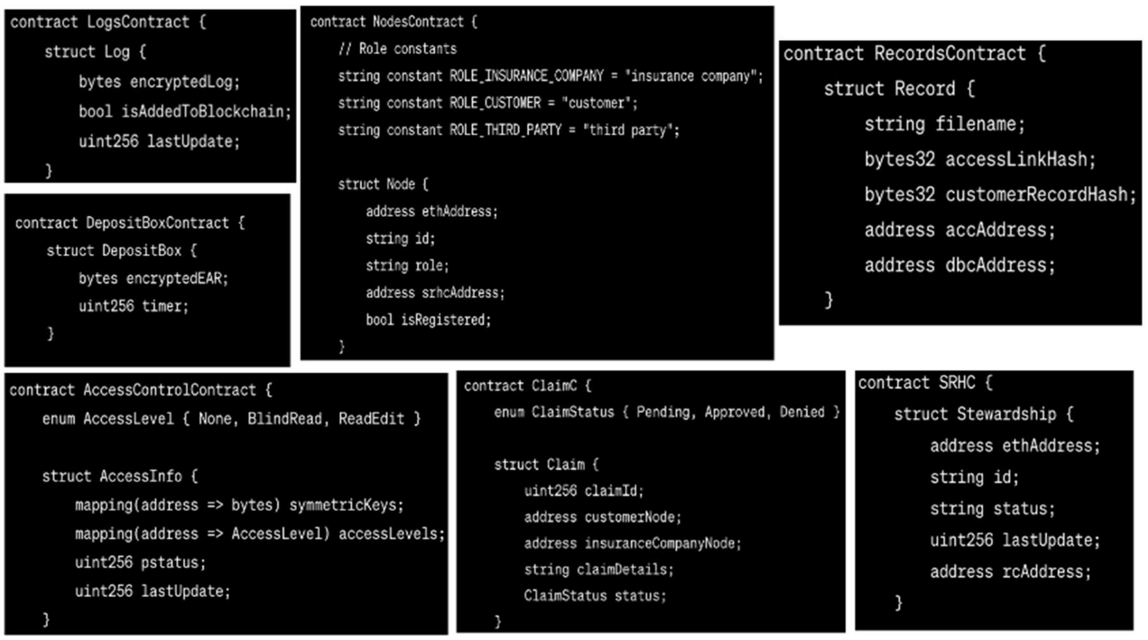
The interface is web-based that is users including customers, insurance company, insurance agent, police, mechanics and other third party such as healthcare providers. The system employs an admin who can create the users with their access level according to the workflow contract. The users can retrieve and create transactions on the database through the claim transaction contract and according to the access control contract, where all of the user's transaction logs are recorded using the logs contract. The interface is built using python with a flask framework.

3.6.2 Database Manager

The proposed system stores the claim data off-chain in a centralized database. The claim data is stored in the database in an encrypted form. The data integrity is verified by storing the hash value of the claim data in the claim transaction contract in the blockchain. The database functions and queries are written in the proposed system using the python programming language

3.6.3 Encryption/Decryption Component

This component is utilized in the proposed system to preserve integrity, confidentiality, and user anonymity. The asymmetric key encryption algorithm RSA 2048 is utilized in the system to generate the public and private keys for the system users. While symmetric key encryption AES is used to encrypt/decrypt the data stored in the database. In addition to ensuring the integrity of the data, SHA 256 hash function is utilized in the system. Moreover, to ensure the confidentiality of the data in transmission the HTTPS protocol is utilized.



```
contract LogsContract {
    struct Log {
        bytes encryptedLog;
        bool isAddedToBlockchain;
        uint256 lastUpdate;
    }
```

```
contract DepositBoxContract {
    struct DepositBox {
        bytes encryptedEAR;
        uint256 timer;
    }
```

```
contract NodesContract {
    // Role constants
    string constant ROLE_INSURANCE_COMPANY = "insurance company";
    string constant ROLE_CUSTOMER = "customer";
    string constant ROLE_THIRD_PARTY = "third party";

    struct Node {
        address ethAddress;
        string id;
        string role;
        address srhcAddress;
        bool isRegistered;
    }
```

```
contract RecordsContract {
    struct Record {
        string filename;
        bytes32 accessLinkHash;
        bytes32 customerRecordHash;
        address accAddress;
        address dbcAddress;
    }
```

```
contract AccessControlContract {
    enum AccessLevel { None, BlindRead, ReadEdit }

    struct AccessInfo {
        mapping(address => bytes) symmetricKeys;
        mapping(address => AccessLevel) accessLevels;
        uint256 pstatus;
        uint256 lastUpdate;
    }
```

```
contract ClaimC {
    enum ClaimStatus { Pending, Approved, Denied }

    struct Claim {
        uint256 claimId;
        address customerNode;
        address insuranceCompanyNode;
        string claimDetails;
        ClaimStatus status;
    }
```

```
contract SRHC {
    struct Stewardship {
        address ethAddress;
        string id;
        string status;
        uint256 lastUpdate;
        address rcAddress;
    }
```

Figure 3: Smart Contracts of the InsureChain System

3.6.4 Ethereum Client

While the proposed system work on a permissioned blockchain network, the Ethereum Blockchain network is used. Therefore, to join the Ethereum network, we use the Ethereum client which has all the required features necessary to access the functionalities of Ethereum.

3.7 InsurChain: Smart Contracts:

The InsurChain smart contract consists of several contracts that are shown in Figure 3.

3.7.1 Nodes Contract (NC)

This contract maps a node identification string to its associated Ethereum address identity, i.e. public key. The adoption of the identification strings rather than the public key directly is to allow already existing IDs to be used. Procedures and Policies coded into the NC regulate adding and registering new IDs. This contract additionally determines the role of each participant node within the system i.e., insurance company node, customer node, Police Officers, Mechanics and third party. This contract recognizes nodes that have already registered and thus avoiding the case of double registration. Additionally, the NC maps the

Ethereum address of the node with its associated Steward-Relation History Contract (SRHC) address.

3.7.2 Steward-Relation History Contract (SRHC)

This contract maintains the steward relationship history of each participants' nodes in the system where the customer's record is stored and managed by the insurance company node. The SRHC locates the history of records by holding a summary list of the steward relationship. For example, if a node role is customer, its SRHC will have references to all insurance companies that it has been engaged with. On the other hand, the SRHC of an insurance company has references to all customers' nodes in which that company serves. Every node within the Blockchain system will have a SRHC that will be created during the registration process.

Generally, the SRHC is identified by the Ethereum address of the SRHC owner node and stores the Ethereum addresses of all associated nodes, their related IDs, a stewardship statue, a last-update date field that indicates the last update on the status field, and an address to the applicable RC. Users notifications can be enabled via the use of the stewardship status field, such as the stewardship is "newly" established, "awaiting pending updates", and "acknowledged customer approval" or "acknowledged customer denial". The stewardship status in the customer SRHC is set by insurance company node in the system every time they update the customer record or as a part of establishing a new stewardship. Thus, customers can be notified upon modifying the stewardship status field as a new stewardship is recommended or an update is available.

3.7.3 Records Contract (RC)

This contract tracks all records in which insurance companies store for customers and is generated when a new steward relation is established between two nodes. The RC includes several data fields with different purposes, and is identified by the Ethereum address of the owner that signifies the customer who owns the record(s). Each record has a filename f, conditions, and AccessInfo. The filename indicates the identity string for the record. The AccessInfo data field of the record specifies the needed information to find the ER Database of a company , i.e. the company's host name and the information for the port in a standard network topology. Moreover, to maintain data integrity, each record has a hash value h (AL) for the access link of the file, and a hash value h(ER) of the stored record. Moreover, references to the AAC address and the DBC address are listed in the RC.

3.7.4 Logs Contract (LC)

This contract tracks all transactions performed on the records to facilitate adding/validating/appending blocks in the Blockchain network. This contract is identified by the Ethereum address of the source of the transaction. It lists the transaction details in an encrypted log data field with a status field that indicates whether the new log has been added to the Blockchain. It also stores the last update of the status field.

3.7.5 Access Control Contract (ACC)

The ACC includes all permissions related information which is specific to every record. It lists the Ethereum addresses for all nodes who have access permissions on the record. This contract specifies the level of that access (i.e. owner, read/edit, and blind-read), and a symmetric key encrypted with the public key of each node. A "read/edit" access level indicates that a node can read and partially edit the ER as it has the symmetric key that is generated to encrypt the record when it is first added (i.e. edit access level means a customer for example can update his address or change his profile picture). The temp-read level indicates that the node can temporary read a record as it has the DBC address that will keep an encrypted copy of the record for a certain time to function the temporary access by a trusted third party. The owner level is assigned to the insurance company node who adds the record. It indicates that a node has full access and control of the ACC as it can add other nodes with the "read/edit" level, remove nodes from the AAC, and also alter the level for any existing nodes. The AAC also contains the "pstatus" field along with the lastupdate in order to notify participants when there is a change in their access level.

3.7.6 Deposit-Box Contract (DBC)

This contract is employed when there is a verified request from a third party to access an ER for a customer. This contract stores an encrypted ER for a certain time to facilitate the record's access by a third party. The stored record will be encrypted via the public key of the third part before storing and will be kept only for a certain time specified by the Timer field.

3.7.7 Claim Contract (ClaimC)

This contract is responsible for managing the claim process and handling the interactions between the involved parties (customer nodes, insurance company nodes, police officer, mechanics and third parties). The Claim Contract includes the policeAccidentReportID field to store the police accident report ID associated with each claim. When a customer node submits a claim, they can provide the police accident report ID along with other claim details. The validateClaim function allows insurance company nodes to validate a claim, and additional validation logic can be added as required (e.g., cross-referencing with PoliceContract). The getClaimDetails function allows anyone to retrieve the details of a claim by its claim ID, and the isClaimValidated function allows anyone to check if a claim has been validated by the insurance company.

3.7.8 PoliceContract

This contract tracks all AccidentReports in which a police officer is issued for customers in an insurance company. This contract includes several data fields including report ID, police Officer, access Info, accessLinkHash, and reportHash.

The report ID represents the identity string for the accident report, while police Officer is the Ethereum address of the police officer who issued the report. The access Info field contains the information needed to access the actual police report off-chain, such as the company's host name and port information. Instead of storing the full access link and the actual report on-chain, the contract stores their respective hash values (accessLinkHash and reportHash) for data integrity. The addAccidentReport function allows police officers to add new accident reports by providing the necessary details, and the AccidentReportAdded event is emitted to inform relevant parties of the new report. The contract also includes functions to retrieve accident report details, access link hash, and report hash for a specific police officer. By storing the report off-chain and using hashes on-chain, the PoliceContract maintains data integrity while providing necessary access for the insurance company to validate and verify accident claims based on the police accident reports.

4. IMPLEMENTATION OF THE INSURCHAIN SYSTEM

4.1 Adding a new customer (Applying for car insurance):

In real-life car insurance scenario, a person (potential customer) visits an insurance company's office or website and expresses their interest in obtaining car insurance. They submit an application to the insurance company. The insurance company's staff (providers) validate and authenticate the received application. They verify the customer's identity, check if the customer is eligible for insurance, and ensure that there is no existing policy for the same customer ID. If the application is accepted and verified, the insurance company proceeds with the next steps. They update their records. After successfully adding the customer to their system, the insurance company creates a new contract known as "Steward-Relation History Contract" (SRHC) for the new customer. This contract will store relevant information and history related to the customer's interactions with the company. The address of the newly created SRHC is then forwarded to the insurance company for reference and future interactions. The insurance company shares the necessary account information with the new customer. This could include policy details, coverage information, terms and conditions, etc.

In blockchain-based scenario, the process of adding a new customer node in the InsureChain system involves validation, data storage in contracts, and information sharing, much like the real-life process of applying for insurance coverage. The InsureChain system provides a decentralized and secure way to manage customer data and interactions between different nodes in the network. The process begins when an insurance company node sends a request. The insurance company node sends the Ethereum address of the new customer node, its ID and the requested role to the NC for validation. Providers' nodes validate and authenticate that received request by ensuring and confirming that the received request is related to a legitimate customer and the non-existence of a registered customer matching that received ID. If the request is accepted and validated, the NC updates its local memory with the customer's ID, its Ethereum address and a "customer" role. The NC creates a new SRHC for the new customer node whose address will be forwarded to the company. The new customer's account information will be sent to the customer node from the insurance company node who forms the request.

4.2 Adding a new Record Via An Insurance company

The procedures of adding a new record starts after establishing a stewardship between the company and the customer nodes and thus having a shared RC. First, internal encryption in the insurance company node begins the process of adding a new record. When a new record is created by an insurance company node, that record will be transferred to the DB Manager. It creates an access link (AL) of a free location in the company existing Database and hashes both the generated access link h (AL) and the record h(ER).

The DB manager forwards the created access link and the record to the Cipher/Decipher Manager for encryption. The Cipher/Decipher Manager generates a symmetric key (SMK), encrypts the new record and link with that key and then encrypts that generated symmetric key with the public keys of the company, customer and set of proxies. The Cipher/Decipher Manager sends the encrypted record to the DB manager to store. In addition, all other encrypted data will be sent to the DB manager to create a log indicating the creation of the new record since the history of all access will be stored in the Blockchain to provide a full view of all events that happened to each record. The hash of the created log will be calculated and stored in the DB manager for block verification later. Thus, ensuring the integrity of data since if any part of the data is changed, all involved nodes will notice the alteration. Then, the log will be sent to the Cipher/Decipher manager for encryption with the public key of the insurance company node.

The insurance company node sends the customer's ID to its SRHC that will return the associated RC address. The insurance company node then sends the record information (filename of the record, hash value of the access, and hash value of the customer's record, the encrypted symmetric keys, and the log) to the RC. The RC stores the filename of the record, the hash value of the access link, and the hash value of the customer's record. The RC then creates a new ACC for the record and forwards the encrypted symmetric keys $E_{pk}(SMK)$. The ACC auto-creates the access and permissions information for the record, i.e. customer and company permissions, and then sends its address to the RC for its reference. On the other hand, the LC updates its entries with the received encrypted log, the associated Ethereum address of the insurance company node, the "new log" status to indicate that the new log has not been added to the Blockchain yet, as well as the timestamp of the last status update. At the end, the encrypted access link is sent to the customer over HTTPS who will store that link in it's cipher/decipher manager and will be used when the customer would like to read his/her record. Additionally, when the new record is created, the insurance company node notifies the NC to updates the associated degree of the insurance company node. The NC informs the REM to add the value of the added record to the node's degree and to return it to perform the update.

In real life, when an insurance company receives a new record (e.g., a new insurance policy, a claim, or customer information), it goes through a series of steps to ensure data integrity and proper storage, and these steps are:

1. Record Creation: The process starts when an insurance company creates a new record, such as a new insurance policy for a customer.

2. Access Link and Record Hashing: The insurance company generates an access link (AL) that points to a free location in its existing database. The AL, along with the record (e.g., insurance policy details), is hashed to ensure data integrity.

3. Encryption: The insurance company sends the AL and the record to a Cipher/Decipher Manager for encryption. The Cipher/Decipher Manager generates a symmetric key (SMK) and uses it to encrypt both the AL and the record. Additionally, the SMK is encrypted using the public keys of the company, customer, and a set of proxies for secure sharing.

4. Data Storage: The encrypted record is sent back to the insurance company's Database Manager for storage in the company's existing database. The Database Manager also receives other encrypted data to create a log that records the creation of the new record. This log is essential for maintaining a full view of all events that happened to each record.

5. Log Verification: The hash of the created log is calculated and stored in the Database Manager. This step ensures all involved nodes will detect the integrity of the data since any change to the data in the Blockchain-based system.

6. Record Contract (RC) Interaction: The insurance company node communicates with its Steward-Relation History Contract (SRHC) to retrieve the associated RC address for the customer's record.

7. Information Forwarding: The insurance company node sends the necessary information, such as the filename of the record, hash value of the access link and customer's record, encrypted symmetric keys, and the log, to the RC.

8. Record and Access Control Creation: The RC stores relevant information about the new record, including the filename, access link hash, and customer's record hash. It creates a new Access Control Contract (ACC) for the record, which automatically sets access and permissions information for the record, such as customer and company permissions.

9. Log and Status Update: The Log Contract (LC) is updated with the received encrypted log, associated Ethereum address of the insurance company node, "new log" status, and timestamp of the last status update.

10. Secure Access Link Sharing: The encrypted access link is sent to the customer over HTTPS, allowing the customer to store the link in their cipher/decipher manager. This link will be used when the customer wants to read their record securely.

11. Degree Update: The insurance company node notifies the Node Controller (NC) about the new record creation, which leads to the update of the associated degree of the insurance company node. The NC informs the Record Evaluation Manager (REM) to add the value of the added record to the node's degree and to return it to perform the update.

4.3 Editing A Record by An Insurance Company

The insurance company node sends the customer's ID to its SRHC to retrieve the associated RC address. Upon receiving the RC address, the insurance company node then sends the filename of the requested record and its Ethereum address to the RC. The RC forwards the request to the ACC to check whether that received Ethereum, address has a permission (i.e. “owner” access level) on the requested record or not. If the insurance company node has a permission, the AAC forwards the company's encrypted symmetric key $E_{pk_{student}}(SMK)$ to the RC. The RC in turns forwards the received key to the insurance company node.

The cipher/decipher manager in the company's node first decrypts the received symmetric key using its private key and then decrypts the access link with that symmetric key. The DB manager of the company's node follows the related access link and then retrieves the encrypted ER from the database for editing. Note, when a record is modified, its hash value will also be changed [28]. Thus, the DB manager, after modifying the record, calculates the new hash of the modified record *h(EAR*). The DB manager sends the customer's ID to its SRHC to retrieve the associated RC address. The new hash value will be sent to the RC for updating. Moreover, the DB manager creates a log indicating the process of record editing, hash the log and then forwards the log to the cipher/decipher manager for encryption. The encrypted log then will be forwarded to the LC. The LC adds a new entry with the received encrypted log, a “new log” status to indicate that the new log has not been added to the Blockchain yet, and a timestamp indicating the last status update. Additionally, when the editing the customer's record, the insurance company node notifies the NC to updates the associated degree of the insurance company node. The NC informs the REM to re-evaluate the value of the record and thus updating the node's degree. The REM performs the calculations and returns the new degree to the NC for updating.

4.4 Access a record from the customer

A customer's node sends the company's ID to its SRHC to retrieve the associated RC address. Upon receiving the RC address, the customer node then sends the filename of the requested record and Ethereum address of the customer to the RC. The RC forwards the request to the ACC to check whether that received Ethereum address has a permission (i.e. “read/edit” access level) on the requested record or not. If the customer node has a permission, the AAC forwards the customer's encrypted symmetric key $E_{pk_{student}}(SMK)$ to the RC. The RC in turns forwards the received key with the database access information to the customer node.

The cipher/decipher manager in the customer 's node first decrypts the received symmetric key using its private key and then decrypts the access link with that symmetric key. The DB manager of the customer's node follows the related access link and retrieves the encrypted ER from the company's database. Since customers can access their nodes via online wallets, any device with Internet connection can perform records access. Thus, improving the interoperability of ER s. Moreover, the DB manager creates a log indicating the process of reading the record, hashes the log and then forwards the log to the cipher/decipher manager for encryption. The encrypted log then will be forwarded to the LC. The LC adds a new entry with the encrypted received log, a “new log” status to indicate that the new log has not been added to the Blockchain yet, and a timestamp indicating the last status update.

4.5 An Insurance Company Reads a Record From Another Insurance Company

The process of reading a record that is stored in an insurance company node from another insurance company node utilizes the timed-based deposit-box mechanism to increase both the accessibility and the security of ER s systems.

Suppose that there are two insurance companies nodes A and B, where company B would like to read a specific customer's record from company A. Company B generates a request to read the record first, signs that generated request by its private key for authorization, and then encrypts the signed request with the public key of company A. Over HTPPS, the encrypted signed request will be sent to company A. Upon receiving the request, company A decrypts the request with its private key and then decrypts it with the public key of company B to ensure that the company is the one that it claims to be.

First, node A will send the ID of the customer to its SRHC to return the associated RC address. Insurance company node A then sends the filename of the record to the RC to retrieve the associated ACC address. The insurance company node a then sends the Ethereum address and the access level request to the ACC. The ACC then forwards the Ethereum address of company B and its request for the NC to verify whether company B is an authorized company registered on the system. Upon receiving the verification from the NC, the ACC generates a new entry with the Ethereum address of node B, the Access Level, and the "request new level" status, and the timestamp of the last status update. The ACC requests a change in the access level from the file owner (i.e. customer), and updates both its status field to "waiting approval" and last-update. If the customer accepts the request, the ACC updates the access level with "temp-read" for the applicable file with the "approved" status and the last-update. Once the request has been approved, the ACC sends a notification to the insurance company node B indicating that a new access level is assigned to it. The ACC also notifies the RC to create a new entry in the DPC with the Ethereum address of company B.

The RC then sends the Ethereum address of company B, and the file name to the insurance company node A. The DB manager retrieves the record and forwards the record with the received public key to the cipher/decipher manager for encryption. The DBC will be updated by storing the encrypted file for a certain time specified by the Time field. Over HTTPS, company A sends the DPC address to company B to access the requested record by decrypting it using its private key. The DB manager of node B will update the LC entry with an encrypted log indicating the process of reading the record. The LC updates it status with "new log" to indicate that the new log has not been added to the Blockchain yet, and the timestamp of the last status update.

4.6 Claim Submission by a customer

The Claim Contract (ClaimC) handles the claim submission process. The Claim struct is defined to store the details of a claim, including the claimant's address, claim details, claim amount, and claim status (pending, approved, or rejected). The claims mapping is used to store all the submitted claims, with each claim having a unique claim ID. When a customer node wants to submit a claim, they call the submit Claim function, passing the claim details and the claim amount as arguments. The function first checks if the sender is a registered customer node by calling the isCustomerNode function from the Nodes Contract (NC). If the sender is a valid customer node, a new claim is created and added to the claims mapping with a pending status. The claim counter is then incremented for the next claim submission. After the claim is added, the function notifies the insurance company nodes about the new claim by calling the notifyInsuranceCompanyNodes function.

4.7 Claim Approval or Rejection:

The insurance company nodes can approve or reject the pending claims using the approveClaim and rejectClaim functions, respectively. To approve or reject a claim, the sender must be a registered insurance company node, which is verified by calling the isInsuranceCompanyNode function from the Nodes Contract (NC). If the claim is in a pending status, the function updates the claim status to either approved or rejected, as per the sender's action. Additionally, the function notifies the claimant about the approval or rejection using the notifyClaimant function.

4.8 Claim Status Retrieval

The getClaim function allows anyone to retrieve the details of a specific claim using its unique claimId. The function returns the claimant's address, claim details, claim amount, and claim status.

Please note that this is a simplified version of the claim submission process and may require further refinement and additional functionalities based on the specific requirements of the insurance system. Also, the smart contract assumes the existence of other contracts such as the Nodes Contract (NC) to handle node registration and authentication, which are referenced in the code using function calls. The actual implementation of those contracts is beyond the scope of this explanation, as it would involve more detailed contract code and interactions.

5. EVALUATION AND DISCUSSION

5.1 Experiment Setup

The software that are necessary to be installed on the machine to run the InsureChain system include Ubuntu operating system, NodeJs, Truffle, and Metamask. A number of steps are performed to run the system including firstly installing the dependencies of the project. Secondly, Truffle compile for generating a.json file in the build directory of the project that corresponds to the smart contract and describes its specific details. Thirdly, Truffle develop for contract deployment. Fourthly, installing and using Ethereum Metamask on Google Chrome:

The proposed system backend is developed using python 3.10. flask 2.03 with HTML, bootstrap 4, CSS, JS, jQuery, and AJAX are used to develop the front end of the system. Bootstrap and CSS are used in front-end development to provide the auto-responsive interface with multiple devices and to improve the accessibility of the system. The PyCharm 2021.3 educational edition is used because it is a python integrated development environment and provides a set of python development tools. In addition, PyCharm is used as a web-based applications development environment.

The ganache tool is used as a private and personal blockchain for development and testing. It is used in the implementation for deploying and testing the smart contract. Also, the solidity programming language is used to develop smart contracts.

5.2 Privacy and Confidentiality evaluation

The privacy required in the proposed system is for both the identity of people interacting with the system and the data in the system. By utilizing the RSA encryption algorithm to generate the user's public and private keys to interact with the system without the need for any personal information, the user's privacy was preserved. The transaction data privacy was preserved in the proposed system by utilizing the authorization method and access levels to access the claim data. In the proposed system the customer cannot access any claim other than his/her claim. On the other hand, only authorized company users can access the claim data.

The confidentiality of the data at rest and in transit was achieved in the proposed system where the AES encryption algorithm was utilized to store the claim data in an encrypted form in the database, in addition to storing only the hash values for the user's private keys, secret phrase, and passwords in the database. On the other hand, the claim hash value is only stored in the blockchain. While the HTTPS protocol is utilized to encrypt and protect the data in transit.

5.3 Integrity Evaluation

While the claim data hash value is calculated using SHA 256 algorithm and stored in the blockchain, when the customer or the company retrieve the claim data, the hash value for the retrieved data is recalculated and compared with the hash value retrieved from the blockchain. By this mechanism the integrity of the claim data can be verified, also any manipulation of the data will be detected.

5.4 Secure Communication Channel evaluation

By utilizing the cryptography in the proposed system, a secure communication channel was achieved. With the utilization of the HTTPS protocol, the data in transit will be encrypted and protected and will not be revealed. In addition to using the RSA algorithm to generate the user's public and private keys to be used as identification for the user instead of any personal data, the customer identity will stay anonymous. In addition, the use of the AES encryption algorithm to encrypt/decrypt the claim data will protect data at rest. All of these cryptography mechanisms achieved the objective of a secure communication channel between the customer and the insurance company.

5.5 Security Evaluation

Security testing for the proposed system web application was conducted using the the HCL AppScan standard tool. Security testing was conducted to detect any vulnerabilities in the proposed system and ensure that the proposed system and the internal users and customers are not vulnerable to attacks. The HCL AppScan is a dynamic application security testing (DAST) tool that analyzes web applications and APIs by simulating real-world attacks. It is capable of detecting a wide range of security issues, including:

- Injection attacks: AppScan can detect various types of injection vulnerabilities such as SQL injection, command injection, and LDAP injection.
- Cross-Site Scripting (XSS): AppScan can identify both stored and reflected XSS vulnerabilities.

- Cross-Site Request Forgery (CSRF): AppScan can detect CSRF vulnerabilities that allow an attacker to perform unauthorized actions on behalf of the user.
-

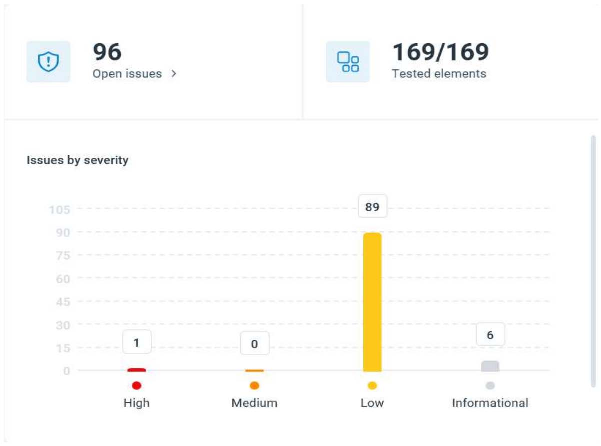


Figure 4: The HCL AppScan testing results on HTTP protocol

- Authentication and session management issues: AppScan can detect weak or broken authentication mechanisms, session fixation attacks, and session hijacking vulnerabilities.
- Information leakage: AppScan can detect sensitive data exposure, such as when an application leaks sensitive information like passwords, credit card details, or personal data.
- Broken access controls: AppScan can detect access control vulnerabilities that allow an attacker to bypass authorization checks and gain access to unauthorized resources.
- Security misconfigurations: AppScan can identify misconfigurations such as open ports, insecure protocols, and weak encryption.
- Other vulnerabilities: AppScan can also detect other types of vulnerabilities like buffer overflows, file inclusion vulnerabilities, and insecure communication protocols.

The security testing was conducted on the web application in two cases, on HTTP protocol and HTTPS protocol.

5.5.1 Security testing On HTTP protocol

On the first test, the web application was running on HTTP protocol, which means without using secure socket layer protocol (SSL). The result of the testing was a high-severity of vulnerability, as shown in Figure 4. The high severity was on the login page where the password was sent on a plain text form.

5.5.2 Security testing On HTTPS protocol

On the second test, the web application was running on HTTPS protocol using an SSL certificate from the OpenSSL library that has been added to the project on python. The result of the testing was with low severity of vulnerability, as shown in Figure 5. The result improved when the HTTS is used, and the high-severity vulnerability was removed because of using the SSL certificate, which encrypts the traffic between the client and the server.

Table 1: Consensus algorithms comparison

Consensus Algorithm	Computational Requirements	Energy Efficiency	Decentralization	Security
Proof of Authority	Low	High	Potentially low	Moderate
Proof of Stake	Moderate	Moderate	Moderate	Moderate
Proof of Work	High	Low	High	High

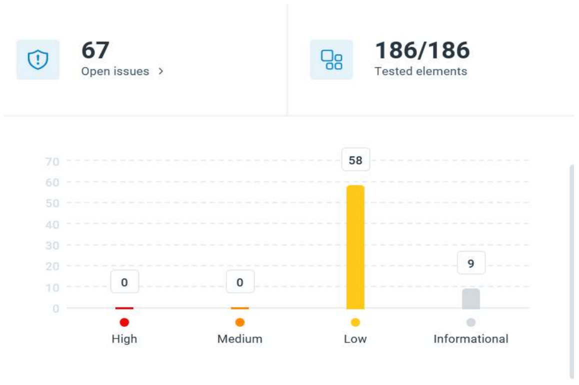


Figure 5: HCL AppScan testing results on HTTPS protocol

5.6 Consensus Algorithms Comparison

The Proof of Authority (PoA) consensus algorithm is used in the proposed system, which is a consensus algorithm that relies on a fixed set of validators who are pre-approved by the network's governing body. Table 1 shows the comparison between Proof of authority (PoA), Proof of Stack (PoS), Proof of Work (PoW) consensus algorithms. PoA improves the scalability of the system since it uses fewer resources, has a far higher throughput than PoW because there are fewer validator nodes in the network, and needs less computing power to operate. PoA also makes it simple to design and maintain decentralized apps, and since it operates in a centralized form, it reduces the number of forks, which in turn reduces the number of potential attacks on the network.

6. CONCLUSION

In this paper, we have proposed InsureChain system which is a Blockchain based InsurTech system that makes use of smart contracts. We have demonstrated that Blockchain with Ethereum smart contracts presents a novel opportunity to circumvent the restrictions and hurdles posed by InsurTech systems. This system enables claiming process from client to be held in a secure and cost effective manner while also protecting the privacy of clients and integrity of data. Because of its high level of transparency, Blockchain technology makes claim process and refunding much easier. Moreover, the proposed system enables the customer to follow up his/her claims easily and save time. Also, the system enables different insurance companies to exchange data between. The systems employs PoA which achieved higher efficiency because small number of nodes participate in the mining process. An incentive mechanism that leverage the degree of block creator was used, thus, achieving fairness among nodes because block creator will have lower chance to create a new block.

However, we have not yet had a complete understanding of all of the potential threats that are associated with the security and scalability of Blockchain based InsurTech systems, so additional research is necessary. In addition, there are many other parts in InsurTech such as medical that in some case need a real-time response to proceed. In previous sections, we just cover claim process in car accidents and we are aware that there are many other parts of insurance should be covered. Future work can also include the design of the system using Rapid and user-centered methods to increase the usability [29][30], and the integration of vehicle technologies, e.g., speed monitoring [31][32].

REFERENCES

[1] I. S. Gómez, “Insurtech: Disrupting the Insurance Industry,” in *The Emerald Handbook of Fintech*, H. K. Baker, G. Filbeck, and K. Black, Eds., Emerald Publishing Limited, 2024, pp. 343–362. doi: 10.1108/978-1-83753-608-520241043.

[2] J. John, M. Joseph, S. Joseph, G. Jacob, N. Rose, and S. Thomas, “Insurtech research dynamics: A bibliometric review of technological innovations in insurance,” *Multidiscip. Rev.*, vol. 7, no. 12, p. 2024288, Aug. 2024, doi: 10.31893/multirev.2024288.

[3] D. Yadav and Ram Milan, “IMPACT OF FUNDING IN INSURTECH ON PREMIUM PERFORMANCE OF INSURANCE BUSINESS,” *ShodhKosh J. Vis. Perform. Arts*, vol. 5, no. 6, Jun. 2024, doi: 10.29121/shodhkosh.v5.i6.2024.1866.

[4] Electronic and Computer Science Dept., University of Southampton, Southampton, UK, H. F. Atlam, A. Alenezi, M. O. Alassafi, and G. B. Wills, “Blockchain with Internet of Things: Benefits, Challenges, and Future Directions,” *Int. J. Intell. Syst. Appl.*, vol. 10, no. 6, pp. 40–48, Jun. 2018, doi: 10.5815/ijisa.2018.06.05.

[5] E.-Y. Daraghmi, Y.-A. Daraghmi, and S.-M. Yuan, “MedChain: A design of blockchain-based system for medical records access and permissions management,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2952942.

[6] M. ASim, M. Petkovic, and T. Ignatenko, “Attribute-based encryption with encryption and decryption outsourcing,” *12th Aust. Inf.*

Secur. Manag. Conf. Held 1-3 Dec., vol. 2014 at Edith Cowan University, p. Western Australia., 2014, doi: 10.4225/75/57B65CC3343D0.

[7] M. Demir, O. Turetken, and A. Ferworn, “Blockchain Based Transparent Vehicle Insurance Management,” in *2019 Sixth International Conference on Software Defined Systems (SDS)*, Rome, Italy: IEEE, Jun. 2019, pp. 213–220. doi: 10.1109/SDS.2019.8768669.

[8] Z. Wan, Z. Guan, and X. Cheng, “PRIDE: A Private and Decentralized Usage-Based Insurance Using Blockchain,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax, NS, Canada: IEEE, Jul. 2018, pp. 1349–1354. doi: 10.1109/Cybermatics_2018.2018.00232.

[9] N. Nizamuddin and A. Abugabah, “Blockchain for automotive: An insight towards the IPFS blockchain-based auto insurance sector,” *Int. J. Electr. Comput. Eng. IJECE*, vol. 11, no. 3, p. 2443, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2443-2456.

[10] J. Sun, X. Yao, S. Wang, and Y. Wu, “Non-Repudiation Storage and Access Control Scheme of Insurance Data Based on Blockchain in IPFS,” *IEEE Access*, vol. 8, pp. 155145–155155, 2020, doi: 10.1109/ACCESS.2020.3018816.

[11] “Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf”, [Online]. Available: https://bitcoin.org/bitcoin.pdf

[12] Daraghmi, Daraghmi, and Yuan, “UniChain: A Design of Blockchain-Based System for Electronic Academic Records Access and Permissions Management,” *Appl. Sci.*, vol. 9, no. 22, p. 4966, Nov. 2019, doi: 10.3390/app9224966.

[13] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, Jun. 2020, doi: 10.1016/j.future.2017.08.020.

[14] E.-Y. Daraghmi, M. Abu Helou, and Y.-A. Daraghmi, “A Blockchain-Based Editorial Management System,” *Secur. Commun. Netw.*, vol. 2021, pp. 1–17, May 2021, doi: 10.1155/2021/9927640.

[15] E. Daraghmi, Y. Daraghmi, R. Daraghma, H. Fouchal, and M. Ayaida, “A Blockchain framework for Enhancing NB-IoT Security and Authentication: Health Monitoring System as a case,” Athens, 2022.

[16] E.-Y. Daraghmi, S. Jayousi, Y.-A. Daraghmi, R. S. M. Daraghma, and H. Fouchal, “Smart Contracts for Managing the Agricultural Supply Chain: A Practical Case Study,” *IEEE Access*, vol. 12, pp. 125462–125479, 2024, doi: 10.1109/ACCESS.2024.3439412.

[17] P. G. Bringas, I. Pastor-López, and G. Psaila, “BlockChain Platforms in Financial Services: Current Perspective,” *Bus. Syst. Res. J.*, vol. 11, no. 3, pp. 110–126, Nov. 2020, doi: 10.2478/bsrj-2020-0030.

[18] “Lightweight End-to-End Blockchain for IoT Applications,” *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 8, Aug. 2020, doi: 10.3837/tiis.2020.08.004.

[19] L. Zavolokina, N. Zani, and G. Schwabe, “Designing for Trust in Blockchain Platforms,” *IEEE Trans. Eng. Manag.*, vol. 70, no. 3, pp. 849–863, Mar. 2023, doi: 10.1109/TEM.2020.3015359.

[20] Sanjeewa Silva, “BETHEL Blockchain Platform,” 2020, doi: 10.13140/RG.2.2.19933.23529.

[21] H. Ye and S. Park, “Reliable Vehicle Data Storage Using Blockchain and IPFS,” *Electronics*, vol. 10, no. 10, p. 1130, May 2021, doi: 10.3390/electronics10101130.

[22] S. Bhatt, S. Hotchandani, K. Gaur, and S. Sirsikar, “Introduction to LifeBlocks: A Blockchain based Insurance Platform:,” in *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications*, Lieusaint - Paris, France: SCITEPRESS - Science and Technology Publications, 2020, pp. 77–81. doi: 10.5220/0009854000770081.

[23] L. Bader, J. C. Burger, R. Matzutt, and K. Wehrle, “Smart Contract-Based Car Insurance Policies,” in *2018 IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018, pp. 1–7. doi: 10.1109/GLOCOMW.2018.8644136.

[24] C. Eckert, C. Neunsinger, and K. Osterrieder, “Managing customer satisfaction: digital applications for insurance companies,” *Geneva Pap. Risk Insur. - Issues Pract.*, vol. 47, no. 3, pp. 569–602, Jul. 2022, doi: 10.1057/s41288-021-00257-z.

[25] F. Loukil, K. Boukadi, R. Hussain, and M. Abed, “CioSy: A Collaborative Blockchain-

Based Insurance System," *Electronics*, vol. 10, no. 11, p. 1343, Jun. 2021, doi: 10.3390/electronics10111343.

[26] A. Hombalimath and N. Mangla, "Blockchain Based Secured Vehicle Insurance Framework using Hyperledger Composer," Sep. 16, 2022. doi: 10.21203/rs.3.rs-2060348/v1.

[27] R. Selvakumar, S. Shibu, R. P. Joy, R. C. R, C. R. Kumar, and S. Kamatchi, "A Federated Consensus for Proof of Authority in IoT-Blockchain Applications," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 11, no. 9, pp. 328–334, Oct. 2023, doi: 10.17762/ijritcc.v11i9.8358.

[28] W. Stallings, *Cryptography and network security: principles and practice*, 4. ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006.

[29] Y.-A. Daraghmi, B. Yahya, and E. Y. Daraghmi, "Has Covid-19 Affected Software Usability: Mobile Accounting System As A Case," *J. Theor. Appl. Inf. Technol.*, vol. 101, no. 2, pp. 785–794, 2023.

[30] Y.-A. Daraghmi and E.-Y. Daraghmi, "RAPD: Rapid and Participatory Application Development of Usable Systems During COVID19 Crisis," *IEEE Access*, vol. 10, pp. 93601–93614, 2022, doi: 10.1109/ACCESS.2022.3203582.

[31] Y.-A. Daraghmi, "Vehicle Speed Monitoring System Based on Edge Computing," Gaza: IEEE, 2021.

[32] Y.-A. Daraghmi, M. A. Helou, E.-Y. Daraghmi, and W. Abu-ulbeh, "IoT-Based System for Improving Vehicular Safety by Continuous Traffic Violation Monitoring," *Future Internet*, vol. 14, no. 11, p. 319, Nov. 2022, doi: 10.3390/fi14110319.