Utilizing Blockchain Technology to Secure Database from Unauthorized Modifications

Cu Vinh Loc, Tran Thi Dinh Xuyen, Truong Xuan Viet, Tran Hoang Viet, Le Hoang Thao, Truong Minh Thai
Can Tho University, Can Tho, Vietnam

Correspondence: Cu Vinh Loc, cvloc@ctu.edu.vn
Received: 14/07/2024, revised: 03/10/20224, accepted: 09/10/2024
Digital Object Identifier: 10.32913/mic-ict-research-vn.v2024.n3.1330

Abstract: Centralized database-based applications are frequently susceptible to internal threats. The data stored in a relational database can be changed by any user who has administrative access to the hosting server or database system. Moreover, this kind of user may alter the associated log files, which would make it very challenging to identify any illegal alteration. It would also be difficult to attribute the assault to privileged users as well. In this study, we propose a solution to build a service on the context of student score management based on Blockchain technology. A database system based on the Blockchain platform along with accompanying services is built to allow storage and query of student learning outcomes. The database security system operates in parallel with the current system while still ensuring transparency, security and integrity of data. We develop a verification module that uses Blockchain to verify the integrity of the corresponding tuples and identify any unwanted modifications. Oracle database, RESTful API, and Hyperledger Fabric have all been used in the implementation of our method. Our findings show that the overhead of integrity checking remains constant per tuple in the results of a query and increases linearly using real data and SQL queries of different types and complexity.

Keywords: *Blockchain, hyperledger, centralized database systems tamper resistance features.*

Tiêu đề: **Giám sát nhiệt độ kho lạnh bằng phương pháp nội suy ba chiều**

Tóm tắt: Internet vạn vật (IoT) được ứng dụng rộng rãi trong nông nghiệp và công nghiệp. Bài báo này đề xuất một mô hình giám sát nhiệt độ ba chiều cho kho lạnh, tăng cường khả năng quan sát và phát hiện những thay đổi nhiệt độ bất thường trong kho. Sử dụng việc thu thập nhiệt độ theo thời gian thực và áp dụng các kỹ thuật nội suy tam tuyến, mô hình này xác định nhiệt độ tại tất cả các vị trí trong kho. Nó chấp nhận các giá trị nhiệt độ và vị trí cảm biến, phân đoạn không gian kho lạnh một cách thích hợp để áp dụng nội suy ba tuyến tính và xác định giá trị nhiệt độ tại mọi vị trí trong kho. Mô hình đã được thử nghiệm dựa trên số lượng cảm biến khác nhau và các vị trí cảm biến khác nhau. Dựa trên đó, chúng tôi cung cấp một giải pháp tối ưu cho số lượng cảm biến cần thiết cho kho lạnh.

Từ khóa: *Blockchain, hyperledger, centralized database systems tamper resistance features.*

I. INTRODUCTION

Enhancing the quality of management and training of educational and training institutions is one of the important goals for developing an effective and sustainable human resource. However, the current education and training system still has certain limitations, the security and storage of centralized data are increasingly improved, but the disadvantages have not been overcome. Private student records hold significant value for both individuals and universities alike. The existing training management system can guarantee that operational faults are kept to a minimum or that outside agent attacks are avoided. This is not the best course of action, though, as there are still issues with making sure that data is accurate during management and with detecting inaccurate or erroneous data. In particular, data can still be illegally or incorrectly altered during operation, and the current system has trouble identifying incidents. Certain vulnerabilities in the integrity and security of the current system have been made clear by the previously described considerations. Data is also not carefully controlled, and identifying and correcting problems in it requires a significant amount of time and effort. Specifically, there is a lack of transparency for fast fixes due to the way learning outcomes are maintained and kept, which makes it challenging to check and identify problems.

A malicious attack on a computer system or network carried out by an individual with authorized system access is known as an inside attack. Since insiders have authorized

access and may be knowledgeable about system policies and procedures as well as network architecture, they have a clear advantage over external attackers. Additionally, because many firms prioritize safeguarding against outsiders, there can be less security against insider attacks. Insider threats are those who intentionally or unintentionally create vulnerabilities or who use permitted access to an organization's cyberassets for malevolent or unauthorized reasons. They could be contracted workers, direct employees, or outside providers of computing and data services. Insider threat cases account for around 23% of all cybercrime occurrences, a ratio that has remained constant over the years despite a large increase in the overall number of attacks. Over 50% of firms report having experienced an insider cyberattack annually. The studies also reveal that insider assaults continue to be the hardest to identify, with 90% of firms believing they are vulnerable to insider threats.

In a relational database system, conventional integrity constraints refer to the process of guaranteeing the integrity of tuples in accordance with established constraints, such as data types of attribute values, foreign key constraints, etc. Malicious alteration of the tuples in a database management system (DBMS) is another source of integrity issues. Firewalls and access control measures are typically used to protect this data integrity. A DBMS's access control policies limit the number of users who are granted administrative privileges. The DBMS server cannot be directly accessed by an outsider thanks to firewalls. On the other hand, an internal modification occurs when a user with legal rights, such as an administrator, abuses their position to get or alter data. Internal modifications are now considered a serious and non-trivial threat to a database management system's ability to protect data integrity.

Passive or active internal alterations are both possible. While active modifications entail manipulating the data and logs to change query results, passive modifications involve obtaining and using the data in an unethical manner. Academic grade record tampering is a straightforward yet insightful example of the second form of attack, and it has been documented on several occasions in the recent past. In order to eliminate paper ballots from elections, several nations have embraced electronic voting machines. Every vote cast on an electronic voting machine is treated as a transaction by the embedded DBMS. Additionally susceptible to tampering and insider attacks are these computerized voting equipment. In terms of effective, varied, and scalable data storage options, traditional databases have advanced significantly over the previous few decades. The application has been effectively addressed by SQL query processing and optimization in conjunction with contemporary hardware. However, as previously mentioned, the modern issues include data security and integrity, of which detecting and preventing internal modification are essential components.

Blockchain is a new technology that offers high assurances of tamper resistance and immutability for decentralized data storage. Blockchains are comparable to native database management systems (DBMSs) that only append logs. But in an internal modification, the illegal users can change login records and logs to erase evidence of data manipulation because they have administrative access. Placing all of the databases on blockchain frameworks would be an easy fix. The majority of Blockchain frameworks and DBMSs enabled by Blockchain technology lack the robust SQL querying interface found in most contemporary DBMSs. Concern over data privacy is also growing as more and more data is being uploaded to public Blockchain networks.

We have found Blockchain being applied to solve issues in many fields like educational data security [1, 2], student certificate management [3], relational database security [4], and traceability management [2, 5–8]. Similar to the existing works, our approach allows storing student grades with authorization. The data will be checked and validated in off-chain database (Oracle) and on-chain database (Blockchain). Smart contract is used to maintain security and consistency. Figure 1 depicts the workflow of the proposed system.

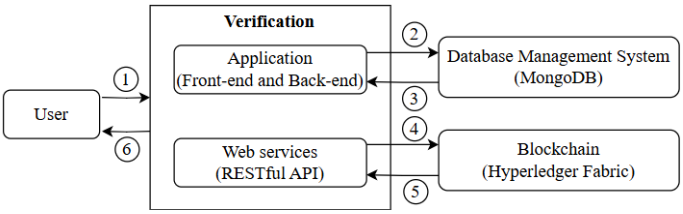

Figure 1. An overview diagram of the proposed approach.

The paper makes the following primary contributions.

- We develop Blockchain database, storage, and query services. The solution makes it feasible to validate and save course scores..
- A history of the creation and adjustment of course scores will be stored on the Blockchain throughout time.
- The query speed of Blockchain and the security of traditional databases are overcome when traditional data storage (including relational and non-relational) and Blockchain are combined.
- Consensus procedures along with the limitations and logic found in smart contracts guarantee that data is reliable and consistent at all times.

II. RELATED WORK

Ali *et al.* [9] put forward three strategies for deploying Blockchain technology to create a fully operational student information system that preserves transactions pertaining to student marks, course registration data, and the records of both students and instructors. Moreover, staying away from positions where data integrity is at risk, such as super administrator or centrally exposed storage. Employing the suggested models facilitates the transition to an electronic community in which legitimate certifications may be quickly generated and disseminated to interested parties without requiring the involvement of a centralized administration. In the work [1], the authors investigated the use of Ethereum's Proof of Authority (PoA) consensus method to create an agent-based Blockchain model system for protecting educational data against insider threats. The study established a basis for furthering the present comprehension of Blockchain systems, creating Blockchain models utilizing the technology's unchangeable characteristics, and integrating intelligent agents. Liang and Xu [10] have designed and implemented a decentralized system for managing student score sharing using blockchain technology. Furthermore, they guarantee the traceability, tamper-proof, distributed storage, and security of the query process by utilizing the Ethereum smart contract to control on-chain and off-chain data storage in addition to on-chain inquiry and verification. The study concludes with examining the development of a prototype system based on the Ethereum framework. The outcomes of the experiment demonstrate the viability of the anti-tampering and traceability plan put forth in this work. Blockchain technology can completely meet the needs of different stakeholders involved in student performance management. Student score information is completely protected because each subject's data is encrypted using a unique private key.

With the introduction of the work [4], the cloud relational database (RDB) known as BC over cloud-RDB received an improved structure based on blockchain technology (BC). It allows the client to identify and stop incorrect RDB manipulation through a self-verification method. Agile BC-based RDB and secure BC-based RDB are the two systems that the authors proposed to enhance cloud-RDB. Based on the Byzantine Fault Tolerance consensus, both are divided among a number of cloud service providers. Furthermore, both depend on employing the SHA-256 to connect records to one another. In addition, a proof-of-work consensus is used by secure BC-based RDB to prevent data offensive operations. Based on security analysis and system performance, the agile BC-based RDB is strongly recommended for the high throughput database. In order to preserve control and ownership of the credentials, a framework for issuing student certifications locally as well as online has been proposed by Ghani *et al.* [3]. With the use of blockchain technology, this framework facilitates electronic certification sharing and verification. The implementation of the blockchain-based certification system in universities is expected to result in reduced latency for the purposes of certification issuance, sharing, and verification. In the study, a blockchain-based framework for exchanging e-certifications is suggested. The framework is evaluated by calculating the latency time between transactions and the average time it takes to issue a certificate.

In another work [11], the authors introduce an Ethereum smart contract-based safe datastore. Initially, the authors investigate the degree to which a datastore powered by smart contracts ought to mimic a conventional database system. Second, they look into the best way to minimize gas consumption and maximize flexibility while storing the data in a datastore that is based on smart contracts. Third, they look for clarification on whether sophisticated processing like data encryption and data analytic algorithms should be included in a datastore built on smart contracts. The authors also come to the conclusion that because of the cost and security risks, sophisticated processing shouldn't be permitted in smart contracts. Furthermore, traceability data is also managed using blockchain. Cao *et al.* [7] examine how blockchain technology might support sustainable food supply chains, which adds to the growing body of study in this area. This study suggests a blockchain-enabled architectural framework for reliable communication about the sustainability features of food goods in an effort to promote the use of blockchain for sustainability in the context of the food supply chain. The multi-layered architecture for using blockchain-based traceability to communicate to customers the reliable sustainability features of a specific food product is outlined in the blockchain-enabled architectural framework. This study provides useful information for industry 4.0 researchers and practitioners, bolstering sustainability communication with blockchain-based traceability.

III. METHODOLOGY

The overview of the proposed system can be described as follows. Users will interact with the system via the website. Users will be provided with all functions depending on their role in the system. The system is built with the following components: NodeJS Server, Oracle Database, Hyperledger Fabric blockchain and client using ReactJS. The system uses ReactJS from the web browser using RESTful API (GET, POST, PUT, DELETE) to interact with the server via HTTP method. For the function of saving mark data. Student test marks will be saved simultaneously in the

Oracle database and the blockchain database. When data is written to the blockchain database, the system will not interact directly with the database, instead Hyperledger Fabric designs an interaction model through smart contracts (also known as chaincode) and the communication will be controlled. The blockchain database will store information including student information and academic results, student information. All other data will be stored in the Oracle database. While operating the system, Oracle database will be prioritized because it will have higher retrieval speed, for the use case blockchain database is only used when creating new users, storing student marks.

1. System architecture

The system architecture is as depicted in the Figure 2, where the Hyperledger Fabric network is configured, launched and managed by Docker, "Blockchain Operator" acts as the identity manager and Hyperledger fabric network. The user can interact with the application through ReactJS. The system will interact with the server using NodeJS. To interact with the blockchain network, the software development kit (SDK) will be used for the above purpose.

Figure 2 is an overview of the architectural system for a Hyperledger Fabric network that includes a blockchain operator, Docker, Fabric SDK, ReactJS, and users. This architecture outlines how these components interact with each other in a decentralized application. The blockchain operator manages the Hyperledger Fabric network, including deploying and maintaining the network components like peers, orderers, and certificate authorities (CAs). The blockchain operator's responsibilities are to configure and manage network settings, monitor the network's health and performance, manage chaincode deployment and updates, and ensure security and compliance within the network. Docker is used to containerize the various components of the Hyperledger Fabric network, enabling easy deployment, scaling, and management. Docker responsibilities consist of creating isolated environments for peers, orderers, and CAs, ensuring that dependencies and versions do not interfere with one another, facilitating the rapid deployment of network components using Docker Compose, and enabling the running of multiple instances of the blockchain network for development, testing, and production scenarios. The Fabric SDK provides developers with the tools to interact programmatically with the Hyperledger Fabric network. It is responsible for facilitating application development by providing APIs for common tasks (e.g., querying the ledger, submitting transactions), handling user authentication, endorsement, and transaction lifecycle, and supporting integration with client applications written in various programming languages.

ReactJS serves as the frontend framework for building user interfaces that interact with the Hyperledger Fabric network. Its responsibilities include providing a dynamic and responsive user experience for end-users, facilitating communication with the backend services through API calls, displaying the results of transactions, queries, and historical data in a user-friendly manner, and handling user authentication and session management. Users are the individuals or systems that interact with the application built on top of the Hyperledger Fabric network. Users are responsible for performing actions such as submitting transactions, querying data, and viewing historical records, participating in the blockchain ecosystem by managing their credentials and interacting with the smart contracts (chaincode), and utilizing the application's interface (built with ReactJS) to engage with the blockchain.

When users interact with the system using chaincode (smart contracts), data is queried or modified within a component that Hyperledger Fabric (or any other blockchain platform) refers to as a ledger. As its name suggests, the ledger maintains a record of all data changes from the system's inception to the present. In other words, the ledger has both the current value of the properties of the objects and the history of changes that led to this current value. The data will not be stored in the traditional way but will be encrypted and then stored in blocks in the ledger.

Figure 3 shows the architecture of a ledger stored on a peer of the system. A ledger in Hyperledger Fabric consists of two distinct parts like the world state and the blockchain. Each component represents a set of facts about a set of stored objects. First, the world state – a database that holds the current values of a set of ledger states. The world state makes it easy for a program to directly access the current value of a state instead of having to calculate it by going through the entire transaction log. By default, ledger states are represented as key-value pairs and we will see later how Hyperledger Fabric provides flexibility in this regard. The world state can change frequently as states can be created, updated and deleted. Second, the blockchain – which contains a transaction log that records all the changes that led to the current state of the world. The transactions are collected inside blocks that are attached to the blockchain. This allows for a historical understanding of the changes that led to the current state of the world. The blockchain data structure is very different from the current state of the world because once it is written, it cannot be modified because it is immutable. In short, once the data is stored in the blockchain, only the word state in the ledger changes, all data stored in the blockchain cannot be changed

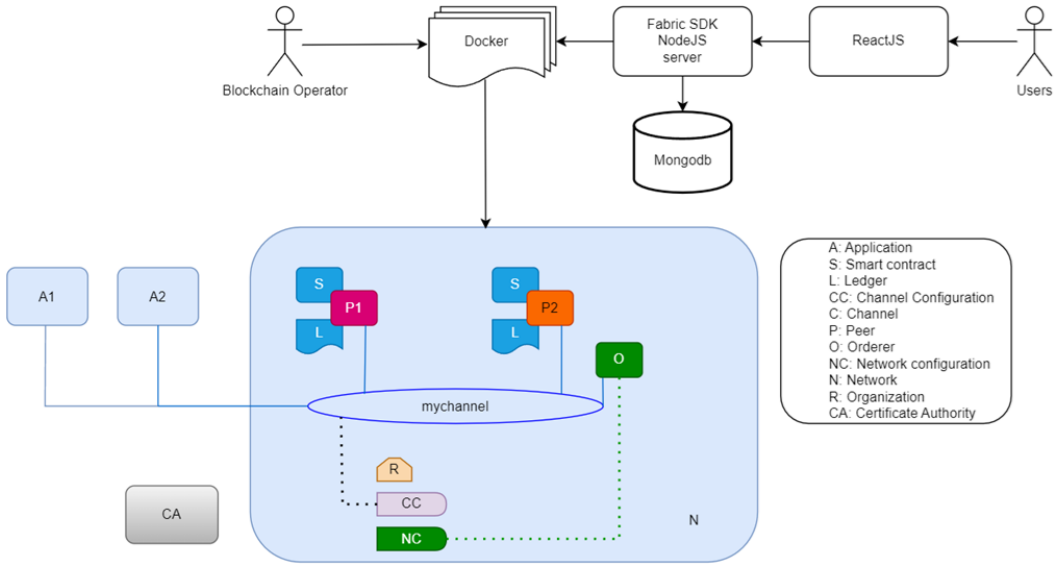

Figure 2. The architecture of the proposed system.

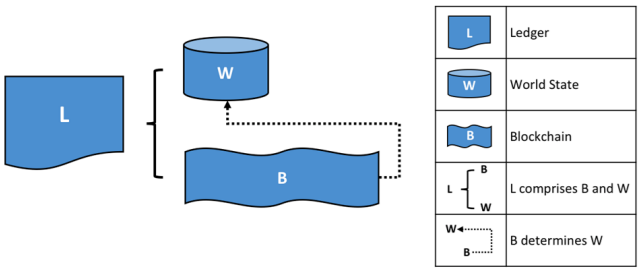

Figure 3. Architecture for hyperledger fabric storage (Source: hyperledger-fabric.readthedocs.io).

but can only be updated.

The system interaction flow can be describes as follows.

1) User Interface: Users interact with the application through a web-based interface created with ReactJS. They can log in, submit transactions, and query data.
2) ReactJS to Fabric SDK: When a user performs an action (like submitting a transaction), the ReactJS app sends the request to the backend, which utilizes the Fabric SDK to communicate with the Fabric network.
3) Fabric SDK to Blockchain Network: The Fabric SDK handles the logic to create transaction proposals, send them to the endorsing peers, collect endorsements, and submit the transactions to the orderer.
4) Docker Management: The Ledgers, peers, orderers, and other components of the Fabric network run in Docker containers. The blockchain operator manages these containers using Docker and Docker Compose, facilitating easy orchestration.
5) Orderer and Peers Processing: The ordering service collates transactions and broadcasts them to all peers. Peers validate the transactions and commit them to their local ledgers.
6) Data Retrieval: Users can query the blockchain to retrieve historical data. ReactJS forward the requests to the Fabric SDK, which fetches the requested data from the ledger and sends it back to the user interface for display.

Algorithm 1 Data Processing Flow in Hyperledger Network

```
Initialize Network Components
Define Chaincode
Deploy Chaincode
Prepare Client Application
Client Prepares Transaction Proposal
Send Proposal to Endorsing Peers
if Endorsementsarevalid then
    Submit Endorsed Transaction to Orderer
    Orderer Distributes Block to Peers
    Peers Validate Transactions
    Commit Transaction to Ledger and Update State
    Client Queries Updated World State
    Implement Logging and Monitoring
else
    Handle Invalid Endorsements
end if
```

2. Data Processing Flow in Hyperledger Fabric

To query or perform functions, the system uses the SDK to query and update the ledger. The steps are as follows.

- Send transaction: Client sends transaction information including sender information to Hyperledger network. SDK takes sender and transaction information to package and propose transaction. After packaging, transaction will be sent to peers in the system for execution.
- Verify signature and execute transaction: The system checks the transaction information for validity before executing the transaction. If the information is invalid, an error will be reported. After being validated, peers will call the chaincode and use the input data to process the data. After executing the transaction, the chaincode will generate the transaction result. The value of the transaction just created will be sent back to the SDK as a "proposal response".
- SDK compares proposal responses: After receiving the return value from the peer, the system will check the validity of the return value (identity of the peer, return result). If the return transaction is valid, the system checks whether the transaction created is a query or valid. The query transaction will send the result to the client and end the execution process, otherwise it will send the transaction to the ordering service.
- Collect transactions and save them to ledger: The SDK will send data to the ordering service (including transaction proposal and transaction response). The data sent will be in the form of a "Transaction Message", which contains key-value pairs for the object to be updated. The Ordering Service will arrange the sent transactions in chronological order.
- Authenticate transaction: Transactions after successful arrangement will send updated data to peers. Each transaction will be then tagged as valid or invalid.
- Update ledger: Each peer will add transactions to the block when the transaction is valid. For each valid transaction, data will be written to the ledger. The updated data will be sent to the system so that users can know.

Algorithm 1 is a high-level algorithm that outlines the general steps involved in data processing within a Hyperledger network, particularly focusing on Hyperledger Fabric.

3. Function to save marks to blockchain

The following requirements are met by the terms of the smart contract.

- (1) Check transaction: Verify the mark's existence in the blockchain, as well as the student's academic performance. If the performance is already recorded in the blockchain, report a mistake.
- (2) Check user identity: Verify whether the certificate authority has registered the user identity or not. The system will alert the user to a mistake if they are not in the roles of administrator, instructor, or rector.
- (3) Check the right to update marks: The system will determine whether or not the user still has the authority to save or amend marks if they are a teacher. The system will alert users to an issue if the time for updating the mark has passed.
- (4) Check semester results: Verify whether or not student learning outcomes have been generated for the semester; if they have, the system will display an error.
- The following will be the description of the conditions of the smart contract.

Algorithm 2 Save Student Marks to Blockchain

Require: StudentID, Subject, Marks
Ensure: Transaction is recorded on the blockchain

Initialize Blockchain Network Components
Define Chaincode for Student Marks
Deploy Chaincode on the Blockchain Network
Prepare Client Application with User Credentials
Create a Transaction Proposal
Set Proposal Inputs: `(StudentID, Subject, Marks)`
Send Proposal to Endorsing Peers
Receive Endorsements from Peers
if Endorsements are valid **then**
    Submit the Endorsed Transaction to the Orderer
    Wait for the Orderer to create a Block
    Receive Block from Orderer
    Distribute Block to all Peers
    Commit Transaction to Ledger
    Update World State with Student Marks
    Log Transaction for Auditing
else
    Handle Invalid Endorsements
    Notify User of Failure
end if

The algorithm 2 outlines the steps needed to create a transaction that records student marks on a blockchain, such as Hyperledger Fabric. The algorithm includes steps for preparing the data, creating a transaction proposal, endorsing it, and committing it to the blockchain.

4. Function to update student marks

The smart contract conditions for updating student marks are described below.

- (1) Check user identity: Verify whether the certificate authority has registered the user identity or not. The system will alert the user to a mistake if they are not in the roles of administrator, instructor, or rector.

- (2) Check the right to update student marks: The system will determine whether or not the user still has the authority to save or amend marks if they are a teacher. The system will alert users to an issue if the time for updating the student mark has passed.
- (3) Check semester results: Verify the semester's learning outcomes; if there are no student learning outcomes, the system will flag an error.
- The conditions of the smart contract are described as follows.

Algorithm 3 Update Student Marks on Blockchain

Require: StudentID, Subject, NewMarks
Ensure: Updated Transaction is recorded on the blockchain

Initialize Blockchain Network Components
Prepare Client Application with User Credentials
Retrieve Current Marks from Blockchain using `GetMark` function
if Current Marks exist **then**
    Create an Update Transaction Proposal
    Set Proposal Inputs: `(StudentID, Subject, NewMarks)`
    Send Proposal to Endorsing Peers for Validation
    Receive Endorsements from Peers
    if Endorsements are valid **then**
        Submit the Endorsed Update Transaction to the Orderer
        Wait for the Orderer to create a Block
        Receive Block from Orderer
        Distribute Block to all Peers
        Commit Updated Transaction to Ledger
        Update World State with NewMarks
        Log Update Transaction for Auditing
    else
        Handle Invalid Endorsements
        Notify User of Failure
    end if
else
    Notify User that No Current Marks exist for Update
end if

The algorithm 3 provides a comprehensive approach to updating student marks on a blockchain, ensuring that all necessary checks and validations are performed to maintain the integrity and security of the data.

5. Function to query the history of mark update

The following is a description of the smart contract conditions used to query the mark updating history.

- (1) Check user identity: Verify whether the certificate authority has the user identity registered. The system will raise an error if the user does not have the role of admin or rector, or does not have access to the smart contract.
- (2) Check learning outcome data.
- The following is a description of the smart contract's requirements for retrieving the mark update history

The relationships between the various verification components for every given SQL query are depicted in Figure 1. The SQL processor parses a SQL query that was started via the web application interface (①) and sends it to the database management system (DBMS) with the deliberate changes needed to ensure integrity (②). Tuples matching this amended query are returned by the DBMS (③). The integrity checking stage of verification is subsequently completed by comparing these tuples to the matching fingerprints that are kept on the blockchain (④, ⑤). Verification provides the end user with the results of the initial SQL query when the check has been successfully completed. As a result, the end user cannot see the integrity checking process and is only informed if the integrity check is unsuccessful. If INSERT, UPDATE, or DELETE requests contain nested SELECT queries, steps ② through ⑤ might need to be performed twice.

Algorithm 4 Query Student Marks Update History

Require: StudentID, Subject
Ensure: History of Marks Updates for the specified StudentID and Subject

Initialize Blockchain Network Components
Prepare Client Application with User Credentials
Create a Query Request
Set Query Inputs: `(StudentID, Subject)`
Send Query Request to the Ledger
Receive Query Results from the Ledger
if Query Results are not empty **then**
    Display the History of Marks Updates
    for each Transaction in Query Results **do**
        Extract Transaction Details:
        Print TransactionID, OldMarks, NewMarks, Timestamp, UpdateBy
    end for
else
    Notify User that No History Exists for the Given StudentID and Subject
end if

The algorithm 4 is used for querying the history of student mark updates on a blockchain. This algorithm outlines the process for retrieving and displaying the transaction

history related to a specific student's marks, including all past updates

IV. EXPERIMENTAL RESULTS

We have put our strategy into practice and tested it in the given circumstance. Imagine an educational setting where grades must be assigned to students. Every semester, the institution hosts a number of courses. There are several enrolled students and one or more staffs teaching each course.

Computation setup: : Hyperledger Fabric was deployed using docker version 27.0.3. We use Oracle as the underlying DBMS and the NodeJS programming language to construct our SQL parser. For testing, Hyperledger Fabric is deployed on one system, and queries and requests are parsed and sent to Hyperledger on another. The machines were networked together using a LAN cable.

Database: : Any relational database can be used to record the grades and enrollment information. Oracle has been utilized in our research. The database's tables are related to the following entities: staffs, students, and courses. The enrollment table, which contains data about students enrolled in several courses taught by a certain staff member, serves as a marker of their relationship. In our case, the enrollment table contains student grade information that needs to be kept private. Since only the staff is permitted to alter grades in our situation, as previously said, we have updated the enrollment table to add columns for the staff's identify and transaction ID. Oracle views are used to restrict which sections of the table are accessible to which entities. This is carried out in order to guard against any illegal access to the data kept in the database.

Web application: : To facilitate communication with the users, we create a web application. NodeJS has been utilized in the application's implementation. Every employee and student has a local web server that they use to retrieve and fill in data from the Oracle database by connecting with their username and password. While staff can alter a student's grade for any student, each student can only view his own grades.

Blockchain: : We are utilizing Hyperledger Fabric for a distributed blockchain because it offers a natural abstraction for use cases involving permissioned blockchains and concentrates on generic data retrieval, timestamping, and archiving rather than the exchange of assets between members. Installing Hyperledger and using our script to connect to a central Hyperledger node is what we expect from each participant. Next, an address for the chain is given to each member, which may be used to grant rights. Students can validate their grades in accordance with protocol because the stream's permissions are distributed based on staff members' addresses that were obtained from public key infrastructure. This permits them to append transactions and mine blocks while permitting everyone to view the blockchain. The verification is conducted as follows.

Every time a student accesses his grades, the server retrieves data from the central database, which also includes the transaction id and identifier. The transaction id found in the row is used by the application to get the transaction from the log-stream. The public key is retrieved by the application using the identification from the public key infrastructure. After then, this public key is used to validate the transaction's signature. The student can view his marks if the signature is successfully validated and the data's hash matches the hash included in the transaction. If any of the aforementioned procedures are not followed, a notification is sent to the relevant authorities to investigate any unauthorized changes made to the database and an alarm is raised on the student's end.

The outcomes of every course the student completes are represented in the Table I. The information will first be stored on the blockchain and then externally as Oracle. The database has a feature called PERMISSIONGRANTED that lists the staffs and administrators who are authorized to modify the learning outcomes of their students. The most recent data operation on the course results is represented by the ACTION field, which is created when the smart contract is called. Its values include create (new course results), update (update course results), and change access (change the right to alter learning results).

The hash of the tuple (SUBJECT_MS, STUDENT_MS, TEACHER_MS, SCORE) is signed on the backend whenever a staff member inserts grades for any student (INSERT query), and the signature is broadcast on log-stream. In its internal log, the application keeps track of the actual inquiry. The application collects the grade's real data and pushes the query to the database after attaching the staff member and transaction identifiers once this transaction has been mined in a block and a predetermined number of confirmations have been obtained.

We perform 43 queries with various complexities on the information obtained from the training management system at Can Tho University. In comparison to Hyperledger lookup, we have found that the time required to conduct queries on Oracle is extremely small. There is a linear relationship between the overall query runtime and the number of tuples a query affects. The average Hyperledger lookup time for a SELECT query is 0.07–0.3 seconds per tuple, but queries that need to modify tuples, such INSERT, UPDATE, and DELETE, often take 0.9–3.1 seconds per tuple. The peer-consensus protocol, which requires adding a fingerprint or changing the number of tuples per table on

Table I
THE STRUCTURE OF BLOCKCHAIN DATA DESIGN.

Attribute name	Data type	Description
Result_Key	varchar(50)	Composite Key learning outcomes
ID	varchar(20)	ID of learning result
GROUP_MA	varchar(10)	Course group identification number
SUBJECT_MS	varchar(10)	Course identification number
STUDENT_MS	varchar(10)	Student identification number
STUDENT_NAME	varchar(20)	Student name
TEACHER_MS	varchar(10)	Staff identification number
SCORE	int	Course grades
PERMISSIONGRANTED	varchar(20)	List of score editors
ACTION	varchar(20)	Type of operation for the course results

the Hyperledger network, is the reason for the longer query times for the tuple updates. In addition, we also conducted 30 experiments by intentionally changing students' grades by directly interfering with the database. The system was able to detect all these unauthorized changes by comparing with the data stored on Hyperledger.

This thorough investigation shows that the average Hyperledger lookup time for a SELECT query is 0.08–0.1 seconds per tuple, but the average search time for queries that need to modify tuples, including INSERT, UPDATE, and DELETE, is roughly 0.8–2.5 seconds per tuple. The peer-consensus protocol's requirement to add a fingerprint or change the number of tuples per table on the Hyperledger network is the reason for the longer query times for tuple updates.

The findings suggest that using blockchains in conjunction with DBMSs leads in overheads that don't seem ideal for high-performance throughput. However, our goal is to be without transferring all of the DBMS data to a blockchain, and thus maintain privacy, whereas we identify an internal modification by using a blockchain's tamper-resistance features.

V. CONCLUSION

In this work, we have introduced a blockchain-based method for identifying internal modification and other unapproved database entry modifications. Because a hash value and a public key are stored in each row in addition to the regular data, the system that is being shown has a low storage overhead. Verification process can assign the modifications to the appropriate owner for any queries that alter the data that are processed through it. Because verification verifies each query's tuple integrity against the blockchain, there is some latency. However, we believe that this latency can be accepted rather than jeopardizing the integrity of the data in systems where the DBMS tuples' integrity is crucial since essential decisions are relied on them. For future work, we are going to improve the time of data verification process in case of extremely large amount of data.

REFERENCES

[1] M. Ubaka, A. Azeta, A. Oni, H. Okagbue, and N. Omoregbe, "Securing educational data using agent-based blockchain technology," *International Journal of Scientific & Technology Research*, vol. 9, pp. 2936–2938, 01 2020.

[2] T. K. Agrawal, V. Kumar, R. Pal, L. Wang, and Y. Chen, "Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry," *Computers and Industrial Engineering*, vol. 154, p. 107130, 2021.

[3] R. Ghani, A. Al-karkhi, A. Khudhair, and L. Aljobouri, "Blockchain-based student certificate management and system sharing using hyperledger fabric platform," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 10, p. 207, 04 2022.

[4] R. Awadallah and A. Samsudin, "Using blockchain in cloud computing to enhance relational database security," *IEEE Access*, vol. 9, pp. 137 353–137 366, 2021.

[5] H. Wu, S. Jiang, and J. Cao, "High-efficiency blockchain-based supply chain traceability," 2022.

[6] F. Casino, V. Kanakaris, T. K. Dasaklis, S. J. Moschuris, S. Stachtiaris, M. Pagoni, and N. P. Rachaniotis, "Blockchain-based food supply chain traceability: a case study in the dairy sector," *International Journal of Production Research*, vol. 59, pp. 5758 – 5770, 2020.

[7] S. Cao, H. Johnson, and A. Tulloch, "Exploring blockchain-based traceability for food supply chain sustainability: Towards a better way of sustainability communication with consumers," *Procedia Computer Science*, vol. 217, pp. 1437–1445, 2023, 4th International Conference on Industry 4.0 and Smart Manufacturing.

[8] G. Varavallo, G. Caragnano, F. Bertone, L. Vernetti-Prot, and O. Terzo, "Traceability platform based on green blockchain: An application case study in dairy supply chain," *Sustainability*, vol. 14, no. 6, 2022.

[9] S. I. M. Ali, H. Farouk, and H. Sharaf, "A blockchain-based models for student information systems," *Egyptian Informatics Journal*, vol. 23, no. 2, pp. 187–196, 2022.

[10] X. Liang and S. Xu, "Student performance protection based on blockchain technology," *Journal of Physics: Conference Series*, vol. 1748, p. 022006, 01 2021.

[11] I. M. Aldyaflah, W. Zhao, H. Upadhyay, and L. Lagos, "The design and implementation of a secure datastore based on ethereum smart contract," *Applied Sciences*, vol. 13, no. 9, 2023.

Cu Vinh Loc received a degree in computer engineering from Can Tho University, Vietnam, in 2001. In 2009, He received his master's degree in Computer Science from the Asian Institute of Technology, Thailand. He received his PhD in Computer Science from the University of La Rochelle, France in 2019. He is currently a lecturer at Can Tho University. His research interests include data hiding, document analysis and recognition, natural language processing, blockchain.
Email: cvloc@ctu.edu.vn

Tran Thi Dinh Xuyen graduated from Can Tho University in 2010 with a degree in Information Technology. Currently, she is in charge of managing and operating the Monitoring and Operation Department (operating the Monitoring and Operation Center of Soc Trang province), the Digital Technology Center under the Department of Information and Communications of Soc Trang province.

Truong Xuan Viet, a former Computer Science Engineer at Can Tho University, holds a Master's degree from Marne-La-Vallée University (France) and a Ph.D. from Paris 6 University (France). His research focuses on a diverse range of areas, including Geographic Information Systems (GIS), Remote Sensing, Agent-Based Simulation, Internet of Things (IoT), Networking, and Machine Learning. He has leveraged his knowledge to develop innovative solutions for Complex Systems, such as Smart Campus, Environmental Management, and Spatial Data Infrastructure systems.
Email: txviet@ctu.edu.vn

Tran Hoang Viet got his first degree, Engineer in Information Technology, from CanTho University of Technology (CTU) in 2000 and had been working for the Faculty of Information Technology until 2001. He had become a researcher at Cantho University Software Center until 2010. Then, he was awarded a Master's scholarship from the MOET of Vietnam and studied at the HAN University of Applied Sciences, Netherlands until October 2011. Then, he was awarded a PhD scholarship from Soongsil University, Korea. His PhD research topic is related to "3D Stereoscopy Techniques in Subtitle, Face Detection, and Simulation Machine". He received the PhD degree in Media Engineering in August 2017.
Since August 2017, he has been working as a researcher at the Cantho University Software Center. He became a lecturer at CIT in 2019 and has been a deputy director there since November 2021.
His research interests focus on Machine Learning, Deep Learning, Computer Vision, Computer Graphics, and Software Engineering.
Email: thviet@ctu.edu.vn

Le Hoang Thao received a degree in computer engineering from Can Tho University, Vietnam, in 2000. In 2024, he received his master's degree in Computer Science from the Asian Institute of Technology, Thailand. He is currently a lecturer at Can Tho University and also the director of Can Tho University Software Center. His research interests includes IoT, Solutions for university digital transformation and E-government.
Email: lhthao@ctu.edu.vn

Truong Minh Thai is a former Bachelor of Engineering in Computer Science student at Can Tho University. He obtained a Master's in Information System Development from HAN University (2010), Netherlands, and his Doctorate in Computer Science from Toulouse 1 Capitole University (2015), France. In his research, he concentrates on Data Warehousing, Multi-Agent-Based Systems, Blockchain, and the Internet of Things (IoT), applying these technologies to enhance software development in the fields of agriculture and aquaculture.
Email: tmthai@cit.ctu.edu.vn