**MITIGATE INSIDER ATTACKS USING BLOCKCHAIN NETWORK**

Submitted By:

Harish Udhay Kumar

Swathi Doraiswamy Gopal

Pragun Mittal

**Contents**

[**Abstract**](#_p0uansv7w1ng) **2**

[**Introduction**](#_7g2j3o3b4ckg) **2**

[DataBase](#_bwkr5vchkgnj) **4**

[Multichain Stream Functionality](#_h7qughcix81i) **5**

[**Methods**](#_cfzxrp5wryqv) **6**

[Signup And Login](#_4w7e0nimmad7) **6**

[Inserting Query](#_jzryx94bcbvk) **6**

[Updating Query](#_v7szer82zb4d) **6**

[**Detecting an Insider Attack**](#_3w1wylokn6dx) **7**

[**Security checks**](#_2xpl7mk6adzf) **7**

[**Possible inclusions**](#_ah0xdsy5x7gk) **7**

[**Results and Discussion:**](#_52nizmb094a5) **8**

[**Contributions**](#_y24hsffigd53) **9**

[**Conclusions**](#_9jpxcp145q4c) **9**

[**Appendix**](#_jk4ux6gcth40) **9**

[**Reference**](#_rjd0utu4q6r7) **10**

# **Abstract**

Insider attacks are common in applications that rely on centralized databases.Insider threat is the potential for an insider to use their authorized access or understanding of an organization to harm that organization. For Example, They could be a departing employee stockpiling data to get a leg up in their next job, a negligent remote worker connected to an unsecured network, or several other kinds of individuals.

Any user with administrative privileges on the Database system has the ability to modify database entries. Here, this issue is addressed by leveraging Blockchain's tamper resistance. Multichain is used to implement and test the solution on an academic grading application.

The goal of this project is to create a blockchain network and an application over the network to secure every database transaction and to identify attempts to attack a group or organization by individuals who have access to information that isn't available to others.

Keywords: Block chain, Multichain, insider attack, Database.

# **Introduction**

Blockchain technology is a sophisticated database mechanism that enables the transparent sharing of information within a business network. Data is stored in blocks that are linked together in a chain in a blockchain database. Because one cannot delete or modify the chain without network consensus, the data is chronologically consistent. All new information that follows that newly added block is compiled into a newly formed block, which is then added to the chain once it is complete. As a result, blockchain technology can be used to create an immutable ledger for tracking orders, payments, accounts, and other transactions. The system includes mechanisms for preventing unauthorized transaction entries and ensuring consistency in the shared view of these transactions.

However, insider attacks are taking the lead. Once an insider threat has occurred, tracing and obtaining evidence becomes extremely difficult. The attacker is a legitimate user and may be system administrators. As a result, the internal network's security facilities are unable to monitor or prevent such attacks. This paper implements and tests an approach to mitigate insider attacks in blockchain on an academic grading application

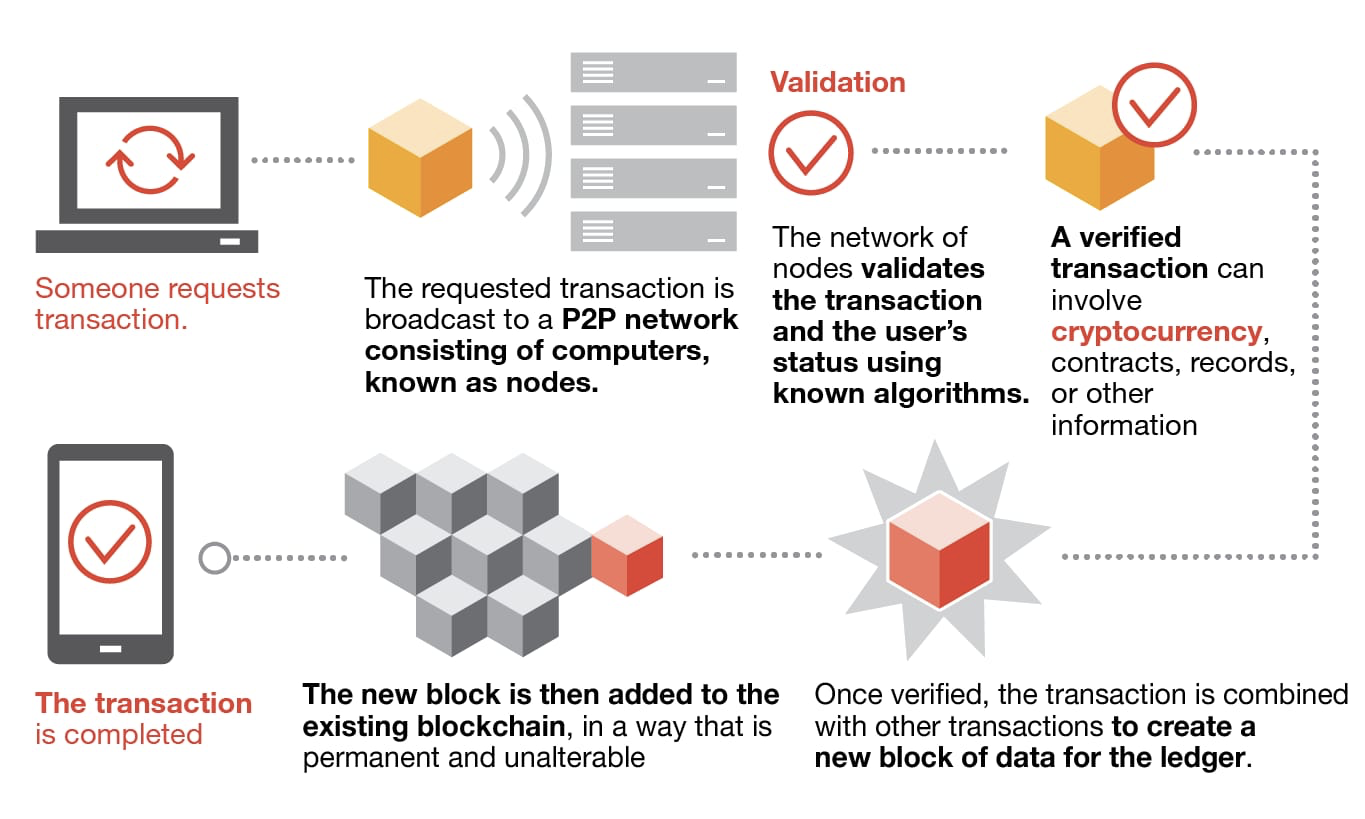
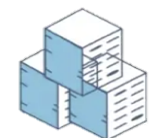


Fig1: Blockchain Concept

In this project Apache’s XAMPP server is used to host the application on localhost and to create the database in PHPMYADMIN of the local host. The existing framework called Multichain is used to build the blockchain network. The project is run on an Linux environment



1

2

3

4

5

Blockchain

Database

Server

Grading Application

Fig2: Architecture of the proposed system

The above figure represents the architecture of the system proposed. The system consists of 4 main components Grading Application, Server, Database and blockchain.

1. The application connects to the server with a request to insert/update the data
2. The server inserts/updates the data in the database
3. The database sends ACK to server
4. Server sends this ACK to the host/application
5. The application updates the transaction onto a node in blockchain

## **DataBase**

There are 3 tables in our database. They are:

1. Creds(credential)
2. Grades
3. Instructors

Creds: This table consists of students name, user id , salt and hash values of their credentials. The student's information is entered by the admin.

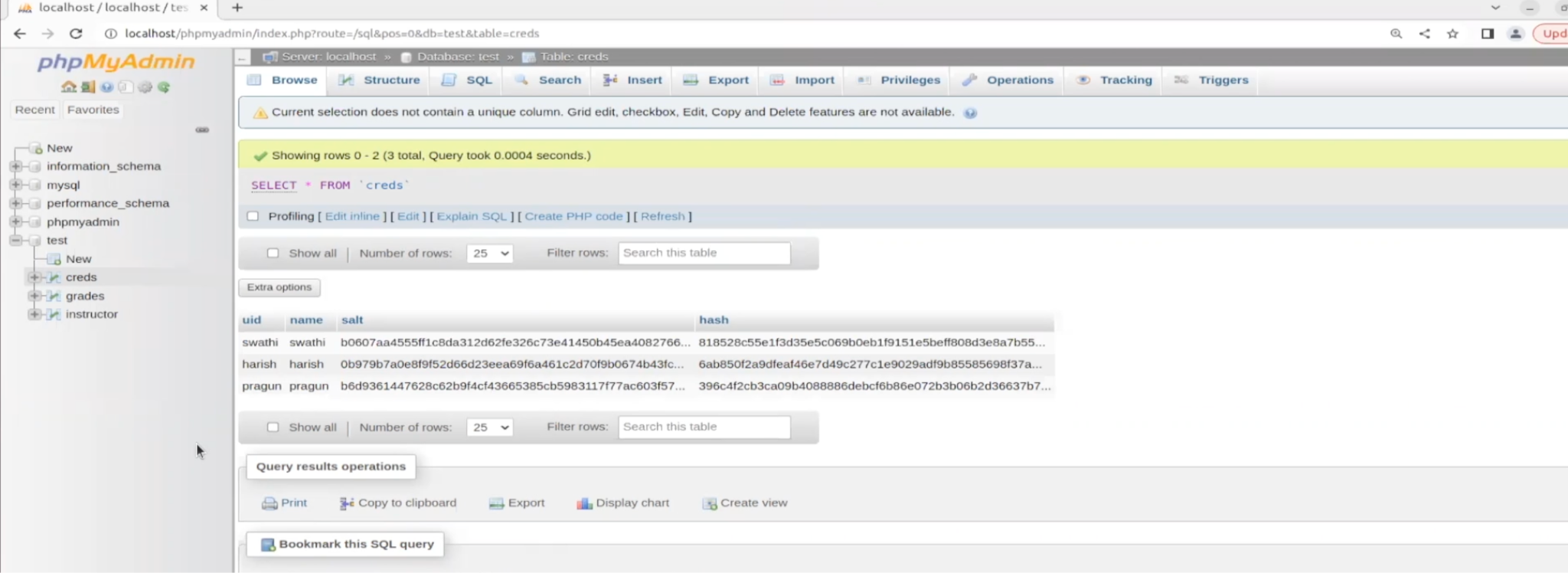


Fig3: Screenshot of Cred table from the database

Grades: This table mainly consists of 5 fields.

* User id:This field stores the user id of the student.
* Grades: This field contains the grades of the student.
* Course: This field specifies the course for the grade.
* Identifier: This field indicates the instructor who provides the grade.
* Transaction id: This field holds the value of transaction id in the block chain.

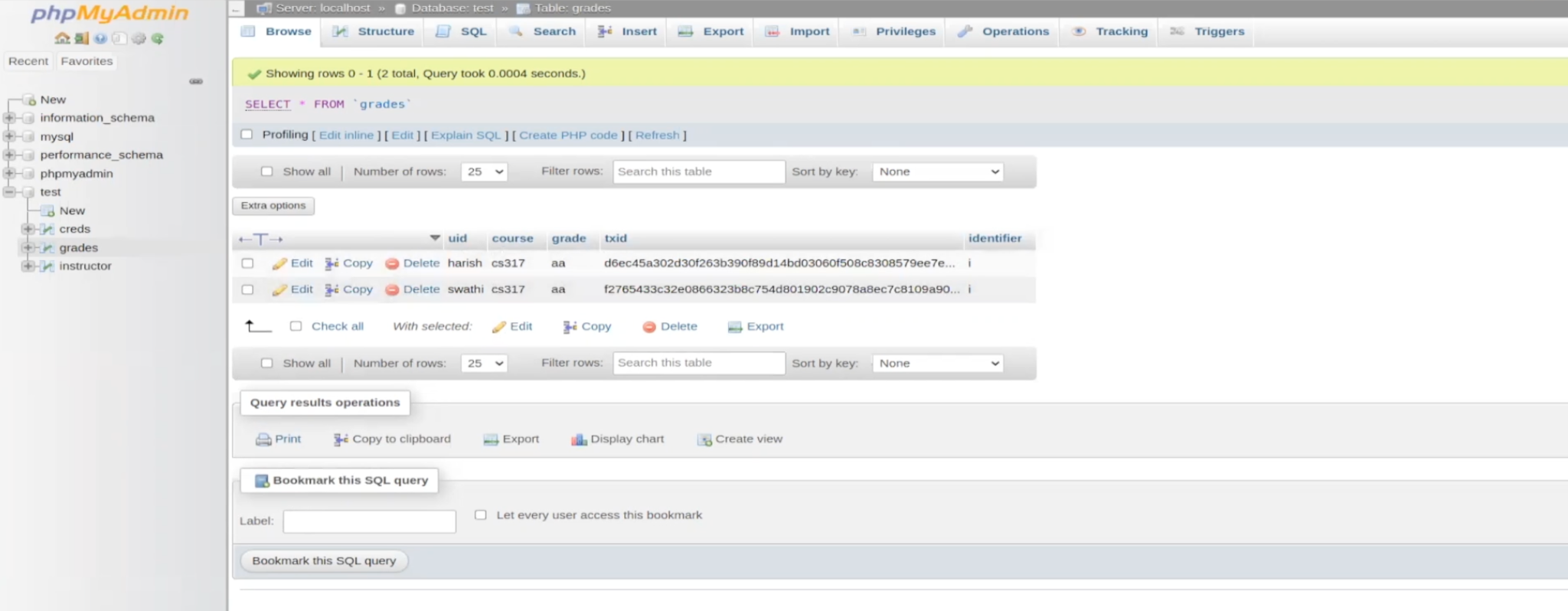


Fig4: Screenshot of grades table from the database

Instructor: This table contains the information of the instructor. The name of the instructor, salt and hash values of their credentials.

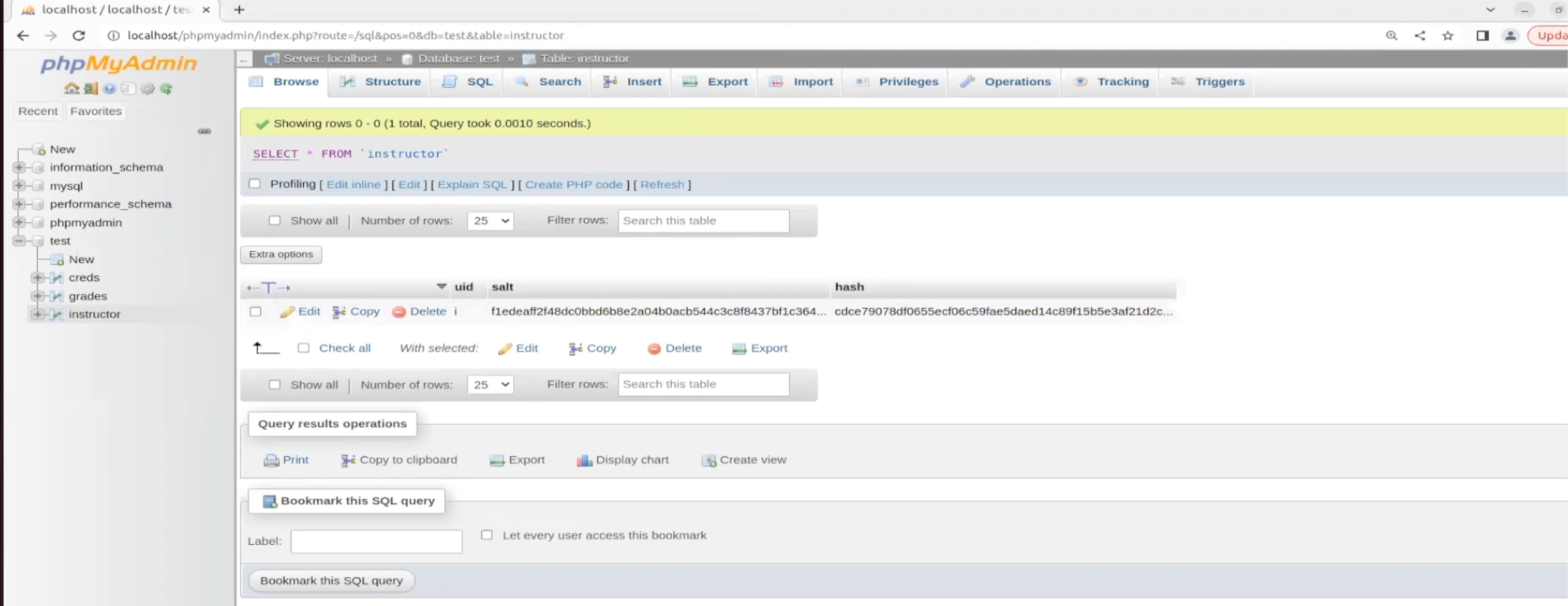


Fig5: Screenshot of instructor table from the database

## Multichain Stream Functionality

Stream functionality is provided by the Multichain network, which allows for data insertion and retrieval on the blockchain network. Streams are a natural abstraction for blockchain use cases that focus on general data retrieval, timestamping, and archiving rather than asset transfer. In a MultiChain blockchain, any number of streams can be created, and each stream acts as an independent append-only collection of items.

Streams can be used to implement three different types of databases on a chain:

* A key-value database or document store
* A time series database that focuses on the ordering of entries
* An identity-driven database where entries are classified based on their author

Here, three streams are created: stream1 for storing database state(transaction id), pubkey stream for storing public key and identifier (instructorID) pairings, and a third for storing instructor-course pairings.

# **Methods**

## **Signup And Login**

This application is built with a Python Tkinter-based GUI. Instructors can sign up for the application and then log in. An RSA key pair is generated when they sign up. The Private Key is encrypted with the passcode entered by the instructor and saved on the person's system. The Public Key is transmitted to the server and broadcasted

## **Inserting Query**

An instructor can enter his students' grades in the courses he teaches. The courses taught by a specific instructor are not stored in the database, but rather appear as a transaction on an instructor stream.

An instructor enters a newline between each space-separated string of a student's userID, course, and grade. When a batch of grades is submitted, each line of grade is digitally signed with the instructor's PrivateKey and sent to the server. The server retrieves the instructor's public key from the public key stream and verifies the signature.

The database is retrieved (ordered by txID) for each grade about to be entered, and each tuple (txID+uid+course+grade+identifier) is concatenated and a hash is obtained.

The current grade is concatenated to this hash (uid+course+grade+identifier) and hashed again. This final hash is then sent to the stream, resulting in a transaction for that grade. It should be noted that the actual implementation is optimized to only query the database once for each group of inserts.

## **Updating Query**

Additionally, instructors can change a student's grade. Updates, unlike Inserts, occur one at a time. When an update is released, the entire database is retrieved and the old grade tuple is deleted. It is now hashed, the new data is concatenated and hashed again (similar to the insert query above), and the transaction is sent to the stream. The old grade and transactionID are then replaced with the new grade and transactionID.

# **Detecting an Insider Attack**

A check is performed before each query (SELECT, INSERT, UPDATE) issued to the database to see if the database is consistent with what is present in the Multichain stream.

To obtain the most recent transaction, the stream is queried. The transactionID and corresponding data (a hash) are retrieved. The entire database is now queried, and all grades except the one corresponding to the most recent transactionID are concatenated. A hash is calculated, then the resultant string is concatenated with this leftover grade (uid+course+grade+identifier) and a hash is calculated over the resultant string. If the hash obtained matches the one stored in the stream corresponding to the most recent transaction, the database is consistent; otherwise, it has been accessed illegally, indicating insider access.

A notification in the grading application is displayed that there is a breach detected.

# **Security checks**

1. A status bar at the top of the window informs you about the current process.
2. When a batch of grade inserts fails, the user is notified of the grades that did not insert (either due to primary key violation or incorrect format of data)
3. An instructor can change (add or update) grades only for the courses he teaches (of course!) and not for others.
4. A student is only permitted to view his or her own grade. He is not authorized to take any other actions.

# **Possible inclusions**

* Separate streams for each course: This is a fairly natural abstraction because each stream acts as an independent append-only collection of items.
* Instead of hashes, a separate stream is used for database statements: This allows for quick recovery in the event that the transaction was pushed to the stream but not to the database.
* Merkle Trees: Merkle trees are widely used in Blockchain. It is a structure that enables efficient and secure content verification in large data sets. They are made by repeatedly hashing pairs of nodes until only one hash remains, known as the Root of the Merkle.

# **Results and Discussion:**

1. Instructor login:

As the instructor logs in, he has the option to either view the grades or enters the grades.

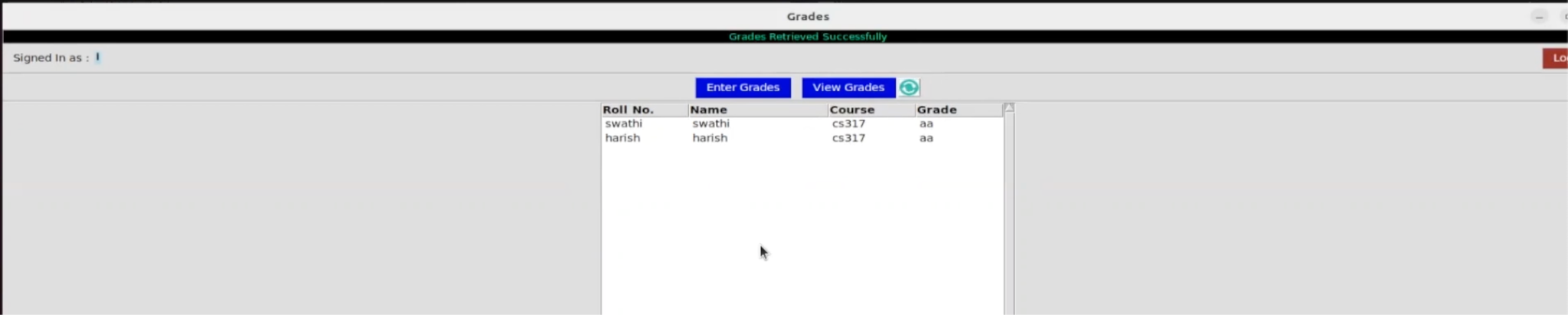


Fig6: Screenshot of grades entered by instructor once logged in

1. Student login:

Students can login and view their grades.

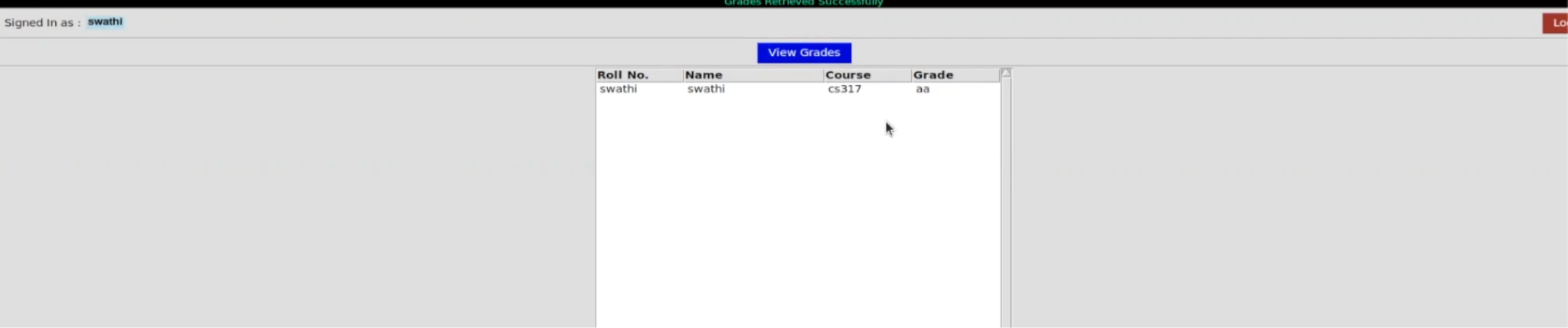


Fig3: Screenshot of grades viewed by the student once logged in

1. Detecting insider attacks.

When there is an illegal attempt to modify/change the data, a breach is detected.

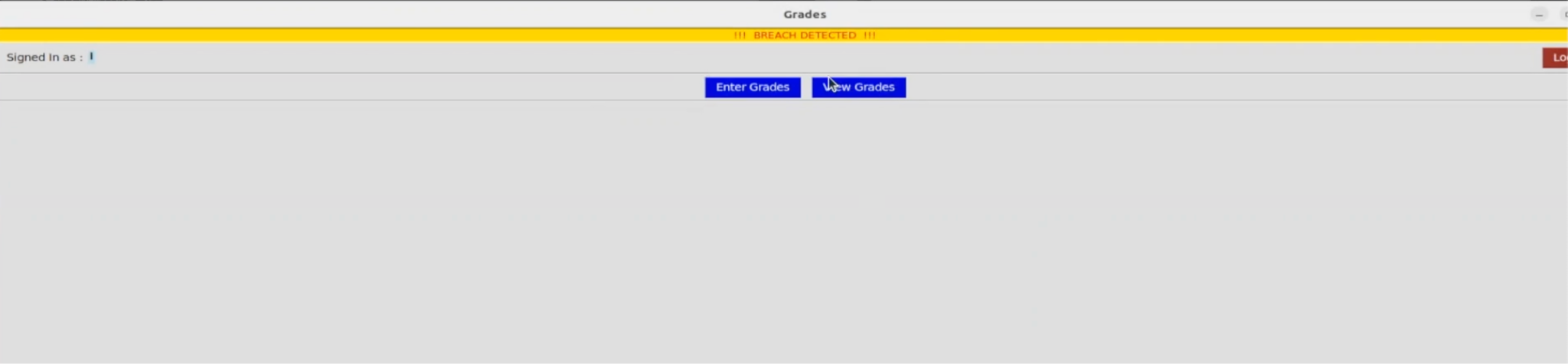


Fig3: Screenshot of Breach Detected notification when an insider attack occurs

# 

# **CONTRIBUTIONS**

* Coding for functions:

Harish:

* PHP scripts: Signup.php, Update.php
* Research about multichain, setting up multi chain environment
* Python based application tool BCD.py
* Documentation, presentation

Swathi:

* PHP scripts: Login.php, Logout.php
* Research about multichain, setting up multi chain environment
* Documentation, presentation

Pragun:

* Insert.php, View.php

Potential Roadblocks during course of the project:

**Conclusions**

With the wide application of blockchain, it is very important to understand the same and minimize the threats. The key takeaway from this project was a rudimentary understanding of blockchain networks, The concepts of streams in blockchain, Importance of insider attack detection and our solution to that using multichain.

# Appendix

**[Project Link](https://github.com/harish-udhay/Multichain-blockchain/tree/main)**

# Reference

[1] Shubham Sharma, Rahul Gupta, Shubham Sahai Srivastava and Sandeep K. Shukla,​ Detecting Insider Attacks on Databases using Blockchains

[2] Hu T, Xin B, Liu X, Chen T, Ding K, Zhang X. Tracking the Insider Attacker: A Blockchain Traceability System for Insider Threats. Sensors (Basel). 2020 Sep 16;20(18):5297. doi: 10.3390/s20185297. PMID: 32947915; PMCID: PMC7570583.

[3] Dr Gideon Greenspan, Founder and CEO, Coin Sciences Ltd, ​MultiChain Private Blockchain-WhitePaper,url:h​ttps://[www.multichain.com/](http://www.multichain.com/)

[4]<https://www.investopedia.com/terms/b/blockchain.asp>