**A BLOCKCHAIN-BASED CREDIBLE AND SECURE**

**EDUCATION DATA MANAGEMENT USING PBFT ALGORITHM**

### A PROJECT REPORT

***Submitted by***

## VALARMATHY.V [211419104295]

## SURUTHI.S [211419104277]

## REVATHI.S [211419104221]

***in partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE**

## (An Autonomous Institution, Affiliated to Anna University, Chennai)

## APRIL 2023

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report **“A Blockchain-Based Credible and Secure Education data Management using PBFT Algorithm”** is the bonafide work of “**VALARMATHY.V (211419104295), SURUTHI.S (211419104277), REVATHI.S(211419104221)”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.L.JABASHEELA,M.E.,Ph.D**  **HEAD OF THE DEPARTMENT,**  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING  COLLEGE,  NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123 | **SIGNATURE**  **Dr.A.HEMLATHADHEVI,M.E.,Ph.,**  **ASSOCIATE PROFESSOR,**  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE,  NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123. |

Certified that the above candidate(s) was/ were examined in the End Semester Project

Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## DECLARATION BY THE STUDENT

We VALARMATHY.V (211419104295), SURUTHI.S (211419104277), REVATHI.S (211419104221) hereby declare that this project report titled “**A Blockchain - Based Credible and Secure Education data Management using PBFT Algorithm**”, under the guidance of, Dr. A. Hemlathadhevi M.E., (Ph.D.) is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

### VALARMATHY.V

1. **SURUTHI.S**

### REVATHI.S

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR, M.E., Ph.D.** and **Dr.SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.,** for providing

us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr.L.JABASHEELA , M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide**, Dr.A.Hemlathadhevi, M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**VALRMATHY.V SURUTHI.S REVATHI.S**

# ABSTRACT

The secured data management systems in the education sector cannot be over emphasized. The storage and sharing model of student education records data still faces many challenges in terms of privacy protection and efficient transmission. The use of block chain technology has been proposed as solution to address the challenges faced in the current centralized education sector.The Practical Byzantine Fault Tolerance (PBFT) consensus algorithm to ensure credible and secure data management. The need for precise privacy data is achieved by constructing a dictionary. Cryptographic techniques such as AES is used for encrypted storage of data and keywords. The random secret key is generated for each record through hashing technique for data security storage in block chain . The storage server store database management system with block size 256 bits using SHA-256. Smart contract provides protection for data keywords, the storage server stores data after data masking. Security analysis, privacy protection and computational cost shows that high efficiency and low cost can be achieved. Meanwhile, this

scheme has better robustness compared to other educational records data sharing models.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | vii |
|  | **LIST OF ABBREVIATIONS** | viii |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 3 |
| **2.** | **LITERATURE SURVEY** | 4 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing System | 9 |
|  | 3.2 Proposed system | 10 |
|  | 3.3 Feasibility Study | 11 |
|  | 3.4 Hardware Environment | 13 |
|  | 3.5 Software Environment | 13 |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1 ER diagram | 14 |
|  | 4.2 Data Flow Diagram | 15 |
|  | 4.3 UML Diagrams | 16 |
|  | 4.4 Activity Diagram | 17 |

|  |  |
| --- | --- |
| **5.** | **SYSTEM ARCHITECTURE PAGE NO.** |
|  | 5.1 Module Design Specification 18 |
|  | 5.2 Algorithms 19 |
| **6.** | **SYSTEM IMPLEMENTATION** |
|  | 6.1 Client-side coding 31 |
|  | 6.2 Server-side coding 59 |
| **7.** | **SYSTEM TESTING** |
|  | 7.1 Unit Testing 76 |
|  | 7.2 Integration Testing 79 |
| **8.** | **CONCLUSION** |
|  | 8.1 Conclusion 80  8.2 Future Enhancements 80 |
|  | **APPENDICES** |
|  | A.1 Sample Screens 81 |
|  | A.2 Publications 84 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Title** | **Page No.** |
| 4.1 | ER Diagram | 10 |
| 4.2 | DFD LEVEL 0 | 11 |
| 4.3 | DFD LEVEL 1 | 12 |
| 4.4  4.5  4.6  5.1  5.2  5.3  5.4  5.5  A.1  A.2  A.3    A.4  A.5 | DFD LEVEL 2  USE CASE  ACTIVITY DIAGRAM  SYSTEM ARCHITECTURE  AES ALGORITHM WORKING  STEPS IN AES  PBFT ALGORITHM  PBFT ALGORITHM GRAPH  USER LOGIN  REQUEST ACADEMIC DATA  STAFF PROVIDE ACCESS& KEY  SENT TO USER  ENTER STAFF APPROVED KEY  ACCESS TO ACADEMIC DATA | 13  16  17  19  20  21  30  31  81  82  82    83  83 |

**LIST OF ABBREVIATION**

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| AES | Advanced Encryption Standard |
| PBFT | Practical Byzantine Fault Tolerance |
| KLOC | Thousand Lines Of Code |
| ER | Entity-Relationship |
| DFD | Data Flow Diagram |
| UML | Unified Modeling Language |
| S-BOX  CD | Substitution Box  Encrypted Data |

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Educational records describe their education-specific processes. These records are of great importance for the further study and career of an individual. The essential attribute of records is primitive, which enables records to restore the actual historical situation, so these records are essential to the students themselves, education institutions and potential employers. With the development of information technology, educational records have been digitized. Compared with the traditional paper physical records, digital records are stored on the storage medium which has a high degree of variability, hence such records could be easily modified during the processes of storage, transmission, and sharing.Centralized storage and management mode is usually adopted,which makes systems that use this mode vulnerable to various attacks. Besides the records of different. Educational stages are stored in separate storage servers of education institutions and these storage servers are usually designed to allow access only by internal staff,without any form of interoperability. Moreover, a server failure could easily cause a data loss or leakage. Therefore, in order to protect personal information, institutions usually adopt security policies to restrict the access and sharing of records. However, there is a lack of secure and effective record sharing mechanisms among institutions. Students may experience difficulties when they transfer from one institution to another, while still preserving their completeness of courses from the previous institution. In cases when students apply for postgraduate programs in another education institution, this process requires the provision of past educational records, such as, courses grades, award certificates and so on, by which time they have completed the previous stage of education, left the previous education institution, etc., but usually, such students have no access to the online educational records system. In this case, they have to visit the previous institution to request a paper copy of their educational records. The data owner encrypts the education records data and stores them with cloud storage, and stores the index and record summary of the education records data into the block chain. For the remote server, we determine that it is malicious, so we firstly use data masking for the part of the student’s private data and then encrypt it and store it on the cloud server. The user must have the authorization of the data owner to query the data, and the verification of the user’s authority is realized using a smart contract. the keyword dictionary sent by the data owner, the smart contract can perform the query record, can be published in the block chain.

### 1.2 PROBLEM DEFINITION

Educational records which is essential for the individual student, schools and the potential employer. However, schools have their own independent storage platforms and databases, resulting in data “islands” problems that make it difficult to share data between educational institutions. This leads to students who need to print out paper documents of data stored in school databases when transferring or entering a higher school. The paper documents are easily damaged and lost, causing a lot of inconvenience.by using this concept we can overcome this problem.

**CHAPTER 2**

**LITERATURE SURVEY**

**PAPER 1:**

**TITLE :** Counterfeit Detection of Documents using Blockchain

**AUTHOR:** Rajesha, Rohini Krishna Mohite, Sahana, Shilpashree, Rakesh K R

**YEAR:** 2021

**ABSTRACT**: Document verification is a domain that involves various challenges and monotonous processes to authenticate the documents. The verification for each type of document is to be processed in distinct ways. The issuing of documents is not transparent and hence fake documents can be created. Documents or certificates generated by organizations play a vital role for students. With the increase of forged documents, credibility of both the students and the organization is imperilled. Verifying a document takes some time and also requires more human resources to request for the confirmation of the data provided. Managing documents or records often comes at the cost loss of data or document counterfeits. In this paper, we aim to enhance the document verification process using block chain. We propose a system which uses the Interplanetary File System [IPFS] protocol and also a peer-to-peer-network for storing and sharing data in a distributed file system. By using this block chain technology, we can provide a more secure and efficient digital certificate validation.

**PAPER 2:**

**TITLE :** Education Degree Fraud Detection and Student Certificate Verification using Blockchain

**AUTHOR:** Jayesh G. Dongre, Dr.Kishore.T.Patil, Sonali M. Tikam, Vasudha B. Ghara

**YEAR:** 2020

**ABSTRACT**:

To verify the authenticity of an academic degree and certificates we propose a system which employs a digital signature scheme and timestamps using blockchain technology. As the number of universities and tertiary education students, the number of graduates is constantly increasing. Due to this verification process of these degree certificates generates a lot of new job opportunities. The sudden changes in the technology and development of new technologies like blockchain is booming, the implementation of blockchain using blockcerts software provides us a solution of plausible business models. In this paper we showcase two financial models balancing where the service rates is been balanced between graduates and employer as to main stakeholders of that service. A proof check of certificates for students is done at low cost and an easy check of the authenticity of the certificate is done from and trustable source while recruiting by the employer.

**PAPER 3:**

**TITLE :** Development and Evaluation of Blockchain based Secure Application for Verification and Validation of Academic Certificates

**AUTHOR:** Elva Leka and Besnik Selimi

**YEAR:** 2021

**ABSTRACT**:

Academic degrees are subject to corruptions, system flaws, forgeries, and imitations. In this paper we propose to develop a blockchain smart contract-based application using Ethereum Platform, to store, distribute and verify academic certificates. It constitutes a trusted, decentralized certificate’s management system that can offer a unified viewpoint for students, academic institutions, as well as for other potential stakeholders such as employers. The article describes the implementation of three main parts of our proposed solution that includes: verification application, university interface and accreditor interface. This application avoids administrative barriers, makes the process of deployment, verification, and validation of certificates faster, efficient, and more secure. Additionally, it offers confidentiality of the data by using AES encryption algorithm before creating transactions and allows bulk submission of multiple academic certificates.

**PAPER 4:**

**TITLE :** Blockchain-based model to track and verify official certificates.

**AUTHOR:** Pooja Mara, Ravi kanth Motupalli

**YEAR:** 2022

**ABSTRACT**:

Document verification is a complicated domain with a variety of difficult and time-consuming methods to validate. Customized verification and authentication processes may be required for various sorts of papers, such as financial papers, government papers, transaction papers, educational credentials, and so on. A huge problem today we are facing is the number of fake certificates that are in circulation, this problem is quite predominant. This has become a new business for a long time. Hardworking people with genuine degrees/certificates have been suffering and they get rejected in the job market because of the lack of identification to differentiate between the original and the fake certificate. At times people are getting jobs through fake certificates and this becomes very dangerous. The scenario calls for a new system that can verify and authenticate certificates, their issuers, and their holders in a way that is much more efficient, simple, and intuitive to use, and efficiently mitigates widespread credential fraud. To combat the counterfeiting of academic certificates, our block chain approach combines a verified distributed ledger with a cryptographic mechanism. Our Block chain technology will also provide a standard sharing platform for storing and accessing documents, reducing overall verification time and allowing companies to quickly monitor and access real papers. Our system that helps to identify original certificates will help the hard-earned people to get their jobs in their desired organizations and all the fake circulation of certificates will be stopped because the system identifies the genuine true certificates.

**PAPER 5:**

**TITLE :** Revolutionizing Verification and Management of Educational Certificates with Self-Sovereign Student Identities using Blockchain

**AUTHOR:** Harshita Bhosale1, Rutuja Kanki, Gayatri Jaiswal

**YEAR:** 2021

**ABSTRACT**:

Educational Institutions have come a long way in transforming education systems, but they still require a better and fraud-proctored system to address the issues that exist even today. The need of a single secure platform for all educational stakeholders such as, e-learning platforms, academic institutes, universities and students to avoid re-verification and maintaining immutable record of a student's digital assets are a driving fuel to significantly transform current system. The main objective of the work is to highlight the existing issues of fraudulent degrees, redundancy in verification process of documents, lack of validation for authenticity of certificates in the current education sector, lack of single authorised identity for students and resolve themusing decentralization, immutability, traceability, consensus mechanism and other features of blockchain. The existence of third parties between Universities, institutions and students gets eliminated by the distributed nature that blockchain offers. The consensus mechanism employed will make sure that only authenticated data is put on chain, quelling the fraud certificates that often end up getting amassed at the employer's desk. The intent is to design a prototype to test the applicability of blockchain in solving above stated issues.

**CHAPTER 3**

**SYSTEM ANALYSIS**

### 3.1 EXISTING SYSTEM

Centralized storage and management mode is usually adopted, which makes systems that use this mode vulnerable to various attacks. Besides, the records of different educational stages are stored in separate storage servers of education institutions and these storage servers are usually designed to allow access only by internal staff, without any form of interoperability. The student’s educational records which is essential for the individual student, schools and the potential employer. a server failure could easily cause a data loss or leakage. Therefore, in order to protect personal information, institutions usually adopt security policies to restrict the access and sharing of records. This leads to students who need to print out paper documents of data stored in school databases when transferring or entering a higher school. The paper documents are easily damaged and lost, causing a lot of inconvenience.

**3.1.1 DISADVANTAGES:**

* Paper documents are easily damaged and lost
* Data leakage and illegal tampering
* Lack of efficiency and failure to guarantee data security and credibility

**3.2 PROPOSED SYSTEM**

A novel scheme is proposed, which integrates educational records storage and sharing among education institutions enabled by blockchain, storage servers and smart contracts. The blockchain is responsible for ensuring the security and auditability of the data, the smart contract is used to define the permissions of the records and to regulate the behaviours of the member nodes. We remark that public blockchain is not suited in this case, because educational records are related to personal privacy and contain sensitive information, such as family address, age, contact details, etc. Moreover, even if the institutions put encrypted data on the public blockchain, it still will expose their operation situations and statistical data. We firstly use data masking for the part of the student’s private data and then encrypt it and store it on the cloud server. The user must have the authorization of the data owner to query the data, and the verification of the user’s authority is realized using a smart contract. Students can take their documents using key from the cloud.

## 3.2.1 ADVANTAGES:

* Lower energy consumption and faster transaction speed
* No chance of document leaks.
* enable secure sharing of student data
* Efficient data storage

**PROPOSED ALGORITHM**

1. **studLoginId=XYZ**
2. **if(StudEnteredID== studLoginId)**

**Login to DB;**

**Gives request to view data;**

**if(request accepted && key sent to student)**

**Enters key;**

**View Academic Data;**

**else if(request not accepted && key not sent)**

**No access to data**

**else**

**Login access not provided;**

* **There is Secret key to protect the academic data of students in our proposed system.**

**3.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

### 3.3.1 ECONOMIC FEASIBILITY

### This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 3.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.3.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## 3.4 HARDWARE ENVIRONMENT

* Processor - Intel i3 ,i5 ,i7, AMD Processor
* RAM - above 4 Gb
* Hard Disk - 500 GB

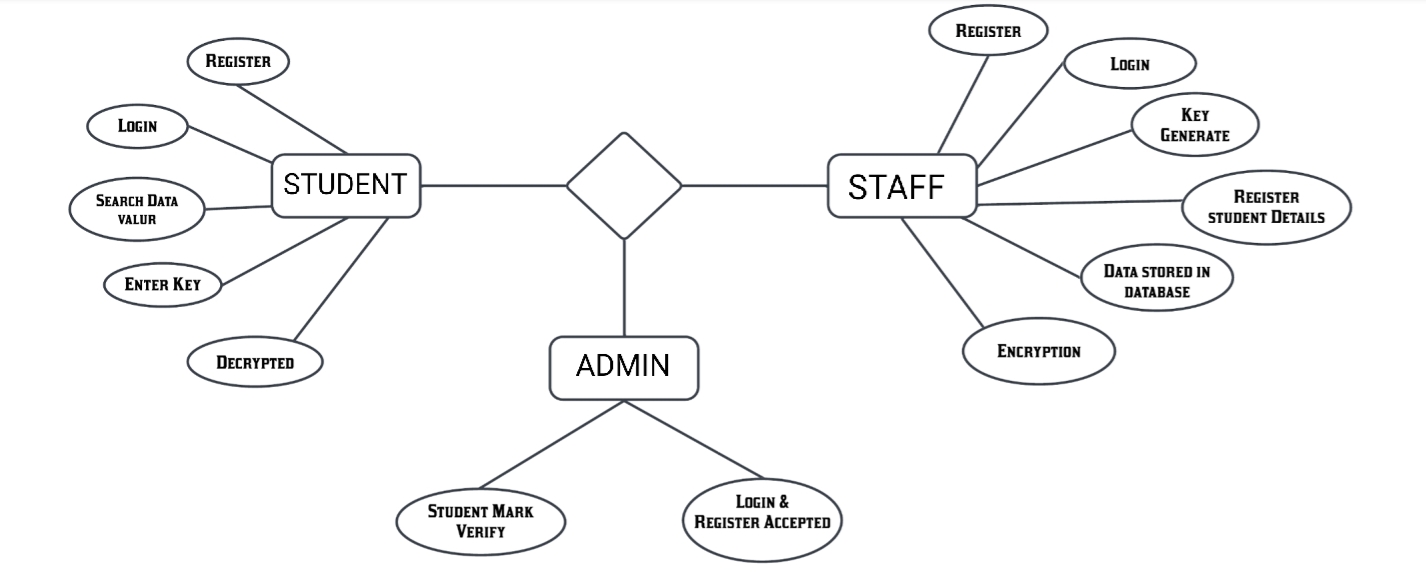
**3.5 SOFTWARE ENVIRONMENT**

* Operating System - Windows 7/8/10
* Front End - GUI
* Tool - Python

**CHAPTER 4**

**SYSTEM DESIGN**

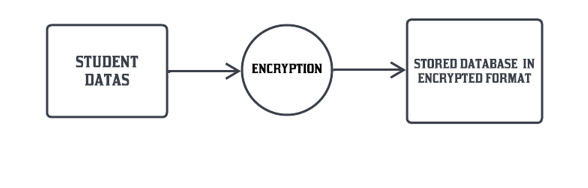
**4.1 ER DIAGRAM:**



**Fig no:4.1**

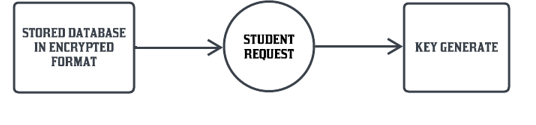
**4.2 DATA FLOW DIAGRAM**

**DFD DIAGRAM:**

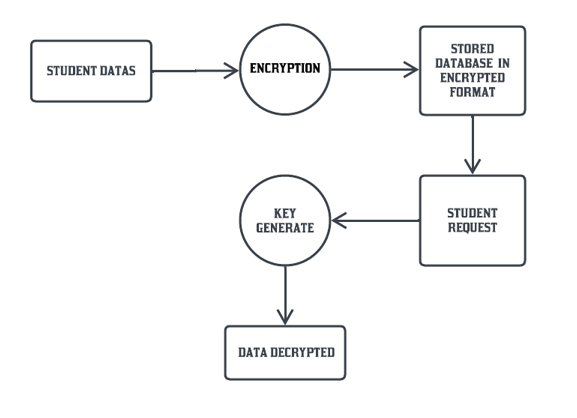
**DFD LEVEL 0:**

**Fig no:4.2**

**DFD LEVEL 1:**

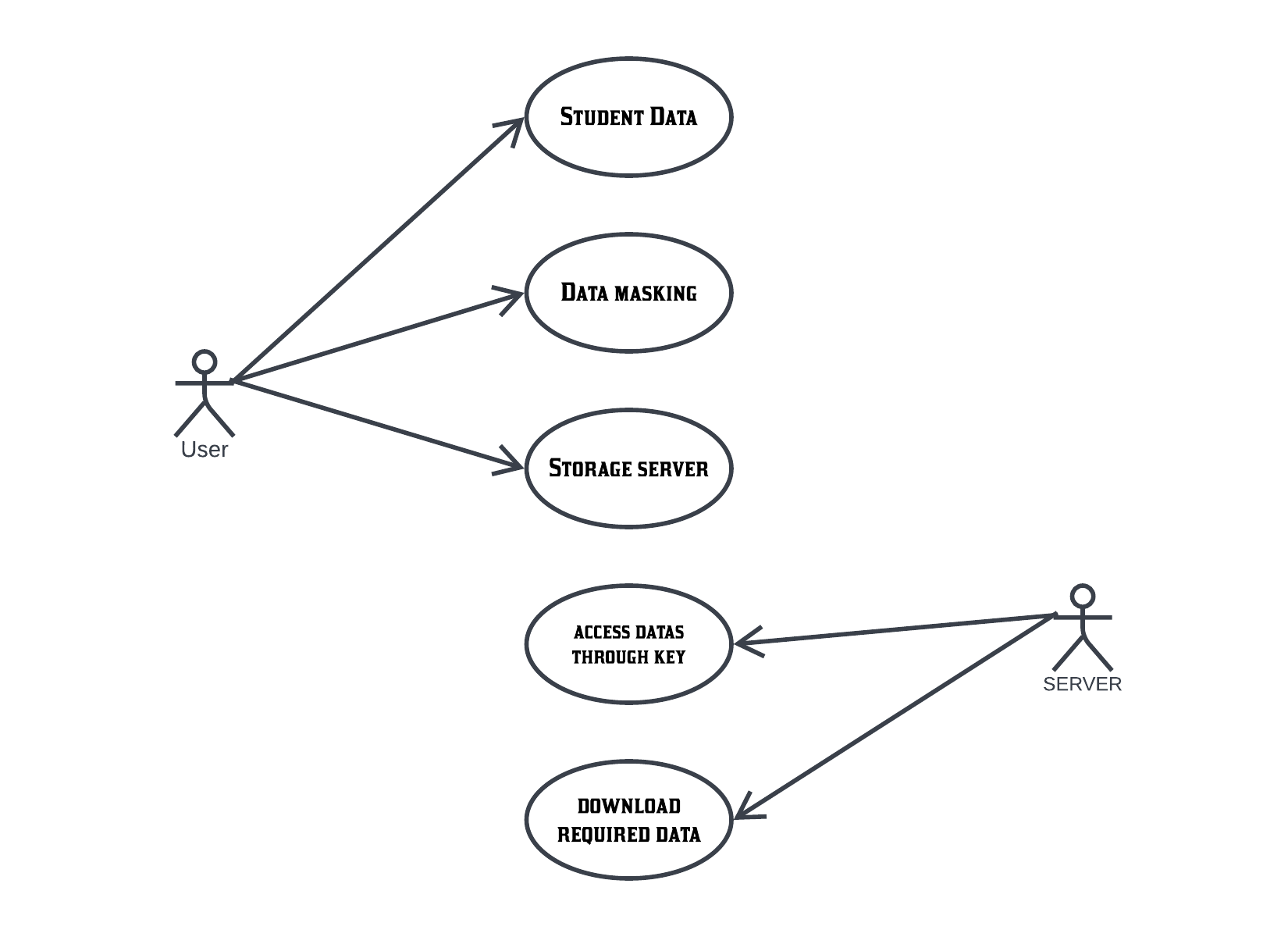
****

**Fig no:4.3**

** OVERALL DFD:**

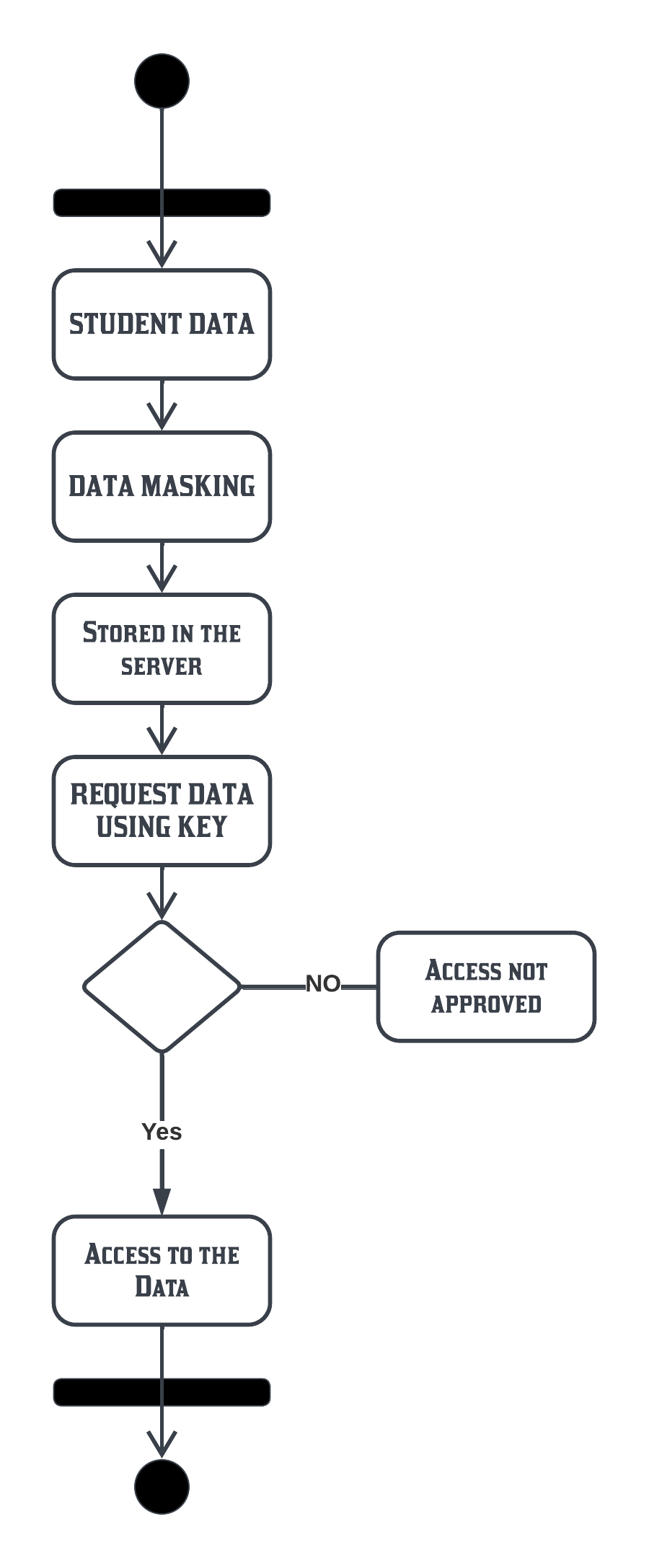
**Fig no:4.4**

**4.3 USE CASE**



**Fig no:4.5**

**4.4 ACTIVITY DIAGRAM**

****

**Fig no:4.6**

**CHAPTER 5**

**SYSTEM ARTCHITECTURE**

**5.1 MODULE DESIGN SPECIFICATION**

**MODULES NAME:**

* User Search Process
* Data request
* Encrypted data strong
* Data access by entering data
* Consensus Phase
* Secure storage of Educational data

**User Search Process**

Generate user information Info, use private key Sk(U) to encrypt information CT(U) = Encrypt(Sk(U), Info). Write multiple search keywords into Q,

run CQ = Encrypt (Sk(U), Q) obtain the encrypted keywords CQ and send the request Req = Send(Q, CT(U), P k(U)) to the data owner.

The user obtains the key of the educational institution through Sk = Decrypt(Key, Sk(U)), decrypts the file D = DENC(CD, Sk), and finally obtains the student’s education records.

**DATA REQUEST:**

The Students Data that is entered and encrypted in the Blockchain Database can only be accessed through Secret Key.Once the student sends Data access request, it then comes under staff approval.When the staff gives approval, the respective secret key for that students data is sent to the student.

**ENCRYPTED DATA STORING:**

The encrypted data is stored in the storage server and their **hash** is put on the blockchain and **keyword** also **generated** for the each student for the security. The amount of data on student education records is huge. For the transmission of big data used **Encryption algorithm**. The original records and files are **encrypted and stored** in the storage server. The cloud database is chosen as the storage server to efficiently store and retrieve data and support encrypted storage of files.

**DATA ACCESS BY ENTERING KEY:**

After storing the data into the databases , that is then available in the **server** but the student have to **enter the keyword** for the accessing their data’s. So, after entering the key, the **student get access** to the storage server and able take his documents easily. The blockchain is applied in several domains and acts as a **trusted data storage** technology. This technology is often used for information **secure storage** and information traceability,because of its decentralized and **anti-tampering** characteristics.

**CONSENSUS PHASE:**

After user obtains the student data, the smart contract submits the query record to the verification node, which requires the digital signature of user and educational institution. If the transaction is passed by the verification node through the PBFT consensus algorithm, it will be published and recorded on the blockchain. The record can also be used to prove the authenticity and credibility of the data source when external institutions do related research based on students educational records.

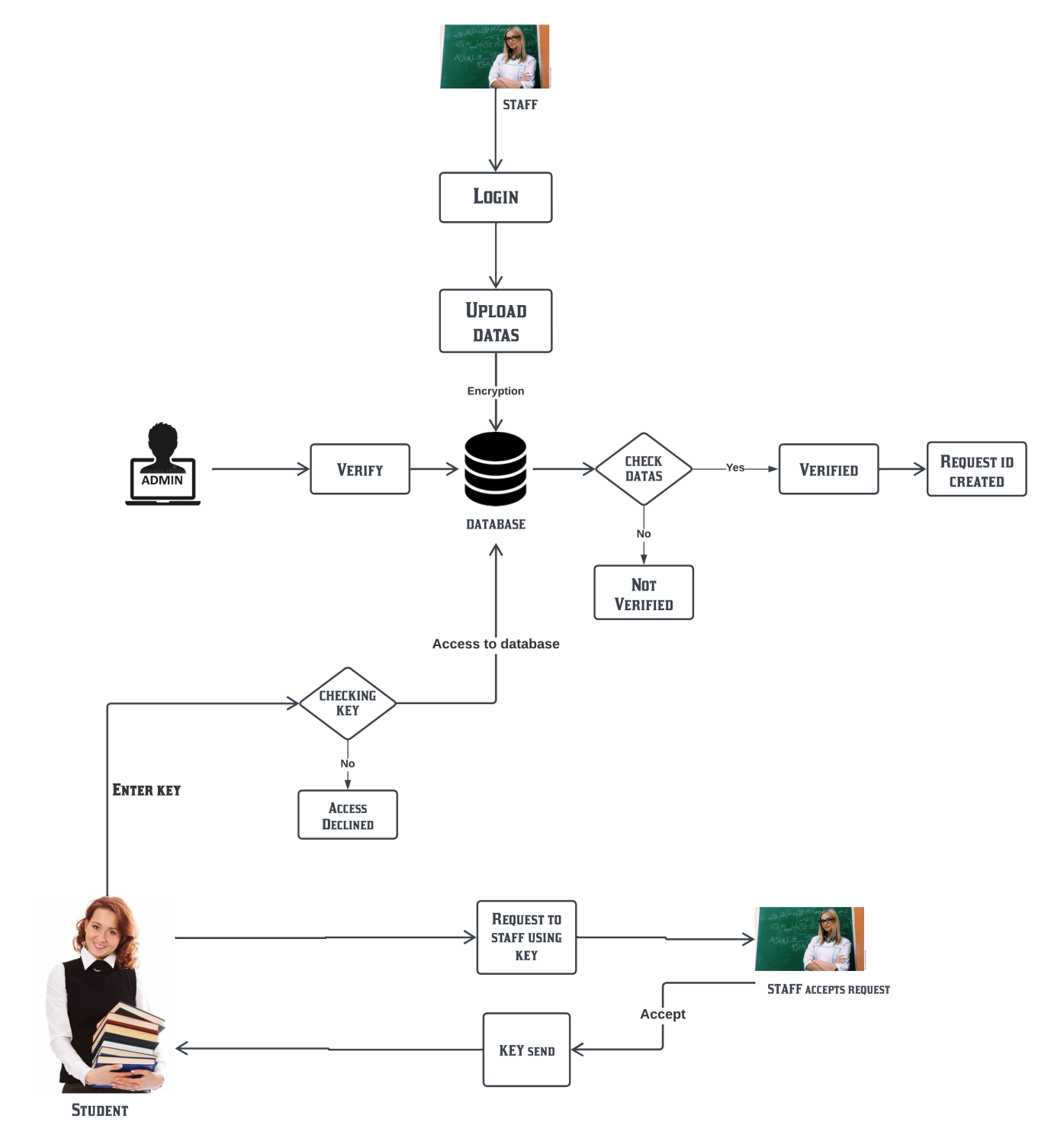
**SECURE STORAGE OF EDUCATIONAL RECORDS:**

Data owner select the private data PD, and extracts keywords W={W1,W2,……Wn}through PD.

The privacy data keywords are stored in the keyword table . The storage address, can be calculated in the following way:

H(CD) = CHash(CD).

**SYSTEM ARCHITECTURE**

****

**Fig no:5.1**

**5.2 ALGORITHMS**

**5.2.1 AES ALGORITHM:**

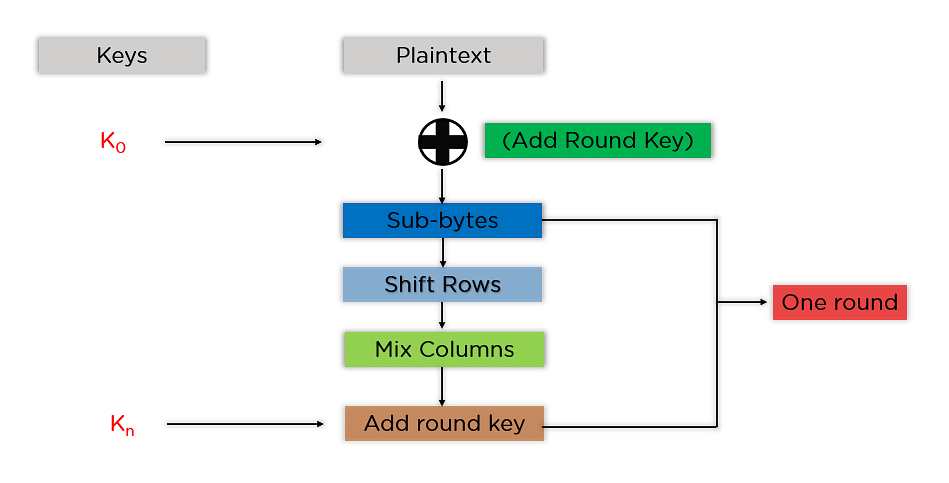
Advanced Encryption Standard is a symmetric block cipher. It converts the individual blocks using different keys. It is one of the best encryption protocols available, letting anyone enjoy their daily online activities without disruption.

Diagram

Description automatically generated

**Fig no:5.2**

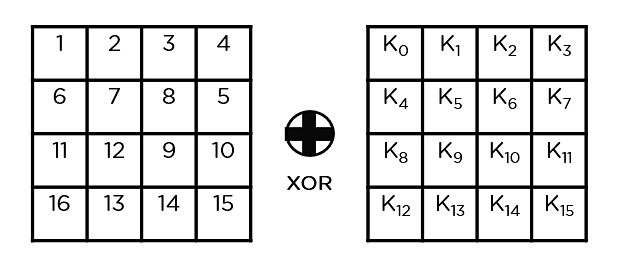
**Steps to be followed in AES:**



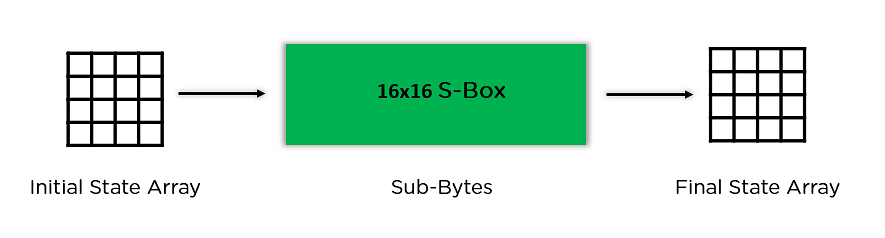
**Fig no:5.3**

The mentioned steps are to be followed for every block sequentially. Upon successfully encrypting the individual blocks, it joins them together to form the final ciphertext. The steps are as follows:

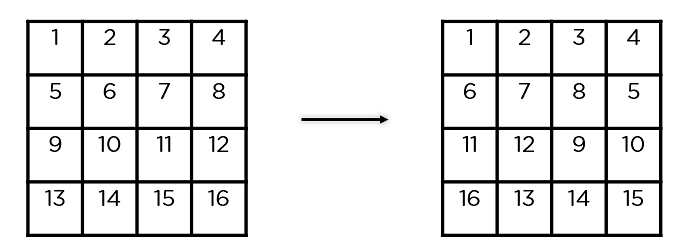
* Add Round Key: You pass the block data stored in the state array through an XOR function with the first key generated (K0). It passes the resultant state array on as input to the next step.



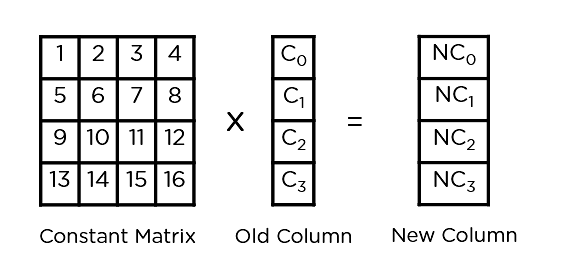
* Sub-Bytes: In this step, it converts each byte of the state array into hexadecimal, divided into two equal parts. These parts are the rows and columns, mapped with a substitution box (S-Box) to generate new values for the final state array.



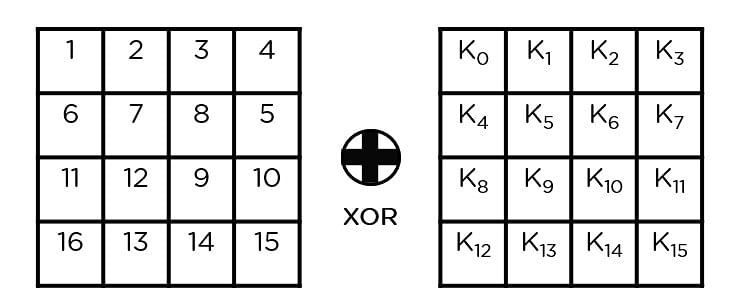
* Shift Rows: It swaps the row elements among each other. It skips the first row. It shifts the elements in the second row, one position to the left. It also shifts the elements from the third row two consecutive positions to the left, and it shifts the last row three positions to the left.



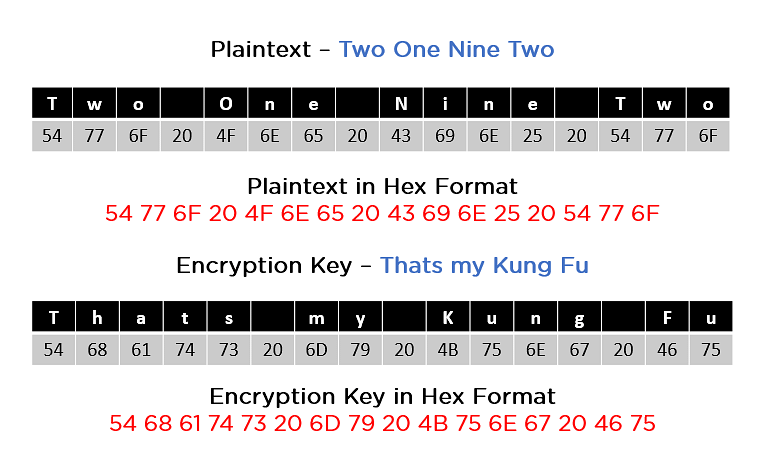
* Mix Columns: It multiplies a constant matrix with each column in the state array to get a new column for the subsequent state array. Once all the columns are multiplied with the same constant matrix, you get your state array for the next step. This particular step is not to be done in the last round.



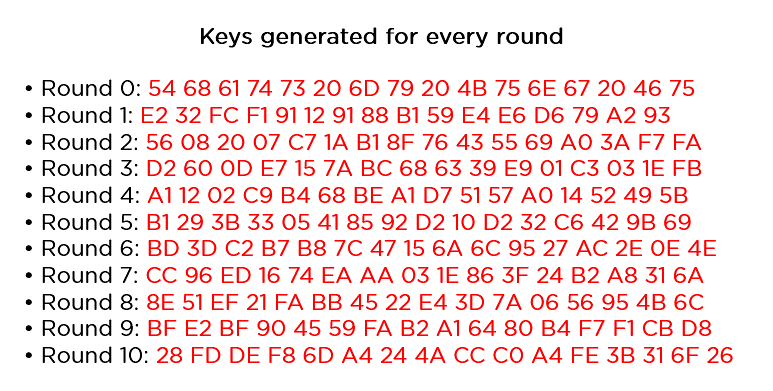
* Add Round Key: The respective key for the round is XOR’d with the state array is obtained in the previous step. If this is the last round, the resultant state array becomes the ciphertext for the specific block; else, it passes as the new state array input for the next round.



Now that you understand the basic steps needed to go through the encryption procedure, understand this example to follow along.

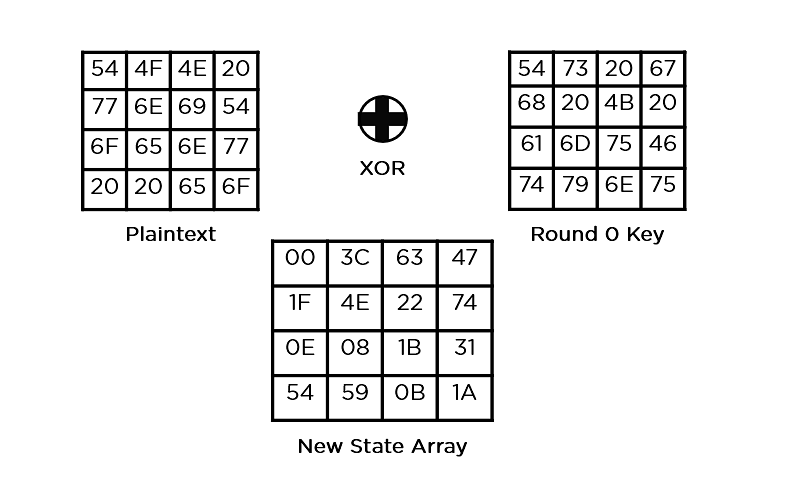


As you can see in the image above, the plaintext and encryption convert keys to hex format before the operations begin. Accordingly, you can generate the keys for the next ten rounds, as you can see below.

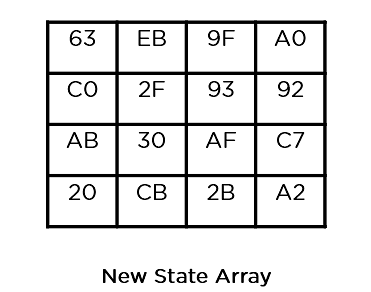


You need to follow the same steps explained above, sequentially extracting the state array and passing it off as input to the next round. The steps are as follows:

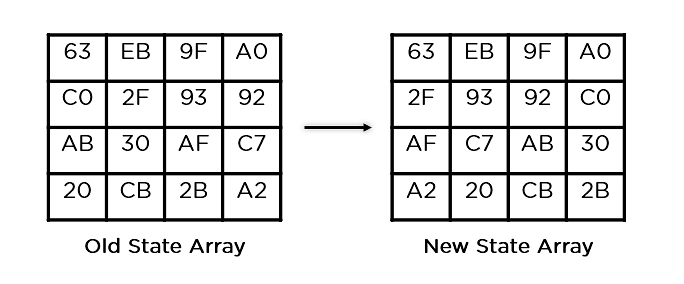
* **Add Round Key:**



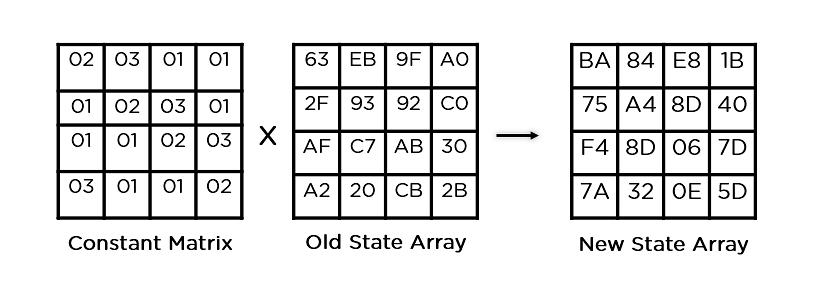
* Sub-Bytes: It passes the elements through a 16x16 S-Box to get a completely new state array.



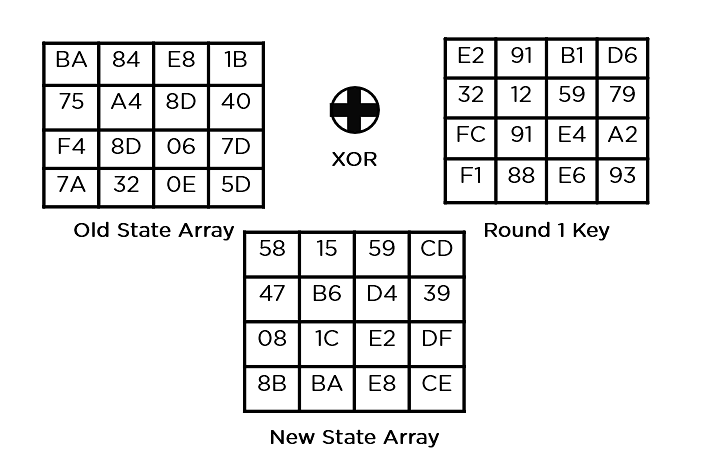
* Shift Rows:



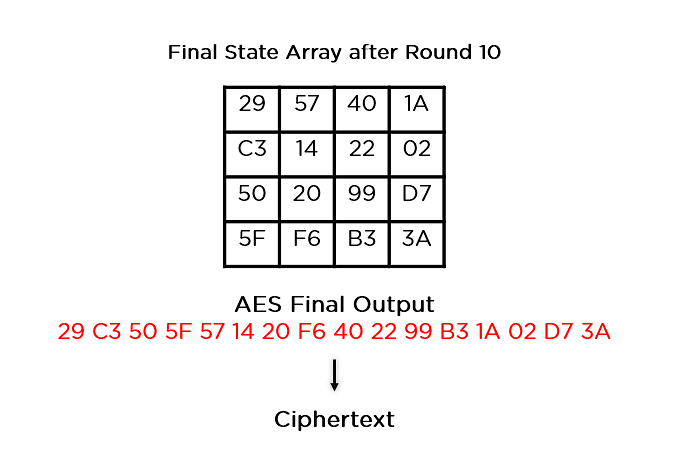
* Mix Columns:



* Add Round Key:



* This state array is now the final ciphertext for this particular round. This becomes the input for the next round. Depending on the key length.
* you repeat the above steps until you complete round 10, after which you receive the final ciphertext.



**5.2.2 PBFT ALGORITHM USED IN REQUEST METHOD**

* PBFT uses cryptographic algorithms such as encryption and hash to ensure that everything stays unforgeable, and indisputable.
* Nodes in a PBFT enabled distributed system are sequentially ordered with one node being the primary (or the leader node) and others referred to as secondary (or the backup nodes).

**5.2.3 PBFT ALGORITHM**

PBFT consensus rounds are broken into 4 phases (refer with the image below):

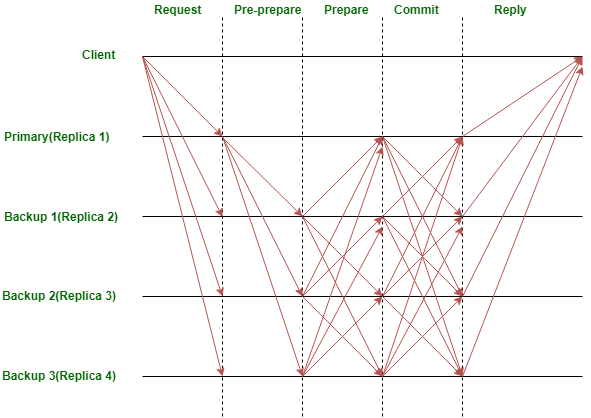
1. The client sends a request to the primary(leader) node.
2. The primary(leader) node broadcasts the request to the all the secondary(backup) nodes.
3. The nodes (primary and secondaries) perform the service requested and then send back a reply

to the client.

1. The request is served successfully when the client receives ‘m+1’ replies from different nodes in

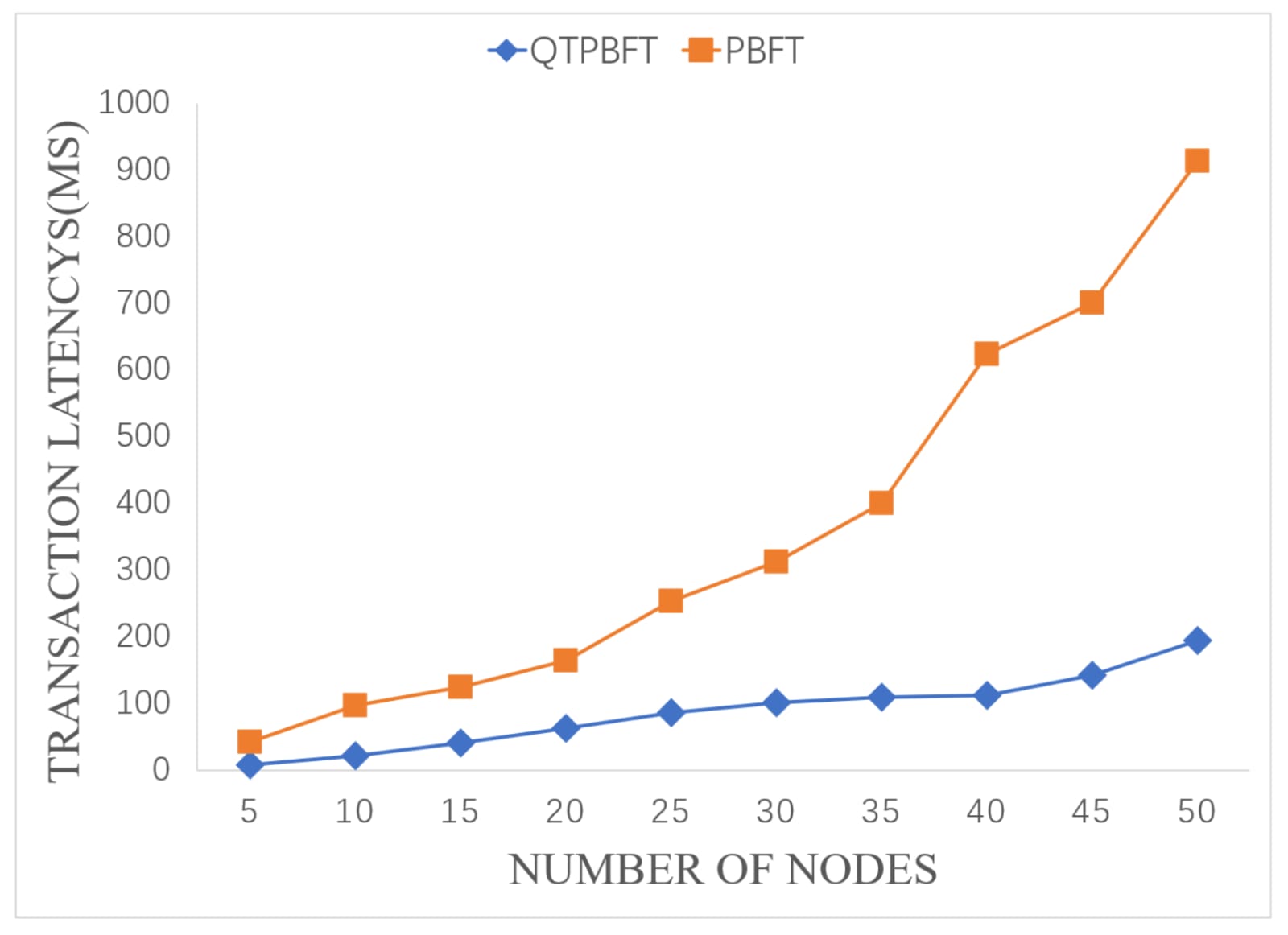
the network with the same result, where m is the maximum number of faulty nodes allowed.

**PBFT ALGORITHM**



**Fig no:5.4**

**PBFT ALGORITHM GRAPH**

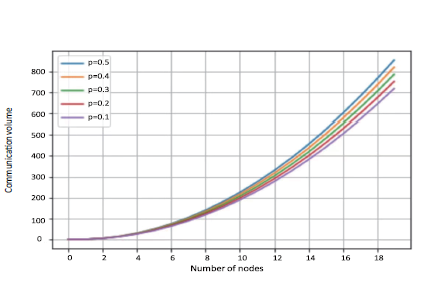
****

**Fig no:5.5**

**PBFT Algorithm-Communication Complexity**

* Assuming the number of nodes of the PBFT algorithm is n,
* The consensus process of PBFT algorithm can be divided into five phases: The request phase(1), pre-prepare phase(n−1), prepare phase (n −1)(n −1), commit phase(n(n −1)), and reply phase(n).
  + - * The communication complexity can be calculated to be 1 + (n −1) + (n −1)(n −1)+n(n −1)+n.
      * The Communication complexity of the PBFT algorithm

C1= 2n 2 − n+1

** The communication volume of the proposed scheme CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 CLIENT-SIDE CODING:**

**user.py**

from tkinter import \*

from tkinter import messagebox,ttk,filedialog

import pymysql

import base64

import hashlib

from Crypto.Cipher import AES

from Crypto import Random

BLOCK\_SIZE = 16

pad = lambda s: bytes(s + (BLOCK\_SIZE - len(s) % BLOCK\_SIZE) \* chr(BLOCK\_SIZE - len(s) % BLOCK\_SIZE),'utf-8')

unpad = lambda s: s[:-ord(s[len(s) - 1:])]

user = Tk()

user.geometry('1920x1080+0+10')

bg = PhotoImage(file='pic1.png')

bgLabel = Label(user,image=bg)

bgLabel.place(x=0, y=0)

con=pymysql.connect(host="localhost",user="root",password="admin", database="studentregister”)

cur = con.cursor()

def user\_login():

login\_bg = '#FFF'

global User\_login\_Frame

User\_login\_Frame = Frame( padx=60, pady=60 , bg=login\_bg )

Label(User\_login\_Frame,text='UserLogin' , font= ('Arial',22,'bold'),bg=login\_bg ).pack(pady=10)

Label(User\_login\_Frame,text='Email',textvariable='email',font=('Arial',14),bg=login\_bg)

.pack(pady=1)

entry01 = ttk.Entry(User\_login\_Frame)

entry01.pack(pady=2)

Label(User\_login\_Frame,text='Password', font= ('Arial',14) , bg=login\_bg ).pack(pady=1)

entry02 = ttk.Entry(User\_login\_Frame,show='\*')

entry02.pack(pady=2)

def login\_Close():

if entry01.get() != '' and entry02.get() != '' :

cur.execute("select\*from user register where Email=%s

and Password=%s",(entry01.get(),entry02.get()))

row = cur.fetchone()

if row != None:

if row[4] == 'Approved':

User\_login\_Frame.destroy()

search\_Bar(row[1])

else:

messagebox.showinfo('Wait','Wait For Admin Approval.')

else:

messagebox.showerror('Failed','Login Failed.')

else:

messagebox.showwarning("Alert","Enter Email & Password Correctly !!")

def open\_Register():

User\_login\_Frame.destroy()

user\_Register()

ttk.Button(User\_login\_Frame,text='Login ✔' ,command=login\_Close).pack(pady=10)

Button(User\_login\_Frame,text='Not Registered ?' , bd= 0 , bg=login\_bg , relief='flat' , overrelief='flat' , command=open\_Register ).pack(pady=10)

User\_login\_Frame.pack()

def user\_Register():

register\_bg = '#FFF'

register\_Frame = Frame( padx=50, pady=20 , bg = register\_bg )

Label(register\_Frame,text='RegisterForUser',font=('Arial',22,'bold'),bg=register\_bg).

pack(pady=10)

Label(register\_Frame, text='Name' ,underline= 0 ,bg=register\_bg ).pack()

reg\_entry01 = ttk.Entry(register\_Frame)

reg\_entry01.pack()

Label(register\_Frame, text='Email' ,bg=register\_bg ).pack()

reg\_entry02 = ttk.Entry(register\_Frame)

reg\_entry02.pack()

ttk.Label(register\_Frame, text='Password' ,background=register\_bg ).pack()

reg\_entry03 = ttk.Entry(register\_Frame , show='\*' )

reg\_entry03.pack()

ttk.Label(register\_Frame, text='Re-Enter Password' ,background=register\_bg ).pack()

reg\_entry04 = ttk.Entry(register\_Frame , show='\*' )

reg\_entry04.pack()

def register():

if reg\_entry01.get() != '' and reg\_entry02.get() != '' and reg\_entry03.get() != '':

if reg\_entry03.get() == reg\_entry04.get():

cur.execute("insert into userregister(Name,Email,Password,Status) values(%s,%s,%s,'Not Approved')",(reg\_entry01.get(),reg\_entry02.get(), reg\_entry03.get() ))

con.commit()

register\_Frame.destroy()

messagebox.showinfo('Success','Registered Successfully.')

user\_login()

else:

messagebox.showerror("Error","Password and Re-Enter Password doesn\'t Match.")

else:

messagebox.showerror('Error','Enter Name,Email and Password Correctly.')

ttk.Button( register\_Frame , text='Register' , command=register ).pack(pady=10)

register\_Frame.pack(pady=20)

user\_login()

def search\_Bar(name):

user.title(f'Welcome {name} ')

def search():

cur.execute("SELECT o.`Sno`, o.`rollnum`, o.`owner\_name`, o.`name`, o.`status`, o.`file\_key` FROM studentregister.studentreg o where Rollnum = %s ",(search\_Entry.get()))

result = cur.fetchall()

admin = Tk()

admin.title('User Search Results')

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=0, pady=20 , bg=admin\_bg )

if result:

list = [ 'S.No', 'Rollnum', 'Owner Name' , 'User Name', 'Status ', 'File Key' ]

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#000',)

i=6

b = {}

for s in result:

if s[len(s)-1] == 'Pending' :

for j in range(len(s)-1):

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

else:

for j in range(len(s)):

if j == len(s)-1:

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(show='\*',state='disabled',justify='center',foreground='#000',font=('Arial'))

elif s[j] == 'Verified':

b[s[j-1]] = ttk.Button(admin\_Frame)

def approve( x= s[j-3] , y= s[j-2] ,z= s[len(s)-1] ):

cur.execute("insert into file\_request(roll\_num,owner\_name,

user\_name,status,file\_key) values(%s,%s,%s,'Requested',%s) ;",(x,y,name,z))

con.commit()

messagebox.showinfo('Succes','Done ✔')

admin.destroy()

b[s[j-1]].config(text= 'Request',command=approve)

b[s[j-1]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

i=i+1

else :

Label(admin\_Frame,text='No Results Founded.').grid()

admin\_Frame.grid(row=5,column=10)

admin.mainloop()

def file\_request(name):

cur.execute("SELECT \* FROM studentregister.file\_request o where user\_name = %s ",(name))

result = cur.fetchall()

admin = Tk()

admin.title('User Search Results')

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=50, pady=20 , bg=admin\_bg )

if result:

list = [ 'S.No', 'Roll Name','Owner Name', 'User Name' , 'Status ', 'File Key' ]

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame,width=16)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#000',)

i=6

b = {}

for s in result:

for j in range(len(s)):

if s[len(s)-2] == 'Accepted':

e = ttk.Entry(admin\_Frame,width=16)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

i+=1

admin\_Frame.pack()

mainloop()

def show\_file():

down\_frame = Frame(background='#fff',padx=10,pady=10)

ttk.Label(down\_frame,text='Enter Your File Key').pack()

l1 = Label(down\_frame,background='#FFF')

admin\_bg = '#b3ffff'

admin\_Frame = Frame( down\_frame, padx=10, pady=50 ,bg=admin\_bg )

l1.pack()

e1 = ttk.Entry()

e1.pack()

def submit():

password = e1.get()

def decrypt(enc, password):

private\_key = hashlib.sha256(password.encode("utf-8")).digest()

enc = base64.urlsafe\_b64decode(enc)

iv = enc[:16]

cipher = AES.new(private\_key, AES.MODE\_CBC, iv)

return unpad(cipher.decrypt(enc[16:]))

cur.execute("SELECT\*FROM studentregister.student reg WHERE File\_key=%s;",(e1.get()))

result = cur.fetchall()

if not result:

messagebox.showerror("ERROR",'key invalid')

else:

import tkinter as tk

admin = tk.Tk()

admin.geometry('1250x700+0+10')

admin\_bg = '#69E2FF'

admin\_Frame = Frame( admin, padx=10, pady=550 ,bg=admin\_bg)

list = ['S.no' ,'Rollnum', 'Name', 'Std', 'Subone', 'Subtwo', 'Subthree', 'Subfour', 'Subfive', 'Subsix', 'Total', 'Grade', 'Subseven', 'Subeight', 'Subnine', 'Subten', 'Subeleven', 'Subtwelve', 'Total1', 'Grade2', 'File\_key', 'owner name']

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame,width=7)

en1.grid(row=2,column=i)

en1.insert(END, list[i])

i=21

b = {}

for s in result:

for j in range(len(s)):

if s[j] == 'Verified':

b[s[j]] = ttk.Button(admin\_Frame,width=16,)

def approve(x1 = s[j-18], x2 = s[j-17], x3 = s[j-16], x4 = s[j-15], x5= s[j-14], x6 = s[j-13], x7 = s[j-12], x8 = s[j-11], x9 = s[j-10], x10 = s[j-9], x11 = s[j-8], x12 = s[j-7], x13 = s[j-6], x14 = s[j-5], x15 = s[j-4], x16 = s[j-3]):

decrypted1 = decrypt(x1, password)

decrypted2 = decrypt(x2, password)

decrypted3 = decrypt(x3, password)

decrypted4 = decrypt(x4, password)

decrypted5 = decrypt(x5, password)

decrypted6 = decrypt(x6, password)

decrypted7 = decrypt(x7, password)

decrypted8 = decrypt(x8, password)

decrypted9 = decrypt(x9, password)

decrypted10 = decrypt(x10, password)

decrypted11 = decrypt(x11, password)

decrypted12 = decrypt(x12, password)

decrypted13 = decrypt(x13, password)

decrypted14 = decrypt(x14, password)

decrypted15 = decrypt(x15, password)

decrypted16 = decrypt(x16, password)

l2=tk.Label(admin,text="SUBONE:"+bytes.decode(decrypted1),width=30,font=('Algerian',16, 'bold'),bg='WHITE',fg='BLACK',)

l2.place(x=200, y=100, width=300)

l3= tk.Label(admin,text="SUB TWO: "+ bytes.decode(decrypted2),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l3.place(x=200, y=150, width=300)

l4=tk.Label(admin,text="SUBTHREE:"+bytes.decode(decrypted3),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l4.place(x=200, y=200, width=300)

l5=tk.Label(admin,text="SUBFOUR:"+bytes.decode(decrypted4),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l5.place(x=200, y=250, width=300)

l6=tk.Label(admin,text="SUBFIVE:"+bytes.decode(decrypted5),width=30,font=('Algerian',16, 'bold'),bg='WHITE’, fg='BLACK',)

l6.place(x=200, y=300, width=300)

l7=tk.Label(admin,text="SUBSIX:"+bytes.decode(decrypted6),width=30,font=('Algerian',16, 'bold'),bg='WHITE', fg='BLACK',)

l7.place(x=200, y=350, width=300)

l8=tk.Label(admin,text="TOTAL:"+bytes.decode(decrypted7),width=30,font=('Algerian',16, 'bold'),bg='WHITE',fg='BLACK',)

l8.place(x=200, y=400, width=300)

l9=tk.Label(admin,text="GRADE:"+bytes.decode(decrypted8),width=30,font=('Algerian', 16, 'bold'),bg='WHITE', fg='BLACK',)

l9.place(x=200, y=450, width=300)

l10=tk.Label(admin,text="SUBONE:"+bytes.decode(decrypted9),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l10.place(x=600, y=100, width=300)

l11=tk.Label(admin,text="SUBTWO: "+ bytes.decode(decrypted10),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l11.place(x=600, y=150, width=300)

l12=tk.Label(admin,text="SUBTHREE:"+bytes.decode(decrypted11),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l12.place(x=600, y=200, width=300)

l13=tk.Label(admin,text="SUBFOUR: "+ bytes.decode(decrypted12),width=30,font=('Algerian', 16, 'bold'),bg='WHITE', fg='BLACK',)

l13.place(x=600, y=250, width=300)

l14=tk.Label(admin,text="SUBFIVE: "+ bytes.decode(decrypted13),width=30,font=('Algerian', 16, 'bold'),bg='WHITE', fg='BLACK',)

l14.place(x=600, y=300, width=300)

l15= tk.Label(admin,text="SUB SIX: "+ bytes.decode(decrypted14),width=30,font=('Algerian', 16, 'bold'),bg='WHITE', fg='BLACK',)

l15.place(x=600, y=350, width=300)

l16 = tk.Label(admin,text="TOTAL: "+ bytes.decode(decrypted15),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l16.place(x=600, y=400, width=300)

l17 = tk.Label(admin,text="GRADE: "+ bytes.decode(decrypted16),width=30,font=('Algerian', 16, 'bold'),bg='WHITE',fg='BLACK',)

l17.place(x=600, y=450, width=300)

l18 =tk.Label(admin, text=' SEMESTER', font=('times new roman', 18, 'bold'), bg='WHITE',

fg='black', )

l18.place(x=200, y=50, width=300)

l19 =tk.Label(admin, text=' MODEL EXAM', font=('times new roman', 18, 'bold'), bg='WHITE',

fg='black', )

l19.place(x=600, y=50, width=300)

b[s[j]].config(text= 'VIEW DATA',command=approve)

b[s[j]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame,width=5)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state ='disabled',justify='center',foreground='#000',font=('Arial'))

i=i+1

admin\_Frame.grid(row=50,column=300)

admin.mainloop()

ttk.Button(down\_frame,text='Submit',command=submit).pack() down\_frame.pack()

search\_Frame=Frame(bg='#FFF',highlightcolor='#000',highlightbackground='#000',padx=10,pady=10,bd=1,)

ttk.Label(search\_Frame,text='EnterROLLNUM',font=('Bold'),background='#FFF').pack(pady=10)

logout\_Btn = ttk.Button(text='Logout' , command=user.destroy)

logout\_Btn.pack(padx=100,pady=10)

search\_Entry = ttk.Entry(search\_Frame,width=100)

search\_Entry.pack(pady=10)

ttk.Button(search\_Frame,text='SEARCH',command=search).pack(pady=10)

ttk.Button(search\_Frame,text=' REQUESTED FILE',command=lambda : file\_request(name)).pack()

ttk.Button(search\_Frame,text='SHOW FILE',command= show\_file ).pack()

search\_Frame.pack()

def tab3():

user.destroy()

import main1

tab3\_b=Button(user, text='HOME', font=('Times New Roman',13), command=tab3)

tab3\_b.place(x=1300, y=20, height=30, width=130,)

mainloop()

staff.py

from tkinter import \*

from tkinter import messagebox,ttk,filedialog

import pymysql,os

from random import \*

from cryptography.fernet import Fernet

con = pymysql.connect(host="localhost",user="root",password="admin",database="studentregister")

cur = con.cursor()

root= Tk()

root.geometry('1920x1080+0+10')

bg = PhotoImage(file='pic1.png')

bgLabel = Label(root,image=bg)

bgLabel.place(x=0, y=0)

def owner\_login():

login\_bg = '#FFF'

global login\_Frame

login\_Frame = Frame( padx=50, pady=20 , bg=login\_bg )

Label(login\_Frame,text= 'Teacher Login' , font= ('Arial',22,'bold') , bg=login\_bg ).pack(pady=10)

Label(login\_Frame,text='Email',textvariable='email' , font= ('Arial',14), bg=login\_bg ).pack(pady=1)

entry01 = ttk.Entry(login\_Frame)

entry01.pack(pady=2)

Label(login\_Frame,text='Password' , font= ('Arial',14), bg=login\_bg ).pack(pady=1)

entry02 = ttk.Entry(login\_Frame,show='\*')

entry02.pack(pady=2)

def login\_Close():

if entry01.get() != '' and entry02.get() != '':

cur.execute("select \* from registerloginform where Email=%s and Password=%s",(entry01.get(),entry02.get()))

row = cur.fetchone()

#print(row)

if row != None:

if row[4] == 'Approved':

login\_Frame.destroy()

staff\_screen(row[1])

else:

messagebox.showinfo('Wait','Wait For Admin Approval.')

else:

messagebox.showerror('Failed','Login Failed.')

else:

messagebox.showwarning("Alert","Enter Email & Password Correctly !!")

def tab3():

root.destroy()

import main1

tab3\_b=Button(root, text='HOME', font=('Times New Roman',13), command=tab3)

tab3\_b.place(x=1200, y=10, height=30, width=130,)

def open\_Register():

login\_Frame.destroy()

owner\_Register()

ttk.Button(login\_Frame,text='Login ,command=login\_Close).pack(pady=10)

Button(login\_Frame,text='Not Registered ?' , bd= 0 , bg=login\_bg , relief='flat' , overrelief='flat' , command=open\_Register ).pack(pady=10)

login\_Frame.pack()

def owner\_Register():

register\_bg = '#FFF'

register\_Frame = Frame( padx=50, pady=20 , bg = register\_bg )

Label(register\_Frame, text='Teacher Register' , font= ('Arial',22,'bold') , bg=register\_bg ).pack(pady=10)

Label(register\_Frame, text='Name' ,underline= 0 ,bg=register\_bg ).pack()

reg\_entry01 = ttk.Entry(register\_Frame)

reg\_entry01.pack()

Label(register\_Frame, text='Email' ,bg=register\_bg ).pack()

reg\_entry02 = ttk.Entry(register\_Frame)

reg\_entry02.pack()

ttk.Label(register\_Frame, text='Password' ,background=register\_bg ).pack()

reg\_entry03 = ttk.Entry(register\_Frame , show='\*' )

reg\_entry03.pack()

ttk.Label(register\_Frame, text='Re-Enter Password' ,background=register\_bg ).pack()

reg\_entry04 = ttk.Entry(register\_Frame , show='\*' )

reg\_entry04.pack()

def register():

if reg\_entry01.get() != '' and reg\_entry02.get() != '' and reg\_entry03.get() != '':

if reg\_entry03.get() == reg\_entry04.get():

cur.execute("insertintoregisterloginform(Name,Email,Password,Status)values(%s,%s,%s,'Not Approved')",(reg\_entry01.get(),reg\_entry02.get(), reg\_entry03.get() ))

con.commit()

register\_Frame.destroy()

owner\_login()

else:

messagebox.showerror("Error","Password and Re-Enter Password doesn\'t Match.")

else:

messagebox.showerror('Error','Enter Name,Email and Password Correctly.')

ttk.Button( register\_Frame , text='Register' , command=register ).pack(pady=10)

register\_Frame.pack(pady=20)

owner\_login()

def file\_request(name):

cur.execute("SELECT \* FROM studentregister.file\_request WHERE Owner\_name = %s ;",(name))

result = cur.fetchall()

# print(\*result)

admin = Tk()

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=0, pady=20 , bg=admin\_bg )

list = [ 'S.No' , 'Roll\_Num' , 'Owner\_Name' , 'User\_Name' , 'Status ' , 'File\_Key' ]

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame,width=16)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#000',)

i=6

b = {}

for s in result:

for j in range(len(s)):

if s[j] == 'Requested':

b[s[j]] = ttk.Button(admin\_Frame,width=16)

def approve( x = s[j+1] ):

cur.execute("UPDATE file\_request SET status = 'Accepted' WHERE file\_key = %s;",(x))

con.commit()

#print(x)

messagebox.showinfo('Success','Done ✔')

admin.destroy()

b[s[j]].config(text= 'Accept Request',command=approve)

b[s[j]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame,width=16)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state ='disabled',justify='center',foreground='#000',font=('Arial'))

admin\_Frame.grid(row=5,column=10)

admin.mainloop()

import base64

import hashlib

from Crypto.Cipher import AES

from Crypto import Random

BLOCK\_SIZE = 16

pad = lambda s: bytes(s + (BLOCK\_SIZE - len(s) % BLOCK\_SIZE) \* chr(BLOCK\_SIZE - len(s) % BLOCK\_SIZE),'utf-8')

unpad = lambda s: s[:-ord(s[len(s) - 1:])]

def staff\_screen(name):

def fileupload():

root.title(f'{name}')

l1 = ttk.Label()

l1.pack()

key = [chr(x) for x in range(65,91) ]

keys = ''

for i in range(6):

keys += choice(key)

Key\_Label = ttk.Label(text=f'Key : {keys}')

Key\_Label.pack()

titleLabel = Label(text='STUDENT DETAILS', font=('Times New Roman', 22, 'bold '), bg='white',fg='black', )

titleLabel.place(x=390, y=70)

rollnumLabel = Label( text='ROLL NUM', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

rollnumLabel.place(x=15, y=140, width=150)

entryRollnum = Entry( font=('times new roman', 18), bg='lightgray')

entryRollnum.place(x=180, y=140, width=150)

nameLabel = Label(text=' NAME', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

nameLabel.place(x=360, y=140, width=150)

entryName= Entry(font=('times new roman', 18), bg='lightgray')

entryName.place(x=530, y=140, width=150)

StdLabel = Label( text='STD', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

StdLabel.place(x=700, y=140, width=150)

entryStd = Entry( font=('times new roman', 18), bg='lightgray')

entryStd.place(x=860, y=140, width=150)

semesterLabel = Label( text=' SEMESTER', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

semesterLabel.place(x=150, y=200, width=150)

modelexamLabel = Label( text=' MODEL EXAM', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

modelexamLabel.place(x=590, y=200, width=200)

sub1Label = Label( text='SUB1', font=('times new roman', 18, 'bold'), bg='white', fg='black', )

sub1Label.place(x=15, y=260, width=150)

Subone = Entry( font=('times new roman', 18), bg='lightgray')

Subone.place(x=250, y=260, width=150)

sub2Label = Label( text='SUB2', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub2Label.place(x=15, y=300, width=150)

Subtwo= Entry(font=('times new roman', 18), bg='lightgray',)

Subtwo.place(x=250, y=300, width=150)

sub3Label = Label( text='SUB3', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub3Label.place(x=15, y=340, width=150)

Subthree = Entry( font=('times new roman', 18), bg='lightgray')

Subthree.place(x=250, y=340, width=150)

sub4Label = Label( text='SUB4', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub4Label.place(x=15, y=380, width=150)

Subfour = Entry(font=('times new roman', 18), bg='lightgray',)

Subfour.place(x=250, y=380, width=150)

sub5Label = Label(text='SUB5', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub5Label.place(x=15, y=420, width=150)

Subfive = Entry(font=('times new roman', 18), bg='lightgray')

Subfive.place(x=250, y=420, width=150)

sub6Label = Label( text='SUB6', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub6Label.place(x=15, y=460, width=150)

Subsix = Entry(font=('times new roman', 18), bg='lightgray',)

Subsix.place(x=250, y=460, width=150)

totalLabel = Label( text='TOTAL', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

totalLabel.place(x=15, y=500, width=150)

Total = Entry( font=('times new roman', 18), bg='lightgray')

Total.place(x=250, y=500, width=150)

gradeLabel = Label(text='GRADE', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

gradeLabel.place(x=15, y=540, width=150)

comboGrade = ttk.Combobox(font=("times new roman", 18,), state="readonly",)

comboGrade['values'] = ("A1", "A","B1","B","C1","C","D1","D")

comboGrade.place(x=250, y=540, width=150)

sub1Label = Label( text='SUB1', font=('times new roman', 18, 'bold'), bg='white', fg='black',)

sub1Label.place(x=500, y=260, width=150)

Subseven = Entry( font=('times new roman', 18), bg='lightgray')

Subseven.place(x=700, y=260, width=150)

sub2Label = Label( text='SUB2', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub2Label.place(x=500, y=300, width=150)

Subeight = Entry(font=('times new roman', 18), bg='lightgray',)

Subeight.place(x=700, y=300, width=150)

sub3Label = Label(text='SUB3', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub3Label.place(x=500, y=340, width=150)

Subnine= Entry( font=('times new roman', 18), bg='lightgray')

Subnine.place(x=700, y=340, width=150)

sub4Label = Label(text='SUB4', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub4Label.place(x=500, y=380, width=150)

Subten = Entry(font=('times new roman', 18), bg='lightgray',)

Subten.place(x=700, y=380, width=150)

sub5Label = Label( text='SUB5', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub5Label.place(x=500, y=420, width=150)

Subeleven = Entry( font=('times new roman', 18), bg='lightgray')

Subeleven.place(x=700, y=420, width=150)

sub6Label = Label( text='SUB6', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

sub6Label.place(x=500, y=460, width=150)

Subtwelve = Entry(font=('times new roman', 18), bg='lightgray',)

Subtwelve.place(x=700, y=460, width=150)

totalLabel = Label( text='TOTAL', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

totalLabel.place(x=500, y=500, width=150)

Total1 = Entry(font=('times new roman', 18), bg='lightgray')

Total1.place(x=700, y=500, width=150)

gradeLabel = Label( text='GRADE', font=('times new roman', 18, 'bold'), bg='white',

fg='black', )

gradeLabel.place(x=500, y=540, width=150)

comboGrade2 = ttk.Combobox( font=("times new roman", 18,), state="readonly",)

comboGrade2['values'] = ("A1", "A","B1","B","C1","C","D1","D")

comboGrade2.place(x=700, y=540, width=150)

check = IntVar()

def upload():

if entryRollnum.get() == '' or entryName.get() == '' or entryStd.get() == '' or Subone.get()==

'' or \Subtwo.get() == '' or Subthree.get() == '' or Subfour.get() == '' or Subfive.get()== '' or \

Subsix.get() == '' or Total.get() == '' or comboGrade.get() == '' or Subseven.get()== '' or \

Subeight.get() == '' or Subnine.get() == '' or Subten.get() == '' or Subeleven.get()== '' or \

Subtwelve.get() == '' or Total1.get() == '' or comboGrade2.get() ==messagebox.showerror('Error', "All Fields Are Required")

else:

try:

password = keys

def encrypt(raw, password):

private\_key = hashlib.sha256(password.encode("utf-8")).digest()

raw = pad(raw)

iv = Random.new().read(AES.block\_size)

cipher = AES.new(private\_key, AES.MODE\_CBC, iv)

return base64.urlsafe\_b64encode(iv + cipher.encrypt(raw))

message1 = Subone.get()

message2 = Subtwo.get()

message3 = Subthree.get()

message4 = Subfour.get()

message5 = Subfive.get()

message6 = Subsix.get()

message7 = Total.get()

message8 = comboGrade.get()

message9 = Subseven.get()

message10 = Subeight.get()

message11 = Subnine.get()

message12 = Subten.get()

message13 = Subeleven.get()

message14 = Subtwelve.get()

message15 = Total1.get()

message16 = comboGrade2.get()

encrypted1 = encrypt( message1, password)

encrypted2 = encrypt( message2, password)

encrypted3 = encrypt( message3, password)

encrypted4 = encrypt( message4, password)

encrypted5 = encrypt( message5, password)

encrypted6 = encrypt( message6, password)

encrypted7 = encrypt( message7, password)

encrypted8 = encrypt( message8, password)

encrypted9 = encrypt( message9, password)

encrypted10 = encrypt( message10, password)

encrypted11 = encrypt( message11, password)

encrypted12 = encrypt( message12, password)

encrypted13 = encrypt( message13, password)

encrypted14 = encrypt( message14, password)

encrypted15 = encrypt( message15, password)

encrypted16 = encrypt( message16, password)

con = pymysql.connect(host='localhost', user='root', password='admin', database='studentregister')

cur = con.cursor()

cur.execute("insert into studentreg (Rollnum, Name, Std, Subone, Subtwo, Subthree, Subfour, Subfive, Subsix, Total, Grade, Subseven, Subeight, Subnine, Subten, Subeleven, Subtwelve, Total1, Grade2, File\_key, Owner\_name, Status) values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'Pending')", (entryRollnum.get(), entryName.get(), entryStd.get(), encrypted1, encrypted2, encrypted3, encrypted4, encrypted5, encrypted6, encrypted7,encrypted8, encrypted9, encrypted10, encrypted11, encrypted12, encrypted13, encrypted14, encrypted15, encrypted16, password,name))

con.commit()

con.close()

messagebox.showinfo('Success', "Registration Successful")

except Exception as e:

showerror('Error', f"Error due to: {e}",)

registerbutton = Button(text="SUBMIT", bd=4, bg='white', command=upload, activebackground='white', activeforeground='white',)

registerbutton.place(x=300, y=620, width=100, height=40)

clearbutton = Button( text="CLEAR", bd=4, bg='white', command=fileupload, activebackground='white', activeforeground='white',)

clearbutton.place(x=500, y=620, width=100, height=40)

upload\_Btn = ttk.Button(text='register',command=fileupload )

upload\_Btn.pack()

file\_request(name)

logout\_Btn = ttk.Button(text='Logout' , command=root.destroy)

logout\_Btn.pack(padx=100,pady=10)

mainloop()

**6.2 SERVER-SIDE CODING:**

main.py

from tkinter import \*

from tkinter import ttk

import pymysql

#from GIF import \*

main = Tk()

main.geometry('1920x1080+0+10')

main.title("main")

main.config(bg = '#FFF')

bg = PhotoImage(file='pic2.png')

bgLabel = Label(main, image=bg)

bgLabel.place(x=0, y=0)

main\_Frame = Frame(main ,pady=10)

m\_x = 10

def tab6():

main.destroy()

import main1

def tab1():

main.destroy()

import Staff\_login

def tab2():

main.destroy()

import admin

def tab3():

main.destroy()

import main1

def tab4():

main.destroy()

import User\_login

def tab5():

main.destroy()

import main1

tab3\_b=Button(main, text='HOME', font=('Times New Roman',20), command=tab3)

tab3\_b.place(x=750, y=100, height=40, width=150,)

tab4\_b=Button(main, text='STAFF LOGIN', font=('Times New Roman',15), command=tab1)

tab4\_b.place(x=750, y=200, height=40, width=150,)

tab1\_b=Button(main, text='USER LOGIN', font=('Times New Roman',15), command=tab4)

tab1\_b.place(x=750, y=300, height=40, width=150,)

tab2\_b=Button(main, text='ADMIN', font=('Times New Roman',15), command=tab2)

tab2\_b.place(x=750, y=400, height=40, width=150,)

mainloop()

admin.py

from tkinter import \*

from tkinter import messagebox,ttk

import pymysql

import smtplib

root= Tk()

root.geometry('1920x1080+0+10')

bg = PhotoImage(file='pic3.png')

bgLabel = Label(root,image=bg)

bgLabel.place(x=0, y=0)

import base64

import hashlib

from Crypto.Cipher import AES

from Crypto import Random

BLOCK\_SIZE = 16

pad = lambda s: bytes(s + (BLOCK\_SIZE - len(s) % BLOCK\_SIZE) \* chr(BLOCK\_SIZE - len(s) % BLOCK\_SIZE),'utf-8')

unpad = lambda s: s[:-ord(s[len(s) - 1:])]

def user\_register():

con=pymysql.connect(host="localhost",user="root",password="admin",database="studenteregister”)

cur = con.cursor()

cur.execute("SELECT \* FROM studentregister.userregister ;")

result = cur.fetchall()

admin = Tk()

admin.title('User Register')

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=50, pady=20 , bg=admin\_bg )

list = [ 'S.No', 'Name', 'Email' , 'Password' , 'Status ' ]

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#00',)

i=6

b = {}

for s in result:

if s[len(s)-1] == 'Approved' :

for j in range(len(s)):

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

else:

for j in range(len(s)):

if s[j] == 'Not Approved':

print(s[j],'->',s[j-2])

b[s[j-2]] = ttk.Button(admin\_Frame)

def approve(x=s[j-2] ):

cur.execute("UPDATE userregister SET Status = 'Approved' WHERE Email %s;",(x))

con.commit()

messagebox.showinfo('Succes','Done ✔')

admin.destroy()

b[s[j-2]].config(text= 'Approve',command=approve)

b[s[j-2]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

i=i+1

admin\_Frame.grid(row=5,column=10)

admin.mainloop()

def owner\_register():

con=pymysql.connect(host="localhost",user="root",password="admin",database="studentregister")

cur = con.cursor()

cur.execute("SELECT \* FROM studentregister.registerloginform ;")

result = cur.fetchall()

admin = Tk()

admin.title('Staff Register')

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=50, pady=20 , bg=admin\_bg )

list = [ 'S.No', 'Name', 'Email' , 'Password' , 'Status ' ]

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#000',)

i=6

b = {}

for s in result:

if s[len(s)-1] == 'Approved' :

for j in range(len(s)):

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

else:

for j in range(len(s)):

if s[j] == 'Not Approved':

b[s[j-2]] = ttk.Button(admin\_Frame)

def approve(x = s[j-2]):

cur.execute("UPDATE registerloginform SET Status = 'Approved' WHERE Email = %s;",(x))

con.commit()

messagebox.showinfo('Succes','Done ✔')

admin.destroy()

b[s[j-2]].config(text= 'Approve',command=approve)

b[s[j-2]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

i=i+1

admin\_Frame.grid(row=5,column=10)

admin.mainloop()

def owner\_verify():

con=pymysql.connect(host="localhost",user="root",password="admin",

database="studentregister")

cur = con.cursor()

cur.execute("SELECT Sno, Rollnum, Name, Std, File\_Key, Owner\_name, Status FROM studentregister.studentreg ;")

result = cur.fetchall()

admin = Tk()

admin.title('Verify Files')

admin.geometry('1200x600+0+10')

admin\_bg = '#FFF'

admin\_Frame = Frame( admin, padx=0, pady=50 , bg=admin\_bg )

list = ['Sno','Rollnum' , 'Name', 'Std' , 'File\_Key ','Owner Name','Status']

for i in range(len(list)):

en1 = ttk.Entry(admin\_Frame,width=8)

en1.grid(row=3,column=i)

en1.insert(END, list[i])

en1.config(state='disabled',foreground='darkblue',justify='center',font=('bold'),background='#000',)

i=8

b = {}

for s in result:

if s[len(s)-1] == 'Verified' :

for j in range(len(s)):

e = ttk.Entry(admin\_Frame,width=8)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

else:

for j in range(len(s)):

if s[j] == 'Pending':

b[s[j-2]] = ttk.Button(admin\_Frame,width=10)

def approve( x = [s[j-2]] ):

cur.execute("UPDATE studentreg SET status = 'Verified' WHERE file\_key = %s;",(x))

con.commit()

messagebox.showinfo('Success','Done ✔')

admin.destroy()

b[s[j-2]].config(text= 'Verify',command=approve)

b[s[j-2]].grid(row=i,column=j)

else:

e = ttk.Entry(admin\_Frame,width=8)

e.grid(row=i,column=j)

e.insert(END, s[j])

e.config(state='disabled',justify='center',foreground='#000',font=('Arial'))

i=i+1

e1 = ttk.Entry(admin\_Frame,width=10)

e1.grid(row=1,column=1)

def view():

password = e1.get()

def decrypt(enc, password):

private\_key = hashlib.sha256(password.encode("utf-8")).digest()

enc = base64.urlsafe\_b64decode(enc)

iv = enc[:16]

cipher = AES.new(private\_key, AES.MODE\_CBC, iv)

return unpad(cipher.decrypt(enc[16:]))

cur.execute("SELECT \* FROM studentregister.studentreg WHERE File\_key = %s;",(e1.get()))

result = cur.fetchall()

i=21

b = {}

for s in result:

#for j in range(len(s)):

#if s[j] == 'Verified':

#b[s[j]] =Button(width=16,)

def approve(x1 = s[j-1], x2 = s[j-2], x3 = s[j-0], x4 = s[j+1], x5= s[j+2], x6 = s[j-13], x7 = s[j-12], x8 = s[j-11], x9 = s[j-10], x10 = s[j+9], x11 = s[j+8], x12 = s[j+7], x13 = s[j+6], x14 = s[j+5], x15 = s[j+4], x16 = s[j+3]):

decrypted1 = decrypt(x1, password)

print(decrypted1)

decrypted2 = decrypt(x2, password)

print(decrypted2)

decrypted3 = decrypt(x3, password)

print(decrypted3)

decrypted4 = decrypt(x4, password)

print(decrypted4)

decrypted5 = decrypt(x5, password)

print(decrypted5)

decrypted6 = decrypt(x6, password)

print(decrypted6)

decrypted7 = decrypt(x7, password)

print(decrypted7)

decrypted8 = decrypt(x8, password)

print(decrypted8)

decrypted9 = decrypt(x9, password)

print(decrypted9)

decrypted10 = decrypt(x10, password)

print(decrypted10)

decrypted11 = decrypt(x11, password)

print(decrypted11)

decrypted12 = decrypt(x12, password)

print(decrypted12)

decrypted13 = decrypt(x13, password)

print(decrypted13)

decrypted14 = decrypt(x14, password)

print(decrypted14)

decrypted15 = decrypt(x15, password)

print(decrypted15)

decrypted16 = decrypt(x16, password)

print(decrypted16)

l2=ttk.Label(admin,text="SUBONE: "+bytes.decode(decrypted2),width=30,font=('Algerian', 16, 'bold'))

l2.place(x=700, y=100, width=300)

l3=ttk.Label(admin,text="SUBTWO:"+bytes.decode(decrypted1),width=30,font=('Algerian', 16, 'bold'))

l3.place(x=700, y=150, width=300)

l4=ttk.Label(admin,text="SUBTHREE:"+bytes.decode(decrypted3),width=30,font=('Algerian', 16, 'bold'))

l4.place(x=700, y=200, width=300)

l5=ttk.Label(admin,text="SUBFOUR:"+bytes.decode(decrypted4),width=30,font=('Algerian', 16, 'bold'))

l5.place(x=700, y=250, width=300)

l6=ttk.Label(admin,text="SUBFIVE:"+bytes.decode(decrypted5),width=30,font=('Algerian', 16, 'bold'))

l6.place(x=700, y=300, width=300)

l7=ttk.Label(admin,text="SUBSIX:"+bytes.decode(decrypted16),width=30,font=('Algerian', 16, 'bold'))

l7.place(x=700, y=350, width=300)

l8 = ttk.Label(admin,text="TOTAL: "+ bytes.decode(decrypted15),width=30,font=('Algerian', 16, 'bold'))

l8.place(x=700, y=400, width=300)

l9 = ttk.Label(admin,text="GRADE: "+ bytes.decode(decrypted14),width=30,font=('Algerian', 16, 'bold'))

l9.place(x=700, y=450, width=300)

l10= ttk.Label(admin,text="SUB ONE: "+ bytes.decode(decrypted13),width=30,font=('Algerian', 16, 'bold'))

l10.place(x=950, y=100, width=300)

111=ttk.Label(admin,text="SUBTWO:"+bytes.decode(decrypted12),width=30,font=('Algerian', 16, 'bold'))

l11.place(x=950, y=150, width=300)

l12=ttk.Label(admin,text="SUBTHREE:"+bytes.decode(decrypted11),width=30,font=('Algerian', 16, 'bold'))

l12.place(x=950, y=200, width=300)

l13=ttk.Label(admin,text="SUBFOUR:"+bytes.decode(decrypted10),width=30,font=('Algerian', 16, 'bold'))

l13.place(x=950, y=250, width=300)

l14= ttk.Label(admin,text="SUB FIVE: "+ bytes.decode(decrypted6),width=30,font=('Algerian', 16, 'bold'))

l14.place(x=950, y=300, width=300)

l15 = ttk.Label(admin,text="SUB SIX: "+ bytes.decode(decrypted7),width=30,font=('Algerian', 16, 'bold'))

l15.place(x=950, y=350, width=300)

l16 = ttk.Label(admin,text="TOTAL: "+ bytes.decode(decrypted8),width=30,font=('Algerian', 16, 'bold'))

l16.place(x=950, y=400, width=300)

l17 = ttk.Label(admin,text="GRADE: "+ bytes.decode(decrypted9),width=30,font=('Algerian', 16, 'bold'))

l17.place(x=950, y=450, width=300)

l18 =ttk.Label(admin, text=' SEMESTER', font=('times new roman', 18, 'bold'))

l18.place(x=700, y=50, width=250)

l19 =ttk.Label(admin, text=' MODEL EXAM', font=('times new roman', 18, 'bold'))

l19.place(x=950, y=50, width=250)

l2=ttk.Button(admin\_Frame,text= 'ENTER',command = approve,width=10)

l2.grid(row=1,column =2)

i=i+1

l1=ttk.Button(admin\_Frame,text= 'CLICK',command = view,width=10)

l1.grid(row=1,column =0)

admin\_Frame.grid(row=10,column=80)

admin.mainloop()

logout\_Btn = ttk.Button(text='Logout' , command=root.destroy)

logout\_Btn.pack(padx=100,pady=10)

img\_user = PhotoImage(file = "Assets/user1.png").subsample(3,3)

ttk.Button(text='USERDETAILS',image=img\_user,compound=TOP,command=user\_register).

pack()

img\_owner = PhotoImage(file = "Assets/Owner.png").subsample(1,1)

ttk.Button(text='STAFF DETAILS' , image=img\_owner , compound=TOP ,command=owner\_register).pack()

img\_verify = PhotoImage(file = "Assets/verified.png").subsample(8,8)

ttk.Button(text='VERIFY STAFF FILE' , image=img\_verify , compound=TOP ,command=owner\_verify ).pack()

def tab3():

user.destroy()

import main1

tab3\_b=Button(user, text='HOME', font=('Times New Roman',13), command=tab3)

tab3\_b.place(x=1300, y=20, height=30, width=130)

mainloop()

**CHAPTER 7**

**SYSTEM TESTING**

**7.1 UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**7.1.1 WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**7.1.2. BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**7.1.3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**7.1.4. DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

**7.1.5. LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and

tested as above.

* Each unit has been separately tested by the development team itself and all the input have been validated.

**7.2 FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**7.3 BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**7.4 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actualy run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**7.5 PERFORMANCE ANALYSIS**

The block generation time of different schemes on centos7, using Intel® Core i7-11390H Processor (12M Cache, up to 5.00 GHz, with IPU). As shown in Figure 7, the calculation cost of this scheme is lower than the EduRSS , EduCTX schemes and the PBFT used in the reference paper.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Computing Cost | Privacy Protection | Energy  Consumption |
| EduRss(PoW) | High | Satisfy | High |
| EduCTX(DPOS) | Medium | Satisfy | Medium |
| Reference  Paper(PBFT) | Low | Satisfy | Low |
| Our Scheme(PBFT) | Very Low | Satisfy | Very Low |

In energy consumption in our scheme is very low compared to the reference paper PBFT. Similarly computing cost and privacy protection is satisfied and very low.

**CHAPTER 8**

**CONCLUSION**

**8.1 CONCLUSION**

Aiming at the need for protection and sharing of educational records, a secure storage and sharing scheme based on the blockchain, referred to as EduRSS is proposed in this paper. In our proposal, the integrity and security of the data can be ensured by the consortium chain between institutions. A distributed institution authentication mechanism is proposed to ensure the security of blockchain nodes. Secure Storage is achieved by combining Blockchain and Storage Server. For records sharing, to achieve cross-institutional sharing of educational records, smart contracts are introduced, the permissions of record and the records of the sharing process are managed by smart contracts on the blockchain. Finally, an anti-tampering inspection mechanism is employed to protect records in the storage server. In theory, the proposed scheme with higher security, efficiency, and credibility, but further research works are still needed.

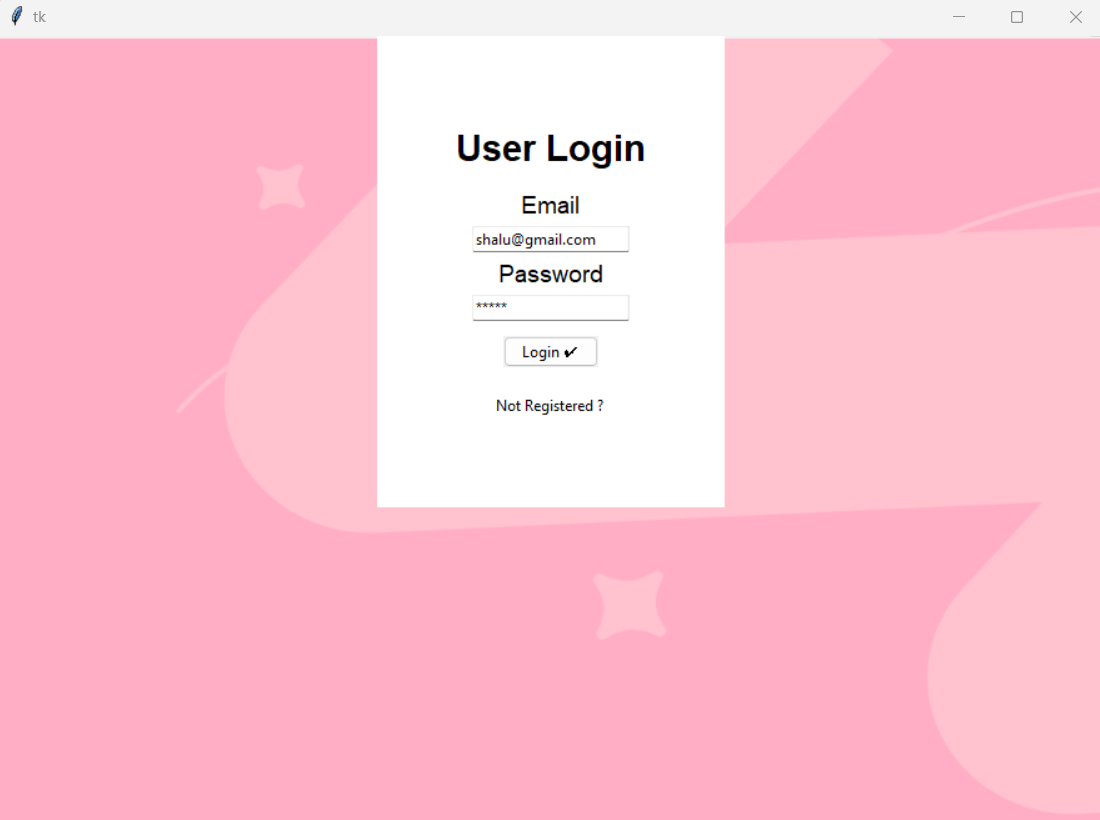
**8. 2 FUTURE WORKS:**

We still need a secure platform to manage these smart contracts in use. Since many smart contracts have been applied in our scheme and there may be more in the future, there is a need for a professional platform for deploying, scheduling, and managing smart contracts. In addition, the security of smart contracts is one of our main focus for future research. More functions need to be introduced into the framework, such as support for educational record certification of external institutions or employers, and encrypted retrieval of educational records. The storage of the off-chain data in our scheme depends on the centralized storage server. In the future, decentralized storage technologies such as the Inter Planetary File System (IPFS) and Storage will be used.

**APPENDICES**

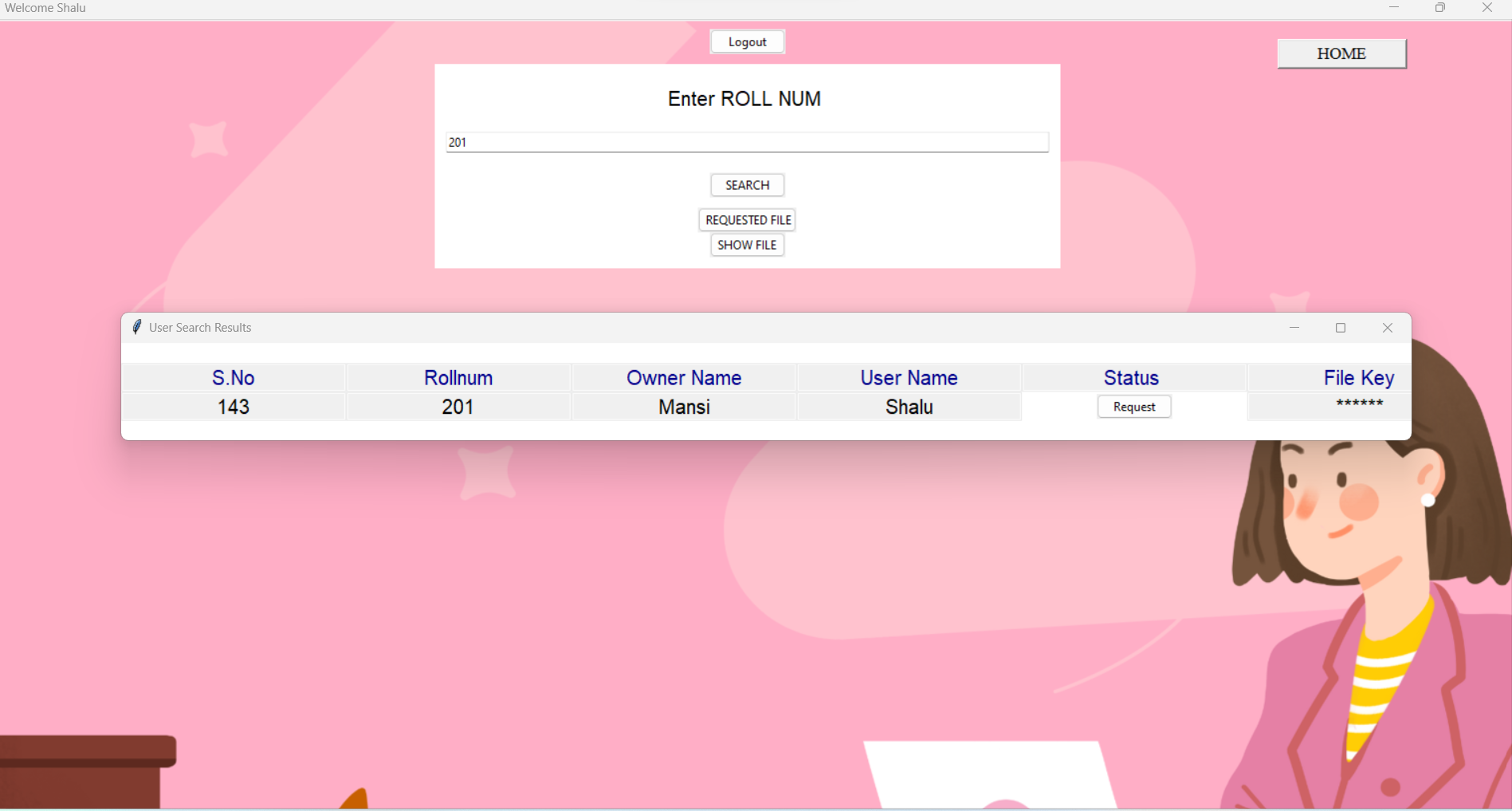
**A.1 SAMPLE SCREENS**

**User login:**

****

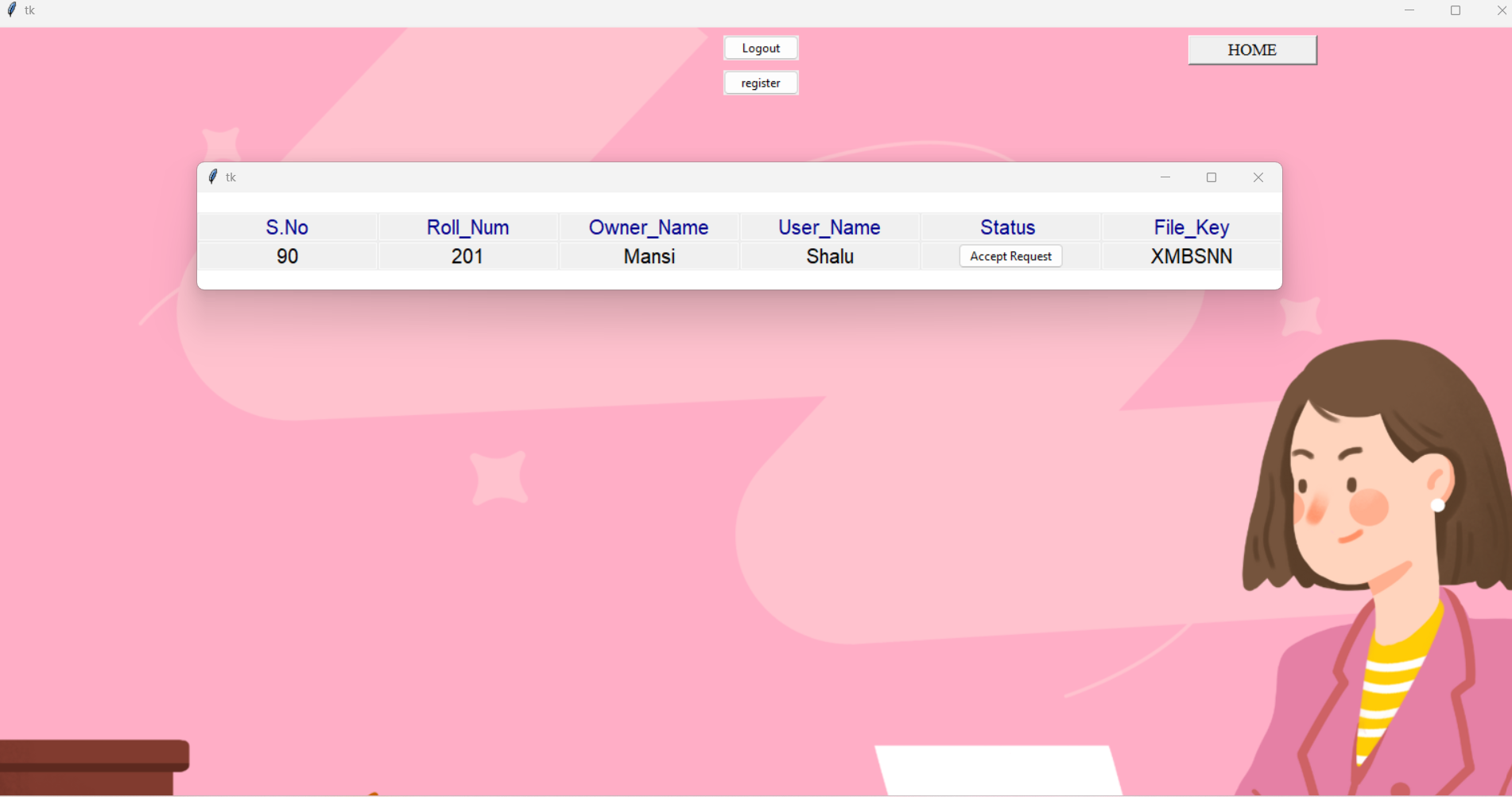
**Fig no:A.1**

**Request academic data:**



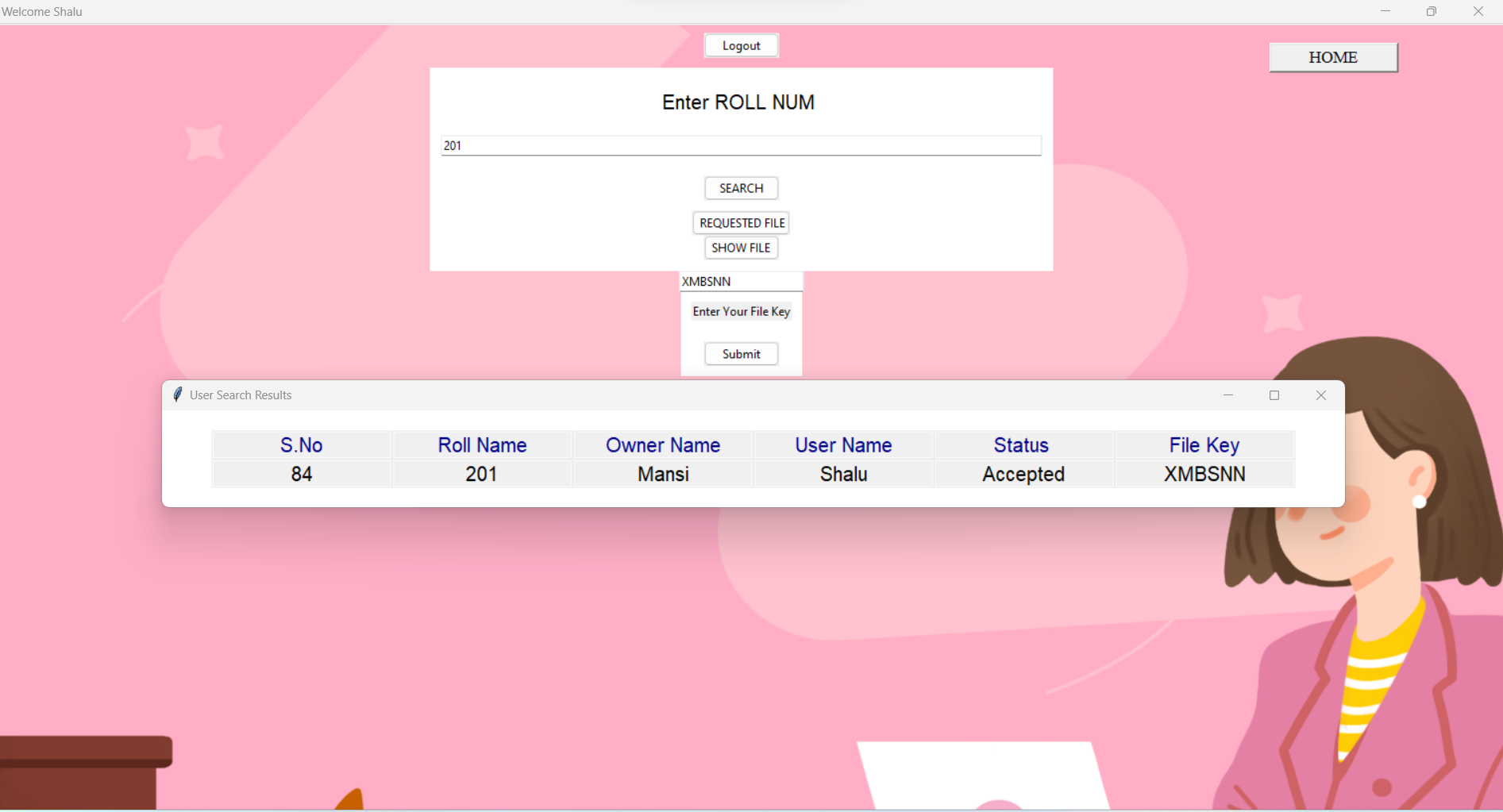
**Fig no: A.2**

**Staff provides access & key sent to user:**

****

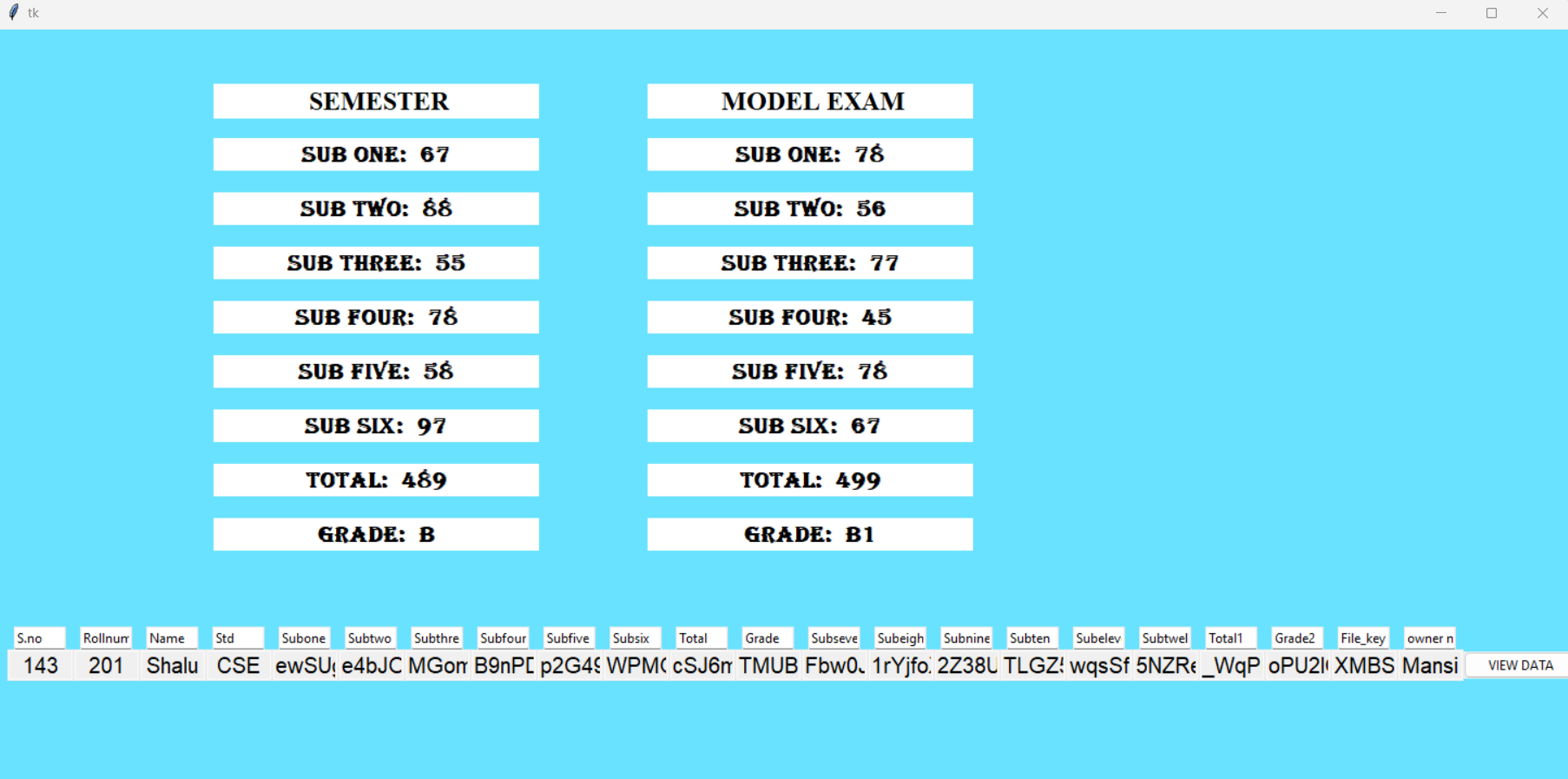
**Fig no: A.3**

**Enter Staff approved key:**

****

**Fig no:A.4**

**Access to Academic data:**



**Fig no:A.5**

**A.2 PUBLICATIONS**

[1] “Review of major global data leakage events in the first half of 2020,” https://www.isccc.gov.cn/xwdt/xwkx/07/903972. shtml, January 2020.

[2] H. Li and D. Han, “Edurss: A blockchain-based educational records secure storage and sharing scheme,” IEEE Access, vol. 7, 2019, pp. 179 273–179 289.[3] A. F. M. S. Akhter, M. Ahmed, et al., “A secured privacypreserving multi-level blockchain framework for cluster based vanet,” Sustainability, vol. 13, no. 1, 2021, p. 400.[4] C. Wang, S. Chen, et al., “Block chain-based data audit and access control mechanism in service collaboration,” in 2019 IEEE International Conference on Web Services (ICWS), 2019, pp. 214–218.

[5] H. Huang, P. Zhu, et al., “A blockchain-based scheme for privacy-preserving and secure sharing of medical data,” Computers & Security, vol. 99, 2020, p. 102010.

[6] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei, and P. Hong, ‘‘An attributebased controlled collaborative access control scheme for public cloud storage,’’ IEEE Trans. Inf. Forensics Security, vol. 14, no. 11, pp. 2927–2942,Nov. 2019.

[7] X. Feng, P. Deng, et al., “Verifiable decentralized access control for distributed databases,” in 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2020, pp. 248.

[8] B. Pillai and K. Biswas, “Cross-chain interoperability among blockchain-based systems using transactions,” Knowledge Engineering Review, vol. 35, 2020, p.1.

[9] A. Derhab, M. Guerroumi, A. Gumaei, L. Maglaras, M. A. Ferrag, M. Mukherjee, and F. A. Khan, ‘‘Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security,’’ Sensors, vol. 19, no. 14, p. 3119, 2019.

[10] A. Wu, Y. Zhang, X. Zheng, R. Guo, Q. Zhao, and D. Zheng, ‘‘Efficient and privacy-preserving traceable attribute-based encryption in blockchain,’’ Ann. Telecommun., vol. 74, nos. 7–8, pp. 401–411, Aug. 2019.