## A Practical Report Submitted in fulfillment of the Degree of

## MASTER OF SCIENCE

In

## COMPUTER SCIENCE

Year 2023-2024

By

## MR.YOGESH VISHWAS DHAMKE

### (Application ID: 41752)

Underguidance of

## PROF.TRUPTI RONGARE



Institute of Distance and Open Learning, Vidya Nagar, Kalina, Santacruz East- 400098.

University of Mumbai



## INSTITUTE OF DISTANCE AND OPEN LEARNING

### Vidya Nagari, Kalina, Santacruz East – 400098 CERTIFICATE

This is to certify that,

**Mr.Yogesh Vishwas Dhamke**, ApplicationID: **41752,**

Student of Master of Science in Computer Sciencehas Sat is factorily Completed the Practical in

**Cyber and Information Security (Network Security)**.

# Name Application ID

Mr. Yogesh Vishwas Dhamke 41752

# Subject In-charge Examiner

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Date** | **List of Practical** | **Signature** |
| 1. |  | Write a program to store username and password in an encrypted for mina database to  Implement integrity lock. |  |
| 2. |  | Write SQL query to retrieve sensitive information from less sensitive queries |  |
| 3. |  | Write SQL query to create view to implement concept of views and commutative filter in  Distributed databases. |  |
| 4. |  | Write a program to implement SSL. |  |
| 5. |  | Write a program to send an encrypted email. |  |
| 6. |  | Write a program to digitally sign MIME to create an ‘opaque’ signature. |  |
| 7. |  | Write a program to generate DSA SSH key. |  |
| 8. |  | Write a program to implement multilevel security. |  |

**PRACTICAL 1:**

WRITE A PROGRAM TO STORE USERNAME AND PASSWORD IN AN ENCRYPTED FOR MINA DATABASE TOIMPLEMENT INTEGRITY LOCK.

##### Python Program:

#PRACTICAL1:

Import mysql.connector as ms Mydb = ms.connect(

host="localhost", user="root", password="student",

)

mycursor = mydb.cursor() defsql\_execute(statement):

mycursor.execute(statement) try:

try:

sql\_execute("DROPDATABASEAR;") except:

pass

sql\_execute("CREATEDATABASEAR;") mydb = ms.connect( host="localhost",

user="root", password="student", database = 'ar'

)

mycursor=mydb.cursor()

sql\_execute("CREATETABLEUSER(usernamevarchar(255),password varchar(255));")

sql\_execute("INSERTINTOUSER(username,password)values ('Abhishek',ENCODE('GoodPassword@123',''));")

sql\_execute("INSERTINTOUSER(username,password)values ('Mandar',ENCODE('worstpassword',''));")

exceptExceptionase: print(e)

mydb.commit()

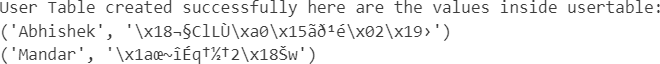
sql\_execute("SELECT\*FROMUSER;")

results=mycursor.fetchall()

print("UserTablecreatedsuccessfullyherearethevaluesinsideusertable:") for x in results:

print(x)

**Output**:



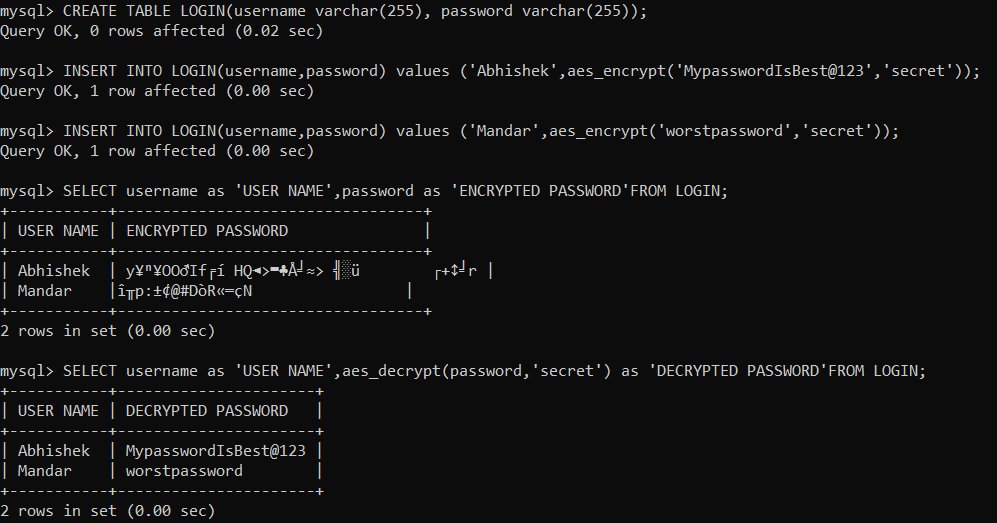
WRITE SQL QUERY TO RETRIEVE SENSITIVE INFORMATION FROM LESS SENSITIVE QUERIES.

##### SQLQuery:

CREATETABLELOGIN(usernamevarchar(255),passwordvarchar(255)); INSERT INTO LOGIN(username,password) values ('Abhishek',aes\_encrypt('MypasswordIsBest@123','secret'));INSERT INTO LOGIN(username,password) values ('Mandar',aes\_encrypt('worstpassword','secret'));

SELECT username as 'USER NAME',password as 'ENCRYPTED PASSWORD'FROM LOGIN; SELECTusernameas'USERNAME',aes\_decrypt(password,'secret')as'DECRYPTED PASSWORD'FROM LOGIN;

**Output:**



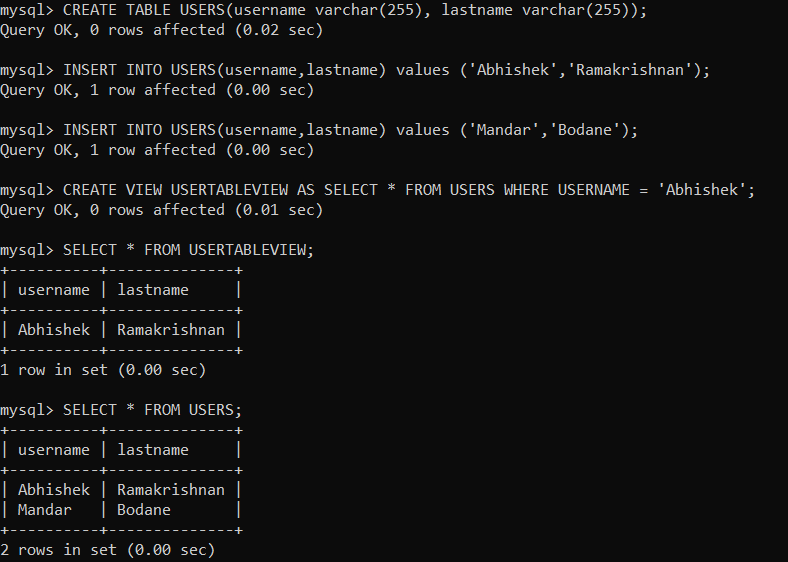
WRITE SQL QUERY TO CREATE VIEW TO IMPLEMENT CONCEPT OF VIEWS AND COMMUTATIVE FILTER IN DISTRIBUTED DATABASES.

##### SQL Query:

CREATETABLEUSERS(usernamevarchar(255),lastnamevarchar(255)); INSERTINTOUSERS(username,lastname)values('Abhishek','Ramakrishnan'); INSERT INTO USERS(username,lastname) values ('Mandar','Bodane');

CREATEVIEWUSERTABLEVIEWASSELECT\*FROMUSERSWHEREUSERNAME='Abhishek'; SELECT \* FROM USERTABLEVIEW;

**Output:**



**WRITE A PROGRAM TO IMPLEMENT SSL.**

##### Python Program:

#Practical4:

importsocket import ssl

hostname=input("Enterwebsitetosecurelyconnect:") context = ssl.create\_default\_context()

print("Websitesecurelyconnectedherearethedetailsoftheconnection:") with socket.create\_connection((hostname, 443)) as sock:

withcontext.wrap\_socket(sock,server\_hostname=hostname)asssock:

print(ssock) print(ssock.version())

**Output:**



#### PRACTICAL 5:

**WRITE A PROGRAM TO IMPLEMENT SSL.**

##### Python Program:

#PRACTICAL5:

importsmtplib,ssl

fromemail.mime.textimportMIMEText

fromemail.mime.multipartimportMIMEMultipart

sender\_email = ["put\_sender\_email@xyz.com"](mailto:put_sender_email@xyz.com)receiver\_email="[put\_reciever\_email@xyz.com"](mailto:put_reciever_email@xyz.com)password = "put\_password\_here"

message=MIMEMultipart("alternative") message["Subject"] = "Test Mail"message["From"] = sender\_email message["To"] = receiver\_email

#Createtheplain-textandHTMLversionofyourmessage text = "This is a digitally encrypted mail"

#Turntheseintoplain/htmlMIMETextobjects part1 = MIMEText(text, "plain")

message.attach(part1)

#Createsecureconnectionwithserverandsendemail context = ssl.create\_default\_context()

withsmtplib.SMTP\_SSL("smtp.gmail.com",465,context=context)asserver: server.login(sender\_email, password)

server.sendmail(

sender\_email,receiver\_email,message.as\_string()

)

print("MailSent!")

**Output:**



WRITE A PROGRAM TO DIGITALLY SIGN MIME TO CREATE AN ‘OPAQUE’ SIGNATURE.

**Python Program:**

import os importemail

fromemail.mime.multipartimportMIMEMultipart from email.mime.text import MIMEText

fromcryptography.hazmat.backendsimportdefault\_backend

fromcryptography.hazmat.primitivesimporthashes,serialization from cryptography.hazmat.primitives.asymmetric import padding

fromcryptography.hazmat.primitives.serializationimportload\_pem\_private\_key

defcreate\_mime\_message(sender,recipient,subject,body): msg = MIMEMultipart()

msg["From"] = sender msg["To"] = recipient msg["Subject"]=subject

msg.attach(MIMEText(body,"plain")) return msg

defsign\_mime\_message(mime\_message,private\_key\_path): with open(private\_key\_path, "rb") as key\_file:

private\_key=load\_pem\_private\_key(key\_file.read(),password=None, backend=default\_backend())

signature=private\_key.sign( mime\_message.as\_bytes(), hashes.SHA256()

)

returnsignature

ifname=="main":sender=["sender@example.com"](mailto:sender@example.com)

recipient=["recipient@example.com"](mailto:recipient@example.com)subject = "Test Email"

body="Thisisadigitallysignedemail."

mime\_message=create\_mime\_message(sender,recipient,subject,body) private\_key\_path = "privatekey.pem"

signature = sign\_mime\_message(mime\_message, private\_key\_path) mime\_message["X-Signature"]=signature.hex()#Addthesignatureasa

headertotheMIMEmessage

#Sendorsavethemime\_messagewiththeaddedsignature print(mime\_message.as\_string())

##### Output:

Content-Type:multipart/mixed;boundary="===============0103498092230740333==" MIME-Version: 1.0

From: [sender@example.com](mailto:sender@example.com) To:[recipient@example.com](mailto:recipient@example.com) Subject: Test Email

X-Signature: 302c02145217fa68a6cf6f1d437cfc020e1b31820a30a6ba021407228577e5f08ddd311a31ef7e 8d7aaa11c5ac55

--===============0103498092230740333==

Content-Type:text/plain;charset="us-ascii" MIME-Version: 1.0

Content-Transfer-Encoding:7bit

Thisisadigitallysignedemail.

--===============0103498092230740333==--

#### PRACTICAL 7:

**WRITE A PROGRAM TO GENERATE DSA SSH KEY.**

##### Python Program:

#PRACTICAL7:

fromCrypto.PublicKeyimportDSA # Create a new DSA key

output=input()

key=DSA.generate(1024)

withopen("{}.pri".format(output),"wb")asf: # Write the private key to file f.write(key.export\_key('PEM')) print("PrivateKeywrittensuccessfully!")

with open("{}.pub".format(output),"wb") as g: g.write(key.public\_key().export\_key('PEM')) print("Public Key written successfully!")

print(key.export\_key()) print(key.public\_key().exportKey())

##### Output:

PrivateKeywrittensuccessfully! Public Key written successfully!

b'-----BEGINPRIVATEKEY-----

\nMIIBTAIBADCCASwGByqGSM44BAEwggEfAoGBAN1nceg37amfW+/JozwnDi4JFylX\nFv3s3WJEfx zr9Eo1Y9hXMmxZhb/A/9JSD4vBZAoaMdbLxTF4MerzdVqfyrRuILH+\nzKkA+8y7rzpKez6tDnIcCY EIWpdJQbTg2PY09dukOuitcqDL1tSPjkZIBpgmwpKv\nPU/x4eoz6bMg8KUpAhUA+n4hdhskRlu38l bU4KqBB7eABE0CgYEAlwgjW1PyKdfZ\nASo4W/ZyeC1GXWXZ3mMll2dYCGZIirXjHC737WqkknKJZT NBrrPXq/Mndz79rnzp\n65Cnx0/JrKwPvflGSH2vMY48pi2O9LM0x8NwVvmUW6TAoTuXrFO7Q/eTBJ e/6RZk\nmxaG/+VhUnhGU4Gd7y2f9rOJH8Oo910EFwIVALF1+ILajBvh8FzPRUz4bBK8wkee\n----

-ENDPRIVATEKEY'

b'-----BEGINPUBLICKEY-----

\nMIIBtzCCASwGByqGSM44BAEwggEfAoGBAN1nceg37amfW+/JozwnDi4JFylXFv3s\n3WJEfxzr9E o1Y9hXMmxZhb/A/9JSD4vBZAoaMdbLxTF4MerzdVqfyrRuILH+zKkA\n+8y7rzpKez6tDnIcCYEIWp dJQbTg2PY09dukOuitcqDL1tSPjkZIBpgmwpKvPU/x\n4eoz6bMg8KUpAhUA+n4hdhskRlu38lbU4K qBB7eABE0CgYEAlwgjW1PyKdfZASo4\nW/ZyeC1GXWXZ3mMll2dYCGZIirXjHC737WqkknKJZTNBrr PXq/Mndz79rnzp65Cn\nx0/JrKwPvflGSH2vMY48pi2O9LM0x8NwVvmUW6TAoTuXrFO7Q/eTBJe/6R ZkmxaG\n/+VhUnhGU4Gd7y2f9rOJH8Oo910DgYQAAoGAR5VEMhiLYhFFGeb5XbQm2ww4Rxub\n5fHO h2bgmZNqAH24zcCfO6tugayDRcSGETMD7CCZx1uCFwGA1G/M/q1Pyqsvsv3B\n0mgX2zLQEqdKCTSl naOOzd7+ASu1Z0xzKswji7TZ2fUZLCYPKi5iy1JDqI0VzfCE\n+OFHQq6+yY8eBR0=\nEND

PUBLICKEY'

#### PRACTICAL 8:

**WRITE A PROGRAM TO IMPLEMENT MULTILEVEL SECURITY.**

##### Python Program:

#Practical8 class User:

definit(self,username,password,level):

self.username=username self.password=password self.level = level

class SecuritySystem: definit(self):

self.users=[]

defadd\_user(self,username,password,level): user = User(username, password, level) self.users.append(user)

defauthenticate(self,username,password): for user in self.users:

ifuser.username==usernameanduser.password==password: return user

returnNone

defhas\_access(self,user,required\_level): return user.level >= required\_level

if name== "main": security\_system=SecuritySystem()

#Add users

security\_system.add\_user("user1","pass1","Regular") security\_system.add\_user("user2", "pass2", "Admin")

#Userlogin

username=input("Enterusername:") password=input("Enterpassword:")

user=security\_system.authenticate(username,password) if user:

print(f"Welcome,{user.username}!Youarea{user.level}user.")

ifsecurity\_system.has\_access(user,"Admin"): print("You have admin privileges.")

else:

print("Youhaveregularuserprivileges.")

else:

print("Authenticationfailed.")

**Output:**

