

INDEX

Sr no	AIM	DATE
1	Write a program to store username and password in an encrypted form in a database to implement integrity lock.	
2	Write SQL query to retrieve sensitive information from less sensitive queries.	
3	Write SQL query to create a view to implement concept of views and commutative filter in distributed databases.	
4	Write a program to implement SSL.	
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	
9	Write a program to Demonstrates how to encrypt and decrypt the content of an XML node using 128-bit CBC AES encryption.	

Practical No 1

Aim: Write a program to store username and password in an encrypted form in a database to implement integrity lock.

Theory:

MD5:

MD5 is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes. MD5 algorithm stands for the message-digest algorithm. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always 128 bits. MD5 was developed in 1991 by Ronald Rivest.

Steps:

Step1: Create a table to store username and password.

```
create table UserNamePasswordDemo(  
U_Id int(10) unsigned NOT NULL AUTO_INCREMENT primary key,  
UserId varchar(255) DEFAULT NULL unique key,  
UserPassword varchar(255) DEFAULT NULL  
);
```

```
mysql> create table UserNamePasswordDemo(  
-> U_Id int(10) unsigned NOT NULL AUTO_INCREMENT primary key,  
-> UserId varchar(255) DEFAULT NULL unique key,  
-> UserPassword varchar(255) DEFAULT NULL  
-> );  
Query OK, 0 rows affected, 1 warning (0.07 sec)  
  
mysql>
```

Step2: Insert records in the table.

```
insert into UserNamePasswordDemo(UserId,UserPassword) values('deep@gg.com',MD5('deep1234'));  
insert into UserNamePasswordDemo(UserId,UserPassword) values('raju@gg.com',MD5('raju1234'));  
insert into UserNamePasswordDemo(UserId,UserPassword) values('magan@gg.com',MD5('magan1234'));
```

```
mysql> insert into UserNamePasswordDemo(UserId,UserPassword) values('deep@gg.com',MD5('deep1234'));  
Query OK, 1 row affected (0.05 sec)  
  
mysql> insert into UserNamePasswordDemo(UserId,UserPassword) values('raju@gg.com',MD5('raju1234'));  
Query OK, 1 row affected (0.04 sec)  
  
mysql> insert into UserNamePasswordDemo(UserId,UserPassword) values('magan@gg.com',MD5('magan1234'));  
Query OK, 1 row affected (0.04 sec)
```

Step3: Select query.

```
Select * from UserNamePasswordDemo;
```

```
mysql> select * from UserNamePasswordDemo;  
+-----+-----+-----+  
| U_Id | UserId      | UserPassword |  
+-----+-----+-----+  
| 1    | deep@gg.com | 594dcb38ce95f3573e9cc1cc13142d3c |  
| 2    | raju@gg.com | fe783d3587fb00d99a0045324acb861c |  
| 3    | magan@gg.com | 8547b147be3290ec7a26c85705cea9ec |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

Conclusion:

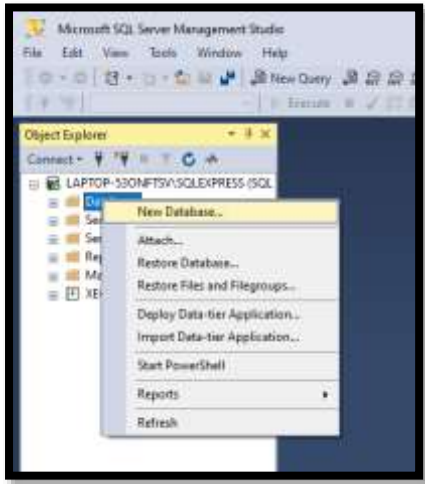
Successfully implemented a program to store username and password in an encrypted form in a database to implement integrity lock.

Practical No 2

Aim: Write SQL query to retrieve sensitive information from less sensitive queries.

Steps:

Step1: Create new database.



Step2: Create table employee.

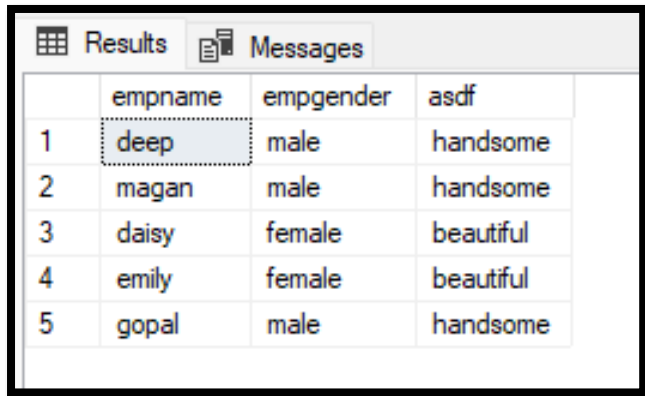
```
Create table employee(  
empid int,  
empname char(20),  
empgender char(20),  
empsalary int  
);
```

Step3: Insert values in employee table.

LAPTOP-53ONFTSV\...c2 - dbo.employee SQLQuery1.sql - L...3ONFTSV\Deep (62))*				
	empid	empname	empgender	empsalary
	1	deep	male	20000
	2	magan	male	15000
	3	daisy	female	25000
	4	emily	female	12000
	5	gopal	male	18000
»*	NULL	NULL	NULL	NULL

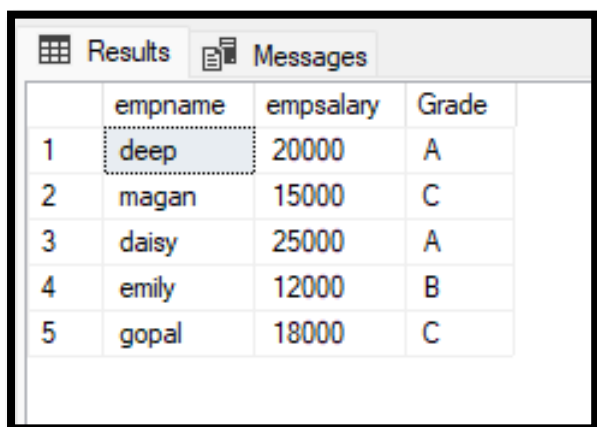
Step4: Write queries for retrieving data.

```
SELECT empname,empgender,  
case empgender  
when 'male' then 'handsome'  
when 'female' then 'beautiful'  
end as 'asdf'  
from dbo.employee
```



	empname	empgender	asdf
1	deep	male	handsome
2	magan	male	handsome
3	daisy	female	beautiful
4	emily	female	beautiful
5	gopal	male	handsome

```
SELECT empname,empsalary,  
case  
when empname like 'd%' then  
case  
when empsalary>=15000 then 'A'  
end  
when empname like 'e%' then  
case  
when empsalary>=12000 then 'B'  
end  
else 'C'  
end as "Grade"  
from dbo.employee
```



	empname	empsalary	Grade
1	deep	20000	A
2	magan	15000	C
3	daisy	25000	A
4	emily	12000	B
5	gopal	18000	C

Conclusion:

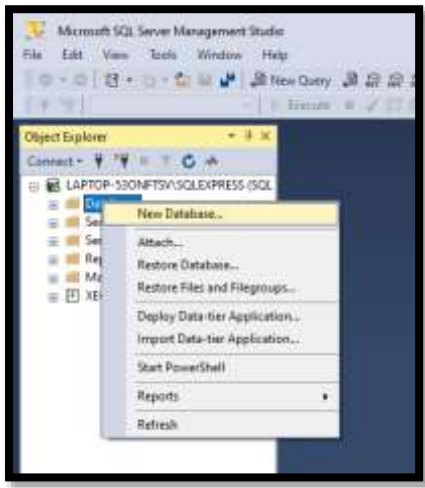
Successfully implemented SQL query to retrieve sensitive information from less sensitive queries.

Practical No 3

Aim: Write SQL query to create a view to implement concept of views and commutative filter in distributed databases.

Steps:

Step1: Create new database.



Step2: Create tables (employees,products,sales) .

```
Create table employees(  
employeeid int not null,  
name nvarchar(50) not null,  
title nvarchar(100) not null,  
hiredate datetime not null,  
vacationhours smallint not null,  
salary decimal(19,4) not null  
);
```

```
Create table products(  
productid int not null,  
productname nvarchar(255) not null,  
price decimal(19,4) not null  
);
```

```
create table sales(  
salesid int not null,  
productid int not null,  
employeeid int not null,  
quantity smallint not null,  
saledate datetime not null  
);
```

Step3: Insert values in tables.

LAPTOP-53ONFTSV\...3 - dbo.employees						
	employeeid	name	title	hiredate	vacationhours	salary
	1	deep	senior	2023-01-01 00:0...	10	20000.0000
	2	magan	junior	2023-01-10 00:0...	15	15000.0000
	3	shyam	senior	2023-02-15 00:0...	5	25000.0000
	4	chagan	intern	2023-03-01 00:0...	5	10000.0000
▶*	NULL	NULL	NULL	NULL	NULL	NULL

LAPTOP-53ONFTSV\...c3 - dbo.products			
	productid	productname	price
	1	shirt	400.0000
	2	jeans	600.0000
	3	trackpant	500.0000
	4	jacket	1000.0000
▶*	NULL	NULL	NULL

LAPTOP-53ONFTSV\S...prac3 - dbo.sales					
	salesid	productid	employeeid	quantity	saledate
	1	1	1	10	2023-01-01 00:0...
	2	2	2	5	2023-01-01 00:0...
	3	3	3	8	2023-01-01 00:0...
	4	4	4	6	2023-01-01 00:0...
▶*	NULL	NULL	NULL	NULL	NULL

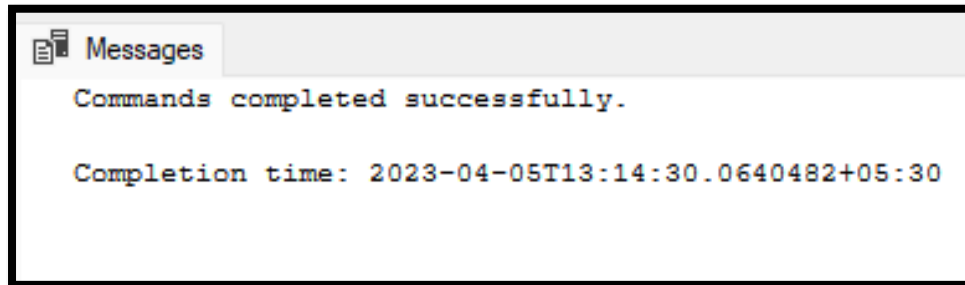
Step4: Print employee table.

```
select * from dbo.employees;
```

Results		Messages				
	employeeid	name	title	hiredate	vacationhours	salary
1	1	deep	senior	2023-01-01 00:00:00.000	10	20000.0000
2	2	magan	junior	2023-01-10 00:00:00.000	15	15000.0000
3	3	shyam	senior	2023-02-15 00:00:00.000	5	25000.0000
4	4	chagan	intern	2023-03-01 00:00:00.000	5	10000.0000

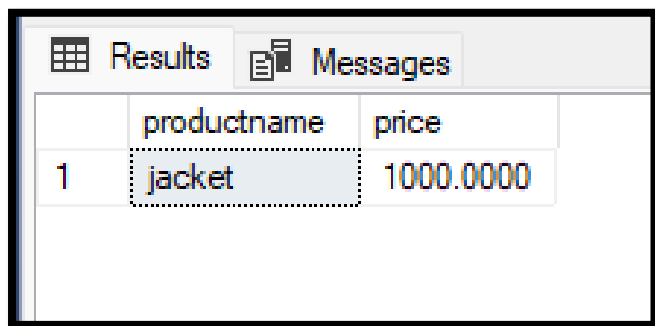
Step5: Create View.

```
create view [ProductsAboveAveragePrice] as  
select productname,price  
from products  
where price>(select avg(price) from products)
```



Step5: Print View.

```
select * from ProductsAboveAveragePrice;
```

A screenshot of the SQL Server Results window. The window has two tabs: "Results" and "Messages". The "Results" tab is active, showing a table with two columns: "productname" and "price". There is one row of data with "jacket" in the "productname" column and "1000.0000" in the "price" column.

	productname	price
1	jacket	1000.0000

Conclusion:

Successfully implemented SQL query to create a view to implement concept of views and commutative filter in distributed databases.

Practical No 4

Aim: Write a program to implement SSL.

Theory:

SSL:

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

In order to provide a high degree of privacy, SSL encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt.

A website that implements SSL/TLS has "HTTPS" in its URL instead of "HTTP."

Steps:

Step1: start netbeans → new project → java web → java webapplication → click on next → click on finish.

Step2: create class server file

Program:

ClassServer.java

```
import java.io.*;
import java.net.*;
import javax.net.*;

public abstract class ClassServer implements Runnable {
    private ServerSocket server = null;

    protected ClassServer(ServerSocket ss)
    {
        server = ss;
    }
}
```

```
newListener();
}
public abstract byte[] getBytes(String path)
throws IOException, FileNotFoundException;
public void run()
{
    Socket socket;
    try {
        socket = server.accept();
    } catch (IOException e) {
        System.out.println("Class Server died: " + e.getMessage());
        e.printStackTrace();
    }
    return;
}
newListener();
try {
    OutputStream rawOut = socket.getOutputStream();
    PrintWriter out = new PrintWriter(
        new BufferedWriter(
            new OutputStreamWriter(
                rawOut)));
    try {
        // get path to class file from header
        BufferedReader in =
            new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
        String path = getPath(in);
        byte[] bytcodes = getBytes(path);
        try {
```

```
out.print("HTTP/1.0 200 OK\r\n");

out.print("Content-Length: " + bytecodes.length +
"\r\n");

out.print("Content-Type: text/html\r\n\r\n");

out.flush();

rawOut.write(bytecodes);

rawOut.flush();

} catch (IOException ie) {
ie.printStackTrace();

return;

}

} catch (Exception e) {
e.printStackTrace();

// write out error response
out.println("HTTP/1.0 400 " + e.getMessage() + "\r\n");

out.println("Content-Type: text/html\r\n\r\n");

out.flush();

}

} catch (IOException ex) {

System.out.println("error writing response: " + ex.getMessage());

ex.printStackTrace();

} finally {

try {

socket.close();

} catch (IOException e) {

}

}

}

private void newListener()
```

```
{
(new Thread(this)).start();
}

private static String getPath(BufferedReader in)
throws IOException
{
String line = in.readLine();
String path = "";

// extract class from GET line
if (line.startsWith("GET /")) {
line = line.substring(5, line.length()-1).trim();
int index = line.indexOf(' ');
if (index != -1) {
path = line.substring(0, index);
}
}

// eat the rest of header
do {
line = in.readLine();
} while ((line.length() != 0) &&
(line.charAt(0) != '\r') && (line.charAt(0) != '\n'));
if (path.length() != 0) {
return path;
} else {
throw new IOException("Malformed Header");
}
}
}
```

Index.html

```
<!DOCTYPE html>

<html>

<body>

<form>

<div class="container">

<label>Username : </label>

<input type="text" placeholder="Enter Username" name="username" required>

<br><br>

<label>Password : </label>

<input type="password" placeholder="Enter Password" name="password" required>

<button type="submit">Login</button>

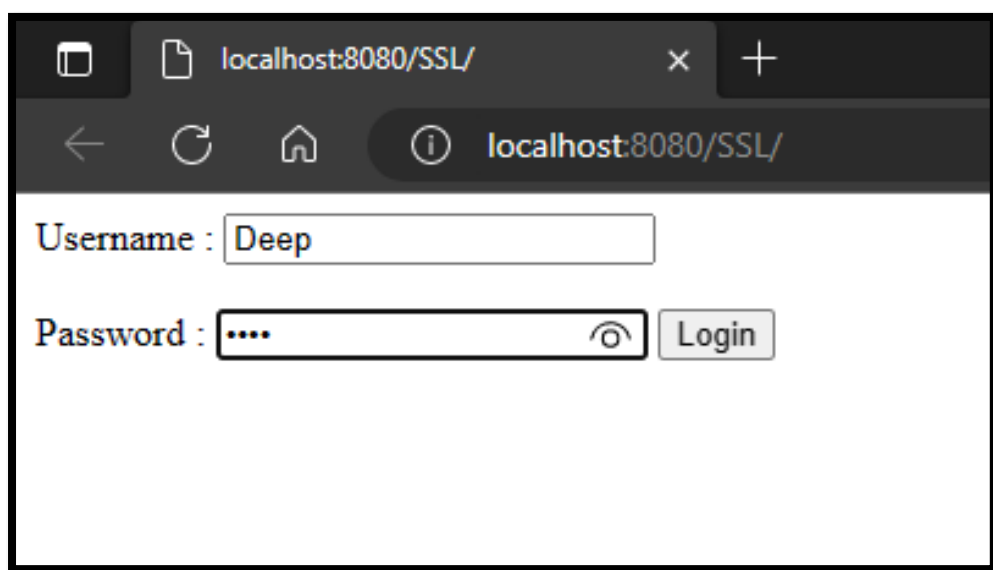
</div>

</form>

</body>

</html>
```

Output:



Conclusion:

Successfully implemented a program to implement SSL.

Practical No 6

Aim: Write a program to digitally sign MIME to create an opaque signature.

Theory:

Opaque signature:

An opaque-signed message or encrypted message in the Internet e-mail message format is identified as a MIME message that consists of exactly one MIME entity. The MIME entity usually has the media type "application/pkcs7-mime" or "application/x-pkcs7-mime". Note, however, that it can alternatively have the media type "application/octet-stream" if a file name, identified by the Content-Type header field, as specified in [RFC2045] section 5, or the Content-Disposition header field, as specified in [RFC2183], has the file extension ".p7m". The content of the MIME body is a Cryptographic Message Syntax (CMS) encapsulation of protected message content, together with all necessary cryptographic metadata.

Steps:

New project → java → java application → next → project name → Finish.

Program:

```
package cyber_pracs;

import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;

public class Cyber_prac6 {
    public static void main(String args[]) throws Exception {
        //Accepting text from user
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter some text");
        String msg = sc.nextLine();
        //Creating KeyPair generator object
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");
        //Initializing the key pair generator
        keyPairGen.initialize(2048);
```

```
//Generate the pair of keys
KeyPair pair = keyPairGen.generateKeyPair();
//Getting the private key from the key pair
PrivateKey privKey = pair.getPrivate();
//Creating a Signature object
Signature sign = Signature.getInstance("SHA256withDSA");
//Initialize the signature
sign.initSign(privKey);
byte[] bytes = "msg".getBytes();
//Adding data to the signature
sign.update(bytes);
//Calculating the signature
byte[] signature = sign.sign();
//Printing the signature
System.out.println("Digital signature for given text: "+new String(signature,
"UTF8"));
}
}
```

Output:

[illegible]

Conclusion:

Successfully implemented a program to digitally sign MIME to create an opaque signature.

Practical No 7

Aim: Write a program to generate DSA SSH key.

Theory:

DSA:

DSA (Digital Signature Algorithm) is also an asymmetric-key encryption algorithm which came much later than RSA. Since its development in 1991, the National Institute of Standards and Technology pushed to adopt the algorithm on a federal level. Despite the widespread popularity of the RSA algorithm in the private sector, DSA became the standard for a lot of US government agencies.

DSA only works with a safer, second edition of the Secure Shell (SSH) network protocol.

DSA is a stream cipher.

RSA and DSA are both used for the same internet protocols and certificates, like Nettle, OpenSSL, wolfCrypt, Crypto++, and cryptlib.

Steps:

New project → java → java application → next → project name → Finish.

Program:

```
package cyber_pracs;

import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;

public class Cyber_prac7 {

    public static void main(String[] args) {

        try {

            // Generate a 1024-bit Digital Signature Algorithm (DSA) key pair
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
```



```

keyGen.initialize(1024);

KeyPair keypair = keyGen.genKeyPair();

PrivateKey privateKey = keypair.getPrivate();

PublicKey publicKey = keypair.getPublic();

System.out.println(privateKey + "\n" + publicKey);

// Generate a 576-bit DH key pair

keyGen = KeyPairGenerator.getInstance("DH");

keyGen.initialize(576);

keypair = keyGen.genKeyPair();

privateKey = keypair.getPrivate();

publicKey = keypair.getPublic();

System.out.println(privateKey + "\n" + publicKey);

// Generate a 1024-bit RSA key pair

keyGen = KeyPairGenerator.getInstance("RSA");

keyGen.initialize(1024);

keypair = keyGen.genKeyPair();

privateKey = keypair.getPrivate();

publicKey = keypair.getPublic();

System.out.println(privateKey + "\n" + publicKey);

} catch (java.security.NoSuchAlgorithmException e) {

}}

```

Output:

```

SunRsaSign RSA private CRT key, 1024 bits
params: null
modulus: 932052522913842067158422443193670591455906772450581143692125873462891277300520239
private exponent: 126716022691950086491318552403308130824910040779020675195642790065983509
params: null
modulus: 932052522913842067158422443193670591455906772450581143692125873462891277300520239
public exponent: 65537
BUILD SUCCESSFUL (total time: 0 seconds)

```

Conclusion:

Successfully implemented a program to generate DSA SSH key.

Practical No 8

Aim: Write a program to implement multilevel security.

Theory:

Multilevel Security:

Multilevel security is a security policy that allows the classification of data and users based on a system of hierarchical security levels combined with a system of non-hierarchical security categories. A multilevel-secure security policy has two primary goals. First, the controls must prevent unauthorized individuals from accessing information at a higher classification than their authorization. Second, the controls must prevent individuals from declassifying information.

Steps:

New project → java → java application → next → project name → Finish.

Program:

```
package cyber_pracs;

class Electronics {

public Electronics(){

System.out.println("Class Electronics");

}

public void deviceType() {

System.out.println("Device Type: Electronics");

}

}

class Television extends Electronics {

public Television() {

System.out.println("Class Television");

}

public void category() {
```

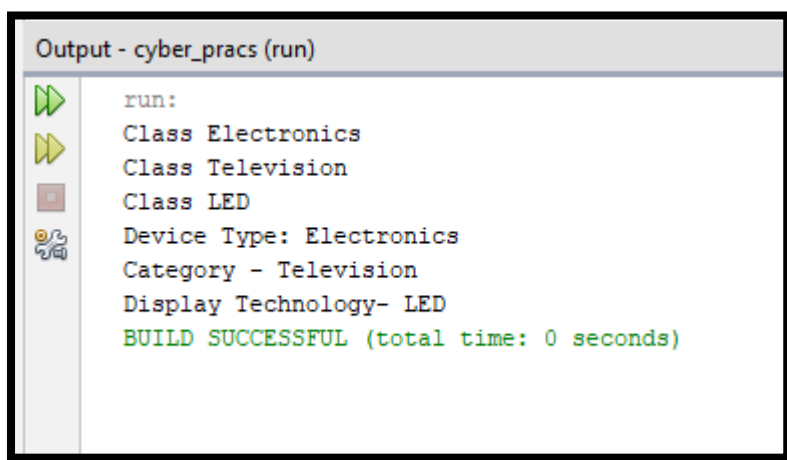
```
System.out.println("Category - Television");
}
}

class LED extends Television {
public LED() {
System.out.println("Class LED");
}

public void display_tech() {
System.out.println("Display Technology- LED");
}
}

public class Tester {
public static void main(String[] arguments) {
LED led = new LED();
led.deviceType();
led.category();
led.display_tech();
}
}
```

Output:



```
Output - cyber_pracs (run)
run:
Class Electronics
Class Television
Class LED
Device Type: Electronics
Category - Television
Display Technology- LED
BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusion:

Successfully implemented a program to implement multilevel security.

Practical No 9

Aim: Write a program to Demonstrates how to encrypt and decrypt the content of an XML node using 128-bit CBC AES encryption.

Theory:

AES:

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember:

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

Steps:

New project → java → java application → next → project name → Finish.

Program:

```
package cyber_pracs;

import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class Cyber_prac9 {

    private static final String encryptionKey = "ABCDEFGHJKLMNOP";
```

```

private static final String characterEncoding = "UTF-8";

private static final String cipherTransformation = "AES/CBC/PKCS5PADDING";

private static final String aesEncryptionAlgorithem = "AES";

/**
 * Method for Encrypt Plain String Data
 *
 * @param plainText
 * @return encryptedText
 */

public static String encrypt(String plainText) {
    String encryptedText = "";

    try {
        Cipher cipher = Cipher.getInstance(cipherTransformation);
        byte[] key = encryptionKey.getBytes(characterEncoding);
        SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithem);
        IvParameterSpec ivparameterspec = new IvParameterSpec(key);
        cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivparameterspec);
        byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF8"));
        Base64.Encoder encoder = Base64.getEncoder();
        encryptedText = encoder.encodeToString(cipherText);
    } catch (Exception E) {
        System.err.println("Encrypt Exception : "+E.getMessage());
    }

    return encryptedText;
}

/**
 * Method For Get encryptedText and Decrypted provided String
 *
 * @param encryptedText
 * @return decryptedText
 */

public static String decrypt(String encryptedText) {
    String decryptedText = "";

    try {
        Cipher cipher = Cipher.getInstance(cipherTransformation);

```

```

byte[] key = encryptionKey.getBytes(characterEncoding);

SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithm);

IvParameterSpec ivparameterspec = new IvParameterSpec(key);

cipher.init(Cipher.DECRYPT_MODE, secretKey, ivparameterspec);

Base64.Decoder decoder = Base64.getDecoder();

byte[] cipherText = decoder.decode(encryptedText.getBytes("UTF8"));

decryptedText = new String(cipher.doFinal(cipherText), "UTF-8");

} catch (Exception E) {

System.err.println("decrypt Exception : "+E.getMessage());

}

return decryptedText;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter String : ");

String plainString = sc.nextLine();

String encryptStr = encrypt(plainString);

String decryptStr = decrypt(encryptStr);

System.out.println("Plain String : "+plainString);

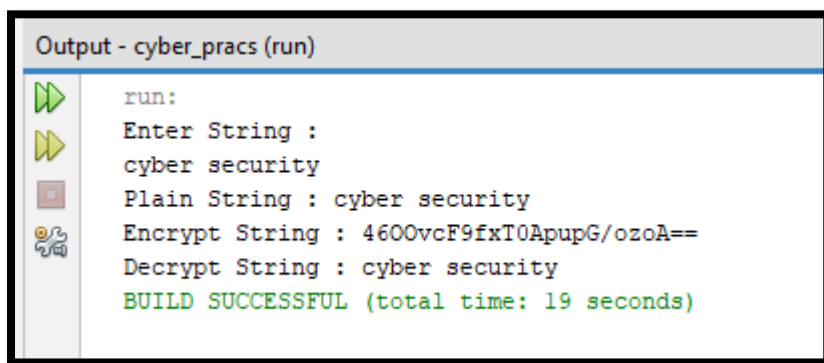
System.out.println("Encrypt String : "+encryptStr);

System.out.println("Decrypt String : "+decryptStr);

}}

```

Output:



```

Output - cyber_pracs (run)
run:
Enter String :
cyber security
Plain String : cyber security
Encrypt String : 4600vcF9fxT0ApupG/ozoA==
Decrypt String : cyber security
BUILD SUCCESSFUL (total time: 19 seconds)

```

Conclusion:

Successfully implemented a program to Demonstrates how to encrypt and decrypt the content of an XML node using 128-bit CBC AES encryption.