

WEEK-06

Name	N.Aasrith
Roll No.	22R21A05P6
Year of Study	IV B.Tech I SEM CSE-D
Date	16-09-2025

PROBLEM STATEMENT:

Write a C/JAVA program to implement the Rijndael algorithm logic

PROGRAM:

```
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;

public class AES {
    public static String asHex(byte[] buf) {
        StringBuilder strbuf = new StringBuilder(buf.length * 2);
        for (int i = 0; i < buf.length; i++) {
            if ((buf[i] & 0xff) < 0x10) {
                strbuf.append("0");
            }
            strbuf.append(Integer.toHexString(buf[i] & 0xff));
        }
        return strbuf.toString();
    }

    public static void main(String[] args) throws Exception {
        String message = "cryptography and networksecurity";
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        kgen.init(128);
        SecretKey skey = kgen.generateKey();
        byte[] raw = skey.getEncoded();
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(message.getBytes());
        System.out.println("Encrypted string: " + asHex(encrypted));
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] original = cipher.doFinal(encrypted);
        String originalString = new String(original);
        System.out.println("Original string: " + originalString);
    }
}
```

OUTPUT:

```
Encrypted string: bc6df72345edd2a8591a289dbe5642b4b3c04640a70aaed99d59d3a6a696580c7cdb6
3c3a0be66c253bb10ebcb81a6e1
Original string: cryptography and networksecurity
```

```
Input your message: cryptography and network security
Encrypted text (bytes): :d`0x>0000002f0000002D00sRn0
Decrypted text: cryptography and network security
```

WEEK-08

Name	N.Aasrith
Roll No.	22R21A05P6
Year of Study	IV B.Tech I SEM CSE-D
Date	07-10-2025

PROBLEM STATEMENT:

Write a Java program to implement RSA algorithm.

PROGRAM:

```
import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;
public class RSA {
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.print("Enter a prime number (p): ");
        BigInteger p = sc.nextBigInteger();
        System.out.print("Enter another prime number (q): ");
        BigInteger q = sc.nextBigInteger();
        BigInteger n = p.multiply(q);
        BigInteger phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        BigInteger e = generateE(phi);
        BigInteger d = e.modInverse(phi);
        System.out.println("Public Key (e, n): (" + e + ", " + n + ")");
        System.out.println("Private Key (d, n): (" + d + ", " + n + ")");
    }
    public static BigInteger generateE(BigInteger phi) {
        Random rand = new Random();
        BigInteger e;
        do {
            e = new BigInteger(phi.bitLength(), rand);
        } while (e.compareTo(BigInteger.TWO) <= 0 || !phi.gcd(e).equals(BigInteger.ONE));
        return e;
    }
}
```

OUTPUT:

```
Enter a prime number (p): 5
Enter another prime number (q): 11
Public Key (e, n): (41, 55)
Private Key (d, n): (1, 55)
```

WEEK-09

Name	N.Aasrith
Roll No.	22R21A05P6
Year of Study	IV B.Tech I SEM CSE-D
Date	14-10-2025

PROBLEM STATEMENT:

Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript

PROGRAM:

```
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
import javax.crypto.spec.DHParameterSpec;
import javax.crypto.spec.DHPublicKeySpec;

public class DiffieHellman {

    public final static int pValue= 47;
    public final static int gValue=71;
    public final static int XaValue=9;
    public final static int XbValue=14;
    public static void main(String[] args) throws Exception {
        BigInteger p=new BigInteger(Integer.toString(pValue));
        BigInteger g=new BigInteger(Integer.toString(gValue));
        BigInteger Xa=new BigInteger(Integer.toString(XaValue));
        BigInteger Xb=new BigInteger(Integer.toString(XbValue));
        createKey();
        int bitLength=512;
        SecureRandom rnd= new SecureRandom();
        p=BigInteger.probablePrime(bitLength, rnd);
        g= BigInteger.probablePrime(bitLength, rnd);
        createSpecificKey(p, g);
    }
    public static void createKey() throws Exception {
        KeyPairGenerator kpg=KeyPairGenerator.getInstance("DiffieHellman");
        kpg.initialize(512);
        KeyPair kp=kpg.generateKeyPair();
        KeyFactory kfactory=KeyFactory.getInstance("DiffieHellman");
        DHPublicKeySpec kspec=(DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),
        DHPublicKeySpec.class);
        System.out.println("Public key is: " + kspec);
    }
    public static void createSpecificKey(BigInteger p, BigInteger g) throws Exception {
```

```
KeyPairGenerator kpg= KeyPairGenerator.getInstance("DiffieHellman");
DHParameterSpec param=new DHParameterSpec(p, g);
kpg.initialize(param);
KeyPair kp= kpg.generateKeyPair();
KeyFactory kfactory=KeyFactory.getInstance("DiffieHellman");
DHPublicKeySpec kspec=(DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),
DHPublicKeySpec.class);
    System.out.println("\nPublic key is : " + kspec);
}
}
```

OUTPUT:

```
Public key is: javax.crypto.spec.DHPublicKeySpec@49c2faae
Public key is : javax.crypto.spec.DHPublicKeySpec@1d56ce6a
```


WEEK-10

Name	N.Aasrith
Roll No.	22R21A05P6
Year of Study	IV B.Tech I SEM CSE-D
Date	28-10-2025

PROBLEM STATEMENT:

Calculate the message digest of a text using the SHA-1 algorithm in Java.

PROGRAM:

```
import java.security.MessageDigest;
public class SHA1 {
    public static void main(String[] args) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA1");
            System.out.println("Message digest object info:");
            System.out.println(" Algorithm = " + md.getAlgorithm());
            System.out.println(" Provider = " + md.getProvider());
            System.out.println(" ToString = " + md.toString());
            String input = "";
            md.update(input.getBytes());
            byte[] output= md.digest();
            System.out.println();
            System.out.println("SHA1 (\\"" + input + "\") = " + bytesToHex(output));
            input = "abc";
            md.update(input.getBytes());
            output= md.digest();
            System.out.println();
            System.out.println("SHA1 (\\"" + input + "\") = " + bytesToHex (output));
            input = "abcdefghijklmnopqrstuvwxy";
            md.update(input.getBytes()); output= md.digest();
            System.out.println();
            System.out.println("SHA1 (\\"" + input + "\") = " + bytesToHex (output));
            System.out.println();
        }
        catch (Exception e) {
            System.out.println("Exception: " + e);
        }
    }
    public static String bytesToHex(byte[] b) {
        char[] hexDigit = {
            '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
        StringBuffer buf = new StringBuffer();
        for (int j = 0; j < b.length; j++) {
```

```
        buf.append(hexDigit[(b[j] >> 4) & 0x0f]);  
        buf.append(hexDigit [b[j] & 0x0f]);  
    }  
    return buf.toString();  
}  
}
```

OUTPUT:

```
Message digest object info:  
Algorithm = SHA1  
Provider = SUN version 23  
ToString = SHA1 Message Digest from SUN, <initialized>  
  
SHA1 ("") = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709  
  
SHA1 ("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D  
  
SHA1 ("abcdefghijklmnopqrstuvwxyz") = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89
```

WEEK-11

Name	N.Aasrith
Roll No.	22R21A05P6
Year of Study	IV B.Tech I SEM CSE-D
Date	04-11-2025

PROBLEM STATEMENT:

Calculate the message digest of a text using the MD5 algorithm in Java.

PROGRAM:

```
import java.security.MessageDigest;
public class MD5 {
    public static void main(String[] args) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            System.out.println("Message digest object info:");
            System.out.println(" Algorithm = "+md.getAlgorithm());
            System.out.println(" Provider = " + md.getProvider());
            System.out.println(" ToString = " + md.toString());
            String input = ""; md.update(input.getBytes());
            byte[] output= md.digest(); System.out.println();
            System.out.println("MD5(\"\" + input + "\"") = " + bytesToHex (output));
            input = "abc";
            md.update(input.getBytes());
            output= md.digest(); System.out.println();
            System.out.println("MD5(\"\" + input + "\"") = " + bytesToHex (output));
            input = "abcdefghijklmnopqrstuvwxy";
            md.update(input.getBytes());
            output= md.digest();
            System.out.println();
            System.out.println("MD5(\"\" + input + "\"") = " + bytesToHex (output));
            System.out.println();
        }
        catch (Exception e) {
            System.out.println("Exception: " + e);
        }
    }
    public static String bytesToHex (byte[] b) {
        char[] hexDigit = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
        StringBuffer buf = new StringBuffer();
        for (int j = 0; j < b.length; j++) {
            buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
            buf.append(hexDigit[b[j] & 0x0f]);
        }
    }
}
```



```
        return buf.toString();  
    }  
}
```

OUTPUT:

```
Message digest object info:  
Algorithm = MD5  
Provider = SUN version 23  
ToString = MD5 Message Digest from SUN, <initialized>  
  
MD5("") = D41D8CD98F00B204E9800998ECF8427E  
  
MD5("abc") = 900150983CD24FB0D6963F7D28E17F72  
  
MD5("abcdefghijklmnopqrstuvwxyz") = C3FCD3D76192E4007DFB496CCA67E13B
```