Program:

```
//RSA
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA{
   private BigInteger p;
   private BigInteger q;
   private BigInteger N;
   private BigInteger phi;
   private BigInteger e;
   private BigInteger d;
   private int bitlength = 1024;
   private Random r;
   public RSA(){
      r = new Random();
      p = BigInteger.probablePrime(bitlength, r);
      q = BigInteger.probablePrime(bitlength, r);
      N = p.multiply(q);
      phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
      e = BigInteger.probablePrime(bitlength / 2, r);
      while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0){ e.add(BigInteger.ONE);  }
      d = e.modInverse(phi);
   }
   public RSA(BigInteger e, BigInteger d, BigInteger N){
      this.e = e;
      this.d = d;
      this.N = N;
   }
   @SuppressWarnings("deprecation")
   public static void main(String[] args) throws IOException{
      RSA rsa = new RSA();
      DataInputStream in = new DataInputStream(System.in);
      String teststring;
      System.out.println("Enter the plain text:");
      teststring = in.readLine();
      System.out.println("String: " + teststring);
      System.out.println("After Encryption: "+ bytesToString(teststring.getBytes()));
      byte[] encrypted = rsa.encrypt(teststring.getBytes());
      byte[] decrypted = rsa.decrypt(encrypted);
      System.out.println("After Decryption: " + new String(decrypted));
   }

   private static String bytesToString(byte[] encrypted){
      String test = "";
      for (byte b : encrypted){ test += Byte.toString(b); }
      return test;
```

```
    }
    public byte[] encrypt(byte[] message){  return (new BigInteger(message)).modPow(e, N).toByteArray();  }
    public byte[] decrypt(byte[] message){  return (new BigInteger(message)).modPow(d, N).toByteArray();  }
}
```

Output:

```
xclusiv@pop-os:~/Documents/ccns$ java RSA
Enter the plain text:
Rahul
String: Rahul
After Encryption: 8297104117108
After Decryption: Rahul
xclusiv@pop-os:~/Documents/ccns$
```

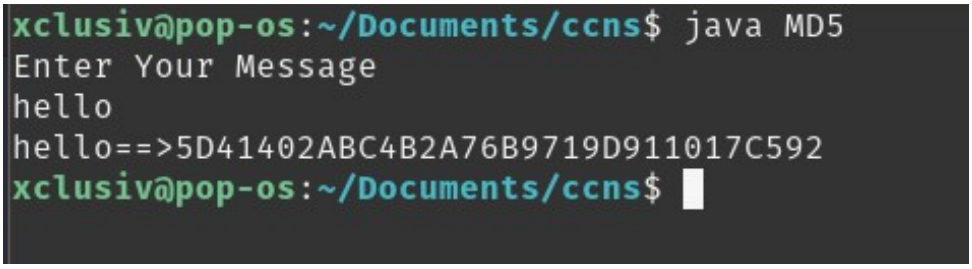
Conclusion:

Program

```
//MD5
import java.util.Scanner;
public class MD5{
  private static final int INIT_A = 0x67452301;
  private static final int INIT_B = (int) 0xEFCDAB89L;
  private static final int INIT_C = (int) 0x98BADCFEL;
  private static final int INIT_D = 0x10325476;
  private static final int[] SHIFT_AMTS ={7, 12, 17, 22, 5, 9, 14, 20, 4,11, 16, 23, 6, 10, 15, 21};
  private static final int[] TABLE_T = new int[64];
  static{ for(int i =0; i <64; i++)
            TABLE_T[i]=(int)(long)((1L <<32)*Math.abs(Math.sin(i +1)));}
  public static byte[] computeMD5(byte[] message){
    int messageLenBytes = message.length;
    int numBlocks =((messageLenBytes +8)>>>6)+1;
    int totalLen = numBlocks <<6;
    byte[] paddingBytes =new byte[totalLen - messageLenBytes];
    paddingBytes[0]=(byte) 0x80;
    long messageLenBits =(long) messageLenBytes <<3;
    for(int i =0; i <8; i++){
      paddingBytes[paddingBytes.length-8+ i]=(byte) messageLenBits;
      messageLenBits >>>=8;      }
    int a = INIT_A;
    int b = INIT_B;
    int c = INIT_C;
    int d = INIT_D;
    int[] buffer =new int[16];
    for(int i =0; i < numBlocks; i++){
      int index = i <<6;
      for(int j =0; j <64; j++, index++)
        buffer[j >>>2]=((int)((index < messageLenBytes)? message[index]: paddingBytes[index -
messageLenBytes])<<24)|(buffer[j >>>2]>>>8);
        int originalA = a;
        int originalB = b;
        int originalC = c;
        int originalD = d;
        for(int j =0; j <64; j++){
          int div16 = j >>>4;
          int f =0;
          int bufferIndex = j;
          switch(div16){
            case 0:
              f =(b & c)|(~b & d);
              break;
            case 1:
              f =(b & d)|(c & ~d);
              bufferIndex =(bufferIndex *5+1)& 0x0F;
              break;
```

```
          case 2:
            f = b ^ c ^ d;
            bufferIndex =(bufferIndex *3+5)& 0x0F;
            break;
          case 3:
            f = c ^(b | ~d);
            bufferIndex =(bufferIndex *7)& 0x0F;
            break;  }
  int temp = b+Integer.rotateLeft(a + f + buffer[bufferIndex]+ TABLE_T[j],SHIFT_AMTS[(div16 <<2)|(j
&3)]);
        a = d;
        d = c;
        c = b;
        b = temp;            }
      a += originalA;
      b += originalB;
      c += originalC;
      d += originalD;      }
    byte[] md5 =new byte[16];
    int count =0;
    for(int i =0; i <4; i++){
      int n =(i ==0)? a :((i ==1)? b :((i ==2)? c : d));
      for(int j =0; j <4; j++){
        md5[count++]=(byte) n;
        n >>>=8;             }          }
    return md5;              }
   public static String toHexString(byte[] b){
    StringBuilder sb =new StringBuilder();
    for(int i =0; i < b.length; i++){
      sb.append(String.format("%02X", b[i]& 0xFF));                }
    return sb.toString();            }
   public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter Your Message");
    String n = sc.next();
    String testStrings = n;
      System.out.println(testStrings +"==>"+toHexString(computeMD5(testStrings.getBytes())));            }      }
```

Output:



Conclusion:

Program:

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class DigitalSignature{
  //RSA
    private BigInteger p;
    private BigInteger q;
    private BigInteger N;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
    private int bitlength = 1024;
    private Random r;
    private static final int INIT_A = 0x67452301;
    private static final int INIT_B = (int) 0xEFCDAB89L;
    private static final int INIT_C = (int) 0x98BADCFEL;
    private static final int INIT_D = 0x10325476;
    private static final int[] SHIFT_AMTS ={7, 12, 17, 22, 5, 9, 14, 20, 4,11, 16, 23, 6, 10, 15, 21};
    private static final int[] TABLE_T = new int[64];
    static{
      for(int i =0; i <64; i++)
                TABLE_T[i]=(int)(long)((1L <<32)*Math.abs(Math.sin(i +1)));
    }
    public DigitalSignature(){
        r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        N = p.multiply(q);
        phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength / 2, r);
        while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0){ e.add(BigInteger.ONE);  }
        d = e.modInverse(phi);
    }
    public DigitalSignature(BigInteger e, BigInteger d, BigInteger N){
        this.e = e;
        this.d = d;
        this.N = N;
    }
    private static String bytesToString(byte[] encrypted){
        String test = "";
        for (byte b : encrypted){ test += Byte.toString(b); }
        return test;
    }
    public byte[] encrypt(byte[] message){  return (new BigInteger(message)).modPow(e, N).toByteArray();  }
    public byte[] decrypt(byte[] message){  return (new BigInteger(message)).modPow(d, N).toByteArray();  }
    //MD5
```

```
  public static byte[] computeMD5(byte[] message){
    int messageLenBytes = message.length;
    int numBlocks =((messageLenBytes +8)>>>6)+1;
    int totalLen = numBlocks <<6;
    byte[] paddingBytes =new byte[totalLen - messageLenBytes];
    paddingBytes[0]=(byte) 0x80;
    long messageLenBits =(long) messageLenBytes <<3;
    for(int i =0; i <8; i++){
      paddingBytes[paddingBytes.length-8+ i]=(byte) messageLenBits;
      messageLenBits >>>=8;
    }
    int a = INIT_A;
    int b = INIT_B;
    int c = INIT_C;
    int d = INIT_D;
    int[] buffer =new int[16];
    for(int i =0; i < numBlocks; i++){
      int index = i <<6;
      for(int j =0; j <64; j++, index++)
        buffer[j >>>2]=((int)((index < messageLenBytes)? message[index]: paddingBytes[index -
messageLenBytes])<<24)|(buffer[j >>>2]>>>8);
        int originalA = a;
        int originalB = b;
        int originalC = c;
        int originalD = d;
        for(int j =0; j <64; j++){
          int div16 = j >>>4;
          int f =0;
          int bufferIndex = j;
          switch(div16){
            case 0:
              f =(b & c)|(~b & d);
              break;
            case 1:
              f =(b & d)|(c & ~d);
              bufferIndex =(bufferIndex *5+1)& 0x0F;
              break;
            case 2:
              f = b ^ c ^ d;
              bufferIndex =(bufferIndex *3+5)& 0x0F;
              break;
            case 3:
              f = c ^(b | ~d);
              bufferIndex =(bufferIndex *7)& 0x0F;
              break;
          }
          int temp = b+Integer.rotateLeft(a + f + buffer[bufferIndex]+ TABLE_T[j],SHIFT_AMTS[(div16 <<2)|(j
&3)]);
```

```
          a = d;
          d = c;
          c = b;
          b = temp;
        }
        a += originalA;
        b += originalB;
        c += originalC;
        d += originalD;
      }
      byte[] md5 =new byte[16];
      int count =0;
      for(int i =0; i <4; i++){
        int n =(i ==0)? a :((i ==1)? b :((i ==2)? c : d));
        for(int j =0; j <4; j++){
          md5[count++]=(byte) n;
          n >>>=8;
        }
      }
      return md5;
    }
    public static String toHexString(byte[] b){
      StringBuilder sb =new StringBuilder();
      for(int i =0; i < b.length; i++){
        sb.append(String.format("%02X", b[i]& 0xFF));
      }
      return sb.toString();
    }
    public static void Sender(String teststring){
      System.out.println("String: " + teststring);
      String r = bytesToString(teststring.getBytes());
      String hash = toHexString(computeMD5(r.getBytes()));
      System.out.println("-----------Sender---------\nSignature sent is:"+hash+"\nData is:"+teststring);
      Receiver(teststring, hash);
    }
    public static void Receiver(String teststring, String hash){
      String r = bytesToString(teststring.getBytes());
      String newHash = toHexString(computeMD5(r.getBytes()));
      System.out.println("-----------Receiver---------\nSignature received is:"+hash+"\nSignature generated to
verify is:"+newHash);
      if(hash == newHash)
        System.out.println("Data verified!");
      else
        System.out.println("Data issues");
    }
    @SuppressWarnings("deprecation")
    public static void main(String[] args) throws IOException{
        DigitalSignature rsa = new DigitalSignature();
```

```
        DataInputStream in = new DataInputStream(System.in);
        String teststring;
        System.out.println("Signature-Based Detection using RSA and MD5");
        System.out.println("Enter the plain text:");
        teststring = in.readLine();
        Sender(teststring);
    }
}
```

Output:

```
xclusiv@pop-os:~/Documents/ccns$ java DigitalSignature
Signature-Based Detection using RSA and MD5
Enter the plain text:
uiregehriuh
String: uiregehriuh
-----------Sender---------
Signature sent is:595AD000604289B1CCA7CA42E1620A88
Data is:uiregehriuh
-----------Receiver---------
Signature received is:595AD000604289B1CCA7CA42E1620A88
Signature generated to verify is:595AD000604289B1CCA7CA42E1620A88
Data verified!
xclusiv@pop-os:~/Documents/ccns$
```

Conclusion:

Program:

```
import java.util.Scanner;
class CaesarC{
	void Caesar(String message, int key){
		String decryptedMessage = "";
		char ch;
		Scanner sc = new Scanner(System.in);
		for(int i = 0; i < message.length(); ++i){
			ch = message.charAt(i);
			if(ch >= 'a' && ch <= 'z'){
		      ch = (char)(ch - key);
		      if(ch < 'a'){	ch = (char)(ch + 'z' - 'a' + 1);	}
		      decryptedMessage += ch;
		    }
		    else if(ch >= 'A' && ch <= 'Z'){
		      ch = (char)(ch - key);
		      if(ch < 'A'){	ch = (char)(ch + 'Z' - 'A' + 1);	}
		      decryptedMessage += ch;
		    }
		    else {	decryptedMessage += ch;		}
		    }
		System.out.println("Decrypted Message (key = "+key+") = " + decryptedMessage);
	}
}
class VigenereCipher{
	public static String encrypt(String text, final String key){
		String res = "";
		text = text.toUpperCase();
		for (int i = 0, j = 0; i < text.length(); i++){
			char c = text.charAt(i);
			if (c < 'A' || c > 'Z')
				continue;
			res += (char) ((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
			j = ++j % key.length();
		}
		return res;
	}
	public static String decrypt(String text, final String key){
		String res = "";
		text = text.toUpperCase();
		for (int i = 0, j = 0; i < text.length(); i++){
			char c = text.charAt(i);
			if (c < 'A' || c > 'Z')
				continue;
			res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');
			j = ++j % key.length();
		}
		return res;	}	}
```

```
public class Cryptanalysis{
   public static void main(String[] args){
        //CAESARCIPHER
        System.out.println("Enter your message");
        Scanner sc = new Scanner(System.in);
        String message = sc.next();
        CaesarC c = new CaesarC();
        VigenereCipher v = new VigenereCipher();
        int keyi= 0;
        System.out.println("-------Caesar Cipher--------");
        for(keyi=0;keyi<26;keyi++)
        c.Caesar(message,keyi);
        //VIGENERECIPHER
        System.out.println("-------Vigenere Cipher--------");
        String key = "VIGENERECIPHER";
        String encryptedMsg = v.encrypt(message, key);
        System.out.println("String: " + message);
        System.out.println("Decrypted message: " + v.decrypt(message, key));
        //MULTIPLICATIVECIPHER
        System.out.println("-------Multiplicative Cipher--------");
        int shift,i,n;
        String str;
        String str1="",str2="";
        str=message.toLowerCase();
        n=str.length();
        char ch1[]=str.toCharArray();
        char ch3, ch4;
        System.out.println("Enter the value by which each letter of the string is to be shifted");
        shift=sc.nextInt();
        System.out.println();
        System.out.println("Decrypted text is");
        for(i=0;i<n;i++){
                if(Character.isLetter(ch1[i])){
                        ch3=(char)(((int)ch1[i]*shift-97)%26+97);
                        str1=str1+ch3;
                }
                else if(ch1[i]==' '){        str1=str1+ch1[i];           }
        }
        System.out.println(str1);
        int q=0,flag=0;
        for(i=0;i<26;i++){
        if(((i*26)+1)%shift==0){
                q=((i*26)+1)/shift;
                break;   }          }
System.out.println(str2);           }          }
```

Output:

```
xclusiv@pop-os:~/Documents/ccns$ java Cryptanalysis
Enter your message
ghmnnmjhfd
-------Caesar Cipher--------
Decrypted Message (key = 0) = ghmnnmjhfd
Decrypted Message (key = 1) = fglmmligec
Decrypted Message (key = 2) = efkllkhfdb
Decrypted Message (key = 3) = dejkkjgeca
Decrypted Message (key = 4) = cdijjifdbz
Decrypted Message (key = 5) = bchiihecay
Decrypted Message (key = 6) = abghhgdbzx
Decrypted Message (key = 7) = zafggfcayw
Decrypted Message (key = 8) = yzeffebzxv
Decrypted Message (key = 9) = xydeedaywu
Decrypted Message (key = 10) = wxcddczxvt
Decrypted Message (key = 11) = vwbccbywus
Decrypted Message (key = 12) = uvabbaxvtr
Decrypted Message (key = 13) = tuzaazwusq
Decrypted Message (key = 14) = styzzyvtrp
Decrypted Message (key = 15) = rsxyyxusqo
Decrypted Message (key = 16) = qrwxxwtrpn
Decrypted Message (key = 17) = pqvwwvsqom
Decrypted Message (key = 18) = opuvvurpnl
Decrypted Message (key = 19) = notuutqomk
Decrypted Message (key = 20) = mnsttspnlj
Decrypted Message (key = 21) = lmrssromki
Decrypted Message (key = 22) = klqrrqnljh
Decrypted Message (key = 23) = jkpqqpmkig
Decrypted Message (key = 24) = ijoppoljhf
Decrypted Message (key = 25) = hinoonkige
-------Vigenere Cipher--------
String: ghmnnmjhfd
Decrypted message: LZGJAISDDV
-------Multiplicative Cipher--------
Enter the value by which each letter of the string is to be shifted
7

Decrypted text is
ahqxxqvhtf

xclusiv@pop-os:~/Documents/ccns$
```

Conclusion: