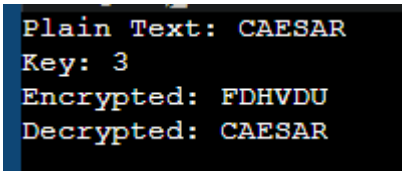
Caesar:

```
import java.util.*;

public class Main
{
    static String enc(String ip, int k) {
        String op = "";
        for(int i = 0; i < ip.length(); i++) {
            int c_INT = (ip.charAt(i) - 'A' + k)%26 + 'A';
            op += (char)c_INT;
        }
        return op;
    }

    static String dec(String ip, int k) {
        String op = "";
        for(int i = 0; i < ip.length(); i++) {
            int c_INT = (ip.charAt(i) - 'A' + 26 - k)%26 + 'A';
            op += (char)c_INT;
        }
        return op;
    }

            public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                        System.out.print("Plain Text & key: ");
                        String pText = sc.nextLine(); int k = sc.nextInt();
                        String cText = enc(pText, k);
                        System.out.println("Encrypted: "+ cText);
                        System.out.println("Decrypted: "+ dec(cText, k));
            }
}
```



PlayFair:

```
import java.util.*;

public class Main
{   static ArrayList<Character> M = new ArrayList<Character>();
    static char[][] kMat = new char[5][5];

    static void setMatrix(String k) {
        String letters = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
        for(char c : (k+letters).toCharArray()) {
            if(!M.contains(c) && c != 'J')
                M.add(c);
        }

        System.out.println("Key Matrix:");
        int i=0,j=0,c=0;
        for(i = 0; i < 5; i++) {
            for(j = 0; j < 5; j++, c++) {
                kMat[i][j] = M.get(c).charValue();
                System.out.print(kMat[i][j] + " ");
            }
            System.out.println("");
        }
    }



    static String playFair(String ip, int enc) {
        String dg = "", op = "";
        for(int i = 0; i < ip.length(); i++) {
            if(i+1 < ip.length() && ip.charAt(i) != ip.charAt(i+1)) {
                dg += ip.charAt(i);
                dg += ip.charAt(i+1);
                i++;
            } else {
                dg += ip.charAt(i);
                dg += 'X';
            }
        }
```

```
        int sh = (enc==1)? 1:4; //shift
        for(int i = 0; i < dg.length()-1; i+=2) {
            char c1 = dg.charAt(i), c2 = dg.charAt(i+1);
            int id1 = M.indexOf(c1), id2 = M.indexOf(c2);
            int i1 = id1/5, j1 = id1%5, i2 = id2/5, j2 = id2%5;
            if(i1 == i2) {
                op += kMat[i1][(j1+sh)%5];
                op += kMat[i1][(j2+sh)%5];
            } else if (j1 == j2) {
                op += kMat[(i1+sh)%5][j1];
                op += kMat[(i2+sh)%5][j1];
            } else {
                op += kMat[i1][j2];
                op += kMat[i2][j1];
            }
        }

        return op;
    }

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
                System.out.print("Plain Text: ");   String pText = sc.nextLine();
                System.out.print("Key: ");          String k = sc.nextLine();
                setMatrix(k);
                String cText = playFair(pText, 1);
                System.out.println("Encrypted: "+ cText);
                System.out.println("Decrypted: "+ playFair(cText, 0));
        }
}
```

```
Plain Text: COMMUNICATE
Key: COMPUTER
Key Matrix:
C O M P U
T E R A B
D F G H I
K L N Q S
V W X Y Z
Encrypted: OMRMPCSGPTER
Decrypted: COMXMUNICATE
```

Hill Cipher

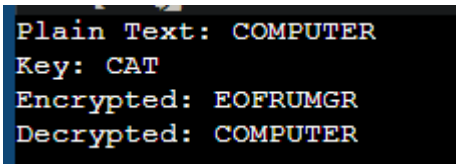
```
import java.util.*;
import java.lang.Math;

public class Main {

    static String hillCipher(String ip, int[][] K) {
        int n = (ip.length()-1)/2 + 1;
        int [][] cols = new int[n][2];
        String op = "";

        for(int i = 0, c=0; i < n; i++, c+=2) {
            if(c+1 < ip.length()) {
                cols[i][0] = ip.charAt(c) - 'A';
                cols[i][1] = ip.charAt(c+1) - 'A';
            } else {
                cols[i][0] = ip.charAt(c) - 'A';
                cols[i][1] = 'Z' - 'A';
            }
        }

        System.out.print("Pre-Op Matrices: ");
        for(int i = 0; i < n; i++)
            System.out.print(cols[i][0] + " " + cols[i][1] + "\t");

        System.out.print("\nPostOp Matrices: ");
        for(int i = 0; i < n; i++) {
            int one = (cols[i][0]*K[0][0] + cols[i][1]*K[0][1])%26;
            int two = (cols[i][0]*K[1][0] + cols[i][1]*K[1][1])%26;
            System.out.print(one + " " + two + "\t");
            op += (char)(one+'A');
            op += (char)(two+'A');
        }

        return op;
    }

    static void dec(String cText, int[][] K) {
        int[][] K1 = new int[2][2];
        System.out.println("\nDecryption: ");
        int d = (K[0][0]*K[1][1] - K[0][1]*K[1][0] + 26)%26, d1=1;
        while ( (d*d1)%26 != 1)
            d1++;
        System.out.println("D: " + d + "\t\tD Inverse: " + d1);
```

```
        System.out.println("K Inverse:");
        K1[0][0] = (d1 * K[1][1])%26;
        K1[0][1] = (d1 * 25 * K[0][1])%26;
        K1[1][0] = (d1 * 25 * K[1][0])%26;
        K1[1][1] = (d1 * K[0][0])%26;
        for(int i = 0; i < 2; i++) {
            System.out.println(K1[i][0] + "\t" + K1[i][1]);
        }

        System.out.println("\nDecrypted: " + hillCipher(cText, K1));
    }

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
                System.out.print("Plain Text: ");   String pText = sc.nextLine();
                System.out.print("Key:\n");         int[][] K = new int[2][2];
                for(int i = 0; i < 2; i++)
                    for(int j = 0; j < 2; j++)
                        K[i][j] = sc.nextInt();
                System.out.println("Encryption:");
                String cText = hillCipher(pText, K);
                System.out.println("\nEncrypted: "+ cText);
                dec(cText, K);
        }
}
```

```
Plain Text: DCODE
Key:
2 3
5 7
Encryption:
Pre-Op Matrices: 3 2    14 3    4 25
PostOp Matrices: 12 3   11 13   5 13
Encrypted: MDLNFN

Decryption:
D: 25           D Inverse: 25
K Inverse:
19      3
5       24
Pre-Op Matrices: 12 3   11 13   5 13
PostOp Matrices: 3 2    14 3    4 25
Decrypted: DCODEZ
```

Vignere

```
import java.util.*;

public class Main
{
   static String enc(String ip, String k) {
      String op = "";
      for(int i = 0; i < ip.length(); i++) {
         int c_INT = (ip.charAt(i) + k.charAt(i%k.length()) - 2*'A')%26 + 'A';
         op += (char)c_INT;
      }
      return op;
   }

   static String dec(String ip, String k) {
      String op = "";
      for(int i = 0; i < ip.length(); i++) {
         int c_INT = (ip.charAt(i) - k.charAt(i%k.length()) + 26)%26 + 'A';
         op += (char)c_INT;
      }
      return op;
   }

         public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
                  System.out.print("Plain Text: ");   String pText = sc.nextLine();
                  System.out.print("Key: ");          String k = sc.nextLine();
                  String cText = enc(pText, k);
                  System.out.println("Encrypted: "+ cText);
                  System.out.println("Decrypted: "+ dec(cText, k));
         }

}
```



Rail Fence:

```
import java.util.*;

public class Main {

    static String enc(String ip) {
        char[][] RF = new char[2][ip.length()];
        String op = "";
                    for(int i = 0, j = 0; j < ip.length(); j++) {
                       RF[i][j] = ip.charAt(j);
                       i = (i+1)%2;
                    }
                    for(int i = 0; i < 2; i++)
                        for(int j = 0; j < ip.length(); j++)
                            if(RF[i][j] != '\0')
                                op += RF[i][j];
                    return op;
    }

    static String dec(String ip) {
        char[][] RF = new char[2][ip.length()];
        String op = "";
        int mid = (ip.length()-1)/2 + 1;
        for(int i = 0, j = mid; i < mid; i++, j++) {
            op += ip.charAt(i);
            if(j < ip.length())
                op += ip.charAt(j);
        }
        return op;
    }

    public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                    System.out.print("Plain Text: ");   String pText = sc.nextLine();
                    String cText = enc(pText);
                    System.out.println("Encrypted: " + cText);
                    System.out.println("Decrypted: " + dec(cText));
            }
}
```

```
Plain Text: SECURITY
Encrypted: SCRTEUIY
Decrypted: SECURITY
```

DES

```
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.util.*;
class des {
    byte[] key = null;
    byte[] encrypt(byte[] word) {
        try {
            key = getKey();
            Cipher cipher = Cipher.getInstance("DES");
            SecretKeySpec spec = new SecretKeySpec(key, "DES");
            cipher.init(Cipher.ENCRYPT_MODE, spec);
            return cipher.doFinal(word);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return word;
    }
    byte[] decrypt(byte[] word) {
        try {
            Cipher cipher = Cipher.getInstance("DES");
            SecretKeySpec spec = new SecretKeySpec(key, "DES");
            cipher.init(Cipher.DECRYPT_MODE, spec);
            return cipher.doFinal(word);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return word;
    }
    byte[] getKey() throws NoSuchAlgorithmException {
        Random random = new Random();
        String rString = String.valueOf(random.nextInt(10000));
        byte[] seed = rString.getBytes();
        KeyGenerator keyGen = KeyGenerator.getInstance("DES");
        SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG");
        secureRandom.setSeed(seed);
        keyGen.init(56, secureRandom);
        SecretKey key = keyGen.generateKey();
        return key.getEncoded();
    }

}
```

AES

```
import javax.crypto.*;
import java.util.*;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
class aes {
    String sKey = "ABCDEFGHIJKLMNOP";
    byte[] key;
    byte[] encrypt(byte[] word) {
        try {
            key = sKey.getBytes("UTF-8");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
            SecretKeySpec spec = new SecretKeySpec(key, "AES");
            IvParameterSpec ivSpec = new IvParameterSpec(key);
            cipher.init(Cipher.ENCRYPT_MODE, spec, ivSpec);
            return cipher.doFinal(word);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return word;
    }
    byte[] decrypt(byte[] word) {
        try {
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
            SecretKeySpec spec = new SecretKeySpec(key, "AES");
            IvParameterSpec ivSpec = new IvParameterSpec(key);
            cipher.init(Cipher.DECRYPT_MODE, spec, ivSpec);
            return cipher.doFinal(word);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return word;
    }
}
public class Main {
    public static void main(String[] args) {
        aes _aes = new aes();
        Scanner sc = new Scanner(System.in);
        String word = sc.nextLine();
        byte[] encrypted = _aes.encrypt(word.getBytes());
        byte[] decrypted = _aes.decrypt(encrypted);
        System.out.println("WORD :" + word);
        System.out.println("ENCRYPTED :" + new String(encrypted));
        System.out.println("DECRYPTED :" + new String(decrypted));
    }
}
```

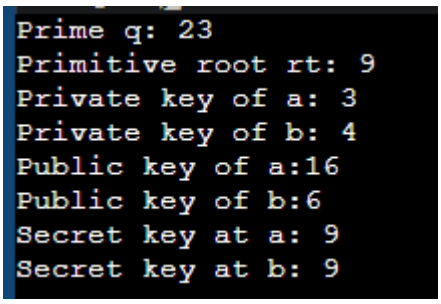
RSA

```
import java.util.*;
import java.math.BigInteger;
public class Main {
   static BigInteger p, q, M, C, n, phi, e, d;

   static void enc() {
      for(long i = 2; i < phi.doubleValue(); i++) {
         BigInteger curr = BigInteger.valueOf(i);
         if(phi.gcd(curr).equals(BigInteger.valueOf(1))) {
            e = BigInteger.valueOf(i);
            break;
         }
      }
      C = M.modPow(e, n);
      System.out.println("e: " + e);
      System.out.println("Encrypted: " + C);
   }

   static void dec() {
      d = e.modInverse(phi);
      M = C.modPow(d, n);
      System.out.println("d: " + d);
      System.out.println("Decrypted: " + M);
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("p: ");    p = new BigInteger(sc.nextLine());
      System.out.print("q: ");    q = new BigInteger(sc.nextLine());
      System.out.print("M: ");    M = new BigInteger(sc.nextLine());

      n = p.multiply(q);
      phi = (p.subtract(BigInteger.valueOf(1))).multiply(q.subtract(BigInteger.valueOf(1)));

      System.out.println("n: " + n);
      System.out.println("phi: " + phi);

      enc();
      dec();
   }
}
```

```
p: 3
q: 11
M: 2
n: 33
phi: 20
e: 3
Encrypted: 8
d: 7
Decrypted: 2
```

Diffie Hellman

```
import java.util.*;
import java.math.BigInteger;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      BigInteger q, rt, xa, xb, ya, yb;
      System.out.print("Prime q: ");              q = new BigInteger(sc.nextLine());
      System.out.print("Primitive root rt: ");     rt = new BigInteger(sc.nextLine());

      System.out.print("Private key of a: ");     xa = new BigInteger(sc.nextLine());
      System.out.print("Private key of b: ");     xb = new BigInteger(sc.nextLine());

      ya = rt.modPow(xa, q);
      yb = rt.modPow(xb, q);

      System.out.println("Public key of a:" + ya);
      System.out.println("Public key of b:" + yb);

      System.out.println("Secret key at a: " + yb.modPow(xa, q));
      System.out.println("Secret key at b: " + ya.modPow(xb, q));

   }
}
```



SHA

```
import java.util.*;
import java.math.BigInteger;
import java.security.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Message: ");
        String msg = sc.nextLine();
        try {
            MessageDigest md = MessageDigest.getInstance("SHA1");
            md.update(msg.getBytes());
            byte[] op_b = md.digest();
            String op = new BigInteger(1, op_b).toString(16);
            System.out.println("Hash: " + op);
        } catch (Exception e){
            System.out.println(e);
        }
    }
}
```

DSA

```
import java.util.*;
import java.security.*;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Message: ");
      String word = sc.nextLine();
      KeyPair keys = null;
      try {
         //get KeyPair
         SecureRandom sr = new SecureRandom();
         KeyPairGenerator keygen = KeyPairGenerator.getInstance("RSA");
         keygen.initialize(2048, sr);
         keys = keygen.generateKeyPair();

         //Create Dig Sig
         Signature sign = Signature.getInstance("SHA1WithRSA");
         sign.initSign(keys.getPrivate());
         sign.update(word.getBytes());
         byte[] op = sign.sign();
         System.out.println("Sign:" + new String(op));

         //verify sig
         Signature sign2 = Signature.getInstance("SHA1WithRSA");
         sign2.initVerify(keys.getPublic());
         sign2.update(word.getBytes());
         boolean flag = sign2.verify(op);
         System.out.println("Verification: "+flag);

      } catch (Exception e){
         System.out.println(e);
      }
   }
}
```