Name:Aaryan Kalbhor
Roll No:28
Div:D15B

RSA key Generating algorithm.

Algorithm:

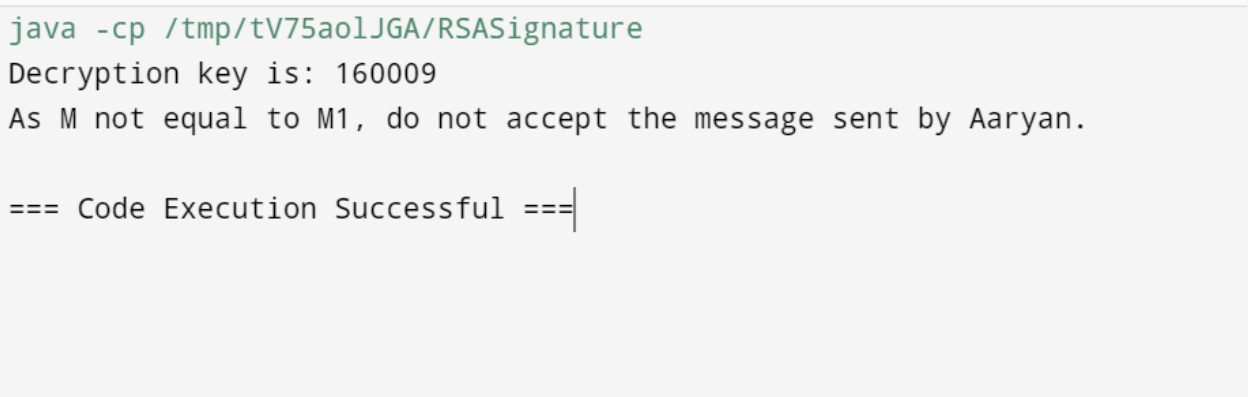
1) Choose two large prime numbers p and q.
2) Calculate n=p*q
3) Select public key e such that it is not a factor of (p-1)*(q-1)
4) Select private key d such that the following equation is true (d*e)mod(p-1)(q-1)=1 or d is inverse of e in modulo (p-1)*(q-1)

Code:

```
import java.util.ArrayList;
import java.util.List;

public class RSASignature {

  // Euclid's Algorithm for GCD
  public static int euclid(int m, int n) {
    if (n == 0) {
      return m;
    } else {
      return euclid(n, m % n);
    }
  }

  // Extended Euclid's Algorithm
  public static int[] exteuclid(int a, int b) {
    int r1 = a, r2 = b;
    int s1 = 1, s2 = 0;
    int t1 = 0, t2 = 1;
    while (r2 > 0) {
      int q = r1 / r2;
      int r = r1 - q * r2;
      r1 = r2;
      r2 = r;

      int s = s1 - q * s2;
      s1 = s2;
```

```
            s2 = s;

            int t = t1 - q * t2;
            t1 = t2;
            t2 = t;

            if (t1 < 0) {
                t1 += a;
            }
        }
        return new int[]{r1, t1};
    }

    public static void rsaSignature() {
        int p = 823;
        int q = 953;
        int n = p * q;
        int Pn = (p - 1) * (q - 1);

        // Generating keys that are co-prime with Pn
        List<Integer> keys = new ArrayList<>();
        for (int i = 2; i < Pn; i++) {
            if (euclid(Pn, i) == 1) {
                keys.add(i);
            }
        }

        int e = 313;

        if (!keys.contains(e)) {
            System.out.println("Key " + e + " is not valid. Choose a different encryption key.");
            return;
        }

        int[] result = exteuclid(Pn, e);
        int r = result[0];
        int d = result[1];

        if (r == 1) {
            System.out.println("Decryption key is: " + d);
        } else {
            System.out.println("Multiplicative inverse for the given encryption key does not
exist.");
            return;
        }

        int M = 19070;
        int S = modPow(M, d, n);
        int M1 = modPow(S, e, n);

        if (M == M1) {
            System.out.println("As M = M1, message sent by Aaryan.");
        } else {
            System.out.println("As M not equal to M1, do not accept the message sent by
Aaryan.");
        }
    }

    // Function to compute (base^exp) % mod
    public static int modPow(int base, int exp, int mod) {
        int result = 1;
        while (exp > 0) {
            if ((exp & 1) == 1) {
                result = (result * base) % mod;
            }
            base = (base * base) % mod;
            exp >>= 1;
        }
        return result;
    }

    public static void main(String[] args) {
        rsaSignature();
    }
}
```

Output:



```
java -cp /tmp/tV75aolJGA/RSASignature
Decryption key is: 160009
As M not equal to M1, do not accept the message sent by Aaryan.

=== Code Execution Successful ===
```