CRYPTOGRAPHY LABORATORY FILE
CS-511

Submitted to:
Dr. Samayveer Singh
Assistant Professor
Department of Computer Science

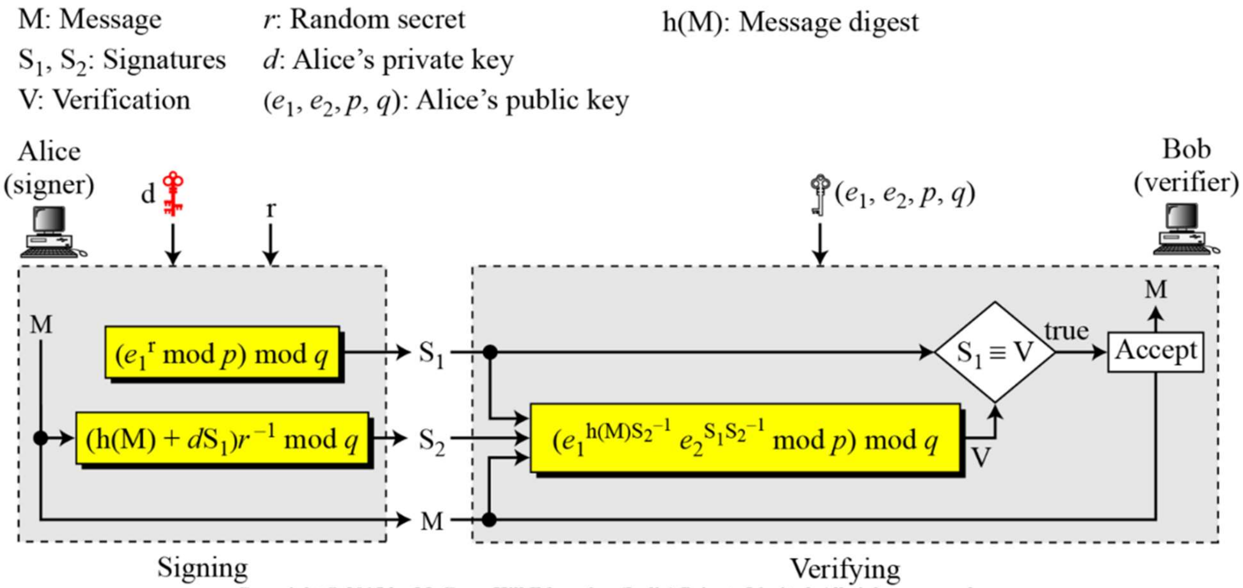
Submitted by:
Shashi Shekhar Azad
Roll No.: 23203029
M. Tech. CSE 1st Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DR. B. R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR

Objective 13

Write a program to implement the digital signature standard for validate the authentication and integrity during data transmission.

Procedure:



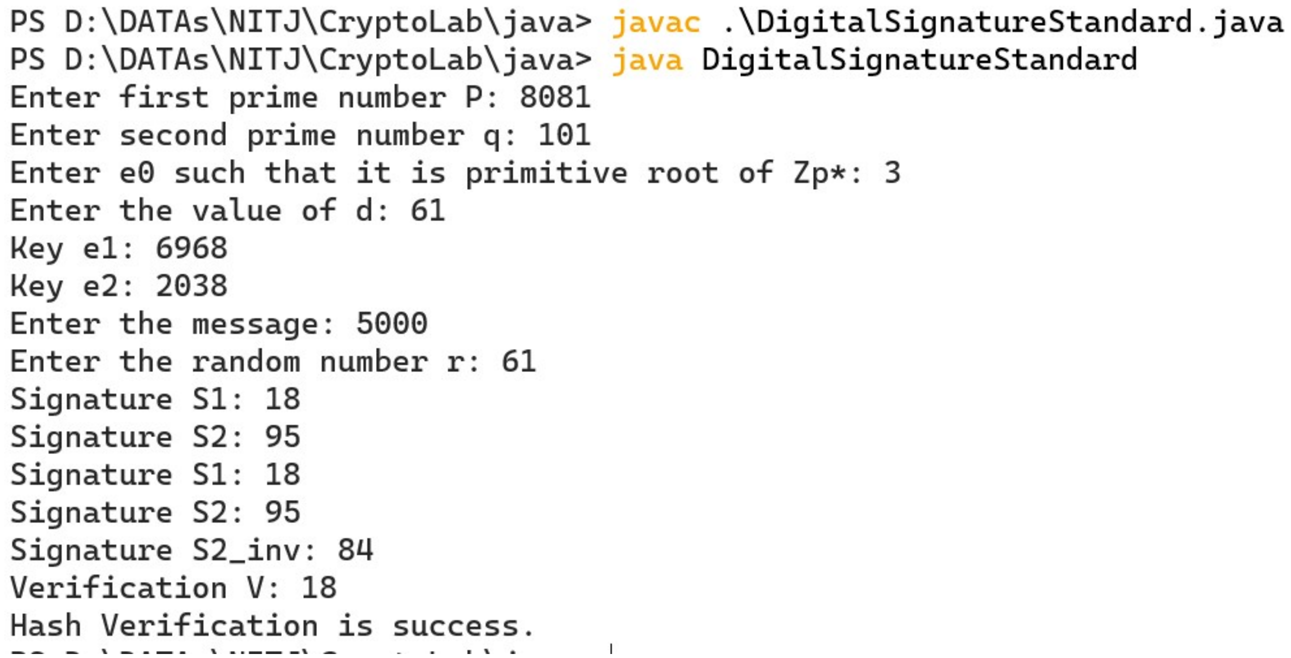
Code:

```
import java.util.Scanner;
public class DigitalSignatureStandard {
    static long e0, e1, e2, p, q, d;
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        keyGeneration();
        System.out.print("Enter the message: ");
        int m = sc.nextInt();
        System.out.print("Enter the random number r: ");
        int r = sc.nextInt();
        long[] signature = signing(m, r, d, p, q);

        if (verification(signature, m)) {
            System.out.println("Hash Verification is success.");
        } else {
            System.out.println("Hash Verification is failed.");
        }
    }
    private static void keyGeneration() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first prime number P: ");
        p = sc.nextInt();
        while (!isPrime(p)) {
            System.out.print(p + " is not a prime number, Enter prime number: ");
            p = sc.nextLong();
        }
        System.out.print("Enter second prime number q: ");
        q = sc.nextInt();
        while (!isPrime(q) || (p == q)) {
            System.out.print(q + " is not a prime number, Enter prime number: ");
            q = sc.nextLong();
        }
        System.out.print("Enter e0 such that it is primitive root of Zp*: ");
        e0 = sc.nextLong();
        while (!isPrimitiveRoot(e0, p)) {
            System.out.print("e0 is not primitive root of Zp*, Enter another: ");
            e0 = sc.nextLong();
        }
        System.out.print("Enter the value of d: ");
        d = sc.nextLong();
        e1 = modPow(e0, (p - 1) / q, p);
        e2 = modPow(e1, d, p);
        System.out.println("Key e1: " + e1);
        System.out.println("Key e2: " + e2);
    }
    private static long[] signing(long m, long r, long d, long p, long q) {
        long[] signature = new long[2];
    long S1 = modPow(e1, r, p) % q;
    long S2 = ((m + (d * S1)) * multiplicativeInverse(r, q)) % q;
    signature[0] = S1;
    signature[1] = S2;
    System.out.println("Signature S1: " + S1);
    System.out.println("Signature S2: " + S2);
    return signature;
}
private static boolean verification(long[] signature, long m) {
    long S2_inv = multiplicativeInverse(signature[1], q);
    long V = (modPow(e1, (m * S2_inv), p) * modPow(e2, (signature[0] * S2_inv), p)) % q;
    V = signature[0];
    System.out.println("Signature S1: " + signature[0]);
    System.out.println("Signature S2: " + signature[1]);
    System.out.println("Signature S2_inv: " + S2_inv);
    System.out.println("Verification V: " + V);
    return V == signature[0];
}
static long multiplicativeInverse(long a, long b) {
    long min = Math.min(a, b);
    long max = Math.max(a, b);
    a = max;
    b = min;
    long t1 = 0, t2 = 1;
    long t = 0, q = 0, r = 1;
    while (r != 0) {
        q = a / b;
        r = a % b;
        t = t1 - (t2 * q);
        a = b;
        b = r;
        t1 = t2;
        t2 = t;
    }
    if (t1 < 0) {
        t1 += t;
    }
    return t1;
}
static long modPow(long base, long exponent, long modulo) {
    long result = 1;
    base = base % modulo;
    while (exponent > 0) {
        if (exponent % 2 == 1) {
            result = (result * base) % modulo;
        }
        exponent = exponent >> 1;
        base = (base * base) % modulo;
    }
    return result;
}
public static boolean isPrime(long n) {
    if (n <= 1) {
        return false;
    }
    for (long i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
private static boolean isPrimitiveRoot(long e0, long p) {
    long phi = p - 1;
```

```
        for (long q = 2; q <= phi; q++) {
            if (phi % q == 0) {
                long result = modPow(e0, phi / q, p);
                if (result == 1) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

Output:



```
PS D:\DATAs\NITJ\CryptoLab\java> javac .\DigitalSignatureStandard.java
PS D:\DATAs\NITJ\CryptoLab\java> java DigitalSignatureStandard
Enter first prime number P: 8081
Enter second prime number q: 101
Enter e0 such that it is primitive root of Zp*: 3
Enter the value of d: 61
Key e1: 6968
Key e2: 2038
Enter the message: 5000
Enter the random number r: 61
Signature S1: 18
Signature S2: 95
Signature S1: 18
Signature S2: 95
Signature S2_inv: 84
Verification V: 18
Hash Verification is success.
```