SurajSG23 /
Cryptography-Assignment

Code Issues Pull requests Actions Projects Security Insights

Cryptography-Assignment / Elgamal

SurajSG23 Update Elgamal c67b32c · now

66 lines (50 loc) · 2.21 KB

Code Blame Raw

```java
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class ElGamal {

    // Method to compute a^b mod p using BigInteger for large numbers
    public static BigInteger modExp(BigInteger a, BigInteger b, BigInteger p) {
        return a.modPow(b, p);
    }

    // Method to find modular inverse of a mod p
    public static BigInteger modInverse(BigInteger a, BigInteger p) {
        return a.modInverse(p);
    }

    public static void main(String[] args) {
        // Scanner for user input
        Scanner scanner = new Scanner(System.in);

        // Large prime p and generator g
        BigInteger p = new BigInteger("7873"); // A small prime for simplicity
        BigInteger g = new BigInteger("2");    // A common generator
        SecureRandom random = new SecureRandom();

        // Key generation
        BigInteger d = new BigInteger(256, random); // Private key d
        BigInteger e1 = modExp(g, d, p); // Public key e1 = g^d mod p
        System.out.println("Public Key e1: " + e1);
        System.out.println("Private Key d: " + d);

        // Taking plaintext input from the user
        System.out.print("Enter the message (integer form): ");
        BigInteger m = scanner.nextBigInteger(); // Message to encrypt
        scanner.close();



        // Encryption process
        BigInteger r = new BigInteger(256, random); // Random r for encryption

        BigInteger c1 = modExp(g, r, p); // c1 = g^r mod p
        BigInteger c2 = m.multiply(modExp(e1, r, p)).mod(p); // c2 = m * e1^r mo

        System.out.println("Encrypted Message:");
        System.out.println("c1: " + c1);
        System.out.println("c2: " + c2);

        // Decryption process
        BigInteger c1_d = modExp(c1, d, p); // c1^d mod p
        BigInteger c1_d_inv = modInverse(c1_d, p); // (c1^d)^-1 mod p
        BigInteger decryptedMessage = c2.multiply(c1_d_inv).mod(p); // m = c2 *

        System.out.println("Decrypted Message: " + decryptedMessage);
    }
}

//Output
Public Key e1: 4363
Private Key d: 68202669719519956000598852477675306652742486999070665318379244089
Enter the message (integer form): 123456789
Encrypted Message:
c1: 808
c2: 1136
Decrypted Message: 276
```