BTI4202 - Exercise Sheet 4

Jan Fuhrer, Andy Bigler, Joel Häberli

April 28, 2023

Contents

1 RSA Encryption Scheme

Task Perform a "Textbook RSA" encryption/decryption for $k = 7$ and $m = 2$

Process Key Generation

- $p = randPrime(\lfloor \frac{k}{2} \rfloor) \rightarrow 5$
- $q = randPrime(\lceil \frac{k}{2} \rceil) \rightarrow 11$
- $n = 5 \cdot 11 = 55$
- $\phi(n) = \phi(55) = 4 \cdot 10 = 40$
- Choose e from $\mathbb{Z}_{40}^{*} \rightarrow 7$
- $d = e^{-1} = 23 \rightarrow$ check with: $7 \cdot 23 \bmod 40 \equiv 1$
- $pk = (55, 7)$ and $sk = (55, 23)$

Process Encryption for $m = 2$

- $c = 2^7 \bmod 55 = 18$

Process Decryption for $c = 18$

- $m = 18^{23} \bmod 55 = 2$

2 ElGamal Encryption Scheme

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

public class ElGamalEncryptionScheme {
    private final BigInteger p; // Prime number
    private final BigInteger g; // Generator
    private final BigInteger q;

    public ElGamalEncryptionScheme(BigInteger p, BigInteger g) {
        this.p = p;
        this.g = g;
        this.q = p.subtract(BigInteger.ONE).divide(BigInteger.valueOf(2));
    }

    public BigInteger[] keyGen() {
        Random random = new SecureRandom();
        BigInteger x = new BigInteger(q.bitLength(), random).mod(q);
        BigInteger h = g.modPow(x, p);
        return new BigInteger[]{x, h};
    }

    public BigInteger[] encpk(BigInteger m, BigInteger h) {
        Random random = new SecureRandom();
        BigInteger y = new BigInteger(q.bitLength(), random).mod(q);
        BigInteger c1 = g.modPow(y, p);
        BigInteger s = h.modPow(y, p);
        BigInteger c2 = m.multiply(s).mod(p);
        return new BigInteger[]{c1, c2};
    }

    public BigInteger decsk(BigInteger[] c, BigInteger x) {
        BigInteger c1 = c[0];
        BigInteger c2 = c[1];
        BigInteger s = c1.modPow(x, p);
        BigInteger sInv = s.modInverse(p);
        BigInteger m = c2.multiply(sInv).mod(p);
        return m;
    }
}
```

```java
import java.math.BigInteger;

public class Main {
    public static void main(String[] args) {
        BigInteger p = new BigInteger("23");
        BigInteger g = new BigInteger("5");

        ElGamalEncryptionScheme elGamal = new ElGamalEncryptionScheme(p, g);

        BigInteger[] keys = elGamal.keyGen();
        BigInteger x = keys[0]; // Private key
        BigInteger h = keys[1]; // Public key

        BigInteger m = new BigInteger("9"); // Message to be encrypted

        BigInteger[] encryptedMessage = elGamal.encpk(m, h);
        BigInteger decryptedMessage = elGamal.decsk(encryptedMessage, x);

        System.out.println("Original Message: " + m);
        System.out.println("Encrypted Message: " + encryptedMessage[0] + ", " +
    encryptedMessage[1]);
        System.out.println("Decrypted Message: " + decryptedMessage);
    }
}
```

3 Security of ElGamal