Asymmetric Ciphers

Foros Valentin

December 4, 2022

1 RSA

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone. The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

The RSA algorithm ensures that the keys, are as secure as possible. The following steps highlight how it works:

1. Generating the keys:
   - Select two large prime numbers, x and y. The prime numbers need to be large so that they will be difficult for someone to figure out.
   - Calculate $n = x * y$
   - Calculate the totient function $\phi(n) = (x\text{-}1)(y\text{-}1)$
   - Select an integer e, such that e is co-prime to $\phi(n) and 1 < e < \phi(n)$. The pair of numbers (n,e) makes up the public key.
   - Calculate d such that $e.d = 1 mod \phi(n)$.
2. Encryption:
   - Given a plaintext P, represented as a number, the ciphertext C is calculated as: $C = P^e mod n$.
3. Decryption:
   - Using the private key (n,d), the plaintext can be found using: $P = C^d mod n$.

2 Implementation

This program implements RSA Algorithm, by creating RSA Class, which has such BigInteger variables as: p, q, $n = (pq), e, f_n = (p-1)(q-1), d = e^(-1) mod(f_n)$.

Program operates with large numbers, that sometimes bigger than int and long, so it's a point to use BigInteger class. Program takes some values from user - (p, q, e) to create Object of RSA class (RSA rsa) Also we need user to give message he wants to encrypt - there are two types of messages:

1. int message - stands for int number to encrypt and decrypt
2. string message - message gets splited in chars $->$ to int by ASCII table.

Each int gets encrypted/decrypted Program implements two types of methods, for each type of message - there are special parts in method's names for them:

1. "ForIntMessage" methods - for int message: encryptForInt computes c (encrypted message) variable; decryptForInt - decodes c returning inputed message.

```
    public BigInteger encryptForInt(BigInteger message){
        BigInteger c = (message.pow(e.intValue())).remainder(n);
        return c;
    }

public BigInteger decryptForInt(BigInteger c){
    BigInteger message;
    message = c.modPow(d,n);

    return message;
}
```

2. "ForStringMessage" - for string messages: encryptForStringMessage - encodes string message by transorming string message into BigInt array , using strToInt() method, returns Bigint array; decryptForStringMessage decryptes every element in BigInt array of encoded values, using decryptForInt() method.

```
    private BigInteger[] encryptForStringMessage(String message){
        BigInteger[] intMessageArr = strToInt(message);
        BigInteger[] encryptedIntMessageArr = new BigInteger[intMessageArr.length];
        for (int i = 0; i < intMessageArr.length; i++) {
            BigInteger EncryptElemOfIntMessage = (intMessageArr[i].pow(e.intValue())).remainder(
            encryptedIntMessageArr[i] = EncryptElemOfIntMessage;
        }
    return encryptedIntMessageArr;
}

private int[] decryptForStringMessage(BigInteger[] array){
        int[] DecryptedIntMessageArr = new int[array.length];
        for (int i = 0; i < array.length; i++) {
            BigInteger DecryptElemOfIntMessage = decryptForInt(array[i]);
            int elOfDecryptMessage = DecryptElemOfIntMessage.intValue();
            DecryptedIntMessageArr[i] = elOfDecryptMessage;
        }
    return DecryptedIntMessageArr;
}
```

3. strToInt - transorms string message into BigInt array;

```
    private BigInteger[] strToInt(String message){
        BigInteger[] intMessageArray = new BigInteger[message.length()];
        for (int i = 0; i < message.length(); i++) {
            BigInteger elOfIntArr = BigInteger.valueOf(((int) message.charAt(i)));
            intMessageArray[i] = elOfIntArr;
        }
    return intMessageArray;
}
```

3 Conclusion:

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys – one public key and one private key – to encrypt and decrypt a message and protect it from unauthorized access or use. While working on this laboratory work, I got familiar with the asymmetric ciphers (RSA) and how it is implemeted and used. RSA encryption depends on using the receiver's public key, so we don't have to share any secret key to receive messages from others. Also, the encryption process is faster than that of the DSA algorithm.