Mini Project 1 - Security 1

Adam Hadou Temsamani (ahad)

October 1, 2023

0.1 Solution

The source file with the solution to the exercises can be found under the file *main.py*.

0.2 Exercise 1

The information that we have includes; shared generator, shared prime and Bob's public key g^x. This means that we have all the information we need to use Elgamal encryption. I have done this by choosing a random y as the private key from the circular group. In the code *"1234"* has been chosen specifically, but this can be interchanged with any random number from the group. I then compute $g^y modp$. This computes my public key. I then need to compute $(g^x)^y modp$. Which I can use to encrypt the message that I want to send by multiplying it with the message; $((g^x)^y modp) * m$. This can be found under the *encrypt()* method.

0.3 Exercise 2

The information that we have is the same as the previous exercise. Since the numbers we are working with are not too high. The fastest way to find find Bob's private key is to simply use a brute forcing method, as can be seen in the method *findPrivateKey()*. This works as it is not computationally difficult to do so.
Now that we have the Bob's private key, it is very simple to reconstruct the message. As we would do exactly what Bob would do when receiving a encrypted message from Alice. This is done by doing the opposite of encrypting it. We take Alice public key to the power of the inverse of Bob's private key modulo the prime, as can be seen in the method *reconstructMessage()*.

0.4 Exercise 3

To tamper with Alice message and double the amount that Alice sent to Bob. We simply have to take the cipher text and multiply it by 2. We can do this since $g^y * m$, is to encrypt the message, but we can double the value of the cihper by simply multiplying it by 2. So when we try to decrypt by dividing with g^y, the plain text will also have the doubled in value. This can be seen in the method *modifyMessage()*

0.5 Answers

All the answers to the exercises can be found in the figure below. These can also be found by running the main function in the *main.py* file.

```
=== Mini Project 1 ===

Exercise 1: Running encrypt()
gy: 2879, c: 1907

Exercise 2: Running findPrivateKey() and reconstructMessage()
Private key: 66
Reconstructed message: 2000

Exercise 3: Running with and without modifyMessage()
Normal message: 2000
Modified message: 4000
```

Figure 1: Exercise Answers.