Mandatory Exercise Set 1 - ElGamal

Simon Lind Kristiansen (silk@itu.dk)

October 4th, 2022

Introduction

I have implemented the solutions of the exercise set in F#. I am running .NET 6.0.304 on a Windows 10 machine, with .NET added to my Path environment. From the root directory of my hand-in I have run the following commands to perform the computations:

```
dotnet fsi
#load "Program.fs"
```

After that, values can be inspected, for example by typing

```
ElGamal.decrypted
```

Exercise 1

To generate r I use the the System.Security.Cryptography.RandomNumberGenerator class available in .NET to compute a cryptographically strong random number. I represent all numbers as BigIntegers whenever possible to avoid overflows as well as the convenient ModPow method. I generate c_1 by computing $g^r \ \% \ p$ and c_2 by computing $m * pk^r \ \% \ p$, m being the message 2000, pk being Bob's public key and p being the shared prime

Exercise 2

Computing Bob's private key is trivial as the key space is tiny. I use a very simple brute force algorithm to calculate it, checking for every possible integer value i if $g^i \ \%p \ = sk$ until sk has been found. More efficient algorithms exist, but I chose the simplest possible algorithm to save time, as the computational power needed is negligible. Bob's private key is 66. To decrypt we use the formula $\frac{c_2}{c_1^{sk}}$. Eve decrypts the message, $\frac{c_2}{c_1^{sk}} = 2000$

Exercise 3

Mallory knows that the encrypted message c_2 is a simple integer, 2000. All they have to do is multiply the encrypted message by 3 as 2000 * 3 = 6000. Calling this new encrypted message c_{2m}, we get $\frac{c_{2m}}{c_1^{sk}} = 6000$