MANDATORY HAND-IN 1

IT-University of Copenhagen

Mikkel Bistrup Andersen

27. september 2024

Copenhagen

Contents

0.1 Assignment 1 3
0.2 Assignment 2 3
0.3 Assignment 3 3
0.4 Output 4

0.1 Assignment 1

In question 1 we are asked to send an encrypted message. We are given a public key to start with, so we will have to generate the shared key. This is done with the following code:

```
func findKey(base, prime, s big.Int) *big.Int {

        result := big.NewInt(0)
        result.Exp(&base, &s, nil)
        result.Mod(result, &prime)
        return result
}
```

The output can then we factored with the message for final encryption.

0.2 Assignment 2

Question 2 asks us to intercept and decrypt the message sent in question 1. To do this we can use the following code to decrypt the encrypted message:

```
    func elgamelDecrypt(smsg, pKey, c big.Int) big.Int {
        sKey := findKey(pKey, *big.NewInt(Prime), smsg)
        result := big.NewInt(0)
        return *result.Div(&c, sKey)
}
```

We can then intercept the message by brute forcing until we have the secret. When we have it, we simply call our decrypting method:

```
    func interceptmsg(target, pKey, c big.Int) (s, msg big.Int) {
        base := big.NewInt(Base)
        prime := big.NewInt(Prime)
        i := big.NewInt(1)
        var limiter big.Int = *big.NewInt(1000)

        for k := *big.NewInt(1); k.Cmp(&limiter) < 0; k.Add(&k, i) {
                key := findKey(*base, *prime, k)

                if key.Cmp(&target) == 0 {
                        msg := elgamelDecrypt(k, pKey, c)
                        return k, msg
                }
        }
        return *big.NewInt(0), *big.NewInt(0)
}
```

0.3 Assignment 3

Question 3 simply asks us to change the message we intercepted earlier from 2000 to 6000. This can be done quite simply by hard coding the modification of the message

0.4 Output

```
$ Public key: 1, Message: 2000
$ Secret is: 66, Message is: 2000
$ Tampered message: 6000
```