1 Introduction

Recall the basic difference between symmetric-key cryptosystems and public-key cryptosystem. In the former, there is a secret key that is used for both encryption and decryption. In the latter, the encryption key is public and separate from the decryption key which is private. One of the interesting applications of public-key cryptosystem is the generation of a secret key over an insecure (but authenticated) channel. A well-known example of such a protocol is proposed by Diffie and Helman, inventors of the idea of public-key cryptography, and is known as *Diffie-Hellman key agreement(exchange)*. Suppose Alice and Bob want to agree on a secret key over an insecure channel. Here is the protocol they need to follow according to this scheme.

1. Either Alice or Bob chooses a large prime, and a generator α of the cyclic group $\mathbb{Z}_p^*$. Both p and α can be made public.
2. Alice chooses a random (and secret) integer a, $1 < a < p$, and Bob chooses a random (and secret) integer b, $1 < b < p$.
3. Alice send $x = \alpha^a$ to Bob, and Bob sends $y = \alpha^b$ to Alice.
4. Alice computes y^a and Bob computes x^b.

Note that all the computations are performed in the group $\mathbb{Z}_p^*$, hence they are mod p, and that both Alice and Bob end up with the same value: $y^a = (\alpha^b)^a = \alpha^{ab} = (\alpha^a)^b = x^b$, which is the common secret key between Alice and Bob.

Eve, of course, would like to be able to retrieve the secret key k from the information available to her: $p, \alpha, \alpha^a, \alpha^b$. This is known as the *Diffie-Hellman problem* (DFH). It turns out that this problem is closely related to another well-known problem, the discrete logarithm problem (DLP): Given prime a p, a generator α of $\mathbb{Z}_p^*$ and α^x, find x.

Exercise 1.0.1 *Show that DLP implies the DFH. That is, if Eve can solve the DLP then she can also solve the DFH.*

Therefore, if there is a polynomial time algorithm for the DLP, then there is a polynomial time algorithm for DFH as well. However, no polynomial time algorithm is known for the DLP. It is also an open problem whether it is possible to solve the DFH problem, in general, without having to solve the DLP. To this date, best known general solution for DFH problem is through DLP. For large enough groups, the DFH (and DLP) is assumed to be a hard problem. The security of Diffie-Hellman key

exchange protocol is based on this assumption. With this assumption the function $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ given by $f(x) = \alpha^x$ is considered to be a one-way function. Recall that the security of the RSA cryptosystem is based on the assumed difficulty of the factorization of large integers.

It is not too hard to modify the Diffie-Hellman protocol so that it works with three participants Alice, Bob and Carol.

Exercise 1.0.2 *Modify the Diffie-Hellman protocol so that Alice, Bob and Carol end up with the same secret key by communicating over an insecure channel.*

2 The ElGamal Cryptosystem and Signature Scheme

2.1 The Cryptosystem

We have seen a key exchange protocol whose security is based on the difficulty of the discrete logarithm problem. There is a full cryptosystem based on this problem. Proposed in 1985 [**?**], this scheme is known as ElGamal cryptosystem. It modifies the Diffie-Hellman protocol so that it can be used as an encryption and decryption system. Its security is also based on the difficulty of the DLP. It can be described as follows:

Alice wants to send secret messages to Bob. Bob chooses a large prime p, a random integer b, $1 \leq b \leq p-1$, and a primitive element α of $\mathbb{Z}_p^*$. He computes $\beta = \alpha^b$, and publishes the information (p, α, β). Alice wants to send her message m privately. She needs to the following.

1. Download Bob's public information.
2. Treat her message m as an integer $0 \leq m < p$. If m is greater than $p-1$ she breaks m into smaller blocks. (Details of this is discussed in the earlier module on the RSA system)
3. Alice chooses a random integer a and computes $R = \alpha^a$.
4. She also computes $M = \beta^a m$.
5. Alice send the pair (R, M) to Bob.

Receiving the pair (R, M), Bob decrypts the message using his private key b. It is clear that if Eve can solve the DLP then she can break the ElGamal system.

Exercise 2.1.1 *Show how exactly Bob decrypts the ciphertext. Verify that his decryption works correctly.*

2.2 The Signature Scheme

The ElGamal cryptosystem can be modified to produce a signature scheme. Suppose Alice wants to send a signed message m to Bob. Here the message is not necessarily secret. What is important is for Bob to verify that it is signed by Alice. To make the system work, Alice first chooses a large prime p, a primitive element α of $\mathbb{Z}_p^*$, and a secret integer a. She then computes $\beta = \alpha^a$ and proceeds as follows.

1. Publish (p, α, β)
2. Choose a random and private integer k such that $\gcd(k, p-1) = 1$
3. Compute $R = \alpha^k$
4. Compute $S \equiv k^{-1}(m - aR) \mod p-1$
5. Send the signed mesage (m, R, S) to Bob

Upon receiving the signed message Bob proceeds as follows

1. Download Alice's public information.
2. Compute $V_1 = \beta^R R^S \mod p$
3. Compute $V_2 = \alpha^m \mod p$
4. Accept the signature if $V_1 \equiv V_2 \mod p$.

It is not hard to see that Bob's verification scheme works.

Exercise 2.2.1 *Show that Bob's verification scheme is correct.*

The security of this signature scheme also depends on the difficulty of DLP. If Eve can find out the value of a (by solving the DLP problem for example) then she can follow the protocol and attach Alice's signature on any document she chooses.

If Alice wants to sign a second document, then it is very important that she uses a different value of the private number k. If not, it is quite likely for Eve to obtain the value of the private key a hence completely breake the system. If Alice signs two different messages m_1 and m_2 using the same random value k then she produces the same $R = \alpha^k$ value for both messages but different S values: $S_1 \equiv k^{-1}(m_1 - aR)$ and $S_2 \equiv k^{-1}(m_2 - aR)$. It follows that $S_1 k - m_1 \equiv -aR \equiv S_2 - m_2 \mod p-1$, and $(S_1 - S_2)k \equiv m_1 - m_2 \mod p-1$. The last congruence has a solution if

$d = \gcd(S_1 - S_2, p-1)$ divides $m_1 - m_2$ in which case it has d solutions for k (the only unknown in the equation) and there is a well known and efficient procedure to find these solutions. Eve can test every possible value of k until she finds the one with $\alpha^k = R$. Next she can solve the congruence $aR \equiv m_1 - kS_1 \mod p-1$ for a. Again there are $\gcd(R, p-1)$ solutions when $\gcd(R, p-1)$ divides $m_1 - kS_1$. She tests each possible solution until she finds the one that satisfies $\alpha^a = \beta$. At that point she completely breaks the system.

Example 2.2.2 .

Let us show this point with a concrete example. Suppose Alice uses the prime $p = 337741$ (which is too small to use in practice but will serve adequately as an illustration), and the generator $\alpha = 173$. She has a secret integer a. Her public information is $(p, 173, \beta)$ where $\beta = 251222$. Suppose her first message to sign is $m_1 = 120324$. She chooses a secret value for k (which is relatively prime with $p-1$). She finds $R = \alpha^k \mod p = 163,949$. Next she computes $S_1 \equiv k^{-1}(m_1 - aR) \equiv 28774 \mod p-1$. Her signed message is then the triple $(120324, 163949, 28774)$.

Now suppose Alice decides to sign another message $m_2 = 201027$ using the same secret value k. Following the protocol, she comes up with the signed message $(201027, 163949, 191293)$. Observing Alice's messages, Eve immediately realizes that she used the same value of k from the fact that the middle values in each message are equal. Eve proceeds as follows to first obtain the value of k, then the value of a.

First, she construct the congruence $(S_1 - S_2)k \equiv m_1 - m_2 \mod p-1$ in which k is the only unknown. In this case the equation turns out to be $-162519k \equiv -80703 \mod p-1$, or $175221k \equiv 257037 \mod 337740$. Since $\gcd(S_1 - S_2, p-1) = 3$ divides 337740, this equation has 3 solutions. These solutions can be found by first diving the congruence by 3 and getting $58407k \equiv 85679 \mod 112580$. Now, $\gcd(58407, 112580) = 1$ so this congruence has a unique solution $\mod 112580$ which is 1237. Then, the 3 solutions of the previous congruence are 1237, 113817, and 226397. Testing each solution Eve finds that the correct value of k is 1237.

Having obtained the value of k, Eve can discover the value of a as well from the equation $aR \equiv m_1 - S_1 k \mod p-1$ in which a is the only unknown for her. Obtained by rewriting the equation for S_1, this gives her the congruence $163949a \equiv 327326 \mod 337740$. Since $\gcd(163949, 337740) = 1$, this congruence has the unique solution 132394, which is the value of the secret key a. Now, Eve can forge Alice's signature on any document she chooses.

2.3 An Implementation of ElGamal Cryptosystem in Maple

We assume that the reader is familiar with the previous module on the RSA cryptosystem and its implementation in Maple. Therefore, we will only explain parts that do not appear in the previous module(including the supplementary Maple files).

Bob chooses a prime number as follows (which is much smaller than what should be used in practice)

```
> N := (rand(10^19 .. 10^20-1))():
> p := nextprime(N);
                        47745199971245512067
> isprime(p);
                                  true
```

Next, he needs to find a generator of the cyclic group $\mathbb{Z}_p^*$. A practical way of doing this is to pick a random value and check that it is a generator (if not try another value). This can be accomplished as follows:

```
> alpha := (rand(10^5 .. 2*10^5))();

                                183417
>Primitive(x-alpha) mod p;

                                 true
```

Finally, Bob chooses a random and secret value b. Let us say he chooses $b = 17305$, and computes

```
> b := 17305;
> beta := `mod`(Power(alpha, b), p);
                        31627624511892192254
```

He then publishes the following information:
$(p, \alpha,\ beta) = (75434670514966341829, 183417, 31627624511892192254)$.

Now it is Alice's turn. She first downloads Bob's public data. Suppose her secret message is "Red door". She converts it to number mod p. She can do that as follows:

```
>plaintext := "Red door";
                              "Red door"
> plaintextNumber := convert(plaintext, bytes);
             [82, 101, 100, 32, 100, 111, 111, 114]
> m := convert(plaintextNumber, base, 256, p);
                      [8245931918569530706]
```

Note that if the message (after converting to a number) was larger than p, she would need to break it into smaller parts. (This is shown in the previous module). Next, she chooses a random integer a. Say she picks $a = 454086$. Then she computes:

```
> a := 454086:
> R := `mod`(Power(alpha, a), p);
                       45601211411269254811
> M := `mod`(Power(beta, a)*m, p);
                      [89575804709965808 20]
```

And she sends the pair (R, M) to Bob. After receiving this pair all Bob needs to do is to compute

```
> decipherModp := `mod`(M*Power(R, p-b-1), p);
                      [8245931918569530706]
```

which is exactly Alice's message in numerical form. What Bob really computes to decrypt is the quantity MR^{-b}. Since Maple's Power function does not accept a negative integer as exponent we used the fact that $R^{-b} = R^{p-1-b} \mod p$ because $R^{p-1} = 1 \mod p$. To get the message in English, Bob converts the numerical value back to characters

```
> decipherMod256 := convert(decipherModp, base, p, 256);
             [82, 101, 100, 32, 100, 111, 111, 114]
> decipherText := convert(decipherMod256, bytes);
                              "Red door"
```