Project 1 Cryptography

Chiara Spadafora
chiara.spadafora@unitn.it

Irene Villa
irene.villa@unitn.it

Trento, 23 May 2023

Your project consists in writing a MAGMA code which implements the NTRUEncrypt algorithm, composed by three parts:

- *Key Generation* for generating encryption key pairs,
- *Encrypt* for encrypting a message,
- *Decrypt* for decrypting a message.

The solutions must be encoded in a .mag file called **nameofyourgroup_yoursurname.mag**. We require you to write the following function:

1. Write a function called *NTRUKeyGeneration(N,p,q,d)* with the following specifications:

```
NAME:  NTRUKeyGeneration(N,p,q,d)
INPUTS:   -N is a prime representing the message length
          -d is a positive integer
          -p is a prime representing how messages can be encoded
          -q is a positive integer
OUTPUTS:  -(sk,pk), the secret key, public key pair
```

2. Write a function called *NTRUEncrypt(pk,m)* with the following specifications:

```
NAME:  NTRUEncrypt(pk,m)
INPUTS:   -pk is the public key
          -m is the message represented as a sequence
OUTPUTS:  -c is the encrypted message
```

3. Write a function called *NTRUDecrypt(sk,c)* with the following specifications:

```
NAME:  NTRUDecrypt(sk,c)
INPUTS:   -sk is the secret key
          -c is the encrypted message
OUTPUTS:  -m is the decrypted message
```

4. Write a function called *NTRUEncryptDecrypt(N,d,p,q,m)* with the following specifications:

```
NAME:  NTRUEncryptDecrypt(N,p,q,d,m)
INPUTS:   -N is a prime representing the message length
          -d is a positive integer
          -p is a prime representing how messages can be encoded
          -q is a positive integer
          -m is the message represented as a sequence
OUTPUTS:  -true or false
```

   This function must generate a NTRUEncrypt key pair with *NTRUKeyGeneration(N,p,q,d)*, encrypt m with the public key (*NTRUEncrypt(pk,m)*) and decrypt the encryption with the private key (*NTRUDecrypt(sk,c)*), returning true if the decryption is equal to the original message m, false otherwise.

Clearly the function will be considered correct if and only if it returns true on every input.

Remark. Every input we will use during the evaluation lecture will be consistent with the specifications provided. The message will be delivered as a sequence whose elements lie in $\mathbb{Z}_p$ starting with the least significant bit, i.e. the sequence $[1, 2, -2, -1]$ in $\mathbb{Z}_5$ corresponds to the polynomial $1 + 2x - 2x^2 - x^3$. As you can see we will choose representative for each elements in the interval centered in 0, i.e. $\mathbb{Z}_5$ is the set $\{-2, -1, 0, 1, 2\}$ and **NOT** the set $\{0, 1, 2, 3, 4\}$. We will use the same format of the test vectors provided.

Since the algorithm is not deterministic and requires random generation of polynomial we are not able to provide usual test vectors for it. We will instead provide you with test vectors containing every random polynomial (namely f, g and r), as well as the message and its encryption.

All the code must be **completely written in MAGMA language**, without calling any external program.
The project will be tested and evaluated during the lecture on **06/06/2023**. If the program fails to work correctly, even on one test vector (or if it does not load, or if it is composed by more than one file), and this problem is not fixed before the end of the lecture, then all the team members will fail and they will have to attend again the whole lab session in Spring 2024. During the lecture, the speed of your algorithm will be evaluated as well.

Participants of the team presenting the faster algorithms will receive one extra point.