Workshop: McEliece CryptoSystem

Setup

Make sure Python 3.13 or higher is installed: https://www.python.org/downloads/

Also, it is beneficial to create a virtual environment to handle the libraries needed for the assignment, Instructions can be found here: https://docs.python.org/3/library/venv.html

Once setup, run the 'pip' command with the requirements.txt with

```
`pip3 install -r requirements.txt`
```

Background Info / Helpful Links:

- Slides
- Handwritten notes
- https://en.wikipedia.org/wiki/McEliece_cryptosystem
- https://en.wikipedia.org/wiki/Hamming_code
- https://www.youtube.com/watch?app=desktop&v=t4kiy4Dsx5Y
- https://docs.sympy.org/latest/modules/matrices/matrices.html#sympy.matrices.matrixbase.MatrixBase.nullspace
- https://numpy.org/doc/2.1/reference/random/generated/numpy.random.Generator.permutation.html#numpy.random.Generator.permutation

NOTE:

We are dealing with binary matrices here, but since we are storing integers are every index, we need to be mindful of converting each entry to binary. This can be achieved with a "%2".

I have linked some articles and libraries will be very useful, especially in making the Generator and Parity Check matrices.

Also, I highly recommend type hinting as it help to treat Python as a statically typed language when dealing with complex functions and classes.

Assignment

You are tasked with making an implementation of the McEliece CryptoSystem using Hamming Codes. This involves using the NumPy Python library to implement the matrices that will be used for the calculations. Due to memory constraints, you will implement a variation that handles 1 character at a time, but strings given may be longer than one character.

Three functions are required and have been abstracted in the CryptoSystem class.

key_gen() -> tuple[tuple, tuple]:

Generates the public and private key of the CryptoSystem.

Returns:

public_key, private_key (tuple[tuple, tuple]):

A tuple containing the public key in the first index and
the private key in the second position

The public_key contains $(\hat{G}, t)$
The private_key contains (S, P, H)

encrypt(message: str, public_key: tuple) -> str:

Given an ASCII string message and the public key, the message is encrypted into a ciphertext

Args:

message (str): the plaintext message to encrypt

public_key (tuple): the public key used to encrypt the plaintext

Returns:

ciphertext (str): An ASCII string output of the encrypted message (ciphertext)

decrypt(ciphertext: str, private_key: tuple) -> str:

Given an encrypted ASCII string ciphertext and the private key, the message is decrypted into the plaintext

Args:

ciphertext (str): the plaintext message to encrypt

private_key (tuple): the private key used to decrypt the ciphertext

Returns:

ciphertext (str): A string output of the decrypted message (plaintext)

Any questions about the assignment feel free to contact me at jdenny3@lsu.edu

Testing

To be able to run the unittest file, run the command

```
` python3 -m unittest ./ McEliece_test.py`
```

Passing all of these tests is an indicator your code functions as intended.