PROJECT 1 - CRACKING RSA

Instructions: Your online submission should consist of a zip file containing

- Code in the form of *.py files.
- Write up in the form of a tex file.

Description

In this project, you goal will be to decrypt a RSA encrypted file without access to the private key. Normally, the transmitting party will encrypt their message using public key (pq, e) and the receiving party will decrypt the coded message using the private key (pq, d).

It is *in theory* possible to determine the private key using the public key. In practice, it is difficult — at least for large primes — due to the lack of a truly efficient algorithm for factoring. The primes in this project are of relatively modest size (between 30 and 60 bits) and so even the most basic methods can factor the modulus on a desktop computer overnight.

You are given two files: `public-key.txt` and `cypher-text.txt`

The first, `public-key.txt`, has two lines. The first line is the modulus pq encoded in hexadecimal. The second line is the public exponent e, also encoded in hexadecimal.

Note that you can convert a hex string to an `int` by specifying the base:

```
n = int('deadbeef', base=16)
```

The second file, `cypher-text.txt`, has many lines. Each line is a single hex string which, when decrypted, gives the integer representation of an ASCII character. To convert from an `int` to a character, use `chr`:

```
print(chr(65),end='') # prints an A
print(chr(10),end='') # prints a new line
```

(To go in the other direction, from a character to an `int`, you can use the `ord` function.)

Tasks

Write code to

- Factor an integer (you can assume that it is the product of two primes)
- Compute the private exponent given a factorization of the modulus pq and the public exponent.
- Decrypt an encoded file using the modulus and private exponenet.

A. C. Knapp, American University, Washington, DC 20016
Email address: `knapp@american.edu`