# Tasnimul Hasnat

# 190041113

### CSE-4743


### December 31, 2023

---

# Usage

The Python Script can be used both as a **standalone script** or a **module** that can be imported.

## Script Usage Instructions

To run the code as a script , follow these steps:

1. You need Python 3 installed. It may also work on Python 2, I personally have not checked it.
2. Open a command line and navigate to the script folder.
3. Run `python rsa.py`.
4. Upon running the script, you will be prompted to choose whether you are Alice or Bob.
    - Either spell out the name or choose 1 or 2.
    - Any form of the word `Alice` or `aLice` or `alICE` will work. Same with `Bob`.
5. After selecting your role, a help menu will come up showing what functionalities are available to you.
6. When prompted for the key size, you will be prompted for the key size, you can either enter a power of 2 for a custom key size or leave it empty for the default key size of 128 bits.
7. After that, you will have the option to perform various actions. Press the corresponding values to choose the action.

1. **Encrypt a Message**: Users can encrypt a message with the public key of the other party.
2. **Decrypt a Message**: Users can decrypt a ciphertext with their own private key.
3. **Print Private Key**: Users can print their private key.
4. **Print Public Key**: Users can print their public key.
5. **Test The Module**: Users can run a set of automated tests to ensure the correctness of the implementation.
6. **Exit**: Users can exit the program.

Thus you can interact with the RSA cryptosystem.

### Example

```
~ python rsa.py
Welcome to RSA Cryptosytem.
Choose who you are?
        1.Alice
        2.Bob
alice
Hello Alice!

  Welcome to  the RSA cryptosystem.
  You have the following features,
    - encrypt() message  with any public key.
      encrypt(message,any_public_key) => ciphertext
    - decrypt() message with your own private key.
      decrypt(ciphertext) => plaintext
    - public_key()
      printout your public_key.
    - private_key()
      In case you want to printout your private_key.


Enter your keysize:
Must be a power of 2!!!
Leave empty for default keysize=128
23
```

```
Invalid Key Length.
Using Default.

Options:
        1. Encrypt a message
        2. Decrypt a message
        3. Print private key
        4. Print public key
        5. Test for Errors
        0. Exit
Enter the choice number: 3
Your public key is
`3457812247555222921661131647579647493738170516231282848283333331314010287891` with
n `4551310183808004332275199875575020266054166876089401316805329645678129129927`

Options:
        1. Encrypt a message
        2. Decrypt a message
        3. Print private key
        4. Print public key
        5. Test for Errors
        0. Exit
Enter the choice number: 0
```

## Module Usage Instructions

To use the `RSA` class as a module, you can straightly import the script as a module. Four functions are publicly available to you,

- `encrypt` : Allows to encrypt a plaintext into ciphertext.
- `decrypt` : Allows to decrypt the encrypted ciphertext into readable plaintext.
- `public_key` : Shows your public key.
- `private_key` : In case you want to, shows your private key.

**Example**

```
>>> from rsa import RSA
>>> A = RSA(512)
>>> B = RSA(256)
>>>
>>> ct = B.encrypt("hello from the other side",A.publicKey)
>>> ct
133510740715905741424262083513923008767676809304625010858561180142805796843920489364434654528770504449917910046339503903116602127168757250993911039704460843686318972380590864755951048143867450511183942871037446022769846998147313310288861957782870260556394986115165219540045426712522347810879236740625903445 10
>>>
>>> pt = A.decrypt(ct)
>>> pt
'hello from the other side'
```