# CSCI 531 Programming Assignment 3 (100 points)

Your task is to implement a system that uses AES to encrypt data and RSA to protect AES keys. You will need to write one program to generate RSA keys, and a second one that uses the RSA keys to perform the encryption and decryption of messages. Any correct implementation is acceptable. Your implementation need not be the best one in terms of run-time.

## Project Details

To complete the project, you will need to write two Python 3.7 programs:

1. **genkeys.py** — generate RSA public and private keys.
   - The program takes a single command line argument: the name of the user for whom the keys will be generated. For test purposes, use the user names **alice** and **bob**.
   - The program must be runnable directly from the command shell, e.g., ./genkeys.py alice
   - The program must generate an RSA public/private key pair using your own code (you *cannot* import RSA code from another module such as PyCrypto). It must use random.SystemRandom or os.urandom() as the source of pseudo-random bytes. The keys must be of practical cryptographic size (e.g., 1024 bits). You cannot use Python built-in functionality to generate prime numbers.
     - **Note:** To test whether generated random number is a prime you may refer to Miller-Rabin primality test. https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test#Miller%E2%80%93Rabin_test
   - The program must produce two output files, one containing the RSA private key (e.g., alice.prv) and the other one containing the RSA public key (e.g., alice.pub). The format of the key files is up to you.
   - You will need to write code to compute modular inverse and to test whether a number is prime. To achieve the latter, you can implement the Miller-Rabin primality test.

2. **crypt.py** — encrypt and decrypt data using AES-128 and RSA.
   - The program takes four command line arguments: a single flag (-e or -d) indicating whether the program will encrypt or decrypt a message, the name of the public or private key file to use (generated by keygen.py), the name of the file to encrypt or decrypt, and the name of the output file. For example, the following command will encrypt the file secret.txt using Alice's public key file alice.pub to produce the cipher text file secret.cip:

     ```
     ./crypt.py -e alice.pub secret.txt secret.cip
     ```

     Then the following command would decrypt the file secret.cip:

     ```
     ./crypt.py -d alice.prv secret.cip secret.txt
     ```

   - Use the following as a content of secret.txt:
     *Cryptography, or cryptology, is the practice and study of techniques for secure communication in the presence of adversarial behavior.*
   - To encrypt a file, the program must generate a random key *K* for AES-128 using random.SystemRandom or os.urandom(), use the key *K* with AES-128 to encrypt the data from

the input file, use RSA with the public key file specified on the command line to encrypt *K* (we refer to the encrypted *K* as *K'* in the following), and write the encrypted data and *K'* to the output file. The format of the output file (how to store *K'* along with the encrypted data) is your choice.
- To decrypt a file, the program must read the encrypted data and *K'* from the input file, RSA-decrypt *K'* to recover the key *K,* use *K* with AES-128 to decrypt the data, and write the decrypted data to the output file.
- You must write the RSA code; you may *not* import RSA code from another module such as PyCrypto.
- You can use a library to implement AES-128 encryption.
- You must choose an appropriate mode of operation for AES-128.
- It's OK to use direct RSA for this assignment. It is never OK to use direct RSA in practice!

In addition to the two Python programs, you must provide a written description of the design of your programs and a screen capture of a session demonstrating that your programs work. For example, a screen capture of the following sequence of commands would be sufficient (this assumes the files message.txt and message2.txt already exist):

```
./genkeys.py alice

./genkeys.py bob

cat message.txt

./crypt.py -e bob.pub message.txt message.cip

cat message.cip

./crypt.py -d bob.prv message.cip message.txt

cat message.txt

cat message2.txt

./crypt -e alice.pub message2.txt message2.cip

cat message2.cip

./crypt -d alice.prv message2.cip message2.txt

cat message2.txt
```

Include a graphic representation describing the flow of the system as implemented by your code.

## Assignment Submission

Submit the assignment on DEN D2L. The submission will consist of three files:

1. A design document (DOC, DOCX, or PDF) providing a brief description of the design of your programs and including a screen capture of the working programs as described above.

2. The program genkeys.py.

3. The program crypt.py.

**Grading**

1. Design document (20 points)
2. Correct implementation of genkeys.py (40 points)
3. Correct implementation of crypt.py (40 points)