Final Project: Deliverable 2, Instructions

Implement the block-based asymmetric key cryptography system RSA with both encryption and decryption components. The specification for RSA is laid out in our textbook1, the Module 4 lecture notes in our course2, and the official RSA specification3.

Assume the inputs to your code are a plaintext file name, a private key file name, a public key file name, and a cipher text file name. **We will be using a 512-bit value for n**, so the RSA encryption is considered to be 512-bit strong encryption. The format of the public or private key file is as follows

n given as 128 hexadecimal digits

new line

e or d given as hexadecimal digits

Sample key pairs are available in the files keyPub1.txt, keyPri1.txt, and keyPub2.txt, keyPri2.txt. *Note that the keys are provided as hexadecimal digits, unlike the DES assignment. You can use methods from StringConversion.py to convert them to binary “strings”. Similarly, the inputs are expected as ASCII strings. Again, you can use methods from StringConversion.py to convert them to binary “strings”.*

**You are given a stubbed-out file RSA.py that you must complete, following the commented instructions provided in it. There are also helper functions you can use in the StringConversion.py file.**

Specifically, you must provide the following:

1. The method RSA\_encrypt(**plaintextFileName**, **publicKeyFileName**, **ciphertextFileName**) that encrypts the alphabetic text in the file named by the first parameter with the public key in the file named in the second parameter. The encrypted text is output to the file named by the third parameter. You will need to use the repeated squares method of exponentiation modulo n.
2. The method RSA\_decrypt(**ciphertextFileName**, **privateKeyFileName**, **plaintextFileName**) that decrypts the text in the file named by the first parameter with the private key in the file named in the second parameter. The decrypted text is written to the file named by the third parameter. Again, you will need to use the repeated squares method of exponentiation modulo n.
3. Test cases for your system and instructions for how to get these test cases to pass on your codebase.
4. Finally, review the attached rubric for this deliverable, and for each performance criterion, please self-evaluate your work by describing whether you completely or partially met expectations, or did not meet expectations, on this criterion.

Format

Please submit your answers to items 1, 2 and 3 as a zip file containing the complete codebase for your implementation, including test cases. For item 5, please provide a readme file with the required self-assessment.

Please keep in mind:

* Your code should run correctly and be implemented efficiently.
* Proper indentation and variable naming should be followed. Use meaningful variable names.

1. *Understanding Cryptography, Christof Paar and Jan Pelzl, Chapter 7*
2. CYB 710: Module 4 Notes
3. [*https://tools.ietf.org/html/rfc8017*](https://tools.ietf.org/html/rfc8017)

CYB 710: Introduction to Cryptography