Report for problem set 2

1. Encryption

The program first judge whether the given file is a text file by file extensions. For text file we use "r" and "w" parameters to read and write. For other files we use "rb" and "wb" to read and write. The reason for this design is that it is different to read binary files and text files by python ‘s open() function.

Then we use AES algorithm to encrypt the file. The reason for this step is decrypting symmetric encrypted message is much faster than decrypting asymmetric encrypted long messages. This improves the speed of encryption and decryption. Further, AES is strong as a symmetric encrypting algorithm. In AES encryption, I use the key size of 128 bits (16 bytes), which proves to be a secured key size. The mode that I choose is CTR. There are several reasons for this design:

1. It computes different blocks at same time, enabling high time efficiency
2. The preprocessing of this algorithm doesn’t need the plaintext, thus improving the throughput of the program.
3. It is proved to be safe just like other modes.
4. The application doesn’t need padding.
5. It enables concurrent encrypting for different blocks.

Encrypt the symmetric key created by AES by destination public key using RSA algorithm. RSA is much safer than AES but consumes longer time than symmetric algorithms. It tends to encrypt smaller contents such as keys.

Sign the message (encrypted_symmetric_key + cipher_text) by sender_private_key to create the signature. This design enables the destination to confirm that it received all the contents that it needs to decrypt the message.

Finally add the signature, encrypted symmetric key, IV, encrypted text, binary_flag encoded by b64 to the cipher_file waiting to be decrypted. b64encode would not create none ASCII symbols. So we could split these parts in decryption parts. For Cryptography, decrypting the AES needs IV, symmetric key and content to solve the message received. binary_flag shows the way we write back to the file. Since no additional file could be sent, we include these parts in the cipher_text.

2. Decryption

After receiving the cipher_file, we first divide them into five parts: signature, encrypted symmetric key, IV, encrypted_text and binary_flag.

We verify the signature by sender public flag and sign_message(encrypted_symmetric_key + cipher_text) retrieved. Then we decrypt the AES symmetric key using dest_private_key and IV.

Use the decrypted symmetric key to decrypt the text we want. Finally, if the binary_flag == 1 then use "wb" to write the output file. Else use "w" to write.