NIS ASSIGNMENT TASK 1 Daniel Collins CLLDAN008

This document will describe and show an example of how symmetric key distribution can be done using asymmetric cryptography. The screenshots provided show Task1A on the left and Task1B on the right.

Both parties use RSA encryption, which is secure based on the assumption of the difficulty of factoring large numbers. So initially both parties will have a public and a private key. The public key is used to encrypt messages and can be used by anyone who has access to it to encrypt and send messages to the owner of the associated private key. The owner of the private key is the only person who can decrypt these messages.

Once parties have exchanged public keys and have verified that the other is the true owner of their associated private keys, a symmetric secret key is exchanged. First, party A will generate and encrypt the secret key with B’s public key and send that message to B. Party A will then sign the same message with their private key to prove that the message was sent by them. Party B will then check this signed message using A’s public key. This ensures that it was truly A that sent the secret key.

Both parties also use OAEP (Optimal Asymmetric Encryption Padding), which is a padding scheme [proven secure](https://cseweb.ucsd.edu/~mihir/papers/oae.pdf) against several attack types, including chosen ciphertext attack and chosen plaintext attack. It also prevents partial decryption or other information leakage. This is the [recommended padding algorithm](http://www.daemonology.net/blog/2009-06-11-cryptographic-right-answers.html) for RSA encryption, according to the Python Cryptography library. This padding scheme using the SHA256 hashing algorithm ensures that messages cannot be tampered with in transmission.

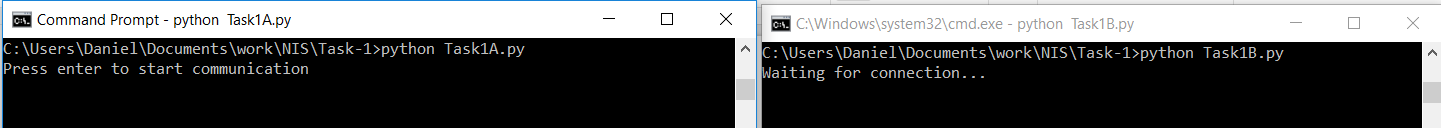
This padding also ensures that repeatedly sending the same encrypted message does not give an attacker any advantage.

For the signing of the secret key by party A, PSS (Probabilistic Signature Scheme) is used. This scheme is often used for verifying RSA signatures.

# Screenshots and Descriptions

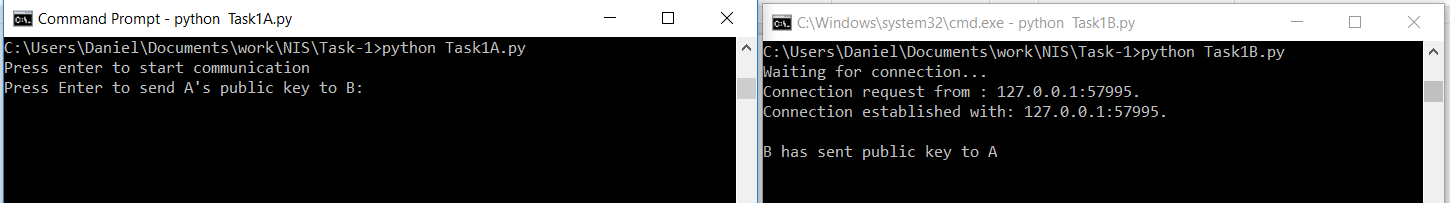
Step 1:

Prompt user to start the communication



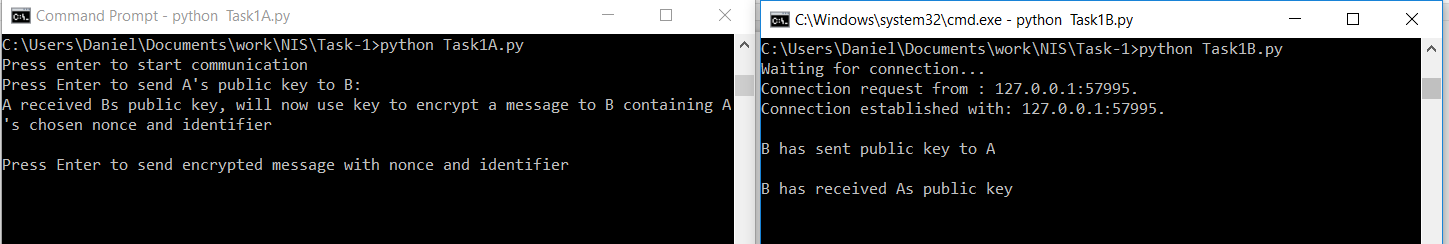
Step 2:

B accepts the communication requests and starts a new thread to handle communication with A. B then immediately sends A a message containing B’s public key. A then sends B its public key.



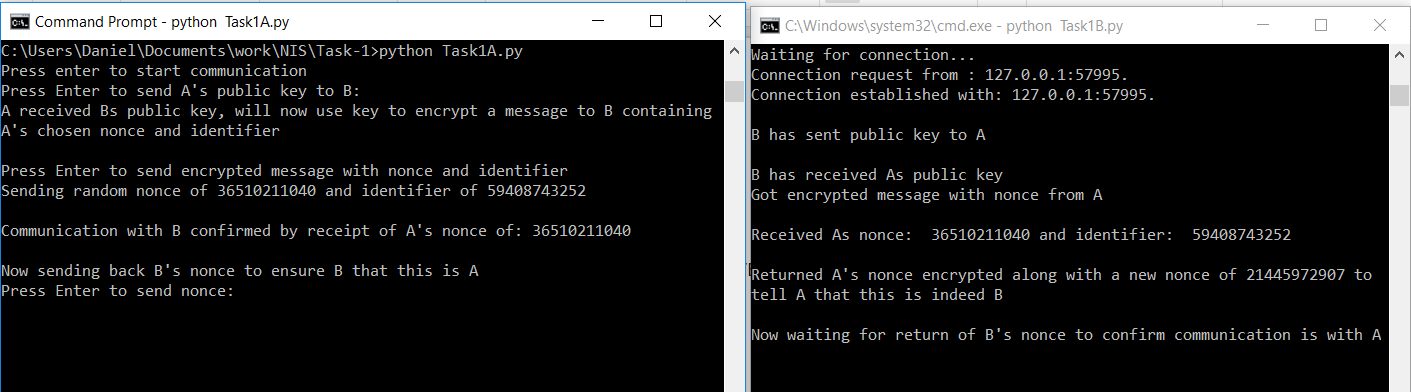
Step 3:

A receives Bs public key, and chooses a random number, the nonce, and an identifier, which is also random for the sake of this assignment. A then concatenates (with a colon as a delimiter) and encrypts these two numbers with B’s public key and sends this to B.



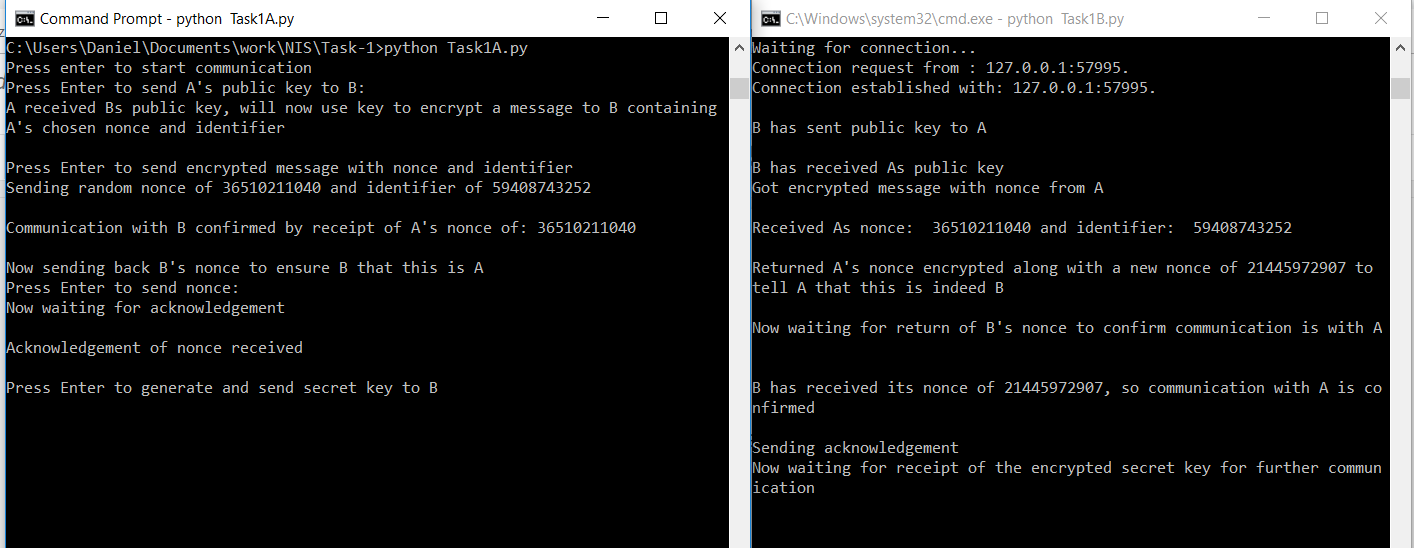
Step 4:

B then decrypts this message with its private key and saves the nonce and identifier. B then chooses its own random number as a nonce, concatenates this nonce with As nonce as done previously, and encrypts this with As public key. This message is then sent to A. A receives this message and decrypts it, and checks if the nonce in the first half of the message is the same as the one it previously sent. If it is the same, then this confirms that it is communicating with B, because only B would have been able to read the nonce A sent by decrypting it with its private key. A then encrypts Bs given nonce with Bs public key and sends this to B. This will tell B that is it truly communicating with A, because only A would be able to decrypt the message with its private key to read the nonce.



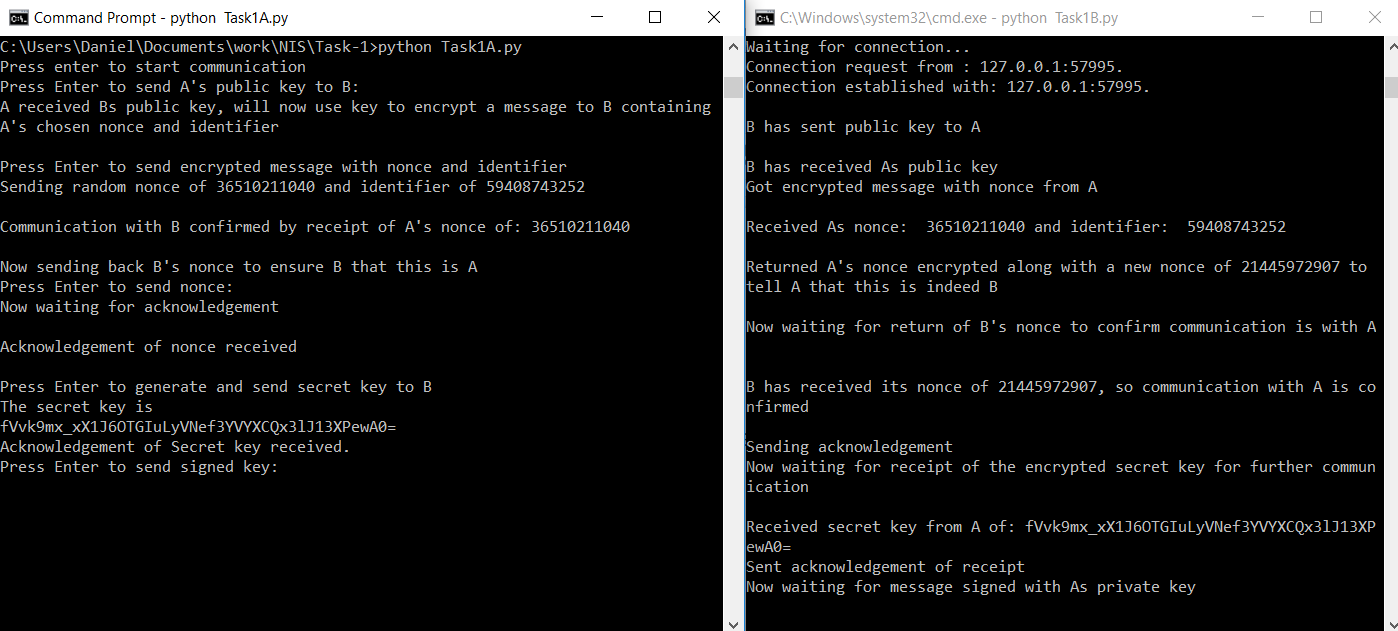
Step 5:

B decrypts the message containing its nonce and compares it to its previously chosen nonce, and if they match then communication with A is confirmed. B then sends acknowledgement of the nonce back to A.



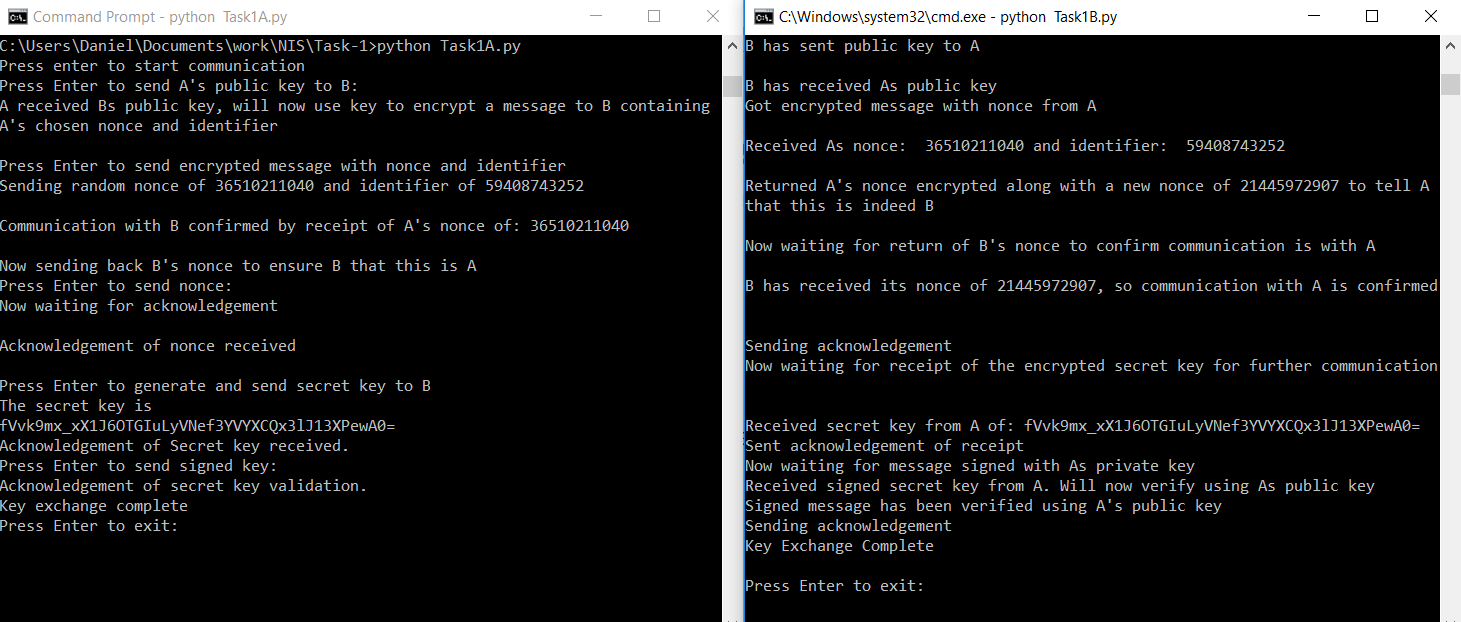
Step 6:

A then generates a random secret key and encrypts it with Bs public key, and sends this to B. B then decrypts this message with this private key to read the secret key and saves this key. This key is safe because only B can decrypt this message to read the secret key. B then sends acknowledgement of receipt back to A and waits for A to send the signed private key.



Step 7:

A then signs the message encrypted with Bs public key containing the secret key with As own private key. This signed message is then sent to B. B then uses As public key to verify that the message was signed with As private key. Only A has access to its private key, so this ensures B that the secret key was sent and signed by A. The secret key is now known by both parties and it has been proven that A was the sender of the key, and B was the receiver. B then sends acknowledgement of the verification to A.



Now that both parties have the same secret key they can use this key to symmetrically encrypt and decrypt messages securely.