Information Security
Project 1

-Akhil Pragallapati
-Siva Sarvana Kashyap Pathiki

Our code provided consists of a Python program for encrypting a file using symmetric-key cryptography, we have used Fernet encryption and sent the encrypted file to a remote server(another laptop in this case) using Secure Shell (SSH) protocol.

Without the key, Fernet promises that a message that has been encrypted with it cannot be altered or deciphered. Symmetric authenticated cryptography commonly referred to as "secret key" cryptography, is used by Fernet.

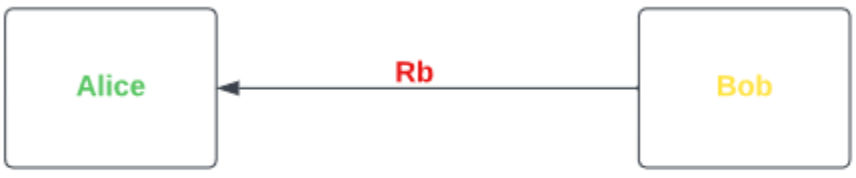
The program begins with Alice code by importing the necessary libraries like paramico and randint and defining the send_file_data function, which we will use later to send files to the remote server(another laptop) via OpenSSH. The function takes the parameters such as file_name, IP_address, username, and password as arguments.

In this function, we first establish an SSH connection to the remote server using the paramiko library and the provided username and password. It then reads the contents of the specified file, uses the SFTP object to upload the file to the remote server (another laptop) in the specified path, and closes the connection.

Next, the program then generates a random number for Alice (Ra_alice) and writes it to a file named Ra_akhil.txt. This file is also sent to the remote server using the send_file_data function.

Ra : Alice's Rnadom Number

Now in Bob's code, a random number for Bob(bob_rand_int) and writes it to a file named bob_rand_int.txt. This file is also sent to the remote server using the send_file_data function.



Rb : Bob's Random Number

The program generates a private key for Bob (B) and Alice (A) using the shared prime number, base, and Bob's random number. Both Alice and Bob now compute the shared secret key (KS)(shared_secret_key) using the Diffie-Hellman key exchange algorithm.

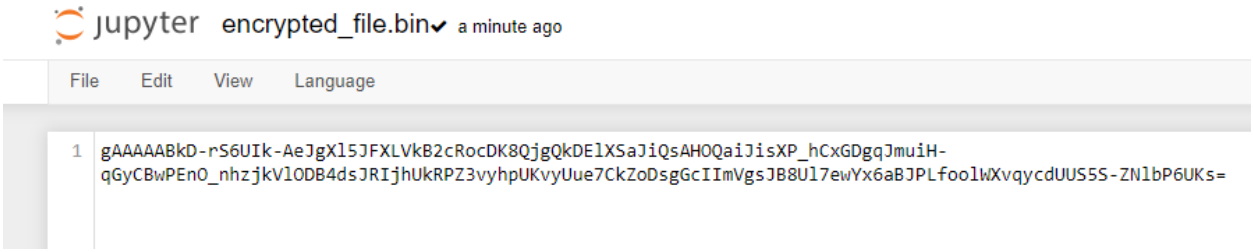
Now Alice's program uses the shared secret key to encrypt a file named file_to_encrypt.txt using Fernet encryption and writes the encrypted data to a new file named encrypted_file.bin. The encrypted file is then sent to the remote server using the send_file_data function.

Finally, Bob's program uses the shared secret key to decrypt the file named ecrypted_file.txt using Fernet encryption and writes the decrypted data to a new file named decrypted_file.bin.

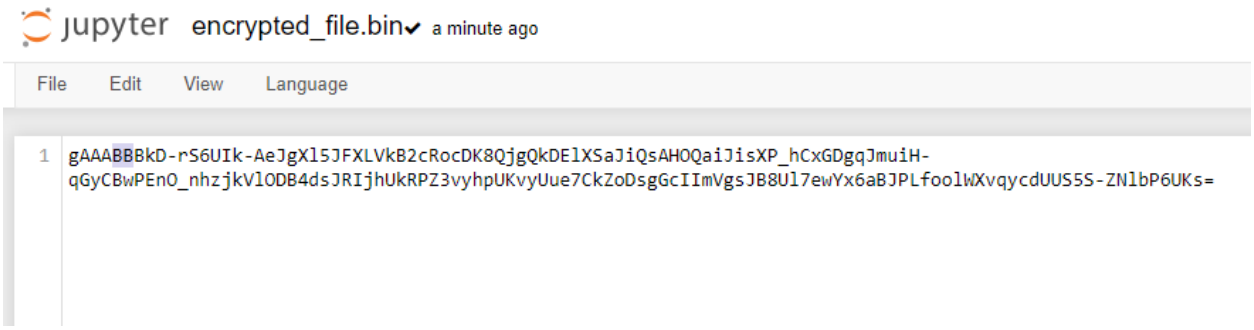
Overall, this code demonstrates an implementation of symmetric-key cryptography for file encryption and Secure Shell (SSH) protocol for secure file transfer.

Scenario I:

Trudy changes the value in the encrypted file and sends the modified encrypted file to bob. Now, Bob tries to decrypt the modified encrypted file with the shared secret key and receives an error, which shows that Integrity is achieved.



Now the changed value is:



After trying to decrypt the encrypted file with changed values, the program throws an error.

```
    # decrypt the data
    decrypted_data = f.decrypt(encrypted_data)
    # write the decrypted data to a new file
    with open("decrypted_file.txt", "wb") as out_file:
        out_file.write(decrypted_data)
print("File is decrypted Successfully !")
```

```
---------------------------------------------------------------------------
InvalidSignature                          Traceback (most recent call last)
~\anaconda3\lib\site-packages\cryptography\fernet.py in _verify_signature(self, data)
    126         try:
--> 127             h.verify(data[-32:])
    128         except InvalidSignature:

~\anaconda3\lib\site-packages\cryptography\hazmat\primitives\hmac.py in verify(self, signature)
     71         ctx, self._ctx = self._ctx, None
---> 72         ctx.verify(signature)

~\anaconda3\lib\site-packages\cryptography\hazmat\backends\openssl\hmac.py in verify(self, signature)
     84         if not constant_time.bytes_eq(digest, signature):
---> 85             raise InvalidSignature("Signature did not match digest.")

InvalidSignature: Signature did not match digest.

During handling of the above exception, another exception occurred:

InvalidToken                              Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1904\3449532776.py in <module>
     26     encrypted_data = file.read()
     27     # decrypt the data
```

Scenario II:

Trudy knows Ra and Rb (random integers from Alice and Bob) from communication between Alice and Bob and tries to compute the shared secret key. As Trudy does not know the base and prime numbers, it is impossible to generate the shared secret key.

Scenario III:

Trudy performs MiM attack by replacing Ra with Rt and sends it to Bob and the replaces Rb with Rt and sends it to Alice. This will not help Trudy to generate the shared symmetric key as Trudy does not know either Bob or Alice private key which are used in creating the shared symmetric key.

Now since Trudy knows Ra and Rb, but still without the private keys of A and B respectively. Trudy can’t determine the shared_secret_key