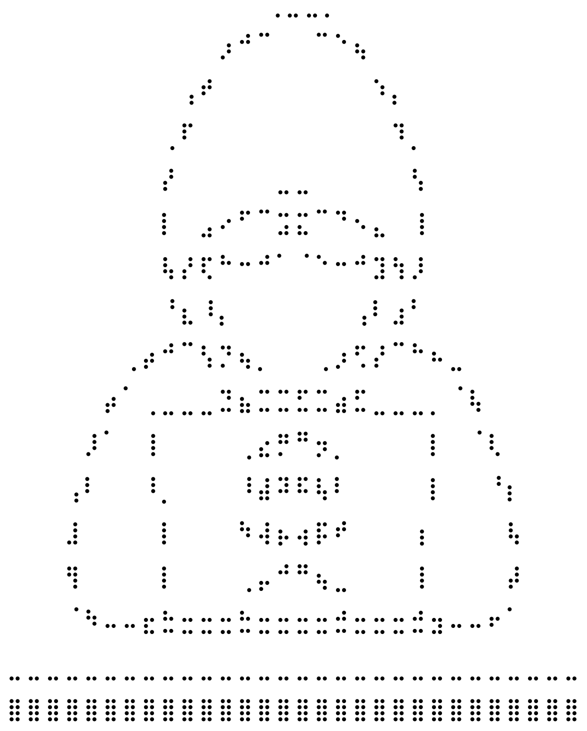
Computer and Network Security, Spring 2023
Project : Ransomware

Team Name: Malware Maestros
Team Number: 37

Final Project Report

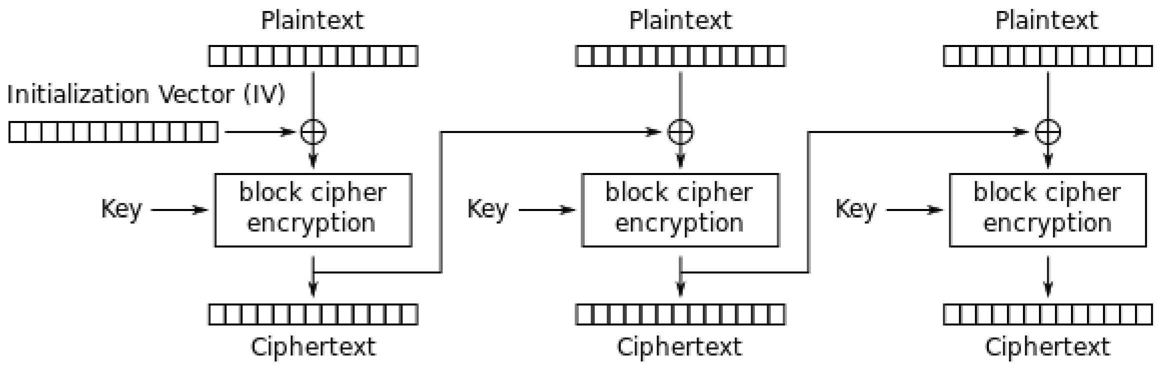


Hadeer ElHussein	**46-0280**
Mariam ElOraby	**46-3294**
Rawan Reda	**46-13489**
Rowan Amgad	**46-5055**

We develop a ransomware using Python, through AES and RSA encryptions and client-server architecture. The following are the steps of our implementation that happen at each of the client and the server. Block diagrams are present at the end to illustrate the process more clearly.

The Client:

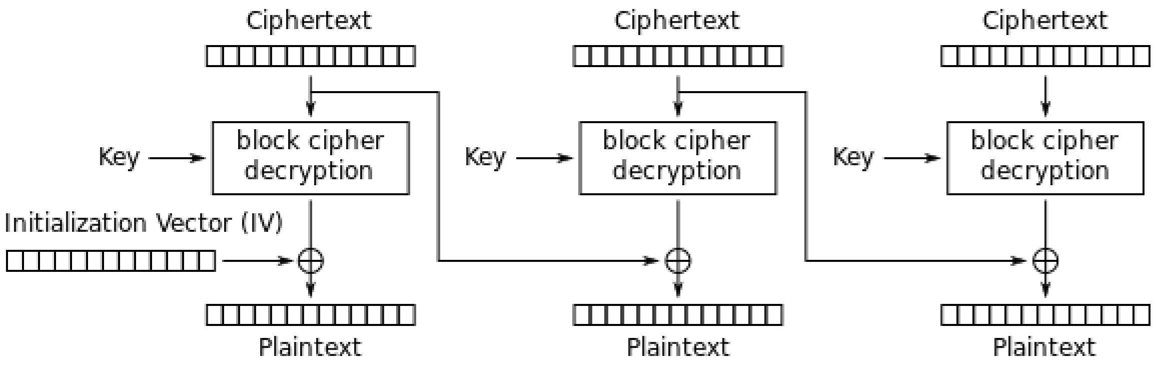
1. Generates a random 128-bit key to be used for AES.
2. Gets all text files and encrypts using `find_txt()`:
   a. Generates a random IV for AES-CBC mode.
   b. Searches for all .txt files in the user's "Documents" directory and its subdirectories. It then calls the `aes()` function on each `.txt` file found, passing in the AES key, IV, and file path as arguments. This encrypts all `.txt` files in the "Documents" directory and its subdirectories using the AES encryption algorithm with the random key and IV.
3. Encrypts using `aes()`:
   a. Takes an AES key, IV, and a file path as input. The function uses the `pycryptodome` library to encrypt the contents of the input file using the AES encryption algorithm in CBC (Cipher Block Chaining) mode.
   We choose AES-CBC mode due to its efficiency.



Cipher Block Chaining (CBC) mode encryption

   b. We convert the AES key from string format to bytes format using the `encode()` method. Then we write the IV to a file named "`IV.txt`".
   c. The function reads the contents of the input text file in binary mode and pads the plaintext to a multiple of the block size using null bytes (`b“\0”`). The block size is 128 bits (16 bytes) for AES-128.
   d. The function then creates an AES cipher object using the key and IV, and uses it to encrypt the plaintext. The encrypted data is written back to the input file, overwriting the original content.
4. Receives the RSA public key from the server, encrypts the AES key using RSA encryption (`rsa_Encode()` function) and sends it to the server.

5. Waits for the user's input to decrypt the files (assuming he paid the ransom 😈💰) :
    a. Request the unencrypted AES key from the server.
    b. `find_txt_decode()` uses the unencrypted AES key to decrypt all text files in the opposite manner of `find_txt()`.



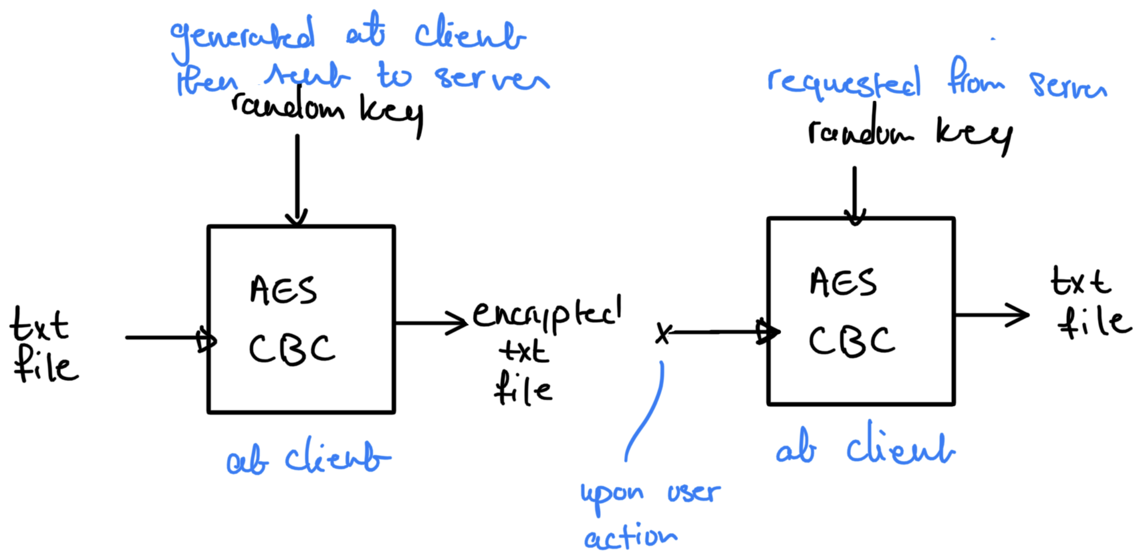
Cipher Block Chaining (CBC) mode decryption

The Server:

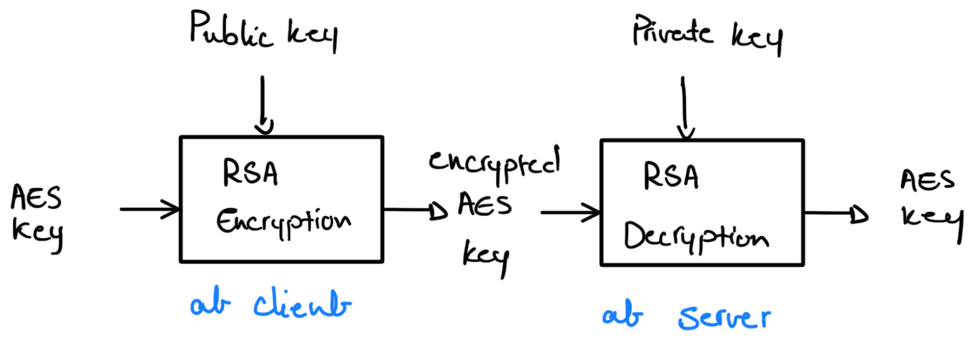
1. Access the .csv file of users' data, extract their emails and send them an email with the .exe of the virus. We achieved this using normal pandas database operations in `get_emails_list()` to extract the emails. `send_emails()` sends an email to each email address in the list returned by `get_emails_list()`. The email contains a message and a .exe file. The function uses the `smtplib` library to connect to an SMTP (Simple Mail Transfer Protocol) server and send the email message.
2. `rsa_generatePublicPrivate`(): generates a new RSA public-private key pair with a key length of 1024 bits. We use the `pycryptodome` library to generate the key pair, specifically the `Crypto.PublicKey` module.
    a. Generate an RSA key pair using the `RSA.generate()` method with the specified key length. Then, we retrieve the public and private keys in PEM format using the `export_key()`.
    b. Save the public and private keys in a single file named `keyPair.key`. We store the public key first, followed by the "@" symbol as a separator, and then the private key. This allows us to store both keys in a single file.
3. Sends the RSA public key to the client.
4. Receives the encrypted AES key generated at the client.
5. Upon request, the server decrypts the AES key using the private RSA key, and sends it back to the client.

Block diagrams:

AES encryption and decryption:



RSA encryption and decryption:



Public and private keys are generated on the server side, and the public key is requested on the client side.