Lab Report 2

Due: 10/31/24

Adrian Nelson Aidan Chin

Code

RSA:

- We first implemented the function `generate_keypair(p,q)` which calculated the n, eulers_totient, preset the e to 65,537 (Fermat Prime with base 2), and called `get_d(e, eulers_totient)`.
- Our `get_d(e, eulers_totient)` has two loops, the first runs the standard Euclidean algorithm. The second then uses the lists created by the first, and reconstructs the coefficients so we have: `original num*x + original modulo*y = GCD(e, euler's totient)`. This function then adjusts the x value to be within the range of `modulo_list[-1]`, ensuring a positive result.

DES:

- We implemented the `substitute(self,d_e)` function by iterating through the `subblock` and passing `block` to `compute_s_block(block, round)`, with `round=0` being defined before iterating.
- In `compute_s_block(block, round)`, we first take the 6-bit `block` and do bit manipulation to `OR` the first and sixth bits to find the row. Next we do bit manipulation to `OR` the second through fifth bits to get the column. The `row` and `column` values used as coordinates to get the correct `box_value`, which we take and turn into a 4 bit binary number using `bin_value(box_value,4)`.
- Going back to `substitute(self,d_e)` function with our 4 bit binary value, we use `result.extend(sbox_output)` to get all the values combined into a 256 bit long list, then we call `string_to_binary(result)` to change the result from a list of strings to a binary value.

Question 1 (5 points): Give a short description for each of the 5 network security principles: **C**onfidentiality, **I**ntegrity, **A**uthentication, **A**ccessibility, and **A**vailability

- Confidentiality
  - Both the sender and the receiver should be the only recipients that can understand the transmission, sender encrypts and receiver decrypts
- Integrity
  - The sent message is verifiably not edited in transit by a third party
- Authentication
  - The sender can confirm the identity of the receiver and the receiver can confirm the identity of the sender
- Accessibility

    - Service must be accessible to all users, for example the security (i.e., encryption and decryption) should be able to work on all different types of hardware

  Availability

    - Service must be made available to use for all users

- Question 2 (3 points): Explain playback attack in the network with an example. How does the use of a random Nonce prevent such an attack?

  A playback attack is when someone intercepts a transmission and plays it back again later maliciously. For example, if someone sends a transaction to a bank, and it gets intercepted, the attacker later could send it again and make the person make another transaction unknowingly. A random nonce helps prevent this as it is a random string of numbers that serve as a one time use transaction ID and if the re-sent transaction matches that ID, the bank knows not to process it.

- Question 3 (2 points): What is the difference between public key encryption and private key encryption?

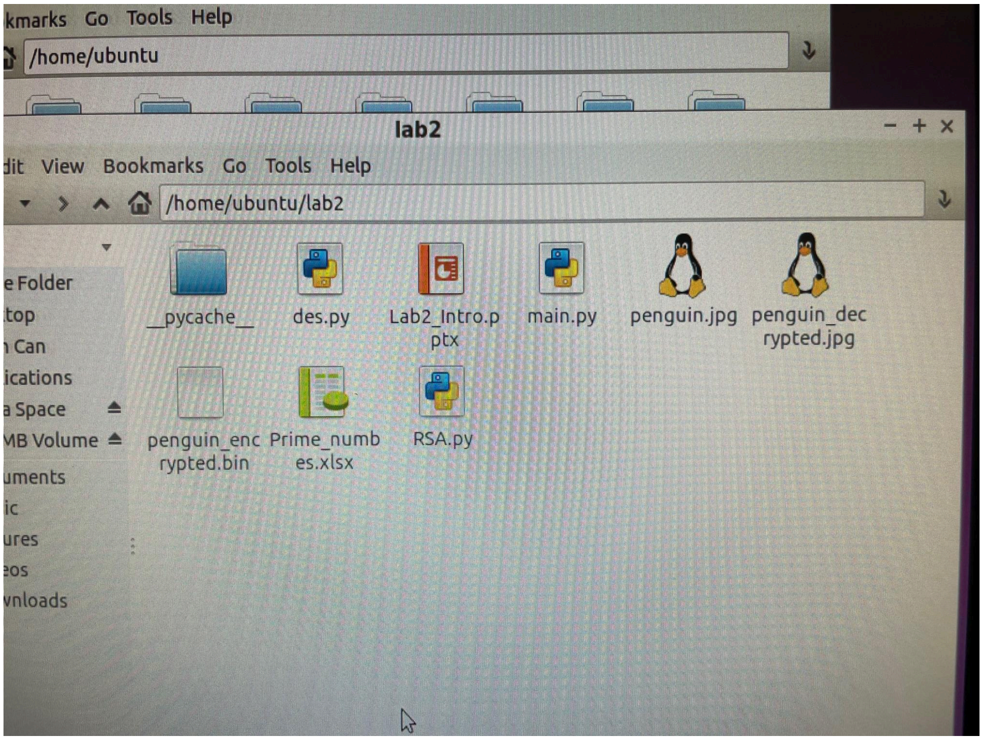
  Public key encryption uses a pair of symmetric keys, where one is private, and the other is public. The public key encrypts the data and the private key decrypts the data. Symmetric keys are slower, but also more secure as only one of the keys is publicly accessible (This type of key is used in RSA).

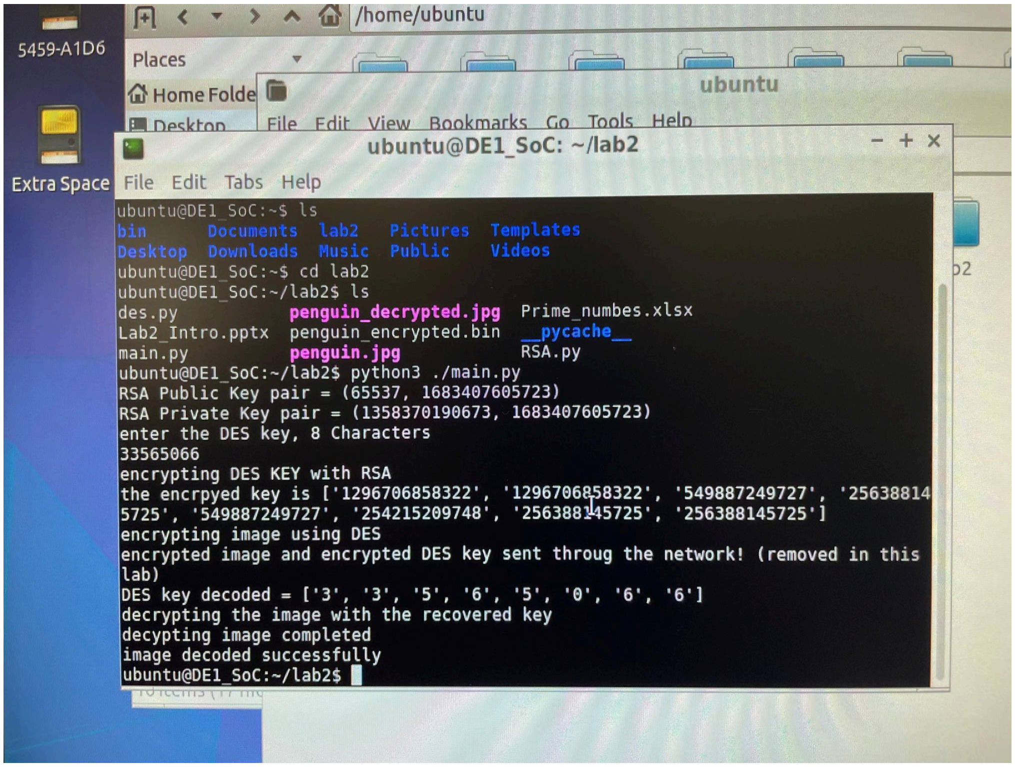
  Private key encryption is when there is one key that encrypts and decrypts the data. It is quick and not computationally heavy, but the one key needs to be securely transferred between users (This type of key is used by DES).

- Question 4 (2 ponts): Why did we use RSA for encrypting the key and not the whole image, and instead used DES for the image?

  The reason we RSA for the key and DES for the image is because DES is fast and suitable for large amounts of data (like an image). RSA is relatively slower in comparison, and that would make encryption and decryption non-accessible to those who don’t have high computing power. Instead, we make DES more secure by encrypting the key using RSA, then we get the best of both methods.

Images below:





```
ubuntu@DE1_SoC:~$ ls
bin       Documents  lab2   Pictures  Templates
Desktop   Downloads  Music  Public    Videos
ubuntu@DE1_SoC:~$ cd lab2
ubuntu@DE1_SoC:~/lab2$ ls
des.py            penguin_decrypted.jpg  Prime_numbes.xlsx
Lab2_Intro.pptx   penguin_encrypted.bin  __pycache__
main.py           penguin.jpg            RSA.py
ubuntu@DE1_SoC:~/lab2$ python3 ./main.py
RSA Public Key pair = (65537, 1683407605723)
RSA Private Key pair = (1358370190673, 1683407605723)
enter the DES key, 8 Characters
33565066
encrypting DES KEY with RSA
the encrpyed key is ['1296706858322', '1296706858322', '549887249727', '25638814
5725', '549887249727', '254215209748', '256388145725', '256388145725']
encrypting image using DES
encrypted image and encrypted DES key sent throug the network! (removed in this
lab)
DES key decoded = ['3', '3', '5', '6', '5', '0', '6', '6']
decrypting the image with the recovered key
decypting image completed
image decoded successfully
ubuntu@DE1_SoC:~/lab2$
```