CSCI 373 - Cybersecurity
Homework 2
Cryptography

Objective: Learn how to implement symmetric and asymmetric encryption techniques. Learn how to use OpenSSL to encrypt/decrypt using public/private keys. Learn how to digitally sign a document and verify the signature.

References:

- Read Chapter 2 and 12 in the "Security in Computing" textbook.
- http://rumkin.com/tools/cipher/
- Elliptic Curve Diffie Hellman https://www.youtube.com/watch?v=F3zzNa42-tQ
- https://raymii.org/s/tutorials/Encrypt_and_decrypt_files_to_public_keys_via_the_OpenSSL_Command_Line.html
- https://raymii.org/s/tutorials/Sign_and_verify_text_files_to_public_keys_via_the_OpenSSL_Command_Line.html

Task 1: Caesarian Shift

Create a program to implement the Caesarian Shift substitution cypher.

1. Use the character set of ASCII code 32 (space) through 126 (~), for a total of 95 characters. Your program should allow the new line character, "\n" to remain unencrypted.
2. Use 19 for the offset key.
3. Read in the file "task1_encrypted_message.txt", decrypt the message, save it to a files named "task1_decrypted_message.txt". Upload that file.
4. Upload your code in a file named "task1.py".

Task 2: Affine Cipher

Create a program to implement the Affine substitution cypher.

1. Use the same character set as task 1.
2. Use a=13 and b=7 as the multiplier (a) and offset (b) keys.
3. Read in the file "task2_encrypted_message.txt", decrypt the message, save it to a files named "task2_decrypted_message.txt". Upload that file.
4. Upload your code in a file named "task2.py".

Task 3: Elliptic Curve Diffie Hellman key exchange

Create a program to implement the Elliptic Curve Diffie Hellman key exchange.

1. Review the video on Elliptic Curve Diffie Hellman
2. Working on the space of integers mod 17:
    a. Write a function to compute point doubling: R = G+G
    b. Write a function to compute the addition of two points: R = P + Q

    c. Write a function to compute scalar multiplication: R = kG (note, the first step is point doubling, and then addition there after). Remember to use "k" modulus "n" (order of G)
3. Working with the parameters: G= (5,1) p=17 a=2, b=2 n=19
    a. Assume "Bob" selected the secret number "beta"= 2. What is the point B?
    b. Assume "Alice" selected the secret number "alpha"= 18. What is the point A?
    c. Show that Bob and Alice will get the same point P, even though Bob does not know alpha and Alice does not know beta, but they both know A and B?
4. Use the X-coordinate of P as the multiplier (a) and Y-coordinate of P as the offset (b) keys for your Affine Cipher from Task 2: decrypt the file "task3_encrypted_message.txt"
5. Upload you code as "task3.py"

Task 4: File encryption with OpenSSL

Using "openssl" program:

1. Decrypt "task4_encrypted_message.dat" using the private key "task4_private.pem" and the encrypted password/key "key.bin.enc"
2. Generate your own public/private key pair
3. Using your public key, encrypt the message you descripted in step 1. Name the resulting file "task4_message.txt.enc" along with the encrypted password/key "mykey.bin.enc".
4. Create a text file named "task4.txt", describe the steps you took to complete the above process. Submit this file, your encrypted file (step 3) and both of your keys (step 2). **Note:** that I will need to decrypt your file, so provide instructions on how to do that in your "task4.txt" file.

Task 5: Verify and Sign a file with OpenSSL

Using "openssl" program:

1. Determine which of the following three files: (task5_message1.txt, task5_message2.txt, task5_message3.txt) are signed by the private key paired with the public key "task5_public.pem", verifying with the file "task5_message_sig.sha256"
2. Sign the file from step 1 with the private key you submitted in task 4, name the signature file "task5.sig"
3. Create a text file named "task5.txt", describe the step you took to complete the above process. Submit this file, your signature file, and the correct message file.