Lab 1: Crypto Basics

Hayder Sarhan

Task 1

Write a program that implements **one** of the following algorithms for both encryption and decryption.

- Substitution Cipher
- Transposition Cipher
- Rotor Machines or Simple XOR

  Substitution Cipher:

  - Using the algorithm with `key=3`:

```
moze@Kirby:~/Documents/FunInfSec/Lab01$ python3 SubstitutionCipher.py
Encrypted message: Khoor wkhuh
Decrypted message: Hello there
```

  - Using the algorithm with `key=4`:

```
moze@Kirby:~/Documents/FunInfSec/Lab01$ python3 SubstitutionCipher.py
Encrypted message: Lipps xlivi
Decrypted message: Hello there
```

Task 2

Hide data (e.g text) in another data file format. The data should be hidden in the following formats

- Video

  For this task I used the tool videostego :

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ git clone https://github.com/JavDomGom/videostego.git
Cloning into 'videostego'...
remote: Enumerating objects: 62, done.
remote: Counting objects: 100% (62/62), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 62 (delta 14), reused 55 (delta 10), pack-reused 0 (from 0)
Receiving objects: 100% (62/62), 3.20 MiB | 655.00 KiB/s, done.
Resolving deltas: 100% (14/14), done.
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ ls
audio.wav  task2.txt  video.mp4  videostego
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ cd videostego/
moze@Kirby:~/Documents/FunInfSec/Lab01/task2/videostego$ ls
img  LICENSE  Makefile  mp4  README.md  src  videostego_doc_v0.03.pdf
moze@Kirby:~/Documents/FunInfSec/Lab01/task2/videostego$ sudo make install
[sudo] password for moze:
2024/09/02 11:46:47 [build] Building videostego binary ...
gcc -Wall -g -o videostego src/*.c
2024/09/02 11:46:48 [build] Done!
2024/09/02 11:46:48 [install] Installing videostego ...
install -m 0755 videostego /usr/local/bin
2024/09/02 11:46:48 [install] Done!
moze@Kirby:~/Documents/FunInfSec/Lab01/task2/videostego$
```

Following the instructions from the manual:

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ ls
audio.wav  task2.txt  video.mp4  videostego
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ ./videostego/videostego -f video.mp4 -w -m "Leave me alone"
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ ./videostego/videostego -f video.mp4 -r
Leave me alone
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$
```

- Audio

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ ls
audio.wav  task2.txt  video.mp4
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ cat task2.txt
Leave me alone
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$
```

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ steghide embed -cf audio.wav -ef task2.txt
Enter passphrase:
Re-Enter passphrase:
embedding "task2.txt" in "audio.wav"... done
```

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ rm task2.txt
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ steghide extract -sf audio.wav
Enter passphrase:
wrote extracted data to "task2.txt".
moze@Kirby:~/Documents/FunInfSec/Lab01/task2$ cat task2.txt
Leave me alone
```

- ICMP/DNS

  For this subtask I used my friends machine as a victim to connect and send messages to it using tunnelshell.

  We ran the following command on my friends machine:

```
anas@anas-Legion:~$ sudo ./tunneld -t icmp -m echo,reply
[sudo] password for anas:
```

  After getting the IP of my friends machine I connected to his machine:

```
moze@Kirby:~/tunnelshell$ sudo ./tunnel -t icmp -m echo,reply [redacted]

Connecting to [redacted]...done.

pwd
/home/anas
```

Task 3

Generate a RSA keypair of key length 2048-bit using OpenSSL. Write your first name in a text file, sign, and verify the integrity of the text file.
Your answer should include these:

- Generate the private key with OpenSSL

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ openssl genrsa -out private_key.pem 2048
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ ls
private_key.pem  task3.txt
```

- Extract the public key from the private key

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ openssl rsa -in private_key.pem -pubout > key.pub
writing RSA key
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ ls
key.pub  private_key.pem  task3.txt
```

- Create a text file that includes your first name

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ ls
key.pub  private_key.pem  task3.txt
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ cat task3.txt
Hayder
```

- Sign the text file with OpenSSL digest (dgst)

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ openssl dgst -sha256 -sign private_key.pem -out task3.txt.sign task3.txt
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ ls
key.pub  private_key.pem  task3.txt  task3.txt.sign
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ cat task3.txt.sign
CQ@ @"@V@@@q1nm@|-#@    ?0@@"D@@@@@n<DF}@@ ƒ@XxN&8@x
?$@g@T□@@+Y@爽@
@@@@@@@^KV@{ @@9@w@:ˬ@@4@@@]N@@,₂@W昌 ?y@@3O@)@~@a@E@`@,W@3M@ZM@@@(:#@qX[]@@#@?@B@j/@@@@~@p@5@@w@&%Q@y@9!=Y@@6*@r@@@.>4"
                                                                                                                                    <>@@LDF;@
9@@@@#@@cmoze@Kirby:~/Documents/FunInfSec/Lab01/task3$
```

- Verify the digital signature using OpenSSL digest (dgst)

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ ls
key.pub  private_key.pem  task3.txt  task3.txt.sign
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ openssl dgst -sha256 -verify key.pub -signature task3.txt.sign task3.txt
Verified OK
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$
```

(reference)

Task 4

Add your last name to the text file from task 3. Now verify the text file by using the previous signature you created for your first name. Is the verification succesful?

```
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ echo "Sarhan" >> task3.txt
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ cat task3.txt
Hayder
Sarhan
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$ openssl dgst -sha256 -verify key.pub -signature task3.txt.sign task3.txt
Verification failure
401769FE797F0000:error:02000068:rsa routines:ossl_rsa_verify:bad signature:../crypto/rsa/rsa_sign.c:430:
401769FE797F0000:error:1C880004:Provider routines:rsa_verify:RSA lib:../providers/implementations/signature/rsa_sig.c:774:
moze@Kirby:~/Documents/FunInfSec/Lab01/task3$
```

It wasn't since we changed the data in the file we're verifying, and the signature we have is dependent on the exact data used to create it.

Task 5

Decode the following ceasar cipher:

> Vwrwbu gcas roho wg ybckb og sbqfmdhwcb. Kvsb dzowb hslh wg
> sbqfmdhsr wh psqcasg ibfsoropzs obr wg ybckb og qwdvsfhslh. Wb o
> Gipghwhihwcb qwdvsf, obm qvofoqhsf ct dzowb hslh tfca hvs uwjsb twlsr
> gsh ct qvofoqhsfg wg gipghwhihsr pm gcas chvsf qvofoqhsf tfca hvs goas
> gsh rsdsbrwbu cb o ysm. Tcf sloadzs kwhv o gvwth ct 1, O kcizr ps
> fsdzoqsr pm P, P kcizr psqcas Q, obr gc cb.

Using the code from task1, I modified it and iterated over multiple values for keys between `1, 19` and found the hidden message when `key=14`

```
moze@Kirby:~/Documents/FunInfSec/Lab01$ python3 CeasarCipher_task5.py > CeasarCipher.txt
moze@Kirby:~/Documents/FunInfSec/Lab01$
```

```
====================13====================
IjejOh TPNf ebUb jT lOPXO bT fOdSZQUjPO. xifO QmbjO UfYU jT fOdSZQUfe jU cfdPNfT VOSfbebcmf b
====================14====================
HidiNg SOMe daTa iS kNOWN aS eNcRYPTiON. wheN PlaiN TeXT iS eNcRYPTed iT becOMeS UNReadable a
====================15====================
GhchMf RNLd cZSZ hR jMNVM ZR dMbQXOShNM. vgdM OkZhM SdWS hR dMbQXOSdc hS adbNLdR TMQdZcZakd Z
====================16====================
FgbgLe QMKc bYRY gQ iLMUL YQ cLaPWNRgML. ufcL NjYgL RcVR gQ cLaPWNRcb gR ZcaMKcQ SLPcYbYZjc Y
```

> HidiNg SOMe daTa iS kNOWN aS eNcRYPTiON. wheN PlaiN TeXT iS eNcRYPTed iT becOMeS UNReadable aNd iS kNOWN aS ciPheRTeXT. IN a sUbSTiTUTiON ciPheR, aNY chaRacTeR Of PlaiN TeXT fROM The giVeN fiXed SeT Of chaRacTeRS iS SUbSTiTUTed bY SOMe OTheR chaRacTeR fROM The SaMe SeT dePeNdiNg ON a keY. FOR eXaMPle WiTh a ShifT Of 1, A WOUld be RePlaced bY B, B WOUld becOMe C, aNd SO ON.