Mastering Cryptography: Foundations & Applications

Practical Tool

GUI BASED

1. **Cryptii.com** (https://cryptii.com/pipes/caesar-cipher)
    a) Ceaser Cipher
2. **Plantencalc.com** (https://planetcalc.com/7751/)
    a) Playfair Cipher

Day-1

3. **HashmyFile**
    a) Hashing
4. **CrackStation** (https://crackstation.net/)
    a) Hash Cracking

Day-2

5. **Cyber Chef** (https://gchq.github.io/CyberChef/)
    a) Symmetric Key Encryption
    b) Hashing
    c) Asymmetric Key Encryption
6. **Advanced Encryption Package** (https://www.aeppro.com/)
    a) Symmetric Encryption
    b) public key generation
    c) email encryption

Day-3

7. **BitLocker**
    Usage:

1)Search Bar -> Local Group Policy Editor -> Administrative Templete -> Windows Components -> Bit Locker Drive Encryption -> Operating System Drive -> Double Click on Require additional authentication at startup -> enable -> Tick Allow BitLocker without a compatible TPM -> OK -> close

2) Control Panel -> System and Security -> BitLocker Crive Encryption -> Turn on BitLocker -> Enter a Password -> Enter and renter -> Next -> Save to a file -> Save -> Next*2 -> Check Run BitLocker system check -> Continue -> Close -> Restart the PC .

By HIMANSHU GOHITE

Day 4 and Day-5

Follow the practical in order for better understanding.

CLI BASED

Openssl : OpenSSL is an open source cryptographic toolkit that facilitates secure communications between endpoints on a network. The toolkit includes three core components: the libcrypto library, the libssl library and a command-line utility for performing cryptographic tasks.

1.Integrity

┌──(kali㉿kali)-[~] **{Creating Folder/Directory}**

└─$ mkdir crypto

┌──(kali㉿kali)-[~] **{Traverse to Directory}**

└─$ cd crypto

┌──(kali㉿kali)-[~/crypto] **{Create a Text file}**

└─$ nano new.txt

┌──(kali㉿kali)-[~/crypto] **{Create the SHA-256 Hash of the file}**

└─$ openssl sha256 -hex -out new.sha256 new.txt

┌──(kali㉿kali)-[~/crypto] **{See the Hash}**

└─$ cat hash.sha256

SHA2-256(hash.txt)= 61fefee5bb8ad37ec62a308d33ec1c8eca06fad43eca74540b878078540f0009

┌──(kali㉿kali)-[~/crypto] **{Edit the Text file}**

└─$ nano new.txt

┌──(kali㉿kali)-[~/crypto] **{Open the Original hash file}**

└─$ cat hash.sha256

SHA2-256(hash.txt)= 61fefee5bb8ad37ec62a308d33ec1c8eca06fad43eca74540b878078540f0009

┌──(kali㉿kali)-[~/crypto] **{Check the SHA256 Hash of Modified File}**

└─$ sha256sum new.txt

f2c2e780ebdc39d3f0b009495dbf29d56c47917805e575ae173f8ddba5c07793 new.txt

As a Result we can see the clear difference in the Modified file's Hash and Original File's Hash…….

2. **Confidentiality**

Symmetric Encryption

┌──(kali㉿kali)-[~/crypto] **{Generate the Secret Key}**

└─$ openssl rand -out secret.key -hex 32

┌──(kali㉿kali)-[~/crypto] **{Check the Secret Key}**

└─$ cat secret.key

17a56ffd08a4c8b6968d2a0c16aa151b83292179cbd2d6870f449543656407fb

┌──(kali㉿kali)-[~/crypto] **{Select the Cipher from List}**

└─$ openssl list -cipher-algorithms

┌──(kali㉿kali)-[~/crypto] **{Encrypt the Original File using Key}**

└─$ openssl aes-256-cbc -in new.txt -out new.enc -e -kfile secret.key

*** WARNING : deprecated key derivation used.

Using -iter or -pbkdf2 would be better.

┌──(kali㉿kali)-[~/crypto] **{Check the Encrypted File}**

└─$ cat new.enc

)82ûz lu1u

┌──(kali㉿kali)-[~/crypto] **{Decrypt the Encrypted File using same Key}**

└─$ openssl aes-256-cbc -in new.enc -out new.dec -d -kfile secret.key

*** WARNING : deprecated key derivation used.

Using -iter or -pbkdf2 would be better.

┌──(kali㉿kali)-[~/crypto] **{Check if the Decrypted file is same as the Original}**

└─$ cat new.dec

Asymmetric Encryption

Part1: Key Generation

┌──(kali㉿kali)-[~/crypto]

└─$ openssl genrsa -out key.pri **{Generate Private Key}**

┌──(kali㉿kali)-[~/crypto] **{Check Private Key}**

└─$ cat key.pri

┌──(kali㉿kali)-[~/crypto] **{Detailed checking of Private Key}**

└─$ openssl rsa -in key.pri -noout -text

—(kali㉿kali)-[~/crypto]

└─$ openssl rsa -in key.pri -pubout -out key.pub **{Get the public key}**

writing RSA key

┌──(kali㉿kali)-[~/crypto] **{Check the public key}**

└─$ cat key.pub

Part2 : Encryption & Decryption

┌──(kali㉿kali)-[~/crypto] **{Encrypting the file}**

└─$ openssl pkeyutl -encrypt -inkey key.pub -pubin -in new.txt -out asymnew.enc

┌──(kali㉿kali)-[~/crypto]

└─$ cat asymnew.enc

",����I�>��V{(�@�H���D��o���<^��<glv�E�\��Z�M�:"�����9�!��:��~��;��,�1����O�ن�ЮǫO

�S���V��h��"�B&<>%C?Z
M���((��i$b9BW~_vMc''+JRPpÕ?ת
G<ضw2瞎vF/W g S=R

┌──(kali㉿kali)-[~/crypto] **{Decrypting the file}**

└─$ openssl pkeyutl -decrypt -inkey key.pri -in asymnew.enc -out asymnew.dec

┌──(kali㉿kali)-[~/crypto]

└─$ cat asymnew.dec

This is the Original.... message

Change

3. Non Repudiation

┌──(kali㉿kali)-[~/crypto] **{ Create a file }**

└─$ nano plain.txt

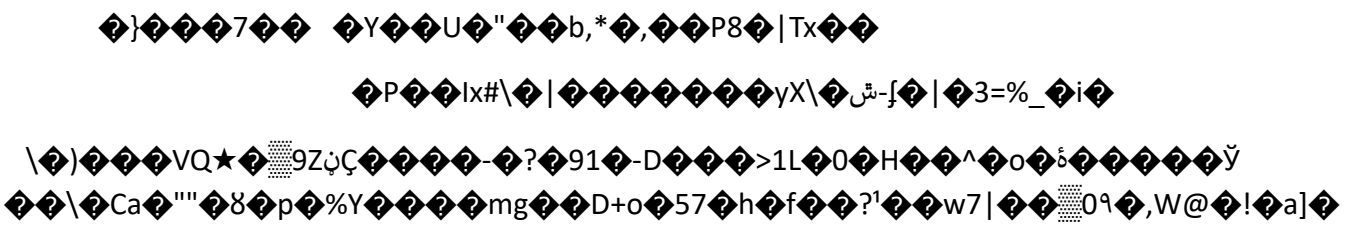
┌──(kali㉿kali)-[~/crypto] **{ Create a sign the file }**

└─$ openssl pkeyutl -sign -inkey key.pri -in plain.txt -out plain.txt.sig

┌──(kali㉿kali)-[~/crypto]

└─$ cat plain.txt.sig

WQ����Ƙ�gi_B��



┌──(kali㉿kali)-[~/crypto] **{ Verify file }**

└─$ openssl pkeyutl -verify -inkey key.pub -pubin -sigfile plain.txt.sig -in plain.txt

Signature Verified Successfully

Hash-Based Signatures

┌──(kali㉿kali)-[~/crypto]

└─$ openssl sha256 -sign key.pri -out plain.txt.hash.sig plain.txt

┌──(kali㉿kali)-[~/crypto]

└─$ openssl sha256 -verify key.pub -signature plain.txt.hash.sig plain.txt

Verified OK

4. Steganography (for this practical download a jpeg file and create a message.txt file in same directory)

```
┌──(kali㉿kali)-[~/Documents]
└─$ mkdir stegano
```

{Create a directory}

```
┌──(kali㉿kali)-[~/Documents]
└─$ cd stegano
```

{Traverse to directory}

```
┌──(kali㉿kali)-[~/Documents/stegano]
└─$ sudo apt install steghide
[sudo] password for kali:
```

{Install steghide tool}

```
┌──(kali㉿kali)-[~/Documents/stegano]
└─$ steghide embed -cf image.jpg -ef message.txt
Enter passphrase:
Re-Enter passphrase:
embedding "message.txt" in "image.jpg"... done
```

{Embed the Secret Message file to Cover file}

```
┌──(kali㉿kali)-[~/Documents/stegano]
└─$ rm message.txt
```

{Remove the Secret Message file}

```
┌──(kali㉿kali)-[~/Documents/stegano]
└─$ steghide extract -sf image.jpg
Enter passphrase:
wrote extracted data to "message.txt".
```

{Extract the Secret Message file from Cover file }

```
┌──(kali㉿kali)-[~/Documents/stegano]
└─$ ls
image.jpg  message.txt
```

{List the Directory}

```
—(kali㉿kali)-[~/Documents/stegano]
└─$ cat message.txt
```

{Open the Secret Message file}