System Security Week 2

Cryptography: DES and AES

Ulster University

Learning Outcomes

In this lab, you will:

- Access and use OpenSSL from the terminal
- Locate and access online sources of OpenSSL documentation
- Apply cryptographic services to:
  - Conceal information within a file (encryption)
  - Retrieve concealed information from a file (Decryption)
  - Verify a file's integrity

Tasks

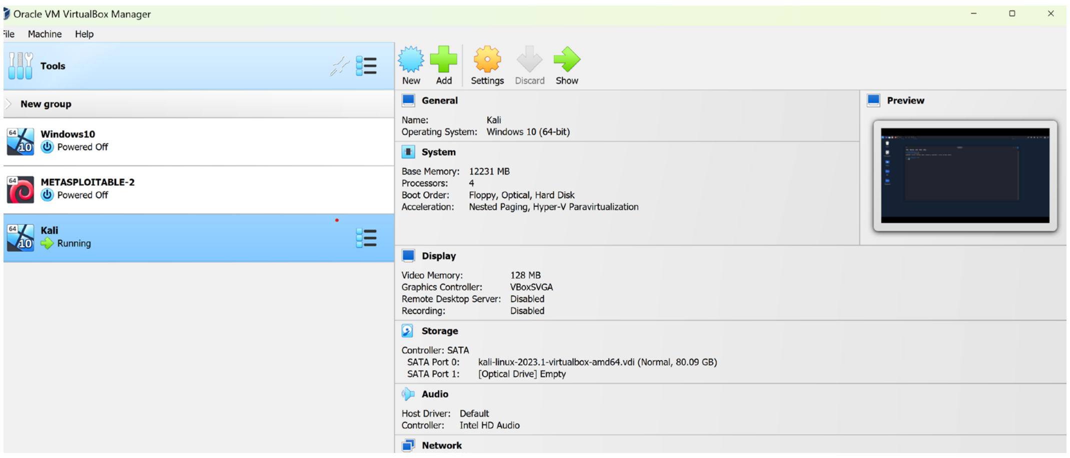
- Download OpenSSL and run basic commands:
- Encrypting a file
- Decrypting a file
- Checking a file

Introduction

In this lab, we will install OpenSSL which is a command-line binary consisting of libraries for performing perform a wide range of cryptographic operations. It can be used in scripts or for accomplishing one-time command-line tasks.

Launching OpenSSL

Open Oracle VirtualBox and then start Kali Linux. The username is **kali** and the password is also **kali.**



Windows10 Powered Off
METASPLOITABLE-2 Powered Off
Kali Running

General
Name: Kali
Operating System: Windows 10 (64-bit)

System
Base Memory: 12231 MB
Processors: 4
Boot Order: Floppy, Optical, Hard Disk
Acceleration: Nested Paging, Hyper-V Paravirtualization

Display
Video Memory: 128 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: SATA
SATA Port 0: kali-linux-2023.1-virtualbox-amd64.vdi (Normal, 80.09 GB)
SATA Port 1: [Optical Drive] Empty

Audio
Host Driver: Default
Controller: Intel HD Audio

Network

Figure 1 Opening VirtualBox and Kali Linux

Once you have started Kali Linux then launch the terminal to use OpenSSL as can be seen in the image below.

The terminal can be found on the taskbar as can be seen in the image below:

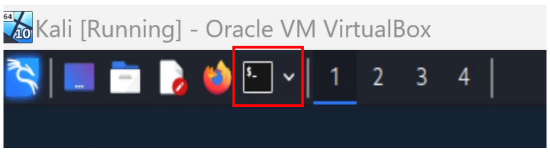
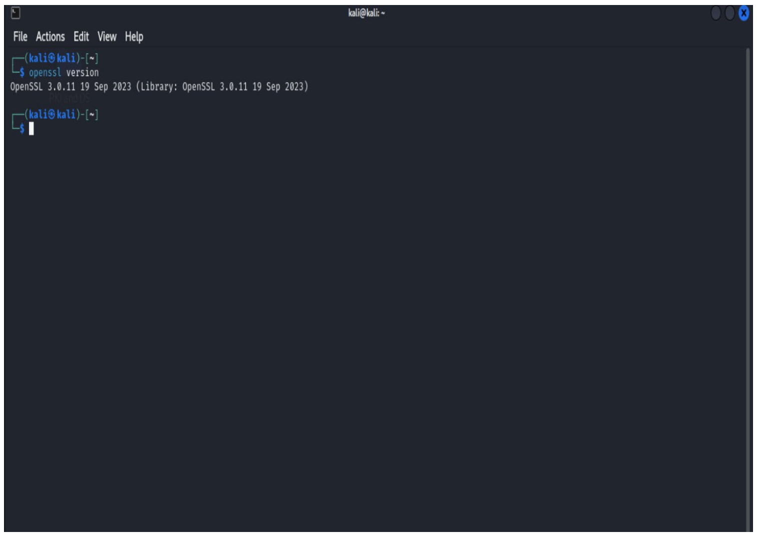


Figure 2 Launching Terminal

Click on the terminal for launching.



```
┌──(kali㉿kali)-[~]
└─$ openssl version
OpenSSL 3.0.11 19 Sep 2023 (Library: OpenSSL 3.0.11 19 Sep 2023)

┌──(kali㉿kali)-[~]
└─$ 
```

Figure 3 Terminal to use OpenSSL

1. Some basics of OpenSSL

Here are some basics of OpenSSL tool:

1.1 How to find OpenSSL version?

openssl version

1.2 List of the available commands

openssl help

1.3 List of available ciphers

openssl ciphers -v

1.4 Benchmarking

openssl speed

2. Key generation using pseudo random number generator

Pseudo Random Number Generator (PRNG) is an algorithm that utilises mathematical formulas for the sequences of random numbers. We will use OpenSSL rand function to generate random numbers that we can use as a key for encryption algorithms. Now, open an OpenSSL command prompt. Automatically, you will be placed in your default home directory. You can switch to any directory by using *cd* command. While you are in your home directory, create a file containing a 56- and 256-bits long pseudo random number. Later, you will use these numbers as your DES and AES key respectively. In an OpenSSL command prompt, enter the following commands:

Key generation for DES:

```
openssl rand -out des_keySS 56
```

Key generation for AES:

```
openssl rand -out aes_keySS 128
```

3. Data Encryption Standard (DES)

The DES is a symmetric-key encryption algorithm. The use of a very short key (i.e., 56 bits) makes DES very insecure, but DES has played highly influential role in the advancement of cryptography.

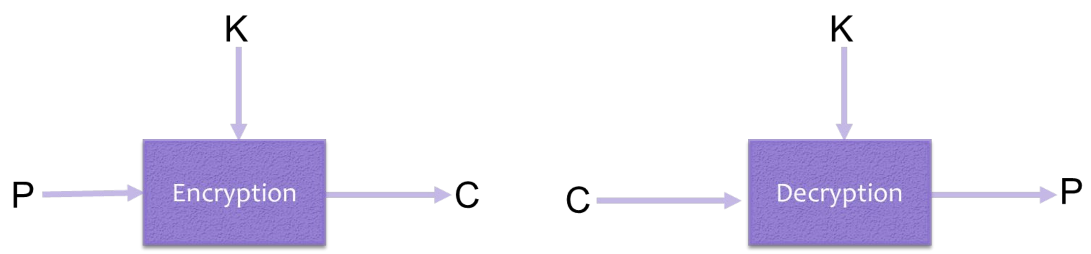


Figure 2 Encryption and decryption process

We will use DES to encrypt and decrypt a file, the encryption and decryption scenario can be seen in Figure 3. For this we will create a text file (i.e., SystemSec.txt) and write some text in it. Now, let's encrypt this file.

```
openssl des -e -kfile des_keySS -in SystemSec.txt -out SystemSec-enc.enc
```

let's analyse this command:

- *des* is the name of the cipher
- when we pass *-e* switch to the cipher, it will encrypt the file.
- *-kfile* is for specifying the key file
- *des_keySS* is the name of the key file
- *-in* is for input
- *SystemSec.txt* input file name
- *-out* is for output
- *SystemSec-enc.enc* output file name

After running this command, you will observe that there is a file created by the name of *SystemSec-enc.enc* in your current working directory. When you open that file, you will see something like this:

```
Salted___6□ûe±ª8^~uÓF"+
```

This is the encrypted output of the text in the file *SystemSec.txt*. The encrypted output might be change on your system, depending on the input text and key.

Now to provide assurance that the process works both ways, we should decrypt the *SystemSec-enc.enc* file that we just created. After we decrypt the file, go ahead and open it in your favourite editor (I am using windows Notepad).

```
openssl des -d -kfile des_keySS -in SystemSec-enc.enc -out SystemSec-dec.dec
```

let's analyse this command:

- *des* is the name of the cipher
- when we pass *-d* switch to the cipher, it will decrypt the file.
- *-kfile* is for specifying the key file
- *des_keySS* is the name of the key file
- *-in* is for input
- *SystemSec-enc.enc* is the input file name
- *-out* is for output file
- *SystemSec-dec.dec* output file name

We would expect the file *SystemSec-dec.dec* to be identical to the file that we have originally created i.e., *SystemSec.txt*. Since identical files will have identical message digests, you can prove that the files are identical by creating and comparing each file's message digest. Enter the following command lines. Then, compare the generated digests (hashes).

```
openssl md5 SystemSec.txt
```

MD5(SystemSec.txt)= *d41d8cd98f00b204e9800998ecf8427e*

```
openssl md5 SystemSec-dec.dec
```

MD5(SystemSec-dec.dec)= *d41d8cd98f00b204e9800998ecf8427e*

4. Advance Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric block cipher developed by NIST. We can use the five operation modes of AES for encryption and decryption as follows:

Encryption:

```
openssl aes-256-cbc -e -in plaintext.txt -out encrypted.enc

openssl aes-256-ecb -e -in plaintext.txt -out encrypted.enc

openssl aes-256-cfb -e -in plaintext.txt -out encrypted.enc

openssl aes-256-ofb -e -in plaintext.txt -out encrypted.enc

openssl aes-256-ctr -e -in plaintext.txt -out encrypted.enc
```

Decryption:

```
openssl aes-256-cbc -d -in encrypted.enc -out plaintext.txt

openssl aes-256-ecb -d -in encrypted.enc -out plaintext.txt

openssl aes-256-cfb -d -in encrypted.enc -out plaintext.txt

openssl aes-256-ofb -d -in encrypted.enc -out plaintext.txt
openssl aes-256-ctr -d -in encrypted.enc -out plaintext.txt
```

Now let’s repeat the above process of encryption and decryption with AES and its 128-bits key. Remember that we have created the aes_keySS in section 2 ‘Key generation using pseudo random number generator’.

```
openssl aes256 -e -kfile aes_keySS -in SystemSec.txt -out SystemSec-aes.enc
```

Now to decrypt this file we will use the following command:

```
openssl aes256 -d -kfile aes_keySS -in SystemSec-aes.enc -out SystemSec-aes.dec
```

The confirmation through the message digest is the same as we did above. We will have to pass the original file (i.e., *SystemSec.txt*) and the AES decrypted file (i.e., *SystemSec-aes.dec*)

DES and AES: Tasks

1. Alice stored an encrypted secret file (Sec.enc) on her computer. Eve some how got access to the file but does not know the encryption key. so, Eve decided to alter the contents of the file. How Alice can verify that whether the contents has been modified or not? To mimic this process, do the following tasks?

    a. Generate an AES key of size 256-bits
    b. Create a Sec.txt file with some text as can be seen below (right-click→New→Text Document)
    c. Encrypt it with AES-CTR mode (output could be Sec.enc) and then calculate the md5 message digest
    d. Now delete some contents from the Sec.enc and then calculate the md5 message digest again
    e. Compare both message digests.