

MODUL PRAKTIKUM 2022

KEAMANAN SISTEM INFORMASI

MODUL 2

Encryption Technology

ASISTEN

Ananda Anggie Nur Aini	ENJI
Andi Sayid Muhammad Qoyyum	ANDI
Arya Bimo Bagas Penggalih	BIMO
Faris Aufar Putra	PTRA
Farras Naim	RAAS
Fie Alfain Nuril Haque	ALFA
Fitria Nikmatul Hidayah	WPIN
M. Alwi Zein	ZEIN
Maulana Malik Ibrahim	IBRA
Milenia Ari Oktaviana	ARII
Muhammad Hafiz Hawarizmi	VISS
Muliya Dewi	MYDE
Ni Made Meliana Listyawati	MELI
Nurul Annisaa	NASA
Rizal Indera	INRA
Ryan Supriadi Ramadhan	RYAN
Syarah Tazkiatun Nupus	AZKI
Wiratama Putra Prakosa	RAKO



I. Tujuan Praktikum

- 1.1 Praktikan dapat memahami konsep dari *encryption technology* dan SSL
- 1.2 Praktikan dapat mengetahui perbedaan antar *symmetric key* dan *asymmetric key*
- 1.3 Praktikan dapat memahami penggunaan dari OpenSSL

II. Alat dan Bahan

- 2.1 PC/Laptop
- 2.2 Sistem Operasi Kali Linux 2020.3
- 2.3 *Virtual Box*

III. Landasan Teori

3.1 *Encryption Technology*

3.1.1 Definisi

Dalam dunia internet saat ini sudah banyak aplikasi yang membutuhkan keamanan lebih untuk mengamankan data yang dimilikinya. Dalam mengamankan sebuah informasi atau data, terdapat dua istilah, yaitu *encryption* dan *decryption*. *Encryption* adalah metode untuk mengamankan informasi atau data, sehingga menjadi pola atau sesuatu yang tidak mudah dipahami tanpa adanya kunci untuk memecahkan pola tersebut. Sedangkan *decryption* merupakan kebalikan dari *encryption*, yaitu proses memecahkan suatu pola tertentu agar informasi atau data yang telah diamankan atau dienkripsi dapat dilihat atau dibaca kembali. *Encryption technology* merupakan suatu teknologi yang digunakan untuk mengamankan suatu informasi maupun data dengan cara mengenkripsi informasi atau data tersebut.

Cryptography merupakan salah satu metode yang dapat digunakan untuk keamanan. *Cryptography* berfungsi untuk mengenkripsi suatu tulisan sehingga isi dari tulisan tersebut hanya dapat diketahui oleh orang yang memiliki kata sandinya. Informasi berupa teks yang belum dilakukan proses enkripsi disebut *plain text* sedangkan yang telah dilakukan proses enkripsi disebut *ciphertext*. Teknik untuk memecahkan sandi *cryptography* disebut dengan *cryptanalysis*. Salah satu kegunaan *cryptography* adalah untuk melindungi isi pesan dalam email, informasi kartu kredit, serta data perusahaan.

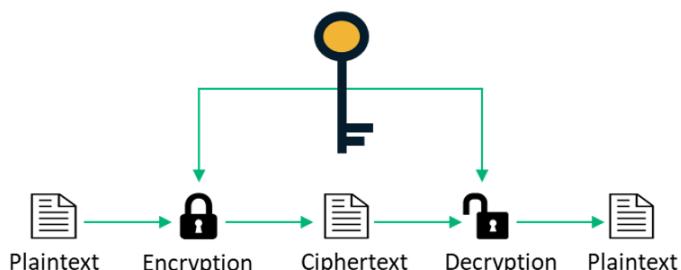
3.1.2 Tipe Algoritma *Cryptography*

Secara umum Algoritma Cryptography dibagi menjadi 3 (tiga) tipe, yaitu sebagai berikut:

- a. *Symmetric Key Encryption*

Symmetric Key Encryption merupakan algoritma yang menggunakan satu kunci tunggal untuk mengenkripsi dan mendekripsikan informasi maupun data.

Sehingga hanya penerima yang memiliki kunci asli yang dapat mendekripsikan pesan. Ilustrasi *symmetric* dapat digambarkan sebagai berikut:



Gambar 3 1 *Symmetric Encryption*

Dari gambar di atas, dapat dilihat kunci yang digunakan untuk melakukan enkripsi dan dekripsi hanya satu, dimana *plain text* (file yang belum dienkripsi) akan melewati proses enkripsi dan akan menghasilkan *ciphertext* (file yang sudah dienkripsi), kemudian akan menghasilkan *plaintext* kembali setelah didekripsi. Berikut beberapa algoritma dari *symmetric*:

1. Data Encryption Standard (DES)

DES menggunakan kunci dan blok 64 bit. Dimana 8 bit sebagai bit paritas yang memberikan kekuatan maksimum 56 bit. Pada OpenSSL DES merupakan blok paling lambat untuk mendukung cipher. Berikut merupakan table *cipher mode* pada DES

Tabel 3 1 *Cipher Mode* pada DES

<i>Cipher Mode</i>	<i>EVP call for cipher code</i>	<i>String for cipher lookup</i>
ECB	EVP_des_ecb()	des -ecb
CBC	EVP_des_cbc()	des -cbc
CFB	EVP_des_cfb()	des -cfb
OFB	EVP_des_ofb()	des -ofb

2. Advanced Encryption Standard (AES)

AES merupakan blok *cipher* yang mendukung ukuran kunci dan blok 128, 192, dan 256 bit. AES hanya tersedia pada aplikasi OpenSSL versi 0.9.7 atau yang lebih terbaru. Berikut merupakan table *cipher mode* pada AES

Tabel 3 2 *Cipher Mode* pada AES

<i>Cipher Mode</i>	<i>Key / Block Size</i>	<i>EVP call for cipher code</i>	<i>String for cipher lookup</i>
ECB	128 bits	EVP_aes_128_ecb()	aes-128-ecb
CBC	128 bits	EVP_aes_128_cbc()	aes-128-cbc
ECB	192 bits	EVP_aes_192_ecb()	aes-192-ecb
CBC	192 bits	EVP_aes_192_cbc()	aes-192-cbc
ECB	256 bits	EVP_aes_256_ecb()	aes-256-ecb
CBC	256 bits	EVP_aes_256_cbc()	aes-256-cbc

3. *International Data Encryption Algorithm (IDEA)*

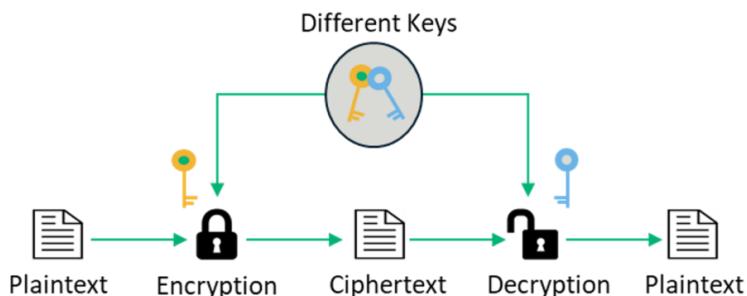
IDEA merupakan blok *cipher* dengan kunci 128 bit dan blok 64 bit IDEA sangat cepat dan dianggap sangat kuat. IDEA dilindungi oleh hak paten yang berada di Amerika Serikat dan Eropa. Berikut merupakan table *cipher mode* pada IDEA

Tabel 3 3 *Cipher Mode* pada IDEA

<i>Cipher Mode</i>	<i>EVP call for cipher code</i>	<i>String for cipher lookup</i>
ECB	EVP_idea_ecb()	idea -ecb
CBC	EVP_ idea _cbc()	idea -cbc
CFB	EVP_ idea _cfb()	idea -cfb
OFB	EVP_ idea _ofb()	idea -ofb

b. *Asymmetric Key Encryption*

Asymmetric Key Encryption merupakan kebalikan dari *symmetric*. *Asymmetric* menggunakan dua kunci yang dapat digunakan untuk mengenkripsi dan mendeskripsi informasi dan data. Dua kunci tersebut adalah kunci *private* yang hanya boleh diketahui oleh orang – orang tertentu dan bersifat rahasia, dan juga kunci *public* yang dapat diketahui atau disebarluaskan secara bebas. Ilustrasi *asymmetric* dapat digambarkan sebagai berikut:



Gambar 3 2 Asymmetric Encryption

Dari gambar di atas, dapat dilihat dalam melakukan enkripsi dan dekripsi menggunakan dua jenis kunci yang berbeda, dimana untuk mengenkripsi dapat menggunakan *public key* sedangkan untuk mendekripsi pesan harus menggunakan *private key* atau *secret key* yang dimiliki oleh yang membuat enkripsi. Berikut merupakan beberapa algoritma dari *asymmetric*:

1. Diffie-Helman (DH)

DH digunakan untuk *key agreement*. Dalam istilah sederhana, *key agreement* adalah pertukaran informasi melalui media tidak aman yang memungkinkan masing-masing dari dua pihak dalam percakapan untuk menghitung nilai yang biasanya digunakan sebagai kunci untuk *cipher* simetris. DH memungkinkan kedua pihak untuk membangun kunci rahasia bersama yang digunakan oleh enkripsi dan *hash* algoritma.

2. Rivest, Shamir, dan Adleman (RSA)

RSA adalah algoritma *public key* yang paling sering digunakan karena menyediakan kerahasiaan, otentikasi, dan enkripsi semua dalam satu paket kecil yang rapi. RSA algoritma mudah untuk di implementasikan dan dimengerti. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran tersebut dilakukan untuk memperoleh *private key*.

3. Digital Signature Algorithm (DSA)

DSA digunakan untuk membuat dan memverifikasi tanda tangan digital. Menyediakan otentikasi dan tidak dapat melakukan enkripsi. DSA memanfaatkan teknologi *public key*. Sepasang *public key* dan *private key* dibuat unik dan tidak ada pasangannya. *Private key* disimpan oleh pemiliknya, dan dipergunakan untuk membuat tanda tangan digital. Sedangkan *public key* dapat diserahkan kepada siapa saja yang ingin memeriksa tanda tangan digital

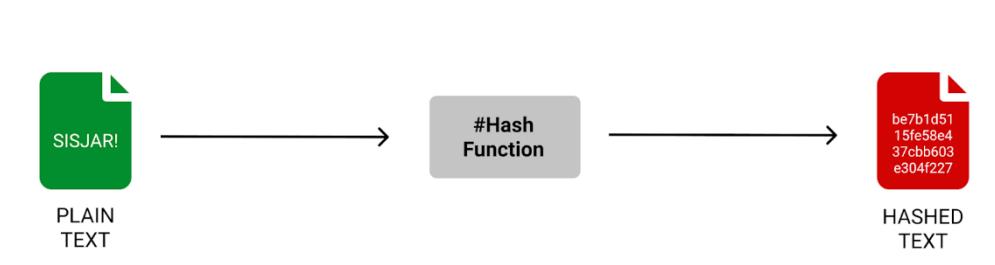
yang bersangkutan pada suatu dokumen. Proses pembuatan dan pemeriksaan tanda tangan ini melibatkan sejumlah teknik *cryptography* seperti *one-way hashing* dan enkripsi asimetris DH.

4. *Elliptic Curve Cryptosystem (ECC)*

ECC adalah *cryptography public key*. Pada *cryptography public key*, masing-masing *user* atau *device* mengambil bagian dalam komunikasi yang memiliki pasangan kunci yaitu *public key* dan *private key*. Hanya pengguna yang cocok yang dapat menggunakan *private key* yang sesuai, tetapi *public key* yang digunakan disebarluaskan kepada pihak yang akan mengirimkan data kepada pemilik *private key*.

c. Hash Function

Algoritma *Hash* atau yang biasa disebut dengan *hash function* merupakan algoritma *cryptography* yang tidak memerlukan kunci. Algoritma ini bersifat satu arah dimana jika kita masukkan data maka akan menghasilkan sebuah *checksum* atau *fingerprint* dari data tersebut. Berikut merupakan ilustrasi dari *hash function*:



Gambar 3 3 *Hash Function*

Algoritma hash biasa digunakan untuk melakukan verifikasi terhadap data untuk memastikan data yang diterima tetap utuh tidak diubah atau rusak pada saat proses pengiriman. Hash function yang biasa digunakan adalah MD4, MD5, SHA-1 dan SHA-2.

1. *Message-Digest Algorithm 5 (MD5)*

MD5 merupakan penyempurnaan dari MD4, dimana algoritma ini dibuat oleh Ron Rivest pada tahun 1991. Pengertian MD5 adalah fungsi *hash cryptography* yang digunakan secara luas dengan *hash value* 128-bit dan memiliki panjang 16 *bytes* (32 karakter). MD5 telah digunakan untuk membangun keamanan aplikasi, khususnya dalam menyamarkan *password* yang tersimpan pada *database*.

2. Secure Hash Algorithm (SHA)

Merupakan fungsi hash dengan satu arah yang dibuat oleh NSA (*National Security Agency*) yang diterbitkan oleh NIST (*National Institute of Standards and Technology*) dan digunakan bersama dengan DSS (*Digital Signature Standard*). SHA dibuat berdasarkan MD4 yang dibuat sedemikian rupa sehingga secara komputasi tidak mungkin menemukan pesan yang diberikan. Jenis SHA antara lain adalah SHA-1, SHA-2, SHA-256, SHA-512. Perbedaan diantara jenis-jenis SHA tersebut adalah panjang karakter.

3.1.3 Perbedaan *Symmetric Key* dan *Asymmetric Key*

Berikut adalah beberapa perbedaan antara *symmetric key* dan *asymmetric key* diantaranya:

- a. *Symmetric* menggunakan kunci tunggal yaitu *public key*. Sedangkan *asymmetric* menggunakan sepasang kunci yaitu *public key* dan *private key*.
- b. *Symmetric* merupakan teknik lama dalam enkripsi. *Asymmetric* merupakan teknik baru dalam enkripsi.
- c. *Asymmetric* diperkenalkan untuk melengkapi masalah *inherent* pada *symmetric*, menghilangkan kebutuhan untuk berbagi kunci dengan menggunakan *public key* dan *private key* untuk enkripsi dan dekripsi pesan ketika berkomunikasi.
- d. *Asymmetric* membutuhkan waktu yang cukup lama dalam melakukan enkripsi dari pada *symmetric*.

3.2 Apache Web Server

Apache adalah layanan *web server* yang paling popular. Berdasarkan survei W3Tech Top ranks *web server* di bulan januari 2020. Apache adalah *web server* yang paling banyak digunakan. Bahkan dari semua *web server* yang ada, Penggunaan apache mencapai 41.5%. Hal tersebut dikarenakan tentu saja karena software ini *Open Source* dan juga sangat fleksibel, Apache sendiri bisa digunakan untuk berbagai sistem operasi. Selain fleksibel, konfigurasi dan pengaturan apache sangat mudah dan simpel. Tidak ada pengaturan lebih lanjut untuk menggunakan *web server* ini. Hal tersebut menjadi alasan banyak orang menggunakan apache sebagai *web server*.

3.3 Secure Socket Layer (SSL)

3.3.1 Definisi

SSL adalah protokol yang digunakan untuk mengamankan suatu *website* yang dapat berguna untuk melindungi data maupun informasi yang terdapat di dalamnya. SSL adalah standar industri untuk komunikasi *web* yang aman dan digunakan untuk melindungi jutaan transaksi *online* setiap hari. Jika komputer lain berada diantara *client* dan *server*, komputer tersebut dapat melihat nomor kartu kredit *client*, *username* dan *password*, dan informasi sensitif lainnya jika hal ini tidak dienkripsi dengan sertifikat SSL.

Ketika sertifikat SSL digunakan, informasi menjadi tidak dapat terbaca oleh siapapun kecuali ke *server* yang dituju saat mengirim informasi tersebut. Hal ini melindungi informasi tersebut dari pihak yang tidak memiliki wewenang atas informasi tersebut. Perkembangan internet dan aplikasi berbasis *online* yang terus berkembang, pengamanan data dari sebuah *website* sangatlah penting.

3.3.2 OpenSSL

OpenSSL merupakan sebuah *toolkit cryptography* yang dijalankan pada protokol jaringan *Secure Socket Layer* (SSL) dan *Transport Layer Security* (TLS). Tujuan diciptakannya OpenSSL adalah untuk keamanan jaringan antara *client* dan *server*, sehingga jalur komunikasi data dari komputer *client* ke suatu *server* yang dituju tersebut aman. OpenSSL mulai diterapkan pada akhir tahun 1998 yang dikembangkan secara *open source*. OpenSSL juga merupakan sebuah proyek kolaborasi dari para programmer di seluruh dunia. Oleh karena itu, OpenSSL menjadi *toolkit* keamanan situs yang banyak digunakan, termasuk situs-situs besar yang ada saat ini seperti Facebook, Google, Amazon, maupun Yahoo. OpenSSL bisa digunakan di sistem operasi berbasis Linux, Windows, dan Mac.

3.3.3 Fungsi OpenSSL

Aplikasi OpenSSL ini merupakan *command line tool* yang menggunakan berbagai fungsi kriptografi OpenSSL's *crypto library* dari *shell* ini dapat digunakan untuk:

1. Membuat RSA, DH dan DSA parameter kunci
2. Membuat sertifikat SSL

3. Menghitung pesan *digests*
4. Enkripsi dan dekripsi
5. Pengujian SSL / TLS *client* dan *server*
6. Penanganan S/MIME *signed or encrypted mail.*

3.3.4 Mode Dasar Block Cipher

Terdapat empat mode dasar yang diimplementasikan oleh OpenSSL, yaitu sebagai berikut:

a. *Electronic Code Book* (ECB)

Pada mode ini, satu *blok plaintext* akan menghasilkan satu *blok ciphertext*. Mode ini merupakan mode yang kurang aman karena sangat rentan terhadap *dictionary attack*, sehingga sangat tidak disarankan untuk menggunakan mode ini dalam kondisi apapun.

b. *Cipher Block Chaining* (CBC)

Mode ini merupakan solusi dari mode ECB dengan adanya *XORing* dimana satu blok *ciphertext* akan diambil dari satu blok *plaintext* berikutnya. Sebagai solusi dari *dictionary attack*, maka pada mode ini memungkinkan kita untuk mengatur *Initialization Vector* (IV). Dengan adanya IV kita dapat mengacak blok *ciphertext* dari *plaintext* yang ada.

c. *Cipher Feedback* (CFB)

Mode ini merupakan salah satu cara untuk mengubah blok *cipher* menjadi *stream cipher*, namun keseluruhan blok *plaintext* harus diterima sebelum enkripsi dilakukan. Mode ini juga menggunakan IV seperti mode CBC. Dalam pengimplementasiannya, menggunakan kunci yang sama untuk setiap *plaintext*-nya sangat tidak disarankan.

d. *Output Feedback* (OFB)

Sama seperti CFB, mode ini juga dapat mengubah blok *cipher* menjadi *stream cipher* namun, mode ini menggunakan *stream cipher* secara tradisional sehingga lebih rentan terhadap serangan *bit-flipping* yang berpengaruh pada *stream cipher*. Mode ini biasa digunakan untuk keperluan yang bersifat *offline* sehingga kita dapat membuat *ciphertext* terlebih dahulu sebelum adanya *plaintext*. OFB juga mengimplementasikan IV sama seperti CFB dan CBC.

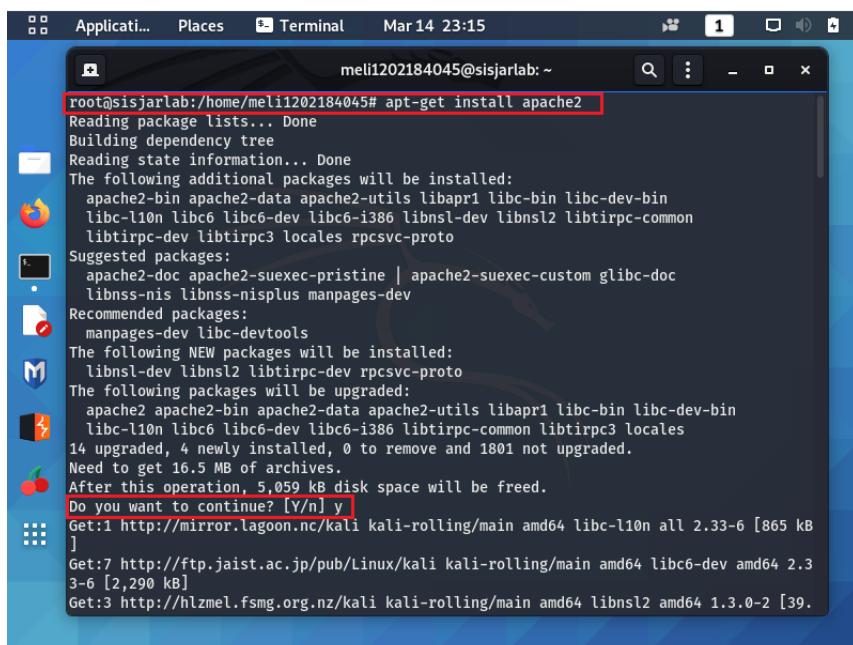
IV. Lab Praktik

NOTE: UNTUK PENULISAN COMMAND DIANJURKAN UNTUK DI KETIK MANUAL, TIDAK DI COPY PASTE!

4.1 Installasi Apache2 Web Server

1. Gunakan perintah berikut untuk installasi apache2 *Web Server* lalu ketik y.

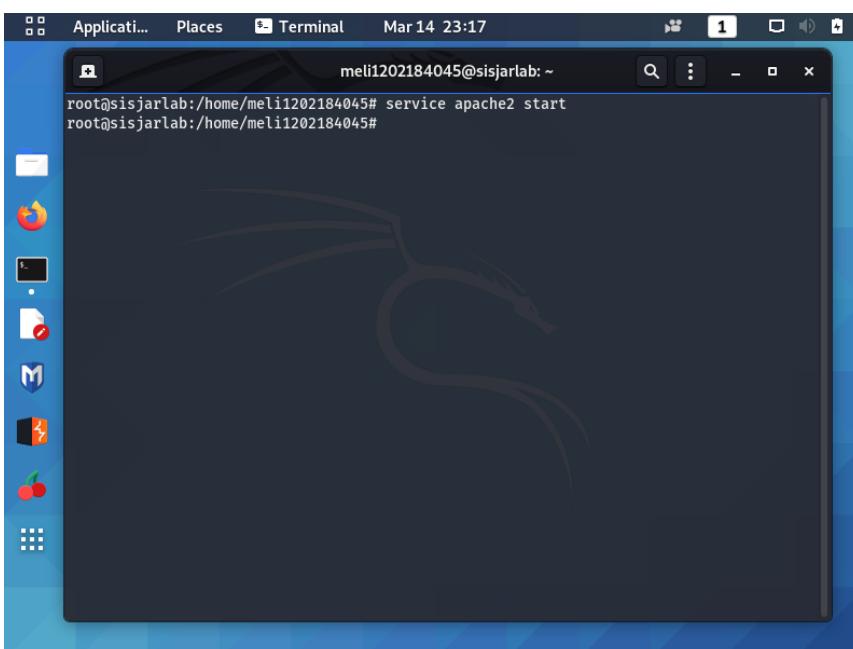
```
root@sisjarlab:/home/meli1202184045# apt-get install apache2
```



```
root@sisjarlab:/home/meli1202184045# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
apache2-bin apache2-data apache2-utils libapr1 libc-bin libc-dev-bin
libc-l10n libc6 libc6-dev libc6-i386 libnsl-dev libnsl2 libtirpc-common
libtirpc-dev libtirpc3 locales rpcsvc-proto
Suggested packages:
apache2-doc apache2-suexec-pristine | apache2-suexec-custom glibc-doc
libnss-nis libnss-nisplus manpages-dev
Recommended packages:
manpages-dev libc-devtools
The following NEW packages will be installed:
libnsl-dev libnsl2 libtirpc-dev rpcsvc-proto
The following packages will be upgraded:
apache2 apache2-bin apache2-data apache2-utils libapr1 libc-bin libc-dev-bin
libc-l10n libc6 libc6-dev libc6-i386 libtirpc-common libtirpc3 locales
14 upgraded, 4 newly installed, 0 to remove and 1801 not upgraded.
Need to get 16.5 MB of archives.
After this operation, 5,059 kB disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 http://mirror.lagoon.nc/kali kali-rolling/main amd64 libc-l10n all 2.33-6 [865 kB]
Get:7 http://ftp.jaist.ac.jp/pub/Linux/kali kali-rolling/main amd64 libc6-dev amd64 2.3
3-6 [2,290 kB]
Get:3 http://hlzmel.fsmg.org.nz/kali kali-rolling/main amd64 libnsl2 amd64 1.3.0-2 [39.
```

2. Lakukan *restart* apache2 yang berhasil di install dengan perintah berikut.

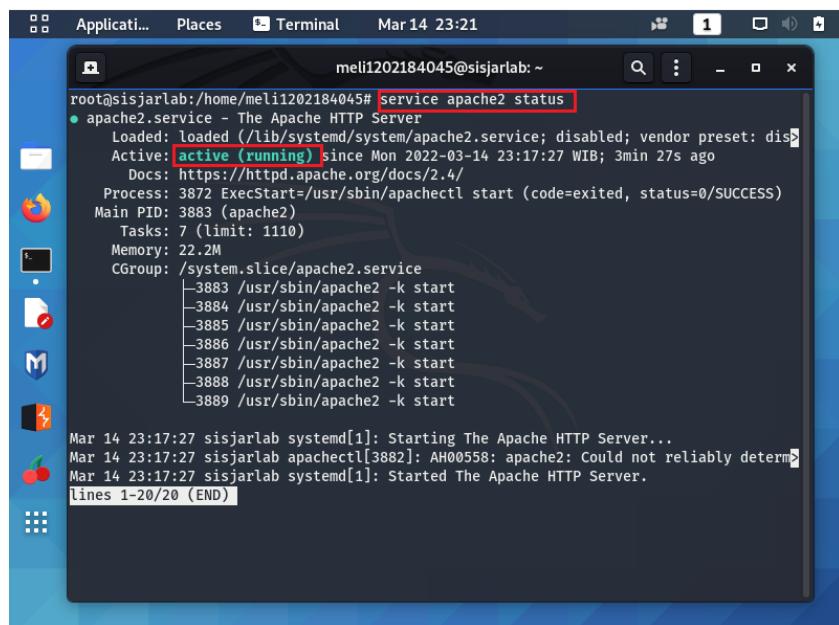
```
root@sisjarlab:/home/meli1202184045# service apache2 start
```



```
root@sisjarlab:/home/meli1202184045# service apache2 start
root@sisjarlab:/home/meli1202184045#
```

- Pastikan status apache2 *active* atau *running*.

```
root@sisjarlab:/home/meli1202184045# service apache2 status
```



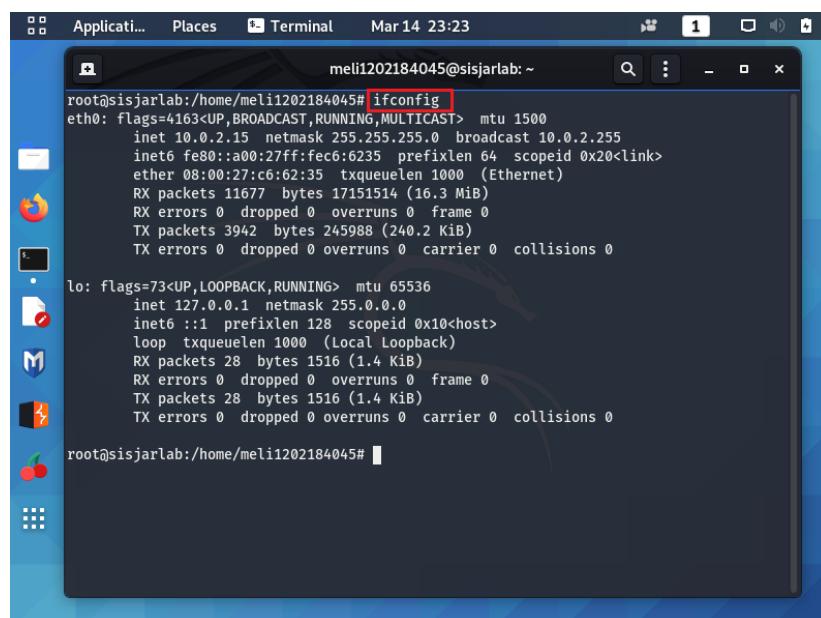
```
root@sisjarlab:/home/meli1202184045# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2022-03-14 23:17:27 WIB; 3min 27s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 3872 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 3883 (apache2)
   Tasks: 7 (limit: 1110)
  Memory: 22.2M
    CGroup: /system.slice/apache2.service
            ├─3883 /usr/sbin/apache2 -k start
            ├─3884 /usr/sbin/apache2 -k start
            ├─3885 /usr/sbin/apache2 -k start
            ├─3886 /usr/sbin/apache2 -k start
            ├─3887 /usr/sbin/apache2 -k start
            ├─3888 /usr/sbin/apache2 -k start
            └─3889 /usr/sbin/apache2 -k start

Mar 14 23:17:27 sisjarlab systemd[1]: Starting The Apache HTTP Server...
Mar 14 23:17:27 sisjarlab apachectl[3882]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1 for Port 80
Mar 14 23:17:27 sisjarlab systemd[1]: Started The Apache HTTP Server.
lines 1-20/20 (END)
```

- Cek IP *address* perangkat kali linux dengan perintah berikut

```
root@sisjarlab:/home/meli1202184045# ifconfig
```

Note: setiap orang kemungkinan memiliki ip *address* yang berbeda

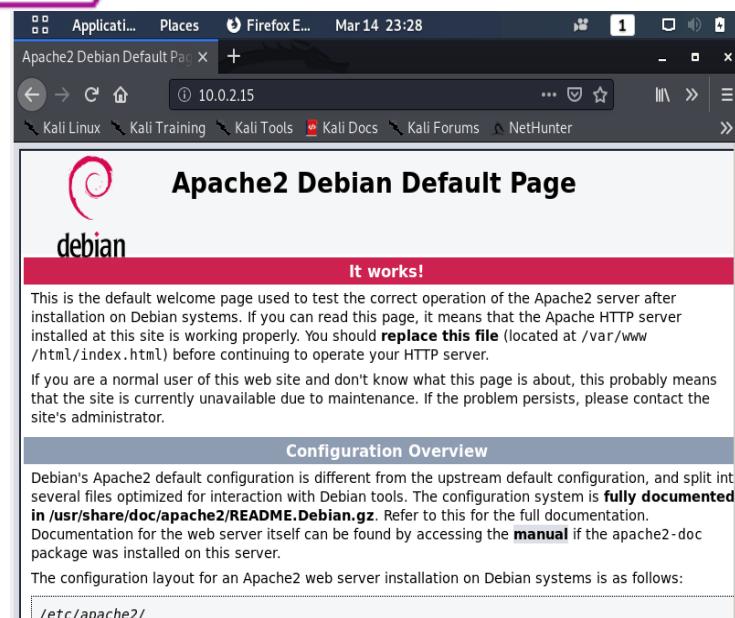


```
root@sisjarlab:/home/meli1202184045# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe00:6235  prefixlen 64  scopeid 0x20<link>
          ether 08:00:27:c6:62:35  txqueuelen 1000  (Ethernet)
            RX packets 11677  bytes 17151514 (16.3 MiB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 3942  bytes 245988 (240.2 Kib)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
          loop  txqueuelen 1000  (Local Loopback)
            RX packets 28  bytes 1516 (1.4 KiB)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 28  bytes 1516 (1.4 KiB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@sisjarlab:/home/meli1202184045#
```

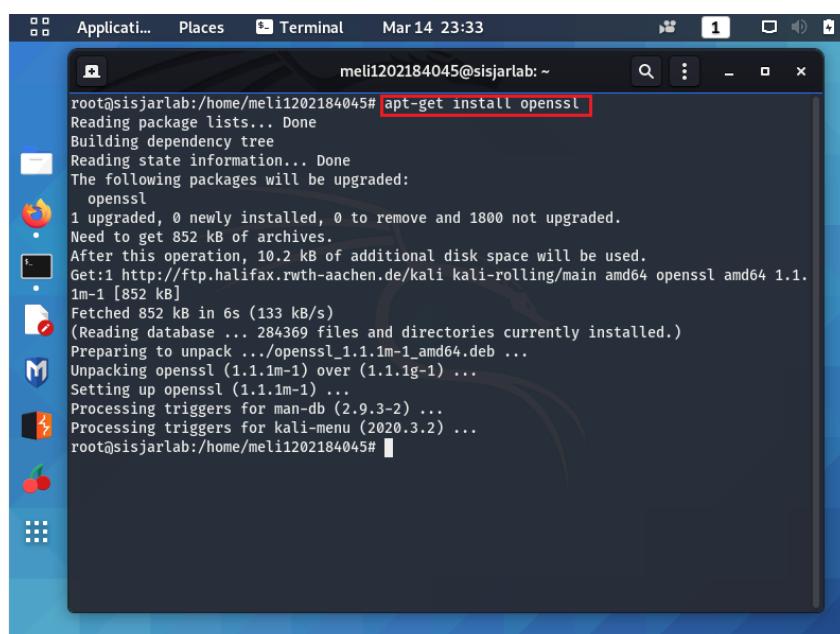
- Kemudian buka *web browser* Mozilla Firefox untuk memeriksa apakah *web server* apache2 sudah berhasil dijalankan dengan mengetikkan IP *address* yang didapat dari perintah sebelumnya. Lalu akan terlihat halaman *web* seperti dibawah ini



4.2 Installasi OpenSSL

1. Gunakan perintah berikut untuk installasi OpenSSL.

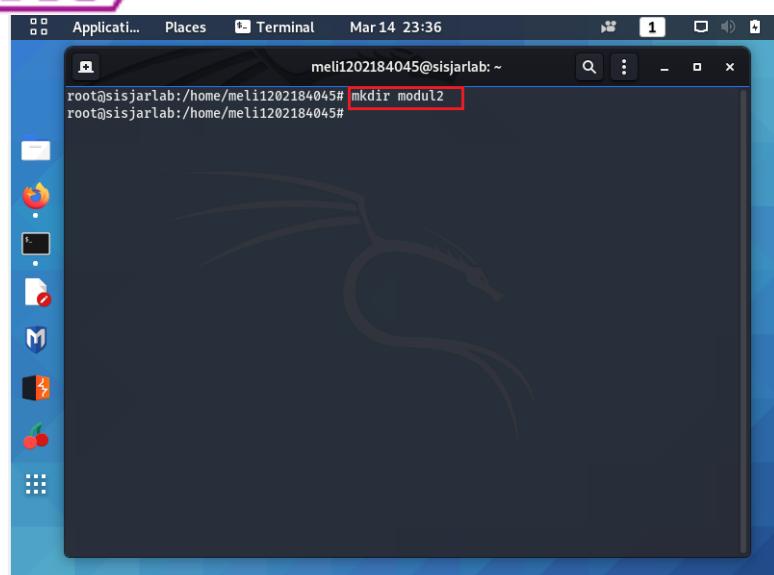
```
root@sisjarlab:/home/meli1202184045# apt-get install openssl
```



4.3 Enkripsi dan Dekripsi Menggunakan AES

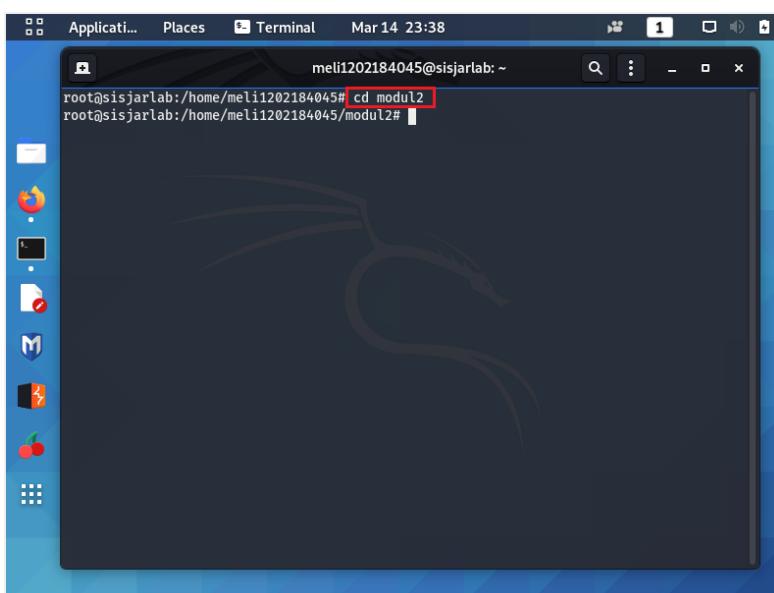
1. Buatlah folder baru bernama modul4 dengan *command* berikut. Folder ini akan digunakan untuk menyimpan file *encrypt* dan *decrypt*

```
root@sisjarlab:/home/meli1202184045# mkdir modul2
```



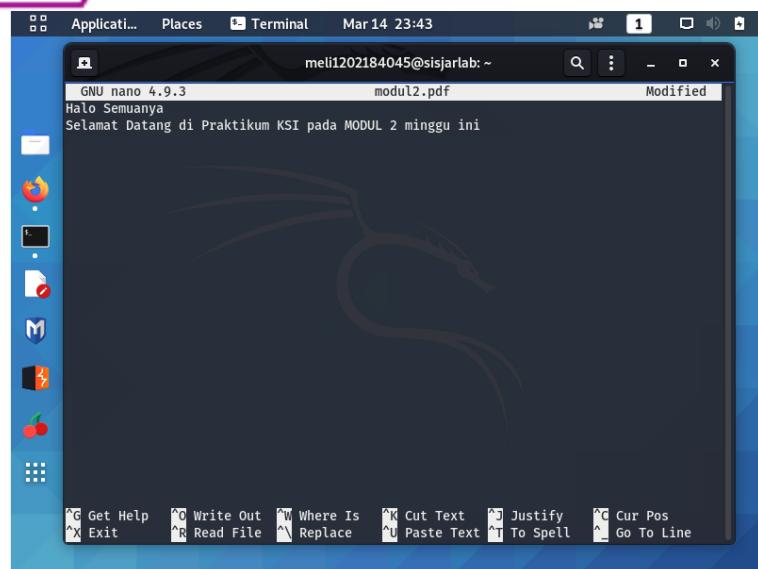
2. Masuk ke dalam direktori modul4 dengan perintah berikut :

```
root@sisjarlab:/home/meli1202184045# cd modul2
```



3. Buatlah *file* dengan nama modul2.pdf dengan *command* berikut dengan ketentuan seperti dibawah ini. *File* ini akan dienkripsi menggunakan AES

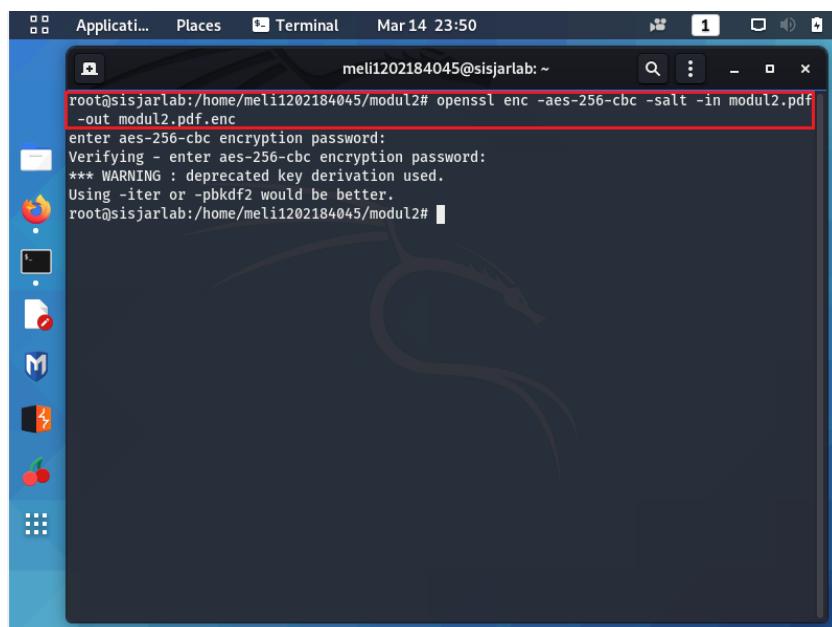
```
root@sisjarlab:/home/meli1202184045/modul2# nano modul2.pdf
```



4. Selanjutnya masukkan *command* berikut untuk mengenkripsi *file* yang sebelumnya telah dibuat. Kemudian masukkan *password* yang akan digunakan untuk mengenkripsi file tersebut (*Password* yang digunakan: **modul2**).

Note: Pastikan semua command diketikkan dalam satu baris perintah

```
root@sisjarlab:/home/meli1202184045/modul2# openssl enc -aes-256-cbc -salt -in modul2.pdf -out modul2.pdf.enc
```



Kemudian tampilkan hasil dari *file* yang telah dienkripsi dengan menggunakan *command* berikut

```
root@sisjarlab:/home/meli1202184045/modul2# ls -al
```

```
meli1202184045@sisjarlab:~/home/meli1202184045/modul2# ls -all
total 16
drwxr-xr-x  2 root      root      4096 Mar 14 23:50 .
drwxr-xr-x 16 meli1202184045 meli1202184045 4096 Mar 14 23:36 ..
-rw-r--r--  1 root      root      71 Mar 14 23:43 modul2.pdf
-rw-r----  1 root      root     96 Mar 14 23:50 modul2.pdf.enc
root@meli1202184045@sisjarlab:~/home/meli1202184045/modul2#
```

5. Berikut tampilan dari *file* modul2.pdf.enc jika dibuka dengan menggunakan *command* berikut:

```
root@meli1202184045@sisjarlab:~/home/meli1202184045/modul2# xxd modul2.pdf.enc
```

```
meli1202184045@sisjarlab:~/home/meli1202184045/modul2# xxd modul2.pdf.enc
00000000: 5361 6c74 6564 5f5f a2f8 2acc 3b86 c044 Salted__.*.;..D
00000010: cc04 3684 d8b0 f564 b453 zee8 f4a8 108f ..6....d.S.....
00000020: 9cf0 d835 2935 49f1 b58c 0b5c 59a2 7fd6 ..5I....\V...
00000030: 7244 61c8 4723 3adb 7e67 a4f9 d003 0090 rba.G#:.-g.....
00000040: 636d 44a5 8b88 ae12 d8ce a1cf 77f5 a376 cmD.....W..v
00000050: 51f9 8fe2 e58a cf58 506a 7553 d11d 6064 Q.....XPjuS..`d
root@meli1202184045@sisjarlab:~/home/meli1202184045/modul2#
```

6. Selanjutnya, untuk mendekripsi isi *file* modul2.pdf.enc, gunakan *command* berikut dan masukkan *password* yang telah diinputkan pada proses enkripsi (*Password* yang digunakan: **modul2**). Maka hasil dari dekripsi *file* akan sama dengan isi dari *file* enkripsi.

```
root@meli1202184045@sisjarlab:~/home/meli1202184045/modul2# openssl enc -aes-256-cbc -salt -d -in modul2.pdf.enc
```

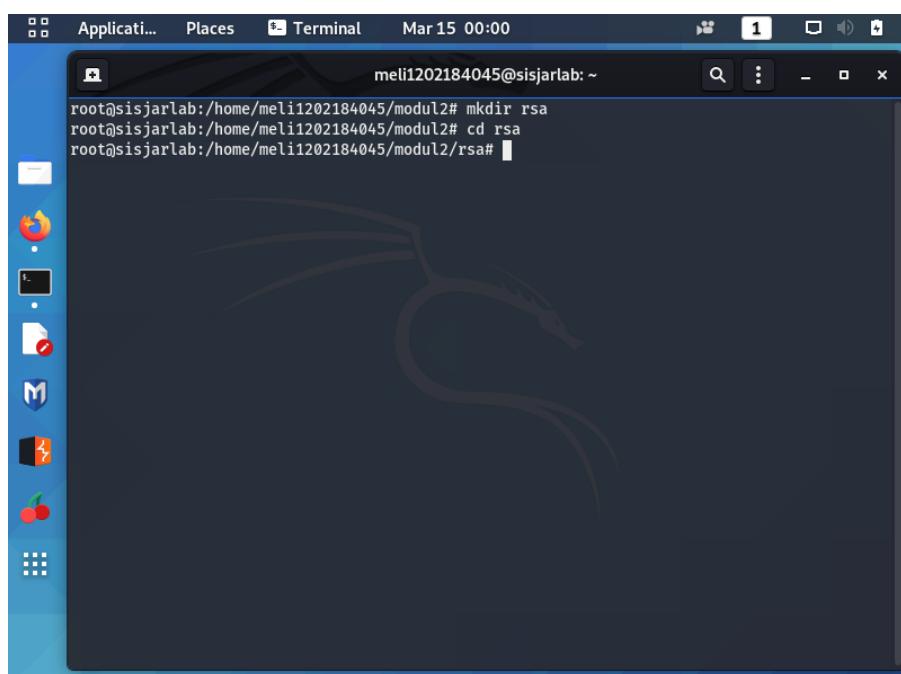
```
root@sisjarlab:/home/sisjar/modul2# openssl enc -aes-256-cbc -salt -d -in modul2.pdf.enc
enter aes-256-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
SELAMAT DATANG DI PRAKTIKUM KSI MODUL 2
root@sisjarlab:/home/sisjar/modul2#
```

4.4 Konfigurasi RSA

4.4.1 Membuat file RSA

- Buatlah folder baru bernama rsa dengan *command* berikut. Folder ini akan digunakan untuk menyimpan file hasil *encrypt* dan *decrypt* dari RSA.

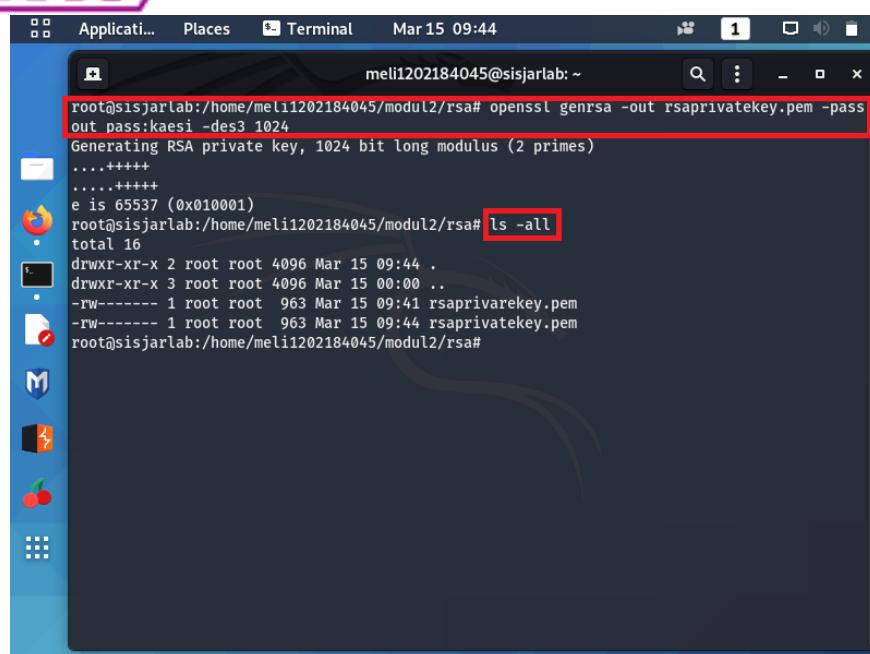
```
root@sisjarlab:/home/meli1202184045/modul2# mkdir rsa
root@sisjarlab:/home/sisjar/modul2# cd rsa
```



- Selanjutnya kita akan men-*generate* 1024-bit RSA *private key* dengan *command* seperti di bawah. Enkripsi menggunakan 3DES dengan *password* kaesi. Selanjutnya *generate* yang dilakukan akan menghasilkan file rsaprivatekey.pem

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl genrsa -out
rsaprivatekey.pem -passout pass:kaesi -des3 1024
```

```
root@sisjarlab:/home/meli1202184045/modul2/rsa#ls
```

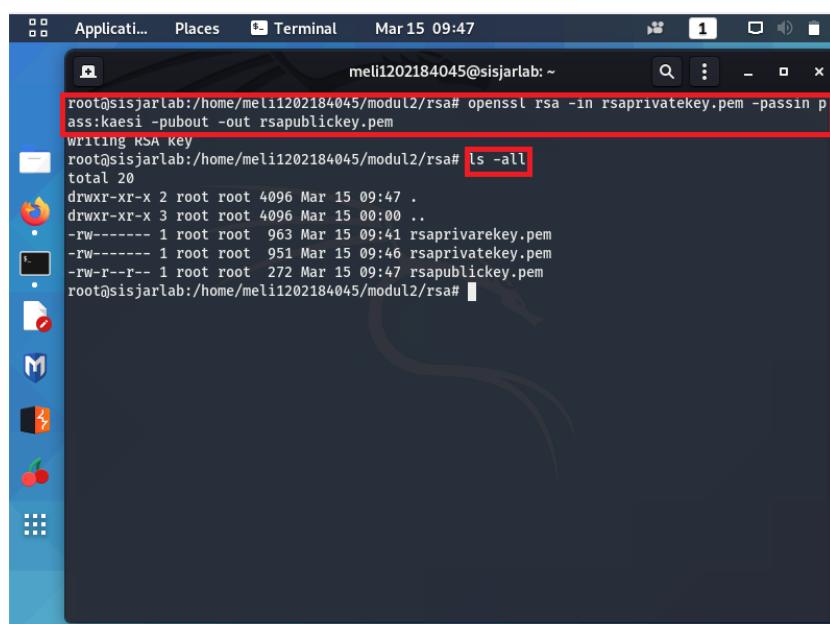


```
meli1202184045@sisjarlab: ~
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl genrsa -out rsaprivatekey.pem -passout pass:kaesi -des3 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls -all
total 16
drwxr-xr-x 2 root root 4096 Mar 15 09:44 .
drwxr-xr-x 3 root root 4096 Mar 15 00:00 ..
-rw----- 1 root root 963 Mar 15 09:41 rsaprivatekey.pem
-rw----- 1 root root 963 Mar 15 09:44 rsaprivatekey.pem
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

3. Lalu kita akan membaca *private key* RSA dari *file* rsaprivatekey.pem dengan menggunakan *password* kaesi dan menulis *public key* yang sesuai ke dalam *file* rsapublickey.pem dengan menggunakan *command* berikut

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsa -in rsaprivatekey.pem -passin pass:kaesi -pubout -out rsapublickey.pem
```

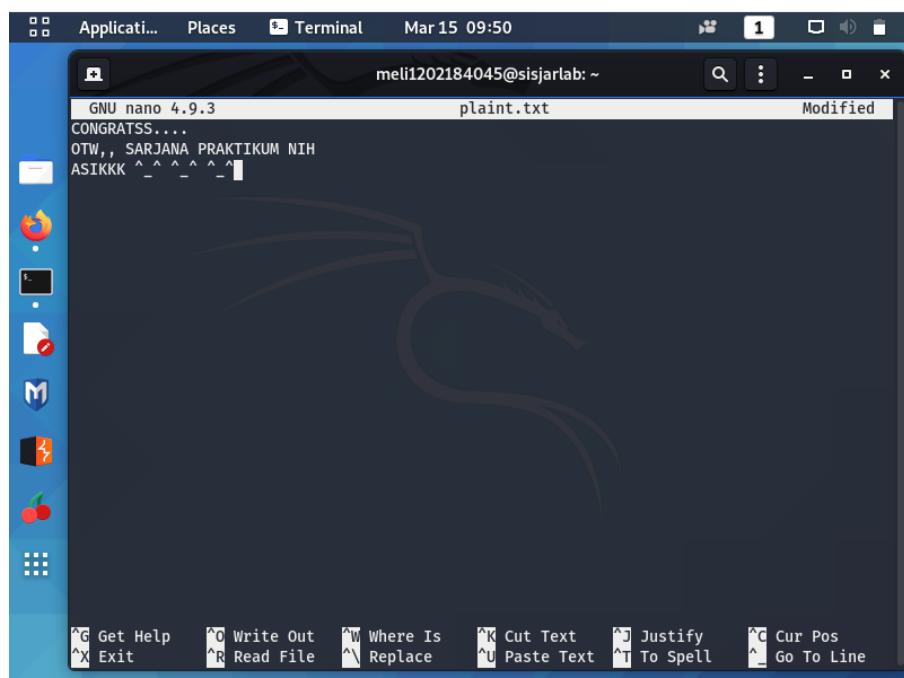
```
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls
```



```
meli1202184045@sisjarlab: ~
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsa -in rsaprivatekey.pem -passin pass:kaesi -pubout -out rsapublickey.pem
writing RSA key
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls -all
total 20
drwxr-xr-x 2 root root 4096 Mar 15 09:47 .
drwxr-xr-x 3 root root 4096 Mar 15 00:00 ..
-rw----- 1 root root 963 Mar 15 09:41 rsaprivatekey.pem
-rw----- 1 root root 951 Mar 15 09:46 rsaprivatekey.pem
-rw-r--r-- 1 root root 272 Mar 15 09:47 rsapublickey.pem
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

4. Kemudian buatlah file plain.txt yang akan digunakan untuk objek *encrypt* dan *decrypt* dan isi sesuai dengan gambar di bawah ini dengan *command* berikut.

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# nano plaint.txt
```

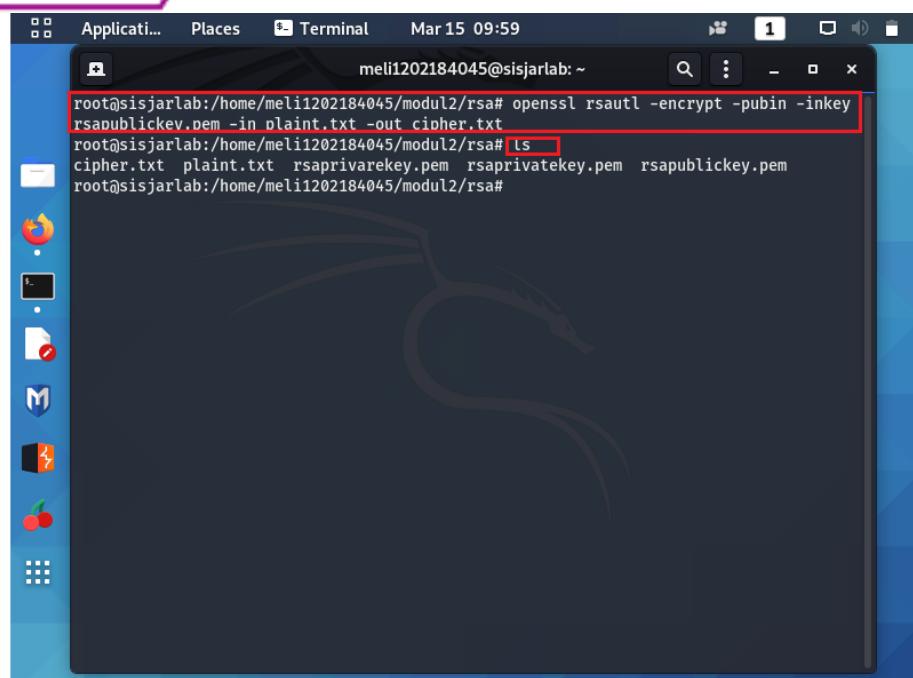


4.4.2 Enkripsi dan Dekripsi Menggunakan RSA

1. Pertama kita akan mengenkripsi file `plaint.txt` dengan menggunakan *public key* RSA dari file `rsapublickey.pem`, dimana isi file `plaint.txt` akan ditulis ke dalam file `cipher.txt` dengan menggunakan *command* berikut ini.

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -encrypt -pubin -inkey rsapublickey.pem -in plain.txt -out cipher.txt
```

```
root@sisjarlab:/home/meli1202184045/modul12/rsa# ls
```



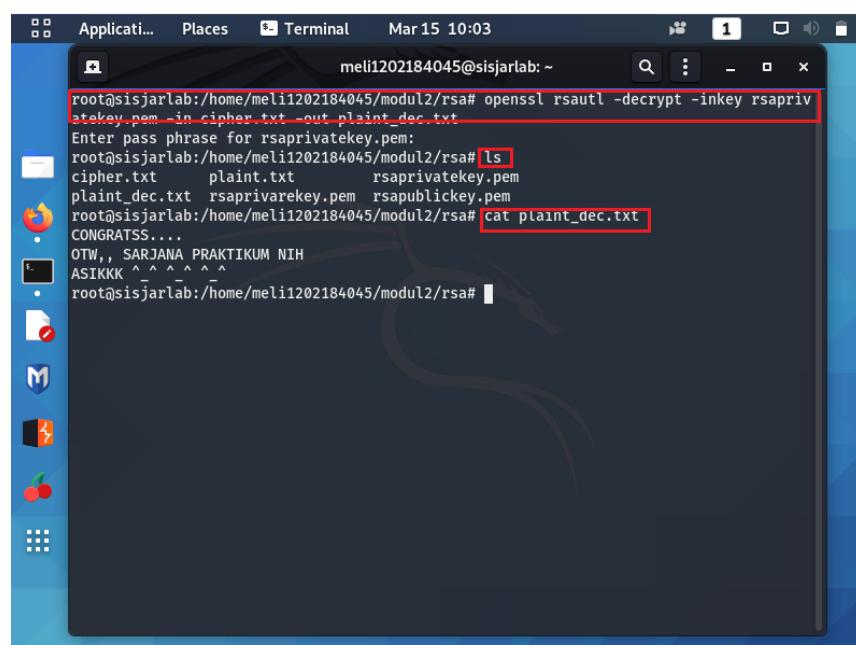
```
meli1202184045@sisjarlab:~  
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsa -encrypt -pubin -inkey  
rsapublickey.pem -in plaint.txt -out cipher.txt  
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls  
cipher.txt plaint.txt rsaprivatekey.pem rsaprivatekey.pem rsapublickey.pem  
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

2. Kemudian dengan menggunakan *private key* RSA dari file rsaprivatekey.pem, isi dari file cipher.txt akan didekripsi dan ditulis ke dalam file plain_dec.txt dengan command berikut, dan isikan *password* dari rsaprivatekey.pem yaitu **kaesi**.

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsa -decrypt -inkey  
rsaprivatekey.pem -in cipher.txt -out plain_dec.txt
```

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls
```

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# cat plain_dec.txt
```



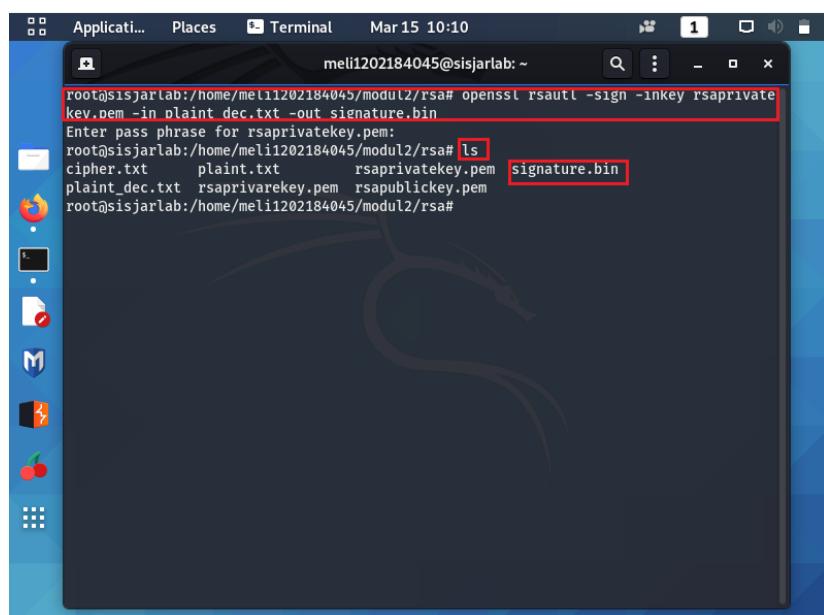
```
meli1202184045@sisjarlab:~  
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsa -decrypt -inkey rsapriv  
atekey.pem -in cipher.txt -out plain_dec.txt  
Enter pass phrase for rsaprivatekey.pem:  
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls  
cipher.txt plaint.txt rsaprivatekey.pem  
plain_dec.txt rsaprivatekey.pem rsapublickey.pem  
root@sisjarlab:/home/meli1202184045/modul2/rsa# cat plain_dec.txt  
CONGRATSS....  
OTW,, SARJANA PRAKTIKUM NIH  
ASIKKK ^ ^ ^ ^ ^  
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

4.4.3 Membuat Signature Menggunakan RSA

- Pertama *file plain_dec.txt* akan diberi *signature digital* dengan menggunakan *private key RSA* dari *file rsaprivatekey.pem* yang kemudian *signature* tersebut dituliskan ke dalam *file signature.bin*. Kemudian isi *password* dari *private key* yang telah dibuat sebelumnya dengan menggunakan *command* berikut. (*Password* yang digunakan: **kaesi**).

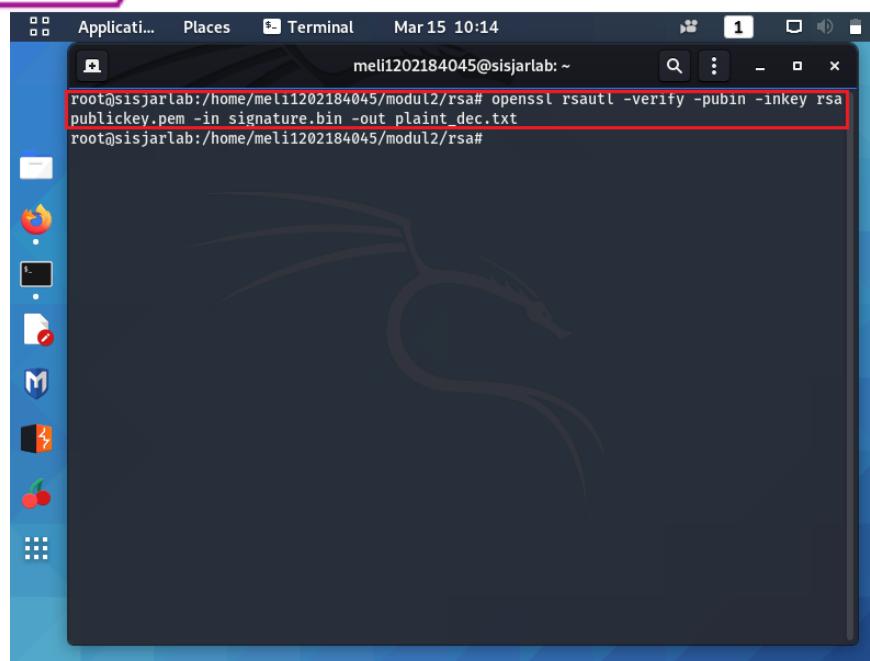
```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -sign -inkey rsaprivatekey.pem -in plain_dec.txt -out signature.bin
```

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# ls
```



- Selanjutnya dengan menggunakan *public key RSA* dari *file rsapublickey.pem*, *signature* pada *file signature.bin* diverifikasi dan data asli yang tidak di-*signature* dituliskan ke *file plain_dec.txt* dengan *command* berikut.

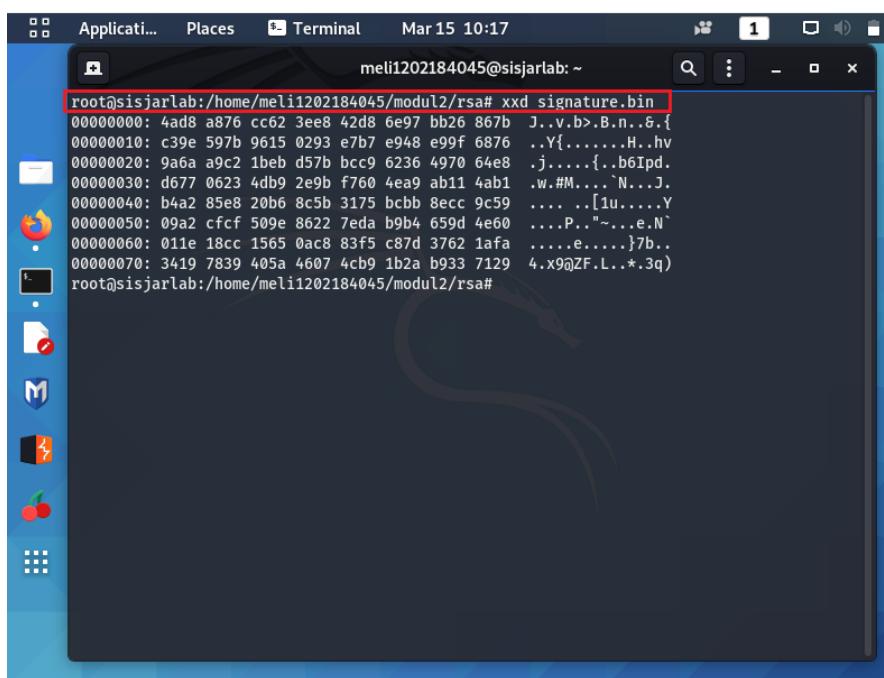
```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -verify -pubin -inkey rsapublickey.pem -in signature.bin -out plain_dec.txt
```



```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -verify -pubin -inkey rsa publickey.pem -in signature.bin -out plain_dec.txt
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

3. Untuk melihat hasil dari file *signature.bin* gunakan *command* berikut.

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# xxd signature.bin
```

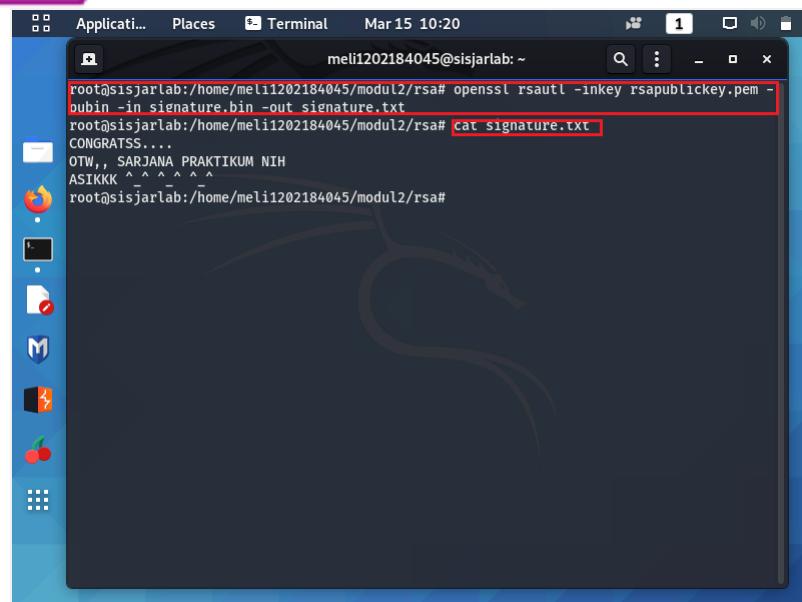


```
root@sisjarlab:/home/meli1202184045/modul2/rsa# xxd signature.bin
00000000: 4ad8 a876 cc62 3ee8 42d8 6e97 bb26 867b J..v.b>.B.n..&.{ 00000010: c39e 597b 9615 0293 e7b7 e948 e99f 6876 ..Y{.....H..hv 00000020: 9a6a a9c2 1beb d57b bcc9 6236 4970 64e8 .j.....{.b6Ipd. 00000030: d677 0623 4db9 2e9b f760 4ea9 ab11 4ab1 .w.#M...`N...J. 00000040: b4a2 85e8 20b6 8c5b 3175 bccb 8ecc 9c59 ..... [1u.....Y 00000050: 09a2 cfcf 509e 8622 7eda b9b4 659d 4e60 ....P..`~...e.N` 00000060: 011e 18cc 1565 0ac8 83f5 c87d 3762 1afa .....e....}7b.. 00000070: 3419 7839 405a 4607 4cb9 1b2a b933 7129 4.x9@ZF.L.*.3q)
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

4. Kemudian untuk mengubah *signature* agar dapat dibaca, gunakan *command* berikut.

```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -inkey rsa publickey.pem -pubin -in signature.bin -out signature.txt
```

```
root@sisjarlab:/home/sisjar/meli1202184045/rsa# cat signature.txt
```



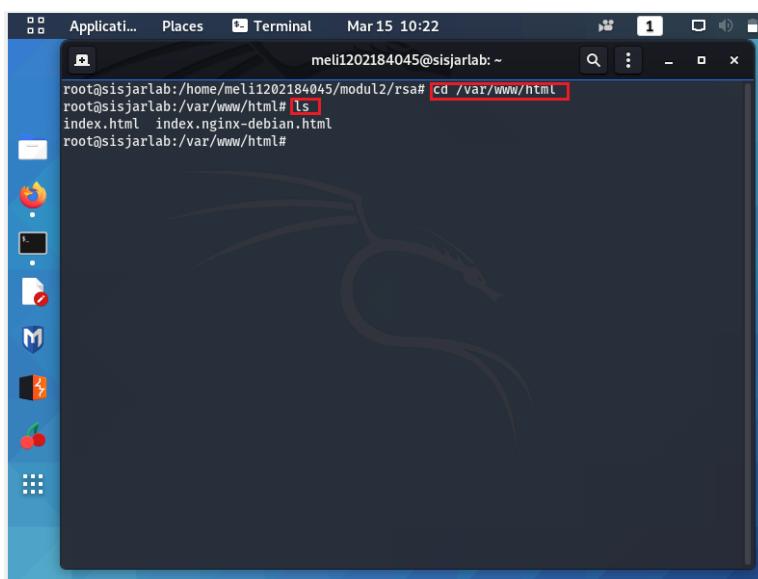
```
root@sisjarlab:/home/meli1202184045/modul2/rsa# openssl rsautl -inkey rsapublickey.pem -dubin -in signature.bin -out signature.txt
root@sisjarlab:/home/meli1202184045/modul2/rsa# cat signature.txt
CONGRATSS.....
OTW,, SARJANA PRAKTIKUM NIH
ASIKKK ^ ^ ^ ^ ^
root@sisjarlab:/home/meli1202184045/modul2/rsa#
```

4.5 Membuat sertifikat SSL

1. Masuk ke direktori *web server*

```
root@sisjarlab:~# cd /var/www/html
```

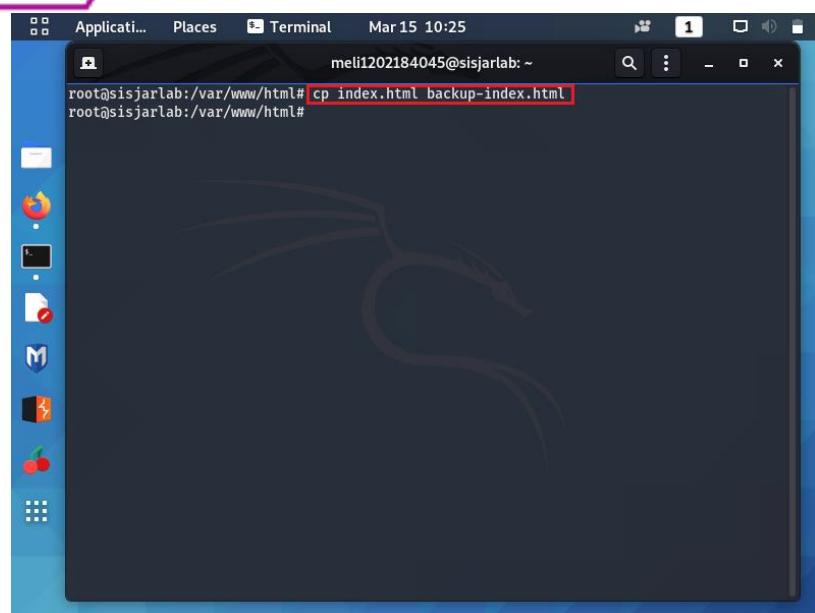
```
root@sisjarlab:/var/www/html# ls
```



```
root@sisjarlab:/home/meli1202184045/modul2/rsa# cd /var/www/html
root@sisjarlab:/var/www/html# ls
index.html index.nginx-debian.html
root@sisjarlab:/var/www/html#
```

2. Lakukan *backup* untuk file index.html dengan command:

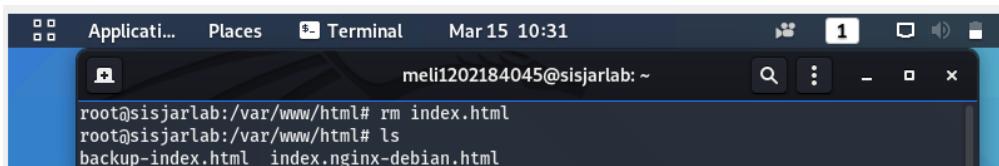
```
root@sisjarlab:/var/www/html# cp index.html backup-index.html
```



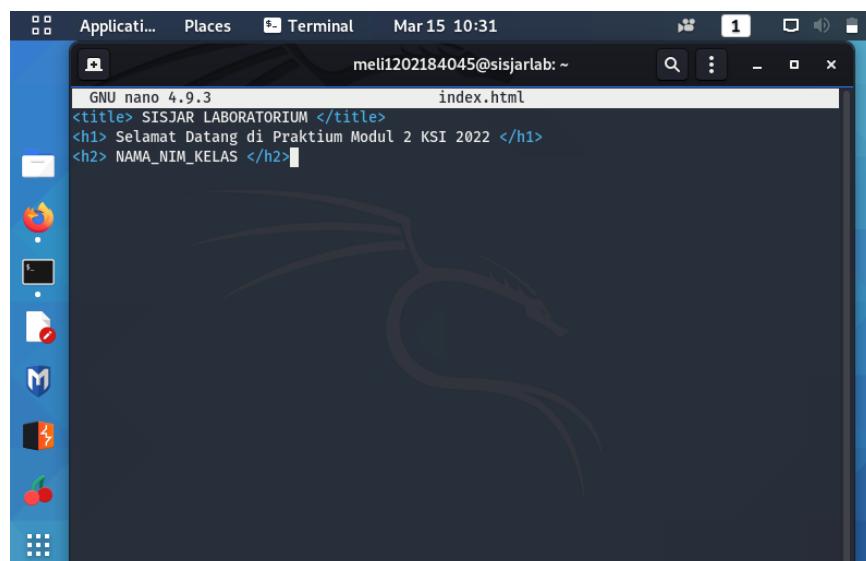
3. Hapus file index.html lalu buat index.html baru dengan *command* berikut.

```
root@sisjarlab:/var/www/html# rm index.html
```

```
root@sisjarlab:/var/www/html# ls
```



```
root@sisjarlab:/var/www/html# nano index.html
```



4. Cek IP address pada perangkat Kali Linux dengan *command* berikut.

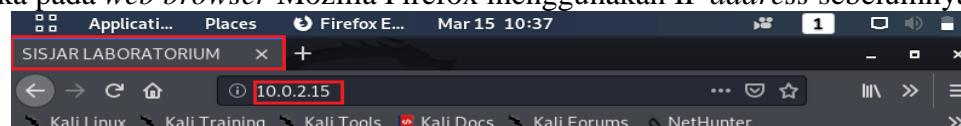
```
root@sisjarlab:/var/www/html# ifconfig
```

```
root@sisjarlab:/var/www/html# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::a00:2ff:fe:6235 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:c6:62:35 txqueuelen 1000 (Ethernet)
            RX packets 19515 bytes 28007504 (26.7 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 7006 bytes 469428 (458.4 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 76 bytes 25765 (25.1 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 76 bytes 25765 (25.1 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@sisjarlab:/var/www/html#
```

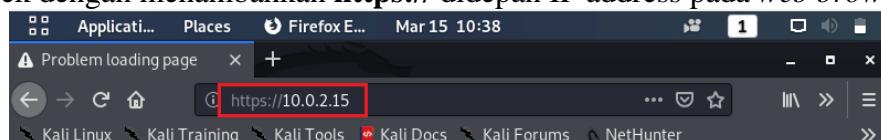
5. Buka pada *web browser* Mozilla Firefox menggunakan IP *address* sebelumnya.



Selamat Datang di Praktium Modul 2 KSI 2022

NAMA_NIM_KELAS

6. Lalu cek dengan menambahkan **https://** didepan IP address pada *web browser*



Unable to connect

Firefox can't establish a connection to the server at
10.0.2.15.



- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

[Try Again](#)

7. Keluar dari direktori dengan mengganti kata sisjar menjadi "**namaNIM**" masing masing praktikan

```
root@sisjarlab:/var/www/html# cd /home/meli1202184045
```

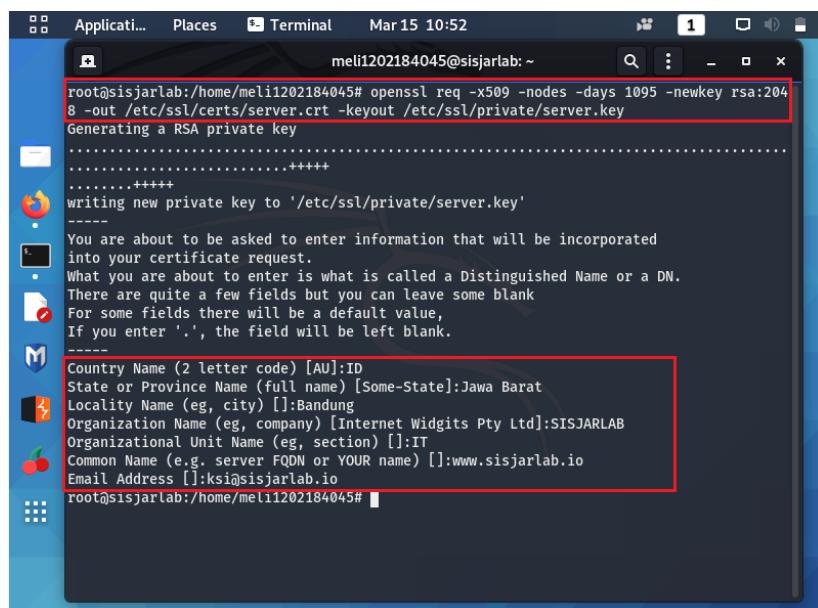
8. Sinkronisasi file .key dan file .csr yang sudah dibuat supaya informasi yang ada di file .csr bisa di terima oleh file .key dengan menggunakan *command* berikut

```
root@sisjarlab:/home/meli1202184045# openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -out /etc/ssl/certs/server.crt -keyout /etc/ssl/private/server.key
```

Note: Pastikan semua command diketikkan dalam satu baris perintah

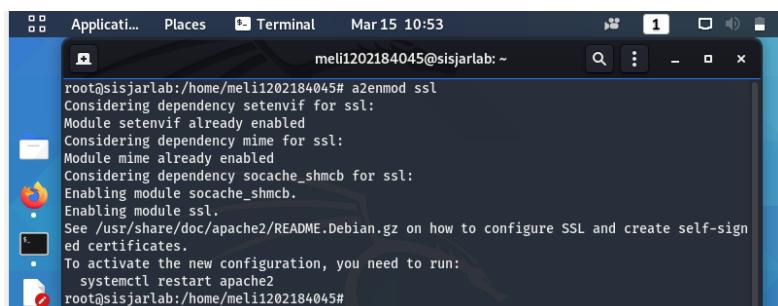
Ketentuan:

Country Name	: ID
State or Province Name	: Jawa Barat
Locality Name	: Bandung
Organization Name	: SISJARLAB
Organization Unit Name	: IT
Common Name	: www.sisjarlab.io
Email Address	: ksi@sisjarlab.io



9. Untuk mengaktifkan SSL di apache2

```
root@sisjarlab:/home/meli1202184045# a2enmod ssl
```

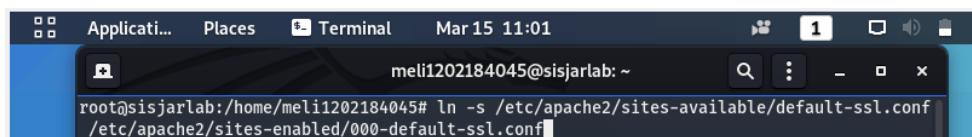


10. Lalu lakukan restart dengan perintah berikut

```
root@sisjarlab:/home/meli1202184045# systemctl restart apache2
```

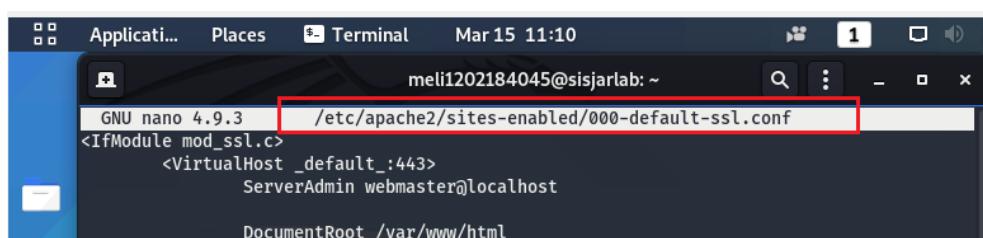
11. Membuat symbolic link untuk file 000-default-ssl.conf

```
root@sisjarlab:/home/meli1202184045# ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-enabled/000-default-ssl.conf
```



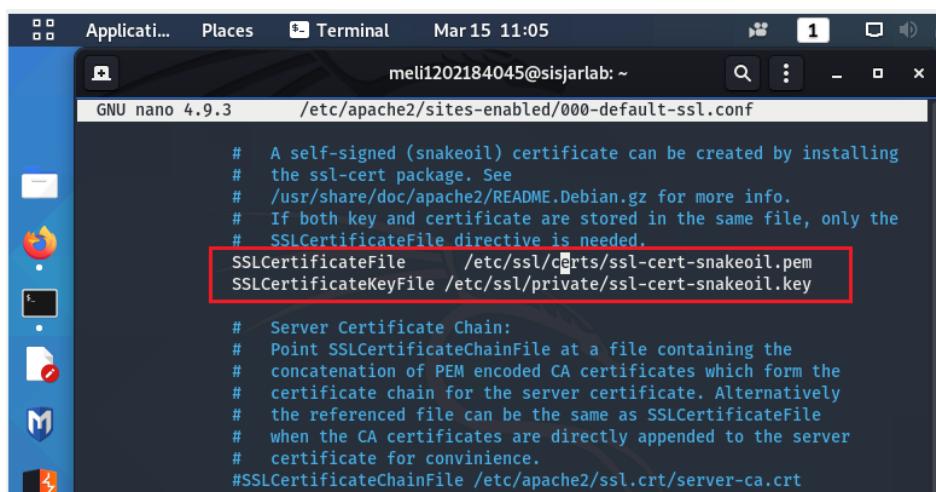
12. Menggunakan teks editor nano, edit file 000-default-ssl.conf dengan command

```
root@sisjarlab:/home/meli1202184045# nano /etc/apache2/sites-enabled/000-default-ssl.conf
```

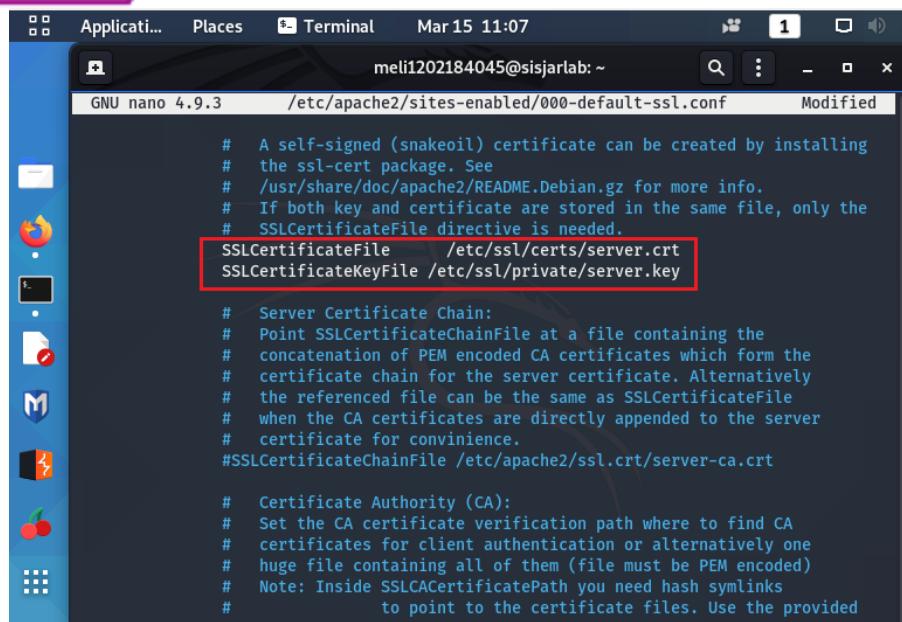


13. Kemudian temukan baris seperti dibawah lalu edit lokasi file menjadi:

```
SSLCertificateFile /etc/ssl/certs/server.crt  
SSLCertificateKeyFile /etc/ssl/private/server.key
```



Hasil dari baris yang sudah diubah :



The screenshot shows a terminal window titled "meli1202184045@sisjarlab: ~" running the command "nano /etc/apache2/sites-enabled/000-default-ssl.conf". The configuration file contains several comments and two lines highlighted with a red box: "#SSLCertificateFile /etc/ssl/certs/server.crt" and "#SSLCertificateKeyFile /etc/ssl/private/server.key". The rest of the file includes sections for certificate chains and CA verification paths.

```
GNU nano 4.9.3      /etc/apache2/sites-enabled/000-default-ssl.conf      Modified

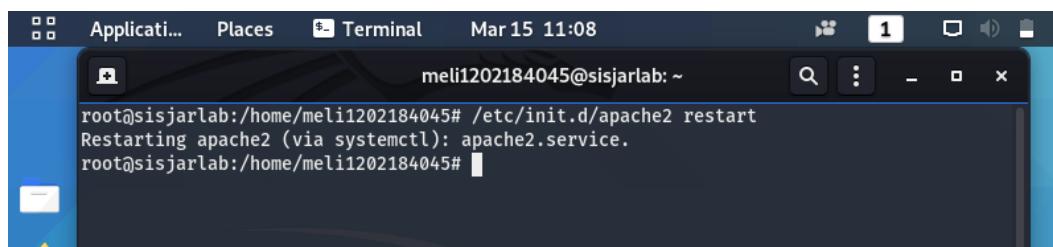
# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
#SSLCertificateFile      /etc/ssl/certs/server.crt
#SSLCertificateKeyFile  /etc/ssl/private/server.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
# to point to the certificate files. Use the provided
```

14. Selanjutnya restart *service apache2* kembali dengan command:

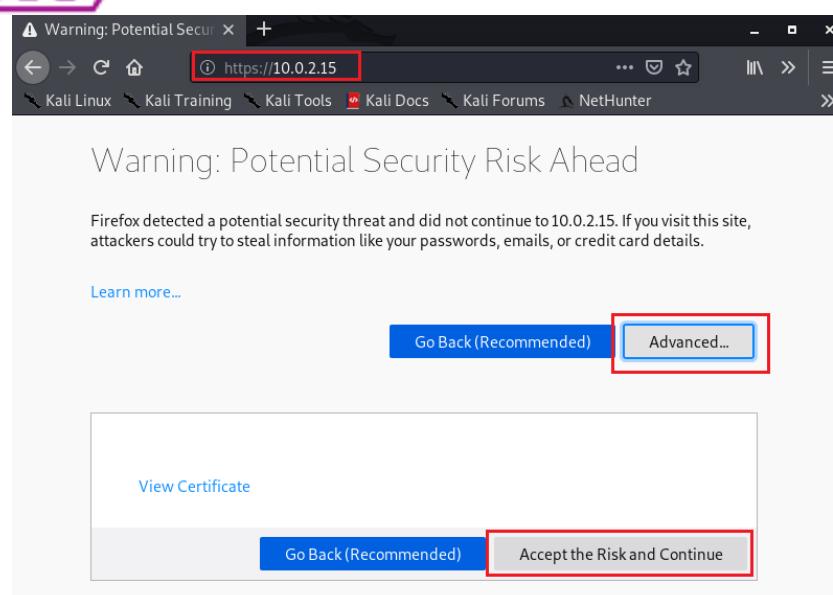
```
root@sisjarlab:/home/meli1202184045# /etc/init.d/apache2 restart
```



The screenshot shows a terminal window titled "meli1202184045@sisjarlab: ~" running the command "/etc/init.d/apache2 restart". The output shows the service is restarting and then prompt for another command.

```
meli1202184045@sisjarlab: ~
root@sisjarlab:/home/meli1202184045# /etc/init.d/apache2 restart
Restarting apache2 (via systemctl): apache2.service.
root@sisjarlab:/home/meli1202184045#
```

15. Buka kembali *web browser* Mozilla Firefox pada Kali Linux dan akses halaman *web apache2* dengan menambahkan **https://** di depan *IP address* masing-masing. Lalu pilih *Advanced* dan pilih *Accept the Risk and Continue*.



16. Berikut hasil dari penambahan sertifikat SSL pada *web server apache2*.



V. Daftar Pustaka

1. Laboratorium Sistem Operasi dan Jaringan Komputer. (2021). *Modul Praktikum Keamanan Sistem Informasi 2021*. Bandung, Laboratorium Sistem Operasi dan Jaringan Komputer.