Jobsheet 14
SYMMETRIC CRYPTOGRAPHY

A. TUJUAN

1. Mengenalkan pada mahasiswa tentang konsep cryptography
2. Mahasiswa mampu membuat program enkripsi Caesar dan RC4
3. Mahasiswa mampu membuat program dekripsi Caesar dan RC4

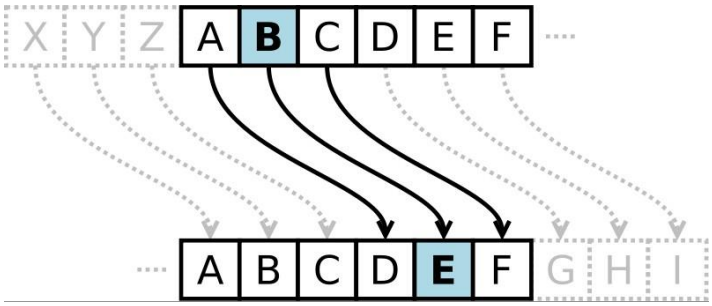
B. DASAR TEORI

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan. Terdapat 2 jenis kriptografi dipandang dari masanya :

1. Kriptografi klasik : Caesar cipher, Affine cipher, Vigenere cipher dll.
2. Kriptografi modern, terbagi 2 yaitu :
   a. Kriptografi simetrik : RC4, DES, AES, IDEA
   b. Kriptografi asimetrik : RSA, DSA, El gama

Kriptografi Klasik (Caesar)

Pada Caesar cipher, tiap huruf disubstitusi dengan huruf ketiga berikutnya dari susunan alphabet yang sama. Dalam hal ini kuncinya adalah jumlah pergeseran huruf (yaitu 3). Susunan alphabet setelah digeser sejauh 3 huruf membentuk sebuah table substitusi sebagai berikut :



Gambar 1. Kriptografi Caesar

Kriptografi Simetrik

Kriptografi simetrik atau dikenal pula sebagai kriptografi kunci rahasia, merupakan kriptografi yang menggunakan kunci yang sama baik untuk proses enkripsi maupun dekripsi. Secara metematis dapat dinyatakan bahwa :

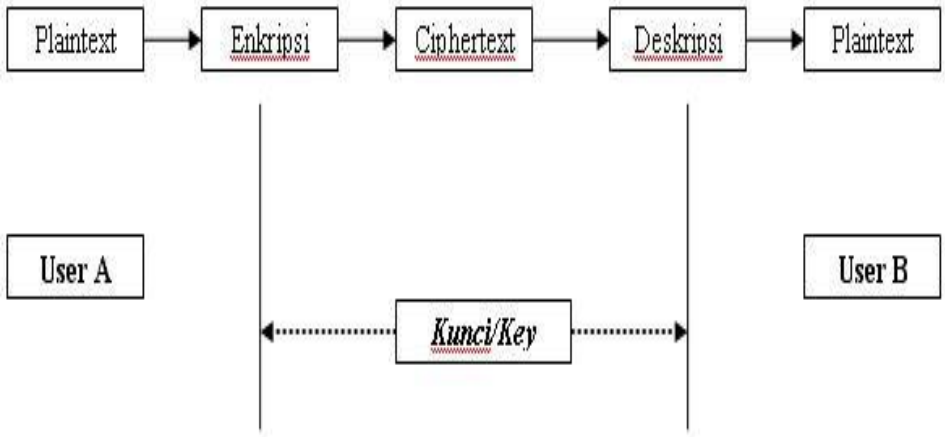
$$E = d = k \quad \text{.....(4}$$
$$E_k(m) = c \quad \text{.....(5}$$
$$D_k(c) = m \quad \text{.....(6}$$

Dalam algoritma *simetri*, kunci yang digunakan dalam proses *enkripsi* dan *dekripsi* adalah sama atau pada prinsipnya identik. Kunci ini pun bisa diturunkan dari kunci lainnya. Oleh karena itu sistem ini sering disebut *secret-key ciphersystem.*

Agar komunikasi tetap aman, kunci yang menggunakan teknik enkripsi ini harus betul-betul dirahasiakan.

Kriptografi simetrik sangat menekankan pada kerahasiaan kunci yang digunakan untuk proses enkripsi dan dekripsi. Oleh karena itulah kriptografi ini dinamakan pula

sebagai kriptografi kunci rahasia. Gambaran proses sederhana *enkripsi* dengan algoritma simetri:



Gambar 2. Blok Diagram algoritma Simetri

Algoritma RC4

RC4 merupakan merupakan salah satu jenis stream cipher, yaitu memproses unit atau input data pada satu saat. Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah input data tertentu sebelum diproses, atau menambahkan byte tambahan untuk mengenkrip. Metode enkripsi RC4 sangat cepat kurang lebih 10 kali lebih cepat dari DES.

RC4 merupakan stream cipher yang didesain oleh Rivest untuk RSA Data *Security* (sekarang *RSA Security*) pada 1987. RC4 menggunakan panjang variabel kunci dari 1 s.d 256 byte untuk menginisialisasi *state* tabel. *State table* digunakan untuk pengurutan menghasilkan *byte pseudo-random* yang kemudian menjadi *stream pseudo-random*. Setelah di-XOR dengan *plaintext* sehingga didapatkan *ciphertext*. Tiap elemen pada *state table* di *swap* sedikitnya sekali. Kunci RC4 sering dibatasi sampai 40 bit, tetapi dimungkinkan untuk mengunakan kunci 128 bit. RC4 memiliki kemampuan penggunaan kunci antara 1 sampai 2048 bit.

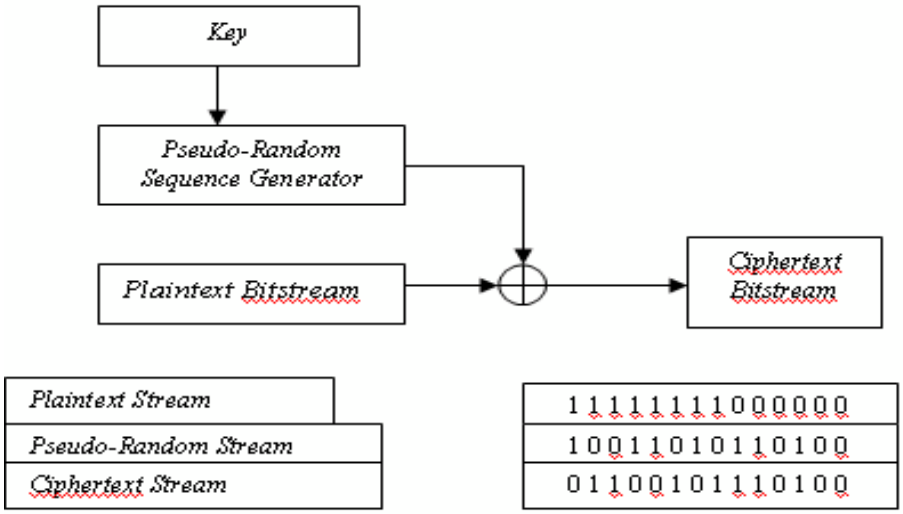
Panjang kunci merupakan faktor utama dalam sekuritas data. RC4 dapat memiliki kunci sampai dengan 128 bit. Protokol keamanan SSL (*Secure Socket Layer*) pada *Netscape Navigator* menggunakan algoritma RC4 40-bit untuk enkripsi simetrisnya.

Algoritma RC4 memiliki dua fase, setup kunci dan pengenkripsian. Setup untuk kunci adalah fase pertama dan yang paling sulit dalam algoritma ini. Dalam setup S-bit kunci (S merupakan panjang dari kunci), kunci enkripsi digunakan untuk menghasilkan variabel enkripsi yang menggunakan dua buah array, state dan kunci, dan sejumlah-S hasil dari operasi penggabungan. Operasi penggabungan ini terdiri dari pemindahan (*swapping*) byte, operasi modulo, dan rumus lain. Operasi modulo merupakan proses yang menghasilkan nilai sisa dari satu pembagian. Sebagai contoh, 11 dibagi 4 adalah 2 dengan sisa pembagian 3, begitu juga jika tujuh modulo empat maka akan dihasilkan nilai tiga.

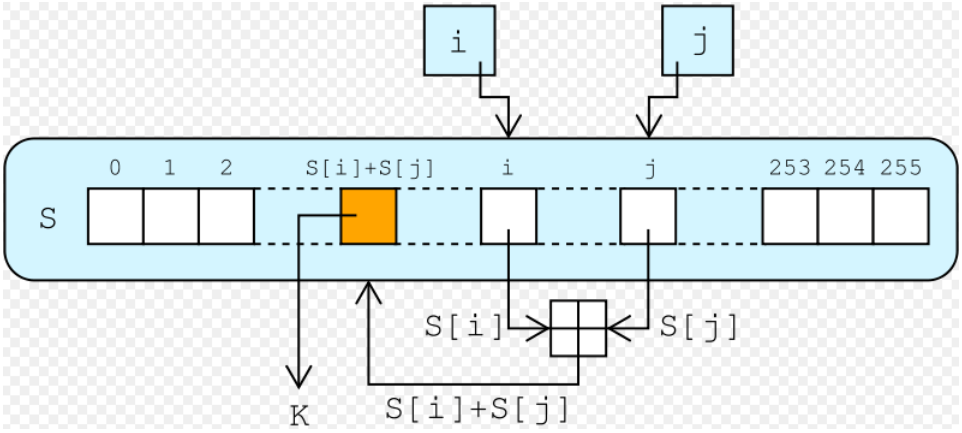
Variabel *enkripsi* dihasikan dari setup kunci dimana kunci akan di XOR-kan dengan *plain text* untuk menghasilkan teks yang sudah terenkripsi. XOR merupakan

operasi logik yang membandingkan dua bit biner. Jika bernilai beda maka akan dihasilkan nilai 1. Jika kedua bit sama maka hasilnya adalah 0. Kemudian penerima pesan akan mendekripnya dngan meng XOR-kan kembali dengan kunci yang sama agar dihasilkan pesan dari *plain text* tersebut.

Untuk menunjukan cara kerja dari algoritma RC4, berikut dapat dilihat pada blok di bawah :



Gambar 3. Blok Diagram algortima RC 4 secara umum



Gambar 4. Proses pembangkitan acak untuk kunci RC4

RC4 menggunakan dua buah kotak substitusi (S-Box) array 256 byte yang berisi permutasi dari bilangan 0 sampai 255 dan S-Box kedua yang berisi permutasi fungsi dari kunci dengan panjang yang variabel.

Cara kerja algoritma RC4 yaitu inisialisasi *Sbox* pertama, S[0],S[1],...,S[255], dengan bilangan 0 sampai 255. Pertama isi secara berurutan S[0] = 0, S[1] = 1,...,S[255] = 255. Kenudian inisialisasi *array* lain (*S-Box* lain), misal *array* K dengan panjang 256. Isi *array* K dengan kunci yang diulangi sampai seluruh *array* K[0], K[1],...,K[255] terisi seluruhnya.

Langkah-langkah algoritma RC4:
1. Proses inisialisasi S-Box (*Array* S)

```
For i = 0 to 255
     S[i] = i
```

2. Proses inisialisasi S-Box(Array K) untuk kunci.
Lakukan padding jika panjang kunci < 256.

```
Array Kunci // panjang kunci"length".
for i = 0 to 255
K[i] = Kunci[i mod length]
```

3.Kemudian lakukan langkah pengacakan S-Box dengan langkah sebagai berikut :

```
j = 0
For i = 0 to 255
  j = (j + S[i] + K[i]) mod 256
  isi S[i] dan isi S[j] ditukar
Endfor
```

4. Dengan demikian berakhirlah proses persiapan kunci RC4. Untuk membangkitkan kunci enkripsi, dilakukan proses sebagai berikut:

```
i = 0
j = 0
for idx = 0 to plainteks-1 do
     i = (i + 1) mod 256
     j = (j + S[i]) mod 256
     isi S[i] dan S[j] ditukar
     t = (S[i] + S[j]) mod 256
     k = S [t]
     c = P[idx] ⊕ k
endfor
```

Perhatikan bahwa k kecil merupakan kunci yang langsung beroperasi terhadap plainteks, sedangkan K besar adalah kunci utama atau kunci induk

Bila terdapat plainteks P, maka operasi enkripsi berupa :

```
C = P ⊕ k
```

Sedangkan operasi dekripsi berupa :

```
P = k ⊕ C
```

C. TUGAS PENDAHULUAN

1. Jelaskan konsep kriptografi simetrik dan asimetrik ?
   Kriptografi Simetrik: Kriptografi simetrik juga dikenal sebagai kriptografi kunci tunggal atau kriptografi rahasia. Konsep ini melibatkan penggunaan kunci yang sama untuk melakukan enkripsi dan dekripsi data. Artinya, penerima pesan menggunakan kunci yang sama yang digunakan oleh pengirim untuk mengenkripsi pesan tersebut. Kunci simetrik harus dijaga dengan aman karena jika jatuh ke tangan yang salah, orang tersebut dapat mengakses dan membaca pesan yang terenkripsi.

Kriptografi Asimetrik: Kriptografi asimetrik juga dikenal sebagai kriptografi kunci ganda atau kriptografi publik. Konsep ini melibatkan penggunaan sepasang kunci yang berbeda, yaitu kunci publik dan kunci pribadi. Kunci publik digunakan untuk enkripsi data, sedangkan kunci pribadi digunakan untuk dekripsi data. Kunci publik dapat didistribusikan secara terbuka kepada siapa pun, sedangkan kunci pribadi harus dijaga dengan sangat rahasia.

2. Jelaskan fase dari algoritma RC4 ?

Algoritma RC4 (Rivest Cipher 4) adalah sebuah algoritma kriptografi simetrik yang digunakan untuk enkripsi dan dekripsi data. Algoritma ini terdiri dari beberapa fase yang menjalankan serangkaian operasi untuk menghasilkan aliran kunci yang digunakan untuk melakukan enkripsi atau dekripsi data. Berikut adalah empat fase utama dalam algoritma RC4:

1. Inisialisasi Kunci: Pada fase ini, kunci yang akan digunakan untuk enkripsi atau dekripsi diinisialisasi. Kunci tersebut biasanya berupa deretan byte yang diberikan oleh pengguna. Selain itu, dalam tahap ini, sebuah array bernama S-box (Substitution Box) juga dibuat dengan mengisi elemennya dengan angka 0 hingga 255 secara berurutan.
2. Pencampuran Kunci (Key Scheduling): Dalam tahap ini, algoritma melakukan pencampuran kunci dengan menggunakan algoritma permutasi. Pada setiap langkahnya, algoritma melakukan pertukaran elemen-elemen pada array S-box berdasarkan nilai kunci. Proses ini dilakukan untuk mengacak elemen-elemen dalam S-box dan menciptakan hubungan yang kompleks antara kunci dan S-box.
3. Generasi Aliran Kunci (Pseudo-Random Generation): Setelah fase pencampuran kunci selesai, algoritma mulai menghasilkan aliran kunci yang akan digunakan untuk enkripsi atau dekripsi. Pada setiap langkah, algoritma melakukan operasi swap dan penjumlahan modulo pada elemen-elemen dalam S-box untuk menghasilkan byte aliran kunci. Byte aliran kunci ini dihasilkan secara berulang-ulang hingga mencukupi untuk digunakan dalam proses enkripsi atau dekripsi.
4. Enkripsi atau Dekripsi: Dalam fase terakhir, aliran kunci yang dihasilkan digunakan untuk melakukan enkripsi atau dekripsi data. Algoritma RC4 menggunakan operasi XOR (exclusive OR) antara byte aliran kunci dan byte-data asli untuk menghasilkan byte-data terenkripsi atau terdekripsi.

3. Jelaskan cara kerja algoritma RC4 ?

Algoritma RC4 (Rivest Cipher 4) adalah algoritma kriptografi simetrik yang menggunakan aliran kunci (stream cipher) untuk melakukan enkripsi dan dekripsi data. Berikut adalah langkah-langkah umum dalam cara kerja algoritma RC4:

1. Inisialisasi Kunci:
- Kunci yang diberikan oleh pengguna dijadikan sebagai input awal algoritma.
- Algoritma membuat array S-box (Substitution Box) dengan mengisi elemennya dengan angka 0 hingga 255 secara berurutan.
- Array S-box diacak berdasarkan nilai kunci yang diberikan.

2. Pencampuran Kunci (Key Scheduling):
- Dalam tahap ini, pertukaran elemen-elemen pada array S-box dilakukan berdasarkan nilai kunci yang telah diacak.
- Operasi swap dan penjumlahan modulo digunakan untuk memperoleh permutasi yang kompleks dalam array S-box.

3. Generasi Aliran Kunci (Pseudo-Random Generation):
- Algoritma RC4 menghasilkan aliran kunci yang digunakan untuk enkripsi atau dekripsi data.

- Dalam setiap langkahnya, operasi swap dan penjumlahan modulo dilakukan pada elemen-elemen dalam array S-box untuk menghasilkan byte aliran kunci.
- Byte aliran kunci dihasilkan secara berulang-ulang hingga mencukupi untuk proses enkripsi atau dekripsi.

4. Enkripsi atau Dekripsi:
- Setelah langkah generasi aliran kunci selesai, byte aliran kunci yang dihasilkan akan digunakan untuk melakukan operasi XOR (exclusive OR) dengan byte-data asli.
- Operasi XOR dilakukan pada setiap byte-data asli dengan byte aliran kunci yang sesuai untuk menghasilkan byte-data terenkripsi atau terdekripsi.

5. Output Data Terenkripsi atau Terdekripsi:
- Setelah proses XOR selesai, data asli akan berubah menjadi data terenkripsi atau terdekripsi, tergantung pada operasi yang dilakukan.

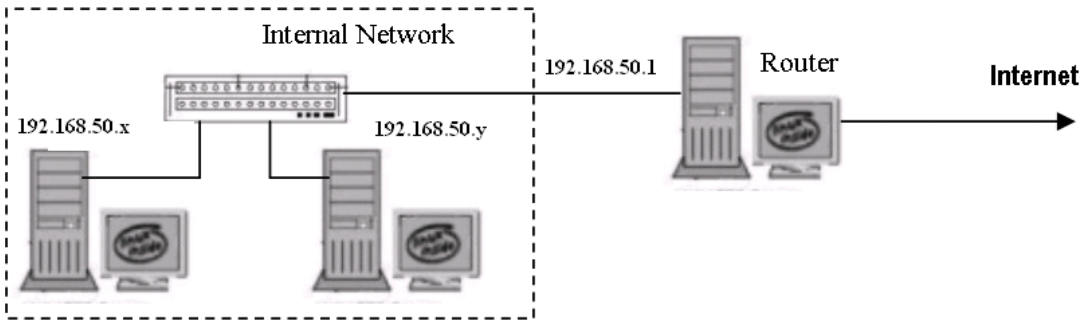
Perlu dicatat bahwa algoritma RC4 adalah algoritma stream cipher yang relatif sederhana dan efisien dalam menghasilkan aliran kunci. Namun, keamanan RC4 telah dikompromikan dan tidak lagi direkomendasikan untuk penggunaan yang kritis. Seiring berjalannya waktu, serangan dan kerentanan terhadap RC4 telah ditemukan, sehingga algoritma ini sebaiknya digantikan dengan algoritma kriptografi yang lebih aman.

D. ALAT DAN BAHAN

1. PC
2. XAMPP
3. Web Browser

E. PERCOBAAN

1. Bangunlah jaringan seperti berikut :



Gambar 5 Jaringan Percobaan

NB:
Gunakan dhclient di masing-masing PC untuk mendapatkan IP dari router.
192.168.50.x & y : IP dari router
Pilih 192.168.50.x sebagai PC Server
Pilih 192.168.50.y sebagai PC Client

2. Instalasi webserver dan php

a. Lakukan instalasi apache2 php5 pada PC Server :

```
# apt-get install apache2
# apt-get install php5 libapache2-mod-php5
```

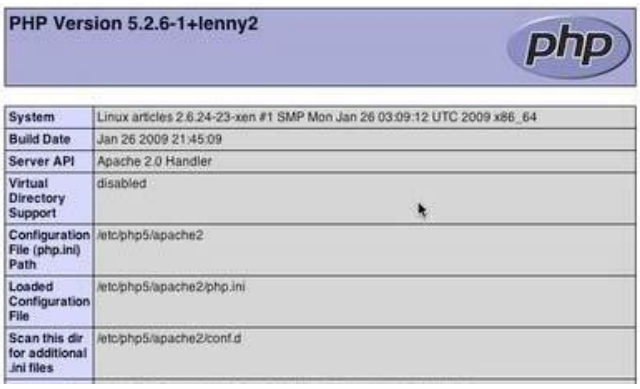
b. Restart apache :

```
# /etc/init.d/apache2 restart
```

c. Buat file php sebagai berikut :

```
# vim /var/www/info.php
<?php
phpinfo();
?>
```

d. Tes konfigurasi dengan mengakses dari PC Client, buka web browser di PC Client dan masukkan alamat :
http://<no_ip_pc_server>/info.php



Gambar 6 Hasil dari php

3. Kriptografi klasik (Caesar cipher)

Pembuatan Form Masukan PHP

a. Buat file untuk masukan plainteks dan key (berupa bilangan), beri nama file : awal.php di PC Server

```
<html>
<head>
<title>FORM UNTUK ENKRIPSI</title>
</head>
<body>
<form action="enkcaesar.php" method="get">
Plainteks     : <input type="text" name="kata"> <br>
Key    : <input type="text" name="key" maxlength="2"> <br>
<input type="submit" value="kirim">
<input type="reset" value="ulangi">
</form>
</body>
</html>
```

Proses Enkripsi dengan Caesar Algorithm

b. Buat file untuk melakukan proses enkripsi, beri nama file : enkcaesar.php di PC Server

```
<?php
$kalimat = $_GET["kata"];
$key = $_GET["key"];
for($i=0;$i<strlen($kalimat);$i++)
 {
        $kode[$i]=ord($kalimat[$i]); //rubah ASCII ke desimal
        $b[$i]=($kode[$i] + $key ) % 256; //proses enkripsi
        $c[$i]=chr($b[$i]); //rubah desimal ke ASCII
 }
 echo "kalimat ASLI : ";
 for($i=0;$i<strlen($kalimat);$i++)
 {
     echo $kalimat[$i];
 }
 echo "<br>";
 echo "hasil enkripsi =";
 $hsl = '';
 for ($i=0;$i<strlen($kalimat);$i++)
 {
      echo $c[$i];
      $hsl = $hsl . $c[$i];
 }
 echo "<br>";
 //simpan data di file
 $fp = fopen ("enkripsi.txt","w");
 fputs ($fp,$hsl);
 fclose($fp);
?>
```

Tes Proses Enkripsi

c. Buka web browser dari PC Client dan akseslah file php dari PC Server http://<no_ip_pc_server>/awal.php

d. Catat hasil enkripsi diatas.

Pembuatan Form untuk proses dekripsi

e. Buat file untuk masukan key (berupa bilangan), agar bisa menghasilkan kembali plainteks maka key harus sama dengan proses enkripsi, beri nama file: akhir.php di PC Server

```
<html>
<head>
   <title>Form untuk Dekripsi</title>
</head>
<body>
<form action="dekCaesar.php" method="get">
Key : <input type="text" name="key" maxlength="2"> <br>
<input  type="submit"  value="kirim">
<input type="reset" value="ulangi">
</form>
</body>
</html>
```

Proses Dekripsi dengan Caesar Algorithm

f. Buat file untuk melakukan proses dekripsi, beri nama file : dekCaesar.php di PC Server

```
<?php
$key = $_GET["key"];
$nmfile = "enkripsi.txt";
$fp = fopen($nmfile,"r"); // buka file hasil enkripsi
$isi = fread($fp,filesize($nmfile));

for($i=0;$i<strlen($isi);$i++)
 {
        $kode[$i]=ord($isi[$i]); // rubah ASII ke desimal
        $b[$i]=($kode[$i] - $key ) % 256; // proses dekripsi Caesar
        $c[$i]=chr($b[$i]); //rubah desimal ke ASCII
 }
 echo "kalimat ciphertext : ";
 for($i=0;$i<strlen($isi);$i++)
 {
     echo $isi[$i];
 }
 echo "<br>";
 echo "hasil dekripsi =";
  for ($i=0;$i<strlen($isi);$i++)
 {
      echo $c[$i];
 }
 echo "<br>";
?>
```

Tes Proses Dekripsi

g. Buka web browser dari PC Client dan akseslah file php dari PC Server http://<no_ip_pc_server>/akhir.php
h. Catat hasil dekripsi diatas.
i. Ubah-ubahlah nilai key, dan catat hasilnya.
j. Ulangi proses 3.c dan masukkan kata yang sama dan berulang-ulang. Setelah itu analisa hasilnya.

4. Kriptografi Modern (Simetrik RC4)

Pembuatan Form Masukan PHP

a. Gunakan kembali file di poin 3.a, beri nama yang berbeda : awalrc4.php . Buat di PC Server, dan rubah hanya baris berikut :

```
...
<form action="penkripsi.php" method="get">
Plainteks     : <input type="text" name="kata"> <br>
Key    : <input type="text" name="kcenkripsi" maxlength="16"> <br>
...
```

NB : untuk kunci, dimasukkan kata tanpa spasi sebanyak 16 karakter.

Proses Pembentukan Kunci Enkripsi dengan RC4 Algorithm

b. Buat file untuk memproses setupkey dan enkripsi RC4, beri nama file penkripsi.php

c. Buat program untuk setupkey :

```
function setupkey() //proses pengacakan kunci di SBox
{
echo "<br>";
$kce = $_GET["kcenkripsi"];
echo "Kunci enkripsi = $kce";
echo "<br>";
       for($i=0;$i<strlen($kce);$i++)
        {
             $key[$i]=ord($kce[$i]); //rubah ASCII ke desimal
        }

 global $m;
 $m=array();

 // buat encrpyt
 for($i=0;$i<256;$i++){
         $m[$i] = $i;
}
 $j = $k = 0;
 for($i=0;$i<256;$i++)
 {
  $a = $m[$i];
  $j = ($j + $m[$i] + $key[$k]) % 256;
  $m[$i] = $m[$j];
  $m[$j] = $a;
  $k++;
  if($k>15)
  {
  $k=0;
  }
 }
} //akhir function
```

Proses Enkripsi Algoritma RC4

d. Tambahkan program untuk enkripsi RC4 dibawah fungsi setupkey

```
function crypt2($inp)
{
 global $m;
 $x=0;$y=0;
 $bb='';
 $x = ($x+1) % 256;
 $a = $m[$x];
 $y = ($y+$a) % 256;
 $m[$x] = $b = $m[$y];
 $m[$y] = $a;
 //proses XOR antara plaintext dengan kunci
 //dengan $inp sebagai plaintext
 //dan $m sebagai kunci
 $bb= ($inp^$m[($a+$b) % 256]) % 256;
 return $bb;
}
```

e. Tampilkan kalimat asli dan hasil enkripsi RC4

```
$kalimat = $_GET["kata"];

setupkey();
for($i=0;$i<strlen($kalimat);$i++)
 {
      $kode[$i]=ord($kalimat[$i]); //rubah ASCII ke desimal
         $b[$i]=crypt2($kode[$i]); //proses enkripsi RC4
         $c[$i]=chr($b[$i]); //rubah desimal ke ASCII
 }
 echo "kalimat ASLI : ";
 for($i=0;$i<strlen($kalimat);$i++)
 {
     echo $kalimat[$i];
 }
 echo "<br>";
 echo "hasil enkripsi =";
 $hsl = '';
  for ($i=0;$i<strlen($kalimat);$i++)
 {
      echo $c[$i];
      $hsl = $hsl . $c[$i];
 }
 echo "<br>";
 //simpan data  di file
 $fp = fopen ("enkripsirc4.txt","w");
 fputs ($fp,$hsl);
 fclose($fp);
```

Tes Proses Enkripsi

f. Buka web browser dari PC Client dan akseslah file php dari PC Server http://<no_ip_pc_server>/awalrc4.php

g. Catat hasil enkripsi diatas.

Pembuatan Form untuk proses dekripsi

h. Gunakan kembali file di 3.e. dan rubah beberapa baris berikut : Buat file untuk masukan key (berupa bilangan), agar bisa menghasilkan kembali plainteks maka key harus sama dengan proses enkripsi, beri nama file: akhirrc4.php di PC Server

```
…
<form action="pdekripsi.php" method="get">
Key : <input type="text" name="kcdekripsi" maxlength="16"> <br>
...
```

NB : agar bisa menghasilkan kembali plainteks maka key harus sama dengan proses enkripsi

Proses Pembentukan Kunci Dekripsi dengan RC4 Algorithm

i. Buat file untuk memproses setupkey dan enkripsi RC4, beri nama file pdekripsi.php

j. Buat program untuk setupkey (proses ini sama dengan proses pembentukan kunci untuk enkripsi) :

```
function setupkey()
{
 $kcd = $_GET["kcdekripsi"];
 echo "Kunci Dekripsi = $kcd";
 echo "<br>";
       for($i=0;$i<strlen($kcd);$i++)
        {
             $key[$i]=ord($kcd[$i]); //rubah ASCII ke desimal
        }
 global $mm;
 $mm=array();
 // buat decrpyt
 $mm=array();
 for($i=0;$i<256;$i++)
          $mm[$i] = $i;

 $j = $k = 0;
 for($i=0;$i<256;$i++)
 {
  $a = $mm[$i];
  $j = ($j + $a + $key[$k]) % 256;
  $mm[$i] = $mm[$j];
  $mm[$j] = $a;
  $k++;
  if($k>15)
  {
  $k=0;
  }
 }
} //akhir function
```

Proses Dekripsi Algoritma RC4

k. Tambahkan program untuk dekripsi RC4 dibawah fungsi setupkey :

```
function decrypt2($inp)
{
 global $mm;
 $xx=0;$yy=0;
 $bb='';
 $xx = ($xx+1) % 256;
 $a = $mm[$xx];
 $yy = ($yy+$a) % 256;
 $mm[$xx] = $b = $mm[$yy];
 $mm[$yy] = $a;
 //proses XOR antara ciphertext dengan kunci
 //dengan $inp sebagai ciphertext
 //dan $m sebagai kunci
 $bb = ($inp^$mm[($a+$b) % 256]) % 256;
 return $bb;
}
```

l. Tampilkan hasil dekripsi RC4

```
setupkey();
$nmfile = "enkripsirc4.txt";
//ambil data dari file enkripsirc4.txt
$fp = fopen($nmfile,"r");
$isi = fread($fp,filesize($nmfile));
echo "Ciphertext : $isi"."<br>";

  for($i=0;$i<strlen($isi);$i++)
 {
        $b[$i]=ord($isi[$i]); // rubah ASCII ke desimal
        $d[$i]=decrypt2($b[$i]); // proses dekripsi RC4
        $s[$i]=chr($d[$i]);  // rubah desimal ke ASCII
 }

 echo "hasil dekripsi =  ";
 for ($i=0;$i<strlen($isi);$i++)
 {
      echo $s[$i];
 }
 echo "<br>";
```

Tes Proses Dekripsi

m. Buka web browser dari PC Client dan akseslah file php dari PC Server http://<no_ip_pc_server>/akhirrc4.php

n. Catat hasil dekripsi diatas.

o. Ubah-ubahlah nilai key, dan catat hasilnya.

p. Ulangi proses 4.f dan masukkan kata yang sama dan berulang-ulang. Setelah itu analisa hasilnya.

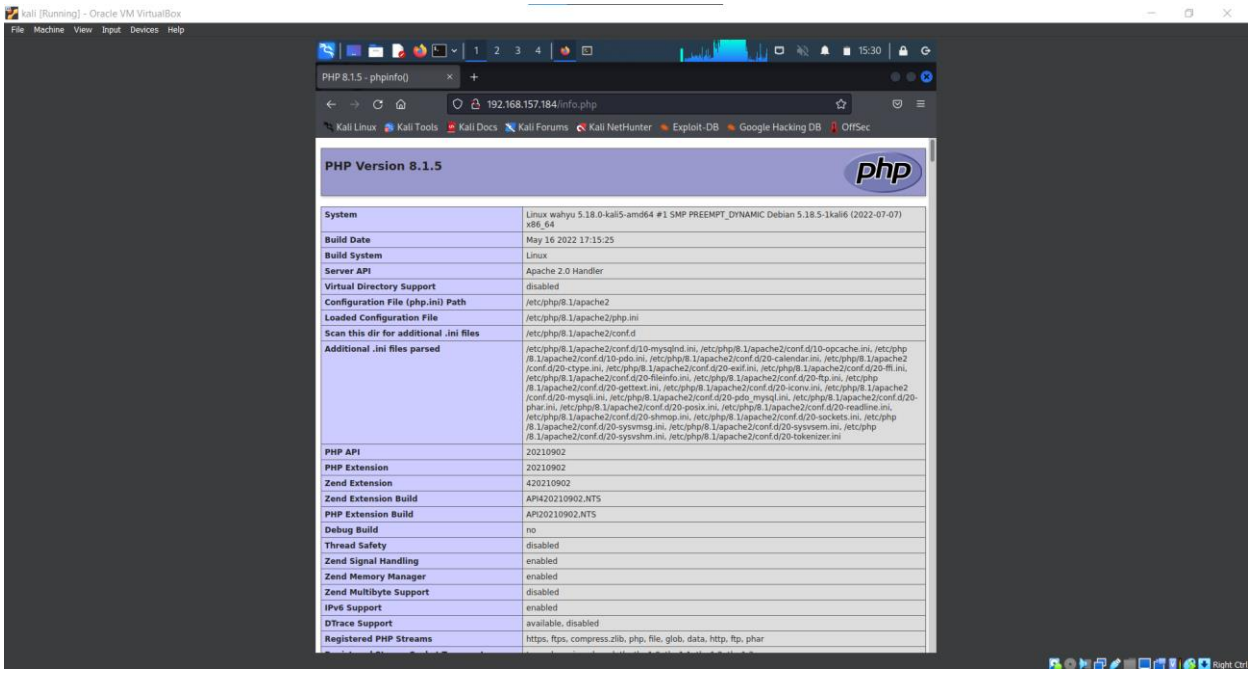
F. LAPORAN RESMI

1. Berikan kesimpulan hasil praktikum yang anda lakukan.
2. Simpulkan perbedaan sistem kriptografi simetris berbasis block dan stream.

LEMBAR ANALISA

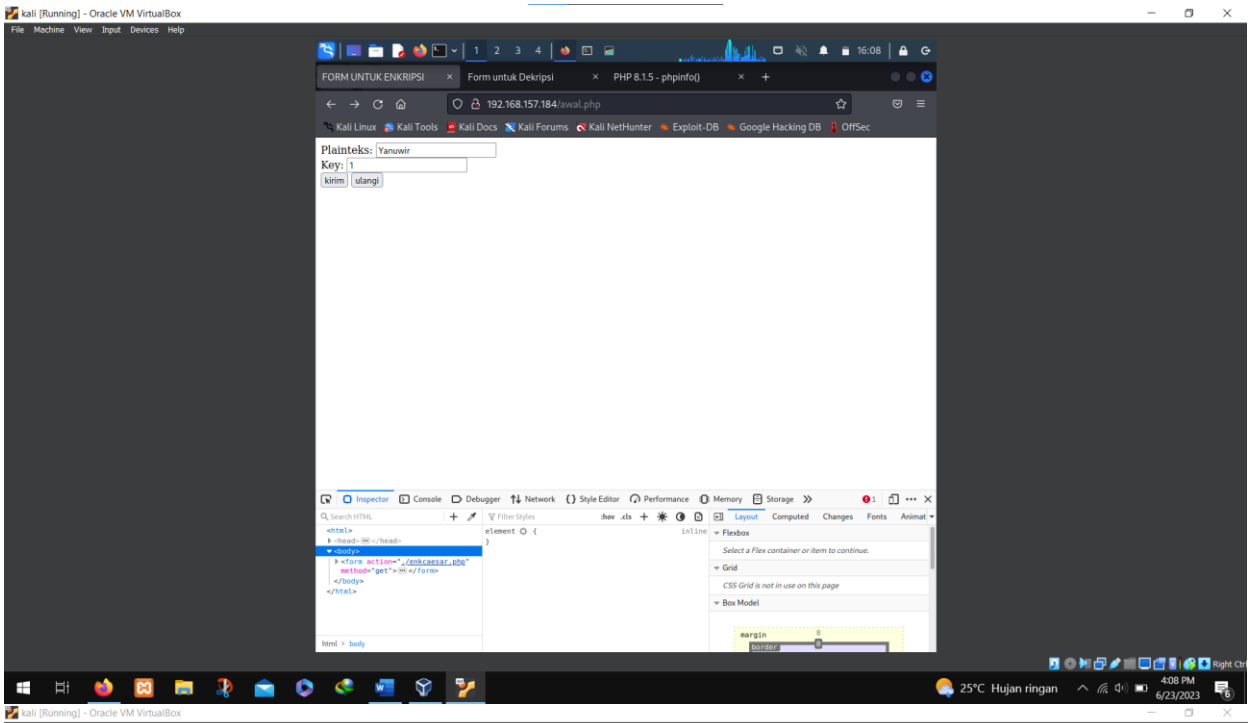
Praktikum Keamanan Jaringan (Symmetric Cryptography)
Tanggal Praktikum : 23 Juni 2023
Kelas : TI-2A
Kelompok : Kelompok Potato Desktop
Nama Anggota :
1. Ilham Yanuar Putra
2. Muhammad Wahyu Anggoro
3. Rizky Dwi Purnama
4. Sufyan Hanif Ariyana

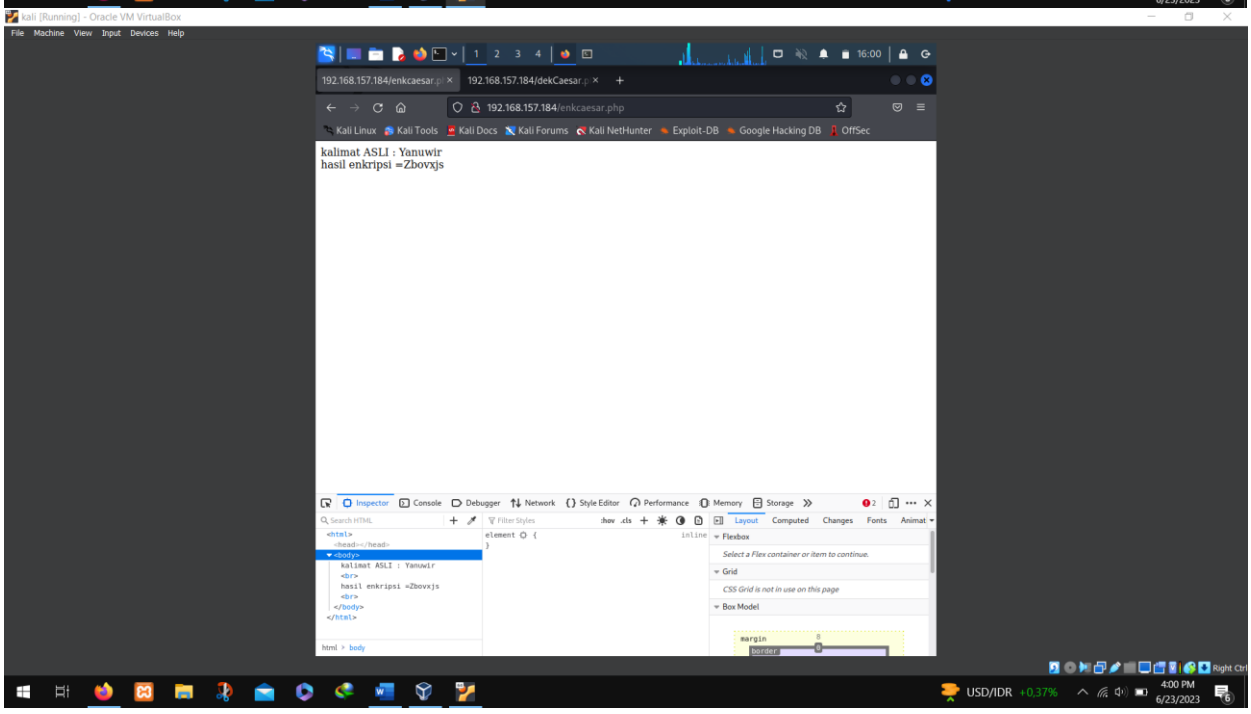
A. Gambar topologi jaringan beserta dengan IP Addressnya, tunjukkan mana PC Server dan Client



B. Catat hasil pada poin 2.d, apakah berhasil menampilkan info tentang PHP

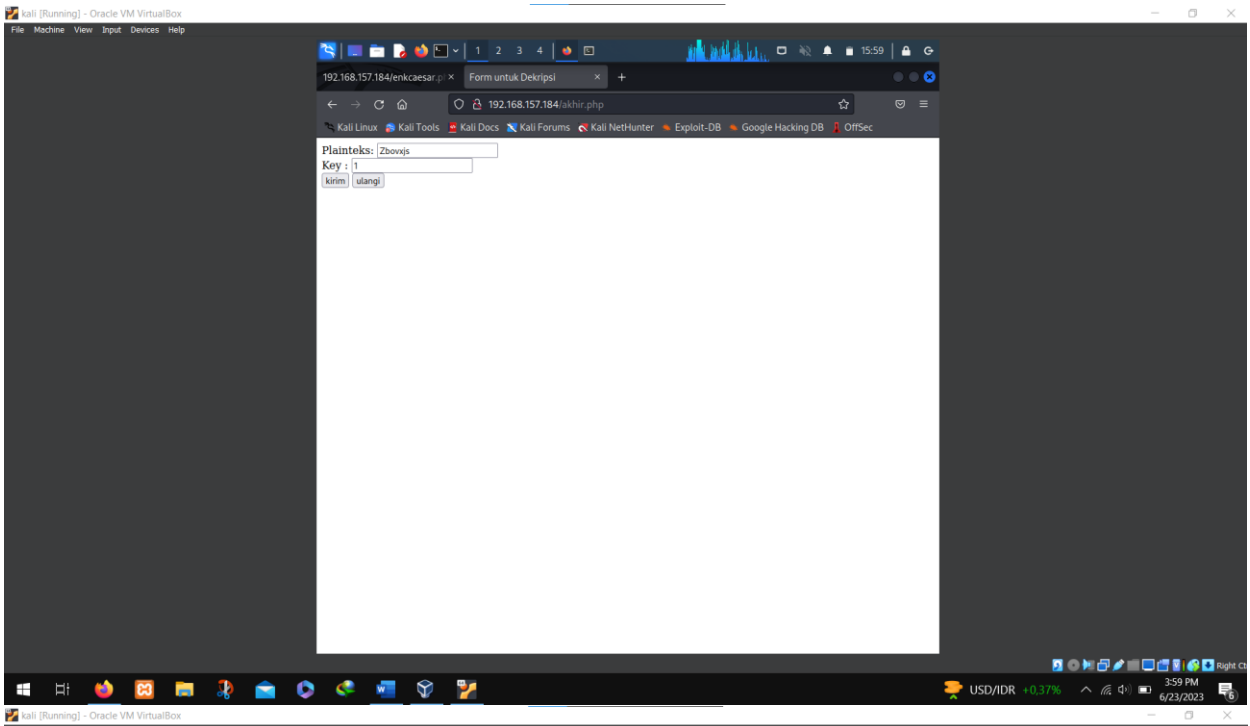
C. Proses enkripsi dengan Caesar algorithm
a. Gambarkan tampilan pada poin 3.a & b (enkripsi)

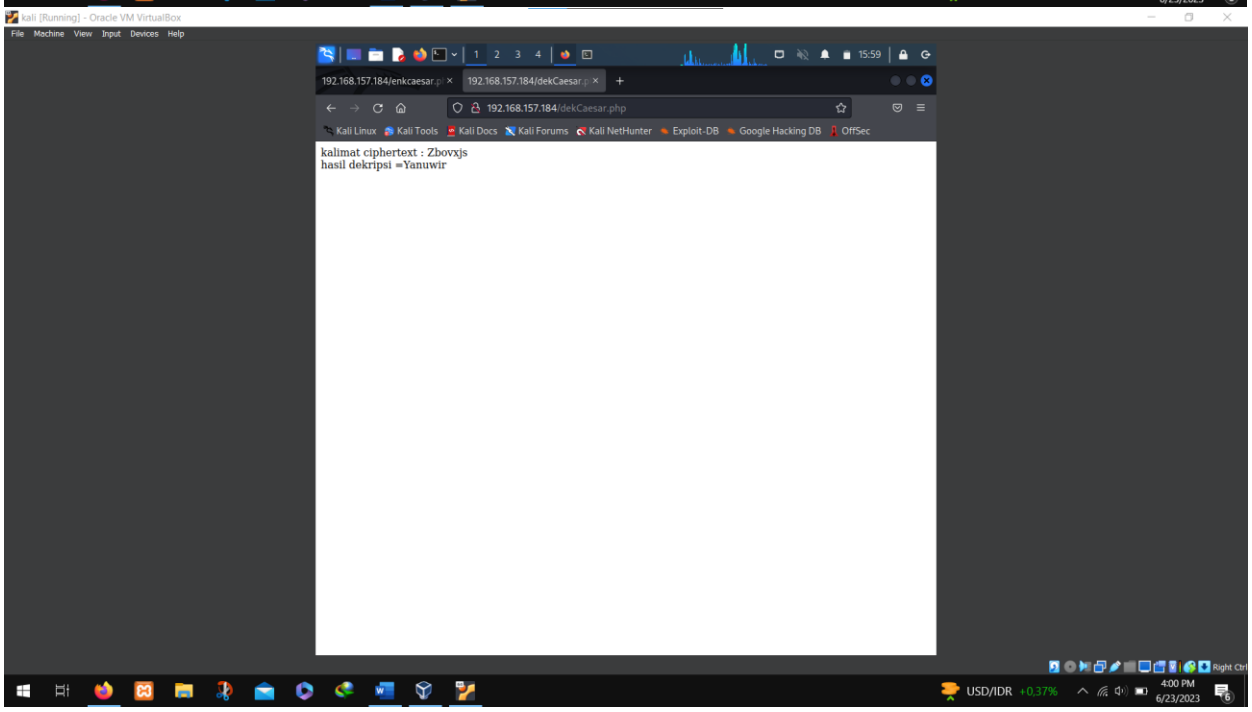




b. Catat hasil proses enkripsi

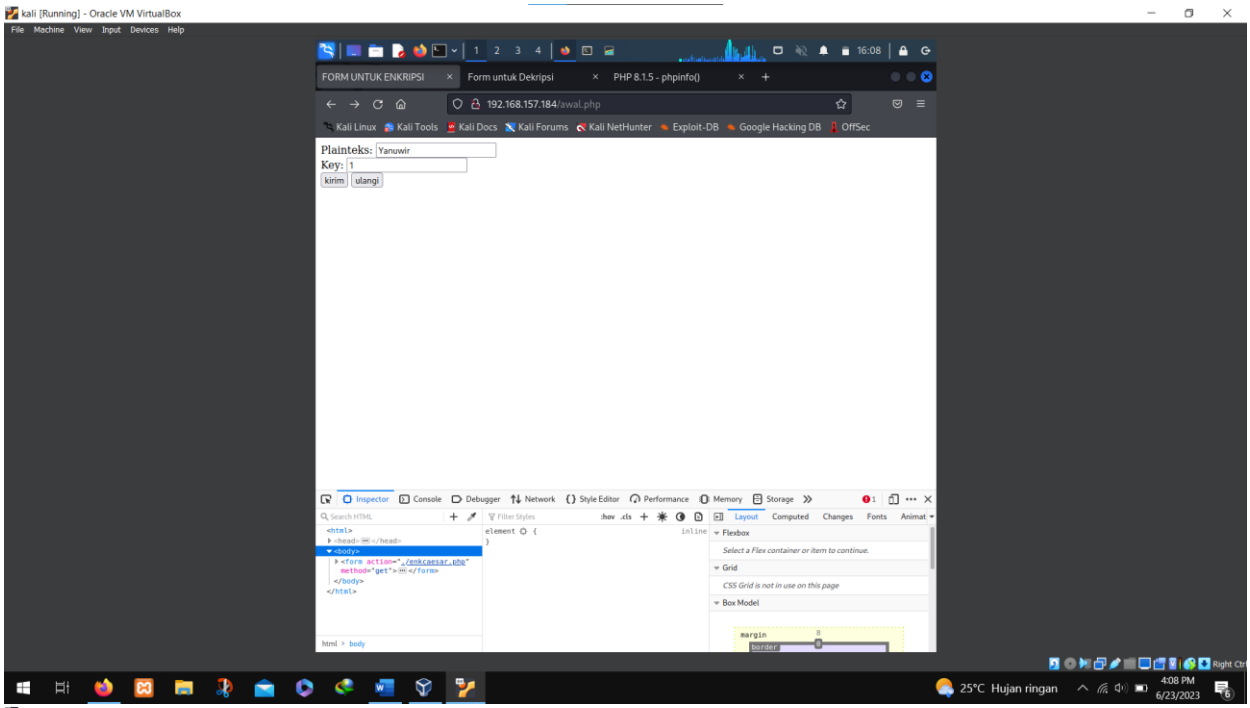
c. Gambarkan tampilan pada poin 3.e & f (dekripsi)

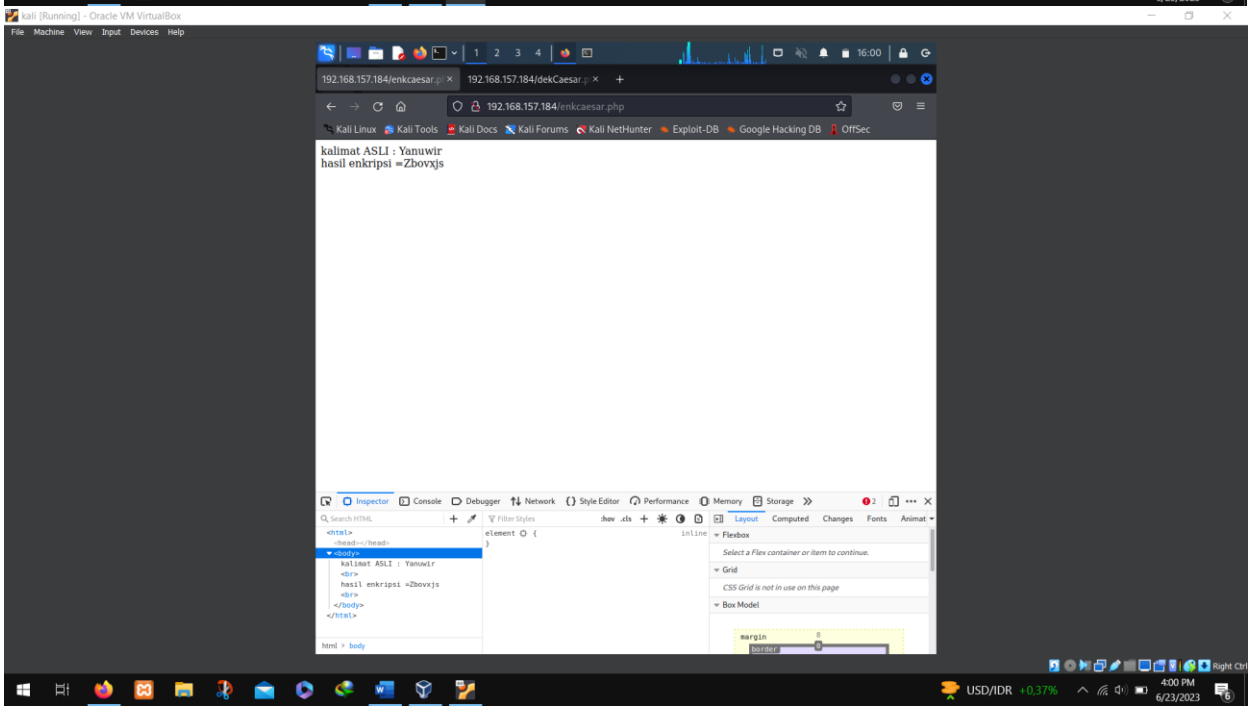




d. Catat hasil proses dekripsi
e. Catat dan analisa pada poin 3.j

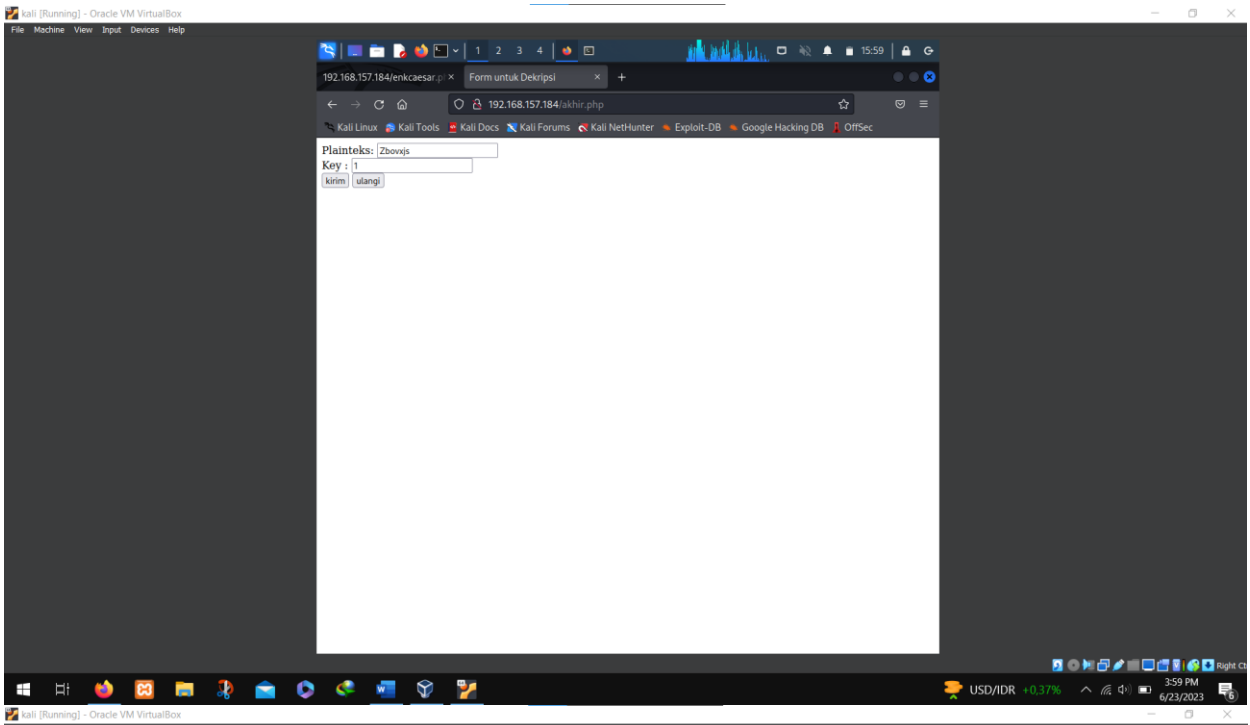
D. Proses enkripsi dengan Caesar algorithm
a. Gambarkan tampilan pada poin 3.a-e (enkripsi)

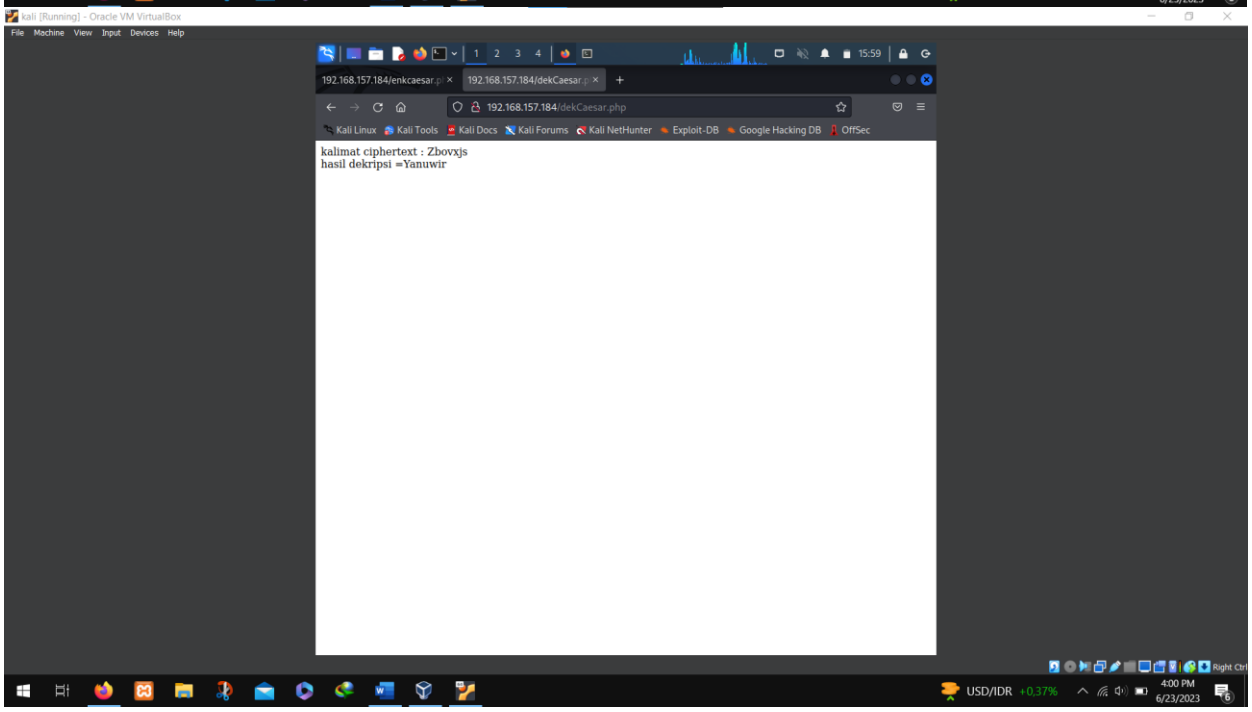




b. Catat hasil proses enkripsi

c. Gambarkan tampilan pada poin 3.h-l (dekripsi)





d. Catat hasil proses dekripsi

e. Catat dan analisa pada poin 3.j

- $kcd akan menerima nilai dari parameter GET dengan nama "kcdekripsi". Variabel $kcd kemudian digunakan untuk mencetak nilai kunci dekripsi tersebut. Setelah itu, dilakukan iterasi sepanjang panjang kunci dekripsi (strlen($kcd)).
- Pada setiap iterasi, nilai ASCII dari karakter kunci dekripsi (ord($kcd[$i])) diubah menjadi bilangan desimal dan disimpan dalam array $key pada indeks yang sesuai.
- Dua kali inisialisasi array $mm terlihat dalam potongan kode ini, yang menunjukkan bahwa kode ini mungkin hanya sebagian dari keseluruhan fungsi.
- Pada inisialisasi pertama, array $mm diisi dengan bilangan 0 hingga 255 secara berurutan.

- Namun, inisialisasi kedua tampaknya melakukan hal yang sama seperti inisialisasi pertama, sehingga potongan kode yang diberikan mungkin tidak sepenuhnya benar.

- Setelah inisialisasi array $mm, dilakukan pengacakan dengan menggunakan algoritma Fisher-Yates Shuffle.
- Pada setiap iterasi, nilai indeks $i dan $j ditukar ($a = $mm[$i], $mm[$i] = $mm[$j], $mm[$j] = $a) berdasarkan perhitungan yang melibatkan variabel $j, $k, dan elemen-elemen array $mm.
- Variabel $j digunakan sebagai indeks acak dalam rentang 0 hingga 255 dengan menggunakan operasi modulus.
- Variabel $k bertanggung jawab untuk mengontrol perulangan 16 iterasi, dimulai dari 0 hingga 15.