Volume III 2008

ISSN: 1907-3526

SRITI

Seminar Nasional Riset Teknologi Informasi

Proceeding

Seminar Nasional Riset Teknologi Informasi 2008

"Membangun Sinergi Riset Perguruan Tinggi dengan Industri Melalui Konvergensi Digital"

Yogyakarta, 09 Agustus 2008

Komputasi
Kecerdasan Buatan
Teknologi Basis Data
Pemodelan dan Aplikasi Sistem Informasi
Sistem Kendali dan Robotika
Signal Processing
Komunikasi Data dan Jaringan Komputer
Games
Pengolahan Citra
Grafika dan Multimedia

Diselenggarakan oleh:

YAYASAN PENDIDIKAN WIDYA BAKTI
STMIK
AKAKOM
YOGYAKARTA
Terakreditasi A*(sangat baik)

DAFTAR ISI

PENGEMBANGAN ALGORITMA MIME BASE64 *ENCODING* SEBAGAI METODE PENYEMBUNYIAN *SOURCE CODE* PHP PADA WEB SERVER

Dwi Retnoningsih
Program Studi Teknik Informatika, STMIK El Rahma
Jl. Sisingamangaraja No76, Telp/Fax (0274) 377982, Yogyakarta
Email : dw1retno@mti.ugm.ac.id

ABSTRAK

Maraknya cybercrime, *membuat para user baik perorangan maupun bidang bisnis mulai menyadari pentingnya melindungi data atau informasi yang berharga. Salah satu cara untuk mengamankan data adalah dengan meng-*encode *data tersebut.* Encode *merupakan proses transformasi dari data asli* (plaintext) *ke dalam bentuk kode-kode yang tersandikan* (ciphertext). *Sedangkan proses transformasi dari* ciphertext *kembali ke bentuk* plaintext *disebut dengan* decode.

Bahasa pemrograman PHP menyediakan fasilitas encoding *dan* decoding *berbasis 64 bit secara* default. *Base64 atau* quadrosexagesimal *adalah penempatan notasi dengan menggunakan bilangan* radix *64. Base64 merupakan bilangan berbasis 2 (*power-of-two*) terbesar yang dapat direpresentasikan dengan menggunakan karakter ASCII.*

Penelitian ini difokuskan pada pengembangan sebuah metode baru berdasarkan algoritma MIME Base64 encoding *dengan merancang sebuah fungsi yang dapat digunakan untuk penyembuyian* source code php *pada web server. Penelitian ini menghasilkan sebuah fungsi yang diberi nama FED (Fungsi* Encode Decode*), mempunyai kemampuan diantaranya dapat digunakan untuk meng*encode *dan me*decode source code php *dengan cara merubah karakter ASCII menjadi karakter Base64. FED ditujukan untuk membantu para administrator dalam upaya mengamankan* source code php *yang bersifat* open source *pada web server.*

Kata-kata kunci : *encode, decode, source code, web server.*

1. PENDAHULUAN

Bahasa pemrograman PHP merupakan bahasa pemrograman berbasis web yang saat ini sangat populer. Hal ini dapat dilihat dengan banyaknya aplikasi berbasis web yang menggunakan PHP. PHP memang memiliki banyak kelebihan dibandingkan dengan bahasa pemrograman web sejenis, diantaranya adalah kecepatan dalam akses dan kemudahan dalam menggunakannya. Selain itu PHP juga memiliki kekurangan, salah satunya adalah mudah membaca *source code* PHP karena *source codenya* yang berbentuk *plaintext*.

Base64 atau *quadrosexagesimal* adalah penempatan notasi dengan menggunakan bilangan *radix* 64. Sehingga relatif lebih cepat waktu yang dibutuhkan untuk proses *encode* dan *decode* dibandingkan dengan algoritma lain yang berbasis lebih besar dari 64 seperti Triple DES (128 bit), Ripemd160 (160 bit), dan lain-lain. Base64 merupakan bilangan berbasis 2 (*power-of-two*) terbesar yang dapat direpresentasikan dengan menggunakan karakter ASCII. Hal ini memungkinkan untuk penggunaannya melakukan *encoding* terhadap *email, source code* PHP dan lainnya yang memiliki format *plaintext*.

TEORI, MODEL, DAN DESAIN

1. TEORI

MIME Base64 *Encoding*

Base64 atau *quadrosexagesimal* adalah penempatan notasi dengan menggunakan bilangan radix 64. Base64 merupakan bilangan berbasis 2 (*power-of-two*) terbesar yang dapat direpresentasikan dengan menggunakan karakter ASCII. Hal ini memungkinkan untuk penggunaannya melakukan *encoding* terhadap email dan lainnya. Base64 menggunakan karakter A–Z, a–z dan 0–9 untuk 62 nilai pertama, sedangkan 2 nilai terakhir digunakan untuk symbol (+ dan /).

Tabel 1 Tabel Base64 *Encoding*

Base64 Encoding Table							
Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	O	32	G	48	w
1	B	17	R	33	H	49	x
2	C	18	S	34	I	50	y
3	D	19	T	35	J	51	z
4	E	20	U	36	K	52	0
5	F	21	V	37	L	53	1
6	G	22	W	38	M	54	2
7	H	23	X	39	N	55	3
8	I	24	Y	40	O	56	4
9	J	25	Z	41	P	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Beberapa metode *encoding* lain seperti *uuencode* dan *binhex* menggunakan 64 karakter yang berbeda untuk mewakili 6 binary digit, namun metode-metode tersebut tidak disebut sebagai *encoding Base64*.

Penggunaan MIME Base64

Berikut ini adalah contoh penggunaan dari MIME Base64 dalam melakukan encoding karakter. Kutipan dari Thomas Hobbes's Leviathan:

> *"Man! is distinguished, not only by his reason, but by this singular passion from other animals,which is a lust of the mind, that by a perseverance of delight in the continued andindefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure". Hasil encoding kata "Man" diganti menjadi "TWFu". Pada table ASCII (lampiran 1) huruf M, a, n disimpan sebagai 77, 97, 110 atau dengan kata lain 010001101, 01100001, 01101110 pada bilangan berbasis 2.*

```
"TWFuIGlzIGRpc3RpbmdlaXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1d
CBieSB0aGlz
IHNpbmdlbGFyIHBhc3Npb24gZnJvbSBvdGhlciEhbmltYWxzLCB3aGljaCBpcyBhIG
xlc3Qgb2Yg
dGhlIGlpbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aG
UgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleG
N1ZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4="
```

Apabila ketiga *byte* tersebut digabungkan, maka akan dihasilkan 24 *bit buffer* yaitu 010001101011000010110111O. Angka tersebut harus dikonversi sehingga berbasis 64, caranya dengan membagi 24 bit tersebut dengan 6. Maka dihasilkan 4 bagian dengan masing-masing 6 bit. Kemudian masing-masing bagian tersebut dikonversi ke nilai yang ada di Base64.

Tabel 2 Hasil konversi dari ASCII ke Base64 untuk kata "Man"

Huruf	M								a								n							
ASCII	77								97								110							
Bit	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19					22					5					46								
Base64 Encode	T					W					F					u								

Proses *padding* akan dilakukan apabila sekelompok karakter yang dimiliki tidak bernilai 3 Byte (24 bit). *Padding* dilakukan dengan menambahkan karakter '='. Contoh penggunaan *padding* dapat dilihat pada tabel berikut.

Tabel 3 Proses *padding* 2 Byte nilai 0

Huruf	I																							
ASCII	33																							
Bit	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index	8								16															
Base64 Encode	I								Q								=							

Apabila terdapat *single byte* maka jumlah *padding* yang ditambahkan adalah 2 yang bernilai 0. Sehingga memenuhi aturan 3 Byte (24 bit), seperti dapat dilihat pada tabel 3. Sedangkan pada tabel 4 jumlah byte *padding* yang ditambahkan adalah 1 Byte karena sebelumnya telah memiliki 2 Byte.

Tabel 4 Proses *padding* 1 Byte nilai 0

Huruf	i								n															
ASCII	105								115															
Bit	0	1	1	0	1	0	0	1	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0
Index	26								23								12							
Base64 Encode	a								X								M							

2. MODEL

Perancangan FED dilakukan dengan beberapa tahapan pengembangan sistem berikut:

1. Analisis kebutuhan sistem.

a) Penentuan masalah dan peluang yang dituju sistem

Adanya beberapa kelemahan dari *script* PHP, yang menyebab kerawanan terhadap serangan para *attacker* terhadap *source code* PHP pada *web server* diantaranya;

- PHP yang bersifat *open source* sehingga mudah diakses oleh semua user
- Mudah membaca *source code* PHP karena berbentuk *plaintext*
- PHP yang ternyata memiliki lubang-lubang keamanan berbahaya seperpti penggunaan variabel *auto* global, fungsi include(), require() atau fopen, penggunaan karakter-karakter *escape* dalam perintah SQL, *upload* yang salah, penggunaan karakter-karakter *escape* html dalam teks, penempatan isi yang

sensitif di luar direktori *root* dokumen, penggunaan *server* bersama, penggunaan kondisi dengan jenis variabel yang tidak jelas, *session spoofing* Sehingga dapat menyebabkan kerawanan terhadap serangan para *attacker*, yang memanfaatkan *vulnerabillity* pada *script* PHP. Maka penulis berfikir untuk mencoba melakukan pengamanan dengan satu langkah yaitu dengan menyandikan *source code* PHP.

Sedangkan peluang yang dituju sistem adalah merubah bentuk asli *source code* PHP yang berupa *plaintext* menjadi kode-kode yang tersandikan (*ciphertext*) dengan cara meng*encode souce code* PHP tersebut berdasarkan algoritma MIME base64.

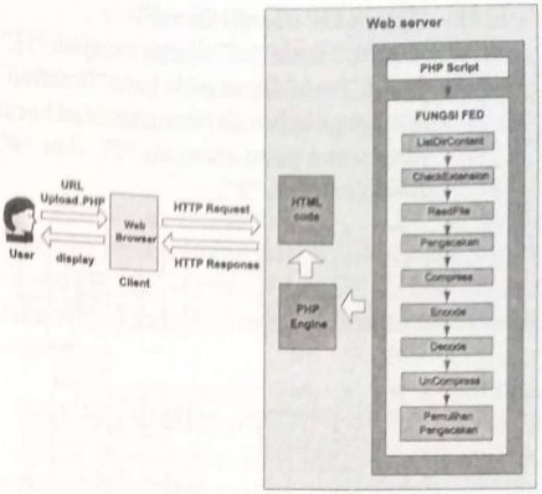
b) Penentuan sasaran sistem baru secara keseluruhan
Peningkatan sistem keamanan *source code* PHP pada *web server*, dengan cara meng*encode souce code* PHP tersebut berdasarkan algoritma MIME base64.

c) Pengidentifikasian para pemakai sistem
Fungsi yang dibangun terutama ditujukan pada para pengelola *web server* untuk mengamankan *source code* PHP, karena saat ini *web* merupakan salah satu layanan informasi yang banyak diakses oleh pengguna internet di dunia.

d) Pembentukan lingkup sistem
Tidak menolak kemungkinan bahwa user memanfaatkan akses *web server* untuk kepentingan pribadi yang mungkin dapat menimbulkan penyerangan. Masalah keamanan merupakan salah satu aspek yang penting, karena kelalaian dalam menangani keamanan *web server* dapat berakibat fatal.

2. Perancangan Sistem

1) Skema mekanisme kerja dari fungsi FED.
   1. *ListDirContent* merupakan fungsi yang digunakan untuk menampilkan lokasi sumber file PHP disimpan.
   2. *CheckExtension* adalah fungsi yang digunakan untuk menyeleksi file-file yang berextensi .PHP.



Gambar 1 Skema detail mekanisme kerja dari fungsi FED

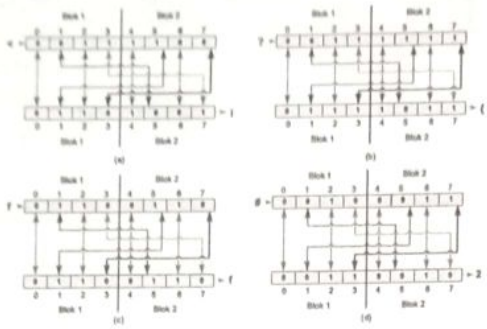
3. Selanjutnya pembacaan file dilakukan pada fungsi *ReadFile*
4. *Compress* yaitu proses untuk memampatkan file yang berukuran besar agar menjadi lebih kecil sehingga prose *encode* menjadi relatif lebih cepat
5. *Encode* adalah proses penyandian *source code* PHP *plaintext* ke dalam bentuk kode-kode yang disandikan (*ciphertext*).
6. *Decode*, yaitu proses pengembalian dari bentuk *ciphertext* ke bentuk aslinya (*plaintext*) kembali.
7. *Uncompress* adalah proses e*xtract* file yang decompress
8. Pemulihan pengacakan, merupakan proses pengembalian file yang telah dilakukan pengacakan pada langkah 4.

2) Metode Pengacakan
Pada fungsi FED akan menerapkan sistem pengacakan dengan metode yang beragam. Pada implementasinya dapat dipilih salah satu dan yang lain dalam status *off*. Contoh sebagian *source code* PHP yang akan di*encode*:

```
<?
function Encode($code){
## proses encode
$code=Compress($code);
$code=base64_encode($code);
## penambahan code untuk decode
$code="$samaran<?
eval(gzinflate(base64_decode('$code'
))) ?>$samaran";
return ($code);
}
function Compress($code){
$code=gzdeflate($code,9);
return ($code);
}
?>
```

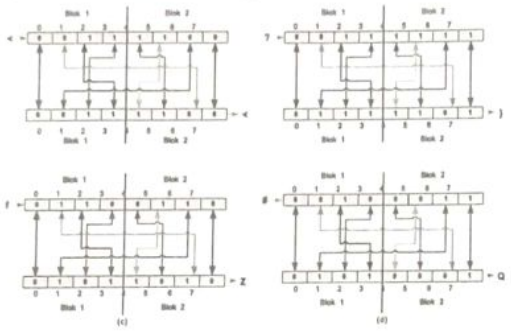
Gambar 3 **Contoh *Source Code* PHP**

a) Metode GP (Ganjil GenaP)
Misalnya kode "<?" diacak menjadi "i{", dan "f" huruf depan pada kata "function" jika diacak kebetulan menghasilkan huruf yang sama yaitu menjadi "f", dan "#" diacak menjadi "2".



Gambar 4 Rancangan pengacakan menggunakan metode GP

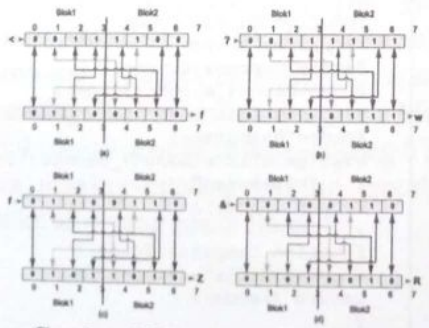
b) Metode S3 (Silang3)



Gambar 5 Rancangan pengacakan menggunakan metode S3

Contoh implementasi pengacakan bilangan biner menggunakan metode S3. Kode "<?" diacak menjadi "<}" dan huruf "f" pada kata "function" menjadi "Z", sedangkan "#" diacak menjadi "Q".

c) Metode LM (Lurus Miring)
Contoh implementasi pengacakan bilangan biner menggunakan metode LM. Kode "<?" diacak menjadi "fw" dan huruf "f" pada kata "function" menjadi "Z", dan simbol "&" diacak menjadi "R".



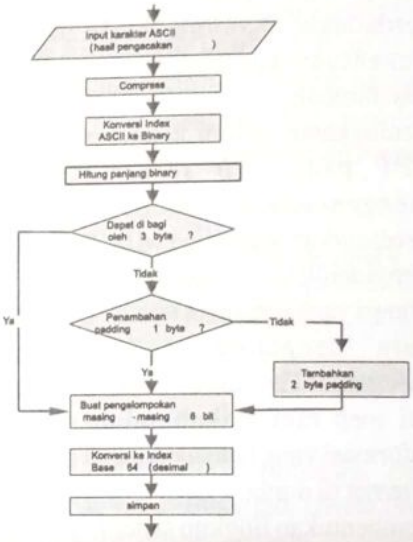
Gambar 6 Rancangan pengacakan menggunakan metode LM

Setelah dilakukan pengacakan langkah selanjutnya adalah proses *encode* berdasarkan MIME Base64 *encoding* dengan alur logika pada gambar 7.

3) Proses *Encode* Berdasarkan MIME Base64 *Encoding*

Proses *encoding*, diawali dengan adanya input file dari hasil pengacakan. Berikut ini adalah langkah-langkah implementasi dari *encode* berdasarkan pada MIME Base64 dalam melakukan *encoding* pada sebagian *plaintext* PHP.

Misalnya kode "<?f" jika diencode akan menjadi "PD9M", dengan langkah-langkah berikut: pada tabel ASCII kode <,?,f disimpan sebagai 60,63,102 atau dengan kata lain 00111100, 00111111, 01100110 pada bilangan berbasis 2.



Gambar 7 *Encode* Berdasarkan MIME Base64

Apabila ketiga *byte* tersebut digabungkan, maka akan dihasilkan 24 *bit buffer* yaitu 001111000011111101100110. Angka tersebut harus dikonversi sehingga berbasis 64, caranya dengan membagi 24 bit tersebut dengan 6. Maka dihasilkan 4 bagian dengan masing-masing 6 bit. Kemudian masing-masing bagian tersebut dikonversi ke nilai yang ada di Base64, dan seterusnya hingga *plaintext* akan ter*encode* semua dari awal hingga selesai seluruhnya.

Tabel 4 Hasil konversi dari ASCII ke Base64 untuk kode "<?f"

Huruf	<								?								f							
ASCII	60								63								102							
Bit	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	1	1	0
Index	15								3				61				38							
Base64 Encode	P								D				9				M							

Tabel 5 **Hasil konversi dari ASCII ke Base64 untuk kode "($"**

Huruf	(		$	
ASCII	40		36	
Bit	0 0 1 0 1 0 0 0		0 0 1 0 0 1 0 0	
Index	10	2	16	0
Base64 Encode	K	C	Q	=

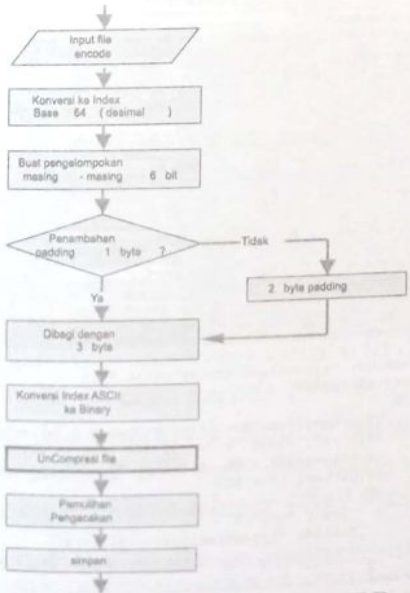
Pada tabel 6 adalah proses *padding* 1 byte bernilai 0, karena kelompok karakter tidak bernilai 3 byte (24 bit). *Padding* dilakukan dengan manambahkan karakter "="

Tabel 6 **Hasil konversi dari ASCII ke Base64 untuk kode "{"**

Huruf	{			
ASCII	123			
Bit	0 1 1 1 1 0 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	
Index	30	48		
Base64 Encode	e	w	=	=

Pada tabel 5 terdapat *single-byte* maka jumlah *padding* yang ditambahkan adalah 2 byte yang bernilai 0. Sehingga memenuhi aturan 3 byte (24 bit).

4) *Decode* Berdasarkan MIME Base64 *Encoding*
   1. File yang telah diencode sebagai inputnya
   2. Proses konversi ke index base64 (desimal)
   3. Pengelompokan sejumlah 6 bit
   4. Selanjutnya akan ditanyakan apakah dalam pengelompokan tersebut merupakan penambahan *padding* 1? Jika ya, maka bit dapat langsung di bagi menjadi 3 byte, jika tidak, berarti bit tersebut dilakukan penambahan 2 *padding* pada proses *encode*.



Gambar 8 *Decode* Berdasarkan MIME Base64

5. Untuk selanjutnya bit tersebut dilakukan pembagian sejumlah 3 *byte*
6. Proses dilanjutkan dengan mengkonversi *index* ASCII ke bilangan *biner*
7. Proses *encode* selesai.

5) Pemulihan Pengacakan

File yang sudah di *decode*, selanjutnya dilakukan proses pemulihan pengacakan, dengan cara kebalikan dari metode pengacakan yang digunakan.

- Pada metode GP, urutan bit pada saat pengacakan adalah 0,1,2,3,4,5,6,7 nilai kode binernya digeser pada posisi 0,5,2,7,4,1,6,3. Urutan pergeseran untuk pemulihan pengacakannya sama yaitu nilai bilangan biner dari posisi 0,5,2,7,4,1,6,3 akan menempati posisi 0,1,2,3,4,5,6,7.
- Pada metode S3, urutan bit pada saat pengacakan adalah 0,1,2,3,4,5,6,7 nilai kode binernya digeser pada posisi 0,6,3,2,5,4,1,7. Urutan pergeseran untuk pemulihan pengacakannya sama yaitu nilai bilangan biner dari posisi 0,6,3,2,5,4,1,7 akan menempati posisi 0,1,2,3,4,5,6,7.
- Pada metode LM, urutan bit pada saat pengacakan adalah 0,1,2,3,4,5,6,7 nilai kode binernya digeser pada posisi 0,4,6,2,5,1,3,7. Urutan pergeseran untuk pemulihan pengacakannya yaitu nilai bilangan biner dari posisi 0,5,3,6,1,4,2,7 akan menempati posisi 0,1,2,3,4,5,6,7.

3. HASIL PENELITIAN

FED terdiri dari kumpulan banyak fungsi antara lain:

Tabel 7 *Function.FileSystem.inc.php*

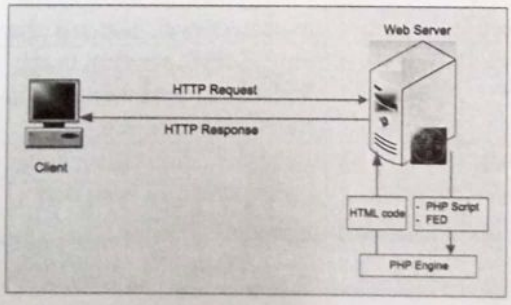
Fungsi	Keterangan
Function ListDirContent	Fungsi ini digunakan untuk melihat *directory* dimana file php yang akan di *encode* disimpan.
Fungsi CheckExtension	Fungsi ini digunakan untuk melakukan cek ekstensi php.
Fungsi ReadFile	Fungsi ini digunakan untuk melakukan pembacaan *file-file* berekstensi php.
Fungsi PutEncodeFile	Fungsi ini digunakan untuk meletakkan file hasil dari proses encode.

Tabel 8 *Function.encode.inc*

Fungsi	Keterangan
Fungsi pengacakan: a. *Function*.gp b. *Function*.s3 c. *Function*.lm	Merupakan fungsi yang digunakan untuk pengacakan kode php menggunakan metode ganjil-genap Merupakan fungsi yang digunakan untuk pengacakan kode php menggunakan metode silang-tiga Merupakan fungsi yang digunakan untuk pengacakan kode php menggunakan metode lurus-miring
Function.compress	Fungsi yang digunakan untuk proses meng - *compress* file
Function.encode	Fungsi yang digunakan untuk proses *encode* yaitu proses merubah dari *plaintext* menjadi *ciphertext*.
Function.decode	Fungsi yang digunakan untuk proses *decode* yaitu proses merubah dari *ciphertext* menjadi *plaintext*.
Function uncompress	Fungsi yang digunakan untuk proses membuka kembali file yang telah di - compress
Fungsi pemulihan pengacakan	Fungsi yang dapat digunakan untuk mengembalikan *file-file* yang telah diacak ke dalambentuk aslinya.

Konsep Kerja Fungsi FED

FED merupakan sebuah fungsi yang dirancang menggunakan PHP. Gambar berikut merupakan ilustrasi posisi FED dalam *web server*. Secara umum algoritma dari fungsi FED terdiri dari 6 langkah, yaitu pengacakan, *compressi file, encode, decode, uncompress*, pemulihan pengacakan.



Gambar 11 Posisi fungsi FED dalam web server

Selanjutnya untuk melakukan proses *encode* dan *decode*, FED akan melakukan pemanggilan terhadap fungsi-fungsi yang lain yaitu *function.filesystem.inc.php* dan *function. encode.inc.php*, dan *file-file* pendukung lainnya.

File yang akan ditampilkan di halaman web, adalah *file* yang telah di-*decode* dari *file* yang ada di *folder encode*. Proses *decoding*, dilakukan oleh program di *file index.php* yang akan memanggil fungsi *decode*, kemudian hasil *decode* di simpan di *folder* tmp. File yang telah di-*decode* di ambil (menggunakan "include") sehingga dapat ditampilkan di halaman web.

Pengujian Sistem

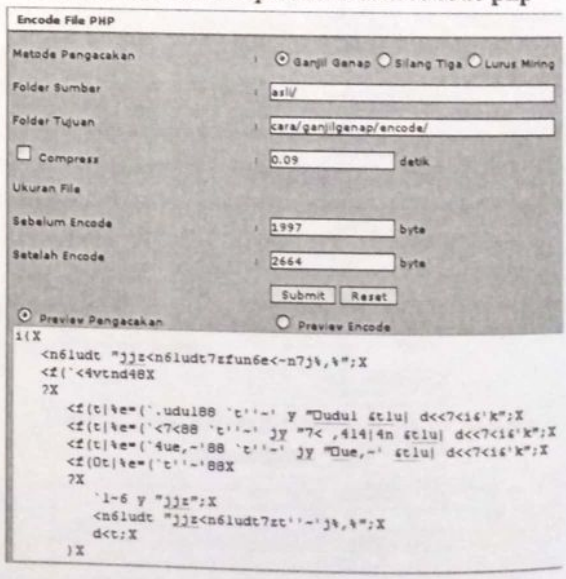
Proses pengujian dilakukan menggunakan spesifikasi komputer processor Intel Celeron M380 (1.60 GHz), RAM 640 MB, Sistem Operasi Windows XP Profesional, Hard disk 40 GB, Resolusi graphic 1024 x 768 (32 bit) (60 Hz).

1. Pengujian Proses *Encode*

Mengambil contoh sebuah file adm_agenda. php dengan ukuran 1.997 byte (2 KB).

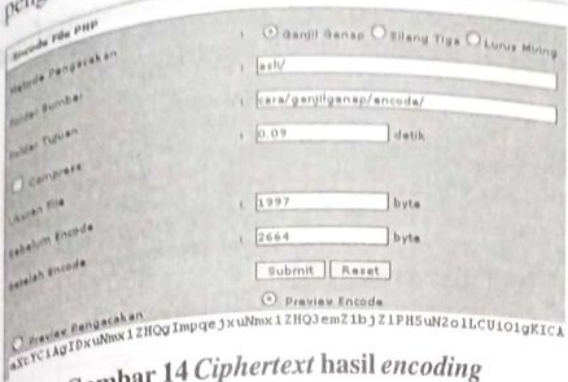
```
<?php
  include "../includes/functions.php";
  if($iagenda)
  {
     if(empty($judul)) $error = "Judul belum diisi<br>";
     if(empty($isi)) $error .= "Isi halaman belum diisi<br>";
     if(empty($author)) $error .= "Author belum diisi<br>";
     if(!empty($error))
     {
        $loc = "../";
        include "../includes/error.php";
        die;
     }
     else
     {
        $query = "insert into tbl_agenda(judul, halaman, author, depan, waktu)
           values('$judul','$isi','$author','$tampil',sysdate())";
        execquery($query);
        header("location: ../adm_index.php?mod=adm_agenda&iamenu=6");
        die;
     }
  }
```

Gambar 12 Contoh *plaintext source code* php



Gambar 13 Pengujian *encoding* menggunakan metode pengacakan GP tanpa *compress*

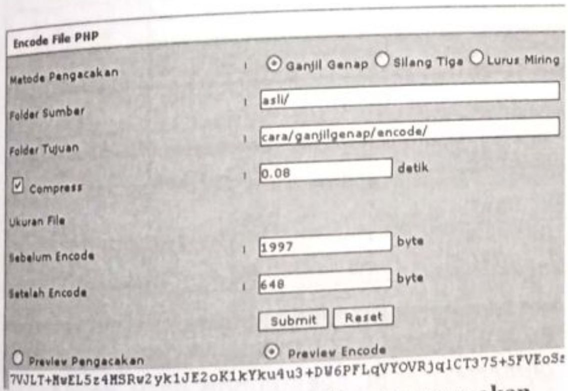
Waktu *encoding* menggunakan metode pengacakan gp tanpa *compress* sebesar 0,09 detik



Gambar 14 *Ciphertext* hasil *encoding* menggunakan pengacakan GP

aXtYCiAgIDxuNmx1ZHQgImpqejxuNmx1ZHQ3emZ1bjZ1PH5uN2olLCUiOlgKICAgPGYoYDw0dnRuZDQ4WAogICA/WAogICAgICA8Zih0fCV1PShgLnVkdWw40CBgdCcnfic
ge5AiDnVkdWwgJnRsdXwgZDw8NzxpJidrIjtYCiAgICAgIDxmKHR8JWU9KGA8Nzw40CBgdCcnficgankgIhw3PCAsNGw0fDRuICZ0bHV8IGQ8PDc8aSYnayI7WAogICAgIC
A8Zih0fCV1PShgNHV1LH4nODggYHQnJ34nIGp5ICIUdWUsficgJnRsdXwgZDw8NzxpJidrIjtYCiAgICAgIDxmKDB0fCV1PShgdCcnfic40FgKICAgICAgPlgKICAgICAgI
CAgYGx+NiB5ICJqanoiOlgKICAgICAgICAgPG42bHVkdCAiamp6PG42bHVkdDd6dCcnfidqJSwlIjtYCiAgICAgICAgIGQ8dDtYCiAgICAgIH1YCiAgICAgIHRsN3RYCiAg
ICAgID9YCiAgICAgICAgIGA1dXQnPSB5ICI8bjd0J2UgPG51fiB1JmxfNHZ0bmQ0KC51ZHVsaCAsNGw0fDRuaCA0dWUsfidyaHJgZTR8JTxscmg3PTdkNGV0KDg4IjtYCiAgICAgICAgIHQtdDY1dXQnPShgNXV0Jz0401gKICAgICAgICA
gIGc0bHV0NyhyYC51ZHVscmhyYDw3PHJocmA0dWUsfidoIGR0JTRuaCB3ND51dThYCiAgICAgICAgICA
00ZHQnKCJsfjY0ZTx+bisgamp6NGR8XzxuZHQtaiUsJXt8fmR5NGR8XzR2dG5kNGI8fHRudX1jIjg7WAogICAgICAgICBkPHQ7WAogICAgICB9WAogICB9WAogICA8Zihgd
DR2dG5kNDhYCiAgID9YCiAgICAgIDxmKHR8JWU9KGAudWR1bDg4IGB0Jyd+JyB5ICI0dWR1bCAmdGx1fCBkPDw3PGkmJ2siOlgKICAgICAgPGYodHw1ZT0oYDw3PDg4IGB0
Jyd+JyBqe5AiHDc8ICw0bDR8NG4gJnRsdXwgZDw8NzxpJidrIjtYCiAgICAgIDxmKHR8JWU9KGA0dWUsfic40CBgdCcnficgankgIhR1ZSx+JyAmdGx1fCBkPDw3PGkmJ2s
iOlgKICAgICAgPGYoMHR8JWU9KGB0Jyd+Jzg4WAogICAgICA/WAogICAgICAgICBgbH42IHkgImpqeiI7WAogICAgICAgICA8bjZsdWR0ICJqano8bjZsdWR0N3p0Jyd+J2
olLCUiOlgKICAgICAgICAgZDx0OlgKICAgICAgfVgKICAgICAgdGw3dFgKICAgICAgPlgKICAgICAgICAgYDV1dCc9IHkgInU1ZDR1dCB1JmxfNHZ0bmQ0IDd0Z3AudWR1b
CB5IHJgLnVkdWxyaCAsNGw0fDRuIHkgcmA8NzxyaFgKICAgICAgICAgICAgNHV1LH4nIHkgcmA0dWUsfidyaCBkdCU0biB5IHJgZTR8JTxsciB3LHQndCA8ZF80dnRuZDQg
eSByYDxkciI7WAogICAgICAgICB0LXQ2NXV0Jz0oYDV1dCc90DtYCiAgICAgICAgICx0NGR0JygibH42NGU8fm4rIGpqejRkfF88baR0LWolLCV7fH5keTRkfF80dnRuZDR
iPHx0bnV5YyI40lgKICAgICAgICAgZDx0OlgKICAgICAgfVgKICAgfVgKICAgPGYoYDd1ZTR2IH15IGUndXQ4WAogICA/WAogICAgICBgNXV0Jz0geSAiN3RsdDZ1IGR0JT
RuIGYnfnwgZSZsXzR2dG5kNCB3LHQndCA8ZF80dnRuZDQgeSByYDxkciI7WAogICAgICBgJ3Q3dWx1IHkgdC10NjV1dCc9KGA1dXQnPTg7WAogICAgICBsPDd1KGA3ZTR1N
HY4IHkgfDO3NWxfZnR1NixfJ353KGAndDd1bGU40lgKICAgICAgPGYoYDd1NGU0diB5eSAiMSI4IGA1dXQnPTEgeSAidSVkNGV0IGUmbF80dnRuZDQgN3R1IGR0JTRuIHkg
ciFyIHcsdCd0IDxkXzR2dG5kNCB5IHJgPGRyIjtYCiAgICAgIHQtdDY1dXQnPShgNXV0Jz0x0DtYCiAgICAgICx0NGR0JygibH42NGU8fm4rIGpqejRkfF88baR0LWolLCV7fH5keTRkfF80dnRuZD
2dG5kNCB5IHJgPGRyIjtYCiAgICAgIHQtdDY1dXQnPShgNXV0Jz0x0DtYCiAgICAgICx0NGR0JygibH42NGU8fm4rIGpqejRkfF88baR0LWolLCV7fH5keTRkfF80dnRuZD
RiPHx0bnV5YyI40lgKICAgfVgKICAgPGYoYCw0JXU3IH15IGUndXQ4WAogICA/WAogICAgICBgNXV0Jz0geSAiZHRsdGV0IGYnfnwgZSZsXzR2dG5kNCB3LHQndCA8ZF80d
nRuZDQgeSByYDxkciI7WAogICAgICBgJ3Q3dWx1IHkgdC10NjV1dCc9KGA1dXQnPTg7WAogICAgICAsdDRkdCcoImx+NjR1PH5uKyBqano0ZHxfPG5kdC1qJ3wle3x+ZHx0
ZHxfNHZ0bmQ0Yjx8dG51eWMiODtYCiAgIH1YCntrWAo=

Gambar 15 *Ciphertext* hasil *encoding* tanpa *compresi* dengan pengacakan GP



Gambar 16 Pengujian *encoding* menggunakan metode pengacakan GP dengan *compress*

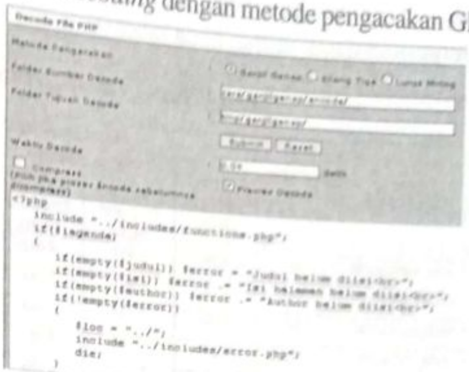
- Waktu yang diperlukan proses *encoding* menggunakan proses *compress* sebesar 0,08 detik. Selisih waktu antara proses *encoding* dengan *compress* dan tanpa proses *compress* sebesar 0,01 detik.
- Ukuran file sebelum *encode* 1997 byte, dan setelah di compress kemudian di*encode* ukuran menjadi 648 *byte*. File mengalami pengurangan ukuran sebesar 1.349 *byte*

7VJLT+MwEL5z4MSRw2yk1JE2oK1kYku4u3+DW6PFLqVYOVRjq1CT375+5FVEoSse4oAPiT3jGc/3uN1eHQGAqAptJEKyWj20e/awMFWhRF0xVZqnyWV8uMhKQe+xkp5HwJ/
wjRa0qZp15baRRnM0JRLSEKgh0fYRaKA2FqQ0TNx0yF3bcafUpYbClas8ZQJyqqml15H11Ki8IbstTkLsxepfbXko4rzL9NjcKnVTeCi0oKF0H30hz4iluKTA/vjYbVAzfP
a9C4Wk518UFU013E0qATXR88h+F1heduQsIUJfgsTUHzf0tzJ81NCtG6oNsmvdFVov155v/4ulfqeoTYVLsRaTVGHG+Q4EPMMiDJbF+fg4m50V5LLEMUW9b36C44RK0xeVx
DWFx9Y2svaRi0GvsFiZ+jrhL9P02FsPv633mdYzqTfByHXAUEEg2aV7H/U0+Viw1K7vIjUh2zkt2j5EWsvBJkeCIGT3VGgml1/Kf0wpeg91DYoYf0LCgTaGGgvVQ1yQxo4Z
fBVa5ZAZrVxiPlotmPLTUDcOdzftjF3o+QJVkc9Js8naHnzstPa6nz6ZJrydd7pf516iHweI480E/9FzekjPJ/inA543azloatFUsFcl1U5TB+dD1HwPMNu7q6N/

Gambar 17 *Ciphertext* hasil proses *compress* dan *encoding* dengan menggunakan pengacakan GP

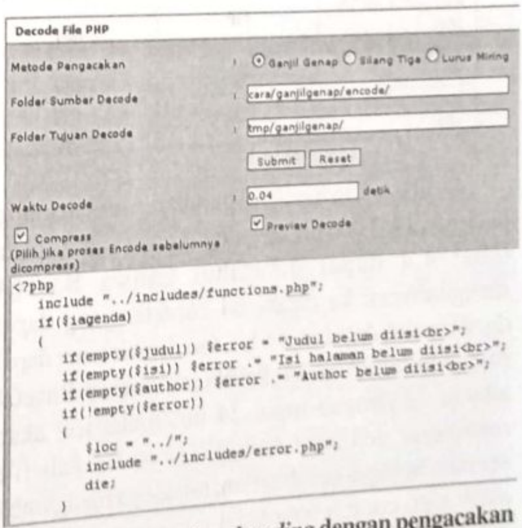
2. **Pengujian Proses Decode.**

   a) *Decoding* dengan metode pengacakan GP



Gambar 18 Pengujian *decoding* dengan pengacakan GP tanpa *compress*

Waktu *decode* tanpa melalui proses *compress* dan tanpa melalui proses *compress* sama sebesar 0,04 detik.



Gambar 19 Pengujian *decoding* dengan pengacakan GP dengan *compress*

Sedangkan perbandingan waktu proses *decoding* berdasarkan ke-3 metode secara terperinci dapat dilihat pada tabel 9.

Tabel 8 Hasil Pengujian Proses *Encoding* Berdasarkan MIME Base64

NAMA FILE	METODE PENG-ACAKAN	TANPA COMPRESSI				DENGAN COMPRESSI				SELISIH WAKTU ENCODE (detik)
		WAKTU ENCODE (detik)	UKURAN FILE		PENAMBAHAN UKURAN FILE (Byte)	WAKTU ENCODE (detik)	UKURAN FILE		PENGURANGAN UKURAN FILE (Byte)	
			SEBELUM ENCODE (Byte)	SETELAH ENCODE (Byte)			SEBELUM ENCODE (Byte)	SETELAH ENCODE (Byte)		
adm_agenda.php	GP	0,09	1.997	2.664	667	0,08	1.997	648	1.349	0,01
adm_agenda.php	S3	0,09	1.997	2.664	667	0,09	1.997	652	1.345	0,00
adm_agenda.php	LM	0,09	1.997	2.664	667	0,09	1.997	660	1.337	0,00
adm_siswa.php	GP	0,23	5.190	6.920	1.730	0,22	5.190	1.396	3.794	0,01
adm_siswa.php	S3	0,23	5.190	6.920	1.730	0,23	5.190	1.396	3.794	0,00
adm_siswa.php	LM	0,23	5.190	6.920	1.730	0,23	5.190	1.400	3.790	0,00
kumpulan 19 file php	GP	1,73	38.969	51.984	13.015	1,72	38.969	12.504	26.465	0,01
kumpulan 19 file php	S3	1,73	38.969	51.984	13.015	1,73	38.969	12.548	26.421	0,00
kumpulan 19 file php	LM	1,73	38.969	51.984	13.015	1,73	38.969	12.632	26.337	0,00

Keterangan : GP : Ganjil genaP S3 : Silang Tiga LM : Lurus Miring

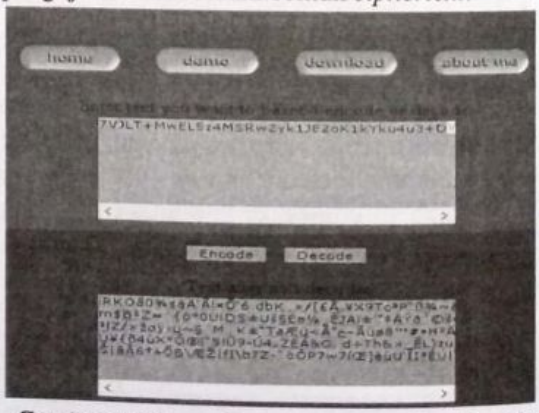
Tabel 9 Hasil Pengujian Proses *Decoding* Berdasarkan MIME Base64

NAMA FILE	METODE PENGACAKAN	UKURAN FILE (Byte)	WAKTU DECODE TANPA COMPRESSI (detik)	WAKTU DECODE DENGAN COMPRESSI (detik)	PERBEDAAN WAKTU (detik)
adm_agenda.php	GP	1.997	0,04	0,04	0,00
adm_agenda.php	S3	1.997	0,04	0,04	0,00
adm_agenda.php	LM	1.997	0,04	0,04	0,00
adm_siswa.php	GP	5.190	0,11	0,11	0,00
adm_siswa.php	S3	5.190	0,11	0,11	0,00
adm_siswa.php	LM	5.190	0,11	0,11	0,00
kumpulan 19 file php	GP	35.889	0,85	0,85	0,00
kumpulan 19 file php	S3	35.889	0,85	0,85	0,00
kumpulan 19 file php	LM	35.889	0,86	0,86	0,00

Berdasarkan hasil pengujian proses *encoding* pada tabel 4.3 dan proses pengujian *decoding* pada tabel 4.4 dapat diketahui bahwa Base64 mengkonversi ke dalam 64 karakter yang dapat dicakup oleh data biner 6 bit, sebab 6 bit biner dapat menampung $2^6 = 64$ karakter. *Output* Base64 adalah 32 dengan input 24 bit, maka file akan membesar dengan faktor 32/24 = 4/3 kali file semula. Sebagai pembuktian, misalnya mengambil salah satu contoh data hasil *encoding* pada tabel 4.3. File adm_siswa.php yang berukuran 5.190 byte, setelah di-*encode* menjadi 6.920 byte dengan perhitungan (4/3) x 5.190 = 6.920 byte. Namun dengan melalui proses kompresi, maka ukuran file dapat diminimalkan.

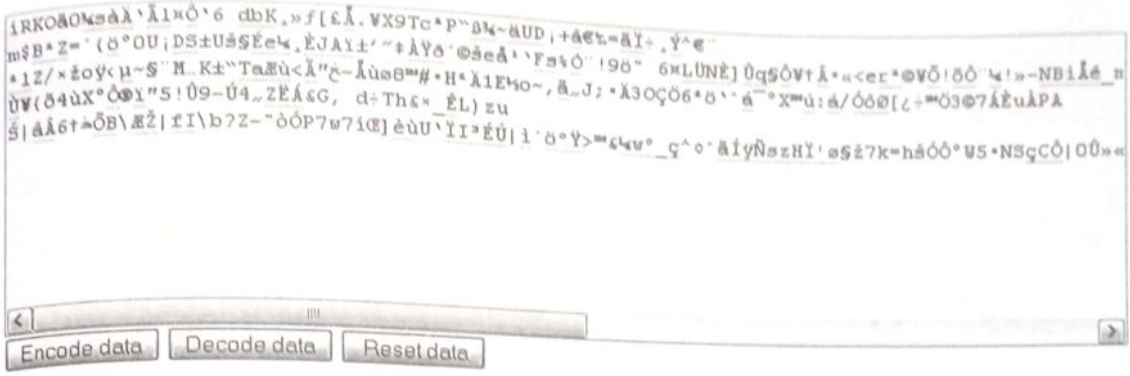
3. Pengujian Validitas

Menggunakan sebuah aplikasi Base64 *Encode and Decoder,* dari http://makcoder.sourceforge.net/demo/base64.php untuk me-*decode* sebuah file *ciphertext* adm_agenda.php hasil *encode* dari FED menggunakan metode pengacakan GP. Hasil pengujian masih dalam bentuk *ciphertext.*



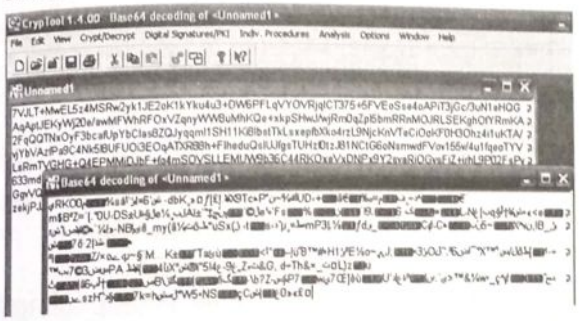
Gambar 20 Proses d*ecode* menggunakan *Base64 encode and decoder*

Menggunakan Aplikasi *online base64 encoder and decoder* dari http://base64-encoder-online.waraxe.us/. Menggunakan *ciphertext* yang sama, jika di *decode* menggunaka aplikasi *online base64 encoder and decoder* hasilnya masih dalam bentuk *ciphertext*.



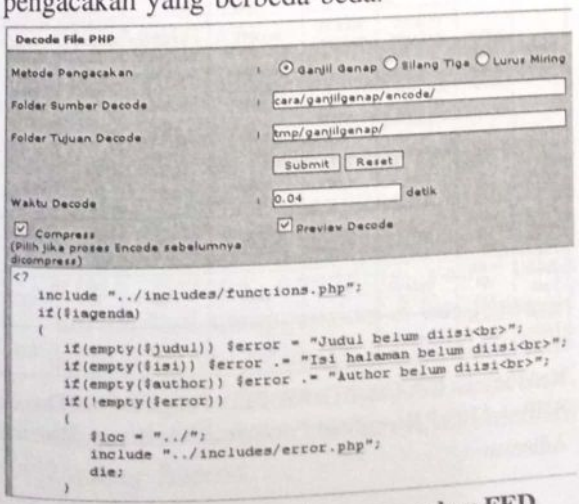
Gambar 21 Hasil *Decode* menggunakan aplikasi *online base64 encoder and decoder*

Menggunakan Aplikasi CrypTool 1.4.00. Menggunakan *ciphertext* yang sama, hasilnya masih dalam bentuk *ciphertext*.



Gambar 22 Proses d*ecode* menggunakan aplikasi CrypTool 1.4.00

Berdasarkan hasil pengujian validitas tersebut, dapat diketahui bahwa *ciphertext* hasil proses *encoding* dari FED sudah tidak standar lagi, karena FED sudah melalui modifikasi algoritma Base64 dengan cara memberikan fitur 3 buah metode pengacakan yang berbeda-beda.



Gambar 23 Hasil *decode* menggunakan FED

Sebagai pembuktian, setelah di *decode* menggunakan FED berdasarkan metode GP, *ciphertext* dapat kembali ke *plaintext* dalam waktu 0,04 detik. Hasilnya dapat dilihat pada gambar 4.35.

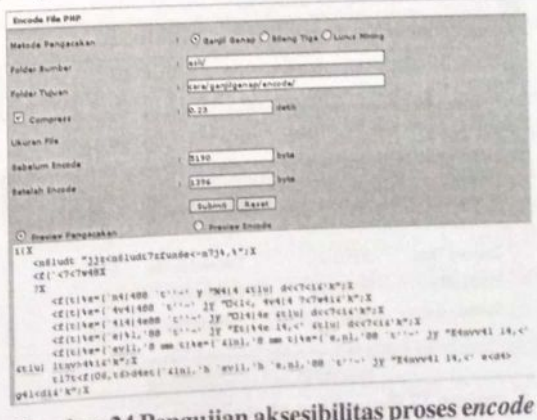
4. Pengujian Aksesibilitas

Pengujian aksesibilitas dimaksudkan untuk mengetahui kecepatan waktu yang digunakan untuk melakukan sebuah proses pada suatu sistem. Pengujian aksesibilitas ini dengan melakukan perbandingan kecepatan akses FED dengan beberapa aplikasi lain yaitu AEP *(Advance Encryption Package)* 2008 Profesioal, aplikasi cryptool berdasarkan algoritma RSA. Pengujian aksesibilitas ini dilakukan dengan spesifikasi sebagai berikut:

- FED berbasis 64 bit, menggunakan metode pengacakan GP 8 bit key, dengan file di *compress*.
- AEP berdasarkan algoritma Rijndael 256 bit key (3 bit key *password*)
- RSA menggunakan key type 512 bit (4 bit key *password*)

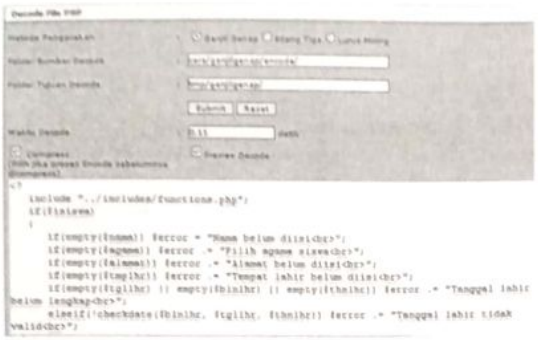
a) Pengujian Aksesibilitas *Encode* dan *Decode* Pada FED

Pada pengujian ini FED menggunakan metode pengacakan GP dengan 8 bit key.



Gambar 24 Pengujian aksesibilitas proses *encode* file *text* pada FED

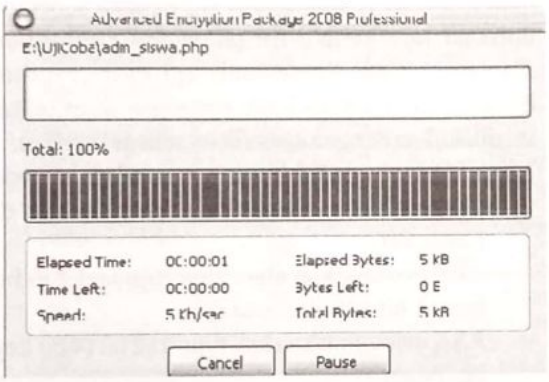
Proses *encode* menggunakan sebuah file *text* adm_siswa.php berukuran 5.190 byte (6 KB). Waktu untuk proses *encode* adalah 0,23 detik dan waktu untuk proses *decode* adalah 0,11 detik.



Gambar 25 Pengujian aksesibilitas proses *decode* file *text* pada FED

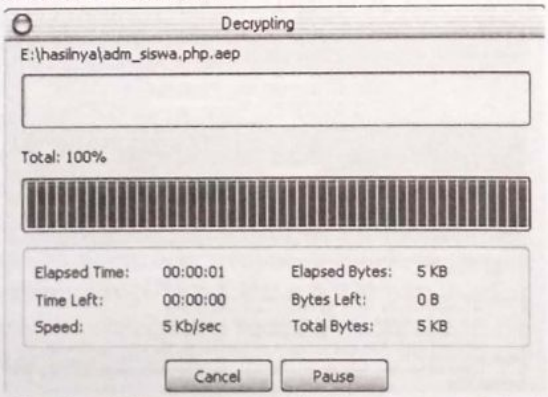
b) Pengujian Aksesibilitas Enkrypsi dan Dekripsi Menggunakan Aplikasi AEP

Proses enkripsi dan dekripsi ini dilakukan dengan menggunakan algoritma Rijndael 256 bit key (*password* 3 bit *key*). Menggunakan file adm_siswa.php berukuran 5.190 byte (6 KB). Waktu akses sebesar 1 detik, speed 5 KB/second



Gambar 26 Pengujian aksesibilitas proses enkripsi file text pada AEP

Pada proses dekripsi AEP waktu aksesibilitasnya sebesar 5 KB/detik.

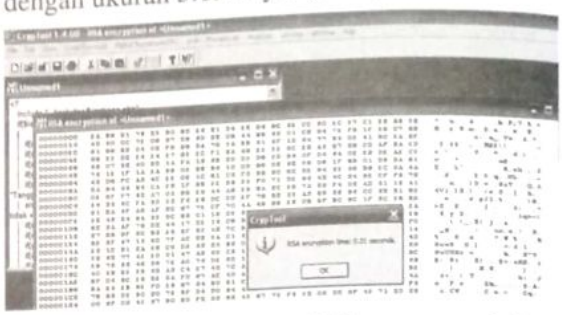


Gambar 27 Pengujian aksesibilitas proses dekripsi file gambar pada AEP

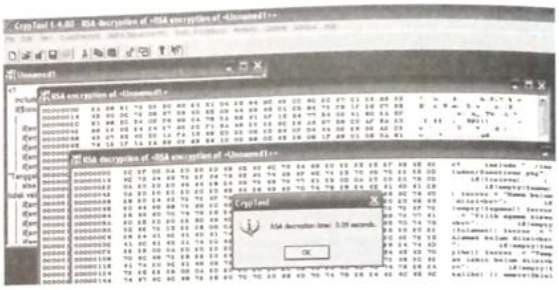
c) Proses Enkripsi dan Dekripsi Berdasarkan Algoritma RSA

Proses enkripsi dan dekripsi ini menggunakan aplikasi Cryptool berdasarkan algoritma RSA 512 bit (*password* 4 bit *key*)

Menggunakan sebuah file *text* adm_siswa.php, dengan ukuran 5.190 byte (6 KB)



Gambar 28 Pengujian aksesibilitas proses enkripsi file text berdasarkan algoritma RSA



Gambar 29 Pengujian aksesibilitas proses dekripsi file text berdasarkan algoritma RSA

Waktu aksesibilitas untuk proses enkripsi sebesar 0,01 detik dan pada proses dekripsinya 0,09 detik.

Perbandingan proses aksesibilitas antara FED, AEP, dan RSA secara terperinci dapat dilihat pada tabel 10.

Tabel 10 Perbandingan Proses Aksesibilitas FED, AEP, dan RSA

NAMA FILE	UKURAN FILE (byte)	FED BASE 64 bit (8 bit key)		AEP (3 bit key)		RSA (4 bit key)	
		WAKTU ENCODE (Pengacakan GP dengan Compress)	WAKTU DECODE (Pengacakan GP dengan Compress)	WAKTU ENKRIPSI Algoritma Rijndael 256 bit key (1 KB/SECOND)	WAKTU DEKRIPSI Algoritma Rijndael 256 bit key (1 KB/SECOND)	WAKTU ENKRIPSI RSA key type 512 bit	WAKTU DEKRIPSI RSA key type 512 bit
adm_agenda.php	1.997 (2 KB)	0,09	0,04	2,00	2,00	0,00	0,03
adm_siswa.php	5.190 (6 KB)	0,23	0,11	6,00	6,00	0,01	0,09
Firewall.gif	16.900 (17 KB)	0,74	0,36	16,00	17,00	0,01	0,31
001 Al Fatihah.mp3	356.304 (347 KB)	14,78	7,30	347,00	348,00	0,46	6,35

Keterangan : GP: Ganjil GenaP, FED: Fungsi *Encode Decode*, AEP: *Advance Encryption Package*, RSA: Rivest, Shamir & Adleman

Tabel 11 Prosentase Selisih Waktu Proses Enkripsi dan Dekripsi FED, AEP, dan RSA

NAMA FILE	UKURAN FILE (byte)	WAKTU ENKRIPSI		WAKTU DEKRIPSI	
		FED BASE64 bit (8 bit key) DENGAN AEP (3 bit key)	FED BASE64 bit (8 bit key) DENGAN RSA (4 bit key)	FED BASE64 bit (8 bit key) DENGAN AEP (3 bit key)	FED BASE64 bit (8 bit key) DENGAN RSA (4 bit key)
adm_agenda.php	1.997 (2 KB)	95,50%	9,00%	98,00%	1,00%
adm_siswa.php	5.19 (6 KB)	96,17%	22,00%	98,17%	2,00%
Firewall.gif	16.9 (17 KB)	95,65%	73,00%	97,88%	5,00%
001 Al Fatihah.mp3	356.304 (347 KB)	95,74%	1432,00%	97,90%	95,00%

a) Perbandingan proses enkripsi

Perbandingan proses enkripsi menggunakan file *text* adm_agenda.php, ukuran file 1.997 byte (2 KB). Perbandingan proses enkripsi antara FED dengan AEP, selisih waktu proses enkripsinya adalah 1-(0,09/2)=0,955 berarti FED lebih cepat 0,955 detik (95,50%)dari pada AEP.

Perbandingan proses enkripsi antara FED dengan RSA, selisih waktu proses enkripsinya adalah 0,09-0,00=0,090 berarti FED lebih lambat 0,090 detik (9,00%)dari pada RSA

b) Perbandingan proses dekripsi

Perbandingan proses dekripsi menggunakan file *text* adm_agenda.php, ukuran file 1.997 byte (2 KB). Perbandingan proses dekripsi antara FED dengan AEP, selisih waktu proses dekripsinya adalah 1-(0,04/2) = 0,980 berarti FED lebih cepat 0,980 detik (98,00%) dari pada AEP.

Perbandingan proses dekripsi antara FED dengan RSA, selisih waktu proses dekripsinya adalah 0,04-0,03=0,01 berarti FED lebih lambat 0,01 detik (1,00%) dari pada RSA.

4. KESIMPULAN

1. Penelitian ini dapat menghasilkan sebuah fungsi yang diberi nama FED (Fungsi *Encode* dan *Decode*). FED merupakan sebuah fungsi yang dirancang menggunakan bahasa pemrograman PHP, berdasarkan algoritma *MIME Base64 Encoding* yang telah dimodifikasi menggunakan 3 buah metode pengacakan. Fungsi ini dapat digunakan untuk merubah dari *plaintext* menjadi *ciphertext*, dengan cara merubah karakter ASCII menjadi karakter Base64.
2. Berdasarkan data hasil pengujian validitas, dapat diketahui bahwa *ciphertext* hasil proses *encoding* dari FED tidak dapat di *decoding* menggunakan aplikasi-aplikasi Base64 lain. Sehingga untuk transformasi ke dalam bentuk *plaintext* kembali harus melalui proses pemulihan dari pengacakan yang terdapat dalam FED.
3. Berdasarkan data pengujian aksesibilitas, dapat diketahui bahwa aksesibilitas FED lebih cepat dari pada AEP. Tetapi tidak lebih cepat dari pada aplikasi berdasarkan RSA. FED lebih cepat dari pada AEP dimungkinkan karena:
   a. FED dirancang berdasarkan algoritma yang berbasis 64 bit, yang lebih kecil dari pada RSA yang dirancang berdasarkan algoritma berbasis 512 bit.
   b. Algoritma yang digunakan pada FED juga relatif lebih sederhana (tidak terlalu rumit) sehingga waktu akses juga semakin cepat.

Keterlambatan FED dibandingkan dengan RSA dikarenakan ;

a. Ukuran file hasil dari *encoding* yang dilakukan dengan menggunakan Base64 akan selalu lebih besar sebanyak 4/3 kali dari ukuran file aslinya. Hal ini sesuai dengan konsep dasar cara kerja dari algoritma Base64 yaitu 1 *byte* karakter ASCII akan dirubah menjadi 3 *byte* dengan menambahkan *padding* 2 *byte*. Sebagai contoh file adm_siswa.php yang berukuran 5.190 byte, setelah di-*encode* menjadi 6.920 byte dengan perhitungan (4/3) x 5.190 = 6.920 byte. Namun dengan melalui proses kompresi, maka ukuran file dapat diminimalkan.
b. FED sudah mengalami modifikasi dalam proses *encoding* dan *decoding*-nya sehingga menjadi tidak standar lagi. Modifikasi tersebut antara lain proses metode pengacakan, proses *compress*, proses *uncompress*, dan proses pemulihan dari pengacakan. Total waktu *encode* = waktu pengacakan + waktu *compress* + waktu *encode*. Total waktu *decode* = waktu *decode* + waktu *uncompress* + waktu pemulihan dari pengacakan. Sehingga waktu yang ditampilkan adalah total dari perhitungan tersebut, dengan demikian waktu proses eksekusi programpun juga bertambah.

4. Algoritma dengan basis bit yang lebih kecil (misalnya 64 bit) belum dapat menentukan bahwa proses aksesibilitasnya juga akan lebih cepat dari pada algoritma yang berbasis lebih besar dari 64 bit. Masih terdapat faktor lain seperti tingkat kerumitan algoritma, *key space* yang disediakan, metode *compress* yang digunakan, juga ikut menentukan waktu akses. Meskipun demikian, fungsi ini dapat digunakan sebagai salah satu alternatif pilihan untuk mengamankan *source code* PHP pada *web server*.

DAFTAR PUSTAKA

Baldwin, R. G., 2004, Understanding Base64 Data, http://www.developer.com/java/other/ article.PHP/3386271, diakses tanggal 20 Juli 2007

Charles, R. M., (2001-2002), How to Base 64, http://www.kbcafe.com/articles/HowTo.Base64.pdf, diakses tanggal 20 Juli 2007

Heriyanto, T. (1999, Juni, 27). Pengenalan Kriptografi, (0.0.5). [OnLine], Sany Asy'ariBlog, http://www.tedi-h.com/papers/p_kripto.html, diakses tanggal 30 Juli 2007

Keamanan Password dan Enkripsi, http://home.netscape.com/assist/security/ smime/overview.html, diakses 20 Juli 2007

Kurniawan, Y, (2004, April), Kriptografi Keamanan Internet dan jaringan Telekomunikas, Bandung: Penerbit Informatika

PHP.net,function.base64_encode,http://id2.PHP.net/manual/en/function.base64decode.PHP, diakses tanggal 21 Juli 2007

PHP.net, function.base64_decode, http://id2.PHP.net/manual/en/function.base64decode.PHP, diakses tanggal 21 Juli 2007

Purwanto, M,F; & Herlambang M,T, 2002, Membangun Web Server dengan Linux, Jakarta : PT Elex Media Komputindo

Stalling, W. 2003, Cryptography and Network Security Principles and Practice, Prestice Hall, Third Edition

Syahputra, A, 2003, Apache Web Server, Yogyakarta : Andi Offset

Teguh, S, P, 2007, Tugas Akhir : Pemanfaatan MIME Base64 untuk Menyembunyikan Source Code PHP, *Magister Teknologi Informasi Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung*, http://www.cert.or.id/~budi/courses/security2006-2007/ReportSalmanTeguhP.pdf, diakses tanggal 9 Mei 2007

Tschabitscher, H. How MIME Works, http://email.about.com/cs/standards/a/ mime.htm, diakses tanggal 20 Juli 2007

Tschabitscher, H. How Base64 Encoding Works, http://email.about.com/cs/ standards/a/base64_encoding.htm, diakses tanggal 20 Juli 2007

Peterson, W. & Tandur S., 2007, Network Defense and Countermeasures, Element K Press LLC, Second Edition

Wikipedia.org, Base64, http://en.wikipedia.org/wiki/Base64, diakses tanggal 20 Juli 2007

Wikipedia.org, Uuencoding, http://en.wikipedia.org/wiki/Uuencoding#Theenco dingprocess, diakses tanggal 14 nopember 2007

CV Penulis

Dwi Retnoningsih, dosen tetap jurusan Teknik Informatika STMIK El Rahma Yogyakarta, menyelesaikan S1 jurusan Manajemen Informatika dan Teknik Komputer di Institiut Sains dan Teknologi AKPRIND Yogyakarta pada tahun 2000, Menyelesaikan S2 Jurusan Teknik Informatika di Magister Teknologi Informasi UGM Yogyakarta, pada tahun 2008.

DAFTAR SUSUNAN PANITIA

PROGRAM COMMITTEE

Prof. Dr.Ir. Prayoto, M.Sc.
Prof. Drs. Setiadji,S.U.
Dr. Ir. Inggriani Liem
Prof. H.Adhi Susanto, M.Sc., Ph.D
Prof.Drs. Suryo Guritno, M.Sc., Ph.D
Dr.Ir. Titon Dutono, M.Eng
Ir. Lukito Edi Nugroho, M.Sc., Ph.D
Drs. Retantyo Wardoyo, M.Sc., Ph.D.

PELAKSANA SEMINAR

Pelindung:
Ketua STMIK AKAKOM Yogyakarta

Penanggung Jawab:
Kepala Puslitbang dan PPM STMIK AKAKOM Yogyakarta

Panitia:
Agung Budi Prasetyo, S.Kom,M.Kom.
Ariesta Damayanti, S.Kom.
Ary Adjidharma AW,S.Kom,MMSi.
Deborah Kurniawati, S.Kom
Dwi Swarsono
Enny Itje Sela, S.Si.,M.Kom.
Fx. Henry Nugroho,S.T.
H. Sri Widodo
Indra Yatini B, S.Kom,M.Kom.
L.N. Harnaningrum,S.Si., M.T.
Ir. Mashudi
Dra. M. Titik Maryanti
Pulut Suryati,S.Kom.
Rita Darundia
Sri Rejeki,S.Si,M.Kom.
Dra. Hj. Syamsu Windarti, Apt,M.T.
Ir. Totok Suprawoto,M.M.M.T.
Wagito,S.T.,M.T.
Yohakim Marwanta, S.Kom.

YAYASAN PENDIDIKAN WIDYA BA
STMIK
AKAKOM
YOGYAKARTA
Terakreditasi A*(sangat baik)