ALGORITMA ENKRIPSI *RC4* SEBAGAI METODE *OBFUSCATION SOURCE CODE PHP*

Okie Setiawan[1*], Rina Fiati[1], Tri Listyorini[2]
[1] Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muria Kudus
Gondangmanis, PO Box 53, Bae, Kudus 59352
[*]Email: okie_setiawan@ymail.com

Abstrak

Source code program web dengan PHP merupakan sebuah bahasa pemrograman yang bersifat interpreter. Oleh karena itu, source code program yang menggunakan bahasa pemrograman PHP dapat dilihat dan dimanipulasi dengan mudah oleh user. Ini tentunya merupakan hal yang merugikan bagi developer program komersial yang menggunakan bahasa pemrograman PHP. Salah satu cara melindungi hak kekayaan intelektual tersebut adalah dengan cara obfuscation. Penelitian ini bertujuan menerapkan algoritma enksripsi untuk obfuscation source code PHP. Algoritma enkripsi yang akan digunakan yaitu RC4. Metode obfuscation yaitu memanipulasi isi source code program PHP yang berupa plain text mengunakan algoritma enkripsi RC4 sehingga tercipta cipher text. Hasil penelitian ini merupakan sebuah metode obfuscation untuk source code PHP dengan memanfaatkan algoritma enkrispi RC4 yang diterapkan dalam aplikasi.

***Kata kunci** : enkrispi, obfuscation, PHP*

1. PENDAHULUAN

PHP telah mengalami perkembangan pesat baik dari segi fungsi maupun penggunanya. Berkembangnya PHP tidak lain karena bahasa pemrograman ini mempunyai beberapa keunggulan. Terutama karena sifatnya yang gratis dan *cross platform* atau dengan kata lain dapat digunakan diberbagai sistem operasi seperti linux, windows, dan mac. Pengembang PHP juga dapat mengintegrasikan PHP dengan salah satu alat database yang berbeda seperti MySQL, SQLite, PostgreSQL, DB2, MS SQL, Oracle, dan sebagainya, untuk membuat konten mereka sedinamis mungkin [1]. Namun sebuah masalah muncul ketika seorang *programmer/developer* dari bahasa pemrograman ini ingin mengkomersialkan hasil karya programnya. *Source code* yang berupa *plaintext* dapat dengan mudah dilihat dan dimanipulasi oleh *user*.

Ada beberapa cara yang dapat dilakukan untuk menyembunyikan *source code* tersebut, yaitu melalui metode *obfuscation*. Obfuscation atau obfuskasi adalah mentransformasi sintaks kode komputer namun dengan tetap memelihara semantik (isi) sehingga kerahasiaan tetap terjaga [2]. Metode ini hanya bersifat transformasi bentuk *source code* kedalam karakter yang tidak mudah dimengerti oleh manusia namun tetap bisa dimengerti oleh mesin. Dalam penelitian ini proses *obfuscation* dilakukan dengan algoritma RC4. RC4 merupakan algoritma enkripsi *stream chiper*. RC4 dirancang agar dapat diimplementasikan di *software* secara sangat efisien. Ini membuat RC4 sangat populer untuk aplikasi internet, antara lain RC4 digunakan dalam standard TLS (transport layer security) dan WEP (wireless equivalent privacy) [3].

Diharapkan aplikasi ini dapat digunakan untuk melindungi *source code* PHP agar tidak mudah dimanipulasi dan dapat membantu para *developer* program web yang menggunakan bahasa PHP dalam menjaga hak cipta atas program yang telah dibuatnya.

2. METODOLOGI

Metodologi penelitian ini menggunakan metodologi pengembangan sistem perangkat lunak. Berikut ini adalah langkah-langkah yang ada pada metodologi penelitian ini:

a. Kebutuhan Sistem

Kebutuhan sistem disini merupakan sebuah tahap untuk mengumpulkan data-data yang berhubungan dengan penelitian. Data-data tersebut diperoleh dari sumber data primer dan sumber data sekunder. Pada sumber data primer, data dihasilkan melalui proses wawancara dan observasi. Sedangkan pada sumber data sekunder diperoleh dari buku, jurnal, dan internet.

b. Spesifikasi Kebutuhan Perangkat Lunak

Analisis merupakan proses untuk menjabarkan segala sesuatu yang akan diproses oleh sebuah perangkat lunak.

c. Desain

1. Pemodelan Sistem

Dalam pemodelan sistem akan digunakan *Flowchart* sebagai penggambaran alur sistem.

2. *Design* Interface

Yaitu merancang tampilan dari aplikasi sebagai sarana interaksi antara *user* dan aplikasi.

d. Implementasi Perangakat Lunak

Yaitu tahapan dimana perancangan sistem yang telah dibuat dikonversi kedalam program aplikasi. Sehingga komputer mampu menjalankan fungsi-fungsi aplikasi dan memberikan layanan kepada pengguna aplikasi tersebut. Dalam pembuatan aplikasi, penulis menggunakan bahasa pemrograman PHP dan berbagai macam bahasa *web application* lainnya sebagai pembantu seperti HTML dan JavaScript.

e. *Testing*

Tahap *testing* yaitu tahap pengujian terhadap aplikasi. Metode yang digunakan untuk testing yaitu *black-box dan white-box testing*. *Black-box testing* merupakan pengujian terhadap fungsional aplikasi dapat beroperasi secara tepat. Sedangkan pengujian white-box didasarkan pada pengamatan yang teliti terhadap detail prosedural. Pada penelitian ini pengujian *black-box* dikatakan berhasil jika fungsi-fungsi yang ada pada aplikasi dapat memenuhi spesfikasi yang telah ditetapkan tersebut. Namun pada pengujian *white-box* hanya berfokus pada *obfuscation*, yaitu pengujian terhadap hasil *output* program. Pengujian *white-box* dikatakan berhasil jika hasil *output* program yang berupa *source code* PHP yang telah ter *obfuscation* dapat berjalan sebagaimana mestinya seperti source code aslinya.

3. HASIL DAN PEMBAHASAN

3.1. Analisa Kebutuhan User

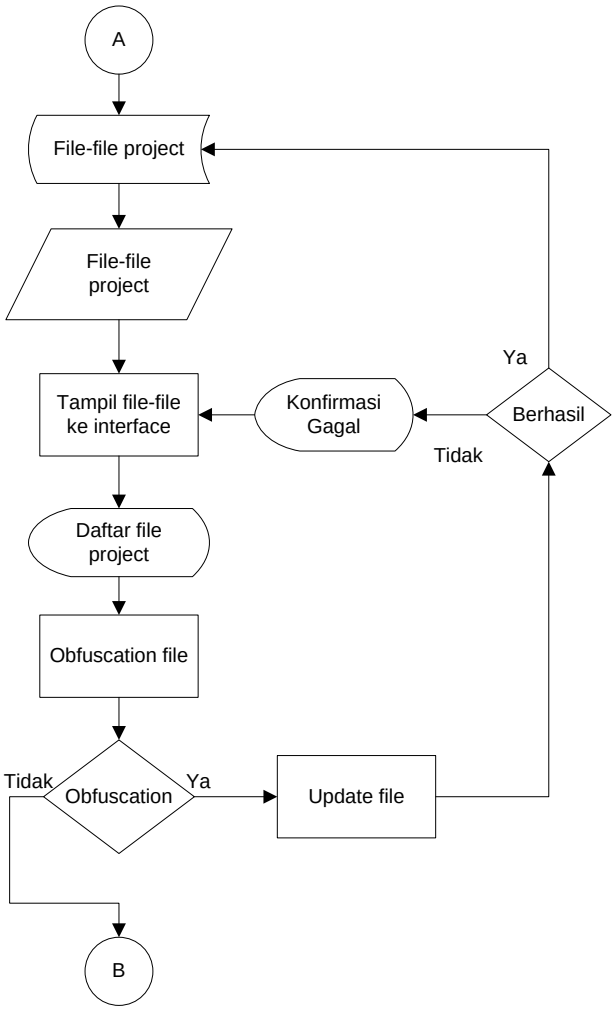
Untuk memenuhi kebutuhan *user* dalam melakukan *obfuscation*, aplikasi yang dibangun harus memenuhi hal - hal berikut ini :

a. Melakukan input *file project* kedalam sistem yang berupa file *archive* zip untuk mempermudah *user* dalam melakukan *input* dikarenakan program dengan bahasa PHP terdiri dari banyak *file*.
b. Menyediakan *interface* yang dapat menampilkan *file* dari isi *file project* yang diinputkan *user*.
c. Melakukan proses *obfuscation* pada *file* PHP yang dikehendaki *user*.
d. Memberikan *output* berupa *project* yang telah terobfuskasi dalam bentuk *archive* zip.

Pada proses *input* yang dilakukan oleh *user*, *project* yang diinputkan yaitu *file project* yang telah terkompresi *archive* zip. Format inputan berupa *archive* dikarenakan pertimbangan untuk mempermudah *user* dalam *input file project* yang berjumlah banyak.

Proses pertama untuk *file* inputan *user* yaitu *uploading* ke *server*. Proses *upload* dimaksudkan untuk memberikan hak akses penuh kepada sistem untuk mengolah *file project* yang diinputkan *user*. Hal ini dikarenakan sistem aplikasi untuk *obfuscation* menggunakan PHP yang mempunyai keterbatasan hak akses untuk mengolah *file* diluar *file server*. Saat file terupload didalam *server*, selanjutnya otomatis sistem akan melakukan proses ekstraksi. Proses ekstraksi ini akan mengekstrak seluruh file yang berada dalam file *archive* zip. Hasil ekstraksi tersebut juga akan tersimpan dalam *server*. Setelah proses ekstraksi berhasil dilakukan, *interface* sistem akan menampilkan daftar *file* PHP dan *directory* hasil ekstrak.

Untuk alur sistem *obfuscation* pada aplikasi yang dirancang dapat dilihat pada gambar 3.1.



Gambar 3.1 Sistem *obfuscation*

Pada gambar 3.1 alur proses dimulai dari daftar *file project* yang telah tersimpan pada *server* ditampilkan di *interface* sistem. Kemudian *user* dapat memilih *file* yang ingin dilindungi atau diobfuskasi *source code* aslinya. Proses ini merupakan proses pengubahan/*update file* yang berisi *script* asli menjadi *script* hasil *obfuscation* (cipher text).

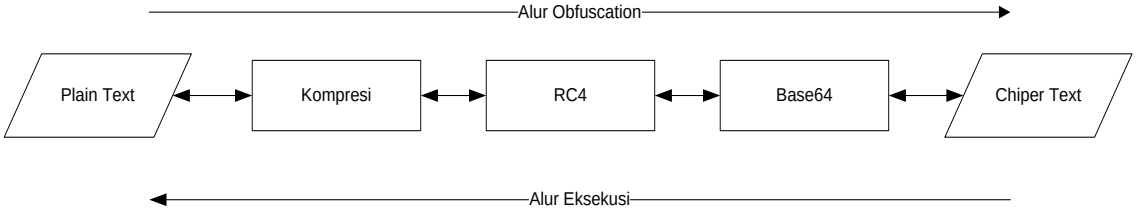
Setelah proses *obfuscation* selesai, *user* dapat mendownload *project* hasil sebagai *output* dari sistem. *Output* dari sistem yaitu berupa *file project* hasil proses *obfuscation* yang terkompresi *archive* zip. Proses dimulai dari *file-file project* yang berada di *server* kemudian di kompresi menjadi zip. *File* zip yang tercipta disimpan di *server* terlebih dahulu dan kemudian dikirimkan ke *user* melalui proses *download*.

3.2. Cara Kerja Sistem *Obfuscation*

Dalam bahasa PHP, terdapat aturan *coding* atau *script* agar bisa dieksekusi oleh mesin pemroses PHP. Oleh karena itu tidak serta merta *script* hasil *obfuscation* dapat diproses PHP. Dibutuhkan teknik sinkronisasi pada *script* hasil *obfuscation* agar dapat diproses sebagaimana mestinya. Prinsip kerja *obfuscation* sebenarnya tidak sama seperti enkripsi. Enkripsi menyembunyikan data dengan kunci dan merubah data tersebut menjadi tidak bermakna bagi siapapun termasuk mesin. Hal ini berbeda dengan *obfuscation* yang merubah data menjadi tidak bermakna bagi *user* (manusia) namun dapat dibaca oleh mesin.

Teknik *obfuscation* umumnya mengubah sintaks *script* tanpa mengubah semantiknya, sehingga walaupun tulisan *script* menjadi susah untuk dibaca namun ketika dieksekusi masih tetap dapat dijalankan seperti sebelum di *obfuscation* [4]. Pada penelitian ini, *obfuscation* dilakukan dengan menerapkan algoritma enkripsi RC4. Agar dapat dieksekusi mesin PHP, kode program

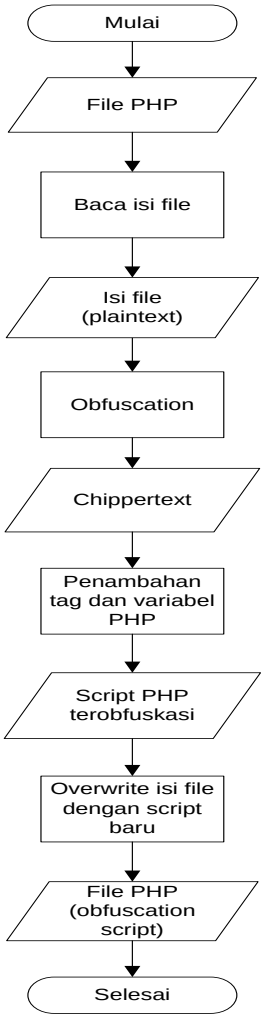
haruslah sesuai dengan kaidah atau aturan yang berlaku dibahasa pemrograman PHP. Untuk itu digunakan fungsi base64 sebagai fungsi transformasi hasil enksripsi dengan RC4 yang berbentuk simbol ASCII ke karakter yang dapat disimpan ke variabel PHP. Penggunaan base64 ini tentunya memberikan karakter yang lebih panjang dan menyebabkan ukuran file menjadi lebih besar. Untuk megatasi hal tersebut dilakukan fungsi kompresi string agar ukuran *file* lebih kecil. Untuk lebih jelas alur obfuscation dapat dilihat pada gambar 3.2.



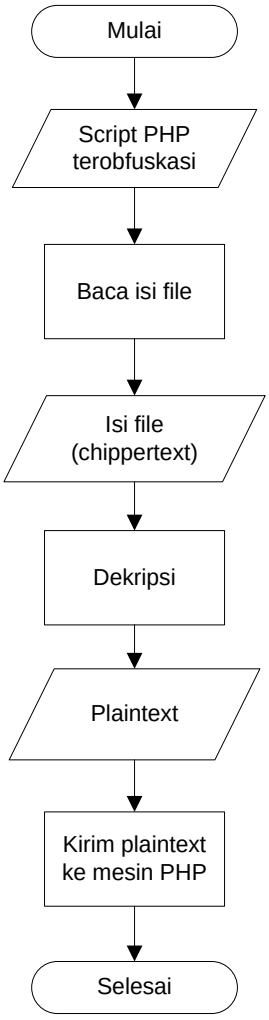
Gambar 3.2 Alur Obfuscation

Sedangkan untuk cara kerja aplikasi dalam pembuatan *obfuscation* pada penelitian ini dapat dilihat pada gambar 3.3.

Setelah *file* PHP baru dengan *script obfuscation* terbuat bukan berarti *file* tersebut sudah dapat dieksekusi. Script yang terenkripsi dengan algoritma RC4 haruslah dikembalikan menjadi kode *script* asli agar dapat dimengerti oleh mesin pembaca *script* di PHP. Untuk itu diperlukan sebuah *file key* untuk dekripsi sebelum kode dikirim ke mesin/komputer. Untuk alur aplikasi dalam eksekusi *file* PHP terobfuskasi dapat dilihat pada gambar 3.4



Gambar 3.3 Cara Kerja Aplikasi



Gambar 3.4 Eksekusi File PHP Terobfuskasi

3.3. Algoritma RC4

Algoritma RC4 mengenkripsi dengan mengombinasikannya dengan *plainteks* dengan menggunakan *bit-wise* Xor (Exclusive-or). RC4 menggunakan panjang kunci dari 1 sampai 256 byte yang digunakan untuk menginisialisasikan tabel sepanjang 256 byte. Tabel ini digunakan untuk generasi yang berikut dari *pseudo random* yang menggunakan XOR dengan *plaintext* untuk menghasilkan ciphertext. Masing - masing elemen dalam tabel saling ditukarkan minimal sekali [5]. Proses dekripsinya dilakukan dengan cara yang sama (karena Xor merupakan fungsi simetrik). Untuk menghasilkan *keystream, cipher* menggunakan *state* internal yang meliputi dua bagian :

1. Tahap *key scheduling* dimana *state automaton* diberi nilai awal berdasar kan kunci enkripsi.

   State yang diberi nilai awal berupa *array* yang merepresentasikan suatu permutasi dengan 256 elemen, jadi hasil dari algoritma KSA adalah permutasi awal. *Array* yang mempunyai 256 elemen ini (dengan indeks 0 sampai dengan 255) dinamakan S. Berikut adalah algoritma KSA dalam bentuk *pseudo-code* dimana *key* adalah kunci enkripsi dan *keylength* adalah besar kunci enkripsi dalam *bytes* (untuk kunci 128 bit, keylength = 16)

```
for i = 0 to 255
  S[i] := i
j := 0
for i = 0 to 255
  j := (j + S[i] + key[i mod keylength]) mod 256
  swap(S[i],S[j])
```

2. Tahap pseudo-random generation dimana state automaton beroperasi dan outputnya menghasilkan *keystream*. Setiap putaran, bagian *keystream* sebesar 1 *byte* (dengan nilai antara 0 sampai dengan 255) dioutput oleh PRGA berdasarkan *state* S. Berikut adalah algoritma PRGA dalam bentuk *pseudo-code*:

```
i := 0
j := 0
loop
  i := (i + 1) mod 256
  j := (j + S[i]) mod 256
  swap(S[i],S[j])
  output S[(S[i] + S[j]) mod 256]
```

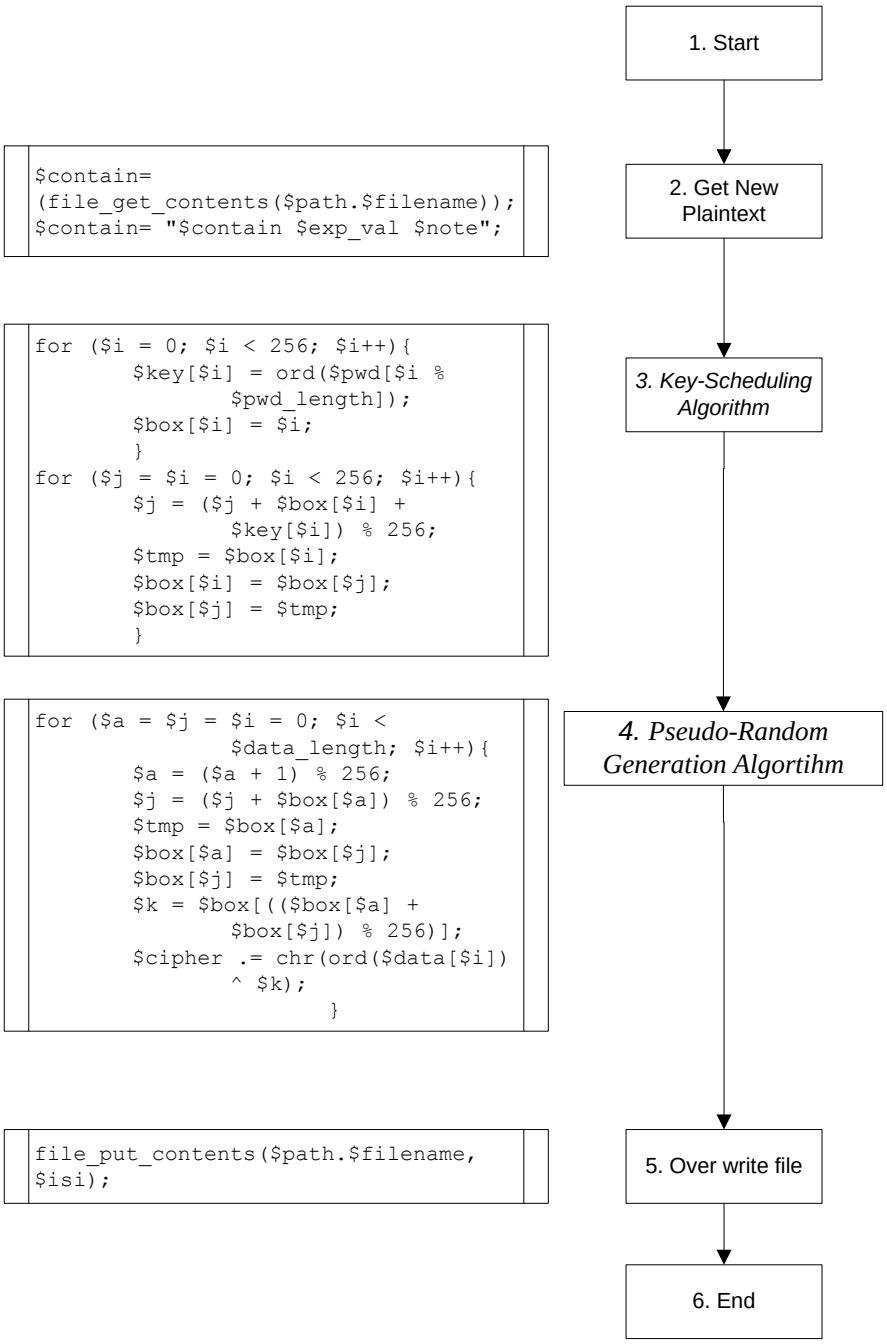
Setelah terbentuk *keystream*, kemudian *keystream* tersebut dimasukkan dalam operasi XOR dengan plaintext yang ada, dengan sebelumnya pesan dipotong-potong terlebih dahulu menjadi *byte-byte*.

3.4. Transformasi Base64

Transformasi base64 merupakan salah satu algoritma untuk *encoding* dan *decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metode yang digunakan untuk melakukan *encoding* (penyandian) terhadap data *binary*. Karakter yang dihasilkan pada transformasi Base64 ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data *binary* atau pengisi pad. Karakter simbol yang akan dihasilkan akan tergantung dari proses algoritma yang berjalan [6].

3.5. Hasil Pengujian

Untuk hasil pengujian *obfuscation* dilakukan dengan *whitebox testing.* Testing dilakukan dengan menganalisa jalur proses yang dilewati oleh aplikasi. Untuk alur proses *obfuscation* pada aplikasi dapat dilihat pada gambar 3.5.



Gambar 3.5 Jalur Proses *Obfuscation*

Dari gambar 3.5 dapat diambil kesimpulan kemungkinan jalur yang dilalui yaitu 1-2-3-4-5-6 Hasil pengujian *whitebox* untuk proses *obfuscation* ditunjukkan pada tabel 3.1.

Tabel 3.1 Hasil Pengujian *Obfuscation*

No	Skenario pengujian	Jalur yang dihrapakan	Jalur pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1	Pembuatan obfuscation	1-2-3-4-5-6	Sesuai	Script terobfuskasi	Sesuai	Valid

Keterangan :

Pada pengujian *whitebox* berbasis *basic path* dilakukan pengujian dengan cara melalui masing-masing jalur minimal satu kali.

Untuk mengetahui keberhasilan *obfuscation* maka dilakukan pengujian terhadap *file output* yang telah dilakukan proses *obfuscation*. Pengujian ini ditekankan pada hasil *output* yang diberikan oleh *file* terobfuskasi. *Obfuscation* dikatakan berhasil jika *output* yang diberikan mempunyai nilai sama dengan *output* dari *file* asli.

Pada pengujian *output*, *project* PHP yang digunakan sebagai *sample* adalah CMS toko online lokomedia versi 1.3.4. Adapun parameter yang diambil pada pengujian yaitu kesesuaian *output*, waktu eksekusi, dan ukuran *file*. Untuk hasil pengujian dapat dilihat pada tabel 3.2.

Tabel 3.2 Hasil Pengujian *File Output*

No	Nama file	Ukuran File (byte)		Waktu eksekusi (detik)		Kesesuaian output	Kesimpulan
		Asli	Obfuscated	Asli	Obfuscated		
1	Media.php	4752	2492	0.05220	0.12948	Sama	Berhasil
2	Koneksi.php	276	664	0.02049	0.02342	Sama	Berhasil
3	Kanan.php	3371	2156	0.02115	0.02445	Sama	Berhasil
4	Kiri.php	1681	1208	0.00300	0.01620	Sama	Berhasil

Keterangan :

Waktu eksekusi dapat berbeda pada pengujian yang berbeda. Hal ini disebabkan beberapa faktor seperti spesifikasi komputer dan proses yang ditangani komputer.

Untuk pengujian waktu eksekusi yaitu dengan menggunakan fungsi *microtime* yang telah disediakan oleh bahasa pemrograman PHP.

Pada hasil pengujian *file output* didapatkan hasil sebagai berikut :

1. Ukuran *file* hasil *obfuscation* cenderung lebih kecil. Hal ini dikarenakan digunakan kompresi agar ukuran *file* tidak terlalu besar akibat pembuatan variabel menggunakan fungsi base64.
2. Waktu eksekusi *file* hasil *obfuscation* lebih lama dikarenakan *server* PHP perlu mendekripsi *chipper text* terlebih dahulu sebelum melanjutkan proses.
3. Hasil selisih rata-rata waktu eksekusi file yaitu sebesar 0.024178 Detik, dengan prosentase 199.8658 % atau dua kali lebih lama.
4. *File* hasil mengalami perubahan/*obfuscated*.
5. Proses *obfuscation* dikatakan berhasil karena *output* eksekusi (di browser) *file* yang terobfuskasi sama dengan yang dihasilkan *file* asli.

4. KESIMPULAN

Dari hasil analisis dan perancangan, serta implementasi dan pembahasan pada bab-bab sebelumnya dalam skripsi ini dapat diambil kesimpulan sebagai berikut:

(5) Algoritma enkripsi dapat digunakan untuk *obfuscation source code* PHP. Salah satunya menggunakan algoritma RC4 untuk mengubah *plaintext script* menjadi *chiper text* yang disimpan pada variabel PHP.

(6) Penelitian menghasilkan perangkat lunak yang dapat digunakan untuk *obfuscation source code* PHP sebagai sarana untuk menjaga keamanan hak cipta dan kerahasiaan *source code* program.

(7) Waktu eksekusi file PHP yang terobfuskasi cenderung lebih lama daripada *file source code* asli dengan prosentase 199.8658 %.

UCAPAN TERIMA KASIH

Pada kesempatan kali ini kami mengucapkan terimakasih kepada rekan-rekan peneliti yang membantu dalam persiapan dan pelaksanaan penelitian dan kepada fakultas teknik Universitas Muria Kudus yang telah menyediakan sarana dalam pengadaan penelitian.

DAFTAR PUSTAKA

MacIntyre, Peter., (2010), *PHP: The Good Parts* : O'Reilly Media, ISBN 978-0-596-80437-4.

Coliberg, CS., (2002), Watermarking, Temper-Proofing and Obfuscation - Tools for Software Protection, IEEE Transc On Software Engineering Vol 28 No 6.

Kromodimoeljo Sentot, (2010), *Teori & Aplikasi Kriptografi* : SPK IT Consulting, ISBN 978-602-96233-0-7.

Wardiana, Wawan.,(2009), *Pencegah Pembajakan Perangkat Lunak dengan Menggunakan Teknik Identity-Based Encryption dan Obfuscation*, Jurnal INKOM, Vol. III, No. 1-2

Purnomo., (2012), *Implementasi Algoritma Kriptografi RC4 Pada DSP TMS320C6713 Sebagai Pendukung Sekuritas Jaringan Komunikasi Voice over Internet Protocol (VoIP)*, Jurnal EECCIS (Electrical Power Electronic Comunication Control Information Seminar) Vol. 6, No. 2

Wahyu., dkk., (2012), *Penerapan Algoritma Gabungan RC4 Dan Base64 Pada Sistem Keamanan E-Commerce*, Jurnal Seminar Nasional Aplikasi Teknologi Informasi, ISSN 1907-5022.