Pencegahan Ancaman Reverse Engineering Source Code PHP dengan Teknik Obfuscation Code pada Extension PHP

Alam Rahmatulloh[1], Rinaldi Munir[2]

Program Studi Magister Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha No. 10, Bandung, Jawa Barat, 40132
[1] alamraya@students.itb.ac.id, [2] rinaldi@informatika.org

Abstrak

Perkembangan media digital sangat melaju dengan pesat, namun keamanan hak cipta dan kekayaan intelektualnya menjadi masalah yang sangat penting. Hal ini dikarenakan maraknya pembajakan media digital seperti pembajak *software* atau *source code*. Aplikasi dapat dengan mudah didapatkan, dimodifikasi (*reverse engineering*), diduplikasi atau disebarkan tanpa seizin pembuatnya. Masalah ini ditemui salah satunya pada *source code PHP*, sehingga memerlukan pengamanan pada *source code* agar tidak dapat dibaca secara langsung. Jadi dalam penelitian ini fokus bagaimana cara melindungi *source code* PHP.
Pada penelitian ini akan dilakukan pengamanan *source code PHP* dengan teknik *obfuscation code* menggunakan *custom execution engine* pada *library PHP* baru. Algoritma enkripsi yang digunakan sebagai *obfuscation* berjenis *stream cipher* (RC4) dan encoding menggunakan 85 karakter ASCII. Dengan menerapkan teknik ini dihasilkan *library PHP* baru yang mampu mengamankan *source code PHP* dengan perubahan ukuran data rata-rata 30,30% dari *file PHP* aslinya dan mampu melakukan perubahan *execution time* rata-rata 28,28% dari *file PHP* aslinya.

Kata kunci : enkripsi, *library PHP*, *obfuscation code, reverse engineering, source code*

1. Pendahuluan

Di era digital ini keamanan hak cipta dan kekayaan intelektual menjadi masalah yang sangat penting seiring perkembangan teknologi yang sangat pesat. Diantaranya mengenai pembajakan perangkat lunak, pembajak *software* atau *source code* aplikasi dapat dengan mudah mendapatkan, memodifikasi (*reverse engineering*), menduplikasi atau menyebarkan *source code* tanpa seizin pembuatnya [1] [2] [3] [4]. Tidak hanya *executable file* saja yang harus diproteksi, *source code* perangkat lunak juga sangat rentan dan tidak aman sehingga memerlukan proteksi. Hal ini dapat ditemukan pada aplikasi berbasis web, dimana *source code* tidak *dicompile* menjadi *executable file*, melainkan hanya di upload dan dijalankan pada sisi *server*. *Source code* yang dibahas pada penelitian ini adalah PHP.

PHP adalah bahasa *server-side scripting* yang banyak digunakan untuk mengembangkan situs dinamis [5]. Hanya dalam beberapa tahun PHP dengan cepat berevolusi dari bahasa kecil yang sederhana menjadi bahasa pengembangan web yang sangat kuat dan banyak digunakan lebih dari 14 juta situs web. Saat ini PHP semakin stabil dan lebih diperluas dikembangkan dari versi sebelumnya.

Sayangnya aplikasi PHP berbasis web didistribusikan dalam bentuk *source code,* sehingga memiliki beberapa kekurangan dan kerentanan celah keamanan. Beberapa kekurangan itu diantaranya *source code* dapat dengan mudah disalin, modifikasi ataupun digunakan sebagian dalam aplikasi lainnya tanpa ada pemberitahuan. Selain itu *source code* yang tidak terproteksi membuat aplikasi yang dibangun sangat rentan, karena *source code*

dapat mengungkapkan beberapa kelemahan dari aplikasi yang dibuat.

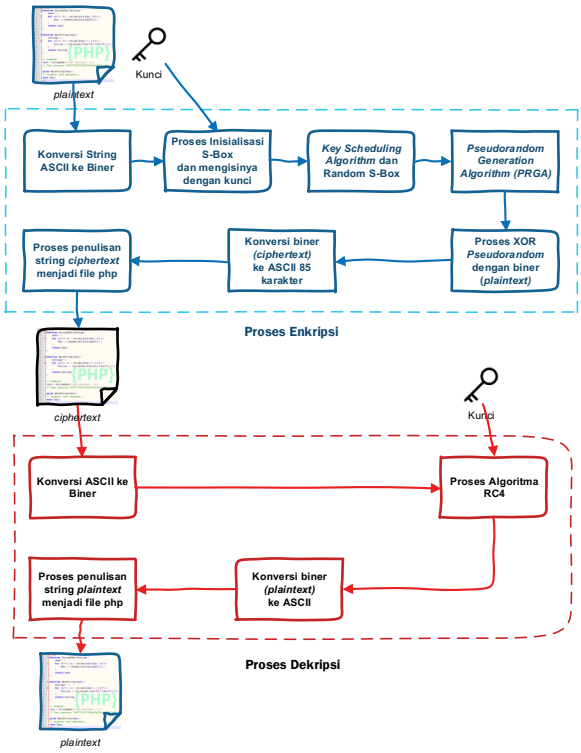
Berdasarkan isu-isu keamanan seperti ini telah banyak dikembangkan berbagai macam metode dan teknik untuk dapat mengamankan *source code*, salah satunya seperti teknik *obfuscation*. *Obfuscated* adalah teknik menyamarkan kode dengan tetap memelihara semantik (isi) data, dengan tujuan agar tidak dengan mudah dibaca orang lain [6] [7] [8] [9]. Namun pada penelitian yang telah dilakukan sebelumnya, teknik ini dapat dengan mudah dibongkar dikarenakan penyimpanan atau penggunaan teknik *obfuscation code* disimpan pada file terpisah saja. Selain itu teknik *obfuscation* yang dilakukan hanya penggunaan *base64 encode* yang sudah jelas untuk proses *decode* nya juga sudah disediakan oleh PHP. Kelemahan lain adalah pada file hasil *encode* menjadi bertambah 30% lebih besar dari file aslinya dan waktu eksekusi semakin lama [10] [11] [12].

Oleh karenanya pada penelitian ini metode yang diusulkan yakni penggunaan teknik *obfuscation code* dengan algoritma *stream cipher (RC4), encode* dengan 85 karakter ASCII dan cara eksekusi (*custom execution engine*) yang berbeda pada *extension PHP* baru [13] [14] [15] [1]. CEE merupakan penambahan *library PHP* baru dengan proses *interpreter PHP* yang berbeda. Pada penelitian ini dipilih algoritma enkripsi RC4 dan teknik CEE berdasarkan beberapa alasan yakni: 1) karena algoritma *stream cipher* cocok untuk digunakan pada enkripsi suatu komunikasi yang berlangsung secara berkelanjutan [7], 2) CEE mampu mengurangi perubahan waktu eksekusi dari file aslinya, 3) penggunaan 85 karakter ASCII mampu mengurangi perubahan ukuran file dari *source code PHP* aslinya [14].

Adapun penelitian ini memiliki tujuan utama yakni 1) membuat *extension PHP* baru dengan teknik *obfuscation code* menggunakan algoritma rc4 dan proses *encoding* oleh 85 karakter ASCII, dan 2) melakukan pengujian ukuran dan kecepatan waktu eksekusi *file PHP* terobfukasi.

2. Usulan Metode

Secara umum metode yang diusulkan pada penelitian ini, mulai dari tahap enkripsi dan dekripsi *source code PHP* dapat dilihat pada Gambar 1.



Gambar 1. Metode Yang Diusulkan

2.1. Proses Enkripsi

Rincian proses enkripsi *source code PHP* dalam penelitian ini adalah sebagai berikut :

Step 1 : Konversi String ke *Biner* per karakter.

Step 2 : Inisialisasi *State Array*. Terdapat dua buah array S dan K. Array S sebesar 256 bit diinisialisasi dengan bilangan 0 sampai 255, sedangkan array K dengan panjang 256 bit dan diisi dengan kunci dengan panjang 1-256 bit secara berulang sampai array K terisi seluruhnya.

Step 3 : *Key Scheduling Algorithm (KSA)* digunakan untuk menginisialisasi permutasi di *array* S. Panjang kunci didefinisikan sebagai jumlah *byte* di kunci dan mempunyai rentang panjang kunci dari 1 sampai 256.

Step 4 : *Pseudo-Random Generation Algortihm (PRGA)*. Setelah memiliki *state array* yang telah teracak, maka lakukan inisialisasi kembali i dan j dengan 0. Untuk menghasilkan kunci enkripsi (*cipher byte*) yang akan di-XOR-kan dengan plainteks, maka lakukan PRGA (*Pseudo-Random Generation Algortihm*) yang bertujuan memodifikasi state dan output sebuah

byte dari *keystream*. Hal ini penting karena banyaknya dibutuhkan iterasi sehingga perlu pembangkitan *random byte*. Dalam setiap iterasi, PRGA menginkremen , menambahkan nilai S yang ditunjuk oleh i sampai j, kemudian menukar nilai S[i] dan S[j], lalu mengembalikan elemen dari S di lokasi S[i] + S[j] (modulo 256). Setiap elemen S ditukar dengan elemen lainnya paling tidak satu kali setiap 256 iterasi.

Step 5 : *XOR pseudorandom* dan *plaintext.* Tahap selanjutnya, setelah menemukan kunci untuk setiap karakter (*byte K)* yang didapatkan dari pembangkitan *random byte* PRGA, maka dilakukan pengulangan sebanyak jumlah karakter dan dilakukan perhitungan XOR tiap karakter pada *plaintext.* Sehingga didapatkan *ciphertext* per karakter.

Step 6 : Konversi *Ciphertext* dari *biner* ke kode ASCII 85 karakter (*Encoding*), dengan persamaan berikut ini :

```
x = N0 * 52200625 + N1 * 614.125 +
N2 * 7225 + N3 * 85 + N4
```

dimana nilai N0 ... N4 berada dalam kisaran 0 sampai 84. Cara alternatif lain untuk melakukan perhitungan yaitu :

```
N0 = (x / 52200625) mod 85
N1 = (x / 614125) mod 85
N2 = (x / 7225) mod 85
N3 = (x / 85) mod 85
N4 = x mod 85
```

2.2. Proses Dekripsi

Pada penelitian ini proses dekripsi dilakukan ketika proses eksekusi file PHP atau proses pengembalian *source code* terobfukasi menjadi file asli. Adapun rincian proses yang dilakukan pada tahap ini adalah sebagai berikut:

Step 1 : Proses decode ASCII 85 menjadi ASCII 0-255.

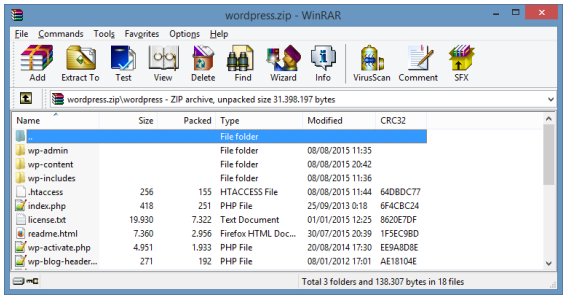
Step 2 : Konversi *string* pada kunci dan *source code PHP* dari kode ASCII menjadi *biner* per karakter. *Binary* kunci akan digunakan untuk proses pengisian *array K* secara berulang sampai *array* tersebut terisi seluruhnya.

Step 3 : Proses dekripsi algoritma RC4. Tahapan ini sama dengan proses enkripsi dari mulai inisialisasi *state array*, KSA, PRGA dan proses *XOR* antara *pseudorandom* dan *ciphertext*.

Step 4 : Proses terakhir adalah konversi kode biner menjadi ASCII per karakter, sehingga *source code* menjadi plaintext semula kemudian lakukan proses penulisan pada sebuah *file* berekstensi *PHP*.

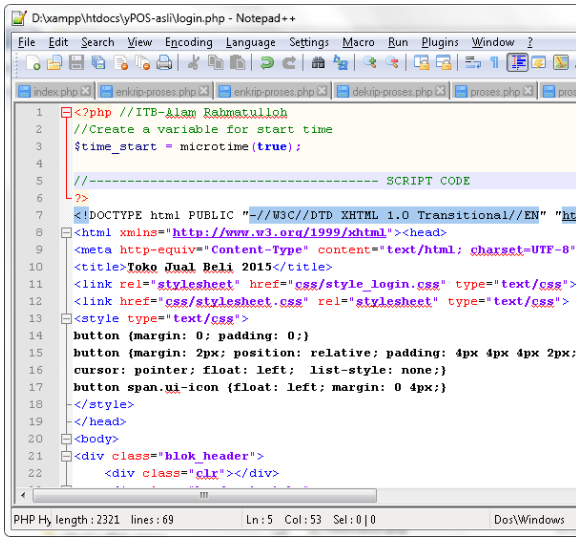
3. Eksperimen dan Pembahasan

Metode usulan diatas diprogram menggunakan bahasa pemrograman C++. Pada aplikasi ini terdapat 1.089 *files*, 106 *folders* PHP dengan kapasitas total seluruh file adalah 17,9 Mb (18.841.600 *bytes*) dan diantaranya dapat dilihat pada Gambar 2. Kunci yang digunakan untuk membangkitkan *keystream* pada eksperimen ini adalah “alam”.



Gambar 2. *Source Code Open Source CMS Wordpress*

Salah satu contoh isi *source code* aplikasi transaksi jual beli yang asli dapat dilihat pada Gambar 3.



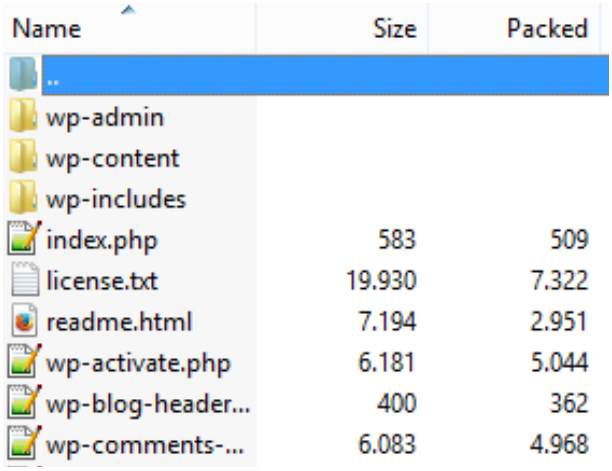
Gambar 3. *Source Code File Login.PHP*

Hasil-hasil eksperimen terhadap beberapa *source code PHP* yang dipilih pada Gambar 2 dituliskan pada bagian di bawah ini.

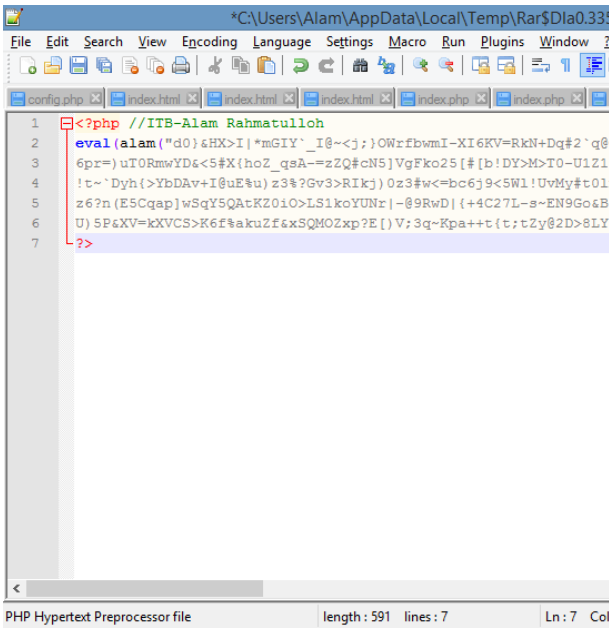
3.1. Hasil Proses Enkripsi

Gambar 4 menunjukkan seluruh *source code PHP* pada aplikasi transaksi jual beli yang telah terobfuskasi. Sedangkan Gambar 5 menunjukkan salah satu contoh isi *source code PHP* yang terobfuskasi.

a. Pengujian Perubahan Ukuran File Terobfuskasi



Gambar 4. *Source Code Terobfuskasi*



Gambar 5. *Source Code File Login.PHP Terobfuskasi*

Hasil obfuskasi pada Gambar IV.8 menghasilkan kapasitas total 18,4 Mb (19.353.600 bytes), sedangkan pada file "*index*.PHP" (Gambar IV.9) menghasilkan ukuran file 583 bytes. Dari beberapa file PHP yang dihasilkan pada eksperimen ini (Gambar IV.8 dan Gambar IV.9) menunjukan bahwa perubahan ukuran file PHP tidak terlalu signifikan.

Tabel 1. Perubahan Ukuran File Hasil Enkripsi

No	File	Ukuran (*bytes*)		
		File Asli	***File Terobfuskasi***	
1	*index.php*	418	583	+39,47%
2	*wp-activate.php*	4830	6030	+24,84%
3	*wp-blog-header.php*	271	400	+47,60%
4	*wp-config.php*	2930	4160	+41,98%
5	*wp-cron.php*	3040	3630	+19,41%
6	*wp-links-opml.php*	2320	2960	+27,59%
7	*wp-signup.php*	2450	3130	+27,76%
8	*wp-login.php*	3340	4100	+22,75%
9	*wp-mail.php*	8050	1010	+25,47%
10	*wp-settings.php*	1070	1350	+26,17%
Rata-rata				+30,30%

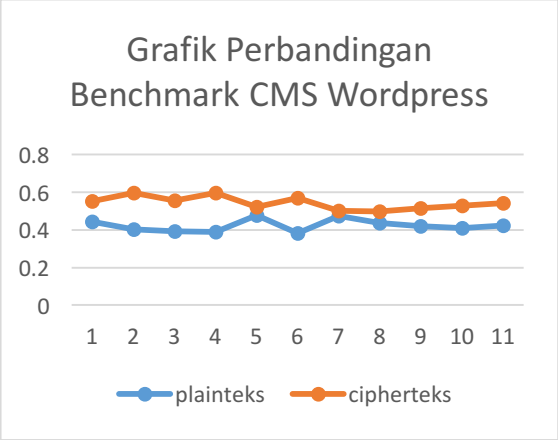
b. Pengujian Perubahan Kecepatan Eksekusi *File* Terobfuskasi

Selain pengujian pada ukuran file yang belum dan sudah terobfuskasi, dilakukan juga pengujian terhadap *respon time* eksekusi *file* yang telah terobfuskasi. Hasil pengujian dapat dilihat pada Tabel 2.

Tabel 2. Pengujian *Source Code CMS Wordpress*

Tabel 2 menunjukkan hasil percobaan waktu eksekusi pada *CMS Wordpress* sebanyak 10 kali percobaan dalam satuan *microtime/second* dengan menghasilkan nilai rata-rata untuk file asli 0,4239 dan 0,5438 untuk file terobfuskasi. Hasil perbandingan ini juga dapat dilihat berupa grafik pada Gambar 6.

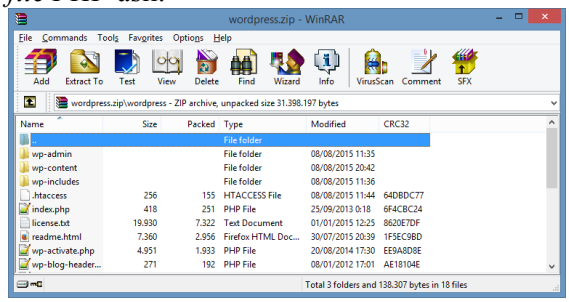
Percobaan 100 *req*	Kecepatan *(Microtime/Second)*		
	Plaintext	***Ciphertext***	***Perubahan***
1	0,444	0,551	+ 24,10 %
2	0,402	0,596	+ 48,26 %
3	0,393	0,556	+ 41,48 %
4	0,391	0,598	+ 52,94 %
5	0,479	0,522	+ 8,98 %
6	0,384	0,571	+ 48,70 %
7	0,476	0,500	+ 5,04 %
8	0,438	0,499	+ 13,93 %
9	0,421	0,516	+ 22,57 %
10	0,411	0,529	+ 28,71 %
Rata-rata	**0,4239**	**0,5438**	**+ 28,28 %**



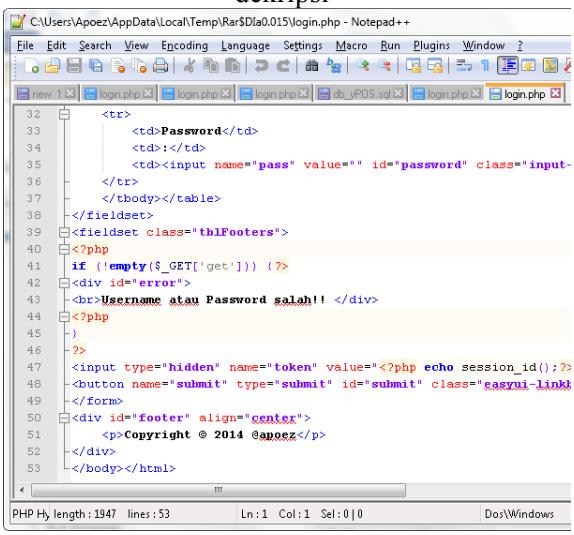
Gambar 6. *Grafik* perbandingan kecepatan eksekusi pada *CMS Wordpress*

3.2. Hasil Proses Dekripsi

Secara umum proses dekripsi pada eksperimen ini dapat dilakukan secara baik, selain itu hasil dekripsi dari seluruh *file PHP* terobfuskasi yang dihasilkan dalam eksperimen ini adalah utuh sama persis seperti *file PHP* asli.



Gambar 7. Ukuran *file* terobfuskasi hasil dekripsi

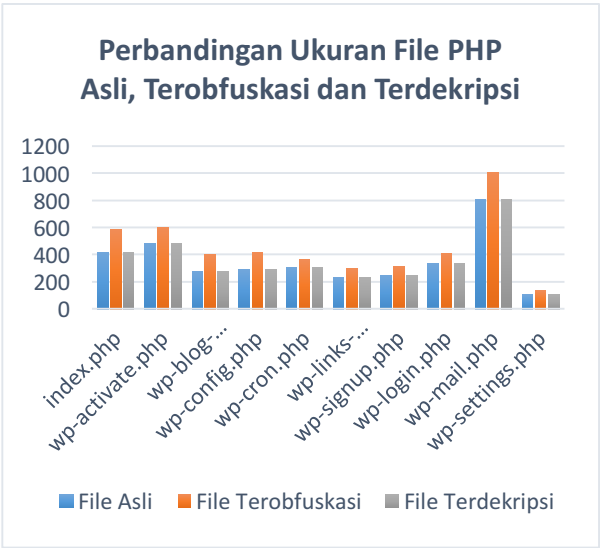


Gambar 8. Isi *file login.php* terobfuskasi hasil dekripsi

Gambar 7 dan Gambar 8 menunjukkan perubahan ukuran *file PHP* terobfuskasi hasil dekripsi. Untuk lebih lengkapnya, keseluruhan dari hasil eksperimen pada penelitian ini dapat dilihat pada Tabel 3 dan Gambar 9.

Tabel 3. Perubahan ukuran file terobfuskasi hasil dekripsi

No	File	Ukuran (*bytes*)			
		File Asli	***File Terobfuskasi***		***File Terdekripsi***
1	*index.php*	418	583	+39,47%	418
2	*wp-activate.php*	4830	6030	+24,84%	4830
3	*wp-blog-header.php*	271	400	+47,60%	271
4	*wp-config.php*	2930	4160	+41,98%	2930
5	*wp-cron.php*	3040	3630	+19,41%	3040
6	*wp-links-opml.php*	2320	2960	+27,59%	2320
7	*wp-signup.php*	2450	3130	+27,76%	2450
8	*wp-login.php*	3340	4100	+22,75%	3340
9	*wp-mail.php*	8050	1010	+25,47%	8050
10	*wp-settings.php*	1070	1350	+26,17%	1070
Rata-rata				+30,30%	



Gambar 9. Grafik Perbandingan Ukuran File PHP Asli, Terobfuskasi dan Terdekripsi

4. Kesimpulan

Sebuah usulan metode dalam proteksi *source code PHP* telah dipresentasikan. Hasil eksperimen ini menunjukkan bahwa teknik *obfuscation code* dengan menggunakan algoritma rc4 pada *library PHP* baru (*custom execution engine*) dan *encoding* 85 karakter ASCII dapat memberikan perlindungan terhadap ancaman *reverse engineering* pada

source code PHP sehingga hak kekayaan intelektual dapat terjaga.

Selain itu perubahan ukuran file dan waktu eksekusi *file* terobfuskasi tidak berbeda terlalu signifikan dari file aslinya, hal ini didasarkan pada hasil pengujian yang mampu menghasilkan perubahan ukuran file rata-rata 30,30% saja dari file aslinya dan perubahan waktu eksekusi rata-rata hanya 28,28% dari file aslinya.

Proses dekripsi dapat dilakukan secara sempurna karena pada hasil pengujian menunjukan ukuran dan waktu eksekusi file sebelum dan sesudah proses dekripsi hasilnya sama percis utuh seperti file asli.

Daftar Pustaka :

[1] C. Thomborson, "Methods for Software Protection," Keynote Address at the International Forum on Computer Science and Advanced Software Technology, 2007.

[2] D. Setiawan, Sistem Keamanan Komputer, Jakarta: PT Elex Media Komputindo, 2005.

[3] E. Eilam, Reversing: Secrets of Reverse Engineering, Indianapolis: Wiley Publishing, Inc., 2005.

[4] C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* pp. 656-715, 1949.

[5] T. B. Hansen dan J. Lengstorf, PHP for Absolute Beginners, Apress, 2014.

[6] C. Collberg dan J. Nagra, Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection, Boston: Addison-Wesley Professional, 2009.

[7] D. Ariyus, Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi, Yogyakarta: CV. Andi Offset, 2008.

[8] W. Komputer, Memahami Model Enkripsi & Security Data, Yogyakarta: Andi Publisher, 2003.

[9] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons Inc, 1996.

[10] A. L. Aprianto dan I. Winarno, "RANCANG BANGUN PHP 5 ENCODER," 2011.

[11] O. Setiawan, R. Fiati dan T. Listyorini, "Algoritma Enkripsi RC4 Sebagai Metode Obfuscation Source Code PHP," dalam *Seminar Nasional Teknologi Industri dan Informatika*, Kudus, 2014.

[12] W. Wardiana, "Pencegah Pembajakan Perangkat Lunak dengan Menggunakan Teknik Identity-Based Encryption dan Obfuscation," *INKOM,* pp. 1-2, 2009.

[13] A. Jevremović, N. Ristić dan M. Veinović, "Improving Protection of PHP Source Code using," *Nis,* pp. 16-19, 2013.

[14] Wikipedia, "ASCII," [Online]. Available: http://id.wikipedia.org/wiki/ASCII.

[15] W. Stallings, Cryptography and Network Security Principles and Practices, Fourth Edition, Prentice Hall, 2005.