PENGAMANAN SOURCE CODE PROGRAM MENGGUNAKAN ALGORITMA ADVANCED ENCRYPTION STANDARD DAN ALGORITMA BASE64

Cecep Saefudin, Gunawan Abdillah, Asri Maspupah
Jurusan Informatika Fakultas Sains dan Informatika
Universitas Jenderal Achmad Yani
Cimahi

saefudincecep@gmail.com

Abstrak—Perkembangan sistem yang terkomputerisasi dewasa ini semakin pesat dan mudah ditemukan dalam berbagai aspek kehidupan, hal tersebut dapat mempermudah pekerjaan manusia dalam membantu aktivitas konvensional kini tergantikan dengan kemudahan sistem. Tetapi seiring berkembangnya teknologi terdapat sisi negatif diantaranya terdapat berbagai ancaman kejahatan yang dapat menyerang ketika proses pengembangan perangkat lunak terjadi, diantaranya adalah pencurian data dan manipulasi data perangkat lunak yang sedang dikembangkan. Salah satu efek yang disebabkan ketika ancaman ini terjadi yaitu terjadinya kerugian bagi developer perangkat lunak yang menjadi korban seperti kerugian material. Upaya mengurangi tingkat resiko ancaman yaitu dengan menerapkan kriptografi. Kriptografi merupakan sebuah cara atau teknik yang berisi berbagai perhitungan matematika untuk mengatasi pencurian data yang dapat diaplikasikan pada sebuah perangkat lunak. Penelitian ini melakukan pengamanan terhadap data source code menggunakan kriptografi simetris algoritma Advanced Encryption Standard (AES) yang umum digunakan untuk mengenkripsi sebuah teks dan Base64 untuk pembentukan masukan kunci publik. Penelitian ini mengindikasikan peningkatan sebesar 54,03% pada 373 sample data proyek yang diuji dikarenakan avalanche effect.

Kata kunci—kriptografi, Advanced Encryption Standard, Base64, Enkripsi, Dekripsi.

I. PENDAHULUAN

Sistem yang terkomputerisasi dewasa ini semakin pesat dan mudah ditemukan dalam berbagai aspek kehidupan. Hal tersebut membuat sistem dapat mempermudah pekerjaan manusia dalam membantu aktivitas yang sebelumnya dikerjakan secara konvensional, kini telah tergantikan oleh sistem yang lebih mudah dikelola. Hal ini tidak lepas dari semakin banyaknya *developer* perangkat lunak yang mengembangkan produk besutannya dengan semakin giat dalam menunjukan berbagai inovasi yang berkaitan dari permasalahan yang ada di lapangan, terutama banyak bermunculan berbagai bahasa pemrograman tingkat tinggi yang lebih mudah digunakan dan lebih mudah diadaptasikan pada berbagai *platform* sehingga memudahkan pengembang atau *developer* untuk dapat membuat sistem yang dapat saling terkoneksi antara satu sistem dengan sistem lainnya dengan mudah dan fleksibel tanpa adanya kendala [1].

Munculnya berbagai perusahaan yang memanfaatkan pengembangan perangkat lunak, memberikan perubahan secara perlahan dari aspek pada tatanan suatu organisasi dengan kemudahan yang ditawarkan [2]. Dengan memanfaatkannya aplikasi pengembangan perangkat lunak, pengembang mendapatkan berbagai kemudahan yang diperoleh dalam membuat dan mengelola suatu pengembangan sistem perangkat lunak.

Tentunya segala kemudahan yang pengembang dapatkan tidak lepas dari sarana internet yang berperan sebagai bahan referensi dan *problem solving* dari berbagai permasalahan yang sulit diperbaiki secara mandiri. Ketika hal tersebut terjadi maka internet adalah jawaban dari setiap *developer* perangkat lunak untuk berbagai permasalahan yang membutuhkan solusi. Tanpa disadari ada berbagai kemungkinan ancaman dari luar akan masuk dan menyerang komputer pengembang, memanipulasi, ataupun mengambil data korban yang diserang untuk keuntungan pribadi bagi pelakunya.

Terdapat berbagai ancaman kejahatan yang dapat menyerang ketika proses pengembangan perangkat lunak terjadi, diantaranya adalah pencurian data dan manipulasi data perangkat lunak yang sedang dikembangkan. Contohnya saja seperti masuknya virus yang tidak terdeteksi seperti trojan, worm, fat, companion yang merupakan contoh berbagai ancaman yang dapat menyerang data pengguna.

Salah satu efek yang disebabkan ketika ancaman ini terjadi yaitu terjadinya kerugian bagi pihak pengembang yang menjadi korban seperti kerugian material bagi pengembang perangkat lunak yang terkena permasalahan manipulasi data

dan pengambilan data perangkat lunak yang dikembangkan tanpa disadari.

Disisi lain kriptografi merupakan sebuah cara atau teknik yang berisi berbagai perhitungan matematika untuk mengatasi pencurian data atau pembajakan [3] yang memungkinkan untuk diaplikasikan pada sebuah perangkat lunak. Dari penelitian terdahulu definisi aman adalah ketika sebuah dokumen atau informasi yang telah tersimpan tidak dapat dibaca oleh orang yang tidak berhak mengaksesnya secara langsung [4][5].

Algoritma Advanced Encryption Standard (AES) merupakan algoritma yang umum digunakan untuk mengenkripsi sebuah teks, algoritma ini beroperasi pada blok 128 bit atau 16 karakter yang memungkinkan untuk mengamankan sebuah berkas berisi teks yang memiliki panjang lebih dari 16 karakter [6]. Algoritma AES termasuk algoritma kriptografi simetris yang hanya memerlukan satu kunci untuk melakukan proses enkripsi dan dekripsi, sehingga waktu yang dibutuhkan untuk melakukan validasi terhadap kunci yang dipakai akan lebih cepat dibanding menggunakan dua kunci yang berbeda. Selain itu algoritma AES juga tahan menghadapi analisis sandi dan dapat secara fleksibel digunakan dalam berbagai perangkat keras dan lunak.

Penelitian terdahulu menggunakan metode Obfuscation untuk mengamankan kode program dengan melakukan transformasi dengan mengamankan isi dari kode program agar tetap terjaga [7]. Penelitian terdahulu mengenai tahapan proses enkripsi pada AES 128, transformasi terhadap state dilakukan secara berulang dalam 10 iterasi. Setiap iterasi melalui 4 transformasi dasar yaitu Subbytes, Shiftrows, Mixcolumns, dan Roundkey [6]. Penelitian sebelumnya menghasilkan bahwa data dapat disamarkan dari ancaman serangan data menggunakan algoritma Base64 [8]. Sementara penelitian lainnya mengenai perbandingan kriptografi AES dengan kriptografi DES, 3DES dan RSA dengan hasil AES lebih efisien dan optimal dalam menyandikan bit data dengan peningkatan performa dekripsi sebesar 13,085% dalam hal kecepatan, waktu, dan hasil [9].

Penelitian ini membangun sistem yang dapat mengamankan *source code* program dari tindakan manipulasi, pembajakan, dan illegal akses pada *source code* perangkat lunak, sehingga diharapkan dapat membantu *developer* perangkat lunak dalam melindungi data sensitif di dalam *source code* yang dikembangkan. Pembangunan sistem menggunakan metode kriptografi algoritma AES dan algoritma Base64 dengan cara melakukan enkripsi pada setiap berkas yang berisi *source code* yang terdaftar dalam kode American Standard Code for Information Interchange (ASCII) tanpa merusak makna yang tersebunyi di dalam hasil enkripsi.

Algoritma Base64 cocok digunakan sebagai algoritma pembentukan kunci publik yang akan digunakan pada proses enkripsi dan dekripsi data. Dikarenakan sifat algoritma ini yang dapat melakukan penyandian terhadap berbagai jenis data ke dalam format *ciphertext*, sehingga dapat digunakan untuk meningkatkan keamanan pada kunci publik.

II. LANDASAN TEORI

A. Source Code

Source code adalah kumpulan pernyataan atau deklarasi bahasa pemrograman komputer yang ditulis dan dapat dibaca manusia. *Source code* memungkinkan programmer untuk berkomunikasi dengan komputer menggunakan beberapa perintah yang telah terdefinisi. *Source code* merupakan sebuah program yang biasanya dibuat dalam satu atau lebih berkas teks, disimpan dalam basis data yang terdapat pada prosedur.

B. Kriptografi

Keamanan adalah hal penting yang sangat diinginkan setiap orang, tidak sedikit seseorang yang sangat sensitif terhadap hal mengenai keamanan. Pada penelitian sebelumnya dijelaskan bahwa kriptografi merupakan sebuah bidang ilmu yang mempelajari tentang cara mengamankan data atau informasi dengan cara menyandikan atau membuatnya menjadi tidak dapat dibaca kembali ataupun dimengerti kembali maknanya agar aman dan kebal dari serangan [10].

Sehingga teknik kriptografi ini memiliki aspek yang sangat penting dalam menjaga kerahasiaan data yang diamankan, dikarenakan aspek otentikasi, dan integritas data terpenuhi menggunakan metode ini. Aspek tersebut berguna bagi orang yang menginginkan keamanan bagi informasi atau data penting yang ingin diamankan.

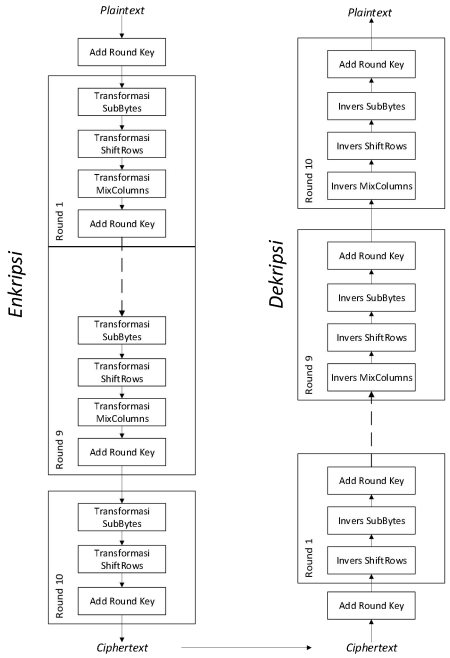
C. Enkripsi dan Dekripsi

Enkripsi adalah proses penyandian sebuah teks dari yang memiliki makna dan masih bisa dibaca atau disebut plaintext, menjadi sebuah pesan yang tidak dapat dikenali lagi maknanya atau disebut ciphertext. [11].

Dekripsi merupakan sebuah cara mengembalikan kembali semua transformasi dasar algoritma enkripsi kembali menjadi kode ASCII yang dapat dibaca dan memiliki makna [6].

D. Algoritma Advanced Encryption Standard

Advanced Encryption Standard (AES) merupakan sebuah algoritma simetris, yakni menggunakan kunci yang sama saat melakukan proses enkripsi dan dekripsi pada setiap bit [12]. Proses pengamanan menggunakan AES dapat dilihat pada Gambar 1.



Gambar 1. Proses algoritma AES

Algoritma AES dapat digunakan untuk mengenkripsi sebuah dokumen atau berkas berisi teks dengan cara melakukan proses enkripsi perblok secara pararel [6]. Algoritma AES memiliki variasi ukuran blok dan variasi kunci lainnya yaitu 192 dan 256 bit [13]. Jumlah setiap variasi bit mempengaruhi jumlah perputaran, perputaran ini disebut Round Function, Round Function adalah proses transformasi Subbytes, Shiftrows, Mixcolumns dan penambahan Roundkey secara berulang sebanyak jumlah putaran [13].

1) Transformasi Subbytes

Subbytes merupakan sebuah proses yang menukarkan isi pada byte dengan menggunakan tabel subtitusi S-Box seperti Gambar 2 berikut.

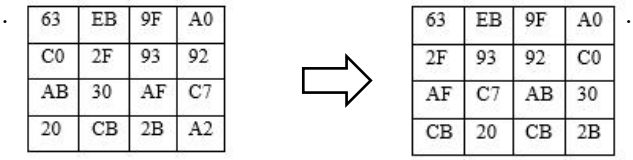
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Gambar 2. S-Box

Proses subtitusi dilakukan dengan menggunakan setiap byte pada larik state, maka hasil nilai subtitusi merupakan perpotongan dari baris x dan kolom y.

2) Transformasi Shiftrows

Adalah proses pergeseran bit, memindahkan posisi bit yang berada dikiri berubah posisi menjadi paling kanan seperti pada Gambar 3.

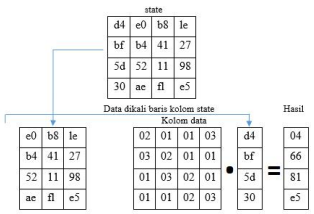


Gambar 3. Transformasi Shiftrows

Dilakukan dengan menggeser baris ke dua sebanyak satu kali, baris ke tiga digeser dua kali dan baris ke empat digeser tiga kali.

3) Transformasi Mixcolumns

Merupakan proses mengalikan data dari setiap kolom larik state dengan matriks hasil transformasi Shiftrows seperti pada Gambar 4.



Gambar 4. Transformasi Mixcolumns

Setaip kolom state dilakukan operasi XOR dengan nilai konstanta pada kolom data hingga dihasilkan matriks dari operasi Mixcolumns.

4) Roundkey

Merupakan proses penambahan Roundkey pada state hasil dari Mixcolumns menggunakan operasi XOR. Setiap Roundkey tersebut kemudian dijumlahkan dengan kolom yang bersesuaian dari state. Langkahnya yakni dengan mengorganisir menjadi empat word (w) pada empat elemen (w[0], w[1], w[2], w[3]).

1. Jika i habis dibagi 4, lakukan w[i]= f (t,i) $\oplus$ w[i-4] dengan fungsi f(t,i)

f(t,i) = Subword (rotword(t)) $\oplus$ RC[i/4] (1)

2. Jika i tidak sama dengan 0, maka lakukan w[i]= t $\oplus$ w[i-4] (2)

E. Algoritma Base64

Algoritma Base64 merupakan algoritma yang digunakan untuk melakukan *encoding* atau menyandikan data biner, skema *encoding* dengan algoritma Base64 yang biasanya digunakan ketika ada kebutuhan untuk menyandikan data biner yang perlu disimpan dan ditransfer [14]. Base64 memungkinkan pengguna untuk mengolah berkas jenis apapun menjadi data berbentuk teks [15].

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	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	/

Gambar 5. Indeks Base64

Karakter yang terbentuk dari hasil penyandian menggunakan algoritma Base64 terdiri dari A...Z, a...z dan 0..9 serta penambahan simbol lain yakni "+" dan simbol "/" serta karakter "=" pada dua karakter terakhir yang digunakan untuk penyesuaian dan melakukan penggenapan pada binary [8].

F. Program Interpreter

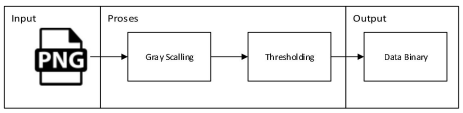
Dalam ilmu komputer, penerjemah atau lebih dikenal dengan interpreter merupakan perangkat lunak yang berfungsi melakukan eksekusi *source code* secara langsung yang ditulis dalam suatu bahasa pemrograman tanpa terlebih dahulu menyusunnya menjadi program bahasa mesin.

III. METODE

Penelitian mengenai pengamanan *source code* program menggunakan dua algoritma, yakni menggunakan algoritma Base64 dan algoritma AES. Untuk melakukan enkripsi serta dekripsi menggunakan algoritma AES, sedangkan Base64 digunakan untuk pembentukan kunci publik yang dipakai AES.

A. Algoritma Base64

Data masukkan pembentuk kunci publik menggunakan berkas gambar berekstensi .JPG atau .PNG berukuran kurang dari 200 KB. Gambar tersebut akan diambil stream berupa biner, lalu dilakukan proses konversi data biner kedalam bentuk hexadesimal. Hasil dari konversi tersebut akan disandingkan dengan tabel indeks Base64 yang tersimpan dalam database sistem.



Gambar 6. Proses mengambil data binary dari gambar

Gray scalling dilakukan untuk mengubah citra dari RGB menjadi citra keabuan, kemudian dilakukan proses thresholding untuk mengubah citra keabuan menjadi citra hitam atau putih dengan memberikan kondisi bila RGB citra kurang dari 128 maka dimasukan kedalam warna hitam atau 0 dan lebih atau sama dengan 128 dimasukan kedalam kategori warna putih atau 1. Hasil binary tersebut dapat langsung diolah pada algoritma Base64 untuk disandikan seperti pada Gambar 6. Diasumsikan contoh data yang akan dilakukan *encoding* masih berjenis teks.

Plaintext: SCDM

TABEL I. PROSES PENCARIAN NILAI DESIMAL

Index	1	2	3	4
Char	S	C	D	M
Desimal	83	67	68	77

Proses pada Base64 akan mengubah setiap masukan menjadi desimal lalu menggabungkan kembali menjadi biner yang akan dibagi setiap 6 bit, kemudian diubah menjadi desimal lalu dilakukan pencarian pada tabel indeks Base64 seperti pada TABEL I dan TABEL II.

Index 1 => ASCII = 83 (0101 0011)

Index 4 => ASCII = 77 (0100 1101)

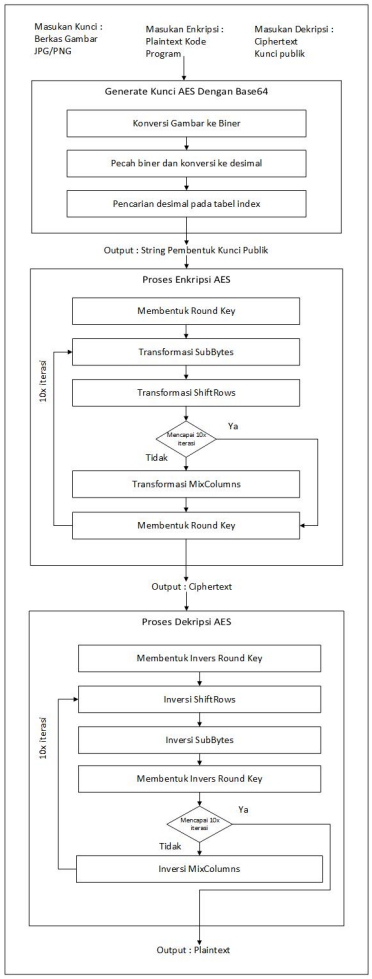
TABEL II. PROSES PENCARIAN BASE64

Index	Binary per 6 bit	Char	ASCII
1	010100	20	U
2	110100	52	0
3	001101	13	N
4	000100	4	E
5	010001	21	V
6	001101	15	P

Sehingga pada proses dari algoritma Base64 menghasilkan nilai yakni binary 01010011010000110100010001001101

B. Algoritma Advanced Encryption Standard

Pada algoritma Advanced Encryption Standard (AES) data yang akan dienkripsi melewati beberapa proses



Gambar 7. Sistem Pengamanan Source Code Program

Proses yang dilakukan menggunakan algoritma AES yakni, proses pembentukan Roundkey, pembentukan kunci publik, transformasi Subbytes, transformasi Shiftrows, transformasi Mixcolumn, sedangkan pada proses dekripsinya dilakukan proses invers pada data *ciphertext*. Proses dekripsi pada AES melewati beberapa tahap yakni, invers Shiftrows, invers Subbytes, dan invers Mixcolumns seperti pada Gambar 7.

Tahapan algoritma AES pada Gambar 7 dilakukan sebanyak sepuluh kali iterasi, pada iterasi kesepuluh transformasi Mixcolumns tidak dimasukan. Ketentuan tersebut berlaku pula saat iterasi proses dekripsi, dimana transformasi invers Mixcolumns tidak dimasukan pada iterasi pertama.

1) Proses Enkripsi

Proses enkripsi dilakukan dengan mengubah setiap karakter plaintext yang akan dienkripsi diorganisir manjadi state yang bernilai hexadesimal. Berikut merupakan tahapan dalam enkripsi pada algoritma AES.

a) Pembentukan Roundkey

Pembentukan kunci yang terbagi menjadi dua proses yakni, pembentukan Roundkey dan pembentukan kunci publik. Contoh data yang akan diamankan yakni sebuah *plaintext* pada tabel ASCII yang diubah kedalam hexadesimal. Berikut merupakan contoh rangkaian karakter ASCII yang diasumsikan untuk diamankan seperti pada Gambar 8 dan Gambar 9.

Plaintext: IF SCDM Cecep S.

I	F		S	C	D	M		C	e	c	e	p		S	.
49	46	20	53	43	44	4d	20	43	65	63	65	70	20	53	2e

Gambar 8. Contoh rangkaian plaintext

Cipherkey : Plainkey AES128.

P	l	a	i	n	K	e	y		A	E	S	1	2	8	.
50	6c	61	69	63	4b	65	79	20	41	45	53	31	32	28	2e

Gambar 9. Contoh rangkaian plainkey

Maka terbentuk sebuah roundkey pertama *Cipherkey* dalam hexadesimal : 50 6C 61 69 63 4B 65 79 20 41 45 53 31 32 28 2E

Lalu dilakukan proses pembagian setiap nilai hexa ke dalam masing masing *word* atau w[i] yang masing-masing *word* berisi 4 byte data.
w[0]=(50,6C,61,69), w[1]=(63,4B,65,79), w[2]= (20,41,45,53), w[3] = (31,32,28,2E). Setelah itu dilakukan pergeseran byte satu kali ke kiri pada w[3]: (32,28,2E,31)

Lalu dilakukan proses subsitusi byte w[3] dengan tabel S-Box : (23,34,31,C7). Dikurangi dengan konstanta sesuai aturan pada AES yakni (01,00,00,00) sehingga menghasilkan (w[3]) = (22,34,31,C7).
Mencari w[4] = w[0] $\oplus$ (w[3]) = (B2,5E,9D,98)

TABEL III. PROSES PENCARIAN ROUNDKEY

0101 0000	0110 1100	0110 0001	0110 1001
1110 0010	0011 0010	1111 1100	1111 0001
1011 0010	0101 1110	1001 1101	1001 1000
B2	5E	9D	98

Setelah dilakukan operasi XOR seperti pada TABEL III untuk mencari w[4]. Maka dilakukan kembali pencarian w[5] sebanyak w[n].

w[5] = w[4] $\oplus$ w[1] = (D1,15,F8,E1), w[6] = w[5] $\oplus$ w[2] = (F1,54,BD,B2), w[7] = w[6] $\oplus$ w[3] = (C0,66,95,9C)

Maka dari operasi pencarian Roundkey tersebut didapatkan hasil yakni: B2 5E 9D 98 D1 15 F8 E4 F154 BD B7 C0 66 95 9C. Proses pencarian Roundkey akan terus diulangi hingga iterasi kesepuluh.

R0: 50 6C 61 69 63 4B 65 79 20 41 45 53 31 32 28 2E

R1: B2 5E 9D 98 D1 15 F8 E4 F1 54 BD B7 C0 66 95 9C

…
R10: 6F 6 46 9C d3 39 E2 66 CF bA 52 37 B4 97 B2 2D

b) Pembentukan Kunci Publik

Penyandian data dengan algoritma Base64 dibuat dengan mengubah serangkaian bit menjadi bilangan desimal melalui proses pembagian tiap bit.

$$\begin{bmatrix} 49 & 43 & 43 & 70 \\ 46 & 44 & 65 & 20 \\ 20 & 4D & 63 & 53 \\ 53 & 20 & 65 & 3E \end{bmatrix} \oplus \begin{bmatrix} 50 & 63 & 20 & 31 \\ 6C & 4B & 41 & 32 \\ 61 & 65 & 45 & 28 \\ 69 & 79 & 53 & 2E \end{bmatrix} = \begin{bmatrix} 19 & 20 & 63 & 41 \\ 2A & 0F & 24 & 12 \\ 41 & 28 & 26 & 7B \\ 3A & 59 & 36 & 10 \end{bmatrix}$$

Gambar 10. Pembentukan state

Proses pembentukan kunci publik dilakukan dengan melakukan operasi XOR pada plaintext dan Roundkey yang didapatkan seperti pada Gambar 10.

c) Subbytes

Transformasi Subbytes dilakukan dengan memetakan setiap byte dari larik state dengan menggunakan tabel S-Box.

$$\begin{bmatrix} 19 & 20 & 63 & 41 \\ 2A & 0F & 24 & 12 \\ 41 & 28 & 26 & 7B \\ 3A & 59 & 36 & 10 \end{bmatrix} = \begin{bmatrix} D4 & B7 & FB & 83 \\ E5 & 76 & 36 & C9 \\ 83 & 34 & F7 & 21 \\ 80 & CB & 05 & CA \end{bmatrix}$$

Gambar 11. Operasi Subbytes

Transformasi Shiftrows dilakukan dengan mencari titik potong pada tabel S-Box yang dapat dilihat pada Gambar 11.

d) Shiftrows

Transformasi Shiftrows melakukan pergeseran bergantung pada nilai baris.

$$\begin{bmatrix} B7 & FB & 83 & D4 \\ 76 & 36 & C9 & E5 \\ F7 & 21 & 83 & 34 \\ 21 & 83 & 34 & F7 \end{bmatrix}$$

Gambar 12. Operasi Shiftrows

Baris kesatu digeser sejauh satu kali kekiri, baris dua digeser sejauh dua kali ke kiri, dan baris tiga digeser sejauh tiga kali kekiri seperti pada Gambar 12.

e) Mixcolumns

Transformasi Mixcolumns mengalikan setiap kolom dari larik state seperti pada Gambar 13 dengan konstanta matriks, blok *plaintext* disimpan di dalam matriks yang bernama state berukuran 4x4. Matriks dikiri adalah konstanta dan matriks yang akan dikalikan adalah hasil dari operasi Shiftrows sebelumnya.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} B7 & FB & 83 & D4 \\ FB & 36 & C9 & E5 \\ F7 & 21 & 83 & 34 \\ 21 & 83 & 34 & F7 \end{bmatrix} = \begin{bmatrix} 2E & 26 & 48 & 28 \\ 2E & 74 & F8 & 06 \\ FD & 19 & D3 & 26 \\ 56 & 9A & 7A & 4C \end{bmatrix}$$

Gambar 13. Operasi Mixcolumns

Contoh proses perhitungan untuk mencari isi matriks 2E dengan cara melakukan operasi baris dikalikan dengan kolom lalu dilakukan operasi XOR.

(02 • B7) ⊕ (03 • FB) ⊕ (01 • F7) ⊕ (01 • 21)

(02 x B7) = (02 • 1011 0111) => 0110 1110 ⊕ 0001 1011

= 0111 0101

(03 x FB) = (03 x 0111 0110) => 1110 1100 ⊕ 0001 1011

= 1111 0111

(01 x F7) = (01 x 1111 0111) => 1110 1110 ⊕ 0001 1011

= 1111 0101

(01 x 21) = (01 x 0010 0001) => 0100 0010 ⊕ 0001 1011

= 0101 1001

Sehingga dihasilkan nilai yakni:

0111 0101⊕1111 0111⊕1111 0101⊕0101 1001

= 0010 1110 (2E)

Selanjutnya hasil dari proses Mixcolums akan dilakukan operasi XOR kembali dengan Roundkey pertama yang didapatkan pada proses pembentukan kunci sebelumnya seperti pada Gambar 14.

$$\begin{bmatrix} 2E & 26 & 48 & 28 \\ 2E & 74 & F8 & 06 \\ FD & 19 & D3 & 26 \\ 56 & 9A & 7A & 4C \end{bmatrix} \oplus \begin{bmatrix} B2 & D1 & F1 & C0 \\ 5E & 15 & 54 & 66 \\ 9D & F8 & BD & 95 \\ 98 & E4 & B7 & 9C \end{bmatrix} = \begin{bmatrix} 9C & F7 & B9 & E8 \\ 70 & 61 & AC & 60 \\ 60 & E1 & 6E & B3 \\ CE & 7E & CD & D0 \end{bmatrix}$$

Gambar 14. Contoh rangkaian plainkey

Sehingga didapatkan hasil round 1: 9C 70 60 CE F7 61 E1 7E B9 AC 6E CD E8 60 B3 D0. Proses dilakukan hingga iterasi kesepuluh dengan cara yang sama, pada iterasi kesepuluh, ciphertext yang dihasilkan yakni 33 35 29 47 9B 46 56 85 88 0A 28 AC CA 64 4A B6.

2) Proses Dekripsi

Proses dekripsi algoritma AES dilakukan dengan melakukan invers pada proses transformasi algoritma AES dengan iterasi sebanyak sepuluh kali.

a) Invers Shifrows

Invers shiftrows merupakan transformasi yang berkebalikan dengan transformasi shiftrows. Pada transformasi ini dilakukan pergeseran bit ke kanan berkebalikan dengan transformasi shiftrows yang melakukan pergeseran bit ke kiri.

b) Invers Subbytes

Invers subbytes merupakan transformasi yang berkebalikan dengan transformasi subbytes. Pada proses ini setiap elemen pada state dipetakan dengan menggunakan table invers S-Box.

c) Invers Mixcolumns

Pada transformasi ini setiap kolom state dikalikan dengan matriks, blok ciphertext disimpan di dalam matriks yang bernama state berukuran 4x4.

$$\begin{bmatrix} 14 & 11 & 13 & 09 \\ 09 & 14 & 11 & 13 \\ 13 & 09 & 14 & 11 \\ 11 & 13 & 09 & 14 \end{bmatrix}$$

Gambar 15. Konstanta invers Mixcolumns

Transformasi invers mixcolumns mengalikan setiap kolom dari larik state. Proses invers Mixcolumns serupa dengan transformasi Mixcolumns, namun dengan perbedaan konstanta matriks yang digunakan sebagai pengali dengan state seperti pada Gambar 15.

IV. HASIL DAN DISKUSI

Perolehan data dilakukan melalui observasi pada situs *repository* Github yang berisi berbagai proyek pengembangan perangkat lunak yang menggunakan sistem Git dan layanan *hosting* internet. Hal ini banyak digunakan pada pengembangan perangkat lunak dikarenakan kelebihannya yang memiliki fitur seperti kolaborasi dan *debugging*. Data yang digunakan yakni *sample* proyek perangkat lunak dari beberapa *platform*.

Pengujian sistem yang dilakukan terdiri dari uji pengaruh terhadap algoritma yang digunakan yaitu, uji engine AES, uji engine Base64, uji kecepatan enkripsi berkas, uji kecepatan dekripsi berkas, uji ukuran data, dan uji integritas data. Pada tahap ini dilakukan analisis dari setiap parameter yang diuji pada sistem pengamanan *source code* program, dimana setiap parameter memiliki skenario pengujian yang telah ditentukan untuk mendapatkan hasil pengujian yang diharapkan.

A. Uji Engine AES

Percobaan dilakukan dengan melakukan proses enkripsi dan dekripsi terhadap *sample* data yakni berkas *source code* Hypertext Preprocessor (PHP) menggunakan *framework* CodeIgniter. *Sample* data *source code* yang akan diamankan memiliki data sebanyak 373 berkas dengan total ukuran yakni 13,2 MB. Hal ini dilakukan untuk mengetahui pengaruh enkripsi terhadap *plaintext*, serta pengaruh *ciphertext* jika dilakukan proses dekripsi terhadap data yang telah diamankan. Pada Gambar 16 dapat dilihat *sample* data *source code* yang masih berbentuk *plaintext*.

absensi.php × Settings

d: ▸ SAMPLE ▸ WEB ▸ CI PHP ▸ absensikaryawan ▸ application ▸ controllers ▸ absensi.php

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Absensi extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->_cek_login();
    }

    //CEK LOGIN
    private function _cek_login()
    {
        if(!$this->session->userdata('useradmin')){
            redirect(base_url().'backend');
        }
    }

    //HALAMAN ABSENSI
    public function index()
    {
        $data = array(
            'data_absensi' => $this->model->GetKaryawanJabAbs("order by id_abs desc")->result_array(),
            'nama' => $this->session->userdata('nama'),
        );

        $this->load->view('absensi/data_absensi', $data);
    }
}
```

Gambar 16. Plaintext berkas source code sebelum dilakukan enkripsi

Hasil setelah dilakukan enkripsi dihasilkan keluaran berupa *ciphertext* seperti pada Gambar 17, hasil enkripsi berisi rangkaian data yang sudah tidak dikenali maknanya.

absensi.php D:\...\absensikaryawan\... absensi.php C:\...\hasil enkripsi\... ●

c: ▸ Users ▸ Cecep Saefudin ▸ Desktop ▸ hasil enkripsi ▸ application ▸ controllers ▸ absensi.php

```
SHg8+54oxwUcjj3mNDrZCgH4hiahyTxxsQ0HT30SNqKcA+uQ0naeOJTv1LN9/
PNDRX+zJLWsYm+pYGlGJ9P+ohpbVpGNvM8XSCAQLWRmLpRiubWEaYNjnQ1fLX
TR9jeZeIR0gVrL4P/D1Q8oBgSM/wgzV6eryMSr/GQRBUOV8jswt4WhK3Z/ovd
PLiqg4tKm8JZ4KgKKJ8lDJPTW/+eIlhXYdeMpp9gNc1x3CglPRT7d5jnNz8lQ
JWwPgGp6DMINUhwbX14gQJI3u9Z6rdlyoe0dSaUxMfK6exx1ohtg1whWpJ7d2
LQaJaDPLtjvja/ffptCbIOLi9ryQuS2ve2OyiAhhHoohznpZB1jotdY4C8JLk
vbrdEvp3pbR00Yq7dod9QxtfNw2C2xsoldTEOC0HrcVnAGdS0nwX2MkdavDw5
AybcgmXwfC3fqqTH5AJR0PbAfdto+T+NO44FBC3iZ+NPgw5nRhsRWwdbWOaj4
Rr2/71FtQMbtdRO/FyfZ4zTtGg9n2ziYcIk5gcLeLb361vDn2FVUhKHtHheCM
t0hKPQbu4JeZHsRTqVIArR7/IKMW0hbjxm9kLbNPynFSv3MIgp/OLa4iGhGWb
VGNlLoWBjOn1iTkAezaQ96Jqzilurkf4IHrEBZn2z/bawqjWmSz9SHygroKEZ
rumDX8Qc0A2V4XbfdzudMJuHlb9nFbgWEc8oLTM92b3S5pybYOi1Ly3B0rbg1
f9TG3+GyCBdLNyzePiIGqKMULTQjxil/+NmCA+RutxlS5hjZFjQrLsTVGrWNs
DFi2sML9GwsCTueMf9er+PRSl2GbSmuRzdN9BwwHwzeZmW9jig/A+go4q2iZH
45XnUJWH2viZjJSgO1gb4=
```

Gambar 17. Hasil enkripsi terhadap source code

Berdasarkan hasil pengujian terhadap engine AES, proses enkripsi menggunakan algoritma AES berhasil mengubah rangkaian *plaintext* berisi *source code* menjadi karakter yang tidak dapat dikenali kembali. Proses dekripsi menggunakan algoritma AES juga berhasil mengembalikan data yang tidak dapat dipahami kembali menjadi *plaintext* awal berisi *source code* pada Gambar 18.

absensi.php D:\...\absensikaryawan\... absensi.php C:\...\hasil enkripsi\... ● absensi.php C:\...\result_dekripsi\... ×

c: ▸ Users ▸ Cecep Saefudin ▸ Desktop ▸ result_dekripsi ▸ application ▸ controllers ▸ absensi.php

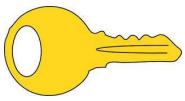
```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Absensi extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->_cek_login();
    }

    //CEK LOGIN
    private function _cek_login()
    {
        if(!$this->session->userdata('useradmin')){
            redirect(base_url().'backend');
        }
    }

    //HALAMAN ABSENSI
    public function index()
    {
        $data = array(
            'data_absensi' => $this->model->GetKaryawanJabAbs("order by id_abs desc")->result_array(),
            'nama' => $this->session->userdata('nama'),
        );

        $this->load->view('absensi/data_absensi', $data);
    }
}
```

Gambar 18. Hasil dekripsi terhadap source code

Oleh karena itu dapat disimpulkan engine AES dapat digunakan untuk melakukan enkripsi dan dekripsi terhadap rangkaian *source code* dengan pemakaian kunci publik yang sama.

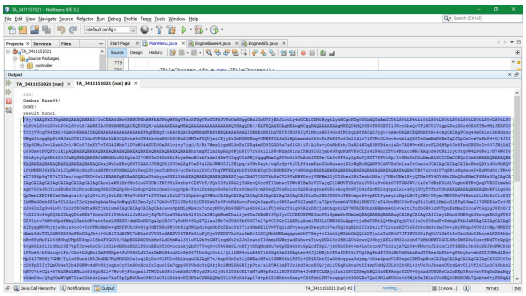
B. Uji Engine Base64

Percobaan dilakukan dengan melakukan proses *encoding* terhadap berkas gambar yang akan digunakan sebagai masukan pembentuk kunci publik pada proses enkripsi maupun dekripsi. Dengan menerapkan standar untuk berkas gambar yang dapat diproses oleh sistem yakni berkas gambar dengan ukuran kurang dari 200 KB dengan format JPG atau PNG. Percobaan dilakukan dengan menggunakan berkas gambar berformat JPG dengan ukuran berkas yakni 7,74 KB dengan dimensi sebesar 316px x 159px. Hal ini dilakukan untuk mengetahui pengaruh proses *encoding* menggunakan engine Base64.



Gambar 19. Contoh berkas gambar masukan Base64

Berdasarkan hasil pengujian terhadap engine Base64, proses pembentukan kunci publik dilakukan menggunakan berkas pada Gambar 19 dengan ukuran berkas kurang dari 200 KB menghabiskan waktu 5,86 detik. Kemudian dilakukan pengujian ulang pada berkas gambar berukuran 6 MB dengan dimensi 6010px x 4012px yang menghasilkan waktu selama 53,73 detik, sehingga besar ukuran berkas gambar dapat mempengaruhi waktu yang diperlukan untuk menghasilkan masukan kunci publik.

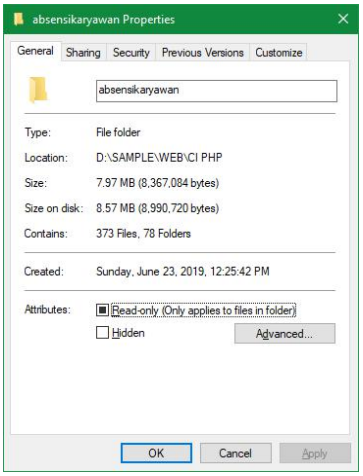


Gambar 20.. Hasil pembentukan kunci dengan Base64

Hasil uji yang dilakukan pada Gambar 19 menghasilkan data yang tidak dapat dikenali pada Gambar 20. Data ini dapat digunakan sebagai masukan untuk kunci publik pada engine AES, dikarenakan hasil *encoding* yang bertipe string dapat digunakan sebagai parameter pada engine AES. Oleh karena itu dapat disimpulkan bahwa engine Base64 dapat digunakan untuk pembentukan masukan kunci publik serta dapat meningkatkan tingkat keamanan kunci yang digunakan untuk enkripsi serta dekripsi data.

C. Uji Kecepatan Enkripsi Berkas

Percobaan dilakukan dengan melakukan perhitungan waktu yang diperlukan sistem saat melakukan enkripsi terhadap berkas yang akan diamankan hingga berkas tersebut berhasil dilakukan enkripsi. Percobaan dilakukan dengan menggunakan sample data *source code framework* CodeIgniter dengan data sebanyak 373 berkas dengan total ukuran yakni 8,57 MB yang terlihat pada Gambar 21.

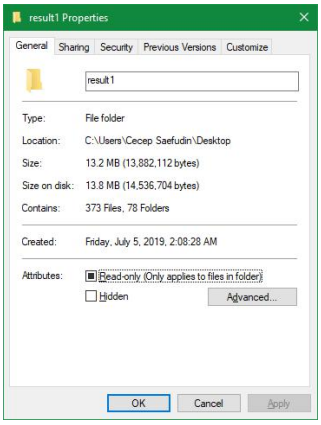


Gambar 21. Detail direktori berkas sebelum dilakukan enkripsi

Berdasarkan hasil pengujian terhadap kecepatan yang dibutuhkan sistem saat melakukan enkripsi. Perhitungan waktu yang telah dilakukan menunjukan hasil bahwa dengan jumlah data sebanyak 373 berkas yang ada pada direktori sampel data yang diuji, waktu yang diperlukan yakni 2,34 menit untuk mengamankan seluruh berkas yang diuji. Dilakukan pula pengujian ulang dengan menggunakan sampel *source code* Java, dengan ukuran berkas sebesar 208 KB menghabiskan waktu selama 14,52 detik.

D. Uji Kecepatan Dekripsi Berkas

Percobaan dilakukan dengan melakukan perhitungan waktu yang diperlukan sistem saat melakukan dekripsi terhadap berkas yang telah diamankan hingga berkas tersebut dapat dikenali kembali maknanya. Percobaan dilakukan dengan menggunakan *sample* data hasil enkripsi berisi *source code framework* CodeIgniter dengan data sebanyak 373 berkas dengan total ukuran setelah dilakukan enkripsi yakni 13.2 MB.



Gambar 22. Detail direktori berkas setelah dilakukan enkripsi

Berdasarkan hasil pengujian terhadap kecepatan yang dibutuhkan sistem saat melakukan dekripsi. Perhitungan waktu yang telah dilakukan menunjukan hasil bahwa dengan jumlah data sebanyak 373 berkas yang ada pada direktori sampel data yang diuji seperti pada Gambar 22, waktu yang diperlukan untuk melakukan dekripsi adalah 24,08 detik untuk

mengembalikan seluruh data kembali seperti semula. Dilakukan pula pengujian ulang pada *source code* Java yang telah diamankan menghasilkan waktu yakni 8,71 detik. Oleh karena itu dapat disimpulkan bahwa proses enkripsi memakan waktu yang lebih lama dibanding proses dekripsi, selisih waktu yang dihasilkan yakni mencapai 2,10 menit pada *framework* CodeIgniter dan 5,81 detik pada *source code* Java.

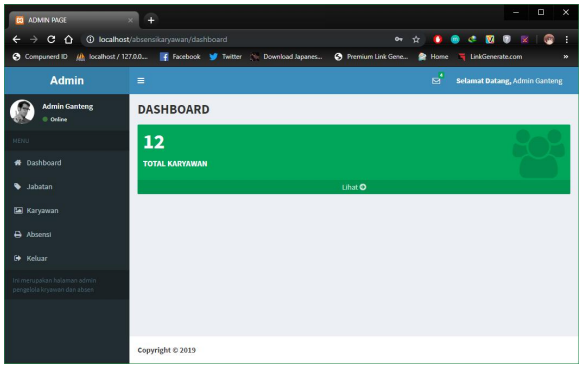
E. Uji Ukuran Data

Percobaan dilakukan dengan melakukan perbandingan ukuran data antara berkas yang belum dilakukan enkripsi dengan berkas yang telah dilakukan enkripsi, pengujian ini dilakukan untuk melihat efek bertambahnya ukuran data setelah proses pengamanan dilakukan.

Berdasarkan hasil pengujian terhadap ukuran data yang diamankan, Gambar 21 menunjukan berkas *source code* dengan *framework* CodeIgniter berukuran 8,57 MB jika dibandingkan dengan berkas yang telah dilakukan enkripsi pada Gambar 22 terjadi perubahan. Perubahan yang terjadi yakni pada ukuran data setelah dilakukan enkripsi bertambah 4,63 MB dari data aslinya, total ukuran data setelah dilakukan enkripsi yakni sebesar 13,2 MB. Pengujian ulang juga dilakukan pada *source code* menggunakan Java dengan ukuran sebesar 208 KB, meningkat 64 KB dari data aslinya menjadi sebesar 272 KB. Oleh karena itu dapat disimpulkan bahwa proses enkripsi pada rangkain *source code* meningkatkan ukuran data hingga 54,03% pada *sample* proyek menggunakan *framework* CodeIginter dan pada pengujian ulang menggunakan Java meningkat 30,7% dari data asli sebelum dilakukan enkripsi.

F. Uji Integritas Data

Percobaan dilakukan dengan menjalankan kembali sampel data yang telah dilakukan proses dekripsi untuk menguji apakah perangkat lunak masih dapat berjalan ketika dilakukan dekripsi terhadap data. Dilakukan pula pengujian untuk membandingkan jumlah karakter pada berkas yang belum dilakukan proses enkripsi dengan berkas yang telah dilakukan proses dekripsi, pengujian ini dilakukan untuk melihat efek terhadap integritas data yang telah diamankan.

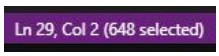


Gambar 23. Sample data hasil dekripsi ketika dijalankan ulang

Berdasarkan hasil pengujian terhadap *sample* data menggunakan *framework* CodeIgniter yang dilakukan, pengujian menghasilkan bahwa berkas data yang berisi *source code* program berhasil dijalankan ulang tanpa ada kesalahan pada proyek perangkat lunak yang pernah diamankan yang dapat dilihat pada Gambar 23.

Gambar 24. Total karakter sebelum dilakukan enkripsi

Lalu dilakukan kembali pengujian terhadap integritas data dengan melakukan pengecekan terhadap jumlah karakter pada sample data *source code* sebelum dilakukan enkripsi pada Gambar 24 dan data setelah dekripsi pada Gambar 25.



Gambar 25. Jumlah karakter setelah dilakukan dekripsi

Berdasarkan hasil pengujian dengan mengambil sampel salah satu berkas pada direktori *source code* perangkat lunak yang diuji. Terdapat kesamaan jumlah karakter antara data setelah dilakukan dekripsi pada Gambar 25 dengan total sebanyak 648 karakter dengan data sebelum dilakukan enkripsi pada Gambar 24. Oleh karena itu dapat disimpulkan bahwa proses enkripsi terhadap data berisi *source code* program tidak merubah isi dan makna pada rangkaian *source code* yang diamankan.

V. KESIMPULAN

Penelitian ini telah menghasilkan sebuah sistem pengamanan *source code* perangkat lunak menggunakan algoritma Advanced Encryption Standard (AES) dan algoritma Base64. Sistem ini telah diuji melalui pengujian perangkat lunak dan pengujian sistem.

Sistem pengamanan *source code* program ini terdiri dari tiga proses utama. Proses pertama yakni pembentukan masukan kunci publik, proses kedua yakni proses enkripsi terhadap rangkaian *source code*, dan proses terakhir yakni proses dekripsi pada data *ciphertext*. Pada pengujian kecepatan enkripsi dan dekripsi, menghasilkan bahwa proses enkripsi menggunakan AES memakan waktu selama 2,34 menit sedangkan proses dekripsinya terbilang lebih cepat yakni memakan waktu 24 detik dengan selisih waktu 2,10 menit. Pengujian dengan membandingkan ukuran data sebelum dan sesudah dilakukan enkripsi mengindikasikan peningkatan sebesar 54,03% pada data yang telah dilakukan enkripsi. Perubahan ukuran data tersebut diakibatkan adanya perubahan nilai pada bit data atau *avalanche effect*.

VI. SARAN

Berdasarkan hasil penelitian yang telah dilakukan ada beberapa saran untuk pengembangan lebih lanjut, diantaranya sebagi berikut :

1. Penelitian selanjutnya dapat meningkatkan kecepatan proses enkripsi dan dekripsi, agar waktu yang digunakan saat mengamankan data lebih efektif dan efisien.

2. Penelitian selanjutnya dapat mengurangi ukuran data hasil enkripsi sehingga data yang dihasilkan tidak akan terlalu bertambah ukurannya.

3. Penelitian selanjutnya dapat meningkatkan kompatibilitas terhadap berbagai jenis bahasa pemrograman terbaru sehingga menghasilkan sebuah sistem keamanan yang lebih aman dan mudah digunakan bagi pemakainya.

REFERENSI

[1] I. S. Andani dan D. L. Fithri, "Aplikasi Pengamanan Dokumen Digital Menggunakan Algoritma Kriptografi Advanced Encryption Standard (Aes-128), Kompresi Huffman Dan Steganografi End Of File (Eof) Berbasis Desktop Pada Cv. Karya Perdana," *Prosiding SNATIF*, no. 1, hal. 269–276, 2017.

[2] M. I. Padli, "Urgensi Keamanan Dalam Sistem Informasi," *Iqra'*, vol. 2, no. 2, 2008.

[3] M. R. Joshi dan R. A. Karkade, "Network Security with Cryptography," *International Journal of Computer Science and Mobile Computing*, vol. 41, no. 1, hal. 201–204, 2015.

[4] Supriyono, "Pengujian sistem enkripsi-dekripsi dengan metode rsa untuk pengamanan dokumen," *Jurnal Sekolah Tinggi Teknologi Nuklir – BATAN*, vol. 2, hal. 179–194, 2008.

[5] Hamdani, S. H. Suryawan, dan A. Septiarini, "Pengujian Algoritma Rivest Code 5 Untuk Enkripsi Struktur File Dokument," *Prosiding Seminar Nasional Teknik Informatika*, no. June, 2013.

[6] A. R. Tulloh, Y. Permanasari, dan E. Harahap, "Kriptografi Advanced Encryption Standard (AES) Untuk Penyandian File Dokumen," *Jurnal Matematika UNISBA*, vol. 2, no. 1, hal. 118–125, 2016.

[7] O. Setiawan, R. Fiati, dan T. Listyorini, "Algoritma Enkripsi Rc4 Sebagai Metode Obfuscation Source Code Php," *Prosiding SNATIF Ke-1*, hal. 113–120, 2014.

[8] A. P. Nugraha dan E. Gunadhi, "Penerapan Kriptografi Base64 Untuk Keamanan URL (Uniform Resource Locator) Website Dari Serangan SQL Injection," *Jurnal Algoritma Sekolah Tinggi Teknologi Garut*, hal. 491–498, 2016.

[9] P. Patil, P. Narayankar, D. G. Narayan, dan S. M. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Computer Science*, vol. 78, no. December 2015, hal. 617–624, 2016.

[10] G. Singh dan S. Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *International Journal of Computer Applications*, vol. 67, no. 19, hal. 33–38, 2013.

[11] F. N. Pabokory, I. F. Astuti, dan A. H. Kridalaksana, "Implementasi Kriptografi Pengamanan Data Pada Pesan Teks, Isi File Dokumen, Dan File Dokumen Menggunakan Algoritma Advanced Encryption Standard," *Jurnal Informatika Mulawarman*, vol. 10, no. 1, 2015.

[12] Syapriadi, Rahmiati, dan S. Erlinda, "Menggunakan Teknik Steganografi End Of File (EOF) dan Algoritma Kriptografi Rivest Code 4 (RC4)," *Sains dan Teknologi Informasi*, vol. 2, hal. 14, 2013.

[13] P. Algoritma, "Perbandingan Algoritma AES256 dan Blowfish," *Jurnal TRANSFORMASI*, vol. 14, no. 1, hal. 84–91, 2018.

[14] I. Sumartono, A. P. U. Siahaan, dan Arpan, "Base64 Character Encoding and Decoding Modeling," *International Journal of Recent Trends in Engineering & Research*, vol. 2, no. 12, hal. 63–68, 2016.

[15] R. Rahim *et al.*, "Combination Base64 Algorithm and EOF Technique for Steganography," *Journal of Physics: Conference Series*, vol. 1007, no. 1, 2018.