Analisis Perbandingan Kinerja Algoritma Kriptografi *Serpent* dan *Twofish* pada Dataset World Bank Projects and Operations

(Performance Comparative Analysis of Serpent and Twofish Cryptographic Algorithms on World Bank Projects and Operations Dataset)

Rio Sudrajat Puji Sutan, Antonius Cahya Prihandoko[*)], Diksy Media Firmansyah
Sistem Informasi, Fakultas Ilmu Komputer, Universitas Jember
Jl. Kalimantan 37 Jember 68121
[*)]Penulis Korespondensi E-mail: antoniuscp@gmail.com

Abstrak

Serpent dan *Twofish* merupakan algoritma kriptografi yang menduduki peringkat kedua dan ketiga dalam kompetisi *Advanced Encryption Standard* (AES). Penelitian ini bertujuan untuk mengetahui algoritma mana yang lebih unggul dalam proses enkripsi dan dekripsi suatu *file*, serta keamanan kedua algoritma dalam menahan serangan *brute force*. Sebagai simulasi, enkripsi dan dekripsi kedua algoritma akan diterapkan pada dataset *World Bank Projects and Operations*. Sebelum enkripsi, *file* dataset yang digunakan akan dibagi menjadi beberapa *file-file* sampel berdasarkan jumlah kolom data. Kemudian *file-file* sampel yang sudah dienkripsi akan didekripsi dengan serangan *brute force*. Pada pengujian pertama, untuk kunci 128 *bit*, kecepatan rata-rata untuk enkripsi dan dekripsi *Twofish* 35% dan 43% lebih tinggi daripada *Serpent*. Pada kunci 192 *bit*, kecepatan rata-rata untuk enkripsi dan dekripsi *Twofish* 40% dan 48% lebih tinggi daripada *Serpent*. Pada kunci 256 *bit*, kecepatan rata-rata untuk enkripsi dan dekripsi *Twofish* 42% dan 48% lebih tinggi daripada *Serpent*. Pengujian kedua menunjukkan bahwa kecepatan serangan *brute force* tidak terpengaruh oleh ukuran kunci. Kecepatan rata-rata serangan *brute force* pada *Twofish* 56% lebih tinggi daripada *Serpent*. Meski dengan perbedaan kecepatan serangan yang tinggi, mendekripsi *file* yang dienkripsi oleh *Twofish* tetap memerlukan waktu berabad-abad. Hal ini disebabkan oleh kombinasi kunci yang banyak sekali bahkan pada kunci 128-*bit*. Secara keseluruhan, *Twofish* lebih unggul daripada *Serpent* tanpa mengorbankan sisi keamanan *Twofish* terhadap serangan *brute force*.

Kata Kunci: *Brute Force*, Perbandingan, *Serpent*, *Twofish.*

Abstract

Serpent and Twofish are cryptographic algorithms which ranked second and third in the Advanced Encryption Standard (AES) competition. This study aims to find out which algorithm is superior in the encryption and decryption process of a file, as well as the security of both algorithms in resisting brute force attacks. As a simulation, the encryption and decryption of the two algorithms will be applied to the World Bank Projects and Operations dataset. Before encryption, the dataset file used will be divided into several sample files based on the number of data columns. Then the encrypted sample files will be decrypted with a brute force attack. In the first test, for a 128-bit key, the average speed for encryption and decryption of Twofish was 35% and 43% higher than Serpent. For 192-bit keys, the average speed for encryption and decryption of Twofish is 40% and 48% higher than Serpent. For a 256-bit key, the average speed for Twofish encryption and decryption is 42% and 48% higher than Serpent. The second test shows that the brute force attack speed is not affected by the key size. The average speed of brute force attacks on Twofish is 56% higher than Serpent's. Even with the high attack speed difference, decrypting files encrypted by Twofish can take centuries. This is caused by a large number of key combinations even on 128-bit keys. Overall, Twofish is superior to Serpent without compromising Twofish's security side against brute force attacks.

Keywords: *Brute Force, Comparative, Serpent, Twofish.*

PENDAHULUAN

Algoritma *Serpent* dan *Twofish* merupakan finalis dalam kompetisi *Advanced Encryption Standard* (AES) yang diadakan pada tahun 1997. Algoritma *Serpent* menduduki peringkat kedua, sementara algoritma *Twofish* menduduki peringkat ketiga. Algoritma *Serpent* dan *Twofish* dirancang untuk menahan serangan *brute force attack*. Kedua algoritma tersebut berstatus *open license* sehingga orang lain bebas menerapkan algoritma tersebut ke dalam *software* mereka. Keamanan yang baik dan status *open license* menyebabkan algoritma *Serpent* dan *Twofish* menjadi sangat populer. Hingga sekarang, algoritma *Serpent* dan *Twofish* masih banyak digunakan untuk mengamankan data karena murah dan dapat diandalkan [1].

Algoritma *Serpent* dan *Twofish* menggunakan jaringan yang berbeda. Algoritma *Serpent* menggunakan jaringan *Feistel*, sementara algoritma *Twofish* menggunakan jaringan Substitusi-Permutasi. Meskipun memiliki jaringan yang berbeda, algoritma *Serpent* dan *Twofish* memiliki beberapa kesamaan sebagai berikut:

1. Panjang kunci yang dapat dimasukkan adalah 128, 192, atau 256 bit.
2. Merupakan algoritma kunci simetris.

3. Memiliki ukuran blok enkripsi yang sama, yaitu sebesar 128 bit.

Berdasarkan hal tersebut, timbul berberapa pertanyaan untuk mengetahui kinerja mana yang lebih unggul antara algoritma *Serpent* dan *Twofish* dalam proses enkripsi dan dekripsi suatu *file*, serta bagaimana tingkat keamanan kedua algoritma dalam menahan serangan *brute force*. Penelitian ini dilakukan untuk menjawab pertanyaan-pertanyaan tersebut.

Sebagai simulasi, enkripsi dan dekripsi kedua algoritma akan diterapkan pada dataset *World Bank Projects and Operations*. Kemudian dataset yang sudah dienkripsi akan didekripsi dengan serangan *brute force*. Dataset ini cocok untuk digunakan sebagai obyek dalam penelitian ini karena dataset semacam inilah yang rawan dari serangan pihak tidak bertanggung jawab.

Penelitian ini menggunakan penelitian terdahulu untuk membantu penyusunan langkah-langkah penelitian dan implementasi algoritma-algoritma yang digunakan di dalam penelitian. Penelitian pertama yang berjudul *Analisis Perbandingan Kinerja Algoritma Blowfish dan Algoritma Twofish pada Proses Enkripsi dan Dekripsi* pada tahun 2015 oleh Dimas Aulia Trianggana dan Herlina Latipa Sari [18] akan mempermudah penentuan langkah-langkah yang sistematis dalam penyusunan penelitian mesikipun salah satu algoritma kriptografi yang digunakan berbeda dengan penelitian ini. Sementara penelitian kedua yang berjudul *Brute Force Attack dan Penerapannya pada Password Cracking* pada tahun 2010 oleh Krisnaldi Eka Pramudita [12] akan membantu penerapan algoritma *brute force* di dalam penelitian ini.

Serpent

Algoritma *Serpent* adalah algoritma *block cipher* dengan 32 putaran jaringan *substitution permutation* (SP). Algoritma *Serpent* bekerja pada *block* dengan ukuran 128 *bit*. Algoritma *Serpent* dapat menerima kunci dengan panjang 128, 192 atau 256 *bit*. Algoritma ini dikembangkan pada tahun 1998 oleh 3 peneliti, yaitu Ross Anderson, Lars Knudsen dan Eli Biham.

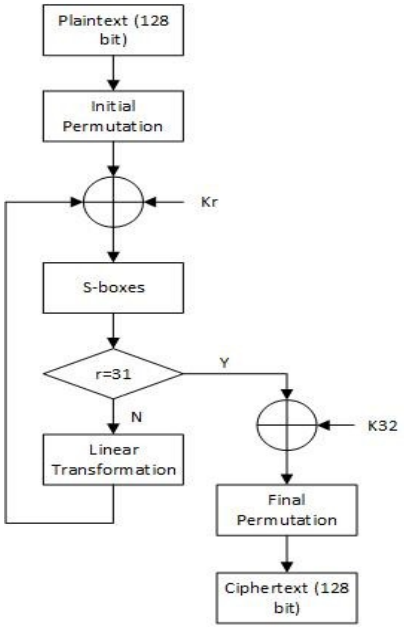
Algoritma *Serpent* merupakan finalis dalam kompetisi *Advanced Encryption Standard* (AES) yang diadakan oleh *National Institute of Standards and Technology of the United States* (NIST) pada tahun 1997. Kompetisi ini diadakan untuk menggantikan algoritma *Data Encryption Standard* (DES). DES perlu digantikan karena kunci 56-*bit* pada DES relatif kecil sehingga rentan terhadap serangan *brute force*. Algoritma *Serpent* menempati urutan kedua dengan 59 suara setelah algoritma *Rijndael* yang menempati urutan pertama dengan 86 suara.

Algoritma *Serpent* berada dalam domain publik dan tidak dipatenkan. Tidak ada batasan atau halangan apa pun terkait penggunaannya. Akibatnya, siapa pun bebas untuk menerapkan algoritma *Serpent* dalam perangkat lunak mereka atau perangkat keras mereka tanpa membayar biaya lisensi.

Algoritma *Serpent* terdiri atas:

1. *Initial permutation* (IP)
2. Putaran yang berjumlah 32, di mana tiap putaran mengandung operasi *key-mixing*, operasi terhadap S-box, dan sebuah transformasi linier (kecuali pada putaran terakhir). Pada putaran terakhir, transformasi linier ini diganti dengan operasi *key-mixing* tambahan.
3. *Final permutation* (FP)

Cara kerja algoritma Serpent dapat dipahami dengan melihat Gambar 1.



Gambar 1. Algoritma *Serpent*

Langkah-langkah dalam algoritma Serpent adalah sebagai berikut:

1. Putaran diberi nomor 0 sampai 31, di mana putaran pertama adalah putaran 0, dan putaran terakhir adalah putaran 31.
2. Initial permutation dioperasikan terhadap plainteks P dan menghasilkan B0 yang merupakan input putaran pertama (putaran 0).
3. B0 di-XOR dengan kunci pertama (K0).
4. Hasil dari B0 XOR K0 dimasukkan pada S-box pertama (S0). Setiap putaran hanya menggunakan satu S-box yang direplika. Putaran pertama menggunakan S0, putaran kedua menggunakan S1, dan seterusnya.
5. Apabila S7 telah digunakan, maka S-box yang digunakan kembali pada S0. Jadi setelah menggunakan S7 pada putaran 7, S0 digunakan lagi pada putaran 8, lalu S0 pada putaran 9, dan seterusnya.
6. Hasil dari S-box ditransformasikan menggunakan transformasi linier dan menghasilkan B1 yang menjadi input putaran kedua (putaran 1).
7. Pada putaran terakhir (putaran 31), hasil dari S-box tidak ditransformasikan menggunakan transformasi linier melainkan di-XOR-kan dengan K32.
8. Hasilnya (B32) lalu dipermutasikan dengan final permutation dan menghasilkan cipherteks.

Twofish

Algoritma *Twofish* merupakan 128-*bit cipher block* yang bisa menerima panjang varibel kunci 128, 192 atau 256 *bit*. *Cipher* tersebut berasal 16-*round* jaringan Feitsel dengan fungsi bijektif *F* yang dilanjutkan dengan empat *key-dependent 8-by-8-bit S-boxes*, satu *fixed 4-by-4 maximum distance separable matrix over GF*(2^8), satu

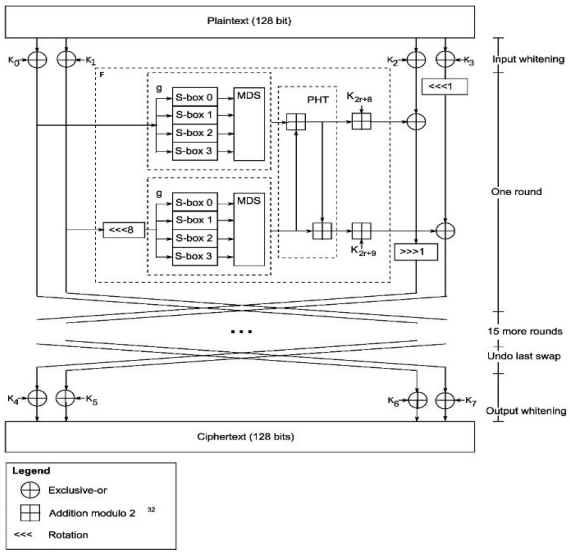
pseudo-Hadamard transform, satu rotasi *bitwise* dan satu *desain key schedule*. *Twofish* dirancang pada tahun 1998 oleh Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, dan Niels Ferguson.

Sama seperti algoritma *Serpent*, algoritma *Twofish* merupakan finalis dalam kompetisi *Advanced Encryption Standard* (AES) yang diadakan oleh *National Institute of Standards and Technology of the United States* (NIST) pada tahun 1997. Namun, algoritma *Twofish* gagal menjadi pemengan dalam kompetisi ini. Algoritma *Twofish* menempati urutan ketiga dengan 31 suara setelah algoritma *Serpent* yang menempati urutan kedua dengan 59 suara.

Algoritma *Twofish* berada dalam domain publik dan tidak dipatenkan. Tidak ada batasan atau halangan apa pun terkait penggunaannya. Akibatnya, siapa pun bebas untuk menerapkan algoritma *Twofish* dalam perangkat lunak mereka atau perangkat keras mereka tanpa membayar biaya lisensi.

Algoritma *Twofish* terdiri atas:
1. *Whitening*
2. Jaringan Feistel
3. *Substitution box* (S-box)
4. Matrik *maximum distance separable* (MDS)
5. *Pseudo-Hadamard transforms* (PHT)

Cara kerja algoritma *Twofish* dapat dipahami dengan melihat Gambar 2.



Gambar 2. Algoritma *Twofish* (Sumber: Schneier dkk., 1998:6)

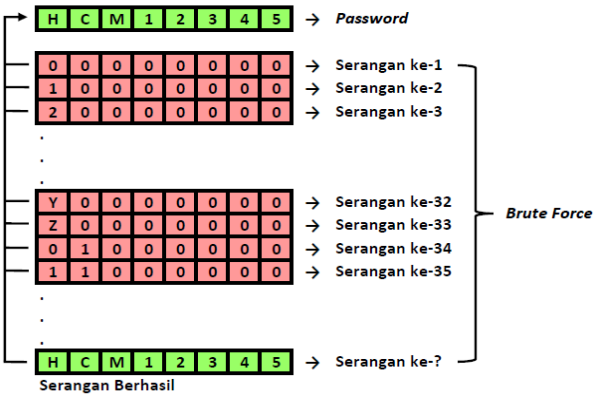
Langkah-langkah dalam algoritma *Twofish* adalah sebagai berikut:
1. Plainteks dibagi menjadi empat kata 32-*bit*.
2. Pada *input whitening*, empat kata tersebut di-XOR-kan dengan empat kata kunci, yaitu K_0, K_1, K_2, dan K_3.
3. Dua kata di sebelah kiri digunakan sebagai dimasukkan pada fungsi *g*. Namun sebelum dimasukkan, salah satunya dirotasi 8 *bit* terlebih dahulu.
4. Di dalam fungsi *g*, dua kata tersebut akan melalui *four byte-wide key-dependent* S-box dan diikuti oleh pencampuran linier pada matriks MDS,
5. Hasil dari fungsi *g* digabungkan dengan menggunakan PHT, dan dua kunci ditambahkan.
6. Kedua hasil kemudian di-XOR-kan ke dalam dua kata di sebelah kanan. Satu kata di sebelah kanan dirotasi ke kiri satu *bit* sebelum di-XOR-kan dan satu kata di sebelah kanan lainnya dirotasi satu *bit* ke kanan setelah di-XOR-kan.
7. Semua kata pada bagian kiri dan kanan akan ditukar untuk putaran selanjutnya.
8. Setelah melewati semua putaran, pertukaran pada putaran terakhir dibalikkan sebelum keempat kata melewati *output whitening*.
9. Pada *output whitening,* keempat kata di-XOR-kan dengan empat kata kunci tambahan untuk menghasilkan cipherteks.

Brute Force

Brute force adalah metode pembuktian matematis di mana pernyataan yang akan dibuktikan dibagi menjadi sejumlah kasus atau set kasus yang setara, dan di mana setiap jenis kasus diperiksa untuk melihat apakah proposisi dalam pertanyaan tersebut berlaku. Sementara serangan *brute force* adalah sebuah teknik serangan terhadap sebuah sistem keamanan komputer dengan memasukkan semua kunci yang mungkin [12]. Dalam kriptografi, serangan *brute force* terdiri dari penyerang yang mengirimkan banyak kata sandi atau frasa sandi dengan harapan pada akhirnya dapat menebak dengan benar. Penyerang secara sistematis memeriksa semua kata sandi dan frasa sandi yang mungkin sampai yang benar ditemukan.

Seiring bertambahnya panjang kata sandi pada *file* yang terenkripsi, jumlah waktu untuk menemukan kata sandi yang benar juga meningkat secara eksponensial. Jumlah waktu yang diperlukan untuk memecahkan sandi sebanding dengan ukuran kunci. Jumlah upaya maksimum sama dengan ukuran 2^k, di mana *k* adalah jumlah *bit* pada kunci tersebut. Sehingga bila suatu *file* dienkripsi dengan kunci yang memiliki ukuran 32 *bit*, maka usaha maksimum yang diperlukan untuk membuka *file* tersebut adalah $2^{32} = 4.294.967.296$.

Dalam serangan *online*, ada beberapa pencegahan yang bisa diambil. Administrator dapat membatasi jumlah upaya memasukkan kata sandi dengan memberikan penundaan waktu, memberikan CAPTCHA atau kode verifikasi tertentu, dan/atau mengunci akun. Administrator juga dapat mencegah alamat IP yang melakukan banyak percobaan untuk masuk terhadap akun tertentu. Contoh serangan *brute force* terhadap *password* dengan delapan karakter dapat dilihat pada Gambar 3.



Gambar 3. Serangan *brute force* pada *password* delapan karakter

Pada Gambar 10, karakter password yang bisa dimasukkan adalah angka 0 – 9 dan huruf a – Z. *Brute force* akan memulai serangan pertama dengan mengisi *password* dengan semua karakter dengan tingkat terkecil terlebih dahulu, yaitu angka 0. Apabila serangan pertama gagal, serangan selanjutnya dilakukan dengan menaikkan karakter pertama satu tingkat lebih tinggi. Apabila karakter pertama telah mencapai tingkat karakter tertinggi (serangan ke-33), maka karakter tersebut kembali ke tingkat paling rendah dan karakter di depannya dinaikkan satu tingkat (serangan ke-34). Serangan akan terus akan terus diulangi hingga semua karakter pada serangan *brute force* sama dengan *password* yang tersimpan dan berhasil masuk.

METODE PENELITIAN

Pengumpulan Data

Dataset yang akan digunakan untuk penelitian ini adalah dataset *World Bank Projects and Operations.* Dataset *World Bank Projects and Operations* adalah dataset yang berisi informasi dasar tentang semua proyek *World Bank*. Dataset ini sesuai untuk digunakan sebagai objek pengujian metode di dalam penelitian ini. Dataset *World Bank Projects and Operations* sesuai untuk digunakan sebagai objek pengujian karena memiliki dua *file* dataset dengan ukuran dan ekstensi yang berbeda. Perbedaan ukuran dan ekstensi *file* dataset diharapkan dapat memberikan hasil yang beragam pada hasil penelitian.

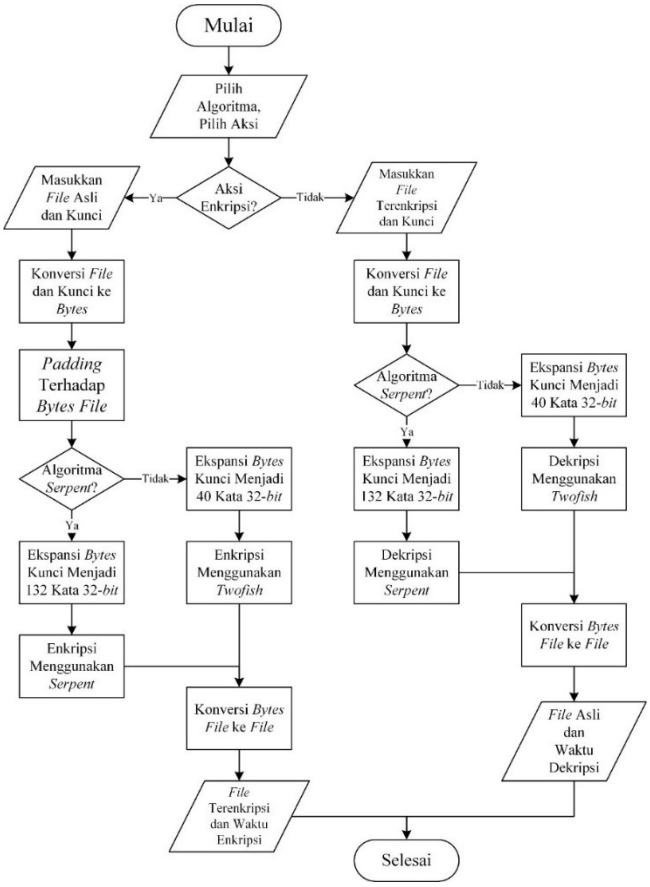
Dataset *World Bank Projects and Operations* mencangkup informasi dasar seperti judul proyek, pengelola tugas, negara, id proyek, sektor, tema, jumlah komitmen, baris produk, pemberitahuan pengadaan, penghargaan kontrak, dan pembiayaan. Dataset sejenis ini merupakan contoh dataset yang penting untuk dilindungi agar tidak bocor dan dimanfaatkan oleh pihak yang salah. Dataset ini didapat dari *website* yang menyediakan banyak dataset yang bernama *Kaggle* [7].

Dataset *World Bank Projects and Operation* terdiri dari dua *file*, yaitu *projects-operations-csv-.csv* dan *projects-operations-xml-.xml*. Dari kedua *file* tersebut akan dibuat *file-file* sampel berdasarkan kolom data di dalam *file* tersebut. Selanjutnya pengujian akan dilakukan kepada masing-masing *file-file* sampel. Hal ini ditujukan untuk menguji kinerja kedua algoritma *Twofish* dan *Serpent* pada ukuran *file* yang beragam.

Implementasi Metode

Algoritma *Serpent* dan *Twofish* akan diimplementasikan menggunakan API *Bouncy Castle*. *Bouncy Castle* adalah API kriptografi yang bersifat bebas dan *open source* yang dikembangkan oleh *Legion of the Bouncy Castle Inc.* API ini berisi beragam algoritma kriptografi, termasuk algoritma *Serpent* dan *Twofish*. Versi *Bouncy Castle* yang digunakan dalam penelitian ini adalah versi 1.61. API *Bouncy Castle* dapat diunduh pada website *bouncycastle.org*.

Program dalam penelitian ini akan dibangun menggunakan bahasa pemrograman *Java* dengan bantuan aplikasi *NetBeans 8.1*. Program ini dapat melakukan proses enkripsi dan dekripsi dari algoritma mana yang akan digunakan untuk proses tersebut serta menampilkan berapa lama waktu proses tersebut berlangsung. Dataset *World Bank Projects and Operations* nantinya akan diuji dengan program sederhana ini. Alur program dapat dilihat pada Gambar 4.



Gambar 4. *Flowchart* program

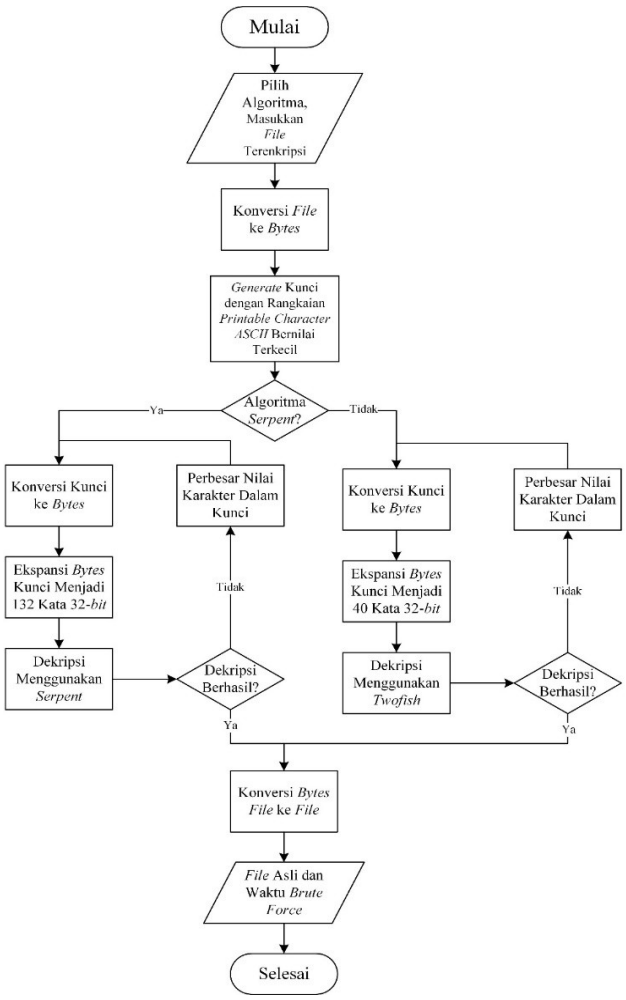
Pengujian

Sebelum pengujian dimulai, penguji akan membuat beberapa *file* sampel yang memiliki panjang kolom tertentu dari *file* asli. Hal ini ditujukan untuk menguji kinerja kedua algoritma *Twofish* dan *Serpent* pada ukuran *file* yang beragam.

Pengujian pertama dilakukan dengan membandingkan kinerja algoritma *Twofish* dan algoritma *Serpent*. Masing-masing algoritma akan diuji kinerjanya dengan membandingkan kecepatan proses enkripsi dan dekripsi, serta besar ukuran setiap *file* sampel sebelum dan sesudah

proses enkripsi dan dekripsi. Panjang kunci yang akan digunakan juga berbeda-beda. Masing-masing *file* akan dienkripsi menggunakan kunci dengan panjang 128, 192, dan 256 *bit* dari kedua algoritma untuk melihat pengaruh kinerja algoritma terhadap panjang kunci yang berbeda-beda. Karakter kunci yang digunakan adalah *printable character ASCII.*

File-file sampel dari dataset *World Bank Projects and Operations* akan dienkripsi menggunakan algoritma *Serpent* dan *Twofish* dengan kunci yang sama pada masing-masing panjang *bit*. Proses ini akan menghasilkan *file* yang terenkripsi dari kedua algoritma. Ketika proses enkripsi berlangsung, akan dilakukan pencatatan terhadap waktu yang diperlukan untuk menyelesaikan proses enkripsi tersebut dari masing-masing algoritma. Setelah proses enkripsi selesai, ukuran kedua *file* yang telah dienkripsi akan dicatat. Kedua *file* yang terenkripsi tersebut kemudian didekripsi dengan kunci yang digunakan sebelumnya. Pada proses dekripsi juga dilakukan pencatatan terhadap waktu yang diperlukan untuk menyelesaikan proses dekripsi tersebut dari masing-masing algoritma. Pengujian ini akan diulangi menggunakan kunci dengan panjang *bit* yang berbeda.



Gambar 5. *Flowchart* serangan *brute force*

Pengujian kedua dilakukan dengan melakukan serangan *brute force* pada *file-file* sampel yang sudah dienkripsi pada pengujian pertama berdasarkan algoritma yang digunakan. Setelah itu waktu yang diperlukan untuk membuka *file-file* tersebut akan dibandingkan satu sama lain. Serangan *brute force* dibangun pada program dengan *flowchart* pada Gambar 5.

Apabila *brute force* gagal mendekripsi *file* dalam waktu 24 jam, maka akan dibuat estimasi waktu yang diperlukan dengan menghitung waktu (*milisecond*) per serangan dikalikan jumlah serangan yang diperlukan. Jumlah serangan yang diperlukan yang dimaksud adalah jumlah serangan yang diperlukan untuk *brute force* agar sampai pada kunci yang digunakan. Jumlah serangan yang diperlukan dapat dihitung dengan rumus berikut.

$$E = 96^{P} - \sum_{i=1}^{P} \left(96 - K_i\right) \times 96^{i-1} \qquad (1)$$

E = estimasi percobaan yang diperlukan
P = panjang karakter kunci
k = nomor *printable character ASCII* pada karakter kunci ke-i

Jumlah *printable character ASCII* adalah 96. Nomor *printable character ASCII* dapat dilihat pada Tabel 1

Tabel 1. *Printable character ASCII*

Nomor	Karakter	Biner	Nomor	Karakter	Biner
1	<SPACE>	0010 0000	49	P	0101 0000
2	!	0010 0001	50	Q	0101 0001
3	"	0010 0010	51	R	0101 0010
4	#	0010 0011	52	S	0101 0011
5	$	0010 0100	53	T	0101 0100
6	%	0010 0101	54	U	0101 0101
7	&	0010 0110	55	V	0101 0110
8	'	0010 0111	56	W	0101 0111
9	(	0010 1000	57	X	0101 1000
10	)	0010 1001	58	Y	0101 1001
11	*	0010 1010	59	Z	0101 1010
12	+	0010 1011	60	[	0101 1011
13	,	0010 1100	61	\	0101 1100
14	-	0010 1101	62	]	0101 1101
15	.	0010 1110	63	^	0101 1110
16	/	0010 1111	64	_	0101 1111
17	0	0011 0000	65	`	0110 0000
18	1	0011 0001	66	a	0110 0001
19	2	0011 0010	67	b	0110 0010
20	3	0011 0011	68	c	0110 0011
21	4	0011 0100	69	d	0110 0100
22	5	0011 0101	70	e	0110 0101
23	6	0011 0110	71	f	0110 0110
24	7	0011 0111	72	g	0110 0111
25	8	0011 1000	73	h	0110 1000
26	9	0011 1001	74	i	0110 1001
27	:	0011 1010	75	j	0110 1010
28	;	0011 1011	76	k	0110 1011
29	<	0011 1100	77	l	0110 1100
30	=	0011 1101	78	m	0110 1101
31	>	0011 1110	79	n	0110 1110

32	?	0011 1111	80	o	0110 1111
33	@	0100 0000	81	p	0111 0000
34	A	0100 0001	82	q	0111 0001
35	B	0100 0010	83	r	0111 0010
36	C	0100 0011	84	s	0111 0011
37	D	0100 0100	85	t	0111 0100
38	E	0100 0101	86	u	0111 0101
39	F	0100 0110	87	v	0111 0110
40	G	0100 0111	88	w	0111 0111
41	H	0100 1000	89	x	0111 1000
42	I	0100 1001	90	y	0111 1001
43	J	0100 1010	91	z	0111 1010
44	K	0100 1011	92	{	0111 1011
45	L	0100 1100	93 |		0111 1100
46	M	0100 1101	94	}	0111 1101
47	N	0100 1110	95	~	0111 1110
48	O	0100 1111	96	<DELETE>	0111 1111

HASIL DAN PEMBAHASAN

Uji Kinerja

Hasil dari pengujian pertama adalah perbandingan kinerja algoritma *Twofish* dan *Serpent*. Hasil tersebut dijabarkan dalam bentuk sebuah tabel dan enam grafik. Tabel akan menampilkan data dari hasil pengujian pertama yang diurutkan berdasarkan panjang kunci dan ekstensi *file* yang digunakan. Sementara keenam grafik akan menampilkan kecepatan proses enkripsi dan dekripsi yang dibagi berdasarkan beragam panjang kunci yang dipakai di dalam penelitian ini, yaitu 128, 192, dan 256 *bit*.

Menurut hasil dari pengujian pertama, ukuran *file* sesudah enkripsi selalu lebih besar daripada ukuran *file* sebelum enkripsi. Hal ini disebabkan oleh proses *padding* yang menambahkan beberapa *bytes* ke dalam *file*. Proses *padding* terjadi karena mode *CBC* tidak dapat bekerja apabila *block* terakhir tidak seukuran dengan *block* lainnya, yaitu delapan *bytes*. Meski demikian, peningkatan ukuran *file* akibat proses *padding* tidak signifikan sehingga tidak perlu dikhawatirkan. Selain dari proses *padding*, tidak ada proses lain pada enkripsi dari algoritma *Twofish* maupun *Serpent* yang dapat menambah ukuran *file*.

Pada pengujian pertama, proses enkripsi algoritma *Twofish* lebih cepat daripada algoritma *Serpent*. Proses dekripsi algoritma *Twofish* juga lebih cepat daripada algoritma *Serpent*. Panjang kunci yang digunakan tidak memiliki pengaruh yang signifikan terhadap kecepatan enkripsi maupun dekripsi algortima *Serpent* dan *Twofish*. Dalam pengujian juga ditemukan bahwa kedua algoritma memiliki proses enkripsi yang lebih cepat daripada proses dekripsi.

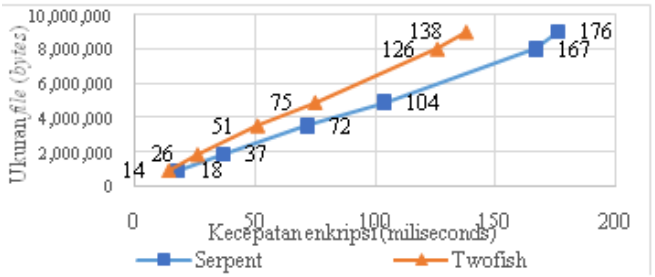
Tabel perbandingan kinerja algoritma *Twofish* dan *Serpent* dapat dilihat pada Tabel 2.

Tabel 2. Perbandingan kinerja algoritma *Serpent* dan *Twofish*

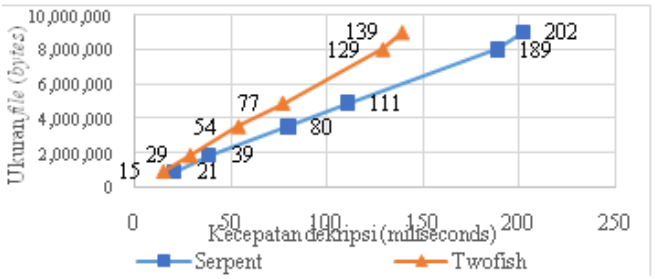
Nama *file* sampel	Panjang kunci (*bit*)	Kunci	Ukuran *file* asli (*bytes*)	Ukuran terenkripsi (*bytes*)		Kecepatan proses enkripsi (*miliseconds*)		Kecepatan proses dekripsi (ms)	
				Serpent	Twofish	Serpent	Twofish	Serpent	Twofish
csv1.csv	128	AunGjF5w5Qy93Trv	914.731	914.736	914.736	18	14	21	15
csv2.csv	128	AunGjF5w5Qy93Trv	8.023.566	8.023.568	8.023.568	167	126	189	129
csv3.csv	128	AunGjF5w5Qy93Trv	9.012.104	9.012.112	9.012.112	176	138	202	139
xml1.xml	128	AunGjF5w5Qy93Trv	1.848.670	1.848.672	1.848.672	37	26	39	29
xml2.xml	128	AunGjF5w5Qy93Trv	3.530.865	3.530.880	3.530.880	72	51	80	54
xml3.xml	128	AunGjF5w5Qy93Trv	4.895.294	4.895.296	4.895.296	104	75	111	77
csv1(2).csv	192	RrtJ8DFuW5SjnUGQkWeNczMs	914.731	914.736	914.736	19	15	22	16
csv2(2).csv	192	RrtJ8DFuW5SjnUGQkWeNczMs	8.023.566	8.023.568	8.023.568	178	123	192	125
csv3(2).csv	192	RrtJ8DFuW5SjnUGQkWeNczMs	9.012.104	9.012.112	9.012.112	191	139	209	140
xml1(2).xml	192	RrtJ8DFuW5SjnUGQkWeNczMs	1.848.670	1.848.672	1.848.672	39	27	44	29
xml2(2).xml	192	RrtJ8DFuW5SjnUGQkWeNczMs	3.530.865	3.530.880	3.530.880	75	53	81	56
xml3(2).xml	192	RrtJ8DFuW5SjnUGQkWeNczMs	4.895.294	4.895.296	4.895.296	106	72	114	75
csv1(3).csv	256	YujRYAsTFX8KUwVdZ6ZC8h8 yCGfKK7rf6Wjh	914.731	914.736	914.736	20	14	22	14
csv2(3).csv	256	YujRYAsTFX8KUwVdZ6ZC8h8 yCGfKK7rf6Wjh	8.023.566	8.023.568	8.023.568	179	124	186	127
csv3(3).csv	256	YujRYAsTFX8KUwVdZ6ZC8h8 yCGfKK7rf6Wjh	9.012.104	9.012.112	9.012.112	196	140	205	142
xml1(3).xml	256	YujRYAsTFX8KUwVdZ6ZC8h8 yCGfKK7rf6Wjh	1.848.670	1.848.672	1.848.672	38	28	45	30
xml2(3).xml	256	YujRYAsTFX8KUwVdZ6ZC8h8	3.530.865	3.530.880	3.530.880	77	55	82	60

		yCGfKK7rf6Wjh							
xml3(3).xml	256	YujRYAsTFX8KUwVdZ6ZC8h8 yCGfKK7rf6Wjh	4.895.294	4.895.296	4.895.296	107	73	112	74

Grafik dari hasil pengujian kinerja kedua algoritma untuk kunci 128 *bit* dapat dilihat pada Gambar 6.



(a)



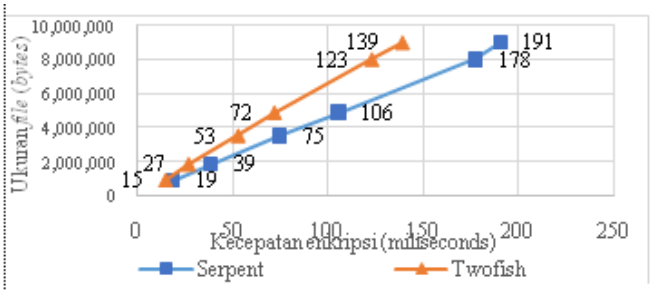
(b)

(a) Enkripsi; (b) Dekripsi

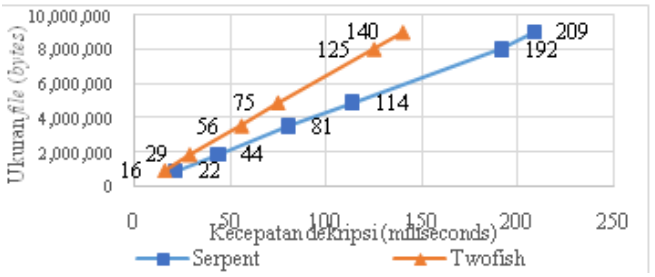
Gambar 6. Grafik kinerja pada kunci 128 *bit*

Pada kunci 128 *bit*, hasil pengujian pertama menunjukkan bahwa proses enkripsi algoritma *Twofish* 28% – 42% lebih cepat daripada algoritma *Serpent*. Proses dekripsi algoritma *Twofish* juga 34% – 48% lebih cepat daripada algoritma *Serpent*. Apabila dilihat dari kecepatan rata-rata, hasil pengujian pertama menunjukkan bahwa kecepatan rata-rata enkripsi algoritma *Twofish* 35% lebih tinggi daripada algoritma *Serpent*. Kecepatan rata-rata dekripsi algoritma *Twofish* juga 43% lebih tinggi daripada algoritma *Serpent*.

Grafik dari hasil pengujian kinerja kedua algoritma untuk kunci 192 *bit* dapat dilihat pada Gambar 7.



(a)



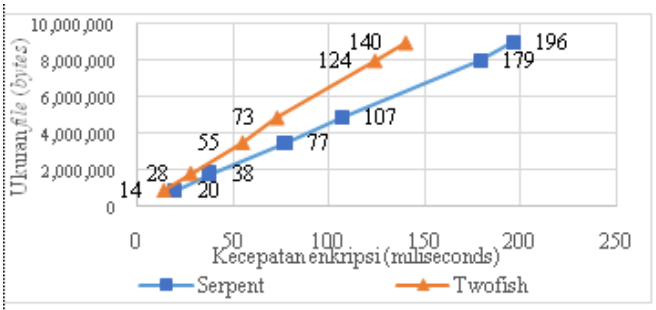
(b)

(a) Enkripsi; (b) Dekripsi

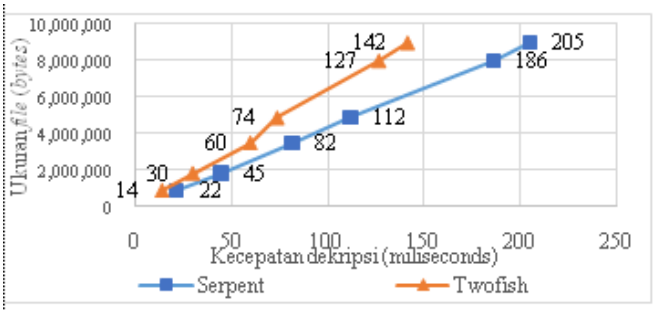
Gambar 7. Grafik kinerja pada kunci 192 *bit*

Pada kunci 192 *bit*, hasil pengujian pertama menunjukkan bahwa proses enkripsi algoritma *Twofish* 27% – 47% lebih cepat daripada algoritma *Serpent*. Proses dekripsi algoritma *Twofish* juga 38% – 54% lebih cepat daripada algoritma *Serpent*. Apabila dilihat dari kecepatan rata-rata, hasil pengujian pertama menunjukkan bahwa kecepatan rata-rata enkripsi algoritma *Twofish* 40% lebih tinggi daripada algoritma *Serpent*. Kecepatan rata-rata dekripsi algoritma *Twofish* juga 48% lebih tinggi daripada algoritma *Serpent*.

Grafik dari hasil pengujian kinerja kedua algoritma untuk kunci 256 *bit* dapat dilihat pada Gambar 8.



(a)



(b)

(a) Enkripsi; (b) Dekripsi

Gambar 8. Grafik kinerja pada kunci 256 *bit*

Pada kunci 256 *bit*, hasil pengujian pertama menunjukkan bahwa proses enkripsi algoritma *Twofish* 36% – 47% lebih cepat daripada algoritma *Serpent*. Proses dekripsi algoritma *Twofish* juga 37% – 57% lebih cepat daripada algoritma *Serpent*. Apabila dilihat dari kecepatan rata-rata, hasil pengujian pertama menunjukkan bahwa kecepatan rata-rata enkripsi algoritma *Twofish* 42% lebih tinggi daripada algoritma *Serpent*. Kecepatan rata-rata dekripsi algoritma *Twofish* juga 48% lebih tinggi daripada algoritma *Serpent*.

Uji Keamanan

Hasil dari pengujian kedua adalah pengujian ketahanan *file* yang dienkripsi oleh algoritma *Twofish* dan *Serpent* terhadap serangan *brute force*. Hasil tersebut dijabarkan dalam bentuk tabel dan sebuah grafik. Tabel akan menampilkan data dari hasil pengujian kedua yang diurutkan berdasarkan panjang kunci dan ekstensi *file* yang digunakan. Grafik yang ditampilkan hanya satu karena tidak terdapat perbedaan kecepatan serangan *brute force* pada ukuran kunci yang berbeda.

Menurut hasil dari pengujian kedua, tidak ada serangan *brute force* yang berhasil menembus *file* yang dienkripsi algoritma *Serpent* maupun *Twofish* dalam jangka waktu yang ditentukan (satu hari). Karena serangan *brute force* gagal menembus semua *file* yang terenkripsi, estimasi waktu yang diperlukan ditambahkan ke semua data di dalam tabel. Estimasi waktu yang diperlukan dihitung dengan mengalikan serangan yang diperlukan dengan kecepatan serangan.

Untuk mendapat serangan yang diperlukan, semua karakter kunci diubah menjadi nomor kunci sesuai yang tertera pada Tabel 3.1 terlebih dahulu. Kemudian nomor-nomor kunci tersebut dimasukkan ke dalam Rumus 3.1 untuk mendapatkan serangan yang diperlukan. Misalnya kunci yang digunakan adalah *AunGjF5w5Qy93Trv*, maka kumpulan nomor yang dihasilkan adalah *65 117 110 71 106 70 53 119 53 81 121 57 51 84 114 118*. Kemudian kumpulan nomor tersebut dimasukkan ke dalam Rumus 3.1 dan menghasilkan serangan yang diperlukan, yaitu sebanyak *47.085.531.238.274.286.947.001.268.017.154*.

Meskipun dengan kemampuan komputer/laptop generasi sekarang, memecahkan *file* yang terenkripsi oleh algoritma *Serpent* atau *Twofish* dengan serangan *Brute Force* membutuhkan waktu yang sangat lama. Kedua algoritma tersebut terbukti sangan aman dari serangan *Brute Force* meskipun panjang kunci yang digunakan adalah 128 *bit*.

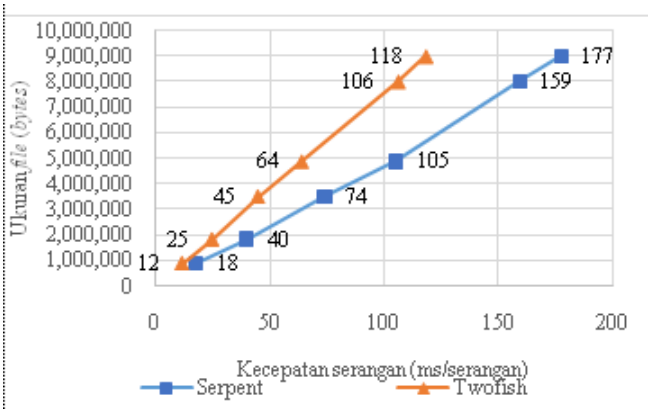
Tabel hasil dari pengujian ketahanan kedua algoritma terhadap serangan *Brute Force* dapat dilihat pada Tabel 3.

Tabel 3. Pengujian ketahanan terhadap serangan *brute force*

Nama *file* sampel	Kunci	Waktu *brute force*		Status serangan	Kecepatan serangan (ms/serangan)		Serangan yang diperlukan	Estimasi waktu yang diperlukan	
		Serpent	*Twofish*		*Serpent*	*Twofish*		*Serpent*	*Twofish*
csv1.csv	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	18	12	47.085.531.238.274.286.947.001.268.017.154	$2{,}68\times10^{22}$ tahun	$1{,}79\times10^{22}$ tahun
csv2.csv	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	159	106	47.085.531.238.274.286.947.001.268.017.154	$2{,}37\times10^{23}$ tahun	$1{,}58\times10^{23}$ tahun
csv3.csv	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	177	118	47.085.531.238.274.286.947.001.268.017.154	$2{,}64\times10^{23}$ tahun	$1{,}76\times10^{23}$ tahun
xml1.xml	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	40	25	47.085.531.238.274.286.947.001.268.017.154	$5{,}97\times10^{22}$ tahun	$3{,}73\times10^{22}$ tahun
xml2.xml	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	74	45	47.085.531.238.274.286.947.001.268.017.154	$1{,}10\times10^{23}$ tahun	$6{,}71\times10^{22}$ tahun
xml3.xml	AunGjF5w5Qy93Trv	>24 jam	>24 jam	Gagal menembus	105	64	47.085.531.238.274.286.947.001.268.017.154	$1{,}57\times10^{23}$ tahun	$9{,}55\times10^{22}$ tahun
csv1(2).csv	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	18	12	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$1{,}86\times10^{38}$ tahun	$1{,}24\times10^{38}$ tahun
csv2(2).csv	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	159	106	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$1{,}64\times10^{39}$ tahun	$1{,}10\times10^{39}$ tahun
csv3(2).csv	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	177	118	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$1{,}83\times10^{39}$ tahun	$1{,}22\times10^{39}$ tahun
xml1(2).xml	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	40	25	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$4{,}14\times10^{38}$ tahun	$2{,}59\times10^{38}$ tahun
xml2(2).xml	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	74	45	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$7{,}65\times10^{38}$ tahun	$4{,}66\times10^{38}$ tahun
xml3(2).xml	RrtJ8DFuW5SjnUGQkWeNczMs	>24 jam	>24 jam	Gagal menembus	105	64	326.447.596.029.809.514.971.290.058.210.440.459.288.842.530.547	$1{,}09\times10^{39}$ tahun	$6{,}62\times10^{38}$ tahun
csv1(3).csv	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6W	>24 jam	>24 jam	Gagal menembus	18	12	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.	$9{,}95\times10^{61}$ tahun	$6{,}63\times10^{61}$ tahun

	jh						377.690	
csv2(3).csv	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6Wjh	>24 jam	>24 jam	Gagal menembus	159	106	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.377.690	$8{,}79\times10^{62}$ tahun	$5{,}86\times10^{62}$ tahun
csv3(3).csv	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6Wjh	>24 jam	>24 jam	Gagal menembus	177	118	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.377.690	$9{,}78\times10^{62}$ tahun	$6{,}52\times10^{62}$ tahun
xml1(3).xml	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6Wjh	>24 jam	>24 jam	Gagal menembus	40	25	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.377.690	$2{,}21\times10_{62}$ tahun	$1{,}38\times10^{62}$ tahun
xml2(3).xml	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6Wjh	>24 jam	>24 jam	Gagal menembus	74	45	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.377.690	$4{,}08\times10^{62}$ tahun	$2{,}49\times10^{62}$ tahun
xml3(3).xml	YujRYAsTFX8KUwVdZ6ZC8h8yCGfKK7rf6Wjh	>24 jam	>24 jam	Gagal menembus	105	64	174.375.823.948.711.917.644.285.141.827.142.460.937.306.828.824.591.378.487.407.926.843.377.690	$5{,}80\times10^{62}$ tahun	$3{,}54\times10^{62}$ tahun

Grafik dari hasil pengujian ketahanan terhadap serangan *brute force* dapat dilihat pada Gambar 9.



Gambar 9. Grafik ketahanan terhadap serangan *brute force*

Pada pengujian kedua, kecepatan serangan *brute force* tidak terpengaruh oleh ukuran kunci. Kecepatan serangan *brute force* hanya terpengaruh oleh ukuran *file* yang terenkripsi. Kecepatan serangan *brute force* pada algoritma *Twofish* 50% – 64% lebih cepat daripada algoritma *Serpent*. Apabila dilihat dari kecepatan rata-rata, hasil pengujian kedua menunjukkan bahwa kecepatan rata-rata serangan *brute force* pada algoritma *Twofish* 56% lebih tinggi daripada algoritma *Serpent*.

Meski dengan perbedaan kecepatan serangan yang tinggi, mendekripsi file yang dienkripsi oleh algoritma Twofish tetap memerlukan waktu yang sangat lama. Hal ini disebabkan oleh kombinasi kunci yang banyak sekali bahkan pada kunci 128-bit. Sebagai contoh, pada data, serangan brute force tercepat yang bisa dilakukan adalah 12 miliseconds per serangan pada file yang berukuran 914 kilobytes. Sehingga dalam waktu satu menit, terdapat sekitar 5.000 serangan yang bisa dilakukan pada file tersebut. Namun dengan kombinasi kunci yang banyak, memecahkan file tersebut membutuhkan waktu berabad-abad.

KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan, didapatkan kesimpulan bahwa pada pengujian kinerja, algoritma *Twofish* bekerja lebih baik daripada algoritma *Serpent*. Proses enkripsi dan dekripsi *Twofish* bekerja lebih cepat daripada *Serpent* pada semua panjang kunci, yaitu 128, 192, dan 256 *bit*. Ukuran *file* sesudah enkripsi pada kedua algoritma selalu lebih besar beberapa *bytes* daripada ukuran *file* sebelum enkripsi, namun ukuran *bytes* yang ditambahkan selalu sama. Penambahan ini disebabkan oleh proses *padding* pada mode *CBC*. *Bytes* yang ditambahkan sangat kecil dan tidak memiliki dampak yang signifikan pada ukuran *file*.

Pada pengujian ketahanan terhadap serangan *brute force*, *file* yang dienkripsi oleh algoritma *Twofish* maupun *Serpent* terbukti sangat aman. Meskipun kecepatan serangan *brute force* pada *file* yang dienkripsi oleh algoritma *Twofish* lebih cepat, kombinasi kunci yang banyak menyebabkan dekripsi *file* dengan serangan *brute force* membutuhkan waktu berabad-abad. Sehingga algoritma *Twofish* lebih unggul daripada *Serpent* tanpa mengorbankan sisi keamanan algoritma *Twofish* terhadap serangan *brute force*.

DAFTAR PUSTAKA

[1] N. Aidinyantz, *A Glossary of Cryptographic Algorithms,* 2017. Diakses 25 Februari 2019, https://www.globalsign.com/en/blog/glossary-of-cryptographic-algorithms.

[2] A. R. Amin, 2006. *Studi Block Cipher Serpent dan Rijndael,* 2006. Diakses 25 Februari 2019, http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2006-2007/Makalah1/Makalah1-023.pdf.

[3] R. Anderson, E. Biham, dan L. Knudsen. *Serpent: A Proposal for the Advanced Encryption Standard*, 1998.

[4] D. Andika, *Pengertian dan Sejarah Kriptografi,* 2016. Diakses 20 Februari 2019, https://www.it-jurnal.com/pengertian-dan-sejarah-kriptografi.
[5] A. Burhanuddin, *Penelitian Kuantitatif dan Kualitatif,* 2013. Diakses 4 April 2019, https://afidburhanuddin.wordpress.com/2013/05/21/penelitian-kuantitatif-dan-kualitatif.
[6] N. Fatin, *Pengertian Kriptografi serta Komponen dan Tujuannya,* 2017. Diakses 7 Maret 2019, http://seputarpengertian.blogspot.com/2017/07/pengertian-kriptografi-serta-komponen.html.
[7] Kaggle, 2019. Diakses pada tanggal 22 Maret 2019, https://www.kaggle.com.
[8] S. Kromodimoeljo, *Teori dan Aplikasi Kriptografi.* SPK IT Consulting, 2010.
[9] H. Maradona, dan Basorudin, Analisis Algoritma Blowfish Pada Proses Enkripsi Dan Dekripsi File. *Riau Journal Of Computer Science*. vol.3, no. 2, pp. 156-168, 2017.
[10] R. Munir, *Kriptografi*. Informatika, Bandung, 2006.
[11] B. Parmaza, *Kriptografi Klasik,* 2017. Diakses 26 Februari 2019, http://itjambi.com/kriptografi-klasik/.
[12] K.E. Pramudita 2010. *Brute Force Attack dan Penerapannya pada Password Cracking,* 2010. Diakses 16 Maret 2019, https://www.academia.edu/8679011/Brute_Force_Attack_dan_Penerapannya_pada_Password_Cracking.
[13] Ratih, Studi dan Implementasi Enkripsi Pengiriman Pesan Suara Menggunakan Algoritma Twofish. *Jurnal Ilmu Komputer dan Teknologi Informasi,* vol 3, no. 2, 2003.
[14] M. Riadi, *Pengertian, Sejarah dan Jenis Kriptografi,* 2014. Diakses 5 Maret 2019, https://www.kajianpustaka.com/2014/01/pengertian-sejarah-dan-jenis-kriptografi.html.
[15] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, dan N. Ferguson. *Twofish: A 128-Bit Block Cipher,* 1998.
[16] W. Setiawan, *Analisa dan Perbandingan Algoritma Twofish dan Rijndael*, 2010. Diakses 26 Februari 2019, http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2010-2011/Makalah1/Makalah1-IF3058-Sem1-2010-2011-028.pdf.
[17] Teddy, *Inilah Cara Melakukan Brute Force untuk Pemula*, 2018. Diakses 20 Maret 2019, https://dosenit.com/jaringan-komputer/security-jaringan/cara-melakukan-brute-force.
[18] D. A. Trianggana, dan H. L. Sari, Analisis Perbandingan Kinerja Algoritma Blowfish dan Algoritma Twofish Pada Proses Enkripsi dan Dekripsi. *Jurnal Pseudocode,* vol.2, no. 1, pp. 37-44, 2015.