ANALISIS PERBANDINGAN KINERJA ALGORITMA TWOFISH DAN TEA (TINY ENCRYPTION ALGORITHM) PADA DATA SUARA

Andi Hendra[1]
[1]Jurusan Matematika MIPA Universitas Tadulako

Abstrak

Selain dokumen yang berupa teks, komunikasi dalam jaringan juga dimungkinkan dengan menggunakan dokumen suara. Makalah ini membahas mengenai implementasi dua algoritma kriptografi chiper blok, Twofish dan TEA pada pengamanan dokumen yang berupa suara. Kinerja algoritma dibandingkan berdasarkan kategori tertentu yang akan menunjukkan bagaimana tingkat keamanan dari kedua algoritma tersebut. Hasil penelitian pada tiga format data suara yang berbeda (mp3, WAV dan MID), menunjukkan bahwa tingkat keamanan algoritma TEA jauh lebih baik dibandingkan dengan algoritma Twofish,

Kata kunci : kriptografi, chiper blok, Twofish, TEA, enkripsi

I. Pendahuluan

Selain dokumen yang berupa teks, komunikasi dalam suatu jaringan juga dimungkinkan dengan menggunakan dokumen suara. Untuk menghindari agar dokumen suara yang kita kirimkan tidak diketahui oleh pihak yang tidak berkepentingan, diperlukan adanya pengamanan pada dokumen tersebut. Salah satu solusi yang dapat dilakukan adalah dengan melakukan enkripsi pada dokumen tersebut. Enkripsi dilakukan pada dokumen suara sebelum dikirimkan sehingga pihak lain yang tidak berhak tidak dapat memahami dokumen tersebut meskipun berhasil di akses[3]. Pada makalah ini, akan digunakan dua algoritma enkripsi yakni Twofish dan TEA.

Algoritma Twofish dan TEA adalah dua jenis algoritma enkripsi yang berupa chiper blok[1]. Cipher blok adalah algoritma enkripsi dimana data yang dienkripsi dibagi-bagi menjadi blok-blok berukuran sama (biasanya 64 bit), dan setiap blok dienkripsi masing-masing. Algoritma Twofish merupakan algoritma kuat yang sampai saat ini dinyatakan aman karena masih belum ada serangan kriptanalisis yang benar-benar dapat mematahkan algoritma ini . Algoritma ini juga tidak dipatenkan, sehingga penggunaannya pada alat enkripsi tidak perlu mengeluarkan biaya[2].

Tiny Encyption Algorithm (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham dari Computer Laboratory, Cambridge University, England pada bulan November 1994. Algoritma ini merupakan algoritma penyandian *block cipher* yang dirancang untuk penggunaan memory yang seminimal mungkin dengan kecepatan proses yang maksimal[7].

Kekuatan tingkat keamanan suatu algoritma dapat diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data chiperteks menjadi plainteksnya[3]. Untuk kriptografi modern, kekuatan algoritma terletak pada kuncinya, yaitu berupa deretan karakter atau bilangan bulat yang dijaga kerahasiaannya. Kunci ini dapat dianalogikan dengan penggunaan PIN pada ATM.

Oleh karena itu, pada makalah ini dipilih analisis perbandingan kinerja algoritma Twofish dan TEA dalam mendekripsi chiperteks ke plainteks, sehingga nantinya bisa dibandingkan tingkat keamanan dari kedua algoritma tersebut.

II. Twofish

Algoritma Twofish merupakan algoritma kuat yang sampai saat ini dinyatakan aman karena masih belum ada serangan kriptanalisis yang benar – benar dapat mematahkan algoritma ini[4]. Algoritma ini juga tidak dipatenkan sehingga penggunaannya pada alat enkripsi tidak perlu mengeluarkan biaya.

II.1. Desain dan Keunggulan Twofish

Algoritma Twofish merupakan salah satu algoritma yang direkomendasikan sebagai AES. Hal ini disebabkan pemenuhan kriteria desain oleh NIST sebagai standar AES yaitu :

1) Blok cliper simetris 128-bit
2) Memiliki panjang kunci antara lain : 128 bit, 192 bit, dan 256 bit.
3) Tidak terdapat kunci – kunci yang lemah.
4) Memiliki efisiensi pada software dan hardware dari platform yang berbeda.
5) Memiliki rancangan yang fleksibel, misalnya menerima panjang kunci tambahan, dapat diterapkan pada software dan hardware dari platform berbeda, cocok untuk stream chipper, fungsi hash dan MAC.
6) Desain yang simpel, memudahkan baik untuk analisa maupun implementasi.

Beberapa keunggulan algoritma kriptografi Twofish yaitu :

- Memiliki varian dengan sebuah nomor variabel dari setiap round.
- Memiliki key schedule yang dapat diprakomputasikan untuk kecepatan maksimum dan penggunaan memori minimum.
- Cocok sebagai stream chipper, fungsi hash satu arah, MAC dan pseudo random number generator, dengan menggunakan metode konstruksi yang dapat dimengerti.
- Memiliki varian famili-key untuk memungkinkan versi chipper yang berbeda dan non interrupterable.

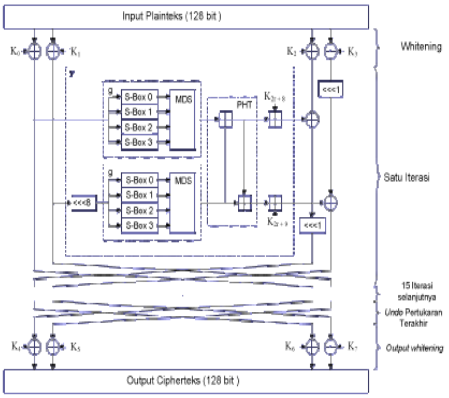
II.2. Algoritma Twofish

Seperti telah dibahas sebelumnya, Twofish beroperasi pada satu blok plainteks, blok plainteks pada Twofish terdiri dari 128 bit. Struktur Algoritma Twofish seperti pada Gambar 1. Pada implementasi algoritma Twofish, terdapat beberapa hal yang harus diperhatikan[6], antara lain:

1. Bit masukan sebanyak 128 bit akan dibagi menjadi empat bagian masing-masing sebesar 32 bit menggunakan konvensi *little-endian*. Dua bagian bit akan menjadi bagian kanan, dua bagian bit lainnya akan menjadi bagian kiri.
2. Bit *input* akan di-XOR terlebih dahulu dengan empat bagian kunci, atau dengan kata lain mengalami proses *whitening*.

   R0,i = Pi □*_Ki* i = 0, ..., 3 Dimana K adalah kunci, Ki berarti sub kunci yang ke-i.

3. Seperti telah dibahas diatas, algoritma Twofish menggunakan struktur jaringan Feistel. Jaringan Feistel yang digunakan oleh Twofish terdiri dari 16 iterasi. Fungsi f dari Twofish terdiri dari beberapa tahap, yaitu:
   a. Fungsi g, yang terdiri dari empat s-box dan matriks MDS
   b. PHT (*pseudo-hadamard transform*/ perubahan *pseudo hadamard*)
   c. Penambahan hasil PHT dengan kunci



Gambar 1. Struktur Algoritma Twofish

III. TEA (Tiny Encryption Algorithm)

Tiny Encryption Algorithm (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham dari Computer Laboratory, Cambridge University, England pada bulan November 1994. Algoritma ini merupakan algoritma penyandian *block cipher* yang dirancang untuk penggunaan memory yang seminimal mungkin dengan kecepatan proses yang maksimal[5].

III.1. Dekripsi Algoritma TEA

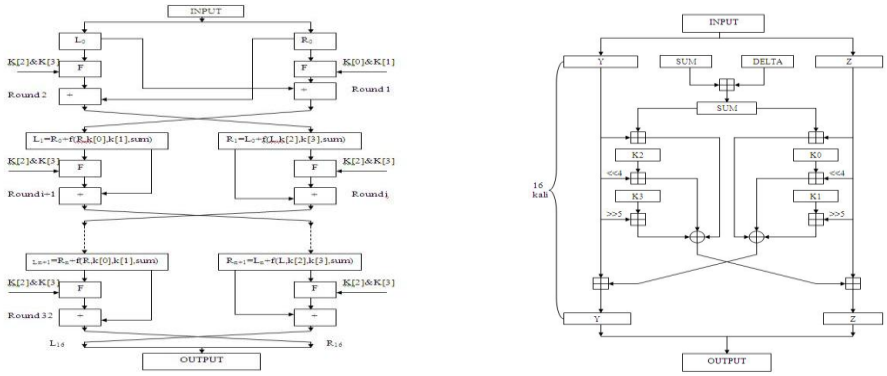
Sistem penyandian TEA menggunakan proses *feistel network* dengan menambahkan fungsi matematik berupa penambahan dan pengurangan sebagai operator pembalik selain XOR. Hal ini dimaksudkan untuk menciptakan sifat non-linearitas. Pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua bit kunci dan data bercampur secara berulang ulang. TEA memproses 64-bit input sekali waktu dan menghasilkan 64-bit output. TEA menyimpan 64-bit input kedalam L_0 dan R_0 masing masing 32-bit. Sedangkan 128-bit kunci disimpan kedalam k(0), k(1), k(2), dan k(3) yang masing masing berisi 32-bit. Diharapkan teknik ini cukup dapat mencegah penggunaan teknik *exshautive search* secara efektif. Hasil outputnya akan disimpan dalam L_{16} dan R_{16}.

III.2. Algoritma TEA

Bilangan delta berasal dari *golden number*, digunakan delta=$(\sqrt{5}-1)2^{31}$. Suatu bilangan delta ganda yang berbeda digunakan dalam setiap roundnya sehingga tidak ada bit dari perkalian yang tidak berubah secara teratur. Berbeda dengan sruktur feistel yang semula hanya mengoperasikan satu sisi yaitu sisi sebelah kanan dengan sebuah fungsi F, pada algoritma TEA kedua sisi dioperasikan dengan sebuah fungsi yang sama. Struktur penyandian TEA pada gambar 2 a).

Proses diawali dengan *input-bit* teks terang sebanyak 64-bit[5]. Kemudian 64-bit teks terang tersebut dibagi menjadi dua bagian, yaitu sisi kiri (L_0) sebanyak 32-bit dan sisi kanan (R_0) sebanyak 32-bit. Setiap bagian teks terang akan dioperasikan sendiri-sendiri. R_0 (z) akan digeser kekiri sebanyak empat kali dan ditambahkan dengan kunci k(0). Sementara itu z ditambah dengan sum (delta) yang merupakan konstanta. Hasil penambahan ini di-XOR-kan dengan penambahan sebelumnya. Kemudian di-XOR-kan dengan hasil penambahan antara z yang digeser kekanan sebanyak lima kali dengan kunci k(1). Hasil tersebut kemudian ditambahkan dengan L_0 (y) yang akan menjadi R_1.

Sisi sebelah kiri akan mengalami proses yang sama dengan sisi sebelah kanan. L_0 (y) akan digeser kekiri sebanyak empat kali lalu ditambahkan dengan kunci k(2). Sementara itu, Y ditambah dengan sum (delta). Hasil penambahan ini di-XOR-kan dengan penambahan sebelumnya. Kemudian di-XOR-kan dengan hasil penambahan antara Y yang digeser ke kanan sebanyak lima kali dengan unci k(3). Hasil tersebut kemudian ditambahkan dengan R_0 (Z) yang akan menjadi L_1.



a) b)

Gambar 2. a). Algoritma TEA, b). Satu cycle TEA (dua round)

Struktur dari penyandian dengan algoritma untuk satu cycle (dua round) dapat dilihat pada gambar 2 b). Berikut adalah langkah langkah penyandian dengan algoritma TEA dalam satu cycle (dua round) [7]:

1. Pergeseran *(shift)*

   Blok teks terang pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.

2. Penambahan

   Setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k(0)-k(3). Sedangkan Y dan Z awal akan ditambahkan dengan sum (delta).

3. Peng-XOR-an

   Setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an dengan rumus untuk satu *round* adalah sebagai berikut :

y = y + (((z<<4)+k(0))^z + sum^((z>>5)+k(1)))

z = z + (((y<<4)+k(2)^y + sum^((y>>5)+k(3))), dalam hal ini sum = sum+delta.

Hasil penyandian dalam satu *cycle* satu blok teks terang 64-bit menjadi 64-bit teks sandi adalah dengan menggabungkan y dan z. Untuk penyandian pada *cycle* berikutnya y dan z ditukar posisinya, sehingga y_1 menjadi z_1 dan z_1 menjadi y_1 lalu dilanjutkan proses seperti langkah-langkah diatas sampai dengan *16 cycle (32 round).*

4. Key Schedule

   Pada algoritma TEA, *key schedule*-nya sangat sederhana. Yaitu kunci k(0) dan k(1) konstan digunakan untuk *round* ganjil sedangkan kunci k(2) dan k(3) konstan digunakan untuk *round* genap.

5. Dekripsi

   Dalam proses dekripsi sama halnya seperti pada proses penyandian yang berbasis *feistel cip*her lainnya. Yaitu pada prinsipnya adalah sama pada saat proses enkripsi. Namun hal yang berbeda adalah penggunaan teks sandi sebagai *input* dan kunci yang digunakan urutannya dibalik. Pada proses dekripsi semua *round* ganjil menggunakan k(1) terlebih dahulu kemudian k(0), demikian juga dengan semua *round* genap digunakan k(3) terlebih dahulu kemudian k(2).

   Pada rumus enkripsi diketahui :

   $L_0 = L_0 + f (R_0 , k(0), k(1), sum)$ dan $R_0 = R_0 + f (L_0, k(2), k(3), sum)$

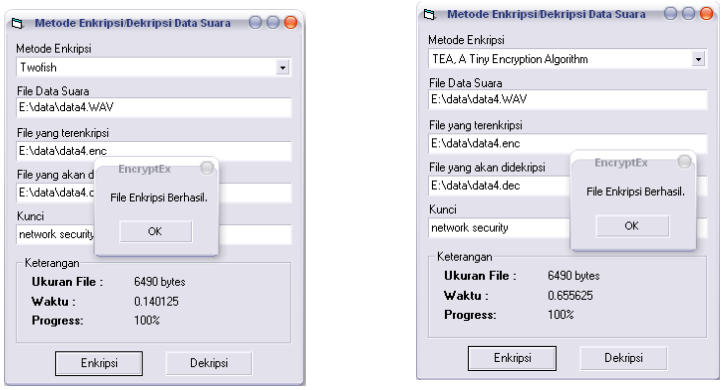
   Sehingga untuk proses dekripsi digunakan rumus :

   $L_0 = L_0 + f (R_0 , k(1), k(0), sum)$ dan $R_0 = R_0 + f (L_0, k(3), k(2), sum)$

IV. Implementasi

Tulisan ini diimplementasikan dengan perangkat lunak Visual Basic 6.0 dengan menggunakan tiga jenis format data suara yang berbeda (mp3, WAV,MID). Hasil enkripsi dan dekripsi masing-masing algoritma ditunjukkan pada sub-bab berikut.

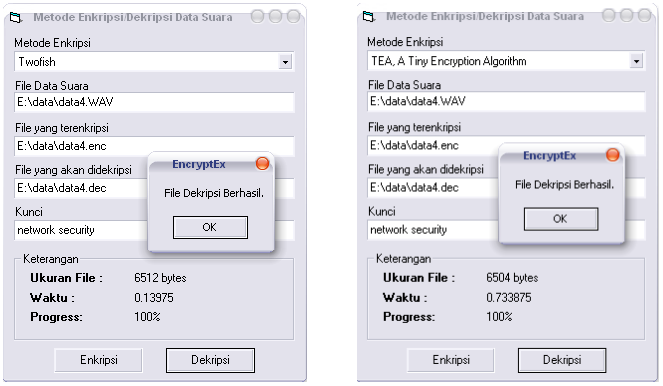
IV.1. Enkripsi Dengan Algoritma Twofish dan Enkripsi Dengan Algoritma TEA



a) b)

Gambar 3. a) dokumen suara dengan format WAV, b) dokumen suara dengan format WAV

IV.2. Dekripsi Dengan Algoritma Twofish dan Dekripsi Dengan Algoritma TEA



a) b)

Gambar 4. a) dokumen suara dengan format WAV, dan b) dokumen suara dengan format WAV

V. Hasil Dan Pembahasan

Hasil dari implementasi algoritma diberikan dalam table 1, 2 dan 3, serta gambar 5 di bawah ini

Tabel 1. Perbandingan hasil implementasi pada format mp3

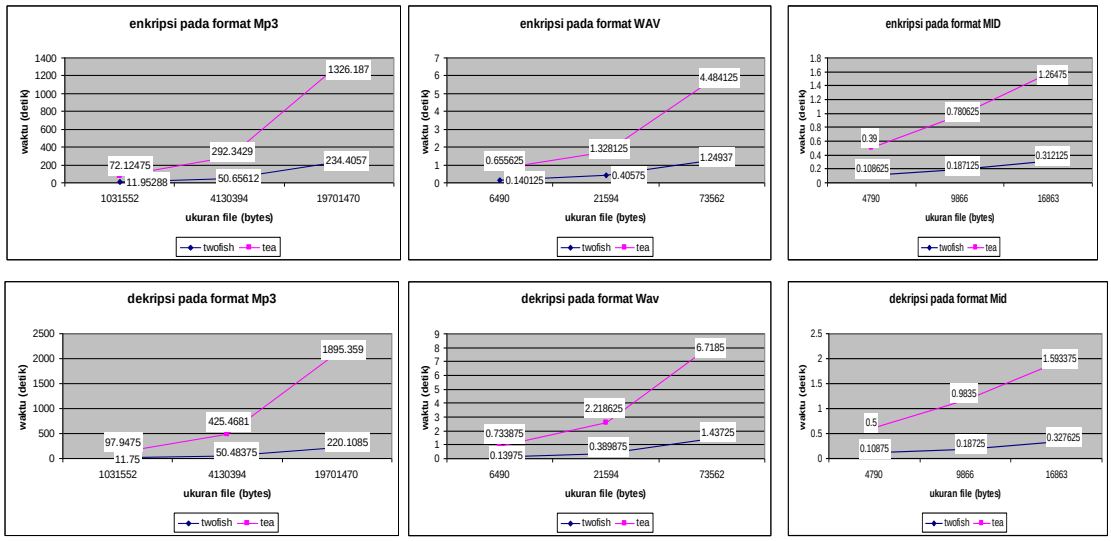
Dokumen	Enkripsi (detik)		Dekripsi (detik)	
	Twofish	TEA	Twofish	TEA
Data1	11.95288	72.12475	11.75	97.9475
Data2	50.65612	292.3429	50.48375	425.4681
Data3	234.4057	1326.187	220.1085	1895.359

Tabel 2. Perbandingan hasil implementasi pada format WAV

Dokumen	Enkripsi (detik)		Dekripsi (detik)	
	Twofish	TEA	Twofish	TEA
Data4	0.140125	0.655625	0.13975	0.733875
Data5	0.40575	1.328125	0.389875	2.218625
Data6	1.24937	4.484125	1.43725	6.7185

Tabel 3. Perbandingan hasil implementasi pada format MID

Dokumen	Enkripsi (detik)		Dekripsi (detik)	
	Twofish	TEA	Twofish	TEA
Data7	0.108625	0.39	0.10875	0.5
Data8	0.187125	0.780625	0.18725	0.9835
Data9	0.312125	1.26475	0.327625	1.593375



Gambar 5. grafik perbandingan hasil enkripsi mp3, WAV, MID dan grafik perbandingan hasil dekripsi mp3, WAV, MID

Dari tabel 1, 2 dan 3 serta gambar 8 terlihat bahwa :

1. Hasil enkripsi dan dekripsi dari kedua algoritma, Twofish dan TEA (tiny encryption algorithm) tidak tergantung pada format dokumennya tetapi pada besarnya ukuran dokumen yang diolah.
2. Waktu enkripsi algoritma Twofish relative lebih cepat dibandingkan algoritma TEA (untuk dokumen dengan format MP3 yang berukuran 4130394 *bytes*, algoritma TEA membutuhkan waktu enkripsi sebesar 292.3429 detik.sementara algoritma Twofish hanya butuh 50.65612 detik.
3. Begitupula dengan waktu untuk mendekripsi, algoritma TEA membutuhkan waktu yang jauh kebih lama dibandingkan algoritma Twofish (untuk dokumen dengan format MP3 yang berukuran 4130394 *bytes*, algoritma TEA membutuhkan waktu dekripsi sebesar 425.4681 detik.sementara algoritma Twofish hanya butuh 50.48375 detik.

VI. Kesimpulan

Karena kekuatan suatu algoritma diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data chiperteks menjadi plainteksnya (dekripsi), Sehingga dapat disimpulkan bahwa untuk enkripsi dokumen yang berupa suara, tingkat keamanan algoritma TEA jauh lebih baik dibandingkan algoritma Twofish, karena waktu dekripsi dengan menggunakan algoritma TEA membutuhkan waktu yang lebih lama dibandingkan dengan menggunakan algoritma Twofish.

VII. Daftar Pustaka

1. Kelsey, John, et al. 1998. *Twofish*: a 128-Bit Block Cipher.
2. Ratih, 2007. "*Studi dan Implementasi Enkripsi Pengiriman Pesan Suara Menggunakan Algoritma Twofish*", National Conference On Computer Science & Information Technology VII,
3. Schneier, Bruce, 1996. *Applied Cryptography 2nd*, John wiley & Sons.
4. http://www.counterpane.com/twofish.html.
5. http://www.cl.cam.ac.uk/Research/Papers/djw-rmn/djw-rmn-tea.html.
6. http://www.schneier.com/twofish.html.
7. http://v318.wordpress.com/category/sharing-ilmu-pengetahuan