Optimasi Enkripsi Teks Menggunakan AES dengan Algoritma Kompresi Huffman

Edmund Ophie - 13512095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*edmund.ophie@yahoo.com*

Komunikasi telah menjadi bagian penting dari kehidupan kita sehari-hari. Komunikasi menjadi sarana untuk mengirimkan informasi antara pengirim dan penerima. Untuk memberikan keamanan dan menjaga kerahasiaan konten dari informasi yang dikirim, maka komunikasi harus dienkripsi. Teknik enkripsi pun berkembang dan semakin bervariasi.

Namun dalam perkembangannya untuk melakukan enkripsi pada pesan atau data yang akan dikirim memakan waktu. Waktu yang dibutuhkan berbanding lurus dengan ukuran pesan yang akan dikirim. Hal ini dikarenakan untuk mengenkripsi pesan ada algoritma dan komputasi yang perlu dilakukan untuk melakukan enkripsi dan algoritma ini semakin hari semakin kompleks untuk menjamin keamanan data yang dienkripsi. Selain itu dalam proses enkripsi biasanya ukuran data yang dihasilkan skurang lebih ama dengan data awal sebelum dienkripsi. Oleh karena itu, diperlukan teknik yang dapat mempersingkat waktu enkripsi serta memperkecil keluaran data hasil enkripsi. Salah satu teknik yang dapat dilakukan adalah menggabungkan enkripsi dengan teknik kompresi data. Salah satu teknik kompresi yang ada adalah Huffman Coding.

Pada makalah ini akan dibahas penerapan teknik kompresi Huffman Coding pada enkripsi teks menggunakan algoritma kriptografi simetris AES serta analisa dampak penerapannya.

***Kata kunci* — Huffman coding, kompresi, enkripsi, kriptografi**

I. PENDAHULUAN

Salah satu cara untuk menjaga keamanan informasi dalam berkomunikasi adalah dengan metode enkripsi. Teknik enkripsi pun terus berkembang. Variasi algoritma enkripsi pun semakin bertambah seiring perkembangan waktu. Penelitan-penelitan terus dilakukan untuk mencari algoritma yang lebih baik dan yang lebih aman. Melihat teknik enkripsi yang sudah ada, algoritma yang diciptakan biasanya akan cenderung bertambah kompleksitasnya untuk mencapai level keamanan yang lebih baik dari waktu ke waktu. Hal ini biasanya dilakukan dalam rangka meningkatkan keamanan serta menjaga data dari kemungkinan dilakukannya kriptanalisis. Selain kompleksitas algoritma yang terus meningkat, di era serba internet sekarang ini, ukuran data juga ikut bertambah besar. Perkembangan internet telah memaksa kita untuk mendigitalisasi hampir semua pekerjaan kita. Dengan demikian kebutuhan akan media penyimpanan pun semakin besar. Oleh karena itu ukuran data yang perlu dienkripsi pun semakin besar. Ukuran data yang semakin besar tentu akan berbanding lurus dengan waktu yang dibutuhkan untuk melakukan enkripsi.

Salah satu solusi untuk mengatasi masalah ini adalah dengan melakukan kompresi pada data yang akan dienkripsi. Dengan dilakukannya kompresi pada data maka dapat mempercepat proses enkripsi serta memperkecil ukuran ciphernya. Algoritma kompresi data pun beragam seperti Shannon-Fano Coding. Huffman Coding, Lempel-Ziv-Wlch, Lempel-Ziv-Markov, dst. Dalam makalah ini Huffman Coding akan dipilih karena algoritmanya yang lossless dan penerapannya yang relatif lebih mudah. Lossles yang dimaksud adalah ketika hasil kompresi dapat dibalikkan menjadi seperti keadaan semula saat didekompresi tanpa kehilangan informasi yang terkandung di dalamanya. Namun algoritma Huffman Coding memiliki keterbatasan yaitu hanya dapat digunakan untuk melakukan kompresi pada data berbentuk teks saja. Hal ini dikarenakan algoritma ini menggunakan probabilitas kemunculan karakter untuk memperkecil ukuran datanya. Untuk enkripsi teks akan digunakan algoritma kriptografi simetris yaitu AES. Pemilihan kriptografi simetris dikarenakan biasanya algoritma kriptografi asimetris seperti RSA tidak digunakan untuk mengenkripsi teks yang panjang. Algoritma kriptografi asimetris biasa digunakan hanya untuk mengenkripsi kunci simetris yang akan dipertukarkan antara pengirim dan penerima pesan oleh karena komputasi yang dibutuhkannya cukup berat. Namun tidak menutup kemungkinan metode kompresi ini digunakan pada algoritma kriptografi asimetris seperti RSA, El Gamal, dll.

Dengan menerapkan algoritma kompresi ini sebelum dienkripsi maka dapat mempersingkat waktu yang dibutuhkan untuk melakukan enkripsi dan memperkecil ukuran cipher yang perlu dikirimkan melalui saluran komunikasi. Selain itu, dengan melakukan kompresi plaintext sebelum menerapkan enkripsi pada teks juga dapat mempersulit kriptanalis untuk memecahkan teks yang dienkripsi dari serangan-serangan kriptanalisis seperti . Dengan melakukan kompresi sebelum enkripsi

maka penyerang akan sulit melakukan kriptanalisis diferensial pada cipher karena akan menjadi lebih sulit bagi penyerang untuk mendeduksi hubungan antara plaintext dan ciphertext.

II. DASAR TEORI

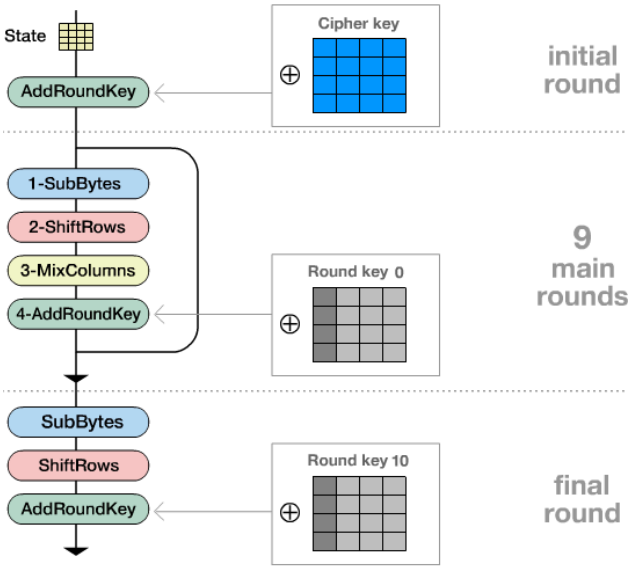
II.1 Advance Encryption Standard (AES)

Algoritma AES dibuat oleh Vincent Rijmen dan Joan Daemen. AES merupakan algoritma kriptografi simetri yang berbasis cipher blok yang teridiri dari kumpulan bit-bit. Pada AES kunci memiliki panjang yang fleksibel dengan kelipatan 32 bit, seperti 128, 192, atau 256 bit. Namun pada kenyataannya AES dengan panjang kunci 192 bit sangat jarang ditemukan. Kunci dengan panjang 128 atau 256 bit lebih sering dipakai.

Algoritma AES memecah bit-bit pada plaintext dan mengelompokkannya menjadi beberapa blok yang terdiri dari kumpulan-kumpulan bit dengan jumlah yang sama. Blok-blok ini nantinya akan diproses satu per satu sesuai dengan operasi-operasi yang ada pada AES. Setiap blok pada AES memiliki bit berjumlah 128 bit.

Pada algoritma AES-128 bit, pertama-tama blok plainteks diambil dan disimpan dalam sebuah matriks berukuran 4x4. Cipher key dari AES juga disimpan dalam sebuah matriks berukuran 4x4. Matriks ini berisi byte-byte dari blok plainteks dan cipher key. Selanjutnya AES memiliki 3 tahap yang disebut tahap awal, tahap 9-round, dan tahap akhir.

Di tahap awal, blok plainteks yang telah disimpan dalam matriks akan di XOR kan dengan cipher key. Hasil dari tahap awal ini kemdian akan diproses selanjutnya dengan 4 macam transformasi di tahap selanjutnya, yaitu tahap 9-round. Adapun transformasinya yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey.



Gambar 1. Tahapan enkripsi pada AES

Di tahap 9-round pertama kali dilakukan operasi SubBytes. Pada transformasi subbytes, setiap elemen matriks pada blok plainteks akan ditukar dengan elemen di S-box. S-box adalah matriks berukuran 16x16 yang elemennya menyimpan nilai byte. Misal elemen pada matriks blok plainteks bernilai f3, maka nilai dari elemen tsb akan diganti dengan elemen pada baris ke f dan kolom 3 di matrisk S-box. Selanjutnya akan dilakukan transformasi ShiftRows pada tiga baris terakhir matriks dengan melakukan pergeseran elemen-elemennya ke kiri secara sirkular. Kemudian pada transformasi MixColumns setiap kolom array pada matriks plainteks dikalikan dengan polinom a(x) mod (x^4+1). Selanjutnya hasilnya akan ditransformasikan lagi dengan operasi AddRoundKey dimana setiap kolom pada matriks plainteks di XOR kan dengan round key.

Pada tahap terakhir, hasil dari tahap 9-round akan dilakukan transformasi lagi mirip dengan yang ada di tahap 9-round hanya saja tanpa transformasi MixColumns. Semua tahapan ini akan diulang hingga semua blok-blok plainteks telah selesai diproses.

II.2 Huffman Coding

Algoritma Huffman merupakan salah satu algoritma kompresi data lossless. Lossless berarti dekompresi akan menghasilkan file yang sama persis seperti keadaan aslinya tanpa adanya kehilangan data/informasi yang dikandungnya. Algoritma Huffman bukanlah algoritma yang memberikan rasio kompresi terbaik jika dibandingkan dengan algoritma kompresi lainnya. Namun algoritma ini cukup sederhana untuk diterapkan. Algoritma ini ditemukan oleh David A. Huffman pada tahun 1950.

Ide dasar algoritma ini cukup sederhanam yaitu dengan mengkodekan setiap karakter ke dalam representasi bit. Encoding setiap karakter teks memiliki jumlah bit yang sama, biasanya 8 bit untuk setiap karakter ASCII. Ide dasar dari algoritma Huffman adalah dengan memetakan frekuensi kemunculan setiap huruf dan menggantikan representasi bit dari huruf yang paling sering muncul menjadi bit yang lebih pendek dan menggantikan huruf yang paling jarang muncul dengan bit yang paling panjang. Semakin sering suatu karakter muncul, maka semakin pendek representasi bitnya dan sebaliknya. Dengan demikian, jumlah bit pada suatu data akan semakin sedikit dan mengakibatkan ukuran data semakin kecil pula.

Simbol	Kode ASCII
A	01000001
B	01000010
C	01000011
D	01000100

Gambar 2. Tabel kode ASCII

Sebagai contoh representasi bit untuk string "ABACCDA" tanpa menggunakan algoritma huffman adalah

01000001010000010010000010100000110100000110100010001000001, setara dengan 7x8 = 56 bit (7 byte). Jika menggunakan algoritma Huffman maka representasi bit nya menjadi 0110010101110, hanya 13 bit. Terlihat bahwa penerapan algoritma Huffman mampu mengkompresi data lebih dari 76%.

Untuk mengubah suatu representasi bit dari suatu string dapat menggunakan sebuah tree dan hal pertama yang harus dilakukan terlebih dahulu adalah mencari frekuensi kemunculan dari setiap karakter. Misal pada kasus diatas representasi kemunculan dari setiap karakter pada string "ABACCDA" adalah:

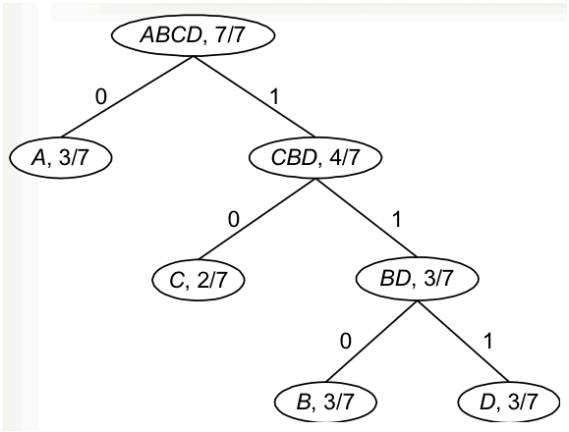
Karakter	Frekuensi	Peluang
A	3	3/7
B	1	1/7
C	2	2/7
D	1	1/7

Tabel 1. Frekuensi kemunculan karakter

Setelah melakukan pemetaan frekuensi karakter, terapkan algoritma huffman yakni sebagai berikut:

1. Pilih dua karakter dengan peluang kemunculan paling kecil. Pada contoh diatas adalah B dan D. Kombinasikan kedua karakter tadi untuk menjadi simpul orangtua dari daun B dan D. Beri nama simpul tsb BD dan simpul tsb akan memiliki peluang 1/7+1/7 = 2/7.
2. Pilih karakter berikutnya yang memiliki peluang terkecil dan letakkan karakter tsb daun pada sisi kiri pohon Huffman.
3. Ulangi langkah ke 1 dan 2 sampai semua karakter telah diproses.

Untuk setiap sisi yang berada disebelah kiri pohon beri tanda 0 dan untuk setiap sisi sebelah kanan pohon beri tanda 1. Untuk mencari representasi bit dari suatu karakter dapat dilakukan dengan menelusuri pohon hingga mencapai daun dari karakter yang dicari. Representasi bit dari karakter tsb adalah penomoran setiap jalur-jalur yang telah ditelusuri untuk mencapai daun dari karakter tsb.



Gambar 3. Pohon Huffman yang dihasulkan untuk string "ABACCDA"

Dapat dilihat pada gambar diatas, representasi bit dari huruf A adalah 0, C adalah 10, B adalah 110, D adalah 111. Setelah menggunakan algoritma huffman, jumlah bit menjadi berkurang secara signifikan.

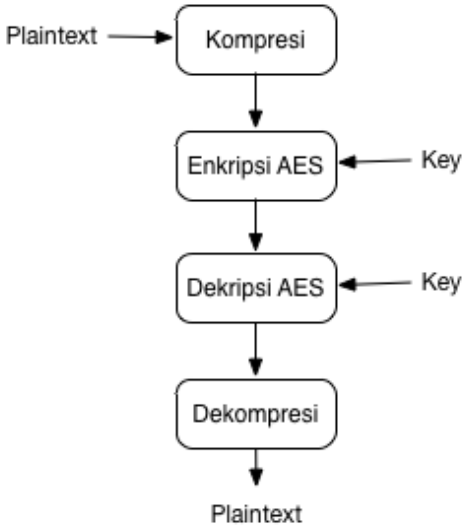
Pada perkembangannya algoritma Huffman memiliki banyak variasi dan peningkatan dari algoritma aslinya dalam usaha untuk mendapatkan hasil kompresi terbaik, diantaranya adalah n-ary Huffman Coding dan Adaptive Huffman Coding.

III. ANALISA DAN PEMBAHASAN

Untuk menerapkan enkripsi dan kompresi sebaiknya kompresi dilakukan terlebih dahulu sebelum sebuah data dienkripsi. Hal ini dikarenakan kompresi bekerja dengan mencari suatu pola tertentu pada data. Jika kita melakukan enkripsi kemudien di kompresi, data yang telah terkenkripsi akan menjadi acak dan tidak beraturan lagi isinya sehingga ketika dilakukan kompresi hasilnya tidak akan maksimal.

Untuk kemudahan pada makalah ini maka akan digunakan algoritma Huffman Coding dasar dan AES-128 bit. Dalam implementasi Huffman Coding kita harus membuat tabel frekuensi kemunculan setiap karakter terlebih dahulu. Frekuensi kemunculan karakter ini dapat dibuat dengan statis ataupun dinamis. Untuk menggunakan frekuensi kemunculan yang statis, kita dapat mencari referensi seberapa sering sebuah karakter muncul dalam sebuah teks di bahasa tertentu. Misalnya frekuensi kemunculan huruf E adalah yang paling sering di bahasa inggris dengan kemunculan sekitar 12%. Namun jika menggunakan tabel frekuensi yang statis memiliki kekurangan karena bisa jadi frekuensi kemunculan huruf tsb tidak akurat dan belum tentu pesan yang dienkripsi selalu berbahasa Inggris. Oleh karena itu sebaiknya digunakan metode tabel frekuensi yang dinamis. Frekuensi kemunculan karakter dihitung satu per satu hingga semuanya telah terdata frekuensinya.

Dari tabel frekuensi ini kemudian dapat dibuat pohon Huffman yang akan digunakan untuk mencari representasai bit yang baru dari setiap karakter.



Gambar 4. Alur kerja program kompresi dan enkripsi

Hasil kompresi kemudian di enkripsi dengan AES. Secara teoritis waktu yang dibutuhkan untuk enkripsi maupun dekripsi ini seharusnya lebih cepat dibanding tanpa kompresi. Jika ukuran cipher file yang didapat berkurang signifikan dari palinteksnya maka kemungkinan besar program kompresi dan enkripsi yang dibuat telah berhasil.

Hasil kinerja enkripsi Huffman Coding ini terbilang cukup baik. Untuk setiap kali percobaan eknripsi, rasio kompresi yang dihasilkan rata-rata ditas 40% dan di beberapa kasus bahkan bisa mencapai 50-60%. Jika plainteks yang akan dikompresi cukup kecil maka rasio kompresi yang dihasilkan tidak akan terlalu tinggi. Namun semakin panjang plainteks yang akan dienkripsi maka akan memberikan rasio kompresi yang semakin tinggi pula. Misal dalam kasus kompresi string lorem ipsum sebagau plainteks sebanyak 1200 karakter hanya memberikan rasio kompresi sekitar 26%. Ketika dilakukan kompresi terhadap string lorem ipsum sebanyak 24000 karakter maka rasio kompresi meningkat cukup drastis hingga 46%.

Ketika dikompresi jumlah bit yang dihasilkan pun ikut berkurang. Pada kasus kompresi teks lorem ipsum sebanyak 24000 karakter jumlah bit berkurang dari 192 ribu bit menjadi hanya sekitar 100 ribu bit. Menurunnya jumlah bit akan ikut berdampak pada kecepatan mengenkripsi teks tsb. Ketika dilakukan enkripsi tanpa kompresi maka waktu yang dibutuhkan sekitar 2.2 ms. Namun setelah dilakukan kompresi, waktu yang dibutuhkan untuk melakukan enkripsi menjadi hanya 0.35 ms. Ukuran ciphertext pun ikut berkurang. Dampak yang diberikan oleh kompresi ini terlihat cukup signifikan. Selain mempercepat waktu enkripsi, waktu dekripsi pun ikut berkurang seiring berkurangnya ukuran ciphertext.

Ukuran Plaintext (byte)	Ukuran Ciphertext (byte)	Rasio Kompresi	Durasi enkripsi tanpa kompresi (ms)	Durasi enkripsi dengan kompresi (ms)
1222	902	26.18%	0.52	0.30
4091	2414	40.99%	1.25	0.10
6559	3731	43.11%	1.87	0.12
12793	7037	44.99%	2.92	0.15
24610	13277	46.05%	2.70	0.44
97029	52198	46.20	4.52	1.66

Tabel 2. Data hasil percobaan kompresi dan enkripsi

Percobaan diatas dilakukan pada komputer Macobook Pro 2012 intel i5 dan memory 4 GB pada sistem operasi OSX Yosemite. pada Dapat terlihat jelas bahwa dengan dilakukannya kompresi sebelum enkripsi telah mempercepat waktu enkripsi cukup signifikan bahkan ukuran file berkurang cukup drastis hampir setengahnya. Rata-rata rasio kompresi yang diberikan berkisar antara 40-46%. Waktu untuk melakukan dekripsi pun ikut berkurang seiring menurunnya ukuran ciphertext.

IV. KESIMPULAN DAN SARAN

Penerapan kompresi dengan Huffman Coding sebelum melakukan enkripsi memberikan dampak signifikan baik dari efisiensi waktu maupun ruang yang dibutuhkan. Dengan melakukan kompresi maka akan memperkecil ukuran plaintext yang menjadi masukan pada proses enkripsi. Kompresi menggunakan Huffman mampu memperkecil ukuran keluaran menjadi hampir setengahnya dan mempercepat proses enkripsi dan dekripsi. Selain itu, kompresi juga membuat penyerang sulit untuk melakukan kriptanalis dengan menggunakan metode kriptanalisis diferensial dikarenakan keterhubungan antara plaintext dan ciphertext akan menjadi lebih sulit dideduksi setelah dilakukannya kompresi.

Meskipun kompresi Huffman Coding berhasil mempercepat proses enkripsi dan dekripsi, waktu yang dibutuhkan untuk melakukan kompresi dan dekompresi masih harus dipertimbangkan karena pada praktiknya kompresi dan dekompresi juga membutuhkan komputasi yang cukup intentsif. Waktu yang dibutuhkan untuk melakukan kompresi dan dekompresi cenderung lebih lama dibandingkan waktu yang dihemat ketika menggunakan enkripsi terkompresi. Oleh karena itu penggunaan teknik Huffman Coding dan kriptografi AES harus digunakan pada konteks yang tepat sehingga bisa memberikan dampak yang efektif dan efisien.

Untuk kedepannya teknik kompresi ini dapat dikembangkan lagi dengan menggunakan variasi algoritma Huffman Coding, yaitu Adaptive Huffman Coding atau dapat menggunakan algoritma lain yang dinilai lebih mangkus seperti Lempel-Ziv-Welch. Dengan mengkombinasikan algoritma kriptografi AES dengan algoritma kompresi yang lain diharapkan kedepannya bukan hanya teks saja yang dapat dikompresi dan dienkripsi tetapi semua berkas baik seperti gambar maupun video.

REFERENSI.

[1] Munir, Rinaldi. Matematika Diskrit, Penerbit INFORMATIKA, Bandung, 2005.

[2] Sangwan Nigam. Text Encryption with Huffman Compression, International Journal of Computer Application, volume 54, No 6, September 2012.

[3] Ardhin, Muhammad. Adaptive Huffman Coding Sebagai Variasi Huffman Coding, Institut Teknologi Bandung, 2010. http://www.cs.utsa.edu/~wagner/laws/huffman.html Diakses 10 Mei 2015 17:00

[4] http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf Diakses 9 Mei 2015 20:00

[5] www.formaestudio.com/.../archivos/Rijndael_**Animation**_v4_eng.swf Diakses 9 Mei 2015 20:15

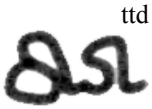
[6] http://blog.superuser.com/2011/03/21/compression-and-encryption/ Diakses 9 Mei 2015 19:00

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2015

ttd



Edmund Ophie - 13512095