Indonesian Journal of Data and Science ISSN: 2715-9930
Vol 1, No 1, Maret 2020, pp. 23-28

Implementasi Kombinasi Algoritma Enkripsi Rivest Shamir Adleman (Rsa) dan Algoritma Kompresi Huffman Pada File Document

Andi Inayah Auliyah[a,1]

[a] *Universitas Muslim Indonesia, Jln. UripSumoharjo Km.5 , Makassar 90231, Indonesia*
[1] *Inayahauliyah97@gmail.com*

INFORMASI ARTIKEL

Diterima : 10 Januari 2020
Direvisi : 13 Februari 2020
Diterbitkan : 31 Maret 2020

Kata Kunci:
Rhivest Shamir Adleman
Huffman
Enkripsi
Konpresi
Keamanan data

ABSTRAK

Enkripsi yaitu proses merubah data asli menjadi data samaran dan dekripsi yaitu proses pengembalian data samara menjadi data asli kembali. Terlepas dari masalah keamanan data, masalah lain yang harus kita perhatikan adalah ukuran data atau informasi yang semakin besar oleh karena itu perlu adanya upaya untuk menanggulangi masalah tersebut salah satunya yaitu dengan menggunkan algoritma kombinasi. Maka dari itu penulis mengkombinasikan kedua metode tersebut yaitu metode Enkripsi dan metode Kombinasi untuk melihat manakah cara yang lebih efisien digunakan antara dienkripsi terlebih dahulu kemudian dikompresi atau sebaliknya.Dengan mengkombinasikan kedua algoritma tersebut penulis menemukan bahwa hasil yang diberikan dari kombinasi tersebut tidak maksimal karena ukuran file hasil proses menjadi lebih besar dibandingkan file asli sebelum diproses. Dari hasil analisis data dalam penelitian ini, maka untuk persentasi keberhasilan prosesnya dapat dibagi menjadi empat bagian.yaitu proses enkripsi-kompresi,kompresi-enkripsi,dekompresi-dekripsi dan dekripsi-dekompresi dengan tingkat keberhasilan 100%. Jika dilihat dari proses tingkat keberhasilan sangat memuaskan namun jika dilihat keberhasilan mengembalikan isi file itu dapat dibagi menjadi dua bagian yaitu proses dekompresi-dekripsi persentasi keberhasilannya itu sebanyak 33,34% dan pesentasi kegagalan sekitar 66,67% sedangkan untuk proses dekripsi-dekopresi persentasi keberhasilan sekitar 33,34% dan persentasi kegagalan sekitar 66,67%.

I. Pendahuluan

Perkembangan teknologi yang semakin meningkat tiap harinya, membuat data atau informasi juga menjadi tidak terpisahkan dari aspek kehidupan manusia [1]. Pentingnya nilai suatu informasi pada setiap aspek dapat menimbulkan suatu permasalahan pada keamanan data atau informasi. Salah satunya yaitu usaha pencurian yang dilakukan oleh pihak ketiga atau pihak yang tidak bertanggung jawab terhadap data atau informasi yang akan kita kirim. Oleh karena itu perlu adanya upaya untuk mencegah hal tersebut, salah satu caranya yaitu dengan menggunakan metode enkripsi [2].

Enkripsi (encryption) yaitu proses merubah data asli (plaintext) menjadi data samaran (chipertext) dan dekripsi (decryption) yaitu proses pengembalian chipertext menjadi plaintext kembali. Terlepas dari masalah keamanan data, masalah lain yang harus kita perhatikan adalah ukuran data atau informasi, salah satu contohnya dapat kita lihat pada sebuah file chipertext yang merupakan hasil enkripsi yang terdiri dari kombinasi plaintext dan kunci yang menyebabkan kapasitasnya lebih besar dibandingkan dengan file plaintextnya. Hal itu dapat kita atasi dengan memperkecil atau memadatkan ukuran data yang disebut dengan teknik kompresi[3].

Berdasarkan permasalahan di atas, penulis akan mengkombinasikan dua metode yaitu metode enkripsi dan metode kompresi dengan menggunakan algoritma RSA untuk metode enkripsi dan algoritma Huffman untuk metode kompresi. Alasan penulis menggunakan kedua metode tersebut yaitu karena algoritma RSA memiliki mekanisme kerja yang cukup sederhana dan mudah dimengerti, tetapi kokoh. Dikarenakan kunci yang digunakan untuk mengenkripsi dan mendekripsi berbeda, sedangkan alasan untuk algoritma kompresi Huffman karena algoritma Huffman merupakan jenis algoritma kompresi yang paling popular digunakan

untuk mengkompresi data teks, Huffman juga merupakan algortma kompresi jenis lossless compression yang dalam pengkompresiannya tidak menghilangkan satu byte data pun[4].

Sehingga dengan alasan tersebut penulis berharap bahwa dengan menggabungkan dua metode dari hasil penelitian ini maka ukuran data akan lebih kecil, begitu juga keamanan data dari hasil enkripsi data bisa lebih terjamin. Oleh karena itu penulis bermaksud untuk melakukan penelitian yang berjudul : "Implementasi Kombinasi Algoritma Enkripsi Rivest Shamir Adleman (Rsa) Dan Algoritma Kompresi Huffman Pada File Document "[5].

II. Metode

A. Enkripsi

Enkripsi (encryption) yaitu proses merubah data asli (plaintext) menjadi data samaran (chipertext) dan dekripsi (decryption) yaitu proses pengembalian chipertext menjadi plaintext kembali [6][7].

B. Kompresi

Kompresi data merupakan cabang ilmu komputer yang bersumber dari teori informasi. Teori informasi sendiri adalah salah satu cabang matematika yang berkembang sekitar dekade 1940-an. Tokoh utama dari teori informasi adalah Claude Shannon dari Bell Laboratory. Teori informasi memfokuskan pada berbagai metode tentang informasi termasuk penyimpanan dan pemrosesan pesan. Teori informasi mempelajari pula tentang redundancy (informasi tidak berguna) pada pesan. Semakin banyak redundancy semakin besar pula ukuran pesan dan upaya mengurangi redundancy inilah yang akhirnya melahirkan subjek ilmu tentang kompresi data.

C. Algoritma Rivest Shamir Adleman (RSA)

Menurut Gunawan (2018), RSA adalah algoritma untuk enkripsi kunci publik. Algoritma ini adalah algoritma pertama yang dik etahui paling cocok untuk menandai (signing) dan untuk enkripsi dan salah satu penemuan besar pertama dalam kriptografi kunci publik. RSA masih digunakan secara luas dalam protokol-protokol perdagangan elektronik dan dipercayai sangat aman karena diberikan kunci-kunci yang cukup panjang dan penerapan-penerapannya yang sangat mutakhir.

Algoritma kriptografi RSA merupakan algoritma kriptografi kunci publik (nirsimetri). Ditemukan pertama kali pada tahun 1977 oleh R. Rivest, A. Shamir, dan L. Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut. Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. Kunci publik boleh diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan digunakan untuk proses dekripsi. Algoritma kriptografi RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi.

1) Proses Pembentukan sepasang kunci

Berikut ini adalah proses pembentukan kunci dalam algoritma kriptografi RSA :

1. Memilih dua bilangan prima yang diberi simbol sebagai p dan q (nilai $p \neq q$).
2. Menghitung nilai n = p.q ($p \neq q$, karena jika p = q , maka nilai n = p2 sehingga nilai p dapat diperoleh dengan menarik akar pangkat dua dari n).
3. Hitung $\varphi(n) = (p - 1)(q - 1)$.
4. Memilih kunci publik e yang relatif prima terhadap (n).
5. Bangkitkan kunci privat dengan persamaan e . d ≡ 1 (mod (n)) dimana 1 < < ( ). Perhatikan bahwa persamaan e . d ≡ 1 (mod (n)) ekivalen dengan e . d = 1 + k (n), sehingga untuk mencari nilai d dapat dihitung dengan d = 1 + k (n)/e.

Hasil dari pembentukan pasangan kunci di atas adalah

1. Kunci publik (e, n).
2. Kunci rahasia (d, n) Nilai n tidak bersifat rahasia karena diperlukan pada saat perhitungan proses enkripsi dan dekripsi.

2) Proses Enkripsi

Berikut ini adalah proses enkripsi dalam algoritma kriptografi RSA:

1. Ambil kunci publik penerima pesan e dan modulus n atau (e,n).
2. Pilih plainteks m dan ubah isi pesan m menjadi pesan dengan nilai ASCII.
3. Potong pesan menjadi blok-blok pesan m1, m2, m3, ... dengan nilai setiap bloknya adalah $0 \leq m \leq n - 1$.
4. Setiap blok m dihitung dengan rumus ci = mie mod n.
5. Susun nilai c hasil enkripsi dengan susunan c1, c2, c3,....,cn sehingga diperoleh cipherteks dari pesan m.

3) Proses Dekripsi

Berikut ini adalah proses dekripsi dalam algoritma kriptografi RSA :

1. Ambil pesan (cipherteks) yang telah diterima.
2. Kemudian ambil kunci rahasia d dan modulus n atau (d,n).
3. Potong pesan menjadi blok-blok pesan c1, c2, c3, ... dengan nilai setiap bloknya adalah $0 \leq c \leq n - 1$.
4. Hitung mi = cid mod n.
5. Susun nilai m hasil dekripsi dengan susunan m1, m2, m3,....,mn sehingga diperoleh plainteks (pesan asli) dari cipherteks yang diterima.

D. Algoritma Kompresi Huffman

Algoritma Huffman adalah suatu algoritma kompresi tertua yang disusun oleh David Huffman pada tahun 1952. Algoritma tersebut digunakan untuk membuat kompresi jenis Lossless compression, yaitu pemanpaatan data dimana tidak ada satu byte data yang hilang sehingga data tersebut utuh dan disimpan sesuai dengan aslinya.

Tahapan proses kompresi algoritma Huffman antara lain :

1. Hitung banyaknya jenis karakter dan jumlah dari masing masing karakter yang terdapat dalam sebuah file.
2. Susun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
3. Buat pohon biner berdasarkan berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar dan member kode untuk tiap karakter.
4. Ganti data yang ada dengan kode bit berdasarkan pohon biner.
5. Simpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

Algoritma Huffman adalah algoritma yang dikembangkan David A. Huffman pada 1952. Algoritma Huffmn menggunakan prinsip pengkodean yang mirip dengan kode morse, yaitu tiap karakter (symbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang, karena prosesnya yang menggunakan kode ini, membuat algoritma Huffman sebagai algoritma keluarga dengan variable codeword length Algoritma Huffman termasuk ke dalam kelas algoritma yang menggunakan metode static, yaitu metode yang selalu menggunakan peta kode yang sama. Metode ini membutuhkan dua fase, dimana fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan.

Dalam mengkompresi dilakukan dengan menggunakan pemilihan algoritma Huffman. Hal ini dikarenakan metode Huffman merupakan salah satu teknik kompresi dengan cara melakukan pengkodean dalam bentuk bit untuk mewakili data karakter. Cara kerja atau algoritma metode ini adalah sebagai berikut :

1. Menghitung banyaknya jenis karakter dan jumlah dari masing-masing karakter yang terdapat dalam sebuah file.
2. Menyusun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.

3. Membuat pohon biner berdasarkan urutan karakter dari yang jumlahnya terkecil ke yang terbesar, dan memberi kode untuk tiap karakter.
4. Mengganti data yang ada dengan kode bit berdasarkan pohon biner.
5. Menyimpan jumlah bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menjadi kode bit sebagai data hasil kompresi.

Contoh teknik kompresi dengan menggunakan metode Huffman pada file teks. Misalkan sebuah file teks yang isinya "AAAABBBCCCCCD". File ini memiliki ukuran 13 byte atau satu karakter sama dengan 1 byte. Berdasarkan pada cara kerja di atas, dapat dilakukan kompresi sebagai berikut:

1. Mencatat karakter yang ada dan jumlah tiap karakter. A = 4, B = 3, C = 12, D = 1.
2. Mengurutkan karakter dari yang jumlahnya paling sedikit ke yang paling banyak yaitu : D, B, A, C.
3. Membuat pohon biner berdasarkan urutan karakter yang memiliki frekuensi terkecil hingga yang paling besar.
4. Mengganti data yang ada dengan kode bit berdasarkan pohon biner yang dibuat. Penggantian karakter menjadi kode biner, dilihat dari node yang paling atas atau disebut node akar : A = 01, B = 001, C = 1, D = 000. Selanjutnya berdasarkan pada kode biner masing-masing karakter ini, semua karakter dalam file dapat diganti menjadi : 01010101001001001111110001111111 Karena angka 0 dan angka 1 mewakili 1 bit, sehingga data bit di atas terdiri dari 32 bit atau 4 byte (1 byte = 8 bit).
5. Menyimpan kode bit dari karakter yang frekuensinya terbesar, jenis karakter yang terdapat di dalam file dan data file teks yang sudah dikodekan.

Cara menyimpan data jenis karakter adalah dengan mengurutkan data jenis karakter dari yang frekuensinya paling banyak sampai ke yang paling sedikit, menjadi : [C,A,B,D] File teks di atas, setelah mengalami kompresi, memiliki ukuran sebesar 1 + 4 + 4 = 9 byte. Jumlah ini terdiri dari 1 byte kode karakter yang memiliki frekuensi terendah, 4 jenis karakter = 4 byte dan 4 byte data kode semua karakter.

III. Hasil dan Pembahasan

A. Hasil Pengujian Proses Enkripsi-Kompresi

Pada proses Enkripsi-Kompresi, beberapa file dengan isi dan jenis file yang berbeda diuji dan menunjukkan keberhasilan

Tabel 1. Pengujian enkripsi kompresi

Nama File	Size Awal	Waktu Eksekusi	Size Hasil	Keterangan
File1.txt	654 Bytes	0,03 detik	802 Bytes	Berhasil
File2.txt	866 Bytes	0,13 detik	1,03 KB	Berhasil
File3.txt	1,40 KB	0,11 detik	1,72 KB	Berhasil
File1.docx	11,6 KB	0,84 detik	12,9 KB	Berhasil
File2.docx	11,8 KB	0,81 detik	13,1 KB	Berhasil
File3.docx	12,2 KB	0,84 detik	13,6 KB	Berhasil
File1.pdf	98 KB	6,93 detik	119 KB	Berhasil
File2.pdf	98,8 KB	6,96 detik	120 KB	Berhasil
File3.pdf	100 KB	7,09 detik	122 KB	Berhasil

B. Hasil Pengujian Proses Kompresi-Enkripsi

Pada proses Kompresi-Enkripsi, beberapa file dengan isi dan jenis file yang berbeda diuji dan menunjukkan keberhasilan

Tabel 2. Pengujian proses kompresi enkripsi

Nama File	Size Awal	Waktu Eksekusi	Size Hasil	Keterangan
File1.txt	654 Bytes	0,05 detik	1,29 KB	Berhasil
File2.txt	866 Bytes	0,04 detik	1,63 KB	Berhasil
File3.txt	1,40 KB	0,06 detik	2,59 KB	Berhasil
File1.docx	11,6 KB	0,66 detik	33,7 KB	Berhasil
File2.docx	11,8 KB	0,68 detik	33,9 KB	Berhasil
File3.docx	12,2 KB	0,71 detik	35,1 KB	Berhasil
File1.pdf	98 KB	5,56 detik	301 KB	Berhasil
File2.pdf	98,8 KB	5,68 detik	303 KB	Berhasil
File3.pdf	100 KB	5,80 detik	306 KB	Berhasil

5.2.3 Hasil Pengujian Proses Dekompresi-Dekripsi

Pada proses Dekompresi-Dekripsi, file hasil dari proses Enkripsi-Kompresi dimasukkan untuk diuji dan menunjukkan keberhasilan pada prosesnya, namun beberapa file hasil mengalami kerusakan sehingga tidak dapat dibuka, dan beberapa file hasil yang lain kosong tapi masih dapat dibuka.

Tabel 3. Pengujian dekompresi dekripsi

Nama File	Size Awal	Waktu Eksekusi	Size Hasil	Keterangan
File1.txt	802 Bytes	0,04 detik	654 Bytes	Berhasil
File2.txt	1,03 KB	0,03 detik	866 Bytes	Berhasil
File3.txt	1,72 KB	0,05 detik	1,40 KB	Berhasil
File1.docx	12,9 KB	0,80 detik	11,6 KB	Berhasil (File Rusak)
File2.docx	13,1 KB	0,79 detik	11,8 KB	Berhasil (File Rusak)
File3.docx	13,6 KB	0,83 detik	12,2 KB	Berhasil (File Rusak)
File1.pdf	119 KB	7,02 detik	98 KB	Berhasil (File Kosong)
File2.pdf	120 KB	7,05 detik	98,8 KB	Berhasil (File Kosong)
File3.pdf	122 KB	7,25 detik	100 KB	Berhasil (File Kosong)

1) Hasil Pengujian Proses Dekripsi-Dekompresi

Pada proses Dekripsi-Dekompresi, file hasil dari proses Kompresi-Enkripsi dimasukkan untuk diuji dan menunjukkan keberhasilan pada prosesnya, namun beberapa file hasil mengalami kerusakan sehingga tidak dapat dibuka, dan beberapa file hasil yang lain kosong tapi masih dapat dibuka.

Tabel 4. Pengujian dekripsi dekompresi

Nama File	Size Awal	Waktu Eksekusi	Size Hasil	Keterangan
File1.txt	802 Bytes	0,04 detik	654 Bytes	Berhasil
File2.txt	1,03 KB	0,03 detik	866 Bytes	Berhasil

Nama File	Size Awal	Waktu Eksekusi	Size Hasil	Keterangan
File3.txt	1,72 KB	0,05 detik	1,40 KB	Berhasil
File1.docx	12,9 KB	0,80 detik	11,6 KB	Berhasil (File Rusak)
File2.docx	13,1 KB	0,79 detik	11,8 KB	Berhasil (File Rusak)
File3.docx	13,6 KB	0,83 detik	12,2 KB	Berhasil (File Rusak)
File1.pdf	119 KB	7,02 detik	98 KB	Berhasil (File Kosong)
File2.pdf	120 KB	7,05 detik	98,8 KB	Berhasil (File Kosong)
File3.pdf	122 KB	7,25 detik	100 KB	Berhasil (File Kosong)

IV. Kesimpulan

Berdasarkan hasil penelitian ini, maka penulis dapat menarik beberapa kesimpulan, yaitu :

1. Mengkombinasikan algoritma enkripsi RSA dan algoritma Kompresi Huffman memberikan hasil yang tidak maksimal dalam mengecilkan ukuran file, karena ukuran file hasil proses menjadi lebih besar dibandingkan ukuran file sebelum proses, ini disebabkan karena ketika proses enkripsi, ukuran data menjadi semakin besar.
2. Jika dibandingkan dari segi urutan proses algoritma antara Enkripsi-Kompresi dan Kompresi-Enkripsi, maka yang memberikan hasil lebih efisien adalah urutan Enkripsi-Kompresi, hal ini dikarenakan ukuran file hasil proses bergantung dari proses Kompresi bukan proses Enkripsi yang dalam hal ini malah memperbesar ukuran data.
3. Pada file bertipe .docx dan .pdf yang diuji, isi dari file gagal dikembalikan karena algoritma yang diterapkan pada program yang dibuat penulis digunakan untuk mengolah teks yang diekstrak atau ditarik dari file yang dimasukkan, karena sebab itulah dua tipe tersebut gagal dikembalikan karena teks didalam file tersebut tidak dapat ditarik atau diekstrak oleh php.

Daftar Pustaka

[1] H. Azis, "Network steganography system using covert channel for LSBS stego data on VOIP communication," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 5, pp. 1448–1449, 2019.

[2] H. Azis and F. Fattah, "Analisis Layanan Keamanan Sistem Kartu Transaksi Elektronik Menggunakan Metode Penetration Testing," *Ilk. J. Ilm.*, vol. 11, no. 2, p. 167, 2019.

[3] A. Djamalilleil, M. Muslim, Y. Salim, E. I. Alwi, H. Azis, and Herman, "Modified Transposition Cipher Algorithm for Images Encryption," *Proc. - 2nd East Indones. Conf. Comput. Inf. Technol. Internet Things Ind. EIConCIT 2018*, pp. 1–4, 2018.

[4] Y. Salim and H. Azis, "Metode Digital Watermark Pada File Penelitian Dosen," *Ilk. J. Ilm.*, vol. 9, no. 2, pp. 161–166, 2017.

[5] H. Azis and R. Wardoyo, "Penerapan Network Steganography Menggunakan Metode Modifikasi LACK Dan Layanan Message Authentication Code Pada Voip Network Steganography System with modification of LACK and Message Authentication Code on VoIP," *Semin. Nas. Komun. dan Inform.*, pp. 13–19, 2015.

[6] M. Nazeri, A. Rezai, and H. Azis, "An Efficient Architecture for Golay Code Encoder," *Proc. - 2nd East Indones. Conf. Comput. Inf. Technol. Internet Things Ind. EIConCIT 2018*, pp. 114–117, 2018.

[7] F. Muharram, H. Azis, and A. R. Manga, "Analisis Algoritma pada Proses Enkripsi dan Dekripsi File Menggunakan Advanced Encryption Standard (AES)," *Pros. Semin. Nas. Ilmu Komput. dan Teknol. Inf.*, vol. 3, no. 2, pp. 112–115, 2018.