Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 1, No 1, November 2021
Hal 23−28

Implementasi Algoritma Adaptive Huffman Code Dalam Kompresi File Teks Terenkripsi Algoritma Loki97

Rahmad Syahputra

Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: putracom.putra@gmail.com

Abstrak−Keamanan dan kerahasiaan data merupakan salah satu aspek yang sangat penting dalam sistem informasi pada saat ini. Disebabkan pesatnya perkembangan ilmu pengetahuan dan teknologi yang memungkinkan munculnya teknik-teknik baru, yang disalah gunakan oleh pihak-pihak tertentu yang mengancam keamanan dari sistem informasi tersebut. Jatuhnya informasi ke tangan pihak lain dapat menimbulkan kerugian bagi pemilik informasi. Oleh sebab itu diperlukan suatu cara yang dapat memberikan keamanan untuk data tersebut, salah satunya menggunakan teknik kriptografi. Teknik kriptografi merupakan suatu ilmu atau seni dalam memenuhi aspek keamanan dalam berkomunikasi dengan mengubah isi informasi menjadi informasi yang tidak dapat dimengerti. Salah satu algoritma yang dapat digunakan adalah algoritma Loki97 yang merupakan salah satu algoritma kriptografi yang menggunakan sistem feistel network. Proses yang dilakukan di dalam penelitian ini adalah dengan mengenkripsi file teks terlebih dahulu menggunakan algoritma Loki97, kemudian hasil dari file yang sudah dienkripsi tersebut akan dikompresi menggunakan algoritma Adaptive Huffman Code, sehingga menghasilkan file yang sudah terenkripsi dan terkompresi.

Kata Kunci: Kompresi; File; Teks; Adaptive Huffman Code; Algoritma Lok197

Abstract−Data security and confidentiality is one of the most important aspects in today's information systems. Due to the rapid development of science and technology that allows the emergence of new techniques, which are misused by certain parties that threaten the security of the information system. The fall of information into the hands of other parties can cause harm to the owner of the information. Therefore we need a way that can provide security for the data, one of which is using cryptographic techniques. Cryptography technique is a science or art in fulfilling security aspects in communicating by changing the content of information into incomprehensible information. One of the algorithms that can be used is the Loki97 algorithm, which is a cryptographic algorithm that uses the Feistel network system. The process carried out in this study is to first encrypt the text file using the Loki97 algorithm, then the results of the encrypted file will be compressed using the Adaptive Huffman Code algorithm, resulting in an encrypted and compressed file.

Keywords: Compression; File; Text; Adaptive Huffman Code; Lok197 Algorithm

1. PENDAHULUAN

Keamanan dan kerahasiaan data merupakan salah satu aspek yang sangat penting dalam sistem informasi pada saat ini. Disebabkan pesatnya perkembangan ilmu pengetahuan dan teknologi yang memungkinkan munculnya teknik-teknik baru, yang disalah gunakan oleh pihak-pihak tertentu yang mengancam keamanan dari sistem informasi tersebut. Jatuhnya informasi ke tangan pihak lain dapat menimbulkan kerugian bagi pemilik informasi. Oleh sebab itu diperlukan suatu cara yang dapat memberikan keamanan untuk data tersebut, salah satunya menggunakan teknik kriptografi.

Teknik kriptografi merupakan suatu ilmu atau seni dalam memenuhi aspek keamanan dalam berkomunikasi dengan mengubah isi informasi menjadi informasi yang tidak dapat dimengerti. Salah satu algoritma yang dapat digunakan adalah algoritma Loki97 yang merupakan salah satu algoritma kriptografi yang menggunakan sistem feistel network[1]. Sistem feistel network merupakan suatu struktur simetris yang digunakan untuk membangun blok cipher dengan membagi blok plainteks.

Pengiriman informasi pada saat sekarang ini tidak hanya membutuhkan keamanan, tetapi kecepatan dalam pengiriman juga diperlukan. Agar pengiriman informasi dapat dilakukan dengan cepat, maka diperlukan suatu cara yang dapat memperkecil ukuran file tersebut tanpa mengurangi kualitas file nya, salah satunya dengan menggunakan teknik kompresi, penggunaan teknik kompresi juga membantu mengurangi ruang penyimpanan di dalam memori komputer.

Teknik kompresi merupakan suatu teknik pemampatan data sehingga diperoleh file dengan ukuran yang lebih kecil daripada ukuran aslinya. Teknik ini dapat mengatasi ukuran file yang besar yang menghambat kecepatan dari pengiriman informasi. Salah satu algoritma teknik kompresi yang dapat digunakan adalah algoritma Adaptive Huffman Code yang merupakan pengembangan dari algoritma Huffman, pada Adaptive Huffman Code tidak diperlukan penyimpanan tabel frekuensi simbol-simbol[2].

Proses yang dilakukan di dalam penelitian ini adalah dengan mengenkripsi file teks terlebih dahulu menggunakan algoritma Loki97, kemudian hasil dari file yang sudah dienkripsi tersebut akan dikompresi menggunakan algoritma Adaptive Huffman Code, sehingga menghasilkan file yang sudah terenkripsi dan terkompresi.

2. METODOLOGI PENELITIAN

2.1 File Teks

File teks merupakan *file* komputer yang tersusun atas rangkaian baris teks. Jenis-jenis *file* yang termasuk dalam kategori ini umumnya berisi rangkaian karakter tanpa informasi format visual. Konten *file* kategori ini biasanya merupa- kan catatan atau daftar personal, artikel, buku dan lain sebagainya. *File* teks mirip dengan *file* yang dihasilkan oleh program pengolah kata yang konten utamanya bersifat tekstual[3].

Jenis *file* yang termasuk kategori ini ada banyak. Beberapa diantaranya adalah sebagai berikut[4]:

1. DOC (.doc)
   File DOC merupakan dokumen pengolah kata yang diciptakan oleh program *Microsoft Word*. Jenis *file* ini bisa berisi susunan teks, *image*, tabel, grafik, *chart*, format halaman dan pengaturan cetak, jenis *file* ini banyak digunakan pada sistem operasi *windows*.
2. DOCX (.docx)
   File DOCX dirancang untuk membuat dokumen yang kontennya dapat diakses dengan mudah. Hal ini dimaksudkan bahwa sebuah dokumen dengan format DOCX bisa memiliki format *file* yang beragam.
3. RTF (.rtf)
   File RTF merupakan sebuah *file* teks yang disimpan dalam format *rich text*.
4. TXT (.txt)
   File TXT merupakan standar dokumen teks yang berisi rangkaian teks yang tidak terformat.

2.2 Algoritma LOKI97

Algoritma LOKI97 melakukan proses enkripsi dan dekripsi data 128 *bit*. Kunci yang digunakan bisa sepanjang 128, 192 atau 256 *bit*. LOKI97 merupakan algoritma blok *cipher* sehingga data keluaran akan menghasilkan panjang yang sama dengan data masukan, yaitu 128 *bit*[5]. Langkah-langkah yang dilakukan pada algoritma LOKI97 untuk melakukan proses enkripsi adalah sebagai berikut[6]:

1. Plainteks dibagi menjadi blok-blok yang masing-masing *n bit*, dalam hal ini menggunakan 128 *bit*.
2. Masing-masing di pecah menjadi dua buah blok sama panjang (L dan R).
3. Setelah plainteks dibagi menjadi dua blok sama panjang kemudian dilakukan putaran sebanyak 16 kali dengan menggunakan jaringan *feistel*.
4. Pada setiap putaran dilakukan proses operasi XOR sebagai berikut:
   $R_i = L_i\text{-}1$ XOR $f(R_i\text{-}1 + SK\ 3_i\text{-}2, SK\ 3_i\text{-}1)$
   $L_i = R_i\text{-}1 + SK\ 3_i\text{-}2 + SK\ 3_i$
5. Hasil putaran terakhir akan mendapatkan cipherteks yang berupa gabungan dari L dan R.

Algoritma LOKI97 menggunakan jadwal pembangkitan sub kunci berdasarkan jaringan *feistel* yang tidak seimbang yang mengoperasikan 4 buah blok sebesar 64 *bit*. Dikarenakan ukuran kunci yang digunakan bisa memiliki 3 buah ukuran yang berbeda, maka inisialisasi kunci untuk masing-masing ukuran
pun berbeda. Misalkan untuk kunci sepanjang 256 *bit*,, inisialisasinya adalah*:*

[K40[K30][K20][K10]= [Ka|Kb|Kc|Kd]

Sedangkan untuk kunci 128 *bit*, inisialisasi pembangkit subkuncinya adalah:

[K40|K30|K20|K10] = [Ka|Kb|Kc|Kd].

Untuk mendapatkan subkunci Ski dengan melakukan putaran sebanyak 48 kali, prosesnya adalah sebagai berikut:

Ski = K1 i = K4 i-1 XOR gi(K1 i-1, K3i -1, K2 i-1)

Dengan i mulai dari 1 sampai dengan 48 dan

Gi(K1,K3,K2i) = f(K1 + K3 + (Delta *i), K2)

Delta = [(Sqrt(5)-1) * 263]

Tiga proses dari penjadwalan kunci sangat dibutuhkan untuk menghasilkan tiga sub kunci untuk setiap iterasi dari perhitungan data. Jadi jumlah keseluruhannya 48 iterasi yang dibutuhkan untuk penjadwala kunci. Proses dekripsi sama dengan enkripsi yaitu menggunakan sub kunci untuk membalikkannya yaitu dengan cara membalikkan sub kunci tersebut[7].

2.6 Adaptive Hufmann Coding

Adaptive Huffman Coding adalah sebuah teknik *Adaptive Coding* berdasarkan *Huffman Coding*. Teknik yang juga disebut *Dynamic Huffman Coding* ini ada untuk menutupi salah satu kekurangan *Huffman Coding* yaitu fakta bahwa *"uuncompresser*" perlu punya "pengetahuan" mengenai probabilitas dari simbol di *file* yang terkompres. Hal ini menambah *bits* yang diperlukan untuk meng-enkodekan *file* dan apabila "pengetahuan" tidak tersedia, mengkompres *file* membutuhkan 2 langkah[9], yaitu:

1. Mencari frekuensi.
2. Mengkompress *file*.

Adaptive Huffman coding pertama kali dikembangkan oleh Faller dan Gallager, kemudian mendapatkan perbaikan oleh Donald Knuth. Sebuah metode lain dikembangkan oleh J.S. Vitter. *File* yang ter-enkodekan secara

dinamis punya dampak besar bagi keefektifan pohon sebagai *encoder* dan *decoder*. *Adaptive Huffman Coding* lebih efektif dari *Huffmann Coding* karena pohon terus berevolusi. Pada *Huffmann coding* yang statis, karakter yang berfrekuensi rendah akan ditempatkan jauh di posisi bawah dan akan memakan banyak *bits* untuk *encode*. Pada *Adaptive huffmann coding* karakter itu akan dimasukkan pada daun tertinggi untuk didekodekan sebelum akhirnya didorong ke bawah pohon oleh karakter berfrekuensi yang lebih besar[10].

3. HASIL DAN PEMBAHASAN

3.1 Pembahasan

Tahap analisa merupakan tahap yang sangat berpengaruh dan menentukan terhadap tahap selanjutnya. Analisa terhadap sistem merupakan tahap yang sangat penting untuk mengetahui proses yang terjadi di dalam aplikasi yang akan dirancang, dalam hal ini menggunakan algoritma *Adaptive Hufmann Coding* untuk mengompress *file* dan algoritma *LOKI97* untuk menyandikan *file*. Pengompressan *file* berguna untuk mengurangi ruang penyimpanan serta mempercepat proses pengiriman suatu *file*. Penyandian *file* berguna untuk meningkatkan keamanan dari suatu *file* agar tidak dapat dimanipulasi dan dimanfaatkan oleh orang-orang yang tidak bertanggung jawab. Proses yang terjadi di dalam sistem yang akan dibuat memiliki 2 proses utama, yaitu *file* yang di-*input*-kan akan disandikan dengan algoritma *LOKI97*, kemudian *file* yang sudah terenkripsi akan dikompress menggunakan algoritma *Adaptive Hufmann Coding*. Hasil dari proses penyandian *file* teks menggunakan algoritma *LOKI97* adalah sebuah *file* yang sudah terenkripsi dan tidak dapat dimengerti, kemudian hasil dari kompresi menggunakan algoritma *Adaptive Hufmann Coding* akan menghasilkan *file* terenkripsi yang memiliki ukuran yang lebih kecil dari ukuran awalnya.

3.1 Menerapkan Algoritma LOKI97 dalam Penyandian File Teks

Algoritma *LOKI97* melakukan proses enkripsi *file* 128 *bit*. Kunci yang digunakan bisa sepanjang 128, 192 atau 256 *bit*. *LOKI97* merupakan algoritma *block cipher* sehingga data keluaran akan menghasilkan panjang yang sama dengan data masukan.

Langkah-langkah dalam proses enkripsi algoritma *LOKI97* dapat dilihat dari contoh berikut:

Isi *file* teks (plainteks) : PUTRABUDIDARMAMD
Kunci : GREEN_CAMPUS_MDN
Jika diubah ke dalam bilangan heksadesimal adalah sebagai berikut:
Plainteks : 50 55 54 52 41 42 55 44 49 44 41 52 4D 41 4D 44
Kunci : 47 52 45 45 4E 5F 43 41 4D 50 55 53 5F 4D 44 4E

1. Plainteks dibagi menjadi *blok-blok* yang masing-masing n *bit*, dalam hal ini menggunakan 128 *bit*.
   PUTRABUDIDARMAMD = 16 Karakter = 128 *bit* = 128/2 = 64 *bit*
2. Masing-masing dipecah menjadi dua blok sama panjang (L dan R)
   L_0 = 50 55 54 52 41 42 55 44
   R_0 = 49 44 41 52 4D 41 4D 44
3. Setelah plainteks dibagi menjadi dua blok sama panjang, kemudian dilakukan putaran sebanyak 16 kali menggunakan jaringan *feistel* dengan membangkitkan kunci terlebih dahulu.
   K1 = 475245454E5F4341 = 64 *bit* pertama kunci
   K2 = 4D5055535F4D444E = 64 *bit* kedua kunci
   $SK_0 = K1_0 = K4_0$ XOR g_0 ($K1_0$, $K3_0$, $K2_0$)
   $K4_0 = K3_0$ = 4D5055535F4D444E
   $K3_0 = K2_0$ = 4D5055535F4D444E
   $K2_0 = K1_0$ = 475245454E5F4341
   Fungsi g_0 (K1,K3,K2) = *f*(K1+K3 + (Delta*I),K2)
   Delta = [(sqrt(5)-1)*2^{63}) =9E3779B97F4A7C15
   g_0 = *f*(475245454E5F4341 + 4D5055535F4D444E + 9E3779B97F4A7C15,
   475245454E5F4341)
   = F9C3CCB0F0AF6925
   SK_0 = 4D5055535F4D444E XOR F9C3CCB0F0AF6925
   = B49399E3AFE22D6B
   Jadi SK_0 = B49399E3AFE22D6B
   Untuk mencari *subkey* (SK) selanjutnya digunakan cara yang sama untuk mencari *subkey* selanjutnya sebanyak 48 putaran, sehingga menghasilkan
   subkey sebagai berikut:
   SK_1 = B49399E3AFE22D6B
   SK_2 = C4FF0D880DA8B7AE
   SK_3 = 5EB87D53C7C77A60
   SK_4 = E7864E671CF56D00
   SK_5 = E291AC29E02DA600

4. Pada setiap putaran dilakukan proses operasi XOR sebagai berikut:
   R_i = L_{i-1} XOR f(R_i-1 + SK_{3i-2}, SK_{3i-1})
   L_i = R_{i-1} + SK_{3i-2} + SK_{3i-1}
   R_1 = L_0 XOR f(R_0 + SK_1, SK_2)
   L_1 = R_0 + SK_2 + Sk_3
   R_1 = 5055545241425544 XOR *f*(494441524D414D44 + B49399E3AFE22D6B + C4FF0D880DA8B7AE)
   = 5055545241425544 XOR C2D6E8BE0ACC325D
   = 9283BCEC4B8E6719
   L_1 = 494441524D414D44 + C4FF0D880DA8B7AE + 5EB87D53C7C77A60
   = 6CFBCC2E22B17F52
   Proses tersebut dilakukan sebanyak 16 putaran dengan menggunakan rumus yang sama sehingga menghasilkan nilai sebagai berikut:
   R_{16} = 61F7C551C7AB09A0
   L_{16} = 40160845CB0BA1C8
5. Hasil putaran terakhir akan mendapatkan cipherteks yang berupa gabungan dari L dan R yaitu: 40160845CB0BA1C861F7C551C7AB09A0 Jika dikonversi menjadi karakter menjadi sebagai berikut:
   @[8]EË[11]¡Èa÷ÅQÇ«[9][160]

Proses dekripsi merupakan proses pengembalian *ciphertext* menjadi *plaintext* sebagai berikut:

Ciphertext : @[8]EË[11]¡Èa÷ÅQÇ«[9][160]
Kunci : GREEN_CAMPUS_MDN

Jika diubah ke dalam bilangan heksadesimal adalah sebagai berikut:

Cipherteks : 40160845CB0BA1C861F7C551C7AB09A0
Kunci : 475245454E5F43414D5055535F4D444E

Langkah-langkah dalam proses dekripsi *LOKI97* adalah sebagai berikut:

1. Cipherteks dibagi menjadi *blok-blok* yang masing-masing n *bit*, dalam hal ini menggunakan 128 *bit*.
   @[8]EË[11]¡Èa÷ÅQÇ«[9][160] = 16 Karakter = 128 *bit* = 128/2 = 64 *bit*
2. Masing-masing dipecah menjadi dua blok sama panjang (L dan R)
   L_0 = 40160845CB0BA1C8
   R_0 = 61F7C551C7AB09A0
3. Setelah cipherteks dibagi menjadi dua blok sama panjang, kemudian dilakukan putaran sebanyak 16 kali menggunakan jaringan *feistel* dengan membangkitkan kunci terlebih dahulu.
   K1 = 475245454E5F4341
   K2 = 4D5055535F4D444E
   SK_0 = $K1_0$ = $K4_0$ XOR g_0 ($K1_0$, $K3_0$, $K2_0$)
   $K4_0$ = $K3_0$ = 4D5055535F4D444E
   $K3_0$ = $K2_0$ = 4D5055535F4D444E
   $K2_0$ = $K1_0$ = 475245454E5F4341
   Fungsi g_0 (K1,K3,K2) = *f*(K1+K3 + (Delta*I),K2)
   Delta = [(sqrt(5)-1)*2^{63}) =9E3779B97F4A7C15
   g_0 = *f*(475245454E5F4341 + 4D5055535F4D444E + 9E3779B97F4A7C15, 475245454E5F4341)
   = F9C3CCB0F0AF6925
   SK_0 = 4D5055535F4D444E XOR F9C3CCB0F0AF6925
   = B49399E3AFE22D6B
   Jadi SK_0 = B49399E3AFE22D6B
   Untuk mencari *subkey* (SK) selanjutnya digunakan cara yang sama untuk mencari *subkey* selanjutnya sebanyak 48 putaran, sehingga menghasilkan *subkey* sebagai berikut:
   SK_1 = B49399E3AFE22D6B
   SK_2 = C4FF0D880DA8B7AE
   SK_3 = 5EB87D53C7C77A60
   SK_4 = E7864E671CF56D00
   SK_5 = E291AC29E02DA600
4. Pada setiap putaran dilakukan proses operasi XOR sebagai berikut:
   R_i = L_{i-1} XOR f(R_i-1 + SK_{46}, SK_{47})
   L_i = R_{i-1} + SK_{47} + SK_{48}
   R_1 = L_0 XOR f(R_0 + SK_{46}, SK_{47})
   L_1 = R_0 + SK_{47} + Sk_{48}
   R_1 = 40160845CB0BA1C8 XOR *f*(61F7C551C7AB09A0 + 21BA05F6D29E A5A7+ C0118051F61962C4)
   = 40160845CB0BA1C8 XOR 43C34B9A9063120B
   = 3D543DF5B68B3C3

L_1 = 61F7C551C7AB09A0+ C0118051F61962C4 + DE4BFA0C9D642359
= 553FB05B288FBD

Proses tersebut dilakukan sebanyak 16 putaran dengan menggunakan rumus yang sama sehingga menghasilkan nilai sebagai berikut:

R_{16} = 494441524D414D44

L_{16} = 5055545241425544

5. Hasil putaran terakhir akan mendapatkan plainteks yang berupa gabungan dari L dan R yaitu: 505554524142554449444152 4D414D44 Jika dikonversi menjadi karakter menjadi sebagai berikut: PUTRABUDIDARMAMD

3.2 Menerapkan Adaptive Hufmann Coding pada Kompresi File Teks

Adaptive Hufmann Coding adalah sebuah teknik *Adaptive* c*oding* berdasarkan *Hufmann coding*. Langkah-langkah proses kompresi *adaptive hufmann coding* pada *file* yang sudah tersandikan algoritma *LOKI97* dapat dilihat dari contoh berikut ini:

Cipherteks = @[8]EË[11]¡Èa÷ÅQÇ«[9][160]

1. Mencari frekuensi kemunculan karakter, dapat dilihat dari tabel frekuensi di bawah ini.

Tabel 1. Frekuensi Kemunculan Karakter

Karakter	Frekuensi	Simbol
@	1	0
	1	0
[8]	1	1
E	1	00
Ë	1	01
[11]	1	10
¡	1	11
È	1	000
a	1	001
÷	1	010
Å	1	011
Q	1	101
Ç	1	110
«	1	111
[9]	1	0000
[160]	1	0111

2. Mengkompress *file*

Berdasarkan frekuensi kemunculan karakter, maka proses kompresi pada algoritma *adaptive hufmann coding* langsung dikompresi menjadi 0010001101100000101001110111011100000111 = 40 *bit*

Pengukuran rasio kompresi dilakukan untuk mengetahui berapa persentase kompresi algoritma *adaptive hufmann coding* dengan ukuran awal adalah 128 *bit*, sehingga rasio kompresinya adalah:

Rasio Kompresi = $\frac{40}{128}$ x 100% = 31,25 %

Berdasarkan perhitungan di atas, dapat disimpulkan bahwa dengan menggunakan algoritma *adaptive hufmann coding* adalah sebesar 31,25 %.

4. KESIMPULAN

Berdasarkan pembahasan dan evaluasi maka peneliti dapat diambil kesimpulan-kesimpulan. Adapun kesimpulan-kesimpulan tersebut yaitu penerapan algoritma Loki dalam penyandian file teks dilakukan dengan membagi file teks menjadi blok-blok cipher untuk dienkripsi dan menggabungkan kembali blok tersebut menjadi cipherteks, begitu juga sebaliknya.Serta ciphertext yang didapat dari proses enkripsi Loki97 dikompresi menggunakan algoritma adaptive Huffman coding dengan mengganti nilai karakter dari file yang akan dikompres ke dalam tabel adaptive Huffman coding, sehingga menghasilkan tabel kode yang akan digunakan untuk menggantikan karakter awal file.

REFERENCES

[1] I.M.Kenjibriel, “Enkripsi Data Kunci Simetris Dengan Algoritma Kriptografi LOKI97”, Institut Teknologi Bandung. 2010
[2] D.F.Djusdek, T.Ahmad, et.al, “Pemampatan Citra Keabuan dengan Algoritma Adaptive Huffman dan LZW”, Jurnal Teknik ITS, Vol.V.2016
[3] Z.Amin, “Desain dan Implementasi Tunneling IPSEC Berbasis UNIX dengan ESP (Encapsulating Security Payload)”, STMIK PalComTech, vol.II, pp109-119. 2012
[4] H.Christian, “Studi tentang Pelaksanaan Rencana Kerja Pembangunan Desa Tahun 2013 di Desa Loa Janan Ulu Kecamatan

Loa Janan Kabupaten Kutai Kartanegara", eJournal Pemerintahan Integratif, vol.III, pp. 190-210. 2015
[5] A.Aneta, "Implementasi Kebijakan Program Penanggulangan Kemiskinan Perkotaan (P2KP) di Kota Gorontalo", Jurnal Administrasi Publik, vol.I, pp. 54-65. 2010
[6] J.Enterprise, Rahasia Manajemen File, Jakarta: PT.Elex Media Komputindo.2010
[7] E. Setyaningsih, Kriptografi & Implementasinyaa Menggunakan Matlab, Yogyakarta: Penerbit Andi.2015
[8] A.A.Ngurah Pradnya Andikha, et.al, "Enkripsi dan Dekripsi Audio Format AMR dengan Algoritma Kriptografi LOKI97", Jurnal Universitas Udayana. 2010
[9] Hidayat, W.Zarman, et.al, "Implementasi Algoritma Kompresi LZW Pada Database Server", Jurnal Komputa, vol.II, pp. 7-14. 2013
[10] M. Ardhin, "Adaptive Huffman Coding Sebagai Variasi Huffman Coding", Jurnal STEI ITB. 2011
[11] H. Delisman, N. Berto dan A. Soeb, "Implementasi Algoritma AES (Advanced Encryption Standard) Untuk Keamanan File Hasil Radiologi di RSU Imelda Medan," dalam KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), Medan, 2020.
[12] F. R. Muhammad, S. Muhammad dan A. Soeb, "Implementasi Pisanc Chiper Untuk Autentikasi Voice Chat," Journal of Computer System and Informatics (JoSYC), vol. 2, no. 4, pp. 288-294, 2021.
[13] A. Nur dan A. Soeb, "Penerapan Algoritma Elias Omega Code Pada Kompresi File Audio Aplikasi Murottal Muzzamil Hasbalah," Pelita Informatika: Informasi dan Informatika, vol. 9, no. 2, pp. 113-119, 2020.
[14] A. Soeb dan S. Muhammad, "Pengaman File Video Menggunakan Algoritma Merkle Hellman Knapsack," JURNAL MEDIA INFORMATIKA BUDIDARMA, vol. 4, no. 2, pp. 461-465, 2020.
[15] M. Tedi, L. G. Guidio, M. Mesran, F. Alwin, A. Soeb dan S. Dodi, "IMPLEMENTASI ALGORITMA J-BIT ENCODING PADA KOMPRESI FILE TEKS," KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), vol. 1, no. 1, 2017.
[16] A. Muhammad dan A. Soeb, "Penarapan Algoritma Prefix Code Dalam Kompresi File Video," dalam KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), Medan, 2021.