PERANCANGAN DAN IMPLEMENTASI *SECURE CLOUD* DENGAN MENGGUNAKAN *DIFFIE-HELLMAN KEY EXCHANGE* DAN *SERPENT CRYPTOGRAPHY ALGORITHM*

DESIGN AND IMPLEMENTATION SECURE CLOUD BY USING DIFFIE-HELLMAN KEY EXCHANGE AND SERPENT CRYPTOGRAPHY ALGORITHM

Eka Cahya Pratama[1], Surya Michrandi Nasution[2], Tito Waluyo Purboyo[3]

Telkom University
Bandung, Indonesia ekacahyapratama@students.telkomuniversity.ac.id[1],
michrandi@telkomuniversity.ac.id[2] titowaluyo@telkomuniversity.ac.id[3]

Abstrak

***Cloud* menyediakan akses kepada pengguna untuk menyimpan data di internet dengan kapasitas penyimpanan yang dapat disesuaikan dengan keinginan penggunanya. Namun, metode ini memiliki kelemahan bahwa *cloud* memiliki masalah keamanan data yang bisa dicuri atau dibajak. Untuk mengatasi masalah tersebut, banyak peneliti memilih kombinasi terbaik dari algoritma enkripsi dan mekanisme yang digunakan. Dalam tulisan ini, kami mengusulkan untuk membuat penyimpanan *secure cloud* menggunakan Algoritma Enkripsi *Serpent* untuk melindungi data yang tersimpan di awan. Desain ini akan mensimulasikan data yang di *upload* dan di *download* milik pengguna yang terdaftar di *Cloud Server*. Data pengguna dienkripsi menggunakan Algoritma *Serpent* sebelum *upload* di Cloud *Server*. Ketika data akan di-*download* oleh pengguna, data akan didekripsi menggunakan metode yang sama.**

Kata kunci: *Cloud*, Kriptografi, Algoritma kriptografi *Serpent*, Pertukaran Kunci *Diffie-Hellman*

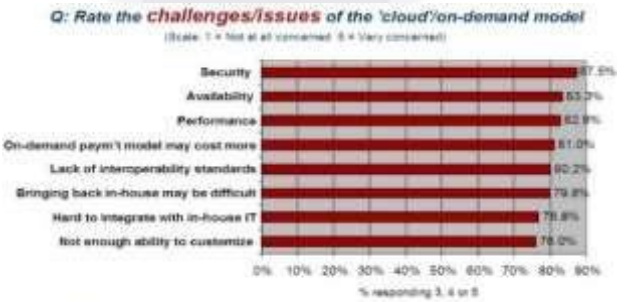
Abstract

Cloud provides access to its users to store data in the internet with a storage capacity that can be adapted to the wishes of its users. However, this method has the disadvantage that the Cloud has security problems of data that can be stolen or hijacked. In order to solve the problem, many researcher choose the best combination of encryption algorithms and the mechanisms are used. In this paper, we proposed to make a secure cloud storage using Serpent Encryption Algorithm to protect the data stored in cloud. This design will simulate a data uploaded and downloaded belonging to users who are registered in the Cloud Server. The user's data is encrypted using Serpent Algorithm before being uploaded in the Cloud Server. When the data will be downloaded by the user, the data will be decrypted using the same method.

Keywords: Cryptography, Serpent Algorithm, Cloud, Diffie-Hellman Exchange Keys

1. Pendahuluan

[1]Dewasa ini kita pasti memiliki banyak data yang ada di komputer kita, baik itu berupa dokumen, gambar, film ataupun data lainnya, tentu hal ini membuat kita memerlukan kapasitas penyimpanan data yang besar. Dengan berkembang pesatnya teknologi internet beserta teknologi pemrosesan dan penyimpanan data membuat biaya dalam *computing* semakin murah, kemampuan yang lebih tinggi, dan juga ketersediaan yang lebih terjamin [1].



Gambar 1.1 Hasil Survei IDC terhadap ancaman pada *Cloud Service*[1]

[1] F. Gens, "New IDC IT Cloud Services Survey : Top Benefits and Challenges," IDC, Desember 2009. [Online]. Available: http://blogs.idc.com/ie/?p=730. [Diakses 15 September 2014].

Salah satu solusi keamanan penyimpanan pada *cloud* untuk menjaga data agar tidak dicuri yaitu dengan menggunakan kombinasi dari enkripsi pada *file* yang akan disimpan di *cloud*. Dua bagian penting dari pengamanan informasi pada lingkungan *cloud* adalah enkripsi dan autentikasi. Secara umum, mekanisme enkripsi menjadi salah satu prioritas dasar dalam menjaga keamanan data di *cloud* [2]. Dengan adanya permasalahan tersebut, kami membuat suatu sistem *cloud* dengan kemampuan enkripsi data menggunakan algoritma *Serpent* dan untuk pengiriman kunci digunakan algoritma distribusi kunci *Diffie-Hellman.*

2. Tinjauan Pustaka

2.1 Keamanan Jaringan

Keamanan jaringan adalah kumpulan peranti yang dirancang untuk melindungi data ketika transmisi terhadap ancaman pengaksesan, pengubahan dan penghalangan oleh pihak yang tidak berwenang. Keamanan jaringan berbeda dengan keamanan komputer. Keamanan komputer adalah kumpulan peranti yang dirancang untuk melindungi komputer sehingga data pada komputer terlindungi [3]. Sistem anti virus komputer merupakan contoh peranti keamanan komputer sedangkan protokol *web* yang aman merupakan contoh peranti keamanan jaringan [4].

2.2 Kriptografi

Pada awalnya kriptografi dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan. Namun pada pengertian modern kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas [4]. Dalam kriptografi pesan asli disebut *plaintext* sedangkan pesan yang tersandikan disebut *ciphertext*, serta terdapat istilah enkripsi dan dekripsi. Enkripsi adalah proses merubah *plaintext* ke *ciphertext*. Sedangkan, dekripsi adalah proses mengembalikan *plaintext* dari *ciphertext* [3]. Algoritma-algoritma dalam kriptografi dapat dibagi menjadi dua jenis, yaitu:

1. Penyandian dengan kunci simetris (Symmetric key encipherment)

Algoritma yang berada pada kategori ini mempunyai kunci enkripsi dan dekripsi yang bernilai sama.

2. Penyandian dengan kunci asimetris (Asymmetric key encipherment)

Algoritma dalam kategori ini mempunyai kunci untuk enkripsi (kunci publik) yang berbeda dengan kunci untuk dekripsi (kunci rahasia). Kunci publik bersifat umum/tidak rahasia.

Algoritma kunci simetri kemudian dapat pula dibagi menjadi dua jenis, yaitu:

1. Cipher aliran (*stream cipher*)

Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit.

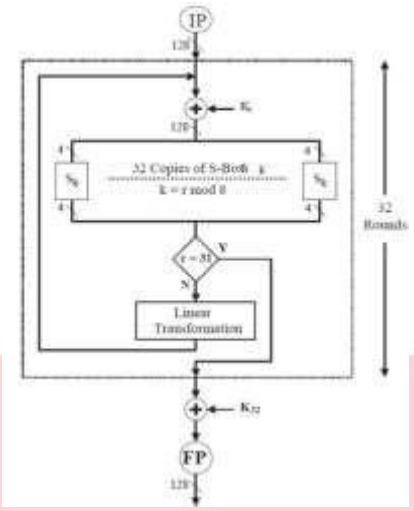
2. Cipher blok (*block cipher*)

Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya.

2.3 Algoritma Kriptografi *Serpent*

Algoritma *serpent* didesain oleh Ross Anderson, Eli Biham dan Lars Knudsen sebagai *block cipher* baru untuk *Advanced Encryption Standard* oleh *National Institute of Standards and Technology* di Amerika Serikat sebagai pengganti dari DES dengan kriteria yaitu lebih cepat dari pada triple DES dan menyamai keamanan yang diberikan oleh triple DES dengan panjang *block* sebesar 128 bit dengan kunci sepanjang 256 bit namun tetap dapat digunakan dengan 128 bit dan 192 bit [5].

Serpent menggunakan 32 putaran melalui jaringan subtitusi-permutasi pada 4 buah 32 bit *word*, sehingga totalnya membuat ukuran *block* dari *serpent* yaitu 128 bit. Semua nilai pada *cipher* direpresentasikan sebagai *bitstream*. Semua bit dihitung dari 0 sampai 31 pada 32 bit *word*, 0 sampai 127 pada 128 bit *block*, 0 sampai 255 bit pada 256 bit kunci dan seterusnya. Semua nilai menggunakan little-endian dimana *word* pertama (*word* 0) adalah *least significant word* dan *word* terakhir adalah *most significant word*, dan bit 0 adalah *least significant bit* dari *word* 0. Semua nilai ditulis pada keluaran dengan 128 bit hexadesimal.



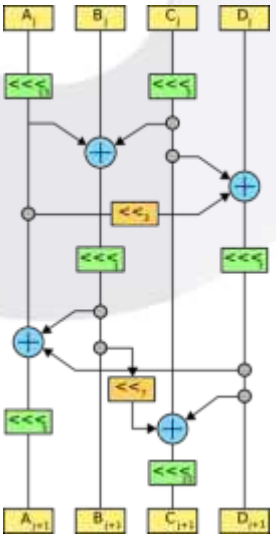
Gambar 2.1 Diagram balok dari *Serpent*[2]

Serpent sendiri terdiri dari :

1. Permutasi awal IP,
2. 32 perulangan, semuanya terdiri dari *key mixing operation*, melewati S-box dan transformasi linier (kecuali pada perulangan terakhir). Pada perulangan terakhir, transformasi linier diganti dengan tambahan *key mixing operation*,
3. Permutasi akhir FP.

Permutasi awal dan permutasi akhir tidak memiliki hasil yang berarti pada kriptografi. *Serpent* terdiri dari 32 putaran. *Plaintext* menjadi data awal $B_0 = P$ dengan permutasi awal yang kemudian akan masuk ke 32 perulangan.

1. *Key Mixing*, pada setiap perulangan, *subkey* sepanjang 128 bit akan di XOR dengan data B_i,
2. S-Box, 128 bit data yang sudah diolah pada *key mixing* kemudian dibagi menjadi 4 bagian (32 bit *word*) yang nantinya akan diolah dengan operasi logika secara berurutan menghasilkan Si(Bi $\oplus$ Ki)
3. *Linier Transformation*, 32 bit dari setiap word hasil dari S-Box kemudian akan diproses dengan transformasi linier.



Gambar 2.2 Transformasi Linier pada *Serpent*[3]

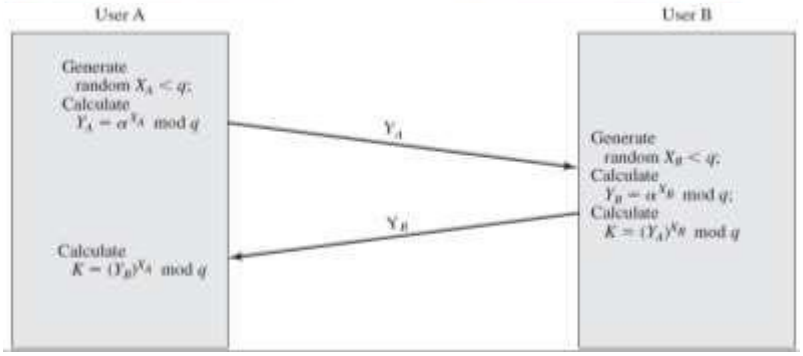
[2] K. Ballentine dan T. Moore, "Serpent Cipher Design and Analysis," 2012, 5.

[3] Dake, "Serpent-linearfunction - Serpent (Cipher)," 1 Agustus 2009. [Online]. Available: http://commons.wikimedia.org/wiki/File:Serpent-linearfunction.png. [Diakses 21 April 2015]

Transformasi linier berlangsung sebanyak 31 kali. Tanda ($<<<_n$) berarti melakukan rotasi sebanyak n kali. Tanda ($<<_n$) berarti melakukan pergeseran sebanyak n kali. Transformasi linier dipilih karena untuk memaksimalkan *avalanche effect*. Alasan lainnya karena transformasi linier lebih simpel dan dapat digunakan pada prosesor modern [5].

2.4 Pertukaran Kunci Diffie-Hellman

Diffie-Hellman Key Exchange atau pertukaran kunci *Diffie-Hellman* adalah metode pertukarang kunci rahasia pada komunikasi menggunakan kriptografi asimetris. Kekuatan dari metode ini adalah pada sulitnya melakukan perhitungan logaritma diskrit antara pengirim dan penerima kunci [6]. Tujuan dari algoritma ini adalah memperbolehkan dua pengguna untuk bertukar kunci secara aman yang nantinya bisa digunakan untuk melakukan enkripsi pesan. Algoritma *Diffie-Hellman* sendiri terbatas pada pertukaran nilai rahasia [3]. Tahap-tahap pertukaran kunci *Diffie-Hellman* adalah sebagai berikut :



Gambar 2.3 Langkah-langkah di pertukaran kunci *Diffie-Hellman* [3, p. 304]

1. Misalkan A dan B adalah pihak-pihak yang berkomunikasi. Mula-mula A dan B menyepakati 2 buah bilangan q dan α. Dimana q adalah bilangan prima yang sangat besar dan α adalah *primitive root* dari q dengan syarat $\alpha < q$.
2. A membuat bilangan bulat acak X_A dengan syarat $X_A < q$ dan mengirim hasil perhitungan berikut kepada B, $Y_A = \alpha^{X_A} \bmod q$
3. B membuat bilangan bulat acak Y_B dengan syarat $Y_B < q$ dan mengirim hasil perhitungan, berikut kepada A, $Y_B = \alpha^{X_B} \bmod q$
4. A lalu menghitung kunci rahasia dengan, $K = (Y_B)^{X_A} \bmod q$
   B menghitung kunci rahasia dengan, $K = (Y_A)^{X_B} \bmod q$

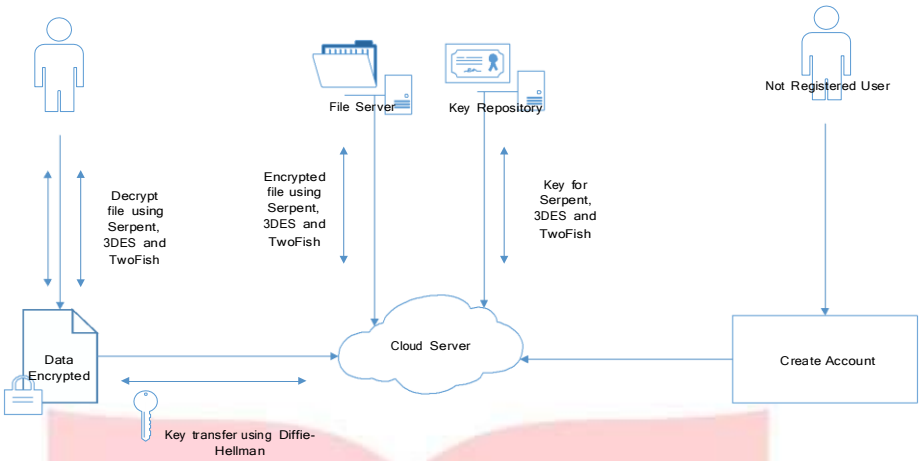
2.5 *Cloud Storage*

Cloud storage adalah model penyimpanan secara *online* dimana data disimpan pada penyimpanan yang tervirtualisasi yang biasanya di selenggarakan oleh pihak ketiga [7]. *Cloud Storage* merupakan salah satu bagian dari *cloud computing*. Kebanyakan layanan *cloud storage* saat ini bertipe *Software as a Service* (SaaS). SaaS menyediakan aplikasi yang berjalan pada insfrastruktur *cloud* yang bisa diakses melalui *web browser* atau aplikasi. Pengguna bisa menggunakan layanan yang diberikan namun tidak bisa melakukan kontrol terhadap insfrastruktur *cloud* termasuk jaringan, *server*, sistem operasi, media penyimpanan atau kemampuan dari aplikasi [8]. Secara sederhana, *cloud storage* dapat berupa satu pengguna dengan akses ke satu *server*. Pengguna akan mengunggah datanya ke *server* untuk disimpan. Hal ini sangat tidak efisien karena jika *server* yang digunakan mati maka pengguna harus menunggu hingga menyala kembali. Agar *cloud storage* dapat digunakan sebagai bisnis, level paling rendah adalah dengan memperluas *server* untuk mengatasi kendala *reliability*. Konsep dari *cloud storage* adalah peningkatan tehnologi yang akan membantu pengguna untuk menyimpan data dengan aman dan dapat dipercaya [9].

3. Perancangan Sistem

3.1 Gambaran Umum Sistem

Gambaran umum sistem secara keseluruhan dapat dilihat pada gambar berikut :



Gambar 3.1 Gambaran Sistem Secara Umum

3.2 Perancangan Antarmuka

Perancangan antarmuka dari sistem yang dibangun terdiri dari :

Gambar 3.2 Rancangan Tampilan Antarmuka

4. Implementasi dan Pengujian Sistem

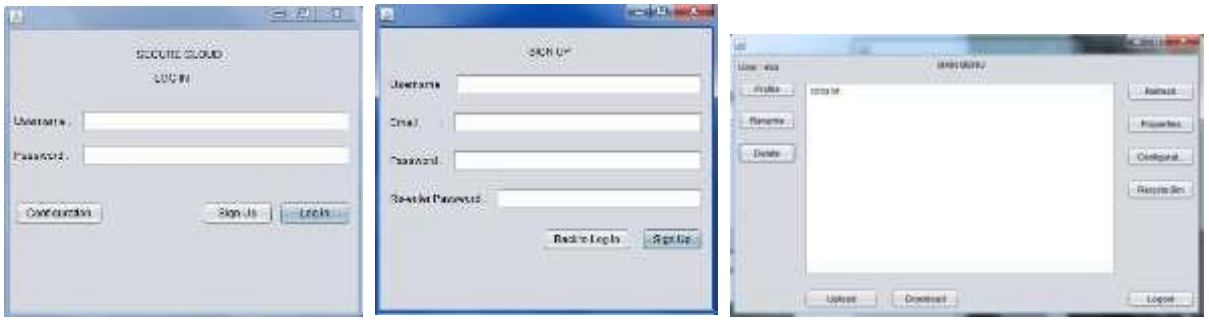
4.1 Implementasi Sistem

Dalam implementasi sistem enkripsi dan dekripsi file, langkah – langkah yang dilakukan ialah sebagai berikut:

1. Masuk kedalam Aplikasi SecureCloud dengan *Log In* atau *Sign Up* terlebih dahulu.
2. Memilih *file* dokumen yang ingin di *upload* ke dalam SecureCloud
3. Saat memilih *file* dokumen user dapat menganti nama *file* tersebut
4. Saat mengunggah *file* dokumen maka secara langsung *file* tersebut terenkripsi
5. Saat mengunduh *file* dokumen maka secara langsung *file* tersebut terdekripsi
6. Menghitung waktu proses enkripsi dan dekripsi dari *file* dokumen tersebut
7. Menghitung nilai *avalanche effect* data biner dari *file* dokumen tersebut
8. Menganalisis pemakain daya yang terjadi saat proses enkripsi dan dekripsi dari *file* dokumen tersebut

4.2 Implementasi Antarmuka

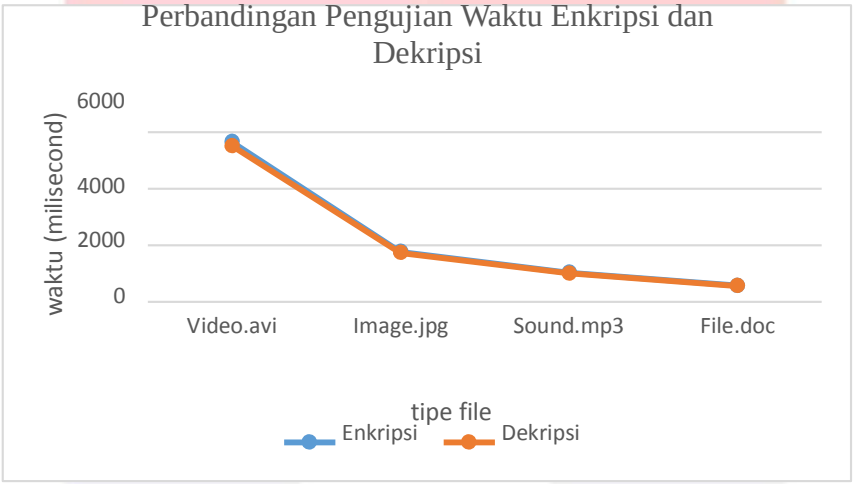
Implementasi antarmuka / *User Interface* merupakan tampilan dari aplikasi *file* yang akan menampilkan *data file* dokumen terenkripsi dan terdekripsi. Berikut merupakan tampilan antarmuka yang telah diimplementasikan :



Gambar 4.1 Implementasi Rancangan Antarmuka

4.3 Pengujian Sistem

4.3.1 Pengujian Waktu Enkripsi dan Dekripsi



Gambar 4.2 Diagram pengujian waktu enkripsi dan waktu dekripsi

Pada pengujian waktu proses enkripsi dan waktu proses dekripsi didapatkan hasil bahwa besar ukuran dari *file* yang diuji memiliki pengaruh dalam lama waktu proses enkripsi maupun lama waktu proses dekripsi. Untuk *file* video.avi memiliki waktu proses enkripsi terlama yaitu 5648 ms dan juga waktu proses dekripsi terlama yaitu 5509.3667 ms. Sedangkan *file*.txt memiliki waktu enkripsi tercepat yaitu 183.3333 ms dan waktu dekripsi paling cepat yaitu 175 ms.

4.3.2 Pengujian Keamanan Sistem

Tabel 4.1 Tabel pengujian keamanan sistem

No	Nama *File*	Jumlah Bit Beda	Avalanche Effect %
1	File.docx	405591.1333	50.0107%
2	Video.avi	4117204	50.0002%
3	Sound.mp3	741901.2333	49.9966%
4	Image.jpg	1309807.4	50.0703%
Total Rata-Rata Avalanche Effect			**50.02%**

Pada pengujian keamanan sistem atau *Avalanche Effect* didapatkan hasil bahwa nilai AE yang tertinggi didapat di Image.jpg dengan nilai 50.0703% dan nilai AE terendah didapat di Sound.mp3 dengan nilai 49.9966%. total rata-rata AE yang didapatkan adalah 50.02% ini sudah tergolong cukup bagus karena hampir mendekati 50% dari total bit yang diujikan.

4.3.3 Pengujian Resources

Tabel 4.2 Tabel pengujian *resources*

No	Nama *File*	HS Enkripsi	HS Dekripsi	UH Enkripsi	UH Dekripsi
1	File.docx	106168320 *byte*	62390272 *byte*	159167232.3 *byte*	13184270.67 *byte*
2	Video.avi	276771635.2 *byte*	191277738.7 *byte*	26389672 *byte*	121361836.3 *byte*
3	Sound.mp3	170393600 *byte*	78101435.73 *byte*	59463272 *byte*	24559464 *byte*
4	Image.jpg	243304585 *byte*	108981998.9 *byte*	68297346 *byte*	43613917 *byte*
Total Resources		**215215855.1 *byte***	**110187861.3 *byte***	**78329380.58 *byte***	**50679871.99 *byte***

Pada pengujian *resources* ini dilakukan dengan menguji daya pemakaian dengan menggunakan *Heap Size* dan *Used Heap*. *Heap Size* adalah pengaplikasian sejumlah memori untuk aplikasi *Java Virtual Machine*. *Used Heap* adalah jumlah memori yang sesungguhnya dipakai oleh proses aplikasi. Dari hasil pengujian didapatkan nilai HS enkripsi paling rendah adalah file.docx dan sound.mp3 dengan 170393600 *byte* dan nilai HS dekripsi paling rendah adalah file.docx dengan 62390272 byte. Untuk nilai UH enkripsi paling rendah adalah video.avi dengan 26389672 byte dan nilai UH dekripsi paling rendah adalah file.docx dengan 13184270.67 byte. Perbandingan nilai HS dan nilai UH pada pengujian menunjukan tingkat pemakaian daya saat aplikasi di jalankan. Semakin nilai dari HS maupun UH rendah maka daya yang digunakan untuk pemrosesan semakin kecil.

4.3.4 Pengujian Diffie-Hellmann

Tabel 4.3 Tabel pengujian Diffie-Hellman

1024 bit		2048 bit	3072 bit	4096 bit	7680 bit	15360 bit
Total Waktu	497.567 ms	3175.567 ms	12343.767 ms	32447.267 ms	515274.8 ms	5672359.6 ms

Pada pengujian waktu proses *Diffie-Hellmann* di ketahui total rata-rata waktu proses selama 497.567 ms untuk 1024 bit, 3175.567 ms untuk 2048 bit, 12343.767 ms untuk 3072 bit, 32447.267 ms untuk 4096 bit, 515274.8 ms untuk 7680 bit dan 5672359.6 ms untuk 15360 bit.

4.3.5 *Big-O Notation*

Dari pengujian *Big-O Notation* didapatkan bahwa algoritma *Serpent* memiliki nilai O(N) dimana N adalah banyak *block* yang harus dikerjakan untuk melakukan enkripsi atau dekripsi suatu *file.* Untuk algoritma *Diffie-Hellman* didapatkan nilai $O((\log N)^4)$ dimana N adalah ukuran *key* yang digunakan pada algoritma *Diffie-Hellman.*

5. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian Tugas Akhir ini adalah :

1. Berdasarkan hasil pengujian waktu proses enkripsi dan proses dekripsi dengan menggunakan algoritma Serpent di ketahui total rata-rata waktu proses enkripsi adalah 4648,4583 ms dan total waktu proses dekripsi adalah 4561,8133 ms. Waktu proses enkripsi dan dekripsi dipengaruhi oleh ukuran dari file karena algoritma Serpent merupakan algoritma *block cipher* yang proses enkripsinya dilakukan per *block* yang berukuran 128 bit.
2. Pada pengujian avalanche effect dengan 20 file yang berekstensi tidak sama di dapatkan total nilai rata-rata avalanche effect adalah 50.0506%. Nilai rata-rata AE yang didapat ini tergolong sangat aman karena mendekati setengah dari nilai bit total.
3. Berdasarkan pada pengujian resources diketahui total rata-rata nilai HS enkripsi adalah 288,248 MB dan total rata-rata nilai HS dekripsinya adalah 143,198 MB, sedangkan total rata-rata nilai UH enkripsi 127,644 MB dan total rata-rata nilai UH untuk dekripsi adalah 71,024 MB. Penggunaan *resources* sendiri dipengaruhi oleh seberapa besar *file* yang sedang di proses.
4. Berdasakan pengujian waktu proses *Diffie-Hellman* diketahui total rata-rata waktu proses selama 261.9 ms untuk 512 bit, 497.567 ms untuk 1024 bit, 3175.567 ms untuk 2048 bit, 12343.767 ms untuk 3072 bit, 32447.267 ms untuk 4096 bit, 515274.8 ms untuk 7620 bit dan 5672359.6 ms untuk 15360 bit.
5. *Big-O Notation* untuk algoritma *serpent* adalah O(N) yang termasuk pada *linier complexity* dimana N adalah banyak *block* yang harus dikerjakan untuk melakukan enkripsi atau dekripsi suatu *file*. *Big-O Notation* untuk algoritma *Diffie-Hellman* adalah $O((\log N)^4)$ yang termasuk pada *logarithmic complexity* dimana N adalah ukuran dari bit *key*.

Daftar Pustaka

[1] M. Mircea, "Addressing Data Security in the Cloud," *World Academy of Science, Engineering and Technology,* vol. 6, 2012.

[2] O. K. Jasim, S. Abbas, E.-S. M. El-Horbaty and A.-B. M. Salem, "Cloud Computing Cryptography "State-of-theArt"," *World Academy of Science, Engineering and Technology,* vol. 7, 2013.

[3] W. Stallings, Cryptography and Network Security - Principle and Practice, Pearson Education, Inc., 2011.

[4] R. Sadikin, Kriptografi untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java, Yogyakarta: Andi, 2011.

[5] R. Anderson, E. Biham and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard," 1998.

[6] D. W. and M. Hellman, "“New directions in cryptography”," *IEEE Transactions on Information Theory,* 1976.

[7] J. N and J. J, "A Cloud Storage System with Daata Confidentiality and Data Forwarding," *International Journal of Soft Computing and Engineering (IJSCE),* vol. 3, no. 1, pp. 391-394, 2013.

[8] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST , Gaithersburg, 2011.

[9] M. Evans, T. Huynh, K. Le and M. Singh, "Cloud Storage," 2011.