Implementasi Perangkat Lunak Verifikasi Integritas File Terenkripsi Pada Server Cloud

Muhammad Fajri Salam, Muchammad Husni dan Henning Titi Ciptaningtyas
Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi,
Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: fajrisalam289@gmail.com, husni@if.its.ac.id, henning@if.its.ac.id

***Abstrak*— *Cloud* adalah teknologi populer yang memungkinkan untuk mengakses data melalui internet yang bahkan dapat menyimpan data sebagai pengganti penyimpanan lokal. *Cloud* memungkinkan pengguna untuk menyimpan data mereka di *cloud* tanpa perlu khawatir akan keakuratan dan keandalannya. Namun menyimpan data di *cloud* menimbulkan tantangan keamanan tertentu. Mengalihdayakan data di cloud menyebabkan pemilik data kehilangan kontrol fisik atas data mereka. Penyedia Layanan *Cloud* tertentu dapat mengakses data dari *cloud* secara tidak legal dan menjualnya kepada pihak ketiga untuk mendapatkan keuntungan. Jadi, meskipun outsourcing data di *cloud* tidak mahal dan mengurangi kompleksitas penyimpanan dan pemeliharaan berdurasi lama, setidaknya ada jaminan integritas data, privasi, keamanan, dan ketersediaan di server cloud. Penelitian ini akan berfokus pada strategi verifikasi integritas untuk data outsourcing. Skema yang diusulkan adalah menggabungkan mekanisme enkripsi dengan verifikasi integritas. Skema enkripsi yang digunakan di sini adalah algoritma kriptografi AES-256 dan fungsi *hash* SHA-256 yang digunakan untuk memastikan kebenaran penyimpanan data pada server yang tidak terpercaya.**

Kata Kunci—Cloud Computing, AES-256, SHA-256..

I. PENDAHULUAN

Seiring dengan perkembangan jaman, teknologi saat ini mengalami perkembangan kearah pencapaian kemudahan dan kenyamanan yang luar biasa, sehingga kegiatan sehari-hari yang di anggap tidak mungkin di kerjakan dalam waktu yang singkat menjadi mungkin untuk dilakukan secara singkat. Pengembangan teknologi komputasi berbasis internet saat ini lebih di arahkan pada proses aplikasi sistem yang mudah dan tidak memerlukan banyak waktu dan tenaga [1].

Perkembangan teknologi komputasi berbasis internet ini salah satunya dikenal sebagai *Cloud Computing*. Ada banyak jenis implementasi dari *Cloud Computing*, salah satunya adalah penyimpanan data secara *online*. Keunggulan menyimpan data secara *online* adalah pengguna dapat mengakses datanya dimanapun dan kapanpun dengan internet.

Perkembangan *Cloud Computing* ini tidak selalu berujung ke hal-hal yang positif. Ada juga penyedia layanan *Cloud Computing* yang melakukan penyalahgunaan data pengguna. Bentuk penyalahgunaan data ini berupa pencurian data, penjualan data dan perubahan data [1].

Ada berbagai macam cara untuk mengamankan data pada server *cloud*, salah satunya adalah dengan cara verifikasi integritas file. Verifikasi integritas file dapat menemukan data yang tidak sesuai atau termodifikasi, sehingga dengan menerapkan verifikasi integritas file dapat dipastikan bahwa file yang tersimpan adalah file yang sesuai atau valid.

Makalah ini membahas tentang konsep membangun sistem yang dapat melakukan verifikasi integritas file terenkripsi dengan menggunakan fungsi *hash* SHA-256, mengenkripsi setiap file yang tersimpan dengan menggunakan AES-256 dan melakukan pengembalian file apabila terdapat file yang hilang atau termodifikasi pada server *cloud.*

II. DASAR TEORI

Bagian ini menjelaskan dasar teori dari verifikasi file, *advance encryption standard* dan *secure hash algorithm* yang diterapkan pada penelitian ini.

A. Verifikasi File

Verifikasi file adalah suatu proses untuk memverifikasi integritas file di komputer. Hal ini dapat dilakukan dengan membandingkan tiap-tiap bagian dari file, tetapi membutuhkan dua Salinan dari file yang sama. Pendekatan yang lebih popular adalah menghasilkan *hash* dari file yang disalin dan membandingkannya dengan *hash* dari file asli [2].

Verifikasi file berbasis *hash* memastikan bahwa file tidak rusak dan termodifikasi dengan membandingkan nilai *hash* file dengan nilai yang dihitung sebelumnya. Jika nilai-nilai ini cocok, file dianggap tidak dimodifikasi. Karena sifat fungsi *hash*, ketidak cocokan dapat menyatakan bahwa file mengalami modifikasi.

B. Advance Encryption Standard

Advanced Encryption Standard (AES) merupakan algoritma kriptografi yang dapat digunkan untuk mengamakan data. Algoritma AES adalah blok chipertext simetrik yang dapat mengenkripsi (encipher) dan dekripsi (decipher) informasi. AES dipilih karena kuat terhadap serangan differential, serangan truncated differential, serangan linear, serangan interpolation, dan serangan square [3].

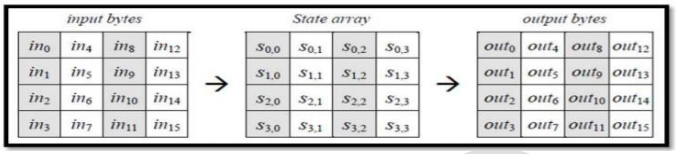
Input dan output dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data yang sudah terbentuk dalam satu kelompok 128 bit tersebut disebut juga sebagai blok data atau plaintext yang nantinya akan dienkripsi menjadi ciphertext. Cipher key dari AES terdiri dari key dengan panjang 128 bit, 192 bit, atau 256 bit. Perbedaan panjang

kunci akan mempengaruhi jumlah round yang akan diimplementasikan pada algoritma AES ini. Berikut ini adalah Tabel 1 yang memperlihatkan jumlah round / putaran (Nr) yang harus diimplementasikan pada masingmasing panjang kunci.

Tabel 1 Jumlah Perbandingan Round dan Key [4]

	Jumlah Key (Nk)	Ukuran Block (Nb)	Jumlah Putaran (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Pada dasarnya, operasi AES dilakukan terhadap array of byte dua dimensi yang disebut dengan state. State mempunyai ukuran NROWS X NCOLS. Pada awal enkripsi, data masukan yang berupa in0, in2, in3, in4, in5, in6, in7, in8, in9, in10, in11, in12, in13, in14, in15 disalin ke dalam array state. State inilah yang nantinya dilakukan operasi enkripsi / dekripsi. Kemudian keluarannya akan ditampung ke dalam array out. Gambar 1 mengilustrasikan proses penyalinan dari input bytes, state array, dan output bytes.



Gambar 1 Proses Input Byte, State Array, dan Output Bytes

Pada saat permulaan, input bit pertama kali akan disusun menjadi suatu array byte dimana panjang dari array byte yang digunakan pada AES adalah sepanjang 8 bit data. Array byte inilah yang nantinya akan dimasukkan atau dicopy ke dalam state dengan urutan dimana r (row / baris) dan c (column/kolom):

$$s[r,c] = in[r + 4c] \; untuk \; 0 \leq r < 4 \; dan \; 0 \leq c < Nb$$

sedangkan dari state akan dicopy ke output dengan urutan:

$$out[r + 4c] = s[r,c] \; untuk \; 0 \leq r < Nb$$

C. Secure Hash Algorithm

Algoritma SHA-256 dapat digunakan untuk menghitung nilai message digest dari sebuah pesan, dimana pesan tersebut memiliki panjang maksimum 2^{64} bit. Algoritma ini menggunakan sebuah message schedule yang terdiri dari 64 element 32-bit word, delapan buah variabel 32-bit, dan variabel penyimpan nilai *hash* 8 buah word 32-bit. Hasil akhir dari algoritma SHA-256 adalah sebuah message digest sepanjang 256-bit [5].

Cara Kerja SHA-256 mengubah pesan masukan ke dalam message digest 256 bit. Berdasarkan Secure *Hash* Signature Standard, pesan masukan yang panjangnya lebih pendek dari 2^{64} bit, harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah message diggest 256-bit [6].

Tahapan-tahapan cara kerja SHA-256 adalah sebagai berikut [6]:

1. *Message Padding*: Pada tahap pertama, pesan yang berupa binary disisipkan dengan angka 1 dan ditambahkan bit-bit pengganjal yakni angka 0 hingga panjang pesan tersebut kongruen dengan 448 modulo 512. Panjang pesan yang asli kemudian ditambahkan sebagai angka biner 64 bit. Setelah itu maka panjang pesan sekarang menjadi kelipatan 512 bit.
2. *Parsing*: Pesan yang sudah dipadding tadi kemudian dibagi menjadi N buah blok 512 bit: $M^{(1)}$, $M^{(2)}$, ..., $M^{(N)}$.
3. *Message Expansion*: Masing-masing blok 512-bit tadi lalu dipecah menjadi 16 buat word 32-bit: $M_0^{(i)}$, $M_1^{(i)}$, ..., $M_{15}^{(i)}$ yang mana nantinya diperluas menjadi 64 word yang diberi label W_0, W_1, ..., W_{63} dengan aturan tertentu yang sudah ditentukan sebelumnya oleh standar SHA-2.

Tabel 2 Nilai Awal pada Variabel H

$A = H_0^{(0)}$	6a09e667
$B = H_1^{(0)}$	bb67ae85
$C = H_2^{(0)}$	3c6ef372
$D = H_3^{(0)}$	a54ff53a
$E = H_4^{(0)}$	510e527f
$F = H_5^{(0)}$	9b05688c
$G = H_6^{(0)}$	1f83d9ab
$H = H_7^{(0)}$	5be0cd19

4. *Message Compression*: Masing-masing dari 64 word yang diberi label W_0, W_1, ..., W_{63} tadi kemudian diproses dengan algoritma fungsi *hash* SHA-256. Dalam proses tersebut, inti utama dari algoritma SHA-256 adalah membuat 8 variabel yang diberikan nilai untuk nilai awal dari $H_0^{(0)}$ - $H_7^{(0)}$ di awal masing-masing fungsi *hash*. Nilai-nilai awal tersebut dapat dilihat pada Gambar 2.
5. Algoritma ini melakukan perhitungan sebanyak 64 kali putaran untuk setiap perhitungan blok. Delapan variabel yang diberi label A, B, C, ..., H tadi nilainya terus berganti selama perputaran sebanyak 64 kali putaran sebagai berikut:

$$T_1 = H + \sum_1 (E) + Ch(E,F,G)[1] + K_t + W_t \quad (1)$$
$$T_2 = \sum_0 (A) + Maj(A,B,C)[1] \quad (2)$$
$$H = G \quad (3)$$
$$G = E \quad (4)$$
$$F = E \quad (5)$$
$$E = D + T_1 \quad (6)$$
$$D = C \quad (7)$$
$$C = B \quad (8)$$
$$B = A \quad (9)$$
$$A = T_1 + T_2 \quad (10)$$

6. Setelah perputaran sebanyak 64 kali tadi, nilai *hash* $H^{(i)}$ kemudian dihitung sebagai berikut:

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

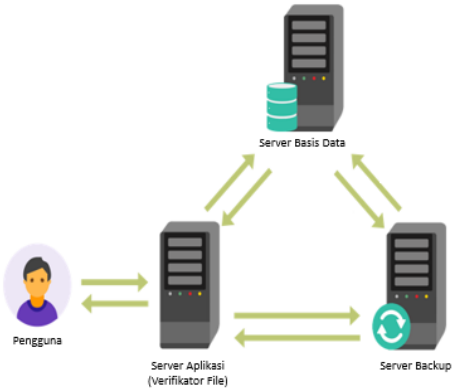
7. Selanjutnya hasil akhir SHA-256 didapat dari penggabungan delapan variabel yang tadi sudah dikomputasi.

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

III. DESAIN

Terdapat tiga komponen utama dari arsitektur sistem yaitu:

- Server Aplikasi
  Server Aplikasi: Server utama yang menjalankan aplikasi dan melakukan verifikasi terhadap data pengguna.
- Server Basis Data
  Server Basis Data: Server yang menyimpan basis data untuk sistem yang berjalan.
- Server Backup
  Server Backup: Server yang menyimpan cadangan data dari server aplikasi dan server basis data.



Gambar 2 Arsitektur Sistem

Pengguna mengunggah data ke server aplikasi melalui aplikasi. Data yang telah diunggah tersebut dienkrip oleh server aplikasi dengan menggunakan AES-256 lalu disimpan di server aplikasi dan server backup. Dari data terenkrip tersebut sistem akan menghasilkan *secret hah key* dengan menggunakan SHA-256 untuk verifikasi integritas data dan mengirimkannya ke server basis data bersamaan dengan informasi dari data asli yang diunggah. Data yang tersimpan di server aplikasi diverifikasi dengan membandingkan *secret hash key* yang ada pada data dengan yang tersimpan pada basis data. Jika terdapat data yang tidak terverifikasi maka server backup akan memberikan data yang benar ke server aplikasi, sehingga bisa nantinya bisa dipastikan kalua data yang tersimpan pada server aplikasi adalah data yang valid.

Tahap 1: Inisialisasi

Pada tahap ini akan menjelaskan mekanisme mengunggah data ke sistem:

a. Pengguna mengunggah file ke sistem
b. File dienkripsi menggunakan AES-256
c. File ternkrip disimpan dan dikirim ke server backup
d. Menghasilkan *secret hash key* menggunakan SHA-256
e. *Secret hash key* dan informasi file disimpan basis data.

Tahap 2: Verifikasi

Pada tahap ini akan menjelaskan bagaimana mekanisme verifikasi data pada sistem:

a. Server Aplikasi membaca basis data.
b. *Hash key* setiap file dibandingkan dengan *secret hash key* yang tersimpan di basis data.
c. Sistem akan memperbarui status file pada basis data apabila terdapat file yang tidak terverifikasi.

Tahap 3: Pengembalian Data

Pada tahap ini akan menjelaskan mekanisme pengembalian data pada sistem:

a. Server backup membaca basis data.
b. Server basis data mengirimkan file ke server aplikasi apabila ada file yang tidak terverifikasi pada server aplikasi dan memperbarui status file pada basis data.

IV. IMPLEMENTASI

Implementasi yang akan dijelaskan meliputi lingkungan pembangunan sistem, implementasi mekanisme pengunggahan file, implementasi mekanisme verifikasi integritas file dan implementasi pengembalian file.

A. Lingkungan Pembangunan Sistem

Lingkungan sistem yang digunakan untuk membangun perangkat lunak ini:

1. Windows 10 Enterprise sebagai sistem operasi
2. Sublime Text Editor 3128 sebagai *Integrated Development Environment (IDE)*
3. Laravel 5.8 sebagai kerangka kerja (*framework*)
4. PHP 7.2.12 sebagai bahasa pemrograman yang digunakan.
5. Python 3.7.3 sebagai bahasa pemrograman untuk program pihak ketiga.
6. MariaDB Server 10.1.37 (MySQL) dan HeidiSQL sebagai sistem manajemen basis data
7. Apache 2.4.37 sebagai *web server*

B. Implementasi Pengunggahan File

Fungsi Pengunggahan File direpresentasikan pada fungsi upload dengan parameter file yang diunggah. *Pseudocode* fungsi ini dapat dilihat pada Kode Sumber 1

```
1.  function upload(file)
2.      GET file
3.      validate file
4.
5.      SET key encryption
6.      encrypt file
7.      save encrypted file
8.      send encrypted file to application
    server
9.      generate hash key
10.
11.     insert file, execution time, hash key
    into     file model
12.     insert file into log model
```

Kode Sumber 1 *Pseudocode* Fungsi Pengunggahan File

C. *Implementasi Verifikasi File*

Script verifikasi file ini dibangun menggunakan bahasa pemrograman Python. *Pseudocode* Fungsi Verifikasi File ini bisa dilihat pada Kode Sumber 2.

```
1.  GET data from file model
2.  FOR row in record THEN
3.      GET file from storage
4.      IF file not valid THEN
5.          update invalid into file model
6.      ELSE IF file not found
7.          update not found into file model
8.      ENDIF
9.  ENDFOR
```

Kode Sumber 2 *Pseudocode* Fungsi Verifikasi File

D. *Implementasi Pengembalian File*

Pada server backup diberikan sebuah *script* dengan bahasa pemrograman Python yang memiliki fungsi sebagai pemulihan file yang telah hilang atau termodifikasi. *Script* yang digunakan ini dibangun menggunakan bahasa pemrograman Python. *Script* ini berjalan setiap lima menit pada server backup. *Pseudocode script* ini dapat dilihat pada kode sumber 3.

```
1.  GET data from file model
2.      FOR row in record THEN
3.          GET file from storage
4.          SEND file to aplication server
```

Kode Sumber 3 *Pseudocode* Fungsi Pengembalian File

V. UJI COBA DAN EVALUASI

A. *Uji Coba Fungsionalitas Sistem*

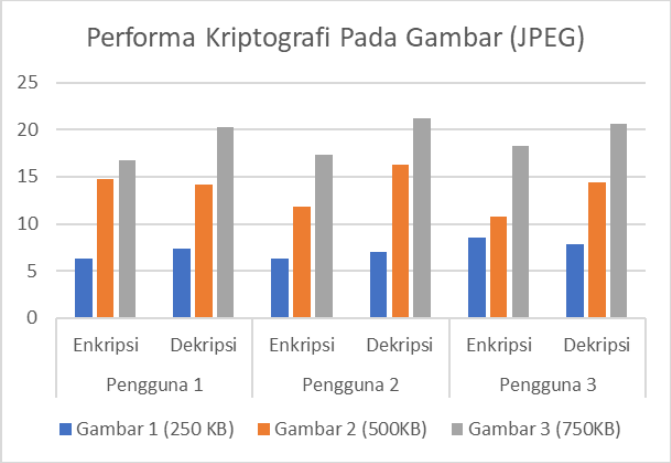
Uji coba dilakukan dengan mencoba setiap kebutuhan fungsional yang terdapat pada sistem. Uji coba menghasilkan semua kebutuhan fungsional pada sistem yang dirancang berjalan semua.

B. *Uji Coba Performa Kriptografi*

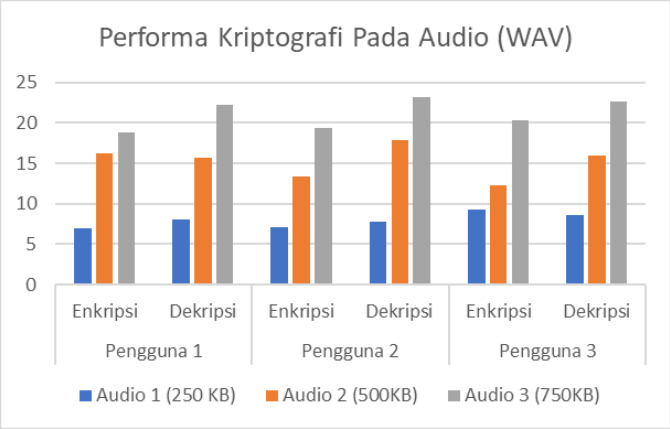
Uji coba dilakukan dengan melakukan unggah dan unduh file pada sistem dengan tiga jenis pengguna, empat jenis file dan empat variasi ukuran file. Jenis file yang digunakan untuk uji coba adalah gambar, audio, video dan dokumen sedangkan variasi ukuran file adalah 250 KB, 500 KB dan 750 KB. Setiap pengguna melakukan uji coba terhadap file sebanyak lima kali kemudian dihitung rata-rata dari kelima percobaan tersebut.

Tabel 3 Uji Coba Fungsionalitas

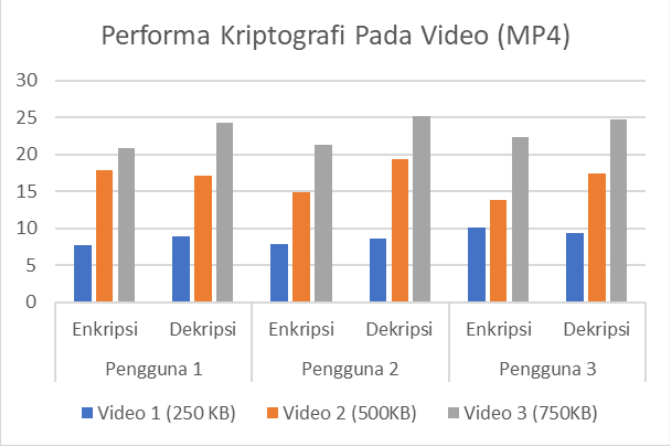
No	Uji Coba	Harapan	Hasil
1	Pengguna mengunggah file ke sistem	File pengguna dienkripsi dan tersimpan di server aplikasi	OK
		File terenkripsi pengguna tersimpan di server backup	OK
		Data file dan aktivitas pengguna tersimpan pada basis data	OK
2	Menjalankan *script* verifikasi file di server aplikasi	Data file pada basis data akan diperbarui apabila file yang diverifikasi pada penyimpanan server aplikasi mengalami modifikasi ataupun hilang	OK
		Data aktivitas sistem tersimpan pada basis data	OK
3	Menjalankan *script* pengembalian file di server backup	File yang telah hilang pada server aplikasi dikembalikan	OK
		Data file yang telah dikembalikan pada server aplikasi akan diperbarui pada basis data	OK
		Data aktivitas sistem tersimpan pada basis data	OK



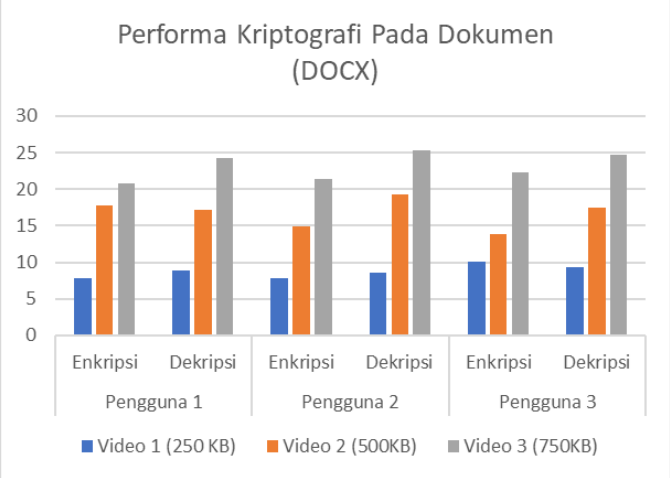
Gambar 3 Grafik Performa Kriptografi Terhadap Jenis File Gambar



Gambar 4 Grafik Performa Kriptografi Terhadap Jenis File Audio



Gambar 5 Grafik Performa Kriptografi Terhadap Jenis File Video



Gambar 6 Grafik Performa Kriptografi Terhadap Jenis File Dokumen

VI. KESIMPULAN

Kesimpulan yang diperoleh dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Penelitian ini telah mengaplikasikan algoritma enkripsi AES-256 pada aplikasi berbasis *website* dengan kerangka kerja Laravel
2. Sistem dapat melakukan verifikasi integritas file untuk memastikan bahwa file yang tersimpan tidak mengalami modifikasi
3. Sistem dapat mengembalikan data yang telah hilang atau termodifikasi pada server cloud

Saran yang diberikan dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Menambahkan server untuk server basis data dan server backup untuk replikasi data.
2. Untuk meningkatkan kredibilitas data pada sistem diperlukan verifikasi pada basis data.

DAFTAR PUSTAKA

[1] S. Shaikh and D. Vora, "Secure Cloud Auditing Over Encrypted Data," p. 5, 2016.

[2] S. F. O. E.-H. David B Little, Digital Data Integrity: The Evolution from Passive Protection to Active Management, San Fransisco: John Wiley & Sons, Ltd, 2007

[3] W. Stallings, The Advanced Encryption Standard., San Jose: The Internet Protocol Journal, 2001.

[4] V. Yuniati, Enkripsi Dan Dekripsi Dengan Algoritma Aes 256 Untuk Semua Jenis File, Jurnal Informatika Universitas Kristen Duta Wacana, 2009.

[5] A. Sebastian, Implementasi dan Perbandingan Performa Algoritma *Hash* SHA-1, SHA-256 dan SHA-512,

[6] Bandung: Institut Teknologi Bandung, 2007.
R. d. N. S. I. Mankar, Implementation of SHA-256 Algorithm, Pune: Pune University, 2013.