PENERAPAN ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES-256) DENGAN MODE CBC DAN *SECURE HASH ALGORITHM* (SHA-256) UNTUK PENGAMANAN DATA FILE

SKRIPSI

Oleh :

SELVIYANI

201710225097

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BHAYANGKARA JAKARTA RAYA

2021

LEMBAR PERSETUJUAN PEMBIMBING

Judul Skripsi	: Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File
Nama Mahasiswa	: Selviyani
Nomor Pokok Mahasiswa	: 201710225097
Program Studi / Fakultas	: Informatika / Ilmu Komputer
Tanggal Lulus Ujian Skripsi	: 14 Juli 2021

Bekasi, 19 Juli 2021

MENYETUJUI,

Pembimbing I	Pembimbing II
Ahmad Fathurrozi, S.E., M.M.S.I.	**Rakhmat Purnomo, S.Pd., S.Kom., M.Kom.**
NIDN. 0327117402	**NIDN. 0322108201**

LEMBAR PENGESAHAN

Judul Skripsi	:	Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File
Nama Mahasiswa	:	Selviyani
Nomor Pokok Mahasiswa	:	201710225097
Program Studi / Fakultas	:	Informatika / Ilmu Komputer
Tanggal Lulus Ujian Skripsi	:	14 Juli 2021

Bekasi, 19 Juli 2021

Mengesahkan,

Ketua Tim Penguji	: **R. Wisnu Prio Pamungkas, S.Kom., M.Kom.** **NIDN. 0321127201**	.....
Penguji (I)	: **Sugiyatno, S.Kom., M.Kom.** **NIDN. 0313077206**	.....
Penguji (II)	: **Ahmad Fathurrozi, S.E., M.M.S.I.** **NIDN. 0327117402**	.....

Mengetahui,

Ketua Program Studi Informatika	Dekan Fakultas Ilmu Komputer
Rakhmat Purnomo, S.Pd., S.Kom., M.Kom. **NIDN. 0322108201**	**Herlawati, S.Si., M.M., M.Kom.** **NIDN. 0311097302**

UNIVERSITAS BHAYANGKARA JAKARTA RAYA
FAKULTAS ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA

LEMBAR PERNYATAAN BUKAN PLAGIASI

Yang bertanda tangan dibawah ini :

Nama	: Selviyani
NPM	: 201710225097
Program Studi	: Informatika
Fakultas	: Ilmu Komputer
Judul Tugas Akhir	: Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File

Dengan ini menyatakan bahwa hasil penulisan skripsi yang telah saya buat ini merupakan **hasil karya saya sendiri dan benar keasliannya**. Apabila dikemudian hari penulisan skripsi ini merupakan plagiat atau penjiplakan terhadap karya orang lain, maka saya bersedia mempertanggungjawabkan sekaligus bersedia menerima sanksi berdasarkan tata tertib di Universitas Bhayangkara Jakarta Raya.

Demikian pernyataan ini saya buat dalam keadaan sadar dan tidak dipaksakan dari pihak manapun.

Bekasi, 19 Juli 2021

Selviyani
NPM 201710225097

ABSTRACT

Selviyani. 201710225097. *The implementation of Advanced Encryption Standard (AES-256) Algorithm with CBC Mode and Secure Hash Algorithm (SHA-256) for Data File Security.*

Data Security is one of the important things to protect important messages and information from corruption, compromise or loss so that messages and information remain safe. Encryption and description techniques are considered to be able to secure data properly by protecting files from being easily read or seen by unauthorized parties. In this case, the authors used data from University of Bhayangkara Jakarta Raya to be able to secure their university data using a cryptography symmetrical algorithm called Advanced Encryption Standard (AES) and Secure Hash Algorithm (SHA) as a solution to existing problems. The AES algorithm process is divided into four steps, the first step is SubBytes, the second step is ShiftRows, the third step is MixColums and the last step is AddRoundKey. And using the SHA algorithm as the hashing function. The algorithm is applied to a desktop-based file description and encryption application with the C sharp programming language.

Keywords: Data File Security, Encryption, Description, Algorithm AES-256, SHA-256

ABSTRAK

Selviyani. 201710225097. Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File.

Pengamanan data atau *data protection* merupakan salah satu hal penting untuk melindungi pesan dan informasi penting dari korupsi, kompromi atau kerugian supaya pesan dan informasi tersebut tetap aman. Teknik enkripsi dan deskripsi dinilai dapat mengamankan data dengan tepat dengan melindungi file agar tidak mudah untuk dibaca atau dilihat oleh pihak yang tidak berwenang. Pada penelitian ini penulis menggunakan data dari Universitas Bhayangkara Jakarta Raya untuk dapat mengamankan data universitas mereka menggunakan algoritma kriptografi simetris *Advanced Encryption Standard* (AES) dan *Secure Hash Algorithm* (SHA) sebagai solusi untuk masalah yang ada. Proses algoritma AES sendiri terbagi menjadi empat langkah, langkah pertama yaitu *SubBytes*, langkah kedua *ShiftRows*, langkah ketiga *MixColums* dan langkah terakhir yaitu *AddRoundKey*. Serta menggunakan algoritma SHA sebagai fungsi hashing-nya. Penerapan algoritma tersebut diterapkan ke dalam aplikasi enkripsi dan deskripsi file berbasis desktop dengan bahasa pemograman C sharp.

Kata Kunci : Pengamanan Data, Enkripsi, Deskripsi, Algoritma AES-256, SHA-256

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIK

Sebagai sivitas akademik Universitas Bhayangkara Jakarta Raya, saya yang bertanda tangan di bawah ini :

Nama	: Selviyani
NPM	: 201710225097
Program Studi	: Informatika
Fakultas	: Ilmu Komputer
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Bhayangkara Jakarta Raya **Hak Bebas Royalti Non-Esklusif (*Non-Exclusive Royalty-Free Right*)**, atas karya ilmiah saya yang berjudul :

"PENERAPAN ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES-256) DENGAN MODE CBC DAN *SECURE HASH ALGORITHM* (SHA-256) UNTUK PENGAMANAN DATA FILE"

beserta perangkat yang ada (bila diperlukan). Dengan hak bebas royalti non-ekslusif ini, Universitas Bhayangkara Jakarta Raya berhak menyimpan, mengalihmediakan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya dan mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemilik hak cipta.

Segala bentuk tuntutan hukum yang timbul atas pelanggaran hak cipta dalam karya ilmiah ini menjadi tanggung jawab saya pribadi

Demikian pernyataan ini saya buat dengan sebenarnya.

Bekasi, 19 Juli 2021

Selviyani

NPM 201710225097

KATA PENGANTAR

Dengan mengucap puji syukur atas nikmat yang diberikan oleh Allah SWT, dan tak lupa sholawat serta salam semoga tercurah kepada Uswah Khasanah Rasulullah SAW. Penulis dapat menyelesaikan tugas akhir ini. Penulisan tugas akhir ini dilakukan dalam rangka memenuhi salah satu syarat akademik untuk mencapai gelar Sarjana Komputer Program Studi Informatika pada Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya. Penulis menyadari bahwa tanpa bantuan dan bimbingan dari beberapa pihak, tugas akhir skripsi ini tidak dapat diselesaikan dengan segera.

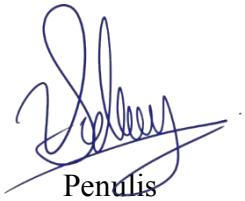
Oleh karena itu, penulis ingin menyampaikan ucapan terimakasih kepada semua pihak yang telah membantu dalam penyusunan Skripsi, dan penulis mengucapkan terimakasih kepada yang terhormat :

1. Bapak Irjen Pol. (Purn) Dr. Drs. Bambang Karsono, S.H., M,M selaku Rektor Universitas Bhayangkara Jakarta Raya.
2. Ibu Herlawati, S.Si., M.M., M.Kom., selaku Dekan Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.
3. Bapak Rakhmat Purnomo, S.Pd., S.Kom., M.Kom., selaku Ketua Program Studi Informatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.
4. Ibu Sri Rezeki, S.Kom., M.Kom., selaku Penasehat Akademik Kelas A2 angkatan 2017 Informatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.
5. Bapak Ahmad Fathurrozi, S.E., M.M.S.I., selaku Dosen Pembimbing Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.
6. Keluarga tercinta, Bapak dan Kakak yang selalu memberikan dukungan dan do'a untuk penulis.
7. Stephan Verbücheln yang selalu menemani dan memberi dukungan serta bantuan dalam menjalani proses penelitian ini.
8. Saudari Neina Corina, Eno Widyasari dan seluruh teman-teman seperjuangan di Program Informatika Angkatan 2017 yang selalu memberikan dukungan selama ini.

Karena kebaikan dari beliau-beliau, maka penulis dapat menyelesaikan skripsi ini dengan baik. Penulis menyadari bahwa penyusunan laporan ini tidak luput dari kesalahan dan kekurangan, maka dengan segala kerendahan hati, penulis menerima kritik dan saran yang membangun dari pembaca.

Akhir kata, semoga Allah Yang Maha Pengasih dan Maha Penyayang melimpahkan berkah dan anugerah-Nya kepada semua pihak dan membalas semua amal ibadahnya. Penulis berharap semoga Skripsi ini dapat memberikan manfaat bagi pihak yang memerlukan.

Bekasi, 19 Juli 2021



Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Maraknya perkembangan teknologi di zaman serba digital sekarang ini sangatlah menunjang kegiatan manusia, terutama dibidang inovasi dan kretifitas dalam bekerja, belajar, penyebaran informasi dan lain lain. Sayangnya semakin maju teknologi di zaman modern ini, semakin maju pula tingkat kejahatan dengan teknologi sekarang. Oleh karena itu edukasi terhadap penggunaan teknologi sangat diperlukan, pengamanan data yang tepat diperlukan guna meningkatkan keamanan yang menjamin agar tidak disalah gunakan oleh orang yang tak bertanggung jawab untuk keperluan yang tidak semestinya. Banyaknya jenis kejahatan dimasa sekarang perlu diwaspadai, salah satunya *cyber crime*. *Cyber Crime* dapat diartikan sebagai tindak kenjahatan di dunia maya yang menjadikan teknologi komputer dan jaringan internet sebagai sasarannya.

Data yang bersifat pribadi menjadi objek yang disenangi oleh *hacker* untuk dimanupulasi, dipermaikan dan digunakan tidak pada semestinya. Oleh karena itu data yang bersifat pribadi atau rahasia perlu dijaga keamanannya. Ada beberapa teknik pengamanan data, diantaranya adalah teknik enkripsi. Enkripsi merupakan sebuah proses pengubahan sebuah pesan atau informasi dari yang bisa dimengerti atau dibaca menjadi sebuah pesan atau informasi yang sulit dimengerti hingga tidak terbaca sama sekali.Teknik enkripsi dapat mengamankan data karena data dapat berubah menjadi tidak terbaca sesuai dengan aslinya. Dan data yang terenkripsi dapat terbaca lagi apabila sudah di deskripsi dengan menggunakan kunci yang tepat. Dan dengan mengenkripsi data file yang penting atau rahasia dapat meningkatkan keamanan data yang bersifat rahasia tersebut.

Kriptografi merupakan studi matematika yang mempunyai hubungan dengan aspek keamanan informasi seperti integritas data, keaslian entitas dan keaslian data (Ratno Prasetyo, 2016). Dalam ilmu kriptografi terdapat dua proses penyandian yang disebut enkripsi dan deskripsi. Enkripsi dilakukan pada proses pengiriman pesan atau informasi dengan cara mengubah data asli kedalam bentuk kode kode yang menjadikannya data rahasia sedangkan deskripsi dilakukan pada

proses penerimaan dengan cara mengubah data yang berisi kode kode rahasia tersebut ke dalam bentuk data yang asli dan mudah dimengerti.

Pada tanggal 27 Desember 2020 berita tentang kebocoran data sensitif dari artis kenamaan ibu kota yang meramaikan *headline* berita di televisi dan *online*. Maka dari itu diperlukannya enkripsi file untuk file yang dianggap penting oleh user. Oleh karena itu universitas Bhayangkara Jakarta Raya sebagai perguruan tinggi swasta yang terletak di kota Bekasi, Jawa Barat. Pada Universitas Bhayangkara Jakarta Raya mempunyai banyak data file penting yang bersifat rahasia suatu lembaga, seperti data data keuangan dan data penting lainnya pada komputer atau laptop di lembaga mereka. Dan apabila data tersebut bisa saja dicuri dan dimanipulasi pada suatu kejadian yang dapat merugikan lembaga. Data keuangan yang tidak terenkripsi atau tidak dirahasiakan dapat dengan sangat mudah dimanipulasi oleh orang yang tidak bertanggung jawab untuk mengambil keuntungan di lembaga yang bergerak dibidang pendidikan tersebut, seperti dikorupsi pada jumlah pengeluaran untuk biaya operasional perusahaan tersebut dan biaya biaya lainnya dan apabila data data tersebut di hack oleh virus yang terjangkit di dalam computer seperti data keuangan dan data penting lainnya maka data tersebut akan terenkrip dengan virus dan tidak bisa dikembalikan lagi datanya. Oleh karena itu penulis berniat untuk menjadikan hal tersebut sebagai bahan penelitian penulis guna menyelesaikan tugas akhir untuk jenjang strata satu yang penulis tempuh. Dengan mengamankan data data rahasia lembaga pendidikan Universitas Bhyangkara Jakarta Raya menggunakan teknik enkripsi dan dekripsi yang penulis terapkan pada penelitian ini semoga dapat membantu lembaga tersebut untuk dapat menjaga kerahasiaan data mereka. Oleh karena itu, terdapat metode algoritma kriptografi yang cocok untuk memecahkan masalah pengamaanan data lembaga tersebut, yaitu salah satunya adalah metode AES dan SHA. Advanced Encryption Standar (AES) adalah algoritma kriptografi simetris modern yang beroperasi dalam mode penyandian blok (block cipher) yang memproses blok data dengan ukuran 128-bit dengan panjang kunci 128-bit, 192-bit, atau 256-bit (Asep Suryana, 2016). Terdapat beberapa mode dalam algoritma AES diantaranya mode CBC, ECB, OFB, CTR dan CFB untuk penyadian dengan metode *block cipher*. Penulis menggunakan mode CBC atau yang sering disebut *Cipher Block Chaining*

ialah metode penyandian blok berulang seperti rantai yang menggunakan veckor inisialisasi (IV) atau sering disebut deret biner unik dengan panjang tertentu untuk tiap enkripsi. Salah satu karakteristik utamanya CBC menggunakan mekanisme rantai yang membuat *chipertext* blok sebelumnya bergantung pada semua blok *chipertext* sebelumnya. Dengan segala pertimbangan mode operasi AES yang ada, penulis memilih untuk menggunakan mode CBC dengan segala kelebihan dan kekuranganya. SHA merupakan algoritma *Hashing* atau yang sering di sebut sebagai fungsi hash merupakan sebuah algoritma yang mengubah teks atau pesan menjadi sederetan karakter acak yang memiliki jumlah karakter yang sama (Ratno Prasetyo, 2016) yang dipublish oleh National Institute Of Standard and Technology (NIST) pada tahun 2001. SHA sendiri mempunyai beberapa jenis yaitu SHA-0, SHA-1 dan SHA-2. Untuk penelitian ini penulis memilih SHA-2 yang memiliki beberapa fungsi hash dengan *digest* yaitu 224, 256, 384 dan 512 bits atau SHA-224, SHA-256, SHA-384 dan SHA-512.

Algoritma kriptografi modern simetri tersebut terbukti pernah dipakai pada penelitian terdahulu mengenai pengamanan data oleh (Muammar Renaldy, 2015) Penelitian tersebut membahas tentang Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode AES dan SHA-1 pada Universitas Budi Luhur. Penelitian tersebut menguji metode AES dan SHA untuk mengkripsi aplikasi S-Diary. Dan algoritma AES ini juga pernah di gunakan oleh (Hadi Fajar, 2019) dalam penelitiannya yaitu Aplikasi Pengamanan File dan Pesan Teks Menggunakan Algoritma AES 256 dan SHA 256 Berbasis Android pada Universitas Pembangunan Nasional "Veteran" Jakarta.

Berdasarkan latar belakang permasalahan diatas maka penelitian yang dilakukan mengambil judul "Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File".

1.2 Identifikasi Masalah

Berdasarkan uraian diatas, maka dapat disimpulkan indentifikasi masalah sebagai berikut.

1. Kebocoran data file yang bersifat sensitif sering terjadi dalam kehidupan sehari hari dan menimbulkan kerugian untuk pihak yang dirugikan serta belum adanya penerapan yang baku untuk pengamanan data file penting pada Universitas Bhayangkara.
2. Di perlukannya sebuah aplikasi baku untuk pengamanan data file untuk menjaga kerahasiaan data tersebut dan belum adanya aplikasi pengamanan file pada Universitas Bhayangkara Jakarta Raya berbasis desktop.

1.3 Rumusan Masalah

Berdasarkan kesimpulan diatas, maka ditetapkan rumusan masalah dalam penelitaan ini sebagai berikut.

1. Bagaimana **Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File?**
2. Bagaimana **Perancangan Aplikasi Untuk Pengamanan Data File Menggunakan AES-256 Dengan Mode CBC dan SHA-256?**

1.4 Tujuan dan Manfaat

Maksud dari penulis dari penelitian pada pengamanan data di Universitas Bhayangkara adalah sebagai berikut.

1. Menerapkan algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) dalam pengamanan data file penting Lembaga kedalam aplikasi yang dibuat oleh penulis yang dinamai *Secret Fichier*. *Secret Fichier* sendiri adalah Bahasa Perancis yang berarti *Secret* yaitu rahasia dan *Fichier* berarti file. Jadi yang dimaksudkan *Secret Fichier* ialah file rahasia.

Sedangkan maksud dan tujuan penulisaan ini adalah untuk memenuhi syarat Skripsi pada Semester Tujuh Program Studi Informatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.

1.5 Batasan Masalah

Pembatasan permasalahan diharapkan tidak menyimpang dari pokok permasalahan, sehingga dalam penyeselesaian masalah ini akan dibatasi dimana ruang lingkup penelitian dilakukan untuk divisi administrasi dalam pengamanan data file penting lembaga yang dituangkan kedalam pembuatan aplikasi enkripsi dan deskripsi berbasis desktop. Adapun batasan masalah dalam pengimplementasian algoritma dari AES-256 dengan mode CBC dan SHA-256 ke dalam aplikasi enkripsi dan deskripsi berbasis desktop di sistem operasi Windows. Maka dari itu pembatasan tersebut akan dijelaskan dibawah ini :

1. Proses enkripsi dan deskripsi menggunakan algoritma AES dengan panjang kunci 256 bit dengan mode operasi CBC. Dan untuk Teknik memasukan kunci, dilakukan dengan proses hashing menggunakan SHA agar dapat menghasilkan kunci sebesar 256 bit.
2. Aplikasi enkripsi dan deskripsi Secret Fichier hanya dapat mengenkripsi dan mengdeskripsi file tunggal (bukan folder).
3. Aplikasi enkripsi dan deskripsi Secret Fichier ini mencakup data yang berjenis dokumen, gambar, suara dan video.
4. Untuk mengdeskripsi suatu file digunakan password (kunci) yang sama pada saat si pengguna mengenkripsi file tersebut.
5. Penulis menerapkan algoritma AES-256 mode CBC dan SHA-256 kedalam aplikasi dan menjelaskan tahapan tahapan algoritma tersebut bekerja kedalam penulisan ini.

1.6 Tempat dan Waktu Penelitian

Penelitian dilakukan di Fakultas Ilmu Komputer, Universitas Bhayangkara Jakarta Raya, Jl. Raya Perjuangan, Bekasi Utara, Kota Bekasi, Jawa Barat 17121, Indonesia. Selama empat bulan yaitu dari September 2020 sampai Desember 2020.

1.7 Sistematika Penulisan

Penelitian ini akan dibagi menjadi lima bab gambaran masing masing bab akan dijelaskan dibawah ini.

BAB I : PENDAHULUAN

Dalam bab ini berisi penjelasan tentang latar belakang masalah, maksud dan tujuan penelitian, rumusan masalah, pembahasaan masalah, metode pengumpulan data dan sistematika penulisan.

BAB II : LANDASAN TEORI

Dalam bab ini menjelaskan tentang memuat tinjauan dan ulasan singkat mengulas pentinnya penelitian yang dilakukan dan menyampaikan teori yang berhubungan dengan permasalahaan yang dibahas sebagai dasar analisa permasalahaan yang diteliti.

BAB III : METODOLOGI PENELITIAN

Dalam bab ini membahas tentang pendekatan studi dan dapat beruba analisis teori, metode eksperimen, kombinasi, rancangan, spesifikasi sistem baik perangkat keras maupun perangkat lunak.

BAB IV : PERANCANGAN SISTEM DAN IMPLEMENTASI

Dalam bab ini membahas mengenai penerapaan algoritma AES dan SHA serta perancangan aplikasi meliputi perangkat lunak berbasis dekstop, pengujian dan implementasi serta hasil keluaran dari sistem aplikasi yang telah dibuat dan di bahas sesuai penelitian dan hipotesis untuk menjawab permasalahan yang ada.

BAB V : PENUTUP

Dalam bab ini memuat beberapa kesimpulan yang di dapatkan dari penelitian dan menjawab tujuan penelitian atau hipotesis. Serta memuat saran saran yang dapat dikembangkan atau dilakukan sebagai penerapan untuk perusahaan kedepannya.

BAB II

LANDASAN TEORI

2.1 Kriptografi

Istilah kriptografi, *cryptography* berasal dari bahasa Yunani yaitu, "cryptos" yang artinya "*secret*" atau rahasia sedangkan "*graphien*" yang artinya "*writing*" atau tulisan, sehingga kriptografi berarti *secret writing* yang artinya tulisan rahasia. Pengertian kriptografi secara lebih luas adalah Kriptografi adalah ilmu yang mempelajari mengenai bagaimana cara mengamankan suatu informasi. Pengamanan ini dilakukan dengan mengenkrip informasi tersebut dengan suatu kunci khusus (Didi Surian, 2006). Dan Menurut *Request for Comments* (RFC), kriptografi merupakan cabang ilmu matematika yang berhubungan dengan transformasi data untuk embuatnya artinya tidak dapat dipahami (untuk menyembunyikan maknanya atau isi dari sebuah data), mencegahnya dari perubahan tanpa izin, atau mencegahnya dari penggunaan yang tidak sah. Jika transformasinya dapat dikembalikan, kriptografi juga bisa diartikan sebagai proses mengubah kembali data yang terenkripsi menjadi bentuk yang dapat dipahami. Jadi dapat disimpulkan bahwa kriptografi dapat diartikan sebagai cabang ilmu matematika untuk menjaga kerahasiaan informasi dengan metode teknik matematika yang mencakup, kerahasiaan, integritas data, autentifikasi, dan non repudiasi.

2.2 Tujuan Kriptografi

Menurut Alfred Menezez, Scott Vanstone dan Paul Oorschot dalam bukunya yang berjudul *Handbook of Applied Cryptography* (2006:4) terdapat empat tujuan mendasar dari kriptografi sebagai bentuk dari keamanan informasi, yaitu :

a. Kerahasiaan (*Confidentiality*)

   Menjaga kerahasiaan informasi dari semua pihak yang tidak berwenang yang mungkin mencoba membaca data atau informasi tersebut.

b. Keutuhan Data (*Integrity*)

Informasi tetap utuh atau tidak ada perubahan dalam proses pengiriman sampai informasi diterima oleh penerima.

c. Autentikasi (*Authentication*)

Pengenalan identitas baik secara kesatuan sistem maupun informasi itu sendiri untuk menjamin keaslian sumber.

d. Anti Penyangkalan (*Nonrepudiation*)

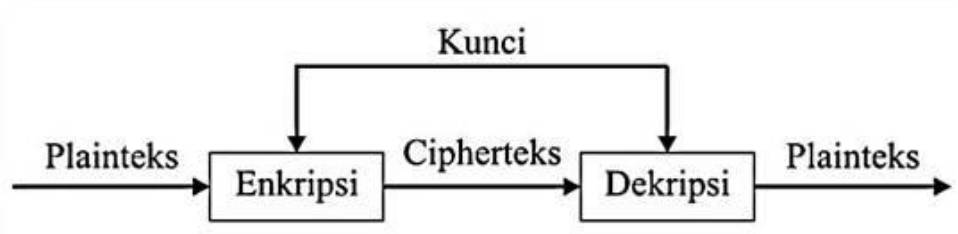
Mencegah terjadinya penolakan atau penyangkalan terhadap informasi yang dikirim atau diterima.

2.3 Jenis Jenis Algoritma Kriptografi

Algoritma Kriptografi dikatagorikan berdasarkan kunci yang dipakainya (Dony, 2008), yaitu :

a. Algoritma Simetri

Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama atau satu kunci untuk kegiatan enkripsi dan dekripsi (Dony, 2008). Jika mengirim pesan menggunaka algoritma ini maka si penerima harus diberitahu atau mengetahui kunci dari pesan yang telah di enkripsi oleh algoritma tersebut agar dapat di deskripsikan oleh penerima dari si pengirim. Secara umum, cipher yang termasuk dalam kriptografi simetri beroperasi dalam mode blok (*block cipher*), yaitu setiap kali enkripsi atau dekripsi dilakukan terhadap satu blok data yang berukuran tertentu , atau beroperasi dalam mode aliran (*stream cipher*), yaitu setiap kali enkripsi atau dekripsi dilakukan terhadap 1 bit atau 1 byte data.



Gambar 2.3.1 Algoritma Simetris

Berikut contoh contoh algoritma kriptogtafi yang memakai kunci simetris, ialah :

1. Blowfish
2. Twofish

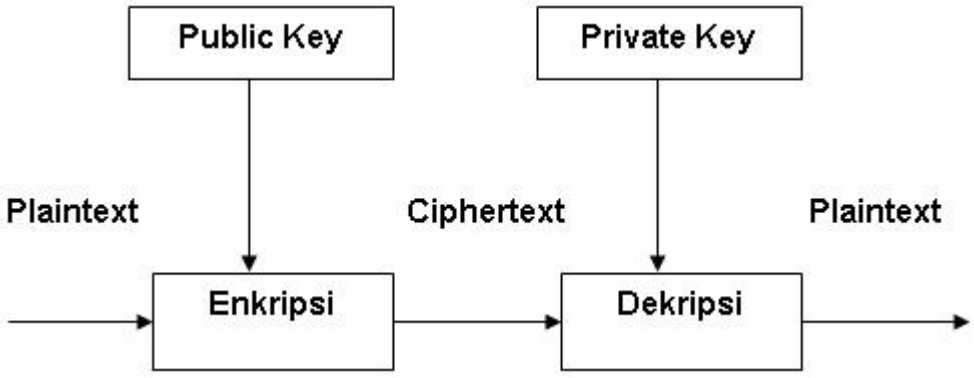
3. DES (Data Encryption Standard)
4. RC2, RC4, RC5, RC6
5. IDEA (International Data Encryption Algorithm)
6. AES (Advanced Encryption Standard), dan sebagainya

b. Algoritma Asimetris

Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda (Dony, 2008). Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu: (Dony, 2008)

1. Kunci umum (*public key*): Kunci yang boleh semua orang tahu (dipublikasikan).
2. Kunci rahasia (*private key*): Kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci kunci tersebut saling berhubungan atau berkaitan dengan satu sama lain. Dengan adanya kunci public si pengirim atau orang tersebut dapat mengenkripsikan pesan tapi tidak dengan mendeskripsikan pesan tersebut, karena berbeda kunci. Dan hanya si penerima atau orang yang mempunyai kunci pribadilah yang dapat mendeskripsikan pesan tersebut. Akan tetapi algoritma asimetris dapat melakukan pengiriman pesan yang lebih aman dari pada algorita simetris karena mempunyai dua kunci yaitu kunci publik dan kunci pribadi.



Gambar 2.3.2 Algoritma Asimetris

Berikut contoh contoh algoritma kriptografi yang memakai kunci asimetris, ialah :

1. DF (Diffle-Hellman)

2. DSA (*Digital Signature Algorithm*)
3. RSA
4. Kriptografi Quantum
5. ECC (*Elliptic Curve Cryptography*) dan sebagainya

c. Fungsi Hash (*Hashing Function*)

Fungsi hash adalah fungsi yang melakukan pemetan pesan dengan panjang sembarang ke sebuah teks khusus dengan panjang tetap (Santi Sulastri, 2018). Fungsi hash juga sering disebut dengan fungsi hash satu arah (one way function), message digest, fingerprint, fungsi kompresi dan message authentication code (MAC). Macam macam algoritma yang memakai fungsi hash ialah:

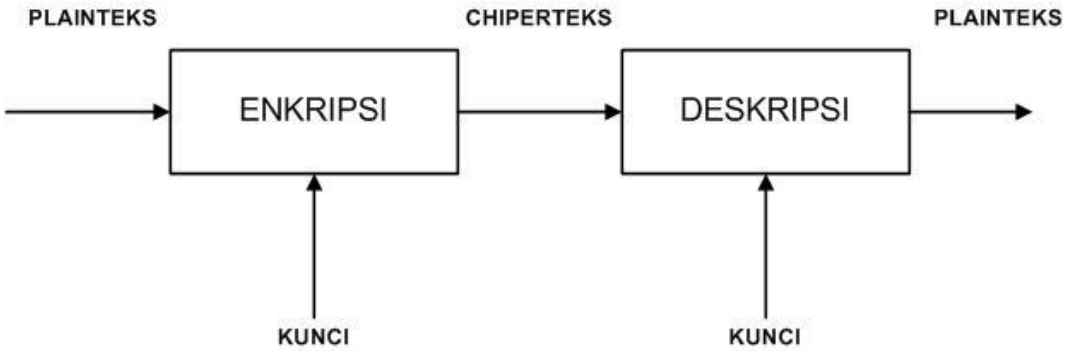
1. MD4, MD5
2. SHA-1, SHA-256, SHA-512
3. Snefru
4. N-Hash
5. RIPE-MD, dan sebagainya

2.4 Enkripsi dan Deskripsi

Dalam kriptografi terdapat proses didalamnya yang disebut sebagai proses enkripsi dan deskripsi. Proses penyandian pesan asli (*plain text*) menjadi pesan yang tidak dapat dibaca (*chipper text*) adalah enkripsi (Pandi Barita, 2018), sedangkan kebalikan dari proses enkripsi ialah deskripsi yaitu mengembalikan pesan yang sudah disandikan tersebut dan tidak dapat terbaca menjadi pesan aslinya yang dapat dibaca kembali, proses tersebut adalah deskripsi (Komariah Fitri, 2018). Pesan tersebut dapat data atau informasi yang berbentuk teks, dokumen, gambar serta suara yang bersifat penting dan rahasia.

Sistem yang mendasari terjadinya sebuah proses enkripsi dan dekripsi ialah hubungan antara dua himpunan yaitu yang berisi sebuah elemen pesan asli (*plaintext*) dan sebuah pesan yang berisi elemen pesan sandi (*ciphertext*). Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. P adalah notasi yang digunakan untuk *plaintext*, C adalah *ciphertext*, E adalah fungsi

Encryption dan D adalah fungsi *Decryption*. Sedangkan untuk kunci dapat dinotasikan sebagai K atau *Key*. Berikut gambar proses enkripsi dan dekripsi :



Gambar 2.4 Proses Enkripsi dan Deskripsi

2.5 *Advanced Encryption Standard* (AES-256)

Ilmu kriptografi terus berkembang dari jaman ke jaman, usaha untuk menjaga kerahasiaan suatu informasi telah ada sejak jaman dahulu kala. Julius Cesar, kaisar romawi telah menggunakan metode enkripsi sederhana pada jamannya untuk mengamankan sebuah informasi yang diterimannya atau informasi yang dikirimnya yang bersifat rahasia. Teknik kriptografi enkripsi yang dia gunakan pada saat itu sangat sederhana, yaitu menggeser setiap karakter dalam pesannya dengan nilai tertentu. Cara tersebut cukup aman pada jamannya, akan tetapi pada teknik enkripsi tersebut dinilai tidak aman lagi karena kemampuan komputasi komputer semakin perkembang dan sangat mudah dipecahkan.

Hingga tahun 1990-an, algoritma kriptografi terus berkembang pada saat itu algoritma DES sangat popular yaitu algoritma kriptografi *Data Encryption Standard*. DES termasuk algoritma enkripsi chipper block, ia adalah cikal bakal algoritma AES lahir. Lalu dari tahun ketahun sering berkembangnya teknologi, algoritma DES yang memiliki 56 bits tidak lagi mampu mempertahankan eksistensinya dan tidak lagi memadai. Pada tahun 1997 kontes pemilihan standard algoritma kriptografi baru pengganti DES dimulai. Diikuti para *crytopgrapher* dari seluruh dunia, ada 21 peserta di dalam kontes tersebut. Algoritma *Rijndael* yang keluar sebagai pemenang kontes pencarian algoritma kriptografi yang baru pengganti DES yang di selanggarakan oleh NIST (National Institute of Standard and Technology) milik pemerintah Amerika Serikat. Algoritma kriptografi *Rijndael*

didesain oleh dua pemudah asal Belgia yang bernama Vincent Rijmen dan John Daemen. Karena algoritma *Rijndael* ini yang paling memenuhi kriteria pada kontes pencarian algoritma pengganti DES pada saat itu. *Rijndael* diumumkan oleh NIST sebagai Standar Pemrosesan Informasi Federal (FIPS) publikasi 197 (FIPS 197) pada tanggal 26 November 2001 setelah proses standarisasi selama 5 tahun, di mana ada 15 desain enkripsi yang disajikan dan dievaluasi, sebelum *Rijndael* terpilih sebagai yang paling cocok. algoritma *Rijndael* terpilih sebagai algoritma kriptografi yang selain aman juga efisien dalam implementasinya dan dinobatkan sebagai *Advanced Standard Encryption*. AES efektif menjadi standar pemerintah Federal pada tanggal 26 Mei 2002 setelah persetujuan dari Menteri Perdagangan (Hadir Fajar, 2017).

Algoritma AES mendukung berbagai variasi ukuran kunci yang digunakannya. Jenis ukuran kunci yang algoritma AES terbagi tiga, yaitu AES-128, AES-193 dan AES-256. Perbedaan jenis ukuran block dan kunci yang algoritma AES miliki yaitu karena perbedaan ukuran kunci yang akan menentukan jumlah proses yang harus dilalui pada saat pengenkripsian dan pengdeskripsian atau lebih mudahnya dan dapat disimpulkan perbedaan pada banyaknya *round* atau putaran yang dipakai pada proses enkripsi dan deskripsi. Akan tetapi dari ketiga ukuran kunci yang AES miliki, ketiganya memiliki ukuran blok yang sama, yaitu 182 bit. Berikut ini adalah tabel yang memperlihatkan jumlah putaran dalam pemprosesannya yang harus diimplemetasikan pada masing masing panjang kunci yang AES miliki.

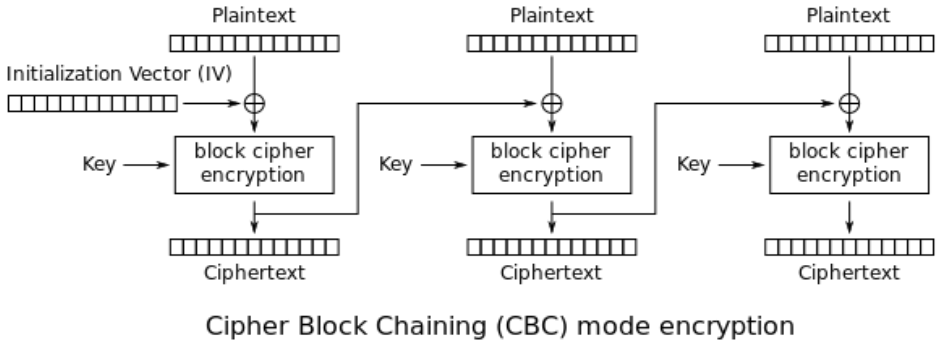
Tabel 2.5 Perbadingan Jumlah Key dan Round Algoritma AES

	Jumlah Key (Nk)	Ukuran Block (Nk)	Jumlah Putaran (Nr)
AES-128	128 bit	128 bit	10
AES-192	192 bit	128 bit	12
AES-256	256 bit	128 bit	14

2.6 AES Mode CBC (Chiper Block Chaining)

Algoritma kriptografi simetris Advanced Encryption Standard (AES) selain mempunyai beberapa panjang kunci yaitu 128, 192 dan 256 juga mempunyai

beberapa mode operasi, walaupun AES hanya bisa meng-enkripsi blok dengan panjang 128 bits (16 bytes) tetapi untuk meng-enkripsi file yang lebih panjang mode operasi penting dipertimbangkan. Salah satu mode operasi AES yaitu CBC atau yang disebut *Chiper Block Chaining*. Pada algoritma *block chiper* seperti AES ini, plaintext atau pesan mentah yang masuk untuk diproses dengan panjang yang tetap yaitu n, akan tetapi jika ukuran datanya terlalu panjang maka dilakukan pemecahan data data tersebut menjadi beberapa blok blok dengan ukuran yang lebih kecil dan sama. Maka dari itu mode operasi diperlukan, salah satunya ialah CBC. Pada CBC, rangkaian bit-bit pada plaintext dibagi menjadi blok blok bit dengan panjang yang sama (Henry. 2016). Mode CBC memerlukan IV (*initialization vector*) untuk menggabungkan dengan plaintext pertama. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini.



Gambar 2.6 Mode Operasi Chiper Block Chaining

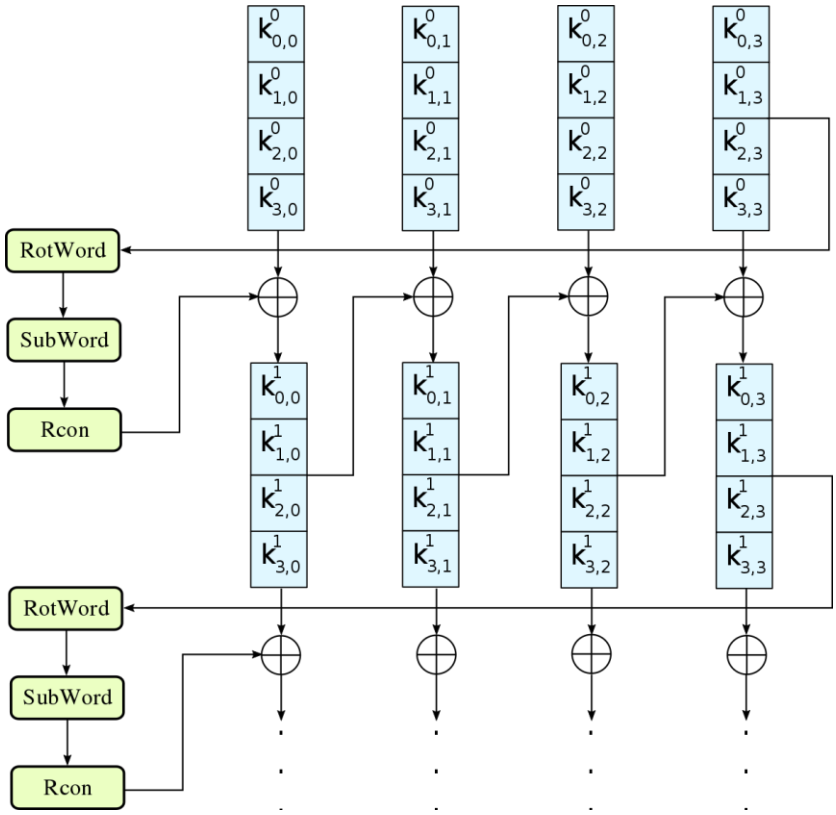
2.7 *Padding*

Pada proses penyandian blok, *plaintext* dengan ukuran data yang besar akan di pecah-pecah menjadi blok blok yang lebih kecil. Akan tetapi jika dalam pemecahaan tersebut menghasilkan blok data yang kurang dari jumlah data dalam blok tersebut maka akan dilakukan proses *padding* yaitu penambahan beberapa bit. Padding dilakukan dengan mengisi byte bernilai N bila dibutuhkan padding sebanyak N byte. Sebagai contoh, bila dibutuhkan padding 3 byte, maka padding berisi ’03 03 03′, bila dibutuhkan padding 5 byte, maka padding berisi ’05 05 05 05 05′ (Awang Harsa, 2016). Untuk menggunakan CBC, *padding* diperlukan. Dalam aplikasi *Secret Fichier*, yaitu aplikasi enkripsi dan deskripsi yang penulis buat untuk lembaga Pendidikan Universitas Bhayangkara Jakarta Raya Kelas AES-256

dari bahasa C# yang digunakan ini menggunakan padding PKCS7. Padding PKCS7 adalah generalisasi padding PKCS5 (juga dikenal sebagai padding standar). Padding PKCS7 bekerja dengan menambahkan N byte dengan nilai chr (N), dimana N adalah jumlah byte yang dibutuhkan untuk membuat blok akhir data berukuran sama dengan ukuran blok.

2.8 *Key Schedule*

Key schedule atau penjadwalan kunci dilakukan dengan tujuan mendapatkan kunci ronde atau *RoundKeys* yang akan digunakan untuk proses enkripsi dan dekripsi pada ronde rondenya, tepatnya pada tahap transformasi *AddRoundKey*. Tanpa proses pembangkitan kunci maka proses enkripsi dan enkripsi tidak akan berjalan sebagaimana mestinya. Berikut ini adalah gambar proses dari key schedule :



Gambar 2.8 proses Key Schedule

2.8.1 Pengertian RotWord, SubWord, dan Rcon

a. RotWord adalah proses pergeseran blok paling atas ke paling bawah, pada kolom terakhir kunci ronde sebelumnya.

b. SubWord adalah pergantian nilai yang terdapat didalam blok kolom array menggunakan tabel S-Box.
c. Rcon adalah array konstant untuk proses perhitungan menggunakan tabel berikut :

Tabel 2.8.1 Tabel Rcon

KR2	KR4	KR6	KR8	KR10	KR12	KR14
01	02	04	08	10	20	40
00	00	00	00	00	00	00
00	00	00	00	00	00	00
00	00	00	00	00	00	00

2.9 Proses Enkripsi AES-256

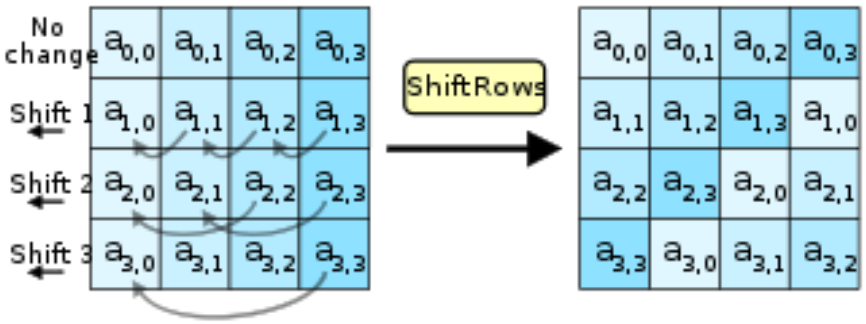
Pada peng-enkripsian AES-256 ini memiliki panjang blok 128 bit dengan panjang kunci 256 bit dan round sebanyak 14. Karena algoritma AES menggunakan tahapan berulang ulang pada prosesnya, tergantung panjang kunci yang dipilih yang disebut *round*. Proses enkripsi AES-256 ialah :

a. Pada tahap awal proses enkripsi, teks asli diubah menjadi sebuah state dengan operasi XOR. Lalu sebelum round ke-1 dimulai, blok teks asli digabungkan dengan kunci round ke-0, pada transformasi inilah disebut dengan *AddRoundKey*. Transformasi AddRoundKey pada proses enkripsi pertama kali pada round=0 dan selanjutnya round = round + 1.
b. Putaran sebanyak Nr-1 kali. Proses yang dilakukan pada setiap putaran adalah :
    1) SubBytes : Byte diganti dengan byte lain mengikuti kotak substitusi tetap atau sering disebut sebagai transformasi subtitusi. Kotak substitusi ini ditentukan dalam standar AES. Berikut ini merupakan tabel subtitusi S-Box.

Tabel 2.9.1 Tabel S-Box

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

2) ShiftRows : Pada tahap ini disebut dengan tahap transformasi permutasi dengan baris di blok digeser ke kiri mengikuti aturan tetap. Baris pertama tetap di tempatnya, baris kedua digeser satu, baris ketiga digeser dua dan seterusnya.



Gambar 2.9.2 Tahapan ShiftRows pada proses enkripsi

3) MixColumns : Pada tahap ini disebut dengan tahap tranformasi pengacakan, dalam tahap ini kolom diacak mengikuti aturan fixe yang ditentukan dalam standar. Operasi campuran ini setara dengan perkalian matriks dengan matriks tetap pada operasi ini terjadi perkalian polynomial didalamnya.

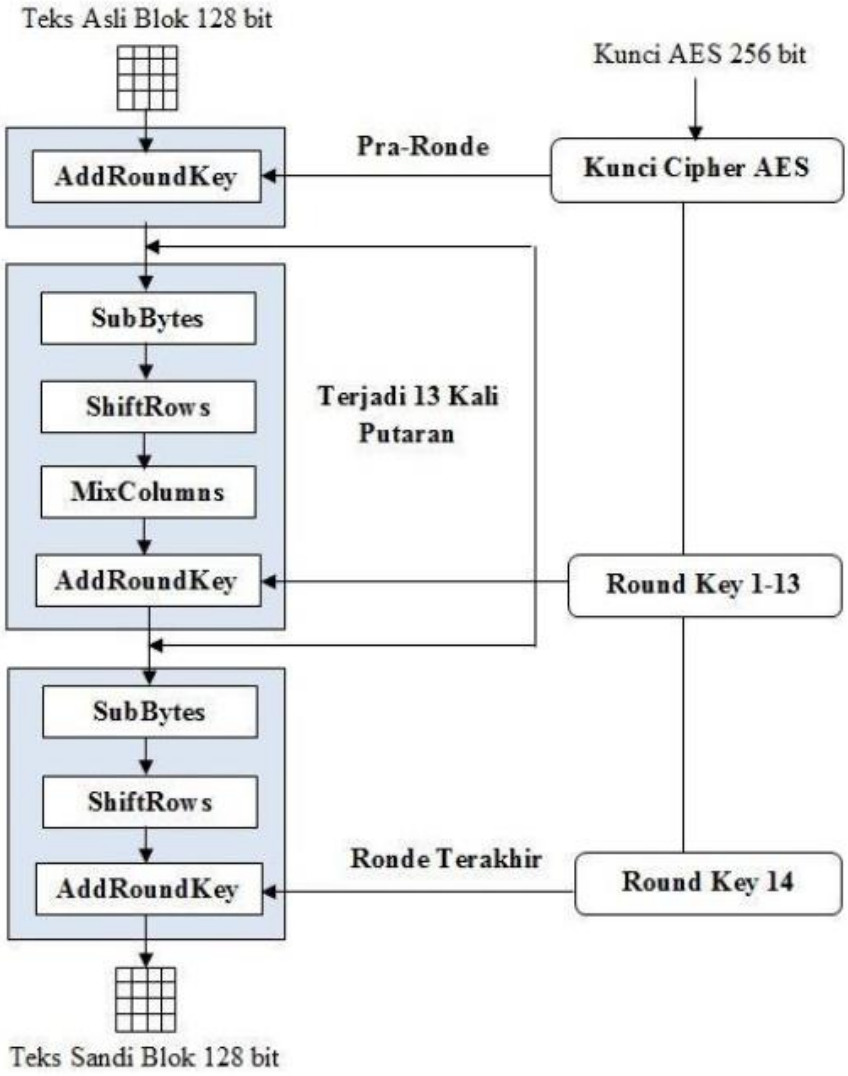
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Gambar 2.9.3 Tahapan MixColumns pada proses enkripsi

4) AddRoundKey : Pada tahap ini disebut dengan tahap transformasi penambahan kunci. melakukan XOR antara *state* sekarang dengan *round key*.

c. Pada Final Round, yaitu round ke-Nr dilakukan transformasi sama seperti *round* yang lainnya, akan tetapi tanpa tahap *MixColumns* yaitu hanya *SubBytes*, *ShiftRows* dan *AddRoundKeys*.

Alur tahapan proses enkripsi AES-256 dapat digambarkan seperti gambar dibawah ini :



Gambar 2.9 Alur Proses Enkripsi AES -256

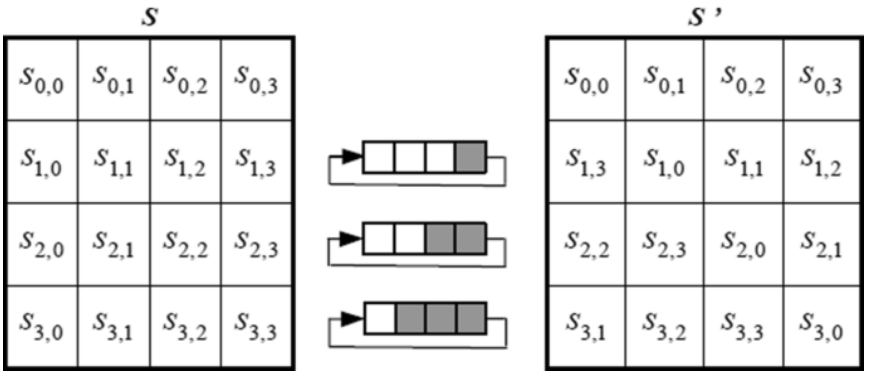
2.10 Proses Deskripsi AES-256

Proses deskripsi yaitu kebalikan dari proses enkripsi. Teks yang sudah di enkripsi atau teks bersandi (*chipper text*) dapat diubah kembali menjadi teks yang bisa terbaca (*plain text*) dan diterapkan ke arah yang berlawan yang disebut *inverse cipher*. Berikut proses deskripsi algoritma AES-256 :

a. Proses pada putaran pertama :

1) *AddRoundKey* : Transformasi *AddRoundKey* pada proses deskripsi diperlukan penambahan *roundkey* pada state dengan operasi XOR. Pada tahap *AddRoundKey* pertama kali pada round=14 dan selanjutnya round = round + 1.

2) *Inverse ShiftRows* : Pada tahap ini baris di blok digeser ke kanan mengikuti aturan tetap. Berikut gambaran transformasi *Inverse ShiftRows*.



Gambar 2.10.2 Tahapan Inverse ShiftRows pada proses deskripsi

3) Inverse Subbytes : pada proses kebalikan dari proses subtitusi bytes pada enkripsi. Tiap elemen pada state di konversi ke dalam tabel Inverse S-box. Berikut ini tabel Inverse S-box.

Tabel 2.10.3 Tabel Inverse S-Box

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

b. Lalu pada proses deskripsi round sebanyak 13 kali. Tahapan yang dilakukan pada proses round ialah :

1. AddRoundKey : diperlukan penambahan *roundkey* pada state dengan operasi XOR.
2. *Inverse MixColumns* : mengalikan setiap kolom *array state* dengan matriks yang sudah ditetapkan. Berikut gambaran *Inverse MixColums*.

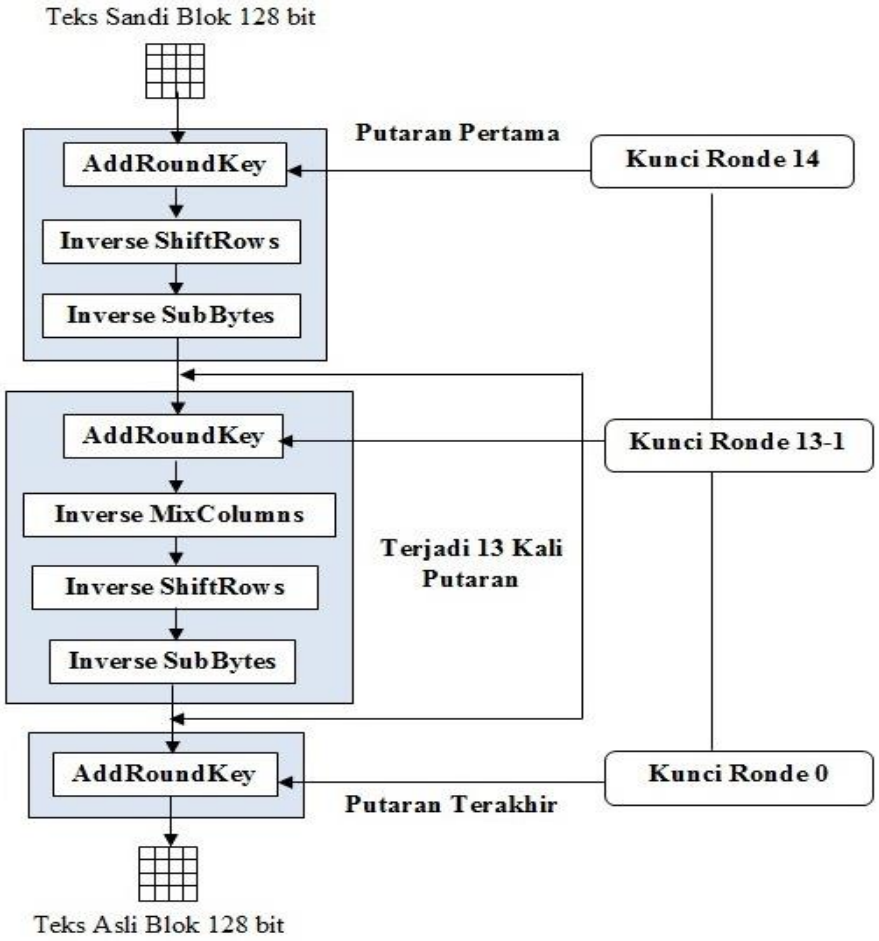
$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar 2.10.4 Tahapan Inverse MixColumns pada proses deskripsi

3. Inverse ShiftRows : Pergeseran basris-baris array *state* ke kanan.
4. Inverse SubBytes : Proses kebalikan dari proses subtitusi *bytes* pada enkripsi. Tiap elemen pada *state* di konversi ke dalam tabel *Inverse S-box*.

c. *Final round* : proses untuk *round* terakhir yaitu *AddRoundKey*.

Alur tahapan proses enkripsi AES-256 dapat digambarkan seperti gambar dibawah ini :



Gambar 2.10 Alur Proses Deskripsi AES-256

2.11 *Secure Hash Algorithm* (SHA-256)

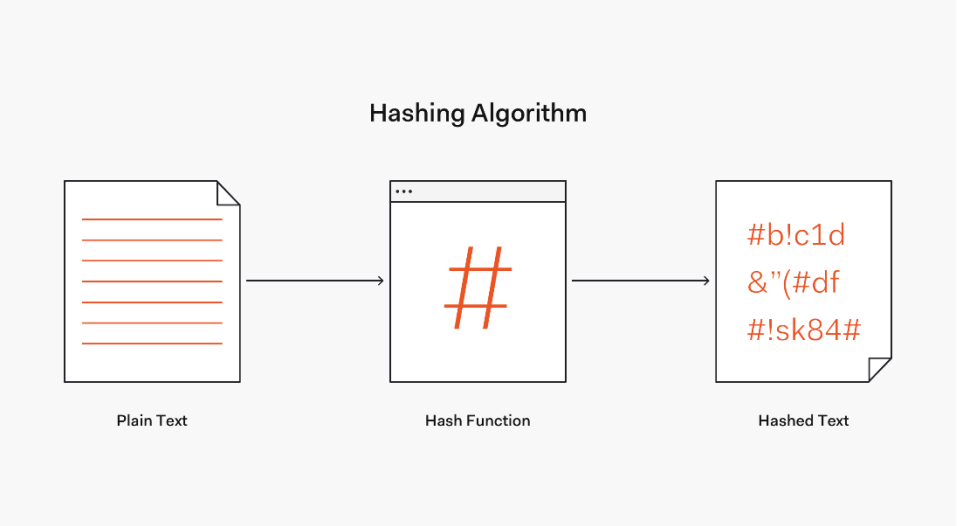
Dalam enkripsi data dikenal suatu fungsi yang di sebut fungsi hash atau *hashing*. Fungsi hash adalah adalah fungsi yang menerima masukan string yang panjangnya sembarang dan dikonversikan menjadi string dengan keluaran yang panjangnya tetap (Santi Sulastri, 2018). Fungsi hash yang berbeda akan menghasilkan output-output yang berbeda ukuran, tetapi kemungkinan ukuran output dari masing-masing algoritma hashing selalu konstan. Sebagai contoh, algoritma SHA-256 hanya akan menghasilkan *message digest* 256 bit, sedangkan SHA-1 selalu akan menghasilkan *digest* 160-bit. Fungsi hash satu arah atau yang sering disebut dengan *one way hash function* dimana hasil hash atau *hash value* sangat sukar dikembalikan ke nilai hash awal. Persamaan fungsi hash dapat dinyatakan sebagai berikut

$$h = H(M)$$

Dimana H sebagai fungsi hash dengan masukan pesan berupa *string* yang disimbolkan sebagai variable M dan menghasilkan suatu nilai *string* pula yang disimbolkan berupa variable h. Fungsi hash mempunyai beberapa nama lain, yaitu bisa disebut pula sebagai fungsi kompresi atau kontraksi, *fingerprint*, *message integrity check* (MIC), *manipulation detection code* (MDC) dan *cryptographic checksum.* Ada beberapa fungsi hash satu arah atau *hash function one way* antara lain MD2, MD4 dan MD5 (*Message Digest*) dan fungsi hash *Secure Hash Algorithm* atau yang sering disebut SHA.

Keamanan SHA-256 pernah diuji sebelumya yang di lakukan oleh peneliti jurnalnya yang berjudul Implementasi KeyedHash Message Authentication Code pada Sistem Keamanan Rumah (Ichwan, 2016). Keamanan SHA-256 pernah diuji pada penelitian yag dilakukan oleh peneliti [8]. Penggunaan SHA-256 yang digabungkan dengan algoritma Message Authentication Code (MAC) dari hasil pengujian 64 round menghasilkan nilai rata-rata avalanche effect (AE) sebesar 85,9%. Ini menunjukan bahwa keluaran SHA-256 memiliki tingkat pengacakan yang bagus. SHA-256 dirancang oleh *The National Institute of Standards and Technology* (NIST) pada tahun 2002.Fungsi hash SHA-256 merupakan fungsi SHA dengan ukuran digest 256 bit pada versi SHA-2. Ada beberapa versi SHA yaitu SHA-0, SHA-1, SHA-2, dan SHA-3. SHA-256 menggunakan enam fungsi logika yang merupakan kombinasi dasar antara lain seperti XOR, OR, AND dan pergeseran bit kekanan (shift right), dan rotasi bit kekanan (rotate right).

SHA-256 mengubah pesan masukan ke dalam message digest 256 bit. Berdasarkan Secure Hash Signature Standar. Pesan masukan yang panjangnya lebih pendek dari 264 bit, harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah message digest 256 bit. Lebih mudahnya plaintext diproses dengan fungsi hash lalu keluaran tersebut menjadi hash text yang tidak dapat terbaca. Berikut ini gambaran dari algoritma hash, yaitu :



Gambar 2.11 Alur Algoritma *Hashing*

2.12 Proses SHA-256

Pada SHA-256 *plaintext* diubah menjadi masukan ke dalam *message digest* 256 bit berdasarkan ketetapan *Secure Hashing Standard*. *Plaintext* yang panjangnya kurang dari 2^{64} bit harus dioperasikan kedalam *blocksize* 512 bits sehingga menjadi sebuah *message digest* sebesar 256 bits. Berikut ini penjelasan dari tahapan proses dari SHA-256, yaitu :

a. *Padding Bits* : Pada tahap ini fungsi hashing dimulai dari penambahan bit ke pesan asli (*plain text*), sehingga panjangnya akan sama dengan panjang standar yang diperlukan oleh fungsi hash. Jumlah bit yang ditambahkan pada pesan asli dihitung sedemikian rupa sehingga setelah penambahan bit, panjang pesan asli harus kurang dari 64 bit dari kelipatan 512 bits. Bit yang ditambahkan ke pesan, harus dimulai dengan '1' dan bit yang ditambahkan berikutnya harus '0' sampai tepat pada 64 bit dan kurang dari kelipatan 512.
b. *Length Bits* : Lalu pada tahap kedua penambahan bit sebelumnya yang setara 64 bits kedalam pesan keseluruhan untuk membuat semuanya menjadi kelipatan 512.
c. *Inisialisasi buffer* : Pada tahap ini permulaan melakukan perhitungan yaitu masing masing blok 512 bit tadi dipecah menjadi 16 buah bit word 32 bit yang mana nantinya diperluas menjadi 64 word yang diberi label

dengan aturan yang di tetapkan oleh *Secure Hashing Standard*. Lalu masing masing label tadi kemudian di proses dengan fungsi hash SHA-256. Didalam inti prosesnya, algoritma SHA-256 membuat 8 variabel yang diberikan nilai awal a=h0 (0) – h=h7(0) diawal masing masing fungsi hash. Nilai nilai awal tersebut di tunjukan sebagai berikut :

```
a = 0x6a09e667
b = 0xbb67ae85
c = 0x3c6ef372
d = 0xa54ff53a
e = 0x510e527f
f = 0x9b05688c
g = 0x1f83d9ab
h = 0x5be0cd19
```

d. Fungsi Kompresi : Dan pada tahap ini, SHA melakukan perhitungan sebanyak 64 kali putaran untuk setiap bloknya dan keluaran yang diperoleh diumpankan sebagai masukan untuk putaran operasi berikutnya. Dan delapan varibel tadi nilainya akan terus berganti selama perputaran sebanyak 64 kali.
e. *Output* : lalu pada setiap putaran berfungsi sebagai input untuk putaran berikutnya dan proses ini terus berlanjut hingga bit terakhir dari pesan tetap ada dan hasil putaran terakhir untuk n last bagian blok pesan akan memberi kita hasil yaitu hash untuk keseluruhan pesan. Panjang keluarannya adalah 256 bit.

2.13 *Unified Modelling Language* (UML)

Unified Modelling Language atau kepanjangan dari UML adalah sebuah bahasa pemodelan yang telah menjadi standar yang dirancang khusus untuk pengembangan, analisis sistem berorientasi objek dan desain dari sebuah perangkat lunak (Hadi Fajar, 2017). UML pertama kali dikembangkan oleh Grady Booch, Jim Rumbaugh, dan Ivars Jacobson pada pertengahan tahun 1994.

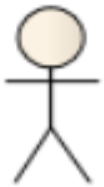
Dalam penelitian ini penulis menggunakan diagram yang difenisikan dalam UML, yang telah menjadi bahasa pemodelan standard untuk pemodelan berorientasi objek. UML terdiri dari beberapa diagram, dalam laporan ini penulis menggunakan *Usecase Diagram* untuk menggambarkan fungsionalitas

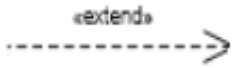
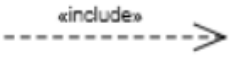
dari aplikasi yang dibuat oleh penulis dan *Activity Diagram* untuk menggambarkan urutan aktivitas di dalam aplikasi dari proses kerja yang ada serta *Sequence Diagram* yang menggambarkan alur interaksi objek dalam urutan atau rangkaian waku di dalam sebuah sistem.

2.14 Use Case

Use case diagram merupakan bentuk dari pemodelan untuk menunjukan sebuah fungsionalitas dalam suatu sistem yang dibuat untuk mencapai tujuan tertentu. *Use case* diagram adalah sebuah diagram yang menunjukan hubungan antara *actors* dan *use cases* yang di gunakan untuk menganalisa dan desain sebuah sistem (The Elements of UML 2.0: Scott W.Ambler, 2005:33). *Use case* dapat memberikan gambaran umum tentang seluruh atau sebagian kebutuhan penggunaan untuk suatu sistem dan mengkomunikasikannya dalam ruang lingkup. Berikut simbol simbol yang digunakan dalam Use Case Diagram yaitu ;

Tabel 2.14 Tabel Use Case Diagram

	*ACTOR* Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari actor adalah gambar orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor.
	*USE CASE* Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesar antar unit atau actor biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case.
	ASOSIASI/ASSOCIATION

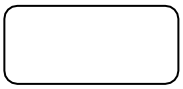
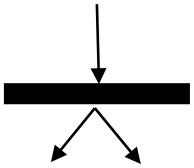
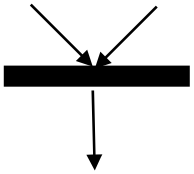
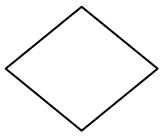
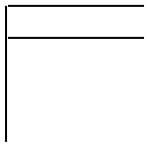
	Komunikasi antara actor dan use case yang berpartisipasi pada use case atau use case memiliki interraksi dengan actor.
	*EKSTENSI/EXTEND* Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan memiliki nama depan yang sama dengan use case yang di tambahkan
	*GENERALISASI/GENERALIZATION* Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
	*MENGGUNAKAN/INCLUDE* Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsional atau sebagai syarat dijalankan use case ini

2.15 Activity Diagram

Activity Diagram adalah suatu diagram yang menggambarkan konsep data atau *workflow* (aliran kerja), aksi yang terstruktur serta dirancang dengan baik dalam suatu sistem (Hendini, 2016). Simbol-simbol yang digunakan dalam *Activity Diagram* yaitu ;

Tabel 2.15 Tabel Activity Diagram

	*Start Point,* diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, merupakan akhir aktivitas

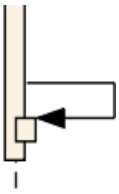
	*Activities*, menggambarkan suatu proses / kegiatan bisnis
	*Fork* / percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu
	*Join* / penggabungan atau rake, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points*, menggambar pengambilan keputusan *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

2.16 Sequence Diagram

Sequence Diagram adalah sebuah diagram yang menggambarkan kolaborasi objek pada *use case* yang saling berinteraksi antar elemen dari suatu kelas. Kegunaan dari *Sequence Diagram* sendiri ialah untuk menunjukan rangkaian pesan yang dikirim antara objek dengan objek. Simbol simbol yang digunakan dalam Sequence Diagram yaitu ;

Gambar 2.16 Tabel Sequence Diagram

	Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari actor adalah gambar orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor

	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation
	*Message*, symbol mengirim pesan antar class
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*,mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi

2.17 Tinjauan Pustaka

Berikut penjabaran penelitian terdahulu pada tabel 2.17 dibawah ini.

Nama	Judul	Metode	Hasil Penelitian
Sandi Yusmantoro, Edy Hermansyah, Rusdi Efendi (2014)	Rancang Bangun Aplikasi Pengaman Keaslian Surat Ijin Tempat Usaha Menggunakan Algoritma Elgamal dan Secure HashAlgorithm 256 (Studi Kasus : Badan Pelayanan Perizinan Terpadu (BPPT) Kota Bengkulu).	Algoritma Elgamal, Secure Hash Algorithm 256	Membangun aplikasi pengamanan keasilan surat ijin tempat usaha di Badan Pelayanan Perizinan Terpada Kota Bengkulu
Muammar Renaldy (2015)	Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode AES – 128 (Advanced Encryption Standard – 128) dan SHA – 1 (Secure Hashing Algorithm – 1).	Algoritma AES-128, SHA-1	Membangun aplikasi berbasis mobile android untuk pengenkripsian pesan teks.

Hadi Fajar Wiguno (2017)	Aplikasi Pengamanan File Dan Pesan Teks Menggunakan Algoritma AES 256 Dan SHA 256 Berbasis Android.	Algoritma AES-256, SHA-256	Membangun aplikasi berbasis android untuk pengenkripsian file dan pesan teks.
Ratno Prasetyo dan Asep Suryana (2016)	Aplikasi Pengamanan Data dengan Teknik Algoritma Kriptografi AES dan Fungsi Hash SHA-1 Berbasis Desktop.	Algoritma AES, SHA-1	Membangun apikasi berbasis Desktop dengan AES dan SHA-1
Santi Sulastri1 dan Riana Defi Mahadji Putri (2018)	Implementasi Enkripsi Data Secure Hash Algorithm (SHA-256) dan Message Digest Algorithm (MD5) pada Proses Pengamanan Kata Sandi Sistem Penjadwalan Karyawan.	SHA-256, MD5	Pengimplementasian enkripsi data dengan fungsi hash untuk pengamanan kata sandi sistem penjadwalan karyawan.
Pandi Barita Nauli Simangunsong Dan Komariah Fitri (2018)	Perancangan Aplikasi Pengamanan Citra Bewarna Dengan Algoritma RSA	RSA	Pengenkripsian Citra Menggunakan Algoritma RSA

BAB III

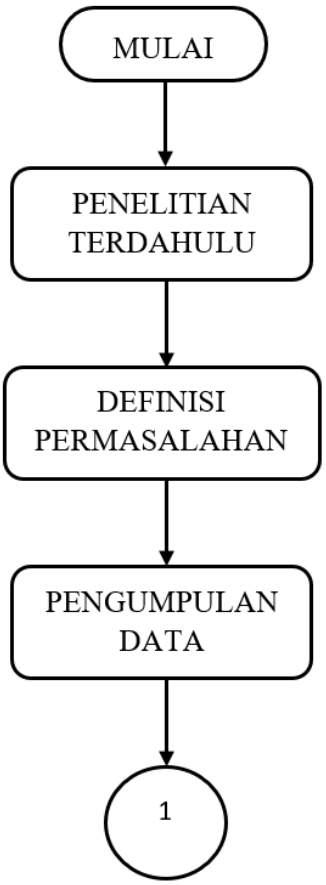
METODOLOGI PENELITIAN

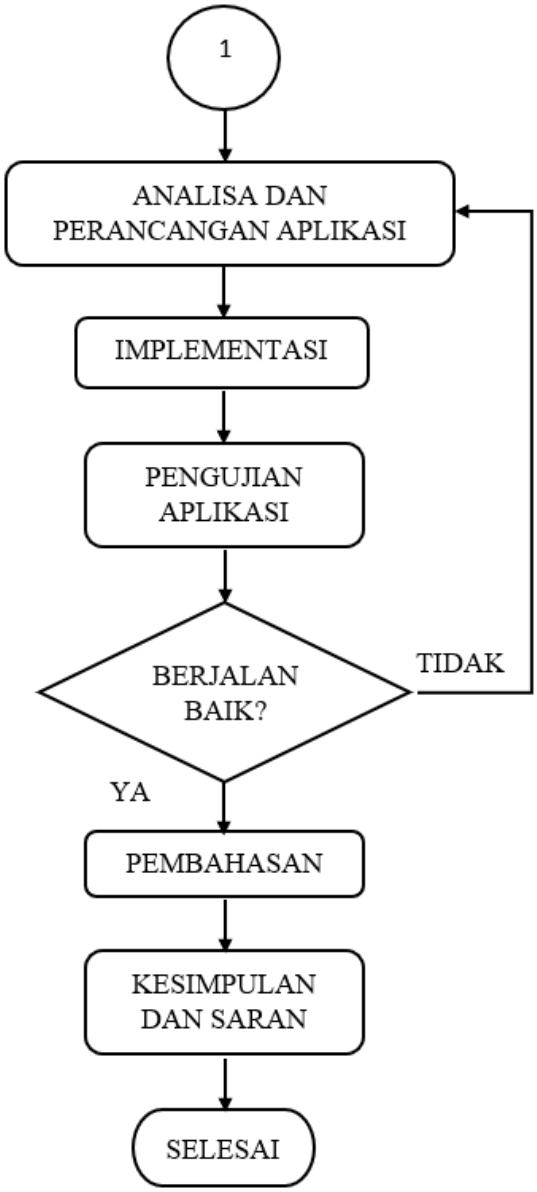
3.1 Objek Penelitian

Pengumpulan informasi dan data sekunder dilakukan pada Fakultas Ilmu Komputer Program Studi Informatika Universitas Bhayangkara Jakarta Raya, Jl. Raya Perjuangan Bekasi Utara, Kota Bekasi, Jawa Barat 17121, Indonesia. Pada divisi administrasi untuk pengamanan data file.

3.2 Kerangka Penelitian

Diagram alur ini menunjukan gambaran kerangka penelitian yang dilakukan penulis tahapan demi tahapan sebagai pedoman dalam penelitian agar yang ingin dicapai tidak menyimpang dari tujuan sehingga mendapatkan hasil yang relevan. Serta menjadi solusi untuk pemecahan masalah di lembaga pendidikan Universitas Bhayangkara Jakarta Raya. Tahap tahap yang akan dilalui dalam metodologi penelitian dapat dilihat pada gambar dibawah ini ;





Gambar 3.2. Kerangka Penelitian

Untuk memahami kerangka penelitian pada gambar diatas, berikut penjelasan secara terperinci mengenai urutan tahapan kerangka penelitian diatas.

1. Penelitian Pendahuluan

Penelitian pendahuluan merupakan tahap awal yang dimaksudkan untuk mencari tahu permasalah yang terjadi di objek penelitian. Mengerti masalah yang ada dan menghubungkannya dengan algoritma untuk menjadi bahan penelitian guna memecahkan masalah yang terjadi. Pada penelitian

pendahuluan ini, proses wawancara juga dilakukan oleh pihak pihak terikait untuk keberlangsungannya proses penelitian ini.

2. Definisi Permasalahan

Setelah melakukan tahapan penelitian pendahuluan, selanjutnya pada tahapan ini dilakukan pendefinisian permasalahan. Pendefenisian permasalahan mencakup identifikasi masalah yang ada di Universitas Bhayangkara serta permasalahan yang ingin diangkat sebagai bahan penelitian serta batasan batasan apa yang dinyatakan dalam penelitian. Selain itu tentunya tujuan dan manfaat penelitian juga didefinisikan, baik untuk Universitas Bhayangkara atau untuk peneliti. File sendiri dapat bersifat pribadi dan sensitive sesuai dengan persepsi setiap individu. File juga memiliki ragam jenis ekstensi yaitu file dokumen, file data keuangan, file gambar dan jenis file lainnya. Pentingnya menjaga suatu file penting bagi suatu lembaga agar tersimpan dengan baik dan terjaga kerahasiannya. Berdasarkan kebutuhan lembaga tersebut untuk menjaga keamanan datanya, maka dari itu penelitian ini menerapkan algoritma kriptografi kedalam pembuatan aplikasi enkripsi dan dekripsi berbasis desktop untuk Universitas Bhayangkara agar dapat mengamankan data mereka dengan baik.

3. Pengumpulan Data

Tahapan ini adalah mengumpulkan data data yang diperlukan guna menjadi bahan penelitian untuk kemudian dijadikan input data pada aplikasi yang dibuat oleh penulis dalama penelitian ini. Pengumpulan data yang dimaksud ialah mencari tahu format atau ekstensi jenis file apa yang dikiranya penting bagi Universitas Bhayangkara untuk datanya diamankan. Pengumpulan data secara langsung ialah dengan mewawancarai pihak terkait dengan menanyakan jenis data apa yang biasanya bersifat rahasia untuk dijaga keamanannya, yaitu apakah data tentang keuangan yang berformat excel atau ekstensi file data rahasia lainnya. Ada 3 teknik pengumpulan data, yaitu :

a. Pengamatan Langsung (Observasi)
b. Data Sekunder

c. Studi Pustaka

4. Analisa dan Perancangan Aplikasi

Setelah melalui tahapan sebelumnya dan merunjuk kepada inti dari permasalahan, maka di buatlah aplikasi yang diharapkan dapat membantu Universitas Bhayangkara untuk memecahkan permasalahan yang ada yaitu pengamanan data. Dimulai dari analisa sistem yang meliputi tahapan sistem yang akan dibuat sebagai konsep, objek dan keterkaitannya serta analisa solusi dari algoritma dan kebutuhan aplikasi. Analisa ini ditranlasikan kedalam bentuk pemodelan UML yaitu use case diagram, activity diagram serta sequence diagram sebagai bentuk dari peracangan sebuah aplikasi yang akan dibuat.

5. Implementasi

Pada tahap implementasi ini dilakukan pembuatan modul modul yang telah dirancang dalam tahap perancangan kedalam bahasa pemrograman tertentu. Tahap implementasi ini dibutuhkan suatu alat dan bahan yang digunakan sebagai perangkat pendukung demi kelancaran tahap implementasi program. Karena sebelum program diimplementasikan, maka program harus bebas dari kesalahan. Kesalahan yang dimaksud ialah kesalahan program yang mungkin terjadi yaitu kesalahan penulisan (coding), kesalahan proses atau kesalahan logika. Dalam hal ini penulis mengimplementasikan bahasa pemrograman C# (c sharp) untuk pembuatan aplikasi enkripsi dan dekripsi berbasis desktop. Berikut perangkat yang digunakan terdiri dari :

a. Perangkat Keras (*hardware*)

   Perangkat keras yang digunakan penulis untuk membuat aplikasi ini antara lain adalah dengan sebuah laptop dengan spesifikasi mempunyai *processor* Intel Core i7-10510U dengan RAM 8 GB dan *Solid State Drive* 1000 GB serta VGA Nvidia MX250 2GB

b. Perangkat Lunak (*software*)

Perangkat lunak yang digunakan dalam membangun aplikasi ini adalah system operasi windows 10 64 bit, Visual Studio 2019 (.NET Core).

6. Pengujian Aplikasi

Tahap selanjutnya adalah testing atau pengujian aplikasi. Setelah aplikasi selesai dirancang, maka dilakukan pengujian untuk melihat apakah aplikasi yang telah dirancang dapat berjalan dengan baik den memberikan hasil keluaran yang baik. Jika aplikasi masih memiliki error atau bug dalam pelaksanaanya maka akan dievaluasi ulang kembali. Pengujian aplikasi meliputi empat parameter yaitu keamanan, integritas data, kecepatan proses dan perubahan kapasitas file. Pengujian ini dilakukan dengan langkah sebagai berikut :

a. *Blackbox* adalah sebuah metode yang digunakan untuk menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah *input* diterima dengan benar dan *output* yang dihasilkan telah sesuai dengan yang diharapkan.
b. *Bruteforce Attack* adalah serangan untuk mengungkap kunci atau password dengan mencoba semua kemungkinan kunci sampai akhirnya menemukan kunci yang tepat.
c. Mencari kekurangan dari system aplikasi yang telah dibangun, hal ini dilakukan setelah aplikasi berhasil dijalankan dan menemukan kelamahan secara konsep yang akan dimasukan dalam kesimpulan dan saran.

7. Pembahasan

Pada tahapan ini pembahasan dan analisa dilakukan adalah mencakup hasil keluaran dari penerapan algoritma. Pada tahapan ini pembahasan pada penerapan algoritma *Advanced Encryption Standard* (AES-256) dan *Secure Hash Algorithm* (SHA-256) dipengaplikasiannya akan dijelaskan pada tahap ini.

8. Kesimpulan dan Saran

Berdasarkan hasil tahapan sebelumnya sampai pada tahapan pembahasan dan analisa, maka dapat ditarik kesimpulan dari hasil pembahasan serta hasil pengujian untuk menjawab semua pertanyaan yang ada pada rumusan masalah pada penelitian ini. Pada tahapan ini sejumlah saran juga akan diberikan untuk mengatasi permasalahan yang ditemukan, sehingga dapat menjadi masukan dan manfaat bagi Universitas Bhayangkara untuk kedepannya.

3.3 Metode Penelitian

Metodologi yang digunakan dalam penulisan penelitian ini adalah dengan metode pengumpulan data dimana untuk mendapatkan data dan bahan penelitian yang sesuai harapan, teknik pengumpulan data yang digunakan ada tiga jenis diantaranya sebagai berikut :

1. Pengamatan Langsung (Observasi)

   Yaitu suatu cara penelitian atau metode pengumpulan data dengan jalan mengadakan pengamatan secara langsung pada objek penelitian yang merupakan sumber data. Pada kesempatan kali ini penulis mengamati objek penelitian yang bertempat di Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya.

2. Data Primer

   Data ini diperoleh melalui wawancara (interview) dengan pihak yang terkait di Fakultas Ilmu Komputer Universitas Bhayangkara dan menanyakan bagaimana mereka menyimpan suatu data yang bersifat rahasia, apakah ada penelitian sebelumnya yang bertemakan algoritma kriptografi untuk pengamanan data dan sebagainya.

3. Studi Literatur

   Studi Literatur ini merupakan cara untuk mendapatkan data-data secara teoritis sebagai bahan penunjang dalam penyusunan laporan penelitian dengan membaca buku literature dari perpustakaan serta dari internet untuk mendukung teori yang ada dan maupun dari buku buku referensi lainnya untuk melengkapi data-data yang ada. Studi literatur sendiri merupakan

metode untuk memeperoleh informasi yang berhubungan dengan latar belakang dan rumusan masalah, informasi yang dikumpulkan berupa konsep dan teori yang berkaitan dengan judul penilitian penulis yaitu algoritma kriptografi Advanced Encryption Standard (AES) dengan mode CBC dan Secure Hash Algorithm (SHA) khususnya yang berhubungan dengan teori enkripsi dan dekripsi suatu file. Langkah yang dilakukan penulis guna memperoleh informasi ialah dengan Studi Pustaka dan Diskusi. Studi Pustaka yaitu langkah untuk melakukan pencarian, menemukan dan mengumpulkan informasi yang sesuai dengan kasus, referensi yang didapat berupa buku, jurnal, artikel dan penelitian terdahulu tentang algoritma AES dengan mode CBC dan SHA. Serta diskusi dengan pakar ahli dibidang sekuriti dengan maksud mencari solusi dan masukan tentang permasalahan yang berhubungan dengan kesulitan yang ada didalam proses pembuatan aplikasi serta pemahaman implementasi kriptografi di dalam perangkat lunak.

3.4 Analisa Permasalahan

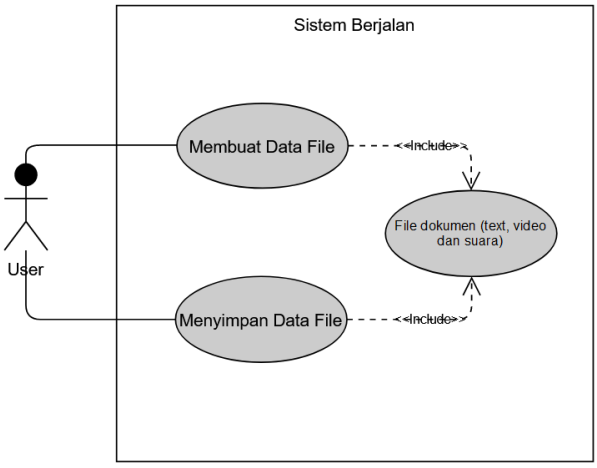
Masalah yang terjadi pada analisa sistem berjalan yang berkaitan dengan pengamanan data adalah sebagai berikut :

1. Berdasarkan hasil wawancara dengan pihak terkait diketahui bahwa masih minimnya pengetahuan tentang pentingnya menjaga keamanan sebuah data yang bersifat penting atau rahasia didalam sebuah komputer, seperti data keuangan dan data penting lainnya.
2. Sebuah data yang bersifat rahasia bisa saja dicuri, dimanipulasi dan dikorupsi apabila tidak dijaga keamanannya. Dan apabila terjadi sebuah komputer terjankit sebuah virus dan virus tersebut mengenkrip data penting tersebut maka resikonya ialah kehilangan data tersebut, olehkarena itu disamping dari pentingnya menjaga keamanan data, proses back-up data juga penting dilakukan.

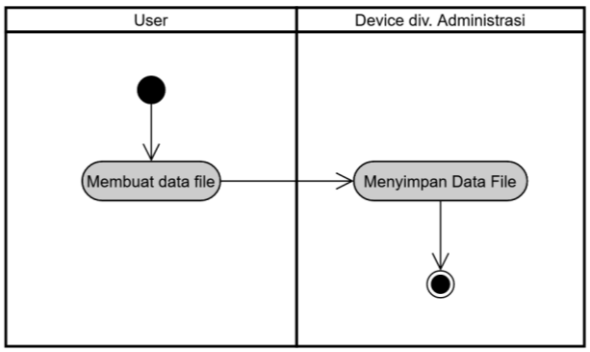
3.5 Analisa Sistem Yang Sedang Berjalan

Pada Fakultas Informatika Univeristas Bhayangkara pada sistem sebelumnya tidak ditemukannya pengamanan data file penting atau bersifat rahasia

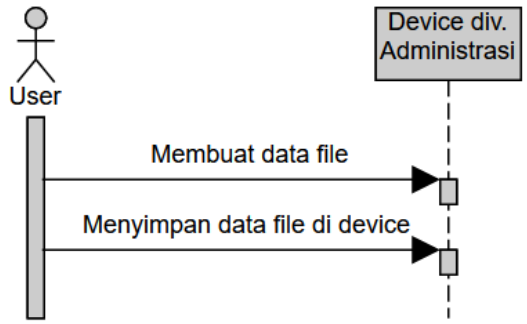
di divisi administrasi Fakultas Informatika Universitas Bhayangkara Jakarta Raya. Penyimpanan data file hanya bersifat standard pada suatu device, belum di enkripsi atau diamankan untuk menjaga kerahasiaan data tersebut. Berikut penulis jabarkan sistem berjalan yang ada pada divisi administasi Fakultas Bhayangkara Jakarta Raya untuk penyimpanan data file pada tiga diagram berikut yaitu Use Case, Activity Diagram dan Sequence Diagram.



Gambar 3.5.1 Use Case Sistem Berjalan



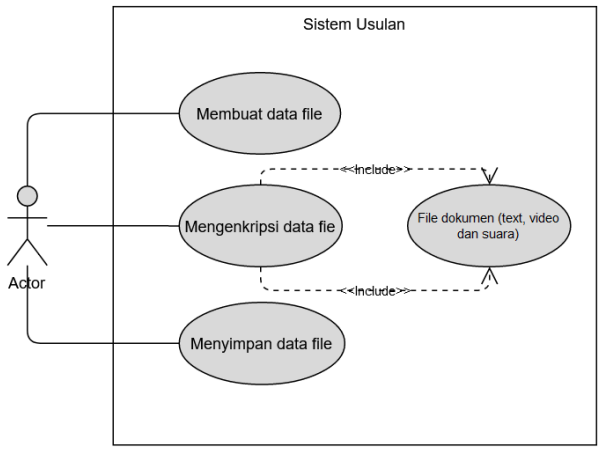
Gambar 3.5.2 Activity Diagram Sistem Berjalan



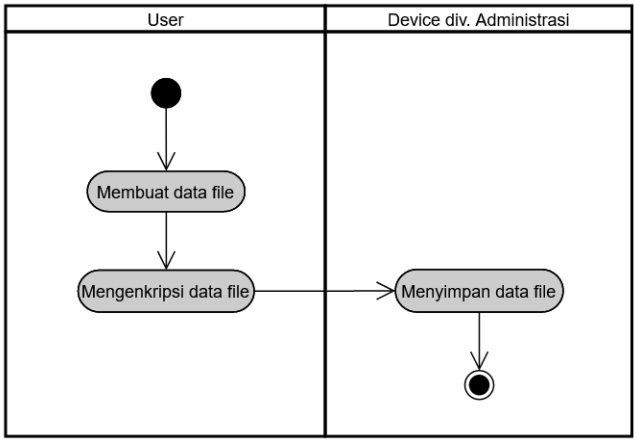
Gambar 3.5.3 Sequence Diagram Sistem Berjalan

3.6 Analisa Sistem Usulan

Analisa usulan sistem ini digunakan untuk mengetahui usulan dari penulis untuk memecahkan masalah yang dihadapi pada sistem yang sedang berjalan. Usulan sistem untuk penyimpanan data file pada dua diagram berikut yaitu Use Case dan Activity yang akan dibahas sebagai berikut :



Gambar 3.6.1 Use Case Sistem Usulan



Gambar 3.6.2 Activity Diagram Sistem Usulan

Sistem usulan yang penulis ajukan ialah data file penting yang ada di divisi administrasi Fakultas Ilmu Komputer harus dienkripsi terlebih dahulu agar data file tersebut tetap aman dalam device penyimapanan.

3.7 Analisa Kebutuhan Sistem

Sebelum melakukan perancangan sebuah aplikasi, penulis melakukan analisa terhadap kebutuhan sistem yang bertujuan untuk menyesuaikan kebutuhan

user dengan aplikasi yang dirancang oleh penulis, sehingga aplikasi yang di rancang oleh penulis dapat memenuhi tujuan penelitian ini. Analisa kebutuhan sistem juga dilakukan dengan cara melakukan observasi atau pengamatan langsung kepada pihak terkait. Dalam perancangan aplikasi ini ada beberapa hal yang diperlukan dalam proses perancangannya adapun hal tersebut ialah :

1. Algoritma *Advanced Encryption Standard* (AES-256) dengan mode CBC

   Salah satu mode operasi AES yaitu CBC atau yang disebut *Chiper Block Chaining*. Pada algoritma blok chipper seperti AES ini, *plaintext* atau pesan mentah yang masuk untuk diproses dengan panjang yang tetap yaitu n, akan tetapi jika ukuran datanya terlalu panjang maka dilakukan pemecahan data data tersebut menjadi blok blok dengan ukuran yang lebih kecil. Pada CBC, rangkaian bit-bit pada plaintext dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (*initialization vector*) untuk menggabungkan dengan plaintext pertama dan chipertext block sebelumnya menjadi IV di block selanjutnya. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini.
2. Algoritma *Secure Hash Algorithm* (SHA-256)

   Fungsi hash adalah adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan dikonversikan menjadi string dengan keluaran yang panjangnya tetap. Fungsi hash yang berbeda akan menghasilkan output-output yang berbeda juga, tetapi kemungkinan ukuran output dari masing-masing algoritma hashing selalu konstan. algoritma SHA-256 hanya akan menghasilkan *message digest* 256 bit. Penggunaan algoritma SHA digunakan untuk proses pembuatan kunci cipher dengan cara menghitung nilai *message digest*. Algoritma SHA-256 tergolong dari algoritma fungsi hash sebelumnya yaitu SHA-2. Ada beberapa series algoritma SHA yaitu SHA-0, SHA-1, SHA-2 dan terbarukan yaitu SHA-3.
3. Data Objek

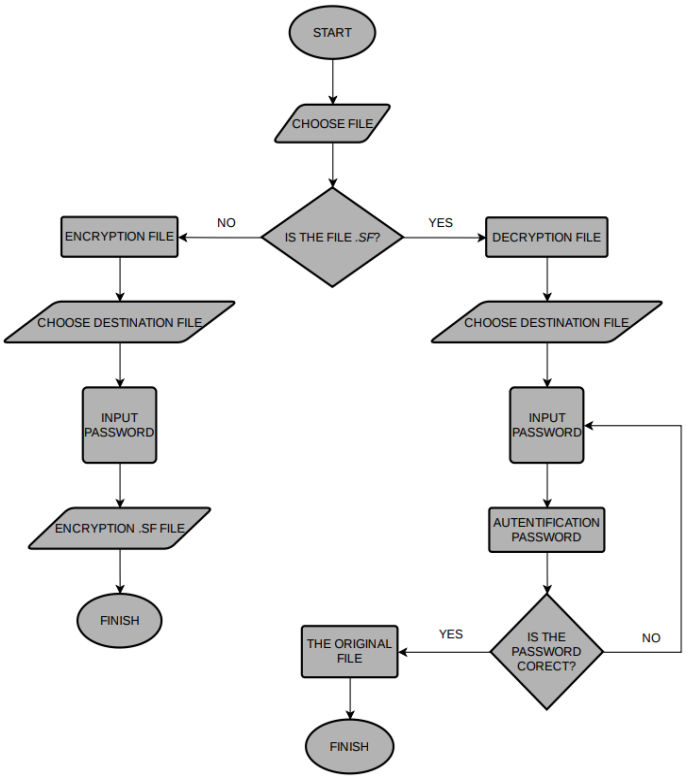
   Data objek yang digunakan sebagai semple data untuk di enkripsi dan dekripsi ialah contoh data yang diambil dari fakultas ilmu komputer

Universitas Bhayangkara Jakarta Raya. Date tersebut berjenis file dokumen, file gambar, file suara dan file video untuk pemrosesan enkripsi dan dekripsi pada aplikasi *Secret Fichier*.

Setelah ditentukan algoritma yang digunakan dalam penelitian ini, yaitu algoritma *Advanced Encryption Standard* (AES) dengan panjang kunci 256 bit untuk pemrosesan enkripsi dan dekripsi file serta *Secure Hash Algortihm* (SHA-256) sebagai algoritma fungsi hashnya. Maka dari itu penelitian ini akan membuat sebuah aplikasi pengamanan data berbasis desktop yang dapat melakukan hal hal berikut ini :

1. Aplikasi enkripsi dan dekripsi *Secret Fichier* dapat berjalan pada sistem operasi *windows*.
2. Aplikasi *Secret Fichier* dapat mengenkripsi jenis file tunggal bukan folder yaitu dokumen (berformat docx, xlxs), gambar (berformat jpg, png) suara dan video (berformat mp4).
3. Aplikasi *Secret Fichier* dapat mengdekripsi file yang sudah di enkripsi sebelumnya pada aplikasi tersebut dengan kunci atau password yang sama dan mengembalikan file tersebut seperti semula.
4. Aplikasi *Secret Fichier* dapat mengamankan data bersifat penting dan mengubahnya menjadi file yang yang tidak terbaca (chipper text) dengan ekstensi file *.sf*.

3.7.1 Flowchart Aplikasi *Secret Ficher*



Gambar 3.6.1 Flowchart Enkripsi dan Dekripsi File

Berikut gambar diatas ialah flowchart dari aplikasi Secret Fichier, yaitu aplikasi enkripsi dan dekripsi file yang menerapkan algoritma AES-256 dan SHA-256. Seperti gambar flowchart diatas, tahapan pertama pada aplikasi enkripsi dan dekripsi file Secret Fichier, yaitu user memilih atau menginput sebuah file (bukan folder) yang ingin di proses, apabila file tersebut berformat .sf maka file tersebut akan didekripsi. setelah user memilih file dan memilih lokasi file setelah di proses, selanjutnya user harus menginputkan kembali password yang user gunakan pada proses enkripsi, password itu akan di autentifikasi apabila password yang diinputkan salah maka proses dekripsi tidak terjadi atau gagal dan jika password yang diinputkan benar, maka file akan kembali seperti semula. apabila file tersebut berjenis file dokumen, suara, gambar dan video maka file tersebut dapat di proses untuk dienkripsi. Setelah user memilih sebuah file yang akan dienkripsi, lalu user diharuskan memilih destinasi lokasi file yang ingin ditaruh setelah proses enkripsi selesai, setelah itu user menginputkan password dan proses enkripsi terjadi. Apabila user

3.8 Alat Penelitian

Demi keberlangsungnya perancangan aplikasi enkripsi dan dekripsi *Secret Fichier* maka dari itu dibutuhkannya perangkat yang mendukung dan memumpuni untuk kemudahan perancangan baik dari segi perangkat keras (*hardware*) maupun perangkat lunak (*software*). Berikut spesifikasi perangkat keras dan perangkat lunak yang digunakan :

1. Perangkat Keras :
   Processor : Intel Core i7-10510U
   Memory RAM : 8 GB
   Solid State Drive : 1000 GB
   VGA : Nvidia MX250 2GB
2. Perangkat Lunak :
   Windows 10 Home 64 bit
   Visual Studo .NET CORE 3.0

3.9 Jadwal Penelitian

Jadwal kegiatan penulisan tugas akhir skripsi ini dirincikan sebagai berikut:

Kegiatan	September (Minggu)				Oktober (Minggu)				November (Minggu)				Desember (Minggu)				Januari (Minggu)				Februari (Minggu)			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Identifikasi Masalah																								
Studi Literatur																								
Penulisan Proposal																								
Analisa dan Perancangan																								
Implementasi																								
Hasil																								

BAB IV

PERANCANGAN SISTEM DAN IMPLEMENTASI

4.1 Penerapan *Secure Hash Algorithm* (SHA-256)

Pada penelitian pengamanan data file ini, penulis menerapkan algoritma SHA. Algoritma SHA yang digunakan ialah SHA-256, yang hanya menghasilkan message digest sebesar 256 bit. SHA-256 mengubah pesan masukan ke dalam message digest 256 bit, berdasarkan *Secure Hash Signature Standard*. Lebih mudahnya plaintext diproses dengan fungsi hash lalu keluaran tersebut menjadi hash text yang tidak dapat terbaca. Berikut ini implementasi proses SHA-256 dari sebuah pesan asli yang berisi "xyz" dengan penyelesaian tahap demi tahap proses dari SHA-256, sebagai berikut :

Sebuah pesan asli disimpan kedalam variable M, maka M = "xyz". Variable M dikonversikan kedalam bentuk binary, sehingga menjadi M = 0111 1000 0111 1001 0111 1010. Lalu dilakukan penambahan *padding* yaitu bit "1" pada variable M sehingga menjadi M = 0111 1000 0111 1001 0111 1010 1.

Walaupum SHA-256 menghasilkan 265 bit akan tetapi *Plaintext* yang panjangnya kurang dari 2^{64} bit harus dioperasikan kedalam *blocksize* 512 bits sehingga menjadi sebuah *message digest* sebesar 256 bits.. Agar *blocksize* mencapai 512 bit, maka selanjutnya tambahkan panjang pesan asli ke dalam bentuk biner. karena ukuran frase "xyz" dalam hal ini terdapat delapan bit per hurufnya dan menjadi 24 bit secara keseluruhan lalu dikonversikan ke binary ialah 24 = 0001 1000 dan menjadi binary dengan ukuran 8 bit. Agar penambahan pesan asli menjadi 512 bit, maka akan ditambahkan bit "0" sebanyak 479 buah (512-25-8 = 479) dan ditambahkan lagi hasil konversi tadi yaitu 00011000. Gambarannya seperti dibawah ini :

01111000011110010111101010000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000011000

Dari 512 bits diatas diubah menjadi variable yang memiliki 32 bit, sehingga 512/32 = 16 variable. Kemudian dilakukan parsing menjadi 16 buah variable dengan 32 bit untuk masing masing blok pesan M.

M0 = 01111000 01111001 01111010 10000000

M1 = 00000000 00000000 00000000 00000000

M2 = 00000000 00000000 00000000 00000000

M3 = 00000000 00000000 00000000 00000000

M4 = 00000000 00000000 00000000 00000000

M5 = 00000000 00000000 00000000 00000000

A6 = 00000000 00000000 00000000 00000000

M7 = 00000000 00000000 00000000 00000000

M8 = 00000000 00000000 00000000 00000000

M9 = 00000000 00000000 00000000 00000000

M10 = 00000000 00000000 00000000 00000000

M11 = 00000000 00000000 00000000 00000000

M12 = 00000000 00000000 00000000 00000000

M13 = 00000000 00000000 00000000 00000000

M14 = 00000000 00000000 00000000 00000000

M15 = 00000000 00000000 00000000 00011000

Tahapan selanjutnya ialah inisialisasi nilai hash awal dalam notasi heksadesimal, inisialisasi nilai hash awal pada SHA-256 bertujuan untuk menyimpan nilai hash

awal dan nilai keluarannya, oleh karena itu digunakan buffer H0, H1, H2, H3, H4, H5, H6, H7. Berikut inisialisasi nilai hash awal dalam notasi heksadesimal ;

H0 = 6a09e667

H1 = bb67ae85

H2 = 3c6ef372

H3 = a54ff53a

H4 = 510e527f

H5 = 9b05688c

H6 = 1f83d9ab

H7 = 5be0cd19

Selanjutnya ialah tahap pemrosesan dari SHA-256, pemrosesan SHA-256 sendiri terdiri atas 1 round yang mempunyai 64 operasi. Untuk memproses setiap satu blok pesan yang terdiri dari 512 bit, di perlukan sebanyak 64 operasi. Setiap blok M1, M2, …., M(n) dengan "n" adalah jumlah blok pesan. Lalu, siapkan penjadwalan pesan dari 16 buah blok hasil parsing tadi menggunakan rumus berikut ;

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

Setelah itu, tahapan selanjutnya ialah inisialisasi *working variable* a, b, c, d, e, f, g dan h untuk blok pesan M tadi dari nilai hash awal.

a = H0

b = H1

c = H2

d = H3

e = H4

f = H5

g = H6

h = H7

dan tahapan selanjutnya menyiapkan 64 koefisian inisialisasi array yang sudah ditetapkan pada standar SHA-2.

```
k[0...63]=
428a2f98, 71374491, b5c0fbcf, e9b5dba5, 3956c25b, 59f111f1, 923f82a4, ab1c5ed5,
d807aa98, 12835b01, 243185be, 550c7dc3, 72be5d74, 80deb1fe, 9bdc06a7, c19bf174,
e49b69c1, efbe4786, 0fc19dc6, 240ca1cc, 2de92c6f, 4a7484aa, 5cb0a9dc, 76f988da,
983e5152, a831c66d, b00327c8, bf597fc7, c6e00bf3, d5a79147, 06ca6351, 14292967,
27b70a85, 2e1b2138, 4d2c6dfc, 53380d13, 650a7354, 766a0abb, 81c2c92e, 92722c85,
a2bfe8a1, a81a664b, c24b8b70, c76c51a3, d192e819, d6990624, f40e3585, 106aa070,
19a4c116, 1e376c08, 2748774c, 34b0bcb5, 391c0cb3, 4ed8aa4a, 5b9cca4f, 682e6ff3,
748f82ee, 78a5636f, 84c87814, 8cc70208, 90befffa, a4506ceb, bef9a3f7, c67178f2
```

Algoritma SHA ini menghitung 0…63 = 64 kali putaran untuk setiap perhitungan blocknya. Delapan variabel yang diberi label tadi yaitu a, b, c, d, e, f, g, dan h nilainya terus berganti selama perputaran sebanyak 64 kali putaran dan hasil dari perhitungan iterasi sebagai berikut :

I = 63

cbfed63a 28e6f83f 95f9f7fb 0ad26aec 47846e35 90815365 c7f695f6 105bc569

lalu hasil dari iterasi terakhir yang tadi kemudian dijumlahkan dengan nilai hash awal, maka akan menghasilkan sebagai berikut :

H0 = cbfed63a + 6a09e667 = 3608bca1

H1 = 28e6f83f + bb67ae85 = e44ea6c4

H2 = 95f9f7fb + 3c6ef372 = d268eb6d

H3 = 0ad26aec + a54ff53a = b0226026

H4 = 47846e35 + 510e527f = 9892c0b4

H5 = 90815365 + 9b05688c = 2b86bbf1

H6 = c7f695f6 + 1f83d9ab = e77a6fa1

H7 = 105bc569 + 5be0cd19 = 6c3c9282

Dan selanjutnya penggabungan hasil akhir dari penjumlahan tadi merupakan perhitungan dari SHA-256, dan terbentuklah message digest variable M, dengan isi pesan asli yaitu "xyz". Maka nilah hash dari pesan M yaitu :

3608bca1e44ea6c4d268eb6db02260269892c0b42b86bbf1e77a6fa16c3c9282

4.2 Penerapan *Advanced Encryption Standard* (AES-256)

Pada penjelasan sebelumnya disebutkan bahwa AES hanya mempunyai panjang block 128 bit dan mempunyai tiga panjang kunci yang diperbolehkan, yaitu 128 bit, 192 bit dan 256 bit. Panjang kunci tersebutlah yang akan menentukan banyaknya perputaran pada "*layer*" AES. Pada penelitian ini, penulis menggunakan panjang kunci 256 bit yang berarti jumlah perputaran pada setiap layernya ialah 14 *round*. Untuk diketahui sebelumnya bahwa setiap teks atau file ialah sebuah aliran dari beberapa bytes. Dan setiap byte adalah delapan bits. Oleh karena itu pada penjelasan tahapan enkripsi file dengan AES-256 dan SHA-256 ini file tersebut akan di pecah menjadi kumpulan bytes. Berikut penjelasan tahapan enkripsi dan dekripsi file menggunakan metode AES-256 dan SHA-256 :

4.2.1 Enkripsi File

Pada tahapan penjelasan enkripsi file dengan aes, penulis akan membagi tiap tiap tahapannya kedalam sub bab - sub bab. Ada empat tahapan yang akan dijelaskan oleh penulis, dari tahapan penurunan kunci (*key derivation*), *padding*, enkripsi blok pertama dan enkripsi blok kedua. Didalam pengenkripsian blok tersebut, ada beberapa tahapan lagi untuk menghasilkan sebuah *chipper text*, yaitu penambahan IV, mencari ronde kunci 2 hingga 14, pengenkripsian ronde pertama dengan tahapan *SubBytes*, *ShiftRows*, *MixColumn* dan *AddRoundKey*. Semua tahapan tersebut akan dijelaskan sebagai berikut.

Siapkan file dan kunci yang akan dienkripsi pada contoh kali ini file yang akan dienkripsi ialah file text berekstensi .txt dengan size file 29 bytes, konversikan file tersebut kedalam code hexadecimal agar dapat melakukan perhitungan tahapan aes. Contoh text dan kunci yang akan digunakan ialah:

Teks asli : "*Hello this is Secret Fichier!*"

Kunci : “xyz”

Tahapan selanjutnya ialah kunci yang kita gunakan disini harus telah di prosess terlebih dahulu dengan fungsi hash atau yang dimaksud dengan istilah *Key Derivation* (penurunan kunci) dengan SHA256. Pada proses sebelumnya, kunci “xyz” telah di proses dengan SHA-256 dan menghasilkan 64 hex digits atau setara dengan 32 bytes (256 bits) dengan kententuan SHA-2. Berikut hasil dari konversi teks asli kedalam heksadesimal dan kunci setelah di proses dengan SHA-256.

Teks asli : 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 63 72 65 74 20 46 69 63 68 69 65 72 21

Kunci : 36 08 bc a1 e4 4e a6 c4 d2 68 eb 6d b0 22 60 26 98 92 c0 b4 2b 86 bb f1 e7 7a 6f a1 6c 3c 92 82

Dikarenakan AES hanya memiliki panjang block dengan ukuran 128 bit sedangkan file teks yang penulis gunakan disini ialah file dengan ukuran 29 bytes (29 x 8 = 232 bit), maka dari itu teks asli akan di bagi menjadi 2 block yang masing masing block tersebut mempiliki panjang 128 bit. Berikut gambaran pembagian block :

48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 |*Hello this is Se*|

63 72 65 74 20 46 69 63 68 69 65 72 21 |*cret Fichier!*|

Dikarenakan pada salah satu panjang block belum mencapai ukuran yang ditetapkan oleh AES, yaitu 128 bit maka di butuhkan penambahan *padding* atau penambahan bit dibelakangnya sebanyak 3 buah (agar menjadi block dengan ukuran 128 bit), yang ditulis dengan bit “03”. Ditulis “03” dikarenakan agar aes mengerti bahwa 2 bit yang dibelakang ialah padding, dan pada saat proses pendekripsian padding tersebut akan di hapus. Berikut gambaran penambahan padding pada block :

48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 |*Hello this is Se*|

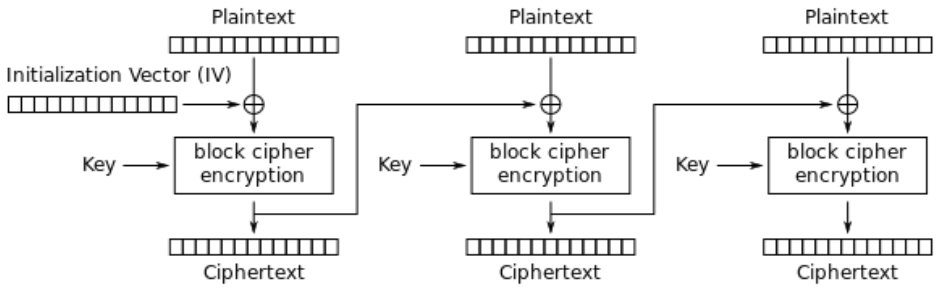
63 72 65 74 20 46 69 63 68 69 65 72 21 03 03 03 |*cret Fichier!*|

Agar lebih mudah penggambaran pada masing masing block, maka penulis akan membuat matriks dengan ukuran 4 x 4 untuk diisi dengan masing masing nilai heksadesimal kedalam block block yang tadi di gambarkan.

Plaintext Block 1			
48	6f	69	73
65	20	73	20
6c	74	20	53
6c	68	69	65

Plaintext Block 2			
63	20	68	21
72	46	69	03
65	69	65	03
74	63	72	03

Dalam pemrosesan AES mode CBC, rangkaian bit-bit pada *plain text* dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (*initialization vector*) untuk menggabungkan dengan *plain text* pertama. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini.



Cipher Block Chaining (CBC) mode encryption

Pada pemrosesan CBC, diperlukannya penambahan IV atau yang disebut sebagai IV (*initialization vector*) . Dalam pemrosesan dengan algoritma aes sudah disebutkan sebelumnya bahwa ada dua input yang dibutuhkan, yaitu input yang pertama ialah teks asli yang sudah di ubah kedalam kode heksadecimal yang akan di proses pengenkripsiannya, yang kedua ialah kunci yang telah di proses dengan fungsi hash sebagai *aeskey* pada aes itu sendiri. Pada penjelasan selanjutnya, penulis akan menjelaskan tahapan pengenkripsian block pertama dengan AES mode CBC.

4.2.2 Enkripsi Blok Pertama

Pada pemrosesan *plain text* block pertama, diperlukan penambahan IV dan *aeskey* pada rumus CBC block pertama, yang di gambarkan sebagai mana berikut ini :

IV : 0000000000000000 0000000000000000 (16 bytes = 128 bits)

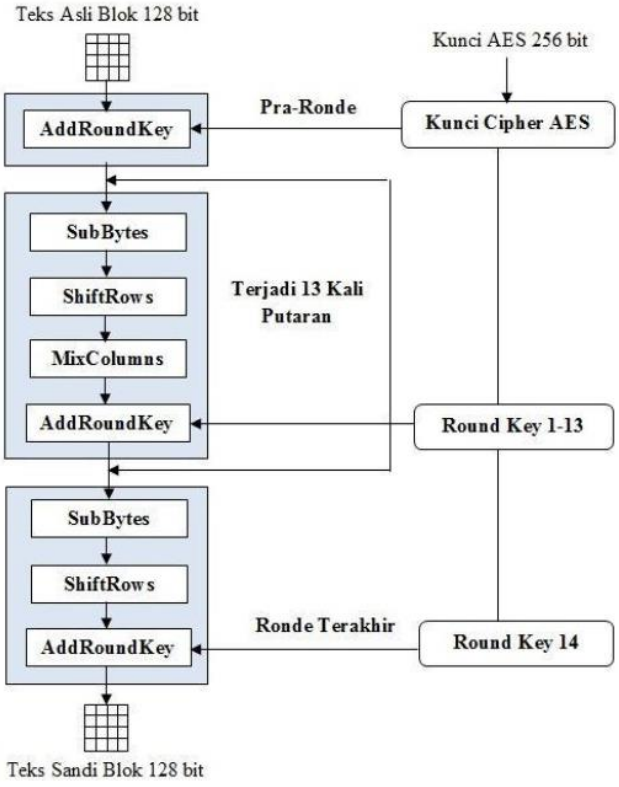
aeskey : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 (32 bytes = 256 bits)

block1 : 48656c48 6f20746f 69732069 73205373

Berikut rumus CBC block pertama :

$$\text{eblock1} = \text{AES}_{\text{aeskey}}(\text{block1} \oplus \text{IV})$$

Selanjutnya, apabila block pertama sudah di proses, maka hasil dari *chiper text* pertama akan menjadi IV selanjutnya pada block kedua dan hasil tersebut juga sekaligus menjadi *chiper text* block pertama, seperti yang digambarkan pada gambar CBC mode diatas. Setelah penambahan IV dan *aeskey* dilakukan, tahapan selanjutnya ialah pengenkripsian di dalam aes. Berikut gambaran proses aes :



Walaupun AES-256 mempunyai panjang kunci 256 bit, tetapi AES hanya memiliki panjang block 128 bit dalam matrik 4x4. Maka block matrik kunci dibagi menjadi 2 bagian. Block bagian pertama dijadikan kunci ronde awal atau RoundKeys 0, dan di bagian kedua dijadikan RoundKeys 1.

RoundKeys 0			
36	e4	d2	b0
08	4e	68	22
bc	a6	eb	60
a1	c4	6d	26

RoundKeys 1			
98	2b	e7	6c
92	86	7a	3c
c0	bb	6f	92
b4	f1	a1	82

Seperti gambar tahapan aes diatas, tahapan pertama yang akan dilakukan ialah proses *initialization key* atau pre round, tahapan tersebut dilakukan proses *AddRoundKey* kunci ronde 0. Pada tahap ini terjadi perhitungan XOR antara teks asli dengan kunci. XOR merupakan kepanjangan dari *Exclusive OR* yang mana keluarannya akan berlogika 1 apabila inputannya berbeda, namun apabila semua inputanya sama maka akan memberikan keluarannya 0. Proses XOR dilakukan pada masing masing blok pada matriks, seperti matriks *plain text* blok kolom [1,1] XOR dengan matriks *RoundKeys* 0 kolom [1,1] dan seterusnya. Berikut akan disimulasikan XOR pada matriks *plain text* blok 1 dengan matriks *RoundKeys* 0.

Untuk dapat melakukan operasi XOR, nilai heksadecimal tadi harus diubah menjadi nilai biner berukuran 8 bit terlebih dahulu. Seperti contoh berikut :

48 = 0100 1000

36 = 0011 0110

Proses XOR disimulasikan seperti dibawah ini :

0	1	0	0	1	0	0	0	
0	0	1	1	0	1	1	0	$\oplus$
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	

Nilai heksadesimal dari 0111 1110 ialah 7E, jadi 48 ⊕ 36 = 7E

Hitung keseluruhan nilai XOR di kolom kolom matriks *plain text* blok 1 dengan *RoundKeys* 0 hingga mengahasilkan blok matriks berukuran 4 x 4.

65 ⊕ 08 = 6d

6c ⊕ bc = d0

6c ⊕ a1 = cd

6f ⊕ e4 = 8b

20 ⊕ 4e = 6e

74 ⊕ bc = d2

68 ⊕ c4 = ac

69 ⊕ d2 = bb

73 ⊕ 68 =1b

20 ⊕ eb = cb

69 ⊕ 6d = 04

73 ⊕ b0 = c3

20 ⊕ 22 = 02

53 ⊕ 60 = 33

65 ⊕ 26 = 43

Berikut matriks dari *plaintext* block 1 XOR RoundKeys 0

Plaintext					RoundKeys 0					Hasil			
48	6f	69	73		**36**	e4	d2	b0		**7e**	8b	8b	c3
65	20	73	20	⊕	**08**	4e	68	22	=	**6d**	6e	1b	02
6c	74	20	53		**bc**	a6	eb	60		**d0**	d2	cb	33
6c	68	69	65		**a1**	c4	6d	26		**cd**	ac	04	43

Pada tahapan sebelumnya penambahan IV dengan operasi XOR dilakukan juga, akan tetapi karena IV awal bernilai 16 hex digit nilai “00” maka

hasilnyaa akan sama seperti aslinya. Selanjutnya pada tahap ini, terdapat perulangan sebanyak 13 kali. Setiap prosess *AddRoundKey* dilakukan operasi XOR terhadap *RoundKeys* 1 sampai dengan *RoundKeys 13*. Berikut merupakan tahapan dari 13 kali pengulangan :

a. Proses *SubBytes*

Pada proses *SubBytes* ini, hasil dari operasi XOR di pre-round akan di di subtitusikan dengan nilai yang ada di table S-Box (tabel 2.8.1 hlm.16). proses dari *SubBytes* ialah sebagai berikut :

7e	8b	8b	c3
6d	6e	1b	02
d0	d2	cb	33
cd	ac	04	43

Lakukan proses subtitusi S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (7) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (e) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil *SubByte*. Seperti contoh gambar berikut :

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Setelah dilakukan subtitusi S-box pada proses *SubBytes* pada nilai 7e, maka didapatkan hasil yaitu **f3**. Lalu lanjutkan subtitusi S-box pada kolom [2.1] matriks diatas yaitu nilai 6d.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai 6d, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses *SubBytes* 6d mendapatkan hasil nilai **3c**. Lalu lanjutkan subtitusi S-box pada kolom [3.1] matriks diatas yaitu nilai d0.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai d0, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses *SubBytes* d0 mendapatkan hasil nilai **70**. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai cd.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai cd, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses *SubBytes* d0 mendapatkan hasil nilai **bd**. Lalu lanjutkan subtitusi S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table S-Box proses SubBytes. Setelah dilakukan proses *SubBytes* pada keseluruhan nilai, maka didapatkan hasil dari proses *SubBytes* ialah sebagai berikut:

Hasil *SubBytes*			
f3	3d	ea	2e
3c	9f	af	77
70	b5	1f	c3
bd	91	f2	1a

b. Proses *ShiftRows*

Lalu tahapan selanjutnya ialah proses *ShiftRows*. Hasil dari proses *SubBytes* tadi akan di geser beberapa bit di dalam proses ini, berikut gambaran yang terjadi pada proses *ShiftRows* ini :

Pada nilai yang ditandai akan di putar lebih satu byte, seperti berikut ini

f3	3d	ea	2e		f3	3d	ea	2e
3c	9f	af	77	→	9f	af	77	3c
70	b5	1f	c3		70	b5	1f	c3
bd	91	f2	1a		bd	91	f2	1a

Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut:

f3	3d	ea	2e		f3	3d	ea	2e
3c	9f	af	77		9f	af	77	3c
70	b5	1f	c3	→	1f	c3	70	b5

bd	91	f2	1a		bd	91	f2	1a

Pada nilai yang ditandai akan diputar lebih dari tiga byte, seperti berikut:

f3	3d	ea	2e		f3	3d	ea	2e
3c	9f	af	77		9f	af	77	3c
70	b5	1f	c3		1f	c3	70	b5
bd	91	f2	1a	➡	1a	bd	91	f2

Hasil dari proses ShiftRows ialah sebagai barikut :

Hasil *ShiftRows*			
f3	3d	ea	2e
9f	af	77	3c
1f	c3	70	b5
1a	bd	91	f2

c. Proses *MixColumns*

Proses selanjutnya adalah proses *MixColumns*, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada proses *ShiftRows* tadi dengan matriks yang telah di tetapkan oleh *Rijndael*. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial dan pergeseran beberapa bit. Berikut merupakan proses dari *MixColumns* :

02	03	01	01	X	f3	3d	ea	2e	➡	C_{11}	C_{12}	C_{13}	C_{14}
01	02	03	01		9f	af	77	3c		C_{21}	C_{22}	C_{23}	C_{24}
01	01	02	03		1f	c3	70	b5		C_{31}	C_{32}	C_{33}	C_{34}
03	01	01	02		1a	bd	91	f2		C_{41}	C_{42}	C_{43}	C_{44}

Operasi perkalian disini maksudnya adalah:

1. Perkalian dengan 01 artinya tidak berubah
2. Perkalian dengan 02 artinya mengalikan dengan perkalian polynomial dan mengubah nilai yang di kali kedalam nilai biner dan di lakukan perkalian polynomial serta modulo pada biner.
3. Perkalian dengan 03 mengalikan dengan perkalian polynomial dan mengubah nilai yang di kali kedalam nilai biner dan di lakukan perkalian polynomial serta modulo pada biner.

Berikut ini merupakan perhitungan pencarian nilai untuk C_{11} :

02	03	01	01		f3
01	02	03	01	X	9f
01	01	02	03		1f
03	01	01	02		1a

C_{11} = {02.f3} ⊕ {03.9f} ⊕ {01.1f} ⊕ {01.1a}

- {02.f3}

  02 = 0000 0010 = x

  f3 = 1111 0011 = $x^7 + x^6 + x^5 + x^4 + x + 1$

  = $(x)(x^7 + x^6 + x^5 + x^4 + x + 1)$

  = $x^8 + x^7 + x^6 + x^5 + x^2 + x$

  = $x^8 + x^7 + x^6 + x^5 + x^2 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 1 1110 0110

  1 0001 1011 (XOR)

  1111 1101 (**fd**)

- {03.9f}

  03 = 0000 0011 = x + 1

  9f = 1001 1111 = $x^7 + x^4 + x^3 + x^2 + x + 1$

  = $(x + 1)(x^7 + x^4 + x^3 + x^2 + x + 1)$

  = $(x^8 + x^5 + x^4 + x^3 + x^2 + x) + (x^7 + x^4 + x^3 + x^2 + x + 1)$

  = $x^8 + x^7 + x^5 + 1$

  = $x^8 + x^7 + x^5 + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 1 1010 0001

  1 0001 1011 (XOR)

  0001 1010 (**ba**)

- {01.1f} = 1f

- {01.1a} = 1a

  Perhitungan : C_{11} = {02.f3} ⊕ {03.9f} ⊕ {01.1f} ⊕ {01.1a}

  C_{11} = {fd} ⊕ {ba} ⊕ {1f} ⊕ {1a} = 42

  Berikut ini merupakan perhitungan pencarian nilai untuk C_{21} :

02	03	01	01	X	f3
01	02	03	01		9f
01	01	02	03		1f
03	01	01	02		1a

  C_{21} = {01.f3} ⊕ {02.9f} ⊕ {03.1f} ⊕ {01.1a}

- {01.f3} = f3
- {02.9f}

  02 = 0000 0010 = x

  9f = 1001 1111 = $x^7 + x^4 + x^3 + x^2 + x + 1$

  = $(x)(x^7 + x^4 + x^3 + x^2 + x + 1)$

  = $x^8 + x^5 + x^4 + x^3 + x^2 + x$

  = $x^8 + x^5 + x^4 + x^3 + x^2 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 1 0011 1110

  1 0001 1011 (XOR)

  0010 0101 (**25**)

- {03.1f}

  03 = 0000 0011 = x + 1

  1f = 0001 1111 = $x^4 + x^3 + x^2 + x + 1$

  = $(x + 1)(x^4 + x^3 + x^2 + x + 1)$

  = $(x^5 + x^4 + x^3 + x^2 + x) + (x^4 + x^3 + x^2 + x + 1)$

  = $x^5 + 1$

  = 0010 0001 (**21**)

- {01.1a} = 1a

  Perhitungan : C_{21} = {01.f3} ⊕ {02.9f} ⊕ {03.1f} ⊕ {01.1a}

  C_{21} = {f3} ⊕ {25} ⊕ {21} ⊕ {1a} = ed

  Berikut ini merupakan perhitungan pencarian nilai untuk C_{31} :

02	03	01	01	X	f3
01	02	03	01		9f
01	01	02	03		1f
03	01	01	02		1a

  C_{31} = {01.f3} ⊕ {01.9f} ⊕ {02.1f} ⊕ {03.1a}

- {01.f3} = f3
- {01.9f} = 9f
- {02.1f}

  02 = 0000 0010 = x

  1f = 0001 1111 = $x^4 + x^3 + x^2 + x + 1$

  = $(x)(x^4 + x^3 + x^2 + x + 1)$

  = $x^5 + x^4 + x^3 + x^2 + x$

  = 0011 1110 (**3e**)

- {03.1a}

  03 = 0000 0011 = x + 1

  1a = 0001 1010 = $x^4 + x^3 + x$

  = $(x + 1)(x^4 + x^3 + x)$

  = $(x^5 + x^4 + x^2) + (x^4 + x^3 + x)$

  = $x^5 + x^3 + x^2 + x$

  = 0010 1110 (**2e**)

  Perhitungan : C_{31} = {01.f3} ⊕ {01.9f} ⊕ {02.1f} ⊕ {03.1a}

  = {f3} ⊕ {9f} ⊕ {3e} ⊕ {2e}

  = **7c**

  Berikut ini merupakan perhitungan pencarian nilai untuk C_{41}:

02	03	01	01	X	f3
01	02	03	01		9f
01	01	02	03		1f
03	01	01	02		1a

C_{41} = {03.f3} ⊕ {01.9f} ⊕ {01.1f} ⊕ {02.1a}

- {03.f3}

  03 = 0000 0011 = x + 1

  f3 = 1111 0011 = $x^7 + x^6 + x^5 + x^4 + x + 1$

  = $(x + 1)(x^7 + x^6 + x^5 + x^4 + x + 1)$

  = $(x^8 + x^7 + x^6 + x^5 + x^2 + x) + (x^7 + x^6 + x^5 + x^4 + x + 1)$

  = $x^8 + x^4 + x^2 + 1$

  = 0000 1110 (**0e**)

- {01.9f} = 9f
- {01.1f} = 1f
- {02.1a}

  02 = 0000 0010 = x

  1a = 0001 1010 = $x^4 + x^3 + x$

  = $(x)(x^4 + x^3 + x)$

  = $x^5 + x^4 + x^2$

  = 0011 0100 (**34**)

Perhitungan : C_{41} = {03.f3} ⊕ {01.9f} ⊕ {01.1f} ⊕ {02.1a}

= {0e} ⊕ {9f} ⊕ {1f} ⊕ {34}

= **ba**

Untuk mendapat hasil yang lainnya C_{12}, C_{22}, C_{32}, C_{42}, C_{13}, C_{23}, C_{33}…C_{44} lakukan perhitungan polynomial dengan cara seperti diatas. Berikut hasilnya :

C_{12} = {02.3d} ⊕ {03.af} ⊕ {01.c3} ⊕ {01.bd}

= {7a} ⊕ {ea} ⊕ {c3} ⊕ {bd} = **ee**

C_{22} = {01.3d} ⊕ {02.af} ⊕ {03.c3} ⊕ {01.bd}

= {3d} ⊕ {45} ⊕ {5e} ⊕ {bd} = **9b**

C_{32} = {01.3d} ⊕ {01.af} ⊕ {02.c3} ⊕ {03.bd}

= {3d} ⊕ {af} ⊕ {9d} ⊕ {dc} = **d3**

C_{42} = {03.3d} ⊕ {01.af} ⊕ {01.c3} ⊕ {02.bd}

= {47} ⊕ {af} ⊕ {c3} ⊕ {61} = **4a**

C_{13} = {02.ea} ⊕ {03.77} ⊕ {01.70} ⊕ {01.91}

= {cf} ⊕ {99} ⊕ {70} ⊕ {91} = **b7**

C_{23} = {01.ea} ⊕ {02.77} ⊕ {03.70} ⊕ {01.91}

= {ea} ⊕ {ee} ⊕ {90} ⊕ {91} = **05**

C_{33} = {01.ea} ⊕ {01.77} ⊕ {02.70} ⊕ {03.91}

= {ea} ⊕ {77} ⊕ {e0} ⊕ {a8} = **d5**

C_{43} = {03.ea} ⊕ {01.77} ⊕ {01.70} ⊕ {02.91}

= {25} ⊕ {77} ⊕ {70} ⊕ {39} = **1b**

C_{14} = {02.2e} ⊕ {03.3c} ⊕ {01.b5} ⊕ {01.f2}

= {5c} ⊕ {44} ⊕ {b5} ⊕ {f2} = **5f**

C_{24} = {01.2e} ⊕ {02.3c} ⊕ {03.b5} ⊕ {01.f2}

= {2e} ⊕ {78} ⊕ {c4} ⊕ {f2} = **60**

C_{34} = {01.2e} ⊕ {01.3c} ⊕ {02.b5} ⊕ {03.f2}

= {2e} ⊕ {3c} ⊕ {71} ⊕ {0d} = **6e**

C_{44} = {03.2e} ⊕ {01.3c} ⊕ {01.b5} ⊕ {02.f2}

= {72} ⊕ {3c} ⊕ {b5} ⊕ {ff} = **04**

Hasil dari proses *MixColumns* adalah sebagai berikut :

Hasil *MixColumns*			
42	ee	b7	5f
ed	9b	05	60
7c	d3	d5	6e
ba	4a	1b	04

d. Proses *AddRoundKeys*

Proses selanjutnya ialah *AddRoundKey*, setelah hasil yang didapat dalam proses *MixColumns* selanjutnya akan melakukan operasi XOR dengan *RoundKeys* 1 terhadap hasil dari *MixColumns*. Seperti gambar berikut ini :

Hasil MixColumn					KeyRound1					Hasil Ronde 1			
42	ee	b7	5f	XOR	98	2b	e7	6c	=	da	c5	50	33
ed	9b	05	60		92	86	7a	3c		7f	1d	7f	5c
7c	d3	d5	6e		c0	bb	6f	92		bc	68	ba	fc
ba	4a	1b	04		b4	f1	a1	82		0e	bb	ba	86

Setelah mendapat hasi dari *AddRoundKey* ronde pertama, selanjutnya pencarian nilai ronde kedua hingga ronde ketiga belas dan melalui proses proses sebelumnya, yaitu proses *SubBytes*, proses *ShiftRows* dan proses *MixColumns* dan pencarian kunci di tiap rondenya untuk pemrosesan XOR dengan *MixColumns* (proses *AddRoundKey*) nanti. Yang kita ketahui kunci yang kita gunakan disini ialah “xyz” yang telah di proses dengan fungsi hash SHA-256 dan menghasilkan 32 bytes atau setara dengan 256 bits, yaitu sebagai berikut :

3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282

Karena tiap ronde kunci aes hanya berisikan 16 bytes tiap rondenya, maka penulis akan membagi dua matriks kunci tersebut dan menaruhnya kedalam matriks 4x4 atau 16 bytes, seperti sebagai berikut ;

RoundKeys 0			
36	e4	d2	b0
08	4e	68	22
bc	a6	eb	60
a1	c4	6d	26

RoundKeys 1			
98	2b	e7	6c
92	86	7a	3c
c0	bb	6f	92
b4	f1	a1	82

Penulis sudah mengetahui ronde kunci 0 dan 1, oleh karena itu langkah selanjutnya ialah mencari ronde kunci 2 hingga 14. Berikut tahapan pencarian ronde ronde kunci yaitu :

a. *RoundKeys* 2

Pencarian kunci ronde 2 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 1.

RotWord RoundKeys 2			
98	2b	e7	c3
92	86	7a	92
c0	bb	6f	82
b4	f1	a1	6c

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box RoundKeys 2			
98	2b	e7	eb
92	86	7a	4f
c0	bb	6f	13
b4	f1	a1	50

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 0 dan kolom 1 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 2.

eb	XOR	36	XOR	01	=	dc
4f		08		00		47
13		bc		00		af
50		a1		00		f1

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 2 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 2 kolom 1 dengan operasi XOR *Roundkeys* 0 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 2 :

RoundKeys 2			
dc	38	ea	5a
47	09	61	43
af	09	e2	82
f1	35	58	7e

b. *RoundKeys* 3

Pencarian kunci ronde 3 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini :

Transformation S-box RoundKeys 3			
dc	38	ea	be
47	09	61	1a
af	09	e2	13
f1	35	58	f3

Hasil transformasi *SubBytes* yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 1. Hasilnya adalah kolom 1 kunci ronde 3.

be	XOR	98	=	26
1a		92		88
13		c0		d3
f3		b4		47

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 3 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 3 kolom 1 dengan operasi XOR *Roundkeys* 1 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *Roundkeys* 3. Berikut hasil dari RoundKeys 3 :

RoundKeys 3			
26	0d	ea	86
88	0e	74	48
d3	68	07	95
47	b6	17	95

c. *RoundKeys* 4

Pencarian kunci ronde 4 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 3.

RotWord RoundKeys 4			
26	0d	ea	48
88	0e	74	95
d3	68	07	95
47	b6	17	86

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box RoundKeys 4			
26	0d	ea	52
88	0e	74	2a
d3	68	07	2a
47	b6	17	44

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 2 dan kolom 2 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 4.

52	XOR	dc	XOR	02	=	8c
2a		47		00		6d
2a		af		00		85
44		f1		00		B5

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 4 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 4 kolom 1 dengan operasi XOR *RoundKeys* 2 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 4 :

RoundKeys 4			
8c	b4	5e	04
6d	64	05	46
85	8c	6e	ec
b5	80	d8	a6

d. *RoundKeys* 5

Pencarian kunci ronde 5 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini :

Transformation S-box *RoundKeys* 5			
8c	b4	5e	f2
6d	64	05	5a
85	8c	6e	ce
b5	80	d8	24

Hasil transformasi **SubBytes** yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 3. Hasilnya adalah kolom 1 kunci ronde 5.

f2	XOR	26	=	d4
5a		88		d2
ce		d3		1d
24		47		63

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 5 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 5 kolom 1 dengan operasi XOR *RoundKeys* 3 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 5.

RoundKeys 5			
d4	d9	33	b5
d2	dc	a8	e0
1d	75	72	e7
63	d5	c2	57

e. *RoundKeys* 6

Pencarian kunci ronde 6 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 5.

RotWord RoundKeys 6			
d4	d9	33	e0
d2	dc	a8	e7
1d	75	72	57
63	d5	c2	b5

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box RoundKeys 6			
d4	d9	33	e1
d2	dc	a8	94
1d	75	72	5b
63	d5	c2	d5

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 4 dan kolom 3 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 6

e1	XOR	8c	XOR	04	=	69
94		6d		00		f9
5b		85		00		de
d5		b5		00		60

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 6 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 6 kolom 1 dengan operasi XOR *RoundKeys* 4 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 6 :

RoundKeys 6			
69	dd	83	87
f9	9d	98	de
de	52	3c	d0
60	e0	38	9e

f. *RoundKeys* 7

Pencarian kunci ronde 7 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini :

Transformation S-box RoundKeys 7			
69	dd	83	17
f9	9d	98	1d
de	52	3c	70
60	e0	38	0b

Hasil transformasi *SubBytes* yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 5. Hasilnya adalah kolom 1 kunci ronde 7.

17	XOR	d4	=	c3
1d		d2		cf
70		1d		6d
0b		63		68

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 7 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 7 kolom 1 dengan operasi XOR *RoundKeys* 5 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 7. Berikut hasil dari RoundKeys 7 :

RoundKeys 7			
c3	1a	29	9c
cf	13	bb	5b
6d	18	6a	8d
68	bd	7f	28

g. *RoundKeys* 8

Pencarian kunci ronde 8 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 7.

RotWord RoundKeys 8			
c3	1a	29	5b
cf	13	bb	8d
6d	18	6a	28
68	bd	7f	9c

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box RoundKeys 6			
c3	1a	29	39
cf	13	bb	5d
6d	18	6a	34
68	bd	7f	de

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 6 dan kolom 4 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 8.

39	XOR	69	XOR	08	=	58
5d		f9		00		a4
34		de		00		ea
de		60		00		be

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 8 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 8 kolom 1 dengan operasi XOR *RoundKeys* 6 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 8 :

RoundKeys 8			
58	85	06	81
a4	39	a1	7f
ea	b8	84	54
be	5e	66	f8

h. *RoundKeys* 9

Pencarian kunci ronde 9 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1

hlm.16). Seperti gambar dibawah ini :

Transformation S-box RoundKeys 9			
58	85	06	0c
a4	39	a1	d2
ea	b8	84	20
be	5e	66	41

Hasil transformasi *SubBytes* yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 7. Hasilnya adalah kolom 1 kunci ronde 9.

0c	XOR	c3	=	cf
d2		cf		1d
20		6d		4d
41		68		29

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 9 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 9 kolom 1 dengan operasi XOR *RoundKeys* 7 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundkKeys* 9.

RoundKeys 9			
cf	d5	fc	60
1d	0e	b5	ee
4d	55	3f	b2
29	94	eb	c3

i. *RoundKeys* 10

Pencarian kunci ronde 10 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 9.

RotWord RoundKeys 10			
cf	d5	fc	ee
1d	0e	b5	b2
4d	55	3f	c3
29	94	eb	60

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box *RoundKeys* 10			
cf	d5	fc	28
1d	0e	b5	37
4d	55	3f	2e
29	94	eb	d0

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 8 dan kolom 5 tabel RCon (hlm.17). Hasilnya adalah kolom 1 kunci ronde 10.

28	XOR	58	XOR	10	=	60
37		a4		00		93
2e		ea		00		c4
d0		be		00		6e

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 10 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 10 kolom 1 dengan operasi XOR *RoundKeys* 8 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 10 :

RoundKeys 10			
60	e5	e3	62
93	aa	0b	74
c4	7c	f8	ac
6e	30	56	ae

j. *RoundKeys* 11

Pencarian kunci ronde 11 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini :

Transformation S-box RoundKeys 11			
60	e5	e3	aa
93	aa	0b	92
c4	7c	f8	91
6e	30	56	e4

Hasil transformasi *SubBytes* yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 9. Hasilnya adalah kolom 1 kunci ronde 11.

aa	XOR	cf	=	65
92		1d		8f
91		4d		dc
e4		29		cd

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 11 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 11 kolom 1 dengan operasi XOR *RoundKeys* 9 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 11.

RoundKeys 11			
65	b0	4c	2c
8f	81	34	da
dc	89	b6	04
cd	59	b2	71

k. *RoundKeys* 12

Pencarian kunci ronde 12 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 11.

RotWord RoundKeys 12			
65	b0	4c	da
8f	81	34	04
dc	89	b6	71
cd	59	b2	2c

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box *RoundKeys* 12			
65	b0	4c	57
8f	81	34	f2
dc	89	b6	a3
cd	59	b2	71

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 10 dan kolom 6 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 12.

57	XOR	60	XOR	20	=	17
f2		93		00		61
a3		c4		00		67
71		6e		00		1f

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 12 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 12 kolom 1 dengan operasi XOR *RoundKeys* 10 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 12 :

RoundKeys 12			
17	f2	11	73
61	cb	c0	b4
67	1b	e3	4f
1f	2f	79	d7

l. *RoundKeys* 13

Pencarian kunci ronde 13 tidak menggunakan transformasi *RotWord*. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini :

Transformation S-box RoundKeys 13			
17	f2	11	8f
61	cb	c0	8d
67	1b	e3	84
1f	2f	79	0e

Hasil transformasi *SubBytes* yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 11. Hasilnya adalah kolom 1 kunci ronde 13.

8f	XOR	65	=	ea
8d		8f		02
84		dc		58
0e		cd		c3

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 13 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 13 kolom 1 dengan operasi XOR *RoundKeys* 11 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 13.

RoundKeys 13			
ea	5a	16	3a
02	83	b7	6d
58	d1	67	63
c3	9a	28	59

m. *RoundKeys* 14

Pencarian kunci ronde 14 dengan menggunakan transformasi *RotWord*. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 13.

RotWord RoundKeys 14			
ea	5a	16	6d
02	83	b7	63
58	d1	67	59
c3	9a	28	3a

Kemudian dilakukan operasi *SubWord* terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16).

Transformation S-Box RoundKeys 14			
ea	5a	16	3c
02	83	b7	fb
58	d1	67	cb
c3	9a	28	80

Hasil *SubBytes* tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 12 dan kolom 6 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 14.

3c	XOR	17	XOR	40	=	6b
fb		61		00		9a
cb		67		00		ac
80		1f		00		9f

Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 14 yang di perlu dilakukan ialah lakukan hasil *RoundKeys* 14 kolom 1 dengan operasi XOR *RoundKeys* 12 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk *RoundKeys* 14 :

RoundKeys 14			
6b	99	88	fb
9a	51	91	25
ac	b7	54	1b
9f	b0	c9	1e

Pada pengenkripsian blok pertama, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah di operasikan XOR dari *RoundKeys* 2 sampai 14. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan terlalu panjangnya tahapan tahapannya. Berikut hasil nilai di setiap rondenya :

Hasil Ronde 2				Hasil Ronde 3				Hasil Ronde 4			
35	19	1d	1b	03	8f	a6	64	85	4d	97	59
00	70	e3	04	b2	1e	d0	fa	3c	3f	7a	83
63	e0	14	1e	d6	d9	9c	be	e9	ab	90	30
d0	eb	4d	44	7b	52	20	20	e1	89	6f	14
Hasil Ronde 5				Hasil Ronde 6				Hasil Ronde 7			
e4	8d	ae	d4	79	31	1e	f3	e1	45	da	fe
f5	64	66	e8	d3	7c	78	22	8d	bf	ea	8a
2a	57	d8	e0	9c	d8	b5	41	b1	df	85	ea
3b	de	e6	d3	35	0a	4b	fc	17	e7	6a	45
Hasil Ronde 8				Hasil Ronde 9				Hasil Ronde 10			
52	bc	76	97	1f	b3	d3	01	07	2c	69	7a
80	20	dd	c5	62	a5	c8	d2	27	8f	f1	93
9d	4f	81	93	c0	2a	16	19	44	7d	0c	90
ee	17	1a	e9	ae	76	a0	56	0d	68	83	be
Hasil Ronde 11				Hasil Ronde 12				Hasil Ronde 13			
31	1d	84	df	18	9c	fe	6e	c1	c0	26	90
1b	de	06	75	25	ff	d5	a1	8d	3c	8e	3d
64	fb	6a	d8	63	3e	4e	1a	3c	e6	08	38
53	be	ef	f4	37	17	01	6b	e8	6d	cc	96

Pada ronde terakhir, yaitu ronde ke 14, process MixColumn ditiadakan, atau dilewatkan. Hasil ronde ke 14 inilah yang menjadi hasil akhir dari pengenkripsian blok pertama sekaligus menjadi chippertext pada blok pertama dan menjadi IV untuk blok selanjutnya. Hasil ronde 14 adalah:

Hasil Ronde 14			
13	23	7f	9b
71	48	b6	78
9c	b0	bf	95
0f	2b	f5	55

4.2.3 Enkripsi Blok Kedua

Pada pemrosesan *plain text* block kedua, diperlukan penambahan IV, IV yang digunakan ialah hasil dari chippertext block pertama dan aeskey yang sama. Pada enkripsi blok pertama, *RoundKeys* 2 sampai 14 belum diketahui oleh sebab itu kita harus mencari tahu terlebih dahulu, pada enkripsi blok kedua penulis tidak perlu mencari *RoundKeys* dikarenakan *RoundKeys* 1 sampai 14 adalah sama dengan *RoundKeys* pada enkripsi blok

pertama. Rumus CBC blok pertama akan digambarkan sebagai mana berikut ini :

IV : 13719c0f 2348b02b 7fb6bff5 9b789555 (16 bytes = 128 bits)

aeskey : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 (32 bytes = 256 bits)

block2 : 63726574 20466963 68696572 21030303

Berikut rumus CBC block kedua :

$$eblock2 = AES(block2 \oplus eblock1, aeskey)$$

Maka dari itu hasi akhir dari kedua enkripsi diatas yaitu enkripsi blok pertama dan enkripsi blok kedua sebagai rumus berikut :

$$Hasil = eblock1 \,||\, eblock2$$

Tahapan selanjutnya ialah *pre round* atau *initialization key*, dimana dilakukannya operasi XOR pada IV dengan plaintext block 2 dengan *RoundKeys* 0 yang sudah diketahui sebelumnya agar mendapatkasih hasil untuk diproses selanjutnya. Berikut gambarannya :

IV					Plaintext block 2					RoundKey 0			
13	23	7f	9b		**63**	20	68	21		**36**	e4	d2	b0
71	48	b6	78	⊕	**72**	46	69	03	⊕	**08**	4e	68	22
9c	b0	bf	95		**65**	69	65	03		**bc**	a6	eb	60
0f	2b	f5	55		**74**	63	72	03		**a1**	c4	6d	26

Hasil			
46	e7	c5	0a
0b	40	b7	59
45	7f	31	f6
da	8c	ea	70

a. Proses *SubBytes*

Pada proses SubBytes ini, hasil dari operasi XOR di pre-round akan di di subtitusikan dengan nilai yang ada di table S-Box (tabel 2.8.1 hlm.16). proses dari SubBytes ialah sebagai berikut :

46	e7	c5	0a
0b	40	b7	59
45	7f	31	f6
da	8c	ea	70

Lakukan proses subtitusi S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (4) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (6) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil *SubByte*. Seperti contoh gambar berikut :

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Setelah dilakukan subtitusi S-box pada proses *SubBytes* pada nilai 46, maka didapatkan hasil yaitu **5a**. Lalu lanjutkan subtitusi S-box pada kolom [2.1] matriks diatas yaitu nilai 0b.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai 0b, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-

box proses *SubBytes* 0b mendapatkan hasil nilai **2b**. Lalu lanjutkan subtitusi S-box pada kolom [2.3] matriks diatas yaitu nilai 45.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai 45, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses *SubBytes* 45 mendapatkan hasil nilai **6e**. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai da.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pada proses Subtitusi S-box pada nilai cd, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses *SubBytes* da mendapatkan hasil nilai **57**. Lalu lanjutkan subtitusi S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table S-Box proses SubBytes. Setelah dilakukan proses SubBytes pada keseluruhan nilai, maka didapatkan hasil dari proses SubBytes ialah sebagai berikut:

Hasil *SubBytes*			
5a	94	a6	67
2b	09	a9	cb
6e	d2	c7	42
57	64	87	51

b. Proses *ShiftRows*

Lalu tahapan selanjutnya ialah proses *ShiftRows*. Hasil dari proses *SubBytes* tadi akan di geser beberapa bit di dalam proses ini, berikut gambaran yang terjadi pada proses *ShiftRows* ini :

Pada nilai yang ditandai akan di putar lebih 1 byte, seperti berikut ini

5a	94	a6	67		5a	94	a6	67
2b	09	a9	cb	→	09	a9	cb	2b
6e	d2	c7	42		6e	d2	c7	42
57	64	87	51		57	64	87	51

Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut:

5a	94	a6	67		5a	94	a6	67
09	a9	cb	2b		09	a9	cb	2b
6e	d2	c7	42	→	c7	42	6e	d2
57	64	87	51		57	64	87	51

Pada nilai yang ditandai akan diputar lebih dari tiga byte, seperti berikut:

5a	94	a6	67		5a	94	a6	67
09	a9	cb	2b		09	a9	cb	2b
c7	42	6e	d2		c7	42	6e	d2
57	64	87	51	→	51	57	64	87

Hasil dari proses *ShiftRows* ialah sebagai barikut :

Hasil *ShiftRows*			
5a	94	a6	67
09	a9	cb	2b
c7	42	6e	d2
51	57	64	87

c. Proses *MixColumns*

Proses selanjutnya adalah proses *MixColumns*, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada prose

ShiftRows tadi dengan matriks yang telah di tetapkan oleh *Rijndael*. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial dan pergeseran beberapa bit. Berikut merupakan proses dari *MixColumns* :

02	03	01	01	X	5a	94	a6	67	➡	C_{11}	C_{12}	C_{13}	C_{14}
01	02	03	01		09	a9	cb	2b		C_{21}	C_{22}	C_{23}	C_{24}
01	01	02	03		c7	42	6e	d2		C_{31}	C_{32}	C_{33}	C_{34}
03	01	01	02		51	57	64	87		C_{41}	C_{42}	C_{43}	C_{44}

Operasi perkalian disini maksudnya adalah :

1. Perkalian dengan 01 artinya tidak berubah.
2. Perkalian dengan 02 artinya perkalian polynomial, nilai tersebut terlebih dahulu diubah kedalam bentuk biner dan dilakukan perkalian polynomial dan modulo untuk nilai binernya.
3. Perkalian dengan 03 artinya perkalian polynomial, nilai tersebut terlebih dahulu diubah kedalam bentuk biner dan dilakukan perkalian polynomial dan modulo untuk nilai binernya.

Berikut ini merupakan perhitungan pencarian nilai untuk C_{11} :

02	03	01	01	X	5a
01	02	03	01		09
01	01	02	03		c7
03	01	01	02		51

C_{11} = {02.5a} ⊕ {03.09} ⊕ {01.c7} ⊕ {01.51}

- {02.5a}

  02 = 0000 0010 = x

  5a = 0101 1010 = $x^6 + x^4 + x^3 + x$

  = $(x)(x^6 + x^4 + x^3 + x)$

  = $x^7 + x^5 + x^4 + x^2$

  = 1011 0100 (**b4**)

- {03.09}

  03 = 0000 0011 = x + 1

  09 = 0000 1001 = $x^3 + 1$

  = $(x + 1)(x^3 + 1)$

  = $(x^4 + x) + (x + 1)$

  = $x^4 + x^3 + x + 1$

  = 0001 1011 (**1b**)

- {01.c7} = c7
- {01.51} = 51

Perhitungan : C_{11} = {02.5a} ⊕ {03.09} ⊕ {01.c7} ⊕ {01.51}

C_{11} = {b4} ⊕ {1b} ⊕ {c7} ⊕ {51} = 39

Berikut ini merupakan perhitungan pencarian nilai untuk C_{21} :

02	03	01	01	X	5a
01	02	03	01		09
01	01	02	03		c7
03	01	01	02		51

C_{21} = {01.5a} ⊕ {02.09} ⊕ {03.c7} ⊕ {01.51}

- {01.5a} = 5a
- {02.09}

  02 = 0000 0010 = x

  09 = 0000 1001 = $x^3 + 1$

  = $(x)(^3 + 1)$

  = $x^7 + x^5 + x^4 + x^2$

  = 1011 0100 (**12**)

- {03.c7}

  03 = 0000 0011 = x + 1

c7 = 1100 0111 $= x^7 + x^6 + x^2 + x + 1$

= $(x + 1)(x^7 + x^6 + x^2 + x + 1)$

= $(x^8 + x^7 + x^3 + x^2 + x) + (x^7 + x^6 + x^2 + x + 1)$

= $x^8 + x^6 + x^3 + 1$

= $x^8 + x^6 + x^3 + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 1 0100 1001

1 0001 1011 (XOR)

0101 0010 (**52**)

- {01.51} = 51

Perhitungan : C_{21} = {01.5a} ⊕ {02.09} ⊕ {03.c7} ⊕ {01.51}

C_{21} = {5a} ⊕ {12} ⊕ {52} ⊕ {51} = 4b

Berikut ini merupakan perhitungan pencarian nilai untuk C_{31} :

02	03	01	01	X	5a
01	02	03	01		09
01	01	02	03		c7
03	01	01	02		51

C_{31} = {01.5a} ⊕ {01.09} ⊕ {02.c7} ⊕ {03.51}

- {01.5a} = 5a
- {01.09} = 09
- {02.c7}

02 = 0000 0010 $= x$

c7 = 1100 0111 $= x^7 + x^6 + x^2 + x + 1$

= $(x)(x^7 + x^6 + x^2 + x + 1)$

= $x^8 + x^7 + x^3 + x^2 + x$

= $x^8 + x^7 + x^3 + x^2 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 1 1000 1110

1 0001 1011 (XOR)

1001 0101 (**95**)

- {03.51}

  03 = 0000 0011 = x + 1

  51 = 0101 0001 = $x^6 + x^4 + 1$

  = $(x + 1)(x^6 + x^4 + 1)$

  = $(x^7 + x^5 + x) + (x^6 + x^4 + 1)$

  = $x^7 + x^6 + x^5 + x^4 + x + 1$

  Perhitungan : C_{31} = {01.5a} ⊕ {01.09} ⊕ {02.c7} ⊕ {03.51}

  C_{31} = {5a} ⊕ {09} ⊕ {95} ⊕ {f3} = 35

  Berikut ini merupakan perhitungan pencarian nilai untuk C_{41}:

02	03	01	01	X	5a
01	02	03	01		09
01	01	02	03		c7
03	01	01	02		51

  C_{41} = {03.5a} ⊕ {01.09} ⊕ {01.c7} ⊕ {02.51}

  - {03.5a}

    03 = 0000 0011 = x + 1

    5a = 0101 1010 = $x^6 + x^4 + x^3 + x$

    = $(x + 1)(x^6 + x^4 + x^3 + x)$

    = $(x^7 + x^5 + x^4 + x^2) + (x^6 + x^4 + x^3 + x)$

    = $x^7 + x^6 + x^5 + x^3 + x^2 + x$

    = 1110 1110 (**ee**)

  - {01.09} = 09
  - {01.c7} = c7
  - {02.51}

02 = 0000 0010 = x

51 = 0101 0001 = $x^6 + x^4 + 1$

= $(x)(x^6 + x^4 + 1)$

= $x^7 + x^5 + x$

= 1010 0010 (**a2**)

Perhitungan : C_{41} = {03.5a} ⊕ {01.09} ⊕ {01.c7} ⊕ {02.51

= {ee} ⊕ {09} ⊕ {c7} ⊕ {a2}

= **82**

Untuk mendapat hasil yang lainnya C_{12}, C_{22}, C_{32}, C_{42}, C_{13}, C_{23}, C_{33}…C_{44} lakukan perhitungan perkalian polynomial dengan cara seperti diatas. Berikut hasilnya :

C_{12} = {02.94} ⊕ {03.a9} ⊕ {01.42} ⊕ {01.57}

= {33} ⊕ {e0} ⊕ {42} ⊕ {57} = **c6**

C_{22} = {01.94} ⊕ {02.a9} ⊕ {03.42} ⊕ {01.57}

= {94} ⊕ {49} ⊕ {c6} ⊕ {57} = **4c**

C_{32} = {01.94} ⊕ {01.a9} ⊕ {02.42} ⊕ {03.57}

= {94} ⊕ {a9} ⊕ {84} ⊕ {f9} = **40**

C_{42} = {03.94} ⊕ {01.a9} ⊕ {01.42} ⊕ {02.57}

= {a7} ⊕ {a9} ⊕ {42} ⊕ {ae} = **e2**

C_{13} = {02.a6} ⊕ {03.cb} ⊕ {01.6e} ⊕ {01.64}

= {57} ⊕ {46} ⊕ {6e} ⊕ {64} = **1b**

C_{23} = {01.a6} ⊕ {02.cb} ⊕ {03.6e} ⊕ {01.64}

= {a6} ⊕ {8d} ⊕ {b2} ⊕ {64} = **fd**

C_{33} = {01.a6} ⊕ {01.cb} ⊕ {02.6e} ⊕ {03.64}

= {a6} ⊕ {cb} ⊕ {dc} ⊕ {ac} = **1d**

C_{43} = {03.a6} ⊕ {01.cb} ⊕ {01.6e} ⊕ {02.64}

= {f1} ⊕ {cb} ⊕ {6e} ⊕ {c8} = **9c**

C_{14} = {02.67} ⊕ {03.2b} ⊕ {01.d2} ⊕ {01.87}

= {ce} ⊕ {7d} ⊕ {d2} ⊕ {87} = **e6**

C_{24} = {01.67} ⊕ {02.2b} ⊕ {03.d2} ⊕ {01.87}

= {67} ⊕ {56} ⊕ {6d} ⊕ {87} = **db**

C_{34} = {01.67} ⊕ {01.2b} ⊕ {02.d2} ⊕ {03.87}

= {67} ⊕ {2b} ⊕ {bf} ⊕ {92} = **61**

C_{44} = {03.67} ⊕ {01.2b} ⊕ {01.d2} ⊕ {02.87}

= {a9} ⊕ {2b} ⊕ {d2} ⊕ {15} = **45**

Hasil dari proses MixColumn adalah sebagai berikut :

Hasil MixColumn			
39	c6	1b	e6
4b	4c	fd	db
35	40	1d	61
82	e2	9c	45

d. Proses *AddRoundKey*

Proses selanjutnya ialah *AddRoundKey*, setelah hasil yang didapat dalam proses *MixColumns* selanjutnya akan melakukan operasi XOR dengan *RoundKeys* 1 terhadap hasil dari *MixColumns*. Seperti gambar berikut ini :

Hasil MixColumn					KeyRound1					Hasil Ronde 1			
39	c6	1b	e6	XOR	98	2b	e7	6c	=	a1	ed	Fc	8a
4b	4c	fd	db		92	86	7a	3c		d9	ca	87	e7
35	40	1d	61		c0	bb	6f	92		f5	fb	72	f3
82	e2	9c	45		b4	f1	a1	82		36	13	3d	c7

Setelah mendapat hasi dari *AddRoundKey* ronde pertama, selanjutnya pencarian nilai ronde kedua hingga ronde ketiga belas dan melalui proses proses sebelumnya, yaitu proses *SubBytes*, proses *ShiftRows* dan proses *MixColumns* dan pencarian kunci di tiap rondenya untuk pemrosesan XOR dengan *MixColumns* (proses *AddRoundKey*) nanti. Yang kita ketahui kunci yang kita gunakan disini ialah "xyz" yang telah di proses dengan fungsi hash SHA-256 dan menghasilkan 32 bytes atau setara dengan 256 bits, yaitu sebagai berikut :

3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1
e77a6fa16c3c9282

Karena tiap ronde kunci aes hanya berisikan 16 bytes tiap rondenya, maka penulis akan membagi dua matriks kunci tersebut dan menaruhnya kedalam matriks 4x4 atau 16 bytes, seperti sebagai berikut ;

RoundKeys 0			
36	e4	d2	b0
08	4e	68	22
bc	a6	eb	60
a1	c4	6d	26

RoundKeys 1			
98	2b	e7	6c
92	86	7a	3c
c0	bb	6f	92
b4	f1	a1	82

Disebutkan sebelumnya, bahwa *RoundKeys* enkripsi blok pertama dan blok kedua adalah sama, sehingga kita tidak perlu mencari lagi RoundKeys 2 sampai 14, dikarenakan RoundKeys nya sama dengan pengenkripsian blok pertama seperti diatas.

Pada pengenkripsian blok kedua, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah di operasikan XOR oleh *RoundKeys* 2 sampai 14. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan terlalu panjangnya tahapan tahapannya. Berikut hasil nilai di setiap rondenya :

Hasil Ronde 2				Hasil Ronde 3				Hasil Ronde 4			
a2	a3	ad	d1	6d	72	a1	31	4a	cf	05	69
9b	60	ae	61	66	cc	7a	19	58	ac	3f	dc
38	5e	96	be	55	cc	8a	e5	29	46	9c	bf
04	da	1b	88	da	1d	f5	de	fe	ed	f5	5b
Hasil Ronde 5				Hasil Ronde 6				Hasil Ronde 7			
2c	fa	84	5e	f7	Ef	8b	46	b6	75	71	6e
7d	1f	75	c5	a1	1c	4b	5a	c6	97	ea	19
b6	4c	31	f1	c1	24	42	e3	9b	69	87	e3
3f	40	f6	a0	2f	41	23	24	0c	4f	e3	8e
Hasil Ronde 8				Hasil Ronde 9				Hasil Ronde 10			
49	d9	ac	8b	78	c2	6d	cf	ef	87	2a	d9
c1	7c	09	92	a5	4b	92	b9	7f	85	8b	e5
29	99	4c	a5	47	e5	cf	fe	27	bd	ef	20
c1	93	4b	2d	f6	3c	0a	6f	c3	2f	29	3a
Hasil Ronde 11				Hasil Ronde 12				Hasil Ronde 13			
3d	40	34	f4	f7	36	a0	7f	40	89	64	35
9f	00	22	7b	81	a5	3b	2b	6b	89	d0	42
aa	a4	36	e3	3b	18	4b	43	15	1b	6c	e7
e4	b6	b9	d7	0c	64	40	33	53	8a	86	87

Pada ronde terakhir, yaitu ronde ke 14, process MixColumn ditiadakan, atau dilewatkan. Hasil ronde ke 14 inilah yang menjadi hasil akhir dari pengenkripsian blok kedua sekaligus menjadi chippertext pada blok kedua dan menjadi akhir dari penjelasan proses pengenkripsian aes. Berikut ini hasil ronde 14 adalah:

Hasil Ronde 14			
62	3e	Cb	6d
3d	21	bd	5a
fc	23	0d	b4
88	5d	b7	5a

Pada rumus hasil yang telah disebutkan sebelumnya diatas bahwa :

Hasil = eblock1 || eblock2

Maka dari itu, chippertext dari pesan asli “Hello this is Secret Fichier!” dengan kunci “xyz” ialah ***13719c0f2348b02b 7fb6bff59b789555 623dfc883e21235d cbbd0db76d5ab45a***.

Berikut penulis tampilkan hasil enkripsi text tersebut dengan Secret Fichier, dan apabila di convert akan menghasilkan hexadecimal seperti diatas.

Gambar 4.2.3 Hasil Enkripsi

4.2.4 Dekripsi File

Sebuah file yang sudah di enkripsi tidak dapat dibaca dan dipahami oleh siapapun. Agar file tersebut dikembalikan seperti semula dan dapat terbaca lagi, maka diperlukan teknik dekripsi untuk mengembalikannya. Pada penjelasan sebelumnya, penulis telah menjelaskan tahapan tahapan algoritma SHA dan AES mode CBC untuk enkripsi file pada *Secret Fichier*. oleh karena itu, pada proses dekripsi file juga harus menggunakan algritma yang sama. Proses deskripsi pesan juga akan berkebalikan dari proses enkripsi, berkebalikan proses disebut sebagai inverse. Didalam pengenkripsian block tersebut, ada beberapa tahapan lagi untuk menghasilkan sebuah *plain text* kembali, yaitu pencarian inverse ronde kunci 2 hingga 14, pengdekripsian ronde pertama dengan tahapan *Inverse SubBytes*, *Inverse ShiftRows*, *Inverse MixColumns* dan *AddRoundKey*. Semua tahapan tersebut akan dijelaskan sebagai berikut.

Pada contoh pendekripsian kali ini, penulis akan menggunakan *chiper text* hasil dari pengenkripsian sebelumnya untuk menjelaskan tahap tahap algoritma AES mode CBC dan sha bekerja pada penerapan algoritma tersebut untuk penelitian pengamanan data penulis. Oleh karena siapkan *chiper text* dan kunci yang akan didekripsi kali ini, perlu diingat bahwa kunci yang digunakan harus sama seperti kunci yang digunakan pada proses enkripsi. Berikut *chiper text* dan kunci yang akan digunakan ialah :

Chiper text : 13719c0f2348b02b 7fb6bff59b789555
623dfc883e21235d cbbd0db76d5ab45a

Kunci : xyz

Tahapan selanjutnya ialah kunci yang kita gunakan disini harus telah di prosess terlebih dahulu dengan fungsi hash atau yang dimaksud dengan istilah *Key Derivation* (penurunan kunci) dengan SHA256. Pada proses sebelumnya, kunci "xyz" telah di proses dengan SHA-256 dan menghasilkan 64 hex digits atau setara dengan 32 bytes (256 bits) dengan kententuan SHA-2. Berikut hasil dari kunci setelah di proses dengan SHA-256.

Chipertext : 13 71 9c 0f 23 48 b0 2b 7f b6 bf f5 9b 78 95 55 62 3d fc 88 3e 21 23 5d cb bd 0d b7 6d 5a b4 5a

Kunci : 36 08 bc a1 e4 4e a6 c4 d2 68 eb 6d b0 22 60 26 98 92 c0 b4 2b 86 bb f1 e7 7a 6f a1 6c 3c 92 82

Dikarenakan AES hanya memiliki panjang block dengan ukuran 128 bit sedangkan *chiper text* yang penulis gunakan disini ialah *chiper text* dengan ukuran 32 bytes (32 x 8 = 256 bit) karena adanya padding, maka dari itu *chiper text* akan di bagi menjadi 2 block yang masing masing block tersebut mempiliki panjang 128 bit. Berikut gambaran pembagian block :

13 71 9c 0f 23 48 b0 2b 7f b6 bf f5 9b 78 95 55 |Chipertext block 1|

62 3d fc 88 3e 21 23 5d cb bd 0d b7 6d 5a b4 5a |Chipertext block 2|

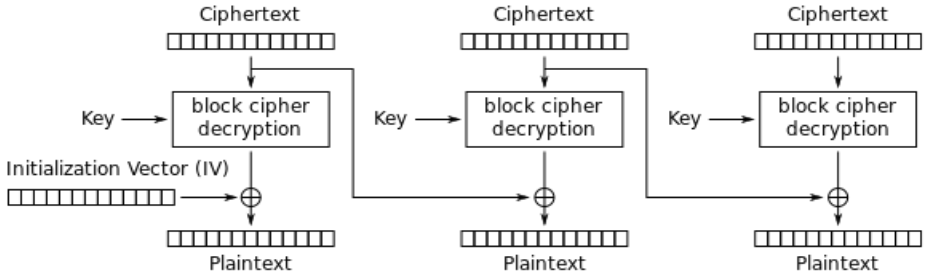
Agar lebih mudah penggambaran pada masing masing block, maka penulis akan membuat matriks dengan ukuran 4 x 4 untuk diisi dengan masing masing nilai heksadesimal kedalam block block yang tadi di gambarkan.

Chipertext Block 1			
13	23	7f	9b
71	48	b6	78
9c	b0	bf	95
0f	2b	f5	55

Chipertext Block 2			
62	3e	cb	6d
3d	21	bd	5a
fc	23	0d	b4
88	5d	b7	5a

Dalam pemrosesan AES mode CBC, rangkaian bit-bit pada *chipertext* dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (*initialization vector*) untuk menggabungkan dengan

chipertext pada tahapan akhir pengdekripsian *chipertext* aes. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini.

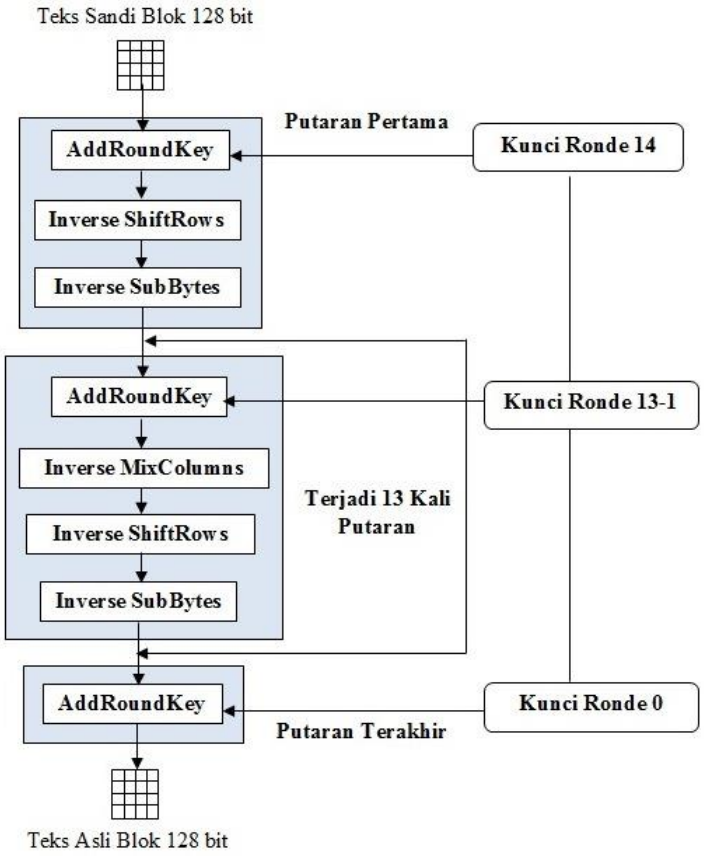


Cipher Block Chaining (CBC) mode decryption

Pada pemrosesan dekripsi CBC, diperlukannya penambahan IV atau yang disebut sebagai IV (*initialization vector*) pada tahap akhir *chipertext* dihasilkan. Dalam pemrosesan dengan algoritma aes sudah disebutkan sebelumnya bahwa ada dua input yang dibutuhkan, yaitu input yang pertama ialah *chipertext* yang akan di proses pendekripsiannya, yang kedua ialah kunci yang telah di proses dengan fungsi hash sebagai aeskey pada aes itu sendiri. Pada penjelasan selanjutnya, penulis akan menjelaskan tahapan pendekripsian block pertama dengan AES mode CBC.

4.2.5 Dekripsi Blok Pertama

Seperti gambar dekripsi aes mode cbc diatas, pada pemrosesan *chipertext* block pertama apabila block pertama sudah di proses, maka hasil dari *chipertext* akhir ronde akan dioperasikan XOR pada IV pertama, yaitu bernilai “00” sebanyak 16 hex digit dan akan menghasilkan *plaintext* block pertama. Selanjutnya pada pedekripsian block kedua, chipertext block pertama akan menjadi IV block kedua yang akan dioperasikan XOR pada hasil akhir ronde cipertext block kedua dan menghasilkan plaintext block kedua. Sebelum tahapan operasi XOR IV, chipertext akan melalui tahapan *AddRoundKey, Inverse MixColumn, Inverse ShiftRows, Inverse SubBytes* sebanyak 14 kali putaran sebagaimana dari gambaran proses dekripsi AES dibawah ini :



setelah mendapat chipertext block pertama dan kuncinya, maka kita akan menaruh cipertext block pertama tersebut kedalam matriks 4 x 4 agar lebih mudah dalam pemrosesan dekripsinya. Untuk *RoundKeys Schedule* sudah didapat pada proses sebelumnya (pada tahapan enkripsi file hal. 61) yaitu *RoundKeys* 1 sampai 14 akan digunakan Kembali pada proses ini. Pada proses dekripsi ini penulis memakai kunci yang sama yaitu kunci "xyz" yang sudah dilakukan proses penjadwalan *RoundKeys*. Berikut matriks *chipertext* block pertama dan keseluruhan hasil *RoundKeys Schedule* yang telah didapat (karena pada proses *inverse* (terbalik) ini, penulis akan menggunakan RoundKeys 14 terlebih dahulu lalu RoundKeys 0 di akhir tahapan) :

Chipertext Block 1			
13	23	7f	9b
71	48	b6	78
9c	b0	bf	95
0f	2b	f5	55

RoundKeys 14				RoundKeys 13				RoundKeys 12			
6b	99	88	fb	ea	5a	16	3a	17	f2	11	73
9a	51	91	25	02	83	b7	6d	61	cb	c0	b4
ac	b7	54	1b	58	d1	67	63	67	1b	e3	4f
9f	b0	c9	1e	c3	9a	28	59	1f	2f	79	d7
RoundKeys 11				RoundKeys 10				RoundKeys 9			
65	b0	4c	2c	60	e5	e3	62	cf	d5	fc	60
8f	81	34	da	93	aa	0b	74	1d	0e	b5	ee
dc	89	b6	04	c4	7c	f8	ac	4d	55	3f	b2
cd	59	b2	71	6e	30	56	ae	29	94	eb	c3
RoundKeys 8				RoundKeys 7				RoundKeys 6			
58	85	06	81	c3	1a	29	9c	69	dd	83	87
a4	39	a1	7f	cf	13	bb	5b	f9	9d	98	de
ea	b8	84	54	6d	18	6a	8d	de	52	3c	d0
be	5e	66	f8	68	bd	7f	28	60	e0	38	9e
RoundKeys 5				RoundKeys 4				RoundKeys 3			
d4	d9	33	b5	8c	b4	5e	04	26	0d	ea	86
d2	dc	a8	e0	6d	64	05	46	88	0e	74	48
1d	75	72	e7	85	8c	6e	ec	d3	68	07	95
63	d5	c2	57	b5	80	d8	a6	47	b6	17	95
RoundKeys 2				RoundKeys 1				RoundKeys 0			
dc	38	ea	5a	98	2b	e7	6c	36	e4	d2	b0
47	09	61	43	92	86	7a	3c	08	4e	68	22
af	09	e2	82	c0	bb	6f	92	bc	a6	eb	60
f1	35	58	7e	b4	f1	a1	82	a1	c4	6d	26

Seperti gambar tahapan dekripsi aes diatas, tahapan pertama yang akan dilakukan ialah proses ronde pertama yang diawali dengan proses *AddRoundKey* kunci ronde 14. Pada proses dekripsi *AddRoundKey* dimulai dari ronde terakhir pada proses enkripsi karena kebalikannya, yaitu dimulai dari ronde terakhir ronde 14 dengan *RoundKeys* terakhir 14 sampai ke ronde 1 dengan *RoundKeys* 0. Pada tahap *AddRoundKey* terjadi perhitungan XOR

antara chipertext tiap blocknya dengan RoundKeys. XOR merupakan kepanjangan dari *Exclusive OR* yang mana keluarannya akan berlogika 1 apabila inputannya berbeda, namun apabila semua inputanya sama maka akan memberikan keluarannya 0. Proses XOR dilakukan pada masing masing blok pada matriks, seperti matriks *chipertext* block 1 kolom [1,1] XOR dengan matriks *RoundKeys* 14 kolom [1,1] dan seterusnya. Berikut akan disimulasikan XOR pada matriks *chipertext* blok 1 dengan matriks *RoundKeys* 14.

Untuk dapat melakukan operasi XOR, nilai heksadecimal yang ada di *chipertext* dan di kunci tadi harus diubah menjadi nilai biner berukuran 8 bit terlebih dahulu. Seperti contoh berikut :

13 = 0001 0011

6b = 0110 1011

Proses XOR disimulasikan seperti dibawah ini :

0	0	0	1	0	0	1	1	
0	1	1	0	1	0	1	1	⊕
0	**1**	**1**	**1**	**1**	**0**	**0**	**0**	

Nilai heksadesimal dari 0111 1000 ialah 78, jadi 13 ⊕ 6b = 78

Hitung keseluruhan nilai XOR di kolom kolom matriks chipertext blok 1 dengan RoundKeys 14 hingga mengahasilkan blok matriks berukuran 4 x 4.

71 ⊕ 9a = eb

9c ⊕ ac = 30

0f ⊕ 9f = 90

23 ⊕ 99 = ba

48 ⊕ 51 = 19

b0 ⊕ b7 = 07

2b ⊕ b0 = 9b

7f ⊕ 88 = f7

b6 ⊕ 91 = 27

bf ⊕ 54 = eb

f5 ⊕ c9 = 3c

9b ⊕ fb = 60

78 ⊕ 25 = 5d

95 ⊕ 1b = 8e

55 ⊕ 1e = 4b

Berikut matriks dari chipertext block 1 XOR RoundKeys 14

Chipertext block 1					RoundKeys 14					Hasil			
13	23	7f	9b		6b	99	88	fb		**78**	ba	f7	60
71	48	b6	78	⊕	9a	51	91	25	=	**eb**	19	27	5d
9c	b0	bf	95		ac	b7	54	1b		**30**	07	eb	8e
0f	2b	f5	55		9f	b0	c9	1e		**90**	9b	3c	4b

Pada tahap selanjutnya dilakukan proses *Inverse ShiftRows, Inverse SubByte* pada putaran ke 14 (tahapan *MixColumns* dilewatkan). Lalu pindah ke putaran 13 ada tahap *AddRoundKeys* dengan *RoundKeys* 13 lalu ada *Inverse MixColumns, Inverse ShiftRows* dan *Inverse SubBytes* sampai seterusnya dan di ulang ulang sampai ke putaran 1 dan mendapatkan hasil plaintextnya. Berikut penjelasan tahapan selanjutnya

a. Proses *Inverse ShiftRows*

Hasil dari proses *AddRoundKeys* tadi akan di geser beberapa bit di dalam proses ini proses ini berkebalikan dari proses *ShiftRows*, berikut gambaran yang terjadi pada proses *Inverse ShiftRows* ini :

Pada nilai yang ditandai akan di putar kedepan tiga bit, seperti berikut ini

78	ba	f7	60		78	ba	f7	60
eb	19	27	5d	➡	5d	eb	19	27
30	07	eb	8e		30	07	eb	8e
90	9b	3c	4b		90	9b	3c	4b

Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut

78	ba	f7	60		78	ba	f7	60
5d	eb	19	27		5d	eb	19	27
30	07	eb	8e	→	eb	8e	30	07
90	9b	3c	4b		90	9b	3c	4b

Pada nilai yang ditandai akan diputar lebih dari satu byte, seperti berikut

78	ba	f7	60		78	ba	f7	60
5d	eb	19	27		5d	eb	19	27
eb	8e	30	07		eb	8e	30	07
90	9b	3c	4b	→	9b	3c	4b	90

Hasil dari Inverse ShiftRows ialah sebagai berikut :

Hasil *Inverse ShiftRows*			
78	ba	f7	60
5d	eb	19	27
eb	8e	30	07
9b	3c	4b	90

b. Proses *Inverse SubBytes*

Pada proses *Inverse SubBytes* ini, hasil *Inverse ShiftRows* akan di subtitusikan dengan nilai yang ada di table *Inverse S-Box* (tabel 2.9.3 hlm.19). proses dari *Inverse SubBytes* ialah sebagai berikut :

78	ba	f7	60
5d	eb	19	27
eb	8e	30	07
9b	3c	4b	90

Lakukan proses subtitusi Inverse S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (7) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (8) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil Inverse SubByte. Seperti contoh gambar berikut :

	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Setelah dilakukan subtitusi Inverse S-box pada proses *Inverse SubBytes* pada nilai 78, maka didapatkan hasil yaitu c1. Lalu lanjutkan subtitusi Inverse S-box pada kolom [2.1] matriks diatas yaitu nilai 5d.

	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Pada proses Subtitusi Inverse S-box pada nilai 5d, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi Inverse S-box proses Inverse SubBytes 5d mendapatkan hasil nilai 8d. Lalu lanjutkan subtitusi Inverse S-box pada kolom [3.1] matriks diatas yaitu nilai eb.

	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

x

Pada proses Subtitusi Inverse S-box pada nilai eb, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi Inverse S-box proses *Inverse SubBytes* eb mendapatkan hasil nilai 3c. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai 9b.

	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

x

Pada proses Subtitusi Inverse S-box pada nilai 9b, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi Inverse S-box proses *Inverse SubBytes* 9b mendapatkan hasil nilai e8. Lalu lanjutkan subtitusi Inverse S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table Inverse S-Box proses *Inverse SubBytes*. Setelah dilakukan proses *Inverse SubBytes* pada keseluruhan nilai, maka didapatkan hasil dari proses Inverse SubBytes ialah sebagai berikut :

Hasil Inverse SubBytes			
c1	c0	26	90
8d	3c	8e	3d
3c	e6	08	38
e8	6d	cc	96

Hasil Ronde 14			
c1	c0	26	90
8d	3c	8e	3d
3c	e6	08	38
e8	6d	cc	96

Setelah mendapat hasil dari *Inverse SubBytes* sekaligues menjadi Hasil Ronde ke 14. Selanjutnya adalah tahap dimana proses ini melakukan 13 kali perulangan sesuai dengan proses dekripsi AES-256 (Gambar 2.9 hlm.20). Tahapan selanjutnya ialah putaran selanjutnya mencari nilai dari hasil ronde ke 13 dan seterusnya sampai ke ronde 1.

a. Proses *AddRoundKey* (Ronde 13)

Pada tahapan ini sama seperti tahapan AddRoundKey diatas, hasil dari ronde 14 dilakukan operasi XOR dengan RoundKeys 13. Seperti gambar berikut ini :

Hasil Ronde 14					RoundKeys 13					Hasil			
c1	c0	26	90		**ea**	5a	16	3a		2b	9a	30	aa
8d	3c	8e	3d	⊕	**02**	83	b7	6d	=	8f	bf	39	50
3c	e6	08	38		**58**	d1	67	63		64	37	6f	5b
e8	6d	cc	96		**c3**	9a	28	59		2b	f7	e4	cf

a. Proses *Inverse MixColumns* (Ronde 13)

Proses selanjutnya adalah proses Inverse MixColumns, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada proses Inverse ShiftRows tadi dengan matriks yang telah di tetapkan oleh Rijndael. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial. Berikut merupakan proses dari Inverse MixColumns :

0e	0b	0d	09	X	2b	9a	30	aa	➡		C_{11}	C_{12}	C_{13}	C_{14}
09	0e	0b	0d		8f	bf	39	50			C_{21}	C_{22}	C_{23}	C_{24}
0d	09	0e	0b		64	37	6f	5b			C_{31}	C_{32}	C_{33}	C_{34}
0b	0d	09	0e		2b	f7	e4	cf			C_{41}	C_{42}	C_{43}	C_{44}

Berikut ini merupakan perhitungan pencarian nilai untuk C_{11} :

0e	0b	0d	09	X	2b
09	0e	0b	0d		8f
0d	09	0e	0b		64
0b	0d	09	0e		2b

C_{11} = {0e.2b} ⊕ {0b.8f} ⊕ {0d.64} ⊕ {09.2b}

- {0e.2b}

  0e = 0000 1110 $= x^3 + x^2 + x$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + x^2 + x)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^7 + x^5 + x^3 + x^2) + (x^6 + x^4 + x^2 + x)$

  $= x^8 + x^7 + x^5 + x$

  $= x^8 + x^7 + x^5 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 1 1010 0010

  1 0001 1011 (XOR)

  1011 1001 (**b9**)

- {0b.8f}

  0b = 0000 1011 $= x^3 + x + 1$

  8f = 1000 1111 $= x^7 + x^3 + x^2 + x + 1$

  $= (x^3 + x + 1)(x^7 + x^3 + x^2 + x + 1)$

  $= (x^{10} + x^6 + x^5 + x^4 + x^3) + (x^8 + x^4 + x^3 + x^2 + x) + (x^7 + x^3 + x^2 + x + 1)$

  $= x^{10} + x^8 + x^7 + x^6 + x^5 + 1$

  $= x^{10} + x^8 + x^7 + x^6 + x^5 + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 101 1110 0001

  100 0110 11 (XOR)

  1 1000 1101

  1 0001 1011

  1001 1110 (**9e**)

- {0d.64}

  0d = 0000 1101 $= x^3 + x^2 + 1$

  64 = 0110 0100 $= x^6 + x^5 + x^2$

$= (x^3 + x^2 + 1)(x^6 + x^5 + x^2)$

$= (x^9 + x^8 + x^5) + (x^8 + x^7 + x^4) + (x^6 + x^5 + x^2)$

$= x^9 + x^7 + x^6 + x^4 + x^2$

$=$ $x^9 + x^7 + x^6 + x^4 + x^2$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

$=$ 1011 0101 00

1000 1101 1 (XOR)

1110 0010 (**e2**)

- {09.2b}

  09 = 0000 1001 $= x^3 + 1$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + 1)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^5 + x^3 + x + 1)$

  $= x^8 + x^6 + x^5 + x^4 + x + 1$

  $=$ $x^8 + x^6 + x^5 + x^4 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  $=$ 1 0111 0011

  1 0001 1011 (XOR)

  0110 1000 (**68**)

  C_{11} = {b9} ⊕ {9e} ⊕ {e2} ⊕ {68} = ad

  Berikut ini merupakan perhitungan pencarian nilai untuk C_{21} :

0e	0b	0d	09	X	2b
09	0e	0b	0d		8f
0d	09	0e	0b		64
0b	0d	09	0e		2b

  C_{21} = {09.2b} ⊕ {0e.8f} ⊕ {0b.64} ⊕ {0d.2b}

- {09.2b}

  09 = 0000 1001 $= x^3 + 1$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + 1)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^5 + x^3 + x + 1)$

  $= x^8 + x^6 + x^5 + x^4 + x + 1$

  $=$ $x^8 + x^6 + x^5 + x^4 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 1 0111 0011

1 0001 1011 (XOR)

0110 1000 (**68**)

- {0e.8f}

  0e = 0000 1110 = $x^3 + x^2 + x$

  8f = 1000 1111 = $x^7 + x^3 + x^2 + x + 1$

  = $(x^3 + x^2 + x)(x^7 + x^3 + x^2 + x + 1)$

  = $(x^{10} + x^6 + x^5 + x^4 + x^3) + (x^9 + x^5 + x^4 + x^3 + x^2) + (x^8 + x^4 + x^3 + x^2 + x)$

  = $x^{10} + x^9 + x^8 + x^6 + x$

  = $x^{10} + x^9 + x^8 + x^6 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 111 0100 0010

  100 0110 11(XOR)

  11 0010 1110

  10 0011 011

  1 0000 1000

  1 0001 1011

  0001 1011 (**1b**)

- {0b.64}

  0b = 0000 1011 = $x^3 + x + 1$

  64 = 0110 0100 = $x^6 + x^5 + x^2$

  = $(x^3 + x + 1)(x^6 + x^5 + x^2)$

  = $(x^9 + x^8 + x^5) + (x^7 + x^6 + x^3) + (x^6 + x^5 + x^2)$

  = $x^9 + x^8 + x^7 + x^3 + x^2$

  = $x^9 + x^8 + x^7 + x^3 + x^2$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 11 1000 1100

  10 0011 011(XOR)

  1 1011 1010

  1 0001 1011

  1010 0001 (**a1**)

- {0d.2b}

0d = 0000 1101 $= x^3 + x^2 + 1$

2b = 0010 1011 $= x^5 + x^3 + x + 1$

$= (x^3 + x^2 + 1)(x^5 + x^3 + x + 1)$

$= (x^8 + x^6 + x^4 + x^3) + (x^7 + x^5 + x^3 + x^2) + (x^5 + x^3 + x + 1)$

$= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$

$= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 1 1101 0111

1 0001 1011 (XOR)

1100 0100 (**c4**)

C_{21} = {68} ⊕ {1b} ⊕ {a1} ⊕ {c4} = 16

Berikut ini merupakan perhitungan pencarian nilai untuk C_{31} :

0e	0b	0d	09	X	2b
09	0e	0b	0d		8f
0d	09	0e	0b		64
0b	0d	09	0e		2b

C_{31} = {0d.2b} ⊕ {09.8f} ⊕ {0e.64} ⊕ {0b.2b}

- {0d.2b}

  0d = 0000 1101 $= x^3 + x^2 + 1$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + x^2 + 1)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^7 + x^5 + x^3 + x^2) + (x^5 + x^3 + x + 1)$

  $= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$

  $= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  = 1 1101 0111

  1 0001 1011 (XOR)

  1100 0100 (**c4**)

- {09.8f}

  09 = 0000 1001 $= x^3 + 1$

  8f = 1000 1111 $= x^7 + x^3 + x^2 + x + 1$

  $= (x^3 + 1)(x^7 + x^3 + x^2 + x + 1)$

$= (x^{10} + x^6 + x^5 + x^4 + x^3) + (x^7 + x^3 + x^2 + x + 1)$

$= x^{10} + x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$

$=$ $x^{10} + x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

$=$ 100 1111 0111

100 0110 11 (XOR)

1001 1011 (**9b**)

- {0e.64}

  0e = 0000 1110 $= x^3 + x^2 + x$

  64 = 0110 0100 $= x^6 + x^5 + x^2$

  $= (x^3 + x^2 + x)(x^6 + x^5 + x^2)$

  $= (x^9 + x^8 + x^5) + (x^8 + x^7 + x^4) + (x^7 + x^6 + x^3)$

  $= x^9 + x^6 + x^5 + x^4 + x^3$

  $=$ $x^9 + x^6 + x^5 + x^4 + x^3$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  $=$ 10 0111 1000

  10 0011 011 (XOR)

  0100 1110 (**4e**)

- {0b.2b}

  0b = 0000 1011 $= x^3 + x + 1$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + x + 1)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^6 + x^4 + x^2 + x) + (x^5 + x^3 + x + 1)$

  $= x^8 + x^5 + x^2 + 1$

  $=$ $x^8 + x^5 + x^2 + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  $=$ 1 0010 0101

  1 0001 1011 (XOR)

  0011 1110 (**3e**)

C_{31} = {c4} ⊕ {9b} ⊕ {4e} ⊕ {3e} = 2f

Berikut ini merupakan perhitungan pencarian nilai untuk C_{41} :

0e	0b	0d	09	X	2b
09	0e	0b	0d		8f
0d	09	0e	0b		64
0b	0d	09	0e		2b

C_{41} = {0b.2b} ⊕ {0d.8f} ⊕ {09.64} ⊕ {0e.2b}

- {0b.2b}

  0b = 0000 1011 $= x^3 + x + 1$

  2b = 0010 1011 $= x^5 + x^3 + x + 1$

  $= (x^3 + x + 1)(x^5 + x^3 + x + 1)$

  $= (x^8 + x^6 + x^4 + x^3) + (x^6 + x^4 + x^2 + x) + (x^5 + x^3 + x + 1)$

  $= x^8 + x^5 + x^2 + 1$

  $=$ $x^8 + x^5 + x^2 + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  $=$ 1 0010 0101

  1 0001 1011 (XOR)

  0011 1110 (**3e**)

- {0d.8f}

  0d = 0000 1101 $= x^3 + x^2 + 1$

  8f = 1000 1111 $= x^7 + x^3 + x^2 + x + 1$

  $= (x^3 + x^2 + 1)(x^7 + x^3 + x^2 + x + 1)$

  $= (x^{10} + x^6 + x^5 + x^4 + x^3) + (x^9 + x^5 + x^4 + x^3 + x^2) + (x^7 + x^3 + x^2 + x + 1$

  $= x^{10} + x^9 + x^7 + x^6 + x + 1$

  $=$ $x^{10} + x^9 + x^7 + x^6 + x + 1$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

  $=$ 110 1100 0011

  100 0110 11 (XOR)

  10 1010 1111

  10 0011 0001

  1001 0001 **(91)**

- {09.64}

  09 = 0000 1001 $= x^3 + 1$

64 = 0110 0100 $= x^6 + x^5 + x^2$

$= (x^3 + 1)(x^6 + x^5 + x^2)$

$= (x^9 + x^8 + x^5) + (x^6 + x^5 + x^2)$

$= x^9 + x^8 + x^6 + x^2$

$= x^9 + x^8 + x^6 + x^2$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 11 0100 0100

10 0011 011 (XOR)

1 0111 0010

1 0001 1011

0110 1001 (**69**)

- {0e.2b}

0e = 0000 1110 $= x^3 + x^2 + x$

2b = 0010 1011 $= x^5 + x^3 + x + 1$

$= (x^3 + x^2 + x)(x^5 + x^3 + x + 1)$

$= (x^8 + x^6 + x^4 + x^3) + (x^7 + x^5 + x^3 + x^2) + (x^6 + x^4 + x^2 + x)$

$= x^8 + x^7 + x^5 + x$

$= x^8 + x^7 + x^5 + x$ modulo $x^8 + x^4 + x^3 + x^1 + 1$

= 1 1010 0010

1 0001 1011 (XOR)

1011 1001 (**b9**)

C_{41} = {3e} ⊕ {91} ⊕ {69} ⊕ {b9} = 7f

Untuk mendapat hasil yang lainnya C_{12}, C_{22}, C_{32}, C_{42}, C_{13}, C_{23}, C_{33}…C_{44} lakukan perhitungan perkalian polynomial dengan cara seperti diatas. Berikut hasilnya :

C_{12} = {0e.9a} ⊕ {0b.bf} ⊕ {0d.37} ⊕ {09.f7}

= {cd} ⊕ {55} ⊕ {48} ⊕ {0e} = **de**

C_{22} = {09.9a} ⊕ {0e.bf} ⊕ {0b.37} ⊕ {0d.f7}

= {26} ⊕ {20} ⊕ {fa} ⊕ {ff} = **03**

C_{32} = {0d.9a} ⊕ {09.bf} ⊕ {0e.37} ⊕ {0b.f7}

= {78} ⊕ {30} ⊕ {11} ⊕ {fb} = **a2**

C_{42} = {0b.9a} ⊕ {0d.bf} ⊕ {09.37} ⊕ {0e.f7}

= {09} ⊕ {fa} ⊕ {94} ⊕ {fd} = **9a**

C_{13} = {0e.30} ⊕ {0b.39} ⊕ {0d.6f} ⊕ {09.e4}

= {3b} ⊕ {98} ⊕ {9d} ⊕ {85} = **bb**

C_{23} = {09.30} ⊕ {0e.39} ⊕ {0b.6f} ⊕ {0d.e4}

= {ab} ⊕ {45} ⊕ {e4} ⊕ {48} = **32**

C_{33} = {0d.30} ⊕ {09.39} ⊕ {0e.6f} ⊕ {0b.e4}

= {6b} ⊕ {ea} ⊕ {2c} ⊕ {56} = **fb**

C_{43} = {0b.30} ⊕ {0d.39} ⊕ {09.6f} ⊕ {0e.e4}

= {cb} ⊕ {0e} ⊕ {3a} ⊕ {0f} = **f0**

C_{14} = {0e.aa} ⊕ {0b.50} ⊕ {0d.5b} ⊕ {09.cf}

= {f6} ⊕ {46} ⊕ {c2} ⊕ {ed} = **9f**

C_{24} = {09.aa} ⊕ {0e.50} ⊕ {0b.5b} ⊕ {0d.cf}

= {8d} ⊕ {4d} ⊕ {03} ⊕ {fc} = **3f**

C_{34} = {0d.aa} ⊕ {09.50} ⊕ {0e.5b} ⊕ {0b.cf}

= {13} ⊕ {e6} ⊕ {2f} ⊕ {68} = **b2**

C_{44} = {0b.aa} ⊕ {0d.50} ⊕ {09.5b} ⊕ {0e.cf}

= {c2} ⊕ {bd} ⊕ {b5} ⊕ {b6} = **7c**

Hasil dari proses Inverse MixColumn adalah sebagai berikut :

Hasil Inverse MixColumns			
ad	de	Bb	9f
16	03	32	3f
2f	a2	fb	b2
7f	9a	f0	7c

b. Proses *Inverse ShiftRows* (Ronde 13)

Berikut ini merupakan hasil proses dari Inverse Shiftrows terhadap hasil dari Inverse MixColumns.

Hasil Inverse ShiftRows			
ad	de	bb	9f
3f	16	03	32
fb	b2	2f	a2
9a	f0	7c	7f

c. Proses *Inverse SubBytes* (Ronde 13)

Berikut ini merupakan hasil proses dari *Inverse SubBytes* terhadap hasil dari Inverse ShiftRows.

Hasil Inverse SubBytes			
18	9c	Fe	6e
25	ff	d5	a1
63	3e	4e	1a
37	17	01	6b

Setelah tahapan tahapan tersebut dilakukan, maka mendapatkan hasil dari ronde 13. Berikut hasil dari ronde 13 :

Hasil Ronde 13			
18	9c	Fe	6e
25	ff	d5	a1
63	3e	4e	1a
37	17	01	6b

Pada pendekripsian blok pertama, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah melalui proses inverse sebelumnya. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan mempersingkat penjelasan terhadap tahapan tahapannya. Berikut hasil nilai di setiap rondenya :

Hasil Ronde 12				Hasil Ronde 11				Hasil Ronde 10			
31	1d	84	df	07	2c	69	7a	1f	b3	d3	01
1b	de	06	75	27	8f	fl	93	62	a5	c8	d2
64	fb	6a	d8	44	7d	0c	90	c0	2a	16	19
53	be	ef	f4	0d	68	83	be	ae	76	a0	56
Hasil Ronde 9				Hasil Ronde 8				Hasil Ronde 7			
52	bc	76	97	e1	45	da	fe	79	31	1e	f3
80	20	dd	c5	8d	bf	ea	8a	d3	7c	78	22
9d	4f	81	93	b1	df	85	ea	9c	d8	b5	41
ee	17	1a	e9	17	e7	6a	45	35	0a	4b	fc
Hasil Ronde 6				Hasil Ronde 5				Hasil Ronde 4			
e4	8d	ae	d4	85	4d	97	59	03	8f	a6	64
f5	64	66	e8	3c	3f	7a	83	b2	1e	d0	fa
2a	57	d8	e0	e9	ab	90	30	d6	d9	9c	be
3b	de	e6	d3	e1	89	6f	14	7b	52	20	20
Hasil Ronde 3				Hasil Ronde 2				Hasil Ronde 1			
35	19	1d	1b	da	c5	50	33	48	6f	69	73
00	70	e3	04	7f	1d	7f	5c	65	20	73	20
63	e0	14	1e	bc	68	ba	fc	6c	74	20	53
d0	eb	4d	44	0e	bb	ba	86	6c	68	69	65

Hasil ronde ke 11 inilah yang menjadi hasil akhir dari pendekripsian blok pertama sekaligus menjadi plaintext pada blok pertama. Berikut hasil plaintext block 1 : ***48656c6c6f207468 6973206973205365***

4.2.6 Dekripsi Blok Kedua

Pada pemrosesan chipertext block kedua, diperlukan penambahan IV diakhir tahapan *AddRoundKey*, IV yang digunakan ialah hasil dari chippertext block pertama. Seperti gambaran dekripsi aes mode cbc diatas.

Untuk *RoundKeys* yang digunakan sama seperti *RoundKeys* yang digunakan pada dekripsi chipertext block pertama, tahapan tahapannya pun sama. Untuk mempersingkat penjelasan penulis akan mempersingkat tahapannya dan menuliskan hasil akhirnya saja seperti dibawah ini :

Chipertext Block 2			
62	3e	cb	6d
3d	21	bd	5a
fc	23	0d	b4
88	5d	b7	5a

a. Proses *AddRoundKey*

Chipertext block 2					RoundKeys 14			
62	3e	cb	6d		6b	99	88	fb
3d	21	bd	5a	⊕	9a	51	91	25
fc	23	0d	b4		ac	b7	54	1b
88	5d	b7	5a		9f	b0	c9	1e

Hasil			
09	a7	43	96
a7	70	2c	7f
50	94	59	af
17	ed	7e	44

b. Proses *Inverse ShiftRows*

Hasil dari proses *AddRoundKeys* tadi akan di geser beberapa bit di dalam proses ini proses ini berkebalikan dari proses ShiftRows. Berikut Hasil dari proses Inverse ShiftRows :

Hasil *Inverse ShiftRows*			
09	a7	43	96
7f	a7	70	2c
59	af	50	94
ed	7e	44	17

c. Proses *Inverse SubBytes*

Hasil *Inverse SubBytes*			
40	89	64	35
6b	89	d0	42
15	1b	6c	e7
53	8a	86	87

Hasil Ronde 14			
40	89	64	35
6b	89	d0	42
15	1b	6c	e7
53	8a	86	87

Hasil dari *Inverse SubBytes* diatas sekaligus hasil dari hasil ronde ke 14. Selanjutnya adalah tahap dimana proses ini melakukan 13 kali perulangan sesuai dengan proses dekripsi AES-256 (Gambar 2.9 hlm.20). Tahapan selanjutnya ialah putaran selanjutnya mencari nilai dari hasil ronde ke 13 dan seterusnya sampai ke ronde 1. Untuk mempersingkat penjelasan tahapan dikarenakan terlalu panjangnya penjelasannya, maka dari itu penulis hanya mencantumkan hasil ronde dari tiap tahapannya sebagai berikut ini :

Hasil Ronde 13				Hasil Ronde 12				Hasil Ronde 11			
f7	36	a0	7f	3d	40	34	f4	ef	87	2a	d9
81	a5	3b	2b	9f	00	22	7b	7f	85	8b	e5
3b	18	4b	43	aa	a4	36	e3	27	bd	ef	20
0c	64	40	33	e4	b6	b9	d7	c3	2f	29	3a
Hasil Ronde 10				Hasil Ronde 9				Hasil Ronde 8			
78	c2	6d	cf	49	d9	ac	8b	b6	75	71	6e
a5	4b	92	b9	c1	7c	09	92	c6	97	ea	19
47	e5	cf	fe	29	99	4c	a5	9b	69	87	e3
f6	3c	0a	6f	c1	93	4b	2d	0c	4f	e3	8e
Hasil Ronde 7				Hasil Ronde 6				Hasil Ronde 5			
f7	ef	8b	46	2c	fa	84	5e	4a	cf	05	69
a1	1c	4b	5a	7d	1f	75	c5	58	ac	3f	dc
c1	24	42	e3	b6	4c	31	f1	29	46	9c	bf
2f	41	23	24	3f	40	f6	a0	fe	ed	f5	5b
Hasil Ronde 4				Hasil Ronde 3				Hasil Ronde 2			
6d	72	a1	31	a2	a3	ad	d1	a1	ed	Fc	8a
66	cc	7a	19	9b	60	ae	61	d9	ca	87	e7
55	cc	8a	e5	38	5e	96	be	f5	fb	72	f3
da	1d	f5	de	04	da	1b	88	36	13	3d	c7

Pada ronde terakhir, yaitu ronde ke 1, Hasil Inverse SubBytes ronde ke 1 akan dilakukan operasi XOR dengan RoundKeys 1 dan XOR dengan

IV, yaitu chipertext block pertama. Hasil ronde ke 1 inilah yang menjadi hasil akhir dari pendekripsian blok kedua sekaligus menjadi plaintext pada blok kedua dan menjadi akhir dari penjelasan proses pendekripsian aes. Berikut ini hasil ronde 14 adalah :

Inverse SubBytes 1					RoundKeys 0					IV			
46	e7	c5	0a		**36**	e4	d2	b0		**13**	23	7f	9b
0b	40	b7	59	⊕	**08**	4e	68	22	⊕	**71**	48	b6	78
45	7f	31	f6		**bc**	a6	eb	60		**9c**	b0	bf	95
da	8c	ea	70		**a1**	c4	6d	26		**0f**	2b	f5	55

Hasil Ronde 1			
63	20	68	21
72	46	69	03
65	69	65	03
74	63	72	03

Pada rumus hasil yang telah disebutkan sebelumnya diatas bahwa :

Hasil = deblock1 || deblock2

Maka dari itu, plaintext dari chipertext "13719c0f2348b02b 7fb6bff59b789555 623dfc883e21235d cbbd0db76d5ab45a" dengan kunci "xyz" ialah ***48656c6c6f207468 6973206973205365 6372657420466963 6869657221030303***.

Apabila di konversikan kedalam huruf alphabeth dari heksadecimal tersebut maka akan didapatkan pesan asli ialah "*Hello this is Secret Fichier!*" Berikut penulis tampilkan hasil dekripsi text tersebut dengan Secret Fichier, dan apabila di convert akan menghasilkan hexadecimal seperti diatas.

Gambar 4.2.6 Hasil Dekripsi

4.3 Perancangan Aplikasi

Pada tahap ini aplikasi mulai di rancang dengan permodelan UML (*Unified Modelling Language*), membuat stuktur yang ada didalam menu dan tampilan antar muka atau yang sering disebut *User Interface* dengan mempertimbangkan keefisiensian suatu aplikasi yang dibangun oleh peneliti. Rancangan aplikasi ini mencakup ;

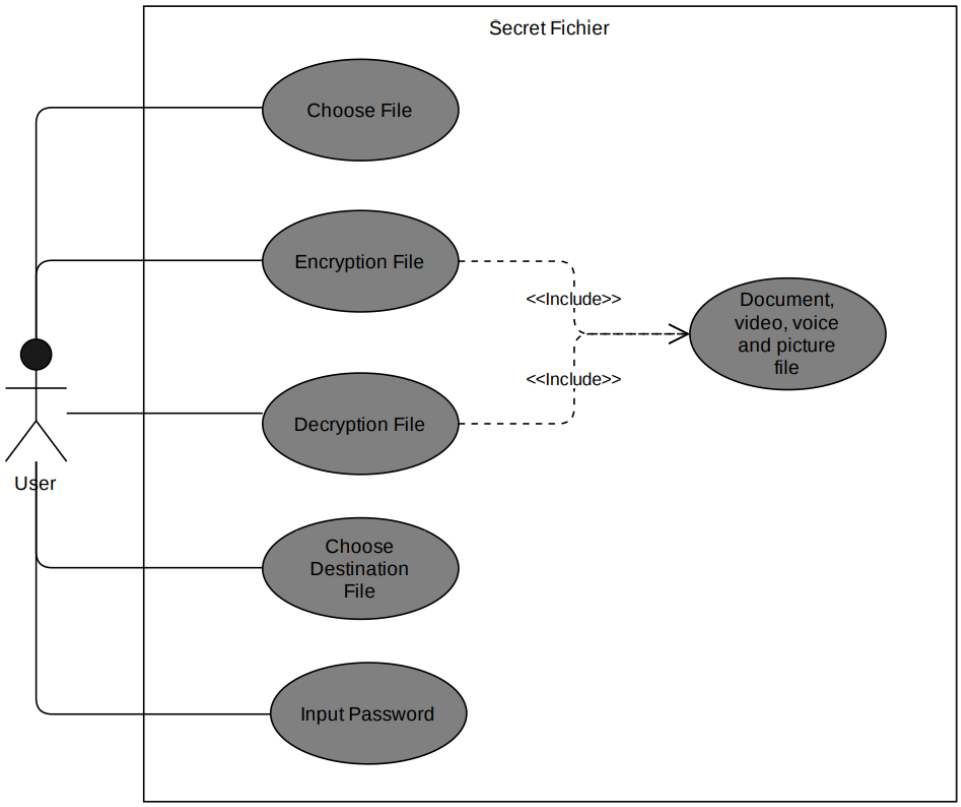
1. sistem yang ada didalam aplikasi yaitu tampilan menu utama yang berisikan penginputan suatu file dengan *radio button* untuk pemilihan opsi proses enkripsi atau dekripsi dan menu proses untuk pemilihan destinasi file apabila telah di proses dan penginputan password atau kunci sebelum terjadinya proses enkripsi dan deksripsi. Diharapkan penulis dengan mengedepankan suatu aplikasi yang efisien tanpa mengurangi suatu kegunaan dari aplikasi tersebut.
2. Pemodelan aplikasi menggunakan *Unified Modelling Language* (UML), yang terdiri dari *Use Case Diagram*, *Activity Diagram* dan *Sequence Diagram*.
3. Perancangan antar muka aplikasi yaitu penggambaran tampilan menu menu yang akan dibuat dan ditampilkan didalam aplikasi yang dibangun.

Berikut penjelasan penggambaran bagaimana interaksi antar user dengan sistem yang ada didalam *Secret Fichier* yang akan digambarkan dengan permodelan UML.

4.3.1 Use Case

Use Case Diagram merupakan pengambaran rangkaian interaksi antara actor dengan sistem. Actor mewakili sebagai user yang berinteraksi kepada sistem yang dimodelkan dengan *Use Case Diagram* didalam

diagram tersebut. Berikut pemodelan aplikasi Secret Fichier dengan UML.



Gambar 4.3.1 Use Case Diagram Secret Fichier

Penjelasan gambar :

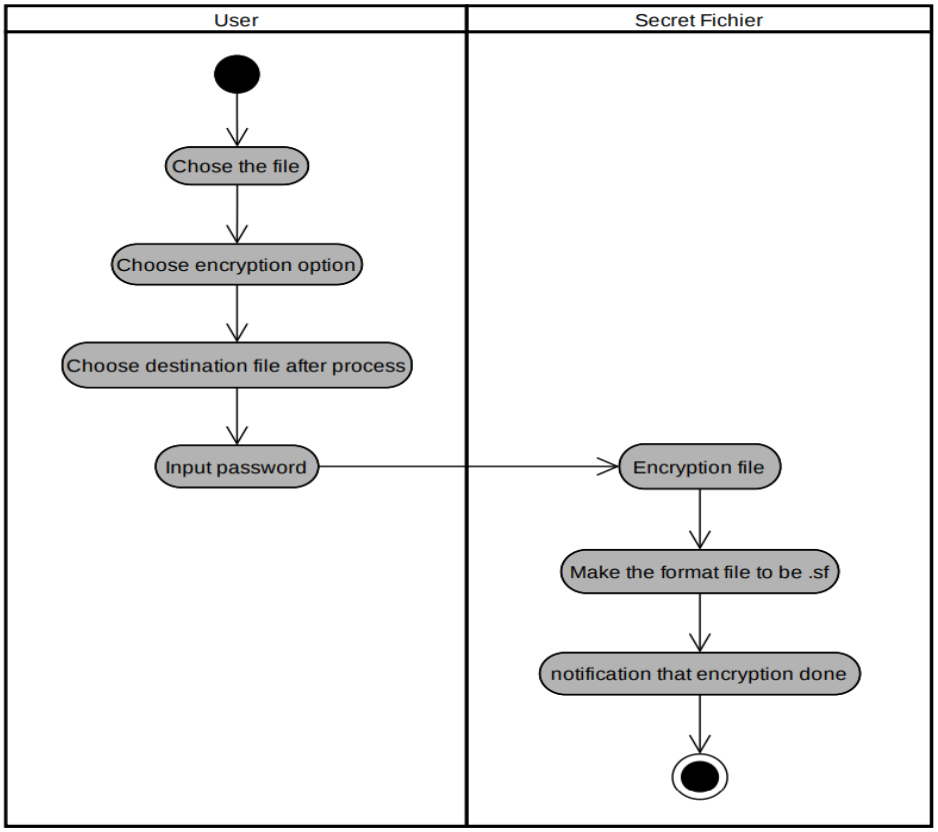
1. Pada *usecase* tersebut, *actor* atau *user* membuka aplikasi *Secret Fichier* maka tampilan awal dari *Secret Fichier* ialah *form dashboard*. Lalu user dapat memilih file yang akan di enkripsi atau di dekripsi dengan *choose file button* dengan catatan file yang akan di enkripsi ialah file tunggal bukan sebuah folder dan file yang akan di dekripsi ialah sebuah file dengan eksitensi format *.sf*.
2. Lalu, setelah user memilih file yang akan di process, ada dua buah opsi radio button yaitu encryption process dan decryption process. Dengan catatan file yang akan diproses yaitu file dokumen, gambar, suara dan video. Setelah dipilih proses mana yang akan dilakukan, lalu tombol start akan aktif dan menampilkan form process.

3. Selanjutnya di form process akan di tampilkan button file destination yang akan digunakan user untuk memilih tempat atau folder untuk file yang sudah di enkripsi atau di dekripsi.
4. Setelahnya user harus menginput sebuah password lebih dari 8 (delapan) character untuk menguci serta memproses file tersebut. Setelah file selesai di proses maka aplikasi akan balik ke halaman awal.

4.3.2 Activity Diagram

Pada activity diagram, penulis menggambarkan aktifitas suatu alurkerja yang ada didalam sistem yang dibuat oleh penulis. Dalam aplikasi Secret Fichier ini terdapat dua aktifitas didalamnya, yaitu proses enkripsi suatu file dan proses dekripsi suatu file.

4.3.2.1 Activity Diagram Enkripsi File

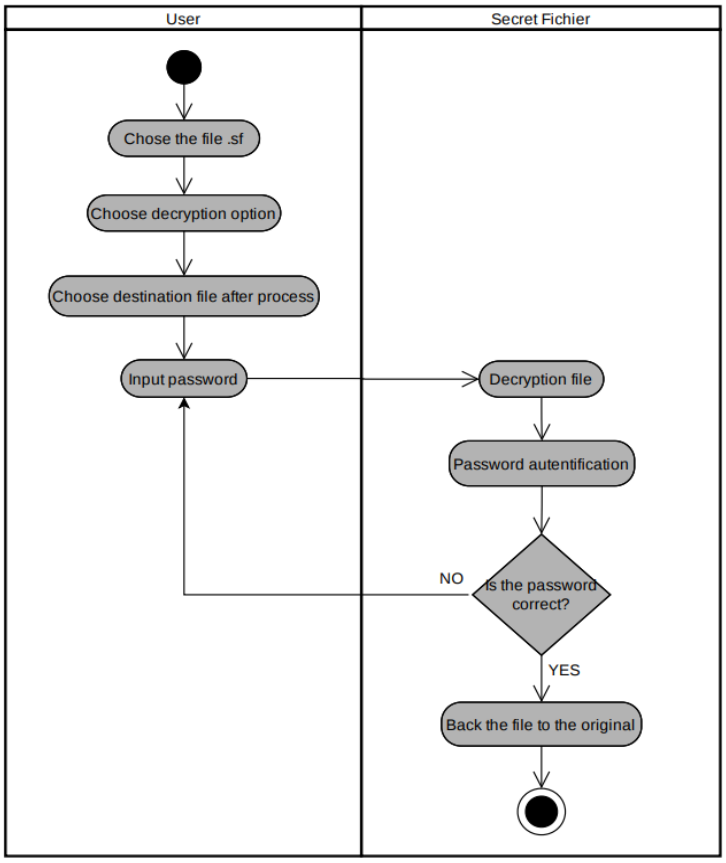


Gambar 4.3.2.1 *Activity Diagram* Enkripsi File

Penjelasan Gambar 4.3.2.1 :

1. Pada awal masuk aplikasi *Secret Fichier*, user akan ditampilkan *form dashboard* yang berisikan tampilan penginputan file yang akan diproses yang dinamain *choose file*. File yang akan di enkripsi hanya suatu file tunggal (bukan folder) yaitu file yang berjenis dokumen, suara, gambar dan video.
2. Lalu pada saat file diinput, tampilan *text box* akan menunjukan file itu berada, dan user akan memilih opsi *encryption button* pada *radio button* yang tersedia.
3. Setelah user mengklik tombol *encryption* dan tombol *start* akan aktif, tombol tersebut menghubungkan user ke form selanjutnya yaitu *form process*.
4. Lalu user akan ditampilkan *form process* dimana *user* akan memilih tempat file yang akan ditaruh apabila proses pengenkripsian telah selesai dan penginputan *password* atau kunci untuk memulai pengenkripsian file tersebut.
5. *User* akan mengklik *button destination* dimana *user* memilih dimana file tersebut akan ditaruh setelah proses selesai. Dalam proses tersebut, user dapat me-rename file tersebut. Mengganti nama file yang ingin dienkrispi berbeda dengan file aslinya.
6. Setelah menentukan letak file setelah proses, user harus menginputkan *password* atau kunci unik yang tidak mudah ditebak sebanyak 8 (delapan) character sebagai kunci file yang akan dienkripsi, kunci tersebut berguna kembali apabila user ingin mendekripsikan kembali file yang telah di enkrip.
7. Lalu user mengklik button process dan pengenkripsian file tersebut terjadi.

4.3.2.2 Activity Diagram Dekripsi File



Gambar 4.3.2.2 *Activity Diagram* Dekripsi File

Penjelasan Gambar 4.4.2.2 :

1. Pada awal masuk aplikasi *Secret Fichier*, user akan ditampilkan *form dashboard* yang berisikan tampilan penginputan file yang akan diproses yang dinamain *choose file*. File yang akan di dekripsi hanya suatu file tunggal (bukan folder) yaitu file yang berformat *.sf*.
2. Lalu pada saat file diinput, tampilan *text box* akan menunjukan file itu berada, dan user akan memilih opsi *decryption button* pada *radio button* yang tersedia.
3. Setelah itu user mengklik tombol start, dan user akan ditampilkan *form process*. Didalam *form process* tersebut process pengdekripsian terjadi.
4. Didalam form process, *user* akan mengklik *button destination* dimana *user* dapat memilih dimana file tersebut akan ditaruh setelah

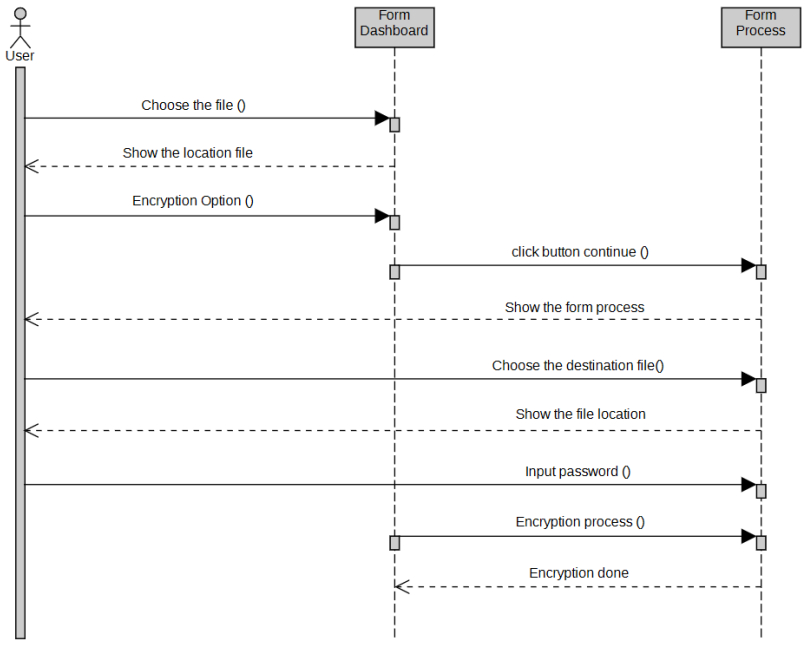
proses selesai. Dalam proses tersebut, *user* dapat me-rename file tersebut. Mengganti nama file yang ingin didekrispi berbeda dengan file enkripsinya.

5. Setelah menentukan letak file setelah di proses, user diharuskan menginputkan kembali *password* atau kunci yang telah dibuat sebelumnya pada proses pengenkripsian file. Setelah mengklik tombol *proses button*, terdapat *decision proses* dimana kunci tersebut benar atau tidak. Jika kunci tersebut benar maka proses pendekripsian selesai.
6. Apabila kunci yang di inputkan salah makan *user* akan diminta kembali memasukan kunci yang benar.
7. Setelah proses selesai user akan di kembalikan ke form dashboard.

4.3.3 Sequence Diagram

Sequence Diagram atau diagram sekuen merupakan diagram yang menggambarkan kolaborasi dari objek objek yang berinteraksi didalam usecase dengan mendekripsikan waktu hidup objek tersebut dan massage yang dikirim dan diterima antar objek tersebut.

4.3.3.1 Sequence Diagram Enkripsi File

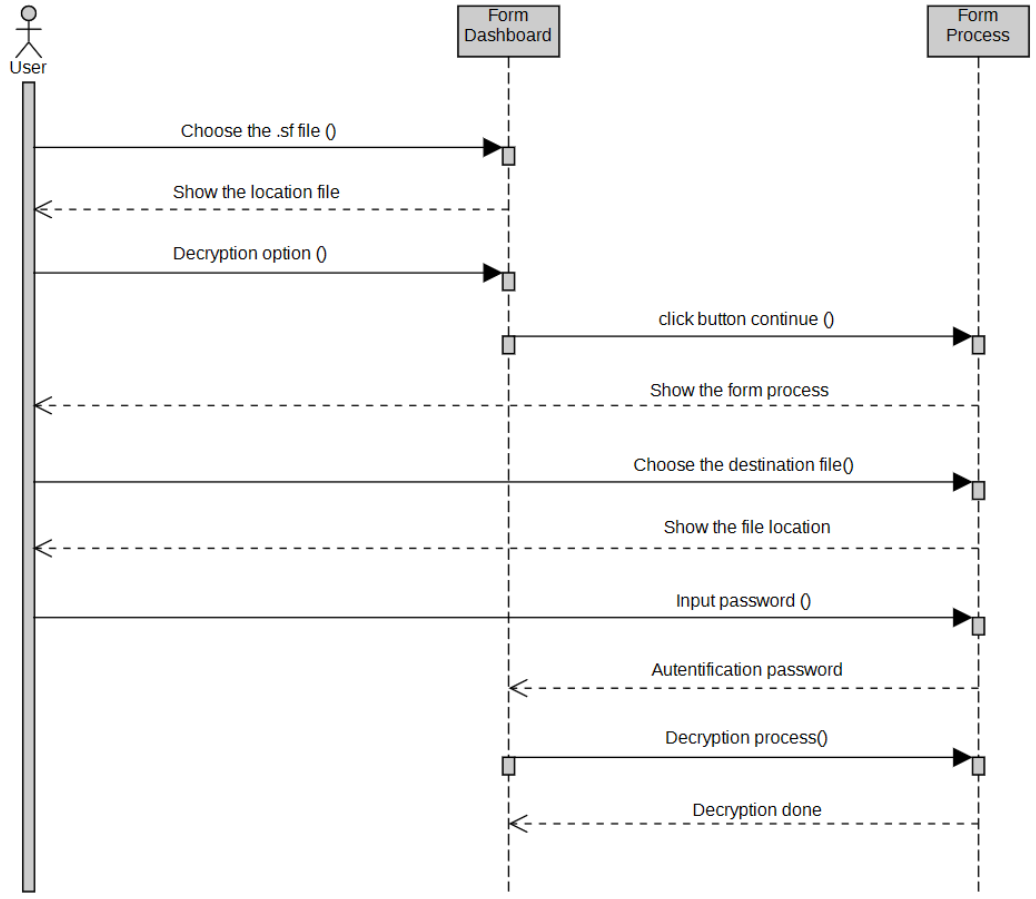


Gambar 4.3.3.1 *Sequence Diagram* Enkripsi File

Penjelasan Gambar :

1. Dalam sequence diagram ini terdapat 1 user atau yang disebur aktor, 2 lifeline yaitu form dashboard dan form process yang ada di secret fichier, dan 10 messages.
2. Pertama kali user membuka aplikasi, tampilan awalnya ialah form dashboard untuk menginput file yang ingin diproses,
3. Lalu user menentukan proses dan klik start button, lalu tampilan form process muncul.
4. Pilih destination file untuk file yang setelah di proses. Dan user menginputkan password lalu proses enkripsi terjadi.

1.3.3.2 Sequence Diagram Dekripsi File



Gambar 4.3.3.2 Sequence Diagram Dekripsi File

Penjelasan Gambar :

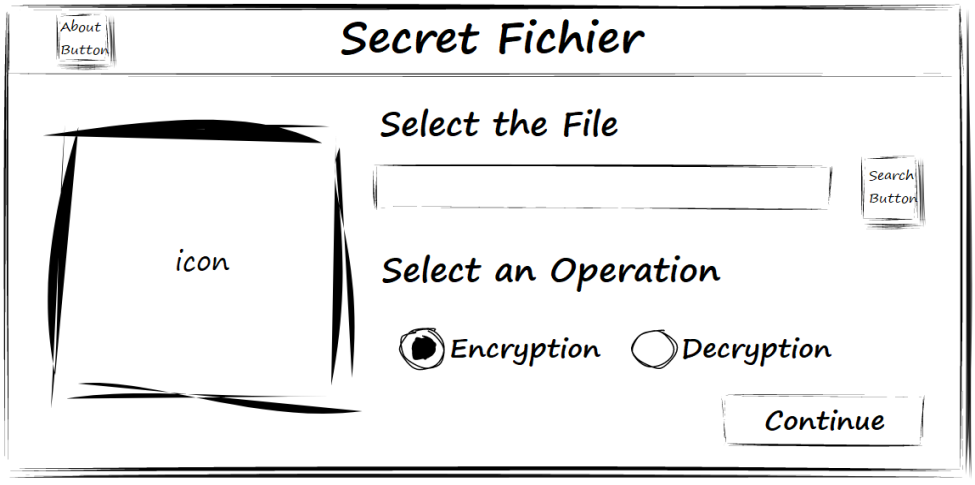
1. Pada sequence diagram dekripsi file ini terdapat 1 user, 2 lifeline yaitu form dashboard dan form process yang ada di dalam aplikasi secret fichier, dan 10 messages.
2. Pertama kali user membuka aplikasi, tampilan awalnya ialah form dashboard untuk menginput file yang ingin diproses, dalam dekripsi file di aplikasi secret fichier ini, format file harus .sf.
3. Lalu user menentukan proses enkripsi dan klik start button, lalu tampilan form process muncul.
4. Pilih destination file untuk lokasi file setelah di proses lalu user harus menginput kembali password awal pada pengengkripsian file.
5. Pada proses tersebut terdapat autentifikasi password, apabila password benar maka file akan kembali seperti semula, apabila password salah maka dekripsi file tidak berhasil.

4.4 Perancangan Antar Muka Aplikasi (*User Interface*)

Perancangan *User Interface* untuk aplikasi Secret Fichier adalah tampak seperti

4.4.1 Desain Tampilan Awal Aplikasi (*Form Dashboard*)

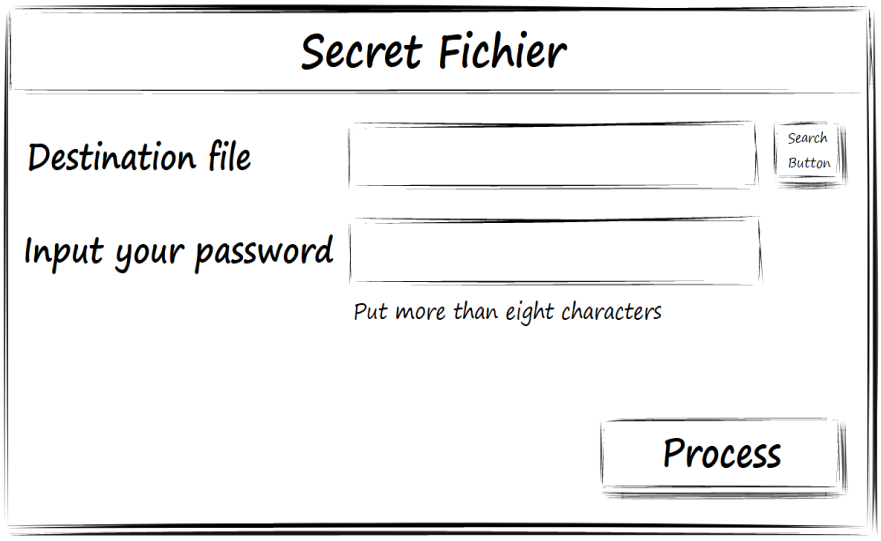
Dalam desain tampilan awal di aplikasi *Secret Fichier* pada awal masuk user akan di tampilkan langsung *dashboard* ada beberapa tombol didalamnya yang akan terkoneksi pada form *process* dan form *about*. Berikut desain tampilan *form dashboard* :



Gambar 4.4.1 Desain *Dashboard*

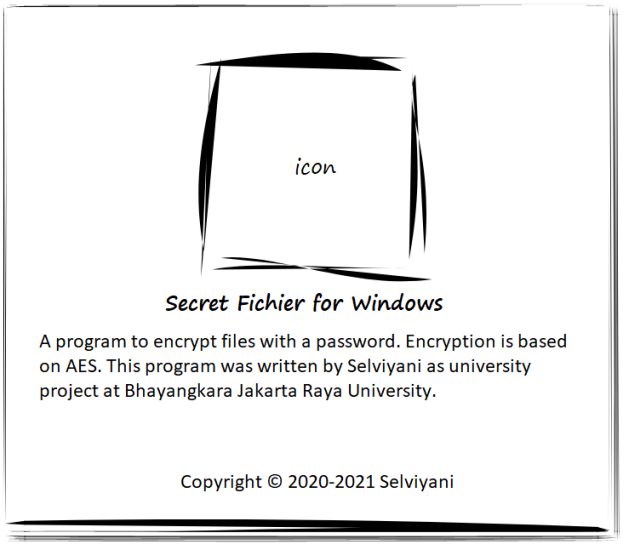
4.4.2 Desain Tampilan Menu Proses (*Form Process*)

Dalam desain tampilan berikutnya di aplikasi *Secret Fichier*, yaitu *form process* dimana saat user mengklik *button continue* dan *form process* muncul. Pada *form process* tersebut dimana proses dari enkripsi dan dekripsi sebuah file bekerja, di form tersebut juga user harus menginputkan sebuah *password* untuk memproses file yang ingin di enkripsi atau di dekrispi. Berikut tampilan dari form process



Gambar 4.4.2 Desain Form *Process*

4.4.3 Desain Tampilan Keterangan Aplikasi (*Form About*)



Gambar 4.4.3 Desain Form *About*

4.5 Fase Implementasi

Sebelum program diimplementasikan, maka program harus bebas dari kesalahan. Kesalahan program yang mungkin terjadi dikarenakan kesalahan dalam penulisan program (*coding*), kesalahan proses atau kesalahan logika.

Dalam fase implementasi aplikasi pengamanan data file berbasis windows "*Secret Fichier*" ini, analisis kebutuhan perangkat pendukung juga sangat diperlukan. Selain itu juga, kebutuhan perangkat lunak pendukung juga harus tersedia demi kelancaran tahap implementasi program. Dalam implementasi program ini ada beberapa proses yang dilakukan yaitu :

1. Memasukan kode program (coding), tahap ini dilakukan dengan menggunakan aplikasi Visual Studio 2019 .NET Core dengan menggunakan bahasa program C# (C Sharp).
2. Menguji aplikasi dengan mengenkripsi dan mendekripsi jenis file dengan format yang berbeda beda serta melakukan *debugging* atau perbaikan program jika perlu.

4.6 Implementasi Perangkat Lunak

Perangkat lunak yang digunakan penulis dalam membangun aplikasi. pengamanan data file "Secret Fichier" adalah System Operasi Windows 10 64 bit dan Visual Studio 2019 (.NET Core). Aplikasi *Secret Fichier* merupakan aplikasi enkripsi dan dekripsi sebuah file dengan penerapan AES-256 serta SHA-256 untuk teknik pengenkripsian pada data file tersebut. Keterbatasan aplikasi ini hanya dapat mengenkripsi sebuah file tunggal bukan folder dengan file berjenis dokumen, video, text dan gambar. Instalasi aplikasi *Secret Fichier* akan dijelakan pada point point berikut ini :

- Pada folder aplikasi *Secret Fichier*, user mengklik file yang bertuliskan .exe atau setup.
- Lalu user menginstall aplikasi seperti biasa. Klik next.
- Dan jika sudah berhasil terinstall, Secret Fichier sudah bisa dipakai untuk mengenkrip dan mengdekrip sebuah file.

- User dapat mendownload resource GitHub aplikasi Secret Fichier pada link berikut :
  https://github.com/selfviyani/SecretFichier.git

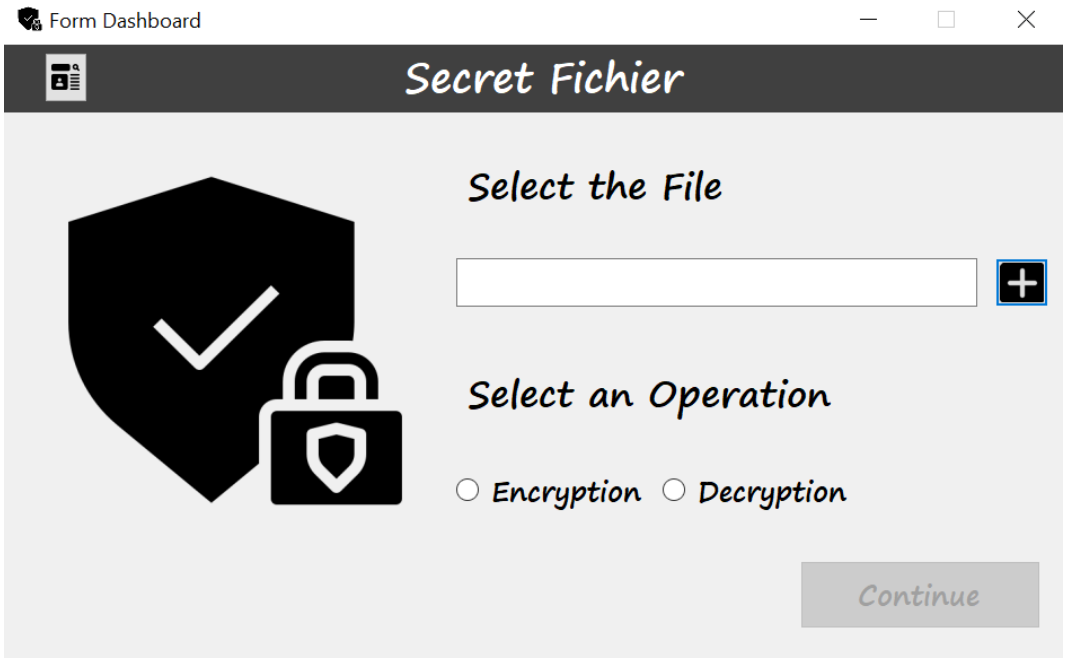
4.7 Implementasi Perangkat Keras

Perangkat keras yang digunakan penulis untuk membuat aplikasi ini antara lain adalah dengan sebuah laptop dengan spesifikasi mempunyai processor Intel Core i7-10510U dengan RAM 8 GB dan Solid State Drive 1000 GB serta VGA Nvidia MX250 2GB.

4.8 Implementasi Aplikasi

4.8.1 *Form Dashboard*

Berikut tampilan awal di aplikasi *Secret Fichier* pada awal masuk user akan di tampilkan langsung form *dashboard* sebagaimana berikut ini :

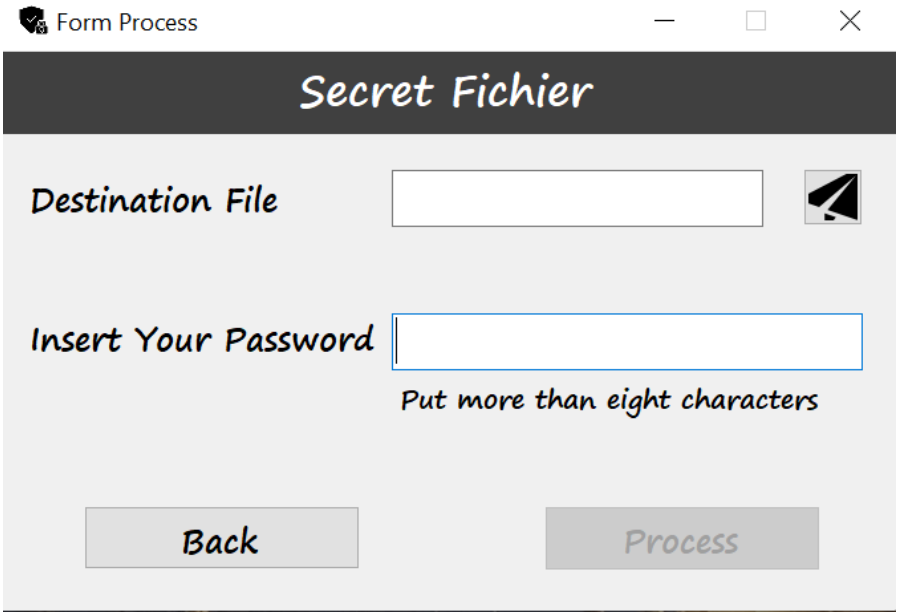


Gambar 4.8.1 Tampilan Form *Dashboard*

4.8.2 *Form Process*

Berikut tampilan pada saat *user* mengklik tombol *continue*, pada form ini ialah *form process* dimana, user akan menginputkan file yang akan

di proses dan *password* sebagai mana kuncinya. Berikut tampilan form process :



Gambar 4.8.2 Tampilan Form *Process*

4.8.3 *Form About*

Form ini menampilkan deklarasi bahwa aplikasi Secret Fichier benar benar dibuat oleh penulis berdasarkan referensi yang ada. Berikut tampilan dari form *about* :

Gambar 4.8.3 Tampilan *Form About*

4.9 Penerapan SHA-256 dan AES-256 Dalam Program

Penerapan algoritma Advanced Encryption Standard (AES-256) mode Chiper Block Chaning (CBC) dan Secure Hash Algorithm (SHA-256) terletak pada form process di aplikasi Secret Fichier. Pada form ini proses pengenkripsian dan pendekripsian sebuah file di process, dari penginputan sebuah password yang memiliki panjang character delapan buah hingga pemilihan lokasi file setelah di process. Setelah sebuah file yang telah di input pada form dashboard selanjutnya di process pada form process, pada form inilah algoritma AES-256 mode CBC bekerja untuk memproses file tersebut, dan juga algoritma SHA-256 sebagai algortima fungsi hash dari kunci yang telah di inputkan agar merubah sebuah password yang tidak bisa dibaca.

Sebuah data file yang telah dienkripsi akan menjadi sebuah file yang tidak bisa terbaca dengan format *.sf* pada data file tersebut. Dan apabila user ingin mengembalikan sebuah data yang sudah dienkripsi diperlukan proses dekripsi untuk mengembalikan data tersebut. Proses dekripsi file juga terjadi di dalam form process, algoritma AES-256 mode CBC bekerja untuk mengembalika data file yang sudah dienkripsi menjadi data file yang bisa terbaca kembali, dengan syarat format file *chipertext* tersebut ialah *.sf* dan user masih mengingat password yang sudah dibuatnya pada proses enkrispi pada file tersebut. Apabila user tidak menginput password yang benar maka algoritma AES-256 mode CBC serta fungsi hash SHA-256 tidak dapat memprosesnya dikarena password yang salah.

Berdasarkan pengamanan data file yang dilakukan oleh aplikasi *Secret Fichier* berbasis windows, dengan sempel data file dari berbagai jenis ekstensi file yaitu file dokumen, gambar, suara serta video akan dilakukan pengamanan data file (enkripsi) sampai menjadi sebuah data file yang tidak terbaca apa informasinya.

4.10 Hasil Pengujian

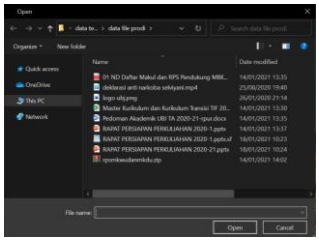
Dalam pengujian aplikasi Secret Fichier, penulis menerapkan pengujian *blackbox*. Sebagaimana disebutkan blackbox ialah metode yang digunakan untuk menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah input diterima dengan benar dan output yang dihasilkan telah sesuai dengan yang diharapkan. Secret Fichier diharapkan dapat menerima input

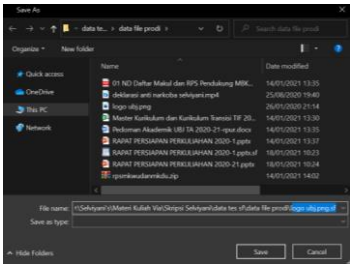
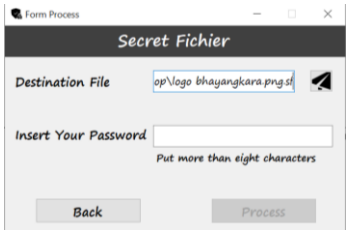
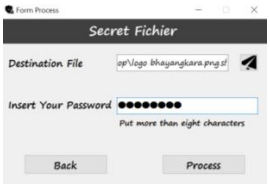
yang diharapkan yaitu dapat mengenkripsi file dengan eksitensi yang berbeda dan mendapatkan output berubah chipertext yang diharapkan tidak dapat terbaca. Secret Fichier juga diharapkan dapat mengdekripsi sebuah file yang telah di enkripsi sebelumnya dengan kunci yang tepat. Pada pengujian ini dilakukan oleh penulis pada minggu ke 2 bulan Januari 2021. Berikut tabel pengujian Secret Fichier :

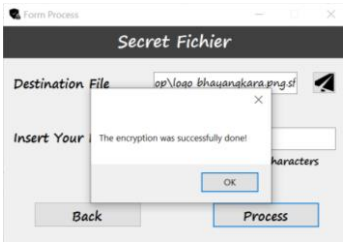
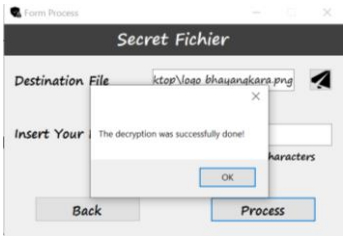
Tabel 4.10.1 Rancangan Pengujian

Kelas Uji	Tes Uji	Jenis Pengujian
Form Dashboard	Memilih file	*Blackbox*
	Menampilkan directory	*Blackbox*
	Memilih opsi proses	*Blackbox*
	Mengklik tombol process setelah menentukan.	*Blackbox*
Form Process	Tampil form process	*Blackbox*
	Memilih destinasi file	*Blackbox*
	Menampilkan directory	*Blackbox*
	Menginput password	*Blackbox*
	Memproses enkripsi	*Blackbox*
	Memproses dekripsi	*Blackbox*

Tabel 4.10.2 Pengujian Implementasi

Kelas Uji	Skenario Pengujian	Hasil Yang Diharapkan	Hasil Pengujian
Form Dashboard	Dapat memilih file pada button search dan menampilkan directory. 	Pemilihan file berjalan dengan lancar dan tidak ada kendala. Serta dapat menampilkan directory 	Berhasil
	Dapat menampilkan lokasi dari file yang ada di	Lokasi file dapat ditampilkan setelah	Berhasil

	directory pada text box yang tersedia. 	memilih file pada text box. 	
	Memilih radio button pada proses file yang akan dijalani, tombol enkripsi dan tombol dekripsi	Dapat memilih proses dengan baik dan proses yang dijalani lancar tanpa erorr.	Berhasil
Form Process	Memilih file destination pada tombol search untuk menyimpan hasil dari proses yang telah dijalani. 	Dapat bejalan dengan lancar tanpa kendala saat memilih destinasi file yang ada di directory yang ingin disimpan. 	Berhasil
	Menampilkan directory file dan mengubah nama file yang ingin disimpan. 	Dapat menampilkan lokasi file yang sesuai dan benar serta dapat me-rename sebuah file. 	Berhasil
Form Process	Menginput password delapan atau lebih karakter. 	Dapat menginput password dengan baik 	Berhasil

Form Process	Memproses enkripsi file dengan baik. 	Dapat mengenkripsi file dengan baik dan menghasilkan file yang tidak terbaca sama sekali 	Berhasil
Form Process	Memproses dekripsi file dengan baik. 	Dapat mengdekripsi file yang telah di enkripsi dengan baik dengan kunci yang tepat.	Berhasil

Semple data file yang di uji cobakan ialah data file yang berada di Program Studi Infomatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya. Berikut penulis jabarkan data file-nya sebagai berikut :

1. Pedoman Akademik UBJ TA 2020-21-rpur.docx (data 1)
2. Master Kurikulum dan Kurikulum Transisi TIF 2019 rpur.xlsx (data 2)
3. Rapat Persiapan Perkuliahan 2020-1.pptx (data 3)
4. 01 ND Daftar Makul dan RPS Pendukung MBKM Update (All).pdf (data 4)
5. Rpsmkwudanmkdu.zip (data 5)
6. Logo ubj.png (data 6)
7. Deklarasi anti narkoba Selviyani.mp4 (data 7)

Tabel 4.10.3 Pengujian Enkripsi dan Dekripsi File.

Nama File	Ukuran File (bytes)	Panjang Password	Ukuran File Setelah Dienkripsi (bytes)	Presentase Perubahan bytes (%)	Waktu Proses Enkripsi (detik)	Waktu Proses Dekripsi (detik)
data1.docs	14.582.885	12 karakter	14.582.896	∞	11.26/det	6.13/det
data2.xlsx	113.794	11 karakter	113.808	0.012%	2.19/det	0.56/det
data3.pptx	1.625.761	10 karakter	1.625.776	0.001%	10.26/det	4.13/det
data4.pdf	761.725	13 karakter	761.728	0.0004%	4.78/det	2.32/det
data5.zip	14.346.667	13 karakter	14.346.672	∞	11.70/det	5.67/det
data6.png	239.800	10 karakter	239.808	0.0033%	2.77/det	1.08/det
data7.mp4	101.895.158	15 karakter	101.895.168	∞	12.48/det	6.32/det

Berdasarkan dari tabel pengujian Secret Fichier, dapat disimpulkan semua fungsionalitas sistem dapat berjalan dengan lancar dan baik. Tidak ditemukannya sebuah *bugs* atau *erros* yang tidak diharapkan. Dan dapat disimpulkan pula setelah mengalami proses enkripsi, suatu file menghasilkan perbedaan ukuran file dengan awal file sebelum di enkripsi dengan sesudah enkripsi, dikarenakannya padding dan penyesuaian block enkripsi yang mempunyai kelipatan 16 bytes. Pada tabel pengujian diatas, rata rata penambahan byte pada file diatas ialah 8 bytes. Serta dapat disimpulkan jika ukuran file dapat mempengaruhi lamanya proses enkipsi dan dekripsi.

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat diambil berdasarkan hasil dari pengujian serta analisis penerapan algoritma *Advanced Encryption Standard* (AES-256) dengan mode *Chiper Block Chaining* (CBC) dan *Secure Hash Algorithm* (SHA-256) terhadap aplikasi yang telah dibuat yaitu aplikasi enkripsi dan dekripsi file berbasis Windows yaitu *Secret Fichier*. Dapat disimpulkan beberapa point sebagai berikut :

a. Untuk Mencegah terjadinya kebocoran data yang bersifat penting dan merugikan beberapa pihak, oleh karena itu data bersifat penting harus di enkripsi terlebih dahulu dengan aplikasi *Secret Fichier* untuk melindungi data penting pada Univeristas Bhayangkara Jakarta Raya.
b. Aplikasi *Secret Fichier* mampu mengenkripsi file dengan berbagai ekstensi seperti file dokumen, file suara (voice note/mp3), file video serta file gambar dengan baik dan dapat didekripsikan kembali dengan kunci yang sama pada aplikasi *Secret Fichier*, aplikasi ini mampu mengamankan data bersifat penting untuk Universitas Bhayangkara Jakarta Raya sebagai penerapan baku dari aplikasi enkripsi berbasis dekstop.
c. Algoritma AES mode CBC dan SHA yang diterapkan di aplikasi dapat berjalan dengan baik tanpa kendala di aplikasi Secret Fichier untuk mengenkripsi file dan mendekripsikan nya di dalam sistem operasi Windows 10 64 bit.
d. Algoritma AES dan SHA dapat dibilang masih cukup aman didalam pemprosesannya dikarenakan mempunyai kunci yang panjang dan tahapan perhitungan yang cukup rumit di dalamnya.
e. Pada proses pengenkripsian file terdapat perbedaan ukuran file asli dengan file yang dienkripsi dikarenkan adanya proses *padding* didalam proses pengenkripsian tersebut, sehingga menunjukan perbedaan ukuran file asli dengan file yang telah di enkripsi.

5.2 Saran

Untuk penelitian lebih lanjut, perlu dipertimbangkan kembali berdasarkan kesimpulan yang telah dipaparkan diatas. berikut untuk saran saran untuk penelitian selanjutnya :

a. Untuk penelitian selanjutnya, disarankan untuk dapat mengenkripsi file dengan berukuran lebih besar dan dapat dijalan di sistem operasi lainnya (tidak hanyak windows).
b. Untuk penelitian selanjutnya, pertimbangkan kembali untuk menggunakan mode operasi yang lainnya, seperti CFB (Chiper FeedBack), OFB (Output FeedBack) atau GCM (Galois Counter Mode).
c. Untuk penelitian selanjutnya, disarankan untuk dapat menggunakan seri SHA terbarukan yaitu SHA-3.
d. Untuk penelitian selanjutnya, pertimbangkan kembali untuk menggunakan algoritma asimetris. Dimana memiliki dua kunci yang berbeda untuk keamanannya.
e. Untuk penelitian selanjutnya, aplikasi *Secret Fichier* dapat dikembangkan dan diterapkan pada *mobile* atau menjadi sebuah fitur *independent* dalam sebuah aplikasi *messenger*.
f. Penerapan Electronic Data Interchange (EDI) bisa menjadi salah satu solusi untuk membuat keamanan dalam transaksi bisnis di Internet dan dalam pertukaran file melalui handshakes file.

DAFTAR PUSTAKA

Ambler, S. W. (2005). *The Elements of UML 2.0 Style*. New York: Cambridge University Press.

Ariyus, D. 2008. Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi, Yogyakarta : Andi.

Henry, Kridalaksana, A. H., & Arifin, Z. (2016). Kriptografi AES Mode CBC Pada Citra Digital Berbasis Android. Prosiding Seminar Ilmu Komputer dan Teknologi Informasi, 45-52.

Ichwan, M., Gustiana, M., & Nurjaman, N. R. (2016). Implementasi Keyed-Hash Message Authentatication Code Pada SistemKeamanan Rumah. *MIND Journal*, 9-18.

Menezez, Alfred., Vanstone, S., & Oorschot, P. (2006). *Handbook of Applied Cryptography.* Boston: Massachusetts Institute of Technology.

Prasetyo, R., & Surayana, A. (2016). Aplikasi Pengamanan Data dengan Teknik Algoritma Kriptografi AES dan Fungsi Hash SHA-1 Berbasis Desktop. *Jurnal SISFOKOM*, 61-65.

Renaldy, M. (2015). Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode AES-128 (Advanced Encryption Standard-128) Dan SHA-1 (Secure Hashing Algorithm-1). *Tugas Akhir*, 7-19.

Simangunsong, P. B. N., & Fitri, K. (2018). Perancangan Aplikasi Pengamanan Citra Bewarna Dengan Algoritma RSA. *Jurnal Teknik Informatika*, 99-107

Surian, D. (2006). Algoritma Kriptografi AES Rijndael. *Jurnal Teknik Elektro*, 97-101.

Sulastri, S., & Putri, R. D. M. (2018). Implementasi Enkripsi Data Secure Hash

Algorithm(SHA-256) dan Message Digest Algorithm (MD5) pada Proses Pengamanan Kata Sandi Sistem Penjadwalan Karyawan. *Jurnal Teknik Elektro*, 70-74.

Wiguno, H. F. (2017). Aplikasi Pengamanan File Dan Pesan Teks Menggunakan AES 256 Dan SHA 256 Berbasis Android. *Tugas Akhir*, 33-49.

Yusmantoro, S., Hermansyah, E., & Efendi, R. (2014). Rancang Bangun Aplikasi Pengamanan Keaslian Surat Izin Tempat Usaha Menggunakan Algoritma Elgamal Dan Secure Hash Algorithm 256 Studi Kasus : Badan Pelayanan Perizinan Terpadu (BPPT) Kota Bengkulu. *Jurnal Rekursif*, 28-36.

UNIVERSITAS BHAYANGKARA JAKARTA RAYA

Jl. Harsono RM No.67, Ragunan, Pasar Minggu, Jakarta Selatan, 12550
Telepon : (021) 27808121, 27808882
Jl. Perjuangan, Bekasi Utara
Telepon : (021) 88955882, Fax : (021) 88955871
Web: *www.ubharajaya.ac.id* **Email:** *fasilkom@ubharajaya.ac.id*

Bekasi, 11 Januari 2021

Nomor : B/008/I/2021/FASILKOM-UBJ
Klasifikasi : Biasa
Perihal : Permohonan mengambil data penelitian

Kepada
Yth. **Ketua Program Studi Informatika**
Universitas Bhayangkara Jakarta Raya
Jl. Raya Perjuangan, Bekasi Utara, Jawa Barat, 17121

Dengan hormat,

Sebagai bagian dari kurikulum Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya, mahasiswa diharuskan melakukan penulisan tugas akhir (skripsi) sebagai salah satu persyaratan kelulusan sebagai Sarjana Ilmu Komputer.

Untuk itu kami mohon bantuan Kepada Ketua Program Studi Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya untuk dapat mengijinkan mahasiswa kami melakukan penelitian/pengambilan data, dengan judul penelitian **Penerapan Algoritma Advanced Encryption Standard (AES-256) Dengan Mode CBC dan Secure Hash Algorithm (SHA-256)** Untuk Pengamanan Data File yang Bapak/Ibu pimpin, mahasiswa kami tersebut adalah :

Nama : **SELVIYANI**
NPM : 2017.10.225.097
Program Studi : Informatika

Atas perhatian dan perkenaannya kami ucapkan terima kasih.

Dekan Fakultas Ilmu Komputer
Universitas Bhayangkara Jakarta Raya

Herlawati, S.Si., MM., M.Kom
NIP: 2001 452

Plagiarism Checker X - Report

Originality Assessment

Overall Similarity: **10%**

Date: Jul 25, 2021

Statistics: 2620 words Plagiarized / 25460 Total words

Remarks: Low similarity detected, check your supervisor if changes are required.

BAB I PENDAHULUAN 1.1 Latar Belakang Maraknya perkembangan teknologi di zaman serba digital sekarang ini sangatlah menunjang kegiatan manusia, terutama dibidang inovasi dan kretifitas dalam bekerja, belajar, penyebaran informasi dan lain lain. Sayangnya semakin maju teknologi di zaman modern ini, semakin maju pula tingkat kejahatan dengan teknologi sekarang. Oleh karena itu edukasi terhadap penggunaan teknologi sangat diperlukan, pengamanan data yang tepat diperlukan guna meningkatkan keamanan yang menjamin agar tidak disalah gunakan oleh orang yang tak bertanggung jawab untuk keperluan yang tidak semestinya. Banyaknya jenis kejahatan dimasa sekarang perlu diwaspadai, salah satunya cyber crime. Cyber crime dapat diartikan sebagai tindak kenjahatan di dunia maya yang menjadikan teknologi computer dan jaringan internet sebagai sasarannya. Data yang bersifat pribadi menjadi objek yang disenangi oleh hacker untuk dimanupulasi, dipermaikan dan digunakan tidak pada semestinya. Oleh karena itu data yang bersifat pribadi atau rahasia perlu dijaga keamanannya. Ada beberapa teknik pengamanan data, diantaranya adalah teknik enkripsi. Enkripsi merupakan sebuah proses pengubahan sebuah pesan atau informasi dari yang bisa dimengerti atau dibaca menjadi sebuah pesan atau informasi yang sulit dimengerti hingga tidak terbaca sama sekali.Teknik enkripsi dapat mengamankan data karena data dapat berubah menjadi tidak terbaca sesuai dengan aslinya. Dan data yang terenkripsi dapat terbaca lagi apabila sudah di deskripsi dengan menggunakan kunci yang tepat. Dan dengan mengenkripsi data file yang penting atau rahasia dapat meningkatkan keamanan data yang bersifat rahasia tersebut. Kriptografi merupakan studi matematika yang mempunyai hubungan dengan aspek keamanan informasi seperti integritas data, keaslian entitas dan keaslian data (Ratno Prasetyo, 2016). Dalam ilmu kriptografi terdapat dua proses penyandian yang disebut enkripsi dan deskripsi. Enkripsi dilakukan pada proses pengiriman pesan atau informasi dengan cara mengubah data asli kedalam bentuk kode kode yang menjadikannya data rahasia sedangkan deskripsi dilakukan pada proses penerimaan dengan cara mengubah data yang berisi kode kode rahasia tersebut ke dalam bentuk data yang asli dan mudah dimengerti. Pada tanggal 29 September 2020 berita tentang serangan ransomeware yang

melumpuhkan salah rumah sakit di Amereka Serikat akibatnya data data penting rumah sakit habis terenkripsi oleh sebuah virus, ransomeware sendiri ialah salah satu jenis malware berbahaya yang menyerang sistem komputer untuk mengenkripsi file didalamnya. Maka dari itu diperlukannya enkripsi file untuk file yang dianggap penting oleh user. Oleh karena itu universitas Bhayangkara Jakarta Raya sebagai perguruan tinggi swasta yang terletak di kota Bekasi, Jawa Barat. Pada Universitas Bhayangkara Jakarta Raya mempunyai banyak data file penting yang bersifat rahasia suatu lembaga, seperti data data keuangan dan data penting lainnya pada komputer atau laptop di lembaga mereka. Dan apabila data tersebut bisa saja dicuri dan dimanipulasi pada suatu kejadian yang dapat merugikan lembaga. Data keuangan yang tidak terenkripsi atau tidak dirahasiakan dapat dengan sangat mudah dimanipulasi oleh orang yang tidak bertanggung jawab untuk mengambil keuntungan di lembaga yang bergerak dibidang pendidikan tersebut, seperti dikorupsi pada jumlah pengeluaran untuk biaya operasional perusahaan tersebut dan biaya biaya lainnya dan apabila data data tersebut di hack oleh virus yang terjangkit di dalam computer seperti data keuangan dan data penting lainnya maka data tersebut akan terenkrip dengan virus dan tidak bisa dikembalikan lagi datanya. Oleh karena itu penulis berniat untuk menjadikan hal tersebut sebagai bahan penelitian penulis guna menyelesaikan tugas akhir untuk jenjang strata satu yang penulis tempuh. Dengan mengamankan data data rahasia lembaga pendidikan Universitas Bhyangkara Jakarta Raya menggunakan teknik enkripsi dan dekripsi yang penulis terapkan pada penelitian ini semoga dapat membantu lembaga tersebut untuk dapat menjaga kerahasiaan data mereka. Oleh karena itu, terdapat metode algoritma kriptografi yang cocok untuk memecahkan masalah pengamaanan data lembaga tersebut, yaitu salah satunya adalah metode AES dan SHA. Advanced Encryption Standar (AES) adalah algoritma kriptografi simetris modern yang beroperasi dalam mode penyandian blok (block cipher) yang memproses blok data dengan ukuran 128-bit dengan panjang kunci 128-bit, 192-bit, atau 256-bit (Asep Suryana, 2016). Terdapat beberapa mode dalam algoritma AES diantaranya mode CBC, ECB, OFB, CTR dan CFB untuk penyadian dengan metode block cipher. Penulis

menggunakan mode CBC atau yang sering disebut Cipher Block Chaining ialah metode penyandian blok berulang seperti rantai yang menggunakan veckor inisialisasi (IV) atau sering disebut deret biner unik dengan panjang tertentu untuk tiap enkripsi. Salah satu karakteristik utamanya CBC menggunakan mekanisme rantai yang membuat chipertext blok sebelumnya bergantung pada semua blok chipertext sebelumnya. Dengan segala pertimbangan mode operasi AES yang ada, penulis memilih untuk menggunakan mode CBC dengan segala kelebihan dan kekuranganya. SHA merupakan algoritma Hashing atau yang sering di sebut sebagai fungsi hash merupakan sebuah algoritma yang mengubah teks atau pesan menjadi sederetan karakter acak yang memiliki jumlah karakter yang sama (Ratno Prasetyo, 2016) yang dipublish oleh National Institute Of Standard and Technology (NIST) pada tahun 2001. SHA sendiri mempunyai beberapa jenis yaitu SHA-0, SHA-1 dan SHA-2. Untuk penelitian ini penulis memilih SHA-2 yang memiliki beberapa fungsi hash dengan digest yaitu 224, 256, 384 dan 512 bits atau SHA-224, SHA-256, SHA-384 dan SHA-512. Algoritma kriptografi modern simetri tersebut terbukti pernah dipakai pada penelitian terdahulu mengenai pengamanan data oleh (Muammar Renaldy, 2015) Penelitian tersebut membahas tentang Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode AES dan SHA-1 pada Universitas Budi Luhur. Penelitian tersebut menguji metode AES dan SHA untuk mengkripsi aplikasi S-Diary. Dan algoritma AES ini juga pernah di gunakan oleh (Hadi Fajar, 2019) dalam penelitiannya yaitu Aplikasi Pengamanan File dan Pesan Teks Menggunakan Algoritma AES 256 dan SHA 256 Berbasis Android pada Universitas Pembangunan Nasional “Veteran” Jakarta. Berdasarkan latar belakang permasalahan diatas maka penelitian yang dilakukan mengambil judul “Penerapan Algoritma Advanced Encryption Standard (AES-256) Dengan Mode CBC dan Secure Hash Algorithm (SHA-256) Untuk Pengamanan Data File”.

1.2 Identifikasi Masalah Berdasarkan uraian diatas, maka dapat disimpulkan indentifikasi masalah sebagai berikut. 1. Kebocoran data file yang bersifat sensitif sering terjadi dalam kehidupan sehari hari dan menimbulkan kerugian untuk pihak yang dirugikan serta belum adanya penerapan yang baku untuk pengamanan data file penting pada Universitas

Bhayangkara. 2. Di perlukannya sebuah aplikasi baku untuk pengamanan data file untuk menjaga kerahasiaan data tersebut dan belum adanya aplikasi pengamanan file pada Universitas Bhayangkara Jakarta Raya berbasis desktop. 1.3 Rumusan Masalah Berdasarkan kesimpulan diatas, maka ditetapkan rumusan masalah dalam penelitaan ini sebagai berikut. 1. Bagaimana Penerapan Algoritma Advanced Encryption Standard (AES-256) Dengan Mode CBC dan Secure Hash Algorithm (SHA-256) Untuk Pengamanan Data File? 2. Bagaimana Perancangan Aplikasi Untuk Pengamanan Data File Menggunakan AES-256 Dengan Mode CBC dan SHA-256? 1.4 Tujuan dan Manfaat Maksud dari penulis dari penelitian pada pengamanan data di Universitas Bhayangkara adalah sebagai berikut. 1. Menerapkan algoritma Advanced Encryption Standard (AES-256) Dengan Mode CBC dan Secure Hash Algorithm (SHA-256) dalam pengamanan data file penting Lembaga kedalam aplikasi yang dibuat oleh penulis yang dinamai Secret Fichier. Secret Fichier sendiri adalah Bahasa Perancis yang berarti Secret yaitu rahasia dan Fichier berarti file. Jadi yang dimaksudkan Secret Fichier ialah file rahasia. Sedangkan maksud dan tujuan penulisaan ini adalah untuk memenuhi syarat Skripsi pada Semester Tujuh Program Studi Informatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya. 1.5 Batasan Masalah Pembatasan permasalahan diharapkan tidak menyimpang dari pokok permasalahan, sehingga dalam penyeselesaian masalah ini akan dibatasi dimana ruang lingkup penelitian dilakukan untuk divisi administrasi dalam pengamanan data file penting lembaga yang dituangkan kedalam pembuatan aplikasi enkripsi dan deskripsi berbasis desktop. Adapun batasan masalah dalam pengimplementasian algoritma dari AES-256 dengan mode CBC dan SHA-256 ke dalam aplikasi enkripsi dan deskripsi berbasis desktop di sistem operasi Windows. Maka dari itu pembatasan tersebut akan dijelaskan dibawah ini : 1. Proses enkripsi dan deskripsi menggunakan algoritma AES dengan panjang kunci 256 bit dengan mode operasi CBC. Dan untuk Teknik memasukan kunci, dilakukan dengan proses hashing menggunakan SHA agar dapat menghasilkan kunci sebesar 256 bit. 2. Aplikasi enkripsi dan deskripsi Secret Fichier hanya dapat mengenkripsi dan mengdeskripsi file tunggal (bukan folder). 3. Aplikasi

enkripsi dan deskripsi Secret Fichier ini mencakup data yang berjenis dokumen, gambar, suara dan video. 4. Untuk mengdeskripsi suatu file digunakan password (kunci) yang sama pada saat si pengguna mengenkripsi file tersebut. 5. Penulis menerapkan algoritma AES-256 mode CBC dan SHA-256 kedalam aplikasi dan menjelaskan tahapan tahapan algoritma tersebut bekerja kedalam penulisan ini. 1.6 Tempat Penelitian Universitas Bhayangkara Jakarta Raya, Jl. Raya Perjuangan, Bekasi Utara, Kota Bekasi, Jawa Barat 17121, Indonesia. 1.7 Sistematika Penulisan Penelitian ini akan dibagi menjadi lima bab gambaran masing masing bab akan dijelaskan dibawah ini. BAB I : PENDAHULUAN Dalam bab ini berisi penjelasan tentang latar belakang masalah, maksud dan tujuan penelitian, rumusan masalah, pembahasaan masalah, metode pengumpulan data dan sistematika penulisan. BAB II : LANDASAN TEORI Dalam bab ini menjelaskan tentang memuat tinjauan dan ulasan singkat mengulas pentinnya penelitian yang dilakukan dan menyampaikan teori yang berhubungan dengan permasalahaan yang dibahas sebagai dasar analisa permasalahaan yang diteliti. BAB III : METODE PENELITIAN Dalam bab ini membahas tentang pendekatan studi dan dapat beruba analisis teori, metode eksperimen, kombinasi, rancangan, spesifikasi sistem baik perangkat keras maupun perangkat lunak. BAB IV : HASIL DAN PEMBAHASAAN Dalam bab ini membahas mengenai penerapaan algoritma AES dan SHA serta perancangan aplikasi meliputi perangkat lunak berbasis dekstop, pengujian dan implementasi serta hasil keluaran dari sistem aplikasi yang telah dibuat dan di bahas sesuai penelitian dan hipotesis untuk menjawab permasalahan yang ada. BAB V : PENUTUP Dalam bab ini memuat beberapa kesimpulan yang di dapatkan dari penelitian dan menjawab tujuan penelitian atau hipotesis. Serta memuat saran saran yang dapat dikembangkan atau dilakukan sebagai penerapan untuk perusahaan kedepannya. BAB II LANDASAN TEORI 2.1 Kriptografi Istilah kriptografi, cryptography berasal dari bahasa Yunani yaitu, “cryptos” yang artinya “secret” atau rahasia sedangkan “graphien” yang artinya “writing” atau tulisan, sehingga kriptografi berarti secret writing yang artinya tulisan rahasia. Pengertian kriptografi secara lebih luas adalah Kriptografi adalah ilmu yang mempelajari mengenai bagaimana cara mengamankan suatu informasi. Pengamanan ini

dilakukan dengan mengenkrip informasi tersebut dengan suatu kunci khusus (Didi Surian, 2006). Dan Menurut Request for Comments (RFC), kriptografi merupakan cabang ilmu matematika yang berhubungan dengan transformasi data untuk embuatnya artinya tidak dapat dipahami (untuk menyembunyikan maknanya atau isi dari sebuah data), mencegahnya dari perubahan tanpa izin, atau mencegahnya dari penggunaan yang tidak sah. Jika transformasinya dapat dikembalikan, kriptografi juga bisa diartikan sebagai proses mengubah kembali data yang terenkripsi menjadi bentuk yang dapat dipahami. Jadi dapat disimpulkan bahwa kriptografi dapat diartikan sebagai cabang ilmu matematika untuk menjaga kerahasiaan informasi dengan metode teknik matematika yang mencakup, kerahasiaan, integritas data, autentifikasi, dan non repudiasi. 2.2 Tujuan Kriptografi Menurut Alfred Menezez, Scott Vanstone dan Paul Oorschot dalam bukunya yang berjudul Handbook of Applied Cryptography (2006:4) terdapat empat tujuan mendasar dari kriptografi sebagai bentuk dari keamanan informasi, yaitu : a. Kerahasiaan (Confidentiality) Menjaga kerahasiaan informasi dari semua pihak yang tidak berwenang yang mungkin mencoba membaca data atau informasi tersebut. b. Keutuhan Data (Integrity) Informasi tetap utuh atau tidak ada perubahan dalam proses pengiriman sampai informasi diterima oleh penerima. c. Autentikasi (Authentication) Pengenalan identitas baik secara kesatuan sistem maupun informasi itu sendiri untuk menjamin keaslian sumber. d. Anti Penyangkalan (Nonrepudiation) Mencegah terjadinya penolakan atau penyangkalan terhadap informasi yang dikirim atau diterima. 2.3 Jenis Jenis Algoritma Kriptografi Algoritma Kriptografi dikatagorikan berdasarkan kunci yang dipakainya (Dony, 2008), yaitu : a. Algoritma Simetri Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama atau satu kunci untuk kegiatan enkripsi dan dekripsi (Dony, 2008). Jika mengirim pesan menggunaka algoritma ini maka si penerima harus diberitahu atau mengetahui kunci dari pesan yang telah di enkripsi oleh algoritma tersebut agar dapat di deskripsikan oleh penerima dari si pengirim. Secara umum, cipher yang termasuk dalam kriptografi simetri beroperasi dalam mode blok (block cipher), yaitu setiap kali enkripsi atau dekripsi dilakukan terhadap satu blok data yang berukuran tertentu , atau

beroperasi dalam mode aliran (stream cipher), yaitu setiap kali enkripsi atau dekripsi dilakukan terhadap 1 bit atau 1 byte data. Gambar 2.3.1 Algoritma Simetris Berikut contoh contoh algoritma kriptogtafi yang memakai kunci simetris, ialah : 1. Blowfish 2. Twofish 3. DES (Data Encryption Standard) 4. RC2, RC4, RC5, RC6 5. IDEA (International Data Encryption Algorithm) 6. AES (Advanced Encryption Standard), dan sebagainya b. Algoritma Asimetris 4Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda (Dony, 2008). Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu: (Dony, 2008) 1. Kunci umum (public key): Kunci yang boleh semua orang tahu (dipublikasikan). 2. Kunci rahasia (private key): Kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang). Kunci kunci tersebut saling berhubungan atau berkaitan dengan satu sama lain. Dengan adanya kunci public si pengirim atau orang tersebut dapat mengenkripsikan pesan tapi tidak dengan mendeskripsikan pesan tersebut, karena berbeda kunci. Dan hanya si penerima atau orang yang mempunyai kunci pribadilah yang dapat mendeskripsikan pesan tersebut. Akan tetapi algoritma asimetris dapat melakukan pengiriman pesan yang lebih aman dari pada algorita simetris karena 37mempunyai dua kunci yaitu kunci publik dan kunci pribadi. Gambar 2.3.2 Algoritma Asimetris Berikut contoh contoh algoritma kriptografi yang memakai kunci asimetris, ialah : 1. DF (Diffle-Hellman) 2. DSA (Digital Signature Algorithm) 3. RSA 4. Kriptografi Quantum 5. ECC (Elliptic Curve Cryptography) dan sebagainya c. Fungsi Hash (Hashing Function) Fungsi hash 3adalah fungsi yang melakukan pemetan pesan dengan panjang sembarang ke sebuah teks khusus dengan panjang tetap (Santi Sulastri, 2018). Fungsi hash juga sering disebut dengan fungsi hash satu arah (one way 4function), message digest, fingerprint, fungsi kompresi dan message authentication code (MAC). Macam macam algoritma yang memakai fungsi hash ialah: 1. MD4, MD5 2. SHA-1, SHA-256, SHA-512 3. Snefru 4. N-Hash 5. RIPE-MD, dan sebagainya 2.4 Enkripsi dan Deskripsi Dalam kriptografi terdapat proses didalamnya yang disebut sebagai proses enkripsi dan deskripsi. Proses penyandian pesan asli (plain text) menjadi pesan yang tidak dapat dibaca (chipper text) adalah enkripsi (Pandi Barita, 2018),

sedangkan kebalikan dari proses enkripsi ialah deskripsi yaitu mengembalikan pesan yang sudah disandikan tersebut dan tidak dapat terbaca menjadi pesan aslinya yang dapat dibaca kembali, proses tersebut adalah deskripsi (Komariah Fitri, 2018). Pesan tersebut dapat data atau informasi yang berbentuk teks, dokumen, gambar serta suara yang bersifat penting dan rahasia. Sistem yang mendasari terjadinya sebuah proses enkripsi dan dekripsi ialah hubungan antara dua himpunan yaitu yang berisi sebuah elemen pesan asli (plaintext) dan sebuah pesan yang berisi elemen pesan sandi (ciphertext). Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. P adalah notasi yang digunakan untuk plaintext, C adalah ciphertext, E adalah fungsi Encryption dan D adalah fungsi Decryption. Sedangkan untuk kunci dapat dinotasikan sebagai K atau Key. Berikut gambar proses enkripsi dan dekripsi : Gambar 2.4 Proses Enkripsi dan Deskripsi 2.5 Advanced Encryption Standard (AES-256) Ilmu kriptografi terus berkembang dari jaman ke jaman, usaha untuk menjaga kerahasiaan suatu informasi telah ada sejak jaman dahulu kala. Julius Cesar, kaisar romawi telah menggunakan metode enkripsi sederhana pada jamannya untuk mengamankan sebuah informasi yang diterimannya atau informasi yang dikirimnya yang bersifat rahasia. Teknik kriptografi enkripsi yang dia gunakan pada saat itu sangat sederhana, yaitu menggeser setiap karakter dalam pesannya dengan nilai tertentu. Cara tersebut cukup aman pada jamannya, akan tetapi pada teknik enkripsi tersebut dinilai tidak aman lagi karena kemampuan komputasi komputer semakin perkembang dan sangat mudah dipecahkan. Hingga tahun 1990-an, algoritma kriptografi terus berkembang pada saat itu algoritma DES sangat popular yaitu algoritma kriptografi Data Encryption Standard. DES termasuk algoritma enkripsi chipper block, ia adalah cikal bakal algoritma AES lahir. Lalu dari tahun ketahun sering berkembangnya teknologi, algoritma DES yang memiliki 56 bits tidak lagi mampu mempertahankan eksistensinya dan tidak lagi memadai. Pada tahun 1997 kontes pemilihan standard algoritma kriptografi baru pengganti DES dimulai. Diikuti para crytopgrapher dari seluruh dunia, ada 21 peserta di dalam kontes tersebut. Algoritma Rijndael yang keluar sebagai pemenang kontes pencarian algoritma kriptografi yang baru pengganti DES yang di selanggarakan oleh NIST (National

Institute of Standard and Technology) milik pemerintah Amerika Serikat. Algoritma kriptografi Rijndael didesain oleh dua pemudah asal Belgia yang bernama Vincent Rijmen dan John Daemen. Karena algoritma Rijndael ini yang paling memenuhi kriteria pada kontes pencarian algoritma pengganti DES pada saat itu. Rijndael diumumkan oleh NIST sebagai Standar Pemrosesan Informasi Federal (FIPS) publikasi 197 (FIPS 197) pada tanggal 26 November 2001 setelah proses standarisasi selama 5 tahun, di mana ada 15 desain enkripsi yang disajikan dan dievaluasi, sebelum Rijndael terpilih sebagai yang paling cocok. algoritma Rijndael terpilih sebagai algoritma kriptografi yang selain aman juga efisien dalam implementasinya dan dinobatkan sebagai Advanced Standard Encryption. AES efektif menjadi standar pemerintah Federal pada tanggal 26 Mei 2002 setelah persetujuan dari Menteri Perdagangan (Hadir Fajar, 2017). Algoritma AES mendukung berbagai variasi ukuran kunci yang digunakannya. Jenis ukuran kunci yang algoritma AES terbagi tiga, yaitu AES-128, AES-193 dan AES-256. Perbedaan jenis ukuran block dan kunci yang algoritma AES miliki yaitu karena perbedaan ukuran kunci yang akan menentukan jumlah proses yang harus dilalui pada saat pengenkripsian dan pengdeskripsian atau lebih mudahnya dan dapat disimpulkan perbedaan pada banyaknya round atau putaran yang dipakai pada proses enkripsi dan deskripsi. Akan tetapi dari ketiga ukuran kunci yang AES miliki, ketiganya memiliki ukuran blok yang sama, yaitu 182 bit. Berikut ini adalah tabel yang memperlihatkan jumlah putaran dalam pemprosesannya yang harus diimplemetasikan pada masing masing panjang kunci yang AES miliki. Tabel 2.5 Perbadingan Jumlah Key dan Round Algoritma AES Jumlah Key (Nk) Ukuran Block (Nk) Jumlah Putaran (Nr) AES-128 128 bit 128 bit 10 AES-192 192 bit 128 bit 12 AES-256 256 bit 128 bit 14 2.6 AES Mode CBC (Chiper Block Chaining) Algoritma kriptografi simetris Advanced Encryption Standard (AES) selain mempunyai beberapa panjang kunci yaitu 128, 192 dan 256 juga mempunyai beberapa mode operasi, walaupun AES hanya bisa meng-enkripsi blok dengan panjang 128 bits (16 bytes) tetapi untuk meng-enkripsi file yang lebih panjang mode operasi penting dipertimbangkan. Salah satu mode operasi AES yaitu CBC atau yang disebut Chiper Block Chaining. Pada algoritma block chiper seperti AES ini, plaintext atau pesan mentah

yang masuk untuk diproses dengan panjang yang tetap yaitu n, akan tetapi jika ukuran datanya terlalu panjang maka dilakukan pemecahan data data tersebut menjadi beberapa blok blok dengan ukuran yang lebih kecil dan sama. Maka dari itu mode operasi diperlukan, salah satunya ialah CBC. Pada CBC, rangkaian bit-bit pada plaintext dibagi menjadi blok blok bit dengan panjang yang sama (Henry. 2016). Mode CBC memerlukan IV (initialization vector) untuk menggabungkan dengan plaintext pertama. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini. Gambar 2.6 Mode Operasi Chiper Block Chaining 2.7 Padding Pada proses penyandian blok, plaintext dengan ukuran data yang besar akan di pecah-pecah menjadi blok blok yang lebih kecil. Akan tetapi jika dalam pemecahaan tersebut menghasilkan blok data yang kurang dari jumlah data dalam blok tersebut maka akan dilakukan proses padding yaitu penambahan beberapa bit. Padding dilakukan dengan mengisi byte bernilai N bila dibutuhkan padding sebanyak N byte. Sebagai contoh, bila dibutuhkan padding 3 byte, maka padding berisi '03 03 03', bila dibutuhkan padding 5 byte, maka padding berisi '05 05 05 05 05' (Awang Harsa, 2016). Untuk menggunakan CBC, padding diperlukan. Dalam aplikasi Secret Fichier, yaitu aplikasi enkripsi dan deskripsi yang penulis buat untuk lembaga Pendidikan Universitas Bhayangkara Jakarta Raya Kelas AES-256 dari bahasa C# yang digunakan ini menggunakan padding PKCS7. Padding PKCS7 adalah generalisasi padding PKCS5 (juga dikenal sebagai padding standar). Padding PKCS7 bekerja dengan menambahkan N byte dengan nilai chr (N), dimana N adalah jumlah byte yang dibutuhkan untuk membuat blok akhir data berukuran sama dengan ukuran blok. 2.8 Key Schedule Key schedule atau penjadwalan kunci dilakukan dengan tujuan mendapatkan kunci ronde atau RoundKeys yang akan digunakan untuk proses enkripsi dan dekripsi pada ronde rondenya, tepatnya pada tahap transformasi AddRoundKey. Tanpa proses pembangkitan kunci maka proses enkripsi dan enkripsi tidak akan berjalan sebagaimana mestinya. Berikut ini adalah gambar proses dari key schedule : Gambar 2.8 proses Key Schedule 2.8.1 Pengertian RotWord, SubWord, dan Rcon a. RotWord adalah proses pergeseran blok paling atas ke paling bawah, pada kolom terakhir kunci ronde sebelumnya. b. SubWord adalah pergantian nilai yang terdapat

didalam blok kolom array menggunakan tabel S-Box. c. Rcon adalah array konstant untuk proses perhitungan menggunakan tabel berikut : Tabel 2.8.1 Tabel Rcon KR2 KR4 KR6 KR8 KR10 KR12 KR14 01 02 04 08 10 20 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2.9 Proses Enkripsi AES-256 Pada peng-enkripsian AES-256 ini memiliki panjang blok 128 bit dengan panjang kunci 256 bit dan round sebanyak 14. Karena algoritma AES menggunakan tahapan berulang ulang pada prosesnya, tergantung panjang kunci yang dipilih yang disebut round. Proses enkripsi AES-256 ialah : a. Pada tahap awal proses enkripsi, teks asli diubah menjadi sebuah state dengan operasi XOR. Lalu sebelum round ke-1 dimulai, blok teks asli digabungkan dengan kunci round ke-0, pada transformasi inilah disebut dengan AddRoundKey. Transformasi AddRoundKey pada proses enkripsi pertama kali pada round=0 dan selanjutnya round = round + 1. b. Putaran sebanyak Nr-1 13kali. Proses yang dilakukan pada setiap putaran adalah : 1) SubBytes : Byte diganti dengan byte lain mengikuti kotak substitusi tetap atau sering disebut sebagai transformasi subtitusi. Kotak substitusi ini ditentukan dalam standar AES. Berikut ini merupakan tabel subtitusi S-Box. Tabel 2.9.1 Tabel S-Box 2) ShiftRows : Pada tahap ini disebut dengan tahap transformasi permutasi dengan baris di blok digeser ke kiri mengikuti aturan tetap. Baris pertama tetap di tempatnya, baris kedua digeser satu, baris ketiga digeser dua dan seterusnya. Gambar 2.9.2 Tahapan ShiftRows pada proses enkripsi 3) MixColumns : Pada tahap ini disebut dengan tahap tranformasi pengacakan, dalam tahap ini kolom diacak mengikuti aturan fixe yang ditentukan dalam standar. Operasi campuran ini setara dengan perkalian matriks dengan matriks tetap pada operasi ini terjadi perkalian polynomial didalamnya. Gambar 2.9.3 Tahapan MixColumns pada proses enkripsi 4) AddRoundKey : Pada tahap ini disebut dengan tahap transformasi penambahan kunci. 40melakukan XOR antara state sekarang dengan round key. c. Pada Final Round, yaitu round ke-Nr dilakukan transformasi sama seperti round yang lainnya, akan tetapi tanpa tahap MixColumns yaitu hanya SubBytes, ShiftRows dan AddRoundKeys. Alur tahapan proses enkripsi AES-256 dapat digambarkan seperti gambar dibawah ini : Gambar 2.9 Alur Proses Enkripsi AES -256 2.10 Proses Deskripsi AES-256 Proses deskripsi

yaitu kebalikan dari proses enkripsi. Teks yang sudah di enkripsi atau teks bersandi (chipper text) dapat diubah kembali menjadi teks yang bisa terbaca (plain text) dan diterapkan ke arah yang berlawan yang disebut inverse cipher. Berikut proses deskripsi algoritma AES-256 : a. Proses pada putaran pertama : 1) AddRoundKey : Transformasi AddRoundKey pada proses deskripsi diperlukan penambahan roundkey pada state dengan operasi XOR. Pada tahap AddRoundKey pertama kali pada round=14 dan selanjutnya round = round + 1. 2) Inverse ShiftRows : Pada tahap ini baris di blok digeser ke kanan mengikuti aturan tetap. Berikut gambaran transformasi Inverse ShiftRows. Gambar 2.10.2 Tahapan Inverse ShiftRows pada proses deskripsi 3) Inverse Subbytes : pada proses kebalikan dari proses subtitusi bytes pada enkripsi. Tiap elemen pada state di konversi ke dalam tabel Inverse S-box. Berikut ini tabel Inverse S-box. Tabel 2.10.3 Tabel Inverse S-Box b. Lalu pada proses deskripsi round sebanyak 13 kali. Tahapan yang dilakukan pada proses round ialah : 1. AddRoundKey : diperlukan penambahan roundkey 27pada state dengan operasi XOR. 2. Inverse MixColumns : mengalikan setiap kolom array state dengan matriks yang sudah ditetapkan. Berikut gambaran Inverse MixColums. Gambar 2.10.4 Tahapan Inverse MixColumns pada proses deskripsi 3. Inverse ShiftRows : Pergeseran basris-baris array state ke kanan. 4. Inverse SubBytes : Proses kebalikan dari proses subtitusi bytes pada enkripsi. Tiap elemen pada state di konversi ke dalam tabel Inverse S-box. c. Final round : proses untuk round terakhir yaitu AddRoundKey. Alur tahapan proses enkripsi AES-256 dapat digambarkan seperti gambar dibawah ini : Gambar 2.10 Alur Proses Deskripsi AES-256 2.11 Secure Hash Algorithm (SHA-256) Dalam enkripsi data dikenal suatu fungsi yang di sebut fungsi hash atau hashing. Fungsi hash adalah 21adalah fungsi yang menerima masukan string yang panjangnya sembarang dan dikonversikan menjadi string dengan keluaran yang panjangnya tetap (Santi Sulastri, 2018). Fungsi hash yang berbeda akan menghasilkan output-output yang berbeda ukuran, tetapi kemungkinan ukuran output dari masing-masing algoritma hashing selalu konstan. Sebagai contoh, algoritma SHA-256 hanya akan menghasilkan message digest 256 bit, sedangkan SHA-1 selalu akan menghasilkan digest 160-bit. Fungsi hash satu arah atau yang sering disebut dengan one

way hash function dimana hasil hash atau hash value sangat sukar dikembalikan ke nilai hash awal. Persamaan fungsi hash dapat dinyatakan sebagai berikut Dimana H sebagai fungsi hash dengan masukan pesan berupa string yang disimbolkan sebagai variable M dan menghasilkan suatu nilai string pula yang disimbolkan berupa variable h. Fungsi hash mempunyai beberapa nama lain, yaitu bisa disebut pula sebagai fungsi kompresi atau kontraksi, fingerprint, message integrity check (MIC), manipulation detection code (MDC) dan cryptographic checksum. Ada beberapa fungsi hash satu arah atau hash function one way antara lain MD2, MD4 dan MD5 (Message Digest) dan fungsi hash Secure Hash Algorithm atau yang sering disebut SHA. Keamanan SHA-256 pernah diuji sebelumya yang di lakukan oleh peneliti jurnalnya yang berjudul Implementasi KeyedHash Message Authentication Code pada Sistem Keamanan Rumah (Ichwan, 2016). Keamanan SHA-256 pernah diuji pada penelitian yag dilakukan oleh peneliti [8]. Penggunaan SHA-256 yang digabungkan dengan algoritma Message Authentication Code (MAC) dari hasil pengujian 64 round menghasilkan nilai rata-rata avalanche effect (AE) sebesar 85,9%. Ini menunjukan bahwa keluaran SHA-256 memiliki tingkat pengacakan yang bagus. SHA-256 dirancang oleh The National Institute of Standards and Technology (NIST) pada tahun 2002.Fungsi hash SHA-256 merupakan fungsi SHA dengan ukuran digest 256 bit pada versi SHA-2. Ada beberapa versi SHA yaitu SHA-0, SHA-1, SHA-2, dan SHA-3. SHA-256 menggunakan enam fungsi logika yang merupakan kombinasi dasar antara lain seperti XOR, OR, AND dan pergeseran bit kekanan (shift right), dan rotasi bit kekanan (rotate right). SHA-256 mengubah pesan masukan ke dalam message digest 256 bit. Berdasarkan Secure Hash Signature Standar. Pesan masukan yang panjangnya lebih pendek dari 264 bit, harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah message digest 256 bit. Lebih mudahnya plaintext diproses dengan fungsi hash lalu keluaran tersebut menjadi hash text yang tidak dapat terbaca. Berikut ini gambaran dari algoritma hash, yaitu : Gambar 2.11 Alur Algoritma Hashing 2.12 Proses SHA-256 Pada SHA-256 plaintext diubah menjadi masukan ke dalam message digest 256 bit berdasarkan ketetapan Secure Hashing Standard. Plaintext yang panjangnya kurang dari 264 bit harus dioperasikan kedalam

blocksize 512 bits sehingga menjadi sebuah message digest sebesar 256 bits. Berikut ini penjelasan dari tahapan proses dari SHA-256, yaitu : a. Padding Bits : Pada tahap ini fungsi hashing dimulai dari penambahan bit ke pesan asli (plain text), sehingga panjangnya akan sama dengan panjang standar yang diperlukan oleh fungsi hash. Jumlah bit yang ditambahkan pada pesan asli dihitung sedemikian rupa sehingga setelah penambahan bit, panjang pesan asli harus kurang dari 64 bit dari kelipatan 512 bits. Bit yang ditambahkan ke pesan, harus dimulai dengan '1' dan bit yang ditambahkan berikutnya harus '0' sampai tepat pada 64 bit dan kurang dari kelipatan 512. b. Length Bits : Lalu pada tahap kedua penambahan bit sebelumnya yang setara 64 bits kedalam pesan keseluruhan untuk membuat semuanya menjadi kelipatan 512. c. Inisialisasi buffer : Pada tahap ini permulaan melakukan perhitungan yaitu masing masing blok 512 bit tadi dipecah menjadi 16 buah bit word 32 bit yang mana nantinya diperluas menjadi 64 word yang diberi label dengan aturan yang di tetapkan oleh Secure Hashing Standard. Lalu masing masing label tadi kemudian di proses dengan fungsi hash SHA-256. Didalam inti prosesnya, algoritma SHA-256 membuat 8 variabel yang diberikan nilai awal a=h0 (0) – h=h7(0) diawal masing masing fungsi hash. Nilai nilai awal tersebut di tunjukan sebagai berikut : d. Fungsi Kompresi : Dan pada tahap ini, SHA melakukan perhitungan sebanyak 64 kali putaran untuk setiap bloknya dan keluaran yang diperoleh diumpankan sebagai masukan untuk putaran operasi berikutnya. Dan delapan varibel tadi nilainya akan terus berganti selama perputaran sebanyak 64 kali. e. Output : lalu pada setiap putaran berfungsi sebagai input untuk putaran berikutnya dan proses ini terus berlanjut hingga bit terakhir dari pesan tetap ada dan hasil putaran terakhir untuk n last bagian blok pesan akan memberi kita hasil yaitu hash untuk keseluruhan pesan. Panjang keluarannya adalah 256 bit. 2.13 Unified Modelling Language (UML) Unified Modelling Language atau kepanjangan dari UML adalah sebuah bahasa pemodelan yang telah menjadi standar yang dirancang khusus untuk pengembangan, analisis sistem berorientasi objek dan desain dari sebuah perangkat lunak (Hadi Fajar, 2017). UML pertama kali dikembangkan oleh Grady Booch, Jim Rumbaugh, dan Ivars Jacobson pada

pertengahan tahun 1994. Dalam penelitian ini penulis menggunakan diagram yang difenisikan dalam UML, yang telah menjadi bahasa pemodelan standard untuk pemodelan berorientasi objek. UML terdiri dari beberapa diagram,

dalam laporan ini penulis menggunakan Usecase Diagram untuk menggambarkan fungsionalitas dari aplikasi yang dibuat oleh penulis dan Activity

Diagram untuk menggambarkan urutan aktivitas di dalam aplikasi dari proses kerja yang ada serta Sequence Diagram yang menggambarkan alur interaksi objek dalam urutan atau rangkaian waku di dalam sebuah sistem. 2.14 Use Case Use case diagram merupakan bentuk dari pemodelan untuk menunjukan sebuah fungsionalitas dalam suatu sistem yang dibuat untuk mencapai tujuan tertentu. Use case diagram adalah sebuah diagram yang menunjukan hubungan antara actors dan use cases yang di gunakan untuk menganalisa dan desain sebuah sistem (The Elements of UML 2.0: Scott W.Ambler, 2005:33). Use case dapat memberikan gambaran umum tentang seluruh atau sebagian kebutuhan penggunaan untuk suatu sistem dan mengkomunikasikannya dalam ruang lingkup. Berikut simbol simbol yang digunakan dalam Use Case Diagram yaitu ; Tabel 2.14 Tabel Use Case Diagram ACTOR Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari actor adalah gambar orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor. USE CASE Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesar antar unit atau actor biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case. ASOSIASI/ASSOCIATION Komunikasi antara actor dan use case yang berpartisipasi pada use case atau use case memiliki interraksi dengan actor. EKSTENSI/EXTEND Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan memiliki nama depan yang sama dengan use case yang di tambahkan GENERALISASI/GENERALIZATION Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. MENGGUNAKAN/INCLUDE Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan

memerlukan use case ini untuk menjalankan fungsional atau sebagai syarat dijalankan use case ini 2.15 Activity Diagram Activity Diagram adalah suatu diagram yang menggambarkan konsep data atau workflow (aliran kerja), aksi yang terstruktur serta dirancang dengan baik dalam suatu sistem (Hendini, 2016). 26Simbol-simbol yang digunakan dalam Activity Diagram yaitu ; Tabel 2.15 Tabel Activity Diagram Start Point, diletakkan pada pojok kiri atas dan merupakan awal aktivitas End Point, merupakan akhir aktivitas Activities, menggambarkan suatu proses / kegiatan bisnis 20Fork / percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu Join / penggabungan atau rake, digunakan untuk menunjukkan adanya dekomposisi Decision Points, menggambar pengambilan keputusan true atau false 12Swimlane, pembagian activity diagram untuk menunjukkan siapa melakukan apa 2.16 Sequence Diagram Sequence Diagram adalah sebuah diagram yang menggambarkan kolaborasi objek pada use case yang saling berinteraksi antar elemen dari suatu kelas. Kegunaan dari Sequence Diagram sendiri ialah untuk menunjukan rangkaian pesan yang dikirim antara objek dengan objek. Simbol simbol yang digunakan dalam Sequence Diagram yaitu ; Gambar 2.16 Tabel Sequence Diagram 14Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun 31simbol dari actor adalah gambar orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation Message, symbol mengirim pesan antar class Recursive, menggambarkan 12pengiriman pesan yang dikirim untuk dirinya sendiri Activation,mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi 2.17 Tinjauan Pustaka Berikut penjabaran penelitian terdahulu pada tabel 2.17 dibawah ini. Nama Judul Metode Hasil Penelitian Sandi Yusmantoro, Edy Hermansyah, Rusdi Efendi (2014) Rancang Bangun Aplikasi Pengaman Keaslian Surat Ijin Tempat Usaha Menggunakan Algoritma Elgamal dan Secure HashAlgorithm 256 (Studi Kasus : Badan Pelayanan Perizinan Terpadu (BPPT) Kota

Bengkulu). Algoritma Elgamal, Secure Hash Algorithm 256 Membangun aplikasi pengamanan keasilan surat ijin tempat usaha di Badan Pelayanan Perizinan Terpada Kota Bengkulu Muammar Renaldy (2015) Implementasi Kriptografi Pada Diary Berbasis Mobile Android Dengan Menggunakan Metode AES – 128 (Advanced Encryption Standard – 128) dan SHA – 1 (Secure Hashing Algorithm – 1). Algoritma AES-128, SHA-1 Membangun aplikasi berbasis mobile android untuk pengenkripsian pesan teks. Hadi Fajar Wiguno (2017) Aplikasi Pengamanan File Dan Pesan Teks Menggunakan Algoritma AES 256 Dan SHA 256 Berbasis Android. Algoritma AES-256, SHA-256 Membangun aplikasi berbasis android untuk pengenkripsian file dan pesan teks. Ratno Prasetyo dan Asep Suryana (2016) Aplikasi Pengamanan Data dengan Teknik Algoritma Kriptografi AES dan Fungsi Hash SHA-1 Berbasis Desktop. Algoritma AES, SHA-1 Membangun apikasi berbasis Desktop dengan AES dan SHA-1 Santi Sulastri1 dan Riana Defi Mahadji Putri (2018) Implementasi Enkripsi Data Secure Hash Algorithm (SHA-256) dan Message Digest Algorithm (MD5) pada Proses Pengamanan Kata Sandi Sistem Penjadwalan Karyawan. SHA-256, MD5 Pengimplementasian enkripsi data dengan fungsi hash untuk pengamanan kata sandi sistem penjadwalan karyawan. Pandi Barita Nauli Simangunsong Dan Komariah Fitri (2018) Perancangan Aplikasi Pengamanan Citra Bewarna Dengan Algoritma RSA RSA Pengenkripsian Citra Menggunakan Algoritma RSA BAB III METODOLOGI PENELITIAN 3.1 Objek Penelitian Pengumpulan informasi dan data sekunder dilakukan pada Fakultas Ilmu Komputer Program Studi Informatika Universitas Bhayangkara Jakarta Raya, Jl. Raya Perjuangan Bekasi Utara, Kota Bekasi, Jawa Barat 17121, Indonesia. Pada divisi administrasi untuk pengamanan data file. 3.2 Kerangka Penelitian Diagram alur ini menunjukan gambaran kerangka penelitian yang dilakukan penulis tahapan demi tahapan sebagai pedoman dalam penelitian agar yang ingin dicapai tidak menyimpang dari tujuan sehingga mendapatkan hasil yang relevan. Serta menjadi solusi untuk pemecahan masalah di lembaga pendidikan Universitas Bhayangkara Jakarta Raya. Tahap tahap yang akan dilalui dalam metodologi penelitian dapat dilihat pada gambar dibawah ini ;

Gambar 3.2. Kerangka Penelitian Untuk memahami kerangka penelitian pada gambar

diatas, berikut penjelasan secara terperinci mengenai urutan tahapan kerangka penelitian diatas. 1. Penelitian Pendahuluan Penelitian pendahuluan merupakan tahap awal yang dimaksudkan untuk mencari tahu permasalah yang terjadi di objek penelitian. Mengerti masalah yang ada dan menghubungkannya dengan algoritma untuk menjadi bahan penelitian guna memecahkan masalah yang terjadi. Pada penelitian pendahuluan ini, proses wawancara juga dilakukan oleh pihak pihak terikait untuk keberlangsungannya proses penelitian ini. 2. Definisi Permasalahan Setelah melakukan tahapan penelitian pendahuluan, selanjutnya pada tahapan ini dilakukan pendefinisian permasalahan. Pendefenisian permasalahan mencakup identifikasi masalah yang ada di Universitas Bhayangkara serta permasalahan yang ingin diangkat sebagai bahan penelitian serta batasan batasan apa yang dinyatakan dalam penelitian. Selain itu tentunya tujuan dan manfaat penelitian juga didefinisikan, baik untuk Universitas Bhayangkara atau untuk peneliti. File sendiri dapat bersifat pribadi dan sensitive sesuai dengan persepsi setiap individu. File juga memiliki ragam jenis ekstensi yaitu file dokumen, file data keuangan, file gambar dan jenis file lainnya. Pentingnya menjaga suatu file penting bagi suatu lembaga agar tersimpan dengan baik dan terjaga kerahasiannya. Berdasarkan kebutuhan lembaga tersebut untuk menjaga keamanan datanya, maka dari itu penelitian ini menerapkan algoritma kriptografi kedalam pembuatan aplikasi enkripsi dan dekripsi berbasis desktop untuk Universitas Bhayangkara agar dapat mengamankan data mereka dengan baik. 3. Pengumpulan Data Tahapan ini adalah mengumpulkan data data yang diperlukan guna menjadi bahan penelitian untuk kemudian dijadikan input data pada aplikasi yang dibuat oleh penulis dalama penelitian ini. Pengumpulan data yang dimaksud ialah mencari tahu format atau ekstensi jenis file apa yang dikiranya penting bagi Universitas Bhayangkara untuk datanya diamankan. Pengumpulan data secara langsung ialah dengan mewawancarai pihak terkait dengan menanyakan jenis data apa yang biasanya bersifat rahasia untuk dijaga keamanannya, yaitu apakah data tentang keuangan yang berformat excel atau ekstensi file data rahasia lainnya. Ada 3 teknik pengumpulan data, yaitu : a. Pengamatan Langsung (Observasi) b. Data Sekunder c. Studi Pustaka 4. Analisa dan

Perancangan Aplikasi Setelah melalui tahapan sebelumnya dan merunjuk kepada inti dari permasalahan, maka di buatlah aplikasi yang diharapkan dapat membantu Universitas Bhayangkara untuk memecahkan permasalahan yang ada yaitu pengamanan data. Dimulai dari analisa sistem yang meliputi tahapan sistem yang akan dibuat sebagai konsep, objek dan keterkaitannya serta analisa solusi dari algoritma dan kebutuhan aplikasi. Analisa ini ditranlasikan kedalam bentuk pemodelan UML yaitu use case diagram, activity diagram serta sequence diagram sebagai bentuk dari peracangan sebuah aplikasi yang akan dibuat. 5. Implementasi Pada tahap implementasi ini dilakukan pembuatan modul modul yang telah dirancang dalam tahap perancangan kedalam bahasa pemrograman tertentu. Tahap implementasi ini dibutuhkan suatu alat dan bahan yang digunakan sebagai perangkat pendukung demi kelancaran tahap implementasi program. Karena sebelum program diimplementasikan, maka program harus bebas dari kesalahan. Kesalahan yang dimaksud ialah kesalahan program yang mungkin terjadi yaitu kesalahan penulisan (coding), kesalahan proses atau kesalahan logika. Dalam hal ini penulis mengimplementasikan bahasa pemrograman C# (c sharp) untuk pembuatan aplikasi enkripsi dan dekripsi berbasis desktop. Berikut perangkat yang digunakan terdiri dari : a. Perangkat Keras (hardware) Perangkat keras yang digunakan penulis untuk membuat aplikasi ini antara lain adalah dengan sebuah laptop dengan spesifikasi mempunyai processor Intel Core i7-10510U dengan RAM 8 GB dan Solid State Drive 1000 GB serta VGA Nvidia MX250 2GB b. Perangkat Lunak (software) Perangkat lunak yang digunakan dalam membangun aplikasi ini adalah system operasi windows 10 64 bit, Visual Studio 2019 (.NET Core). 6. Pengujian Aplikasi Tahap selanjutnya adalah testing atau pengujian aplikasi. Setelah aplikasi selesai dirancang, maka dilakukan pengujian untuk melihat apakah aplikasi yang telah dirancang dapat berjalan dengan baik den memberikan hasil keluaran yang baik. Jika aplikasi masih memiliki error atau bug dalam pelaksanaanya maka akan dievaluasi ulang kembali. Pengujian aplikasi meliputi empat parameter yaitu keamanan, integritas data, kecepatan proses dan perubahan kapasitas file. Pengujian ini dilakukan dengan langkah sebagai berikut : a. Blackbox adalah sebuah metode yang digunakan untuk

menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah input diterima dengan benar dan output yang dihasilkan telah sesuai dengan yang diharapkan. b. Bruteforce Attack adalah serangan untuk mengungkap kunci atau password dengan mencoba semua kemungkinan kunci sampai akhirnya menemukan kunci yang tepat. c. Mencari kekurangan dari system aplikasi yang telah dibangun, hal ini dilakukan setelah aplikasi berhasil dijalankan dan menemukan kelamahan secara konsep yang akan dimasukan dalam kesimpulan dan saran. 7. Pembahasan Pada tahapan ini pembahasan dan analisa dilakukan adalah mencakup hasil keluaran dari penerapan algoritma. Pada tahapan ini pembahasan pada penerapan algoritma Advanced Encryption Standard (AES-256) dan Secure Hash Algorithm (SHA-256) dipengaplikasiannya akan dijelaskan pada tahap ini. 8. Kesimpulan dan Saran Berdasarkan hasil tahapan sebelumnya sampai pada tahapan pembahasan dan analisa, maka dapat ditarik kesimpulan dari hasil pembahasan serta hasil pengujian untuk menjawab semua pertanyaan yang ada pada rumusan masalah pada penelitian ini. Pada tahapan ini sejumlah saran juga akan diberikan untuk mengatasi permasalahan yang ditemukan, sehingga dapat menjadi masukan dan manfaat bagi Universitas Bhayangkara untuk kedepannya. 3.3 Metode Penelitian Metodologi yang digunakan dalam penulisan penelitian ini adalah dengan metode pengumpulan data dimana untuk mendapatkan data dan bahan penelitian yang sesuai harapan, teknik pengumpulan data yang digunakan ada tiga jenis diantaranya sebagai berikut : 1. Pengamatan Langsung (Observasi) Yaitu suatu cara penelitian atau metode pengumpulan data dengan jalan mengadakan pengamatan secara langsung pada objek penelitian yang merupakan sumber data. Pada kesempatan kali ini penulis mengamati objek penelitian yang bertempat di Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya. 2. Data Primer Data ini diperoleh melalui wawancara (interview) dengan pihak yang terkait di Fakultas Ilmu Komputer Universitas Bhayangkara dan menanyakan bagaimana mereka menyimpan suatu data yang bersifat rahasia, apakah ada penelitian sebelumnya yang bertemakan algoritma kriptografi untuk pengamanan data dan sebagainya. 3. Studi Literatur Studi Literatur ini merupakan cara untuk mendapatkan

data-data secara teoritis sebagai bahan penunjang dalam penyusunan laporan penelitian dengan membaca buku literature dari perpustakaan serta dari internet untuk mendukung teori yang ada dan maupun dari buku buku referensi lainnya untuk melengkapi data-data yang ada. Studi literatur sendiri merupakan metode untuk memeperoleh informasi yang berhubungan dengan latar belakang dan rumusan masalah, informasi yang dikumpulkan berupa konsep dan teori yang berkaitan dengan judul penilitian penulis yaitu algoritma kriptografi 10Advanced Encryption Standard (AES) dengan mode CBC dan Secure Hash Algorithm (SHA) khususnya yang berhubungan dengan teori enkripsi dan dekripsi suatu file. Langkah yang dilakukan penulis guna memperoleh informasi ialah dengan Studi Pustaka dan Diskusi. Studi Pustaka yaitu langkah untuk melakukan pencarian, menemukan dan mengumpulkan informasi yang sesuai dengan kasus, referensi yang didapat berupa buku, jurnal, artikel dan penelitian terdahulu tentang algoritma AES dengan mode CBC dan SHA. Serta diskusi dengan pakar ahli dibidang sekuriti dengan maksud mencari solusi dan masukan tentang permasalahan yang berhubungan dengan kesulitan yang ada didalam proses pembuatan aplikasi serta pemahaman implementasi kriptografi di dalam perangkat lunak. 3.4 Analisa Permasalahan Masalah yang terjadi pada analisa sistem berjalan yang berkaitan dengan pengamanan data 32adalah sebagai berikut : 1. Berdasarkan hasil wawancara dengan pihak terkait diketahui bahwa masih minimnya pengetahuan tentang pentingnya menjaga keamanan sebuah data yang bersifat penting atau rahasia didalam sebuah komputer, seperti data keuangan dan data penting lainnya. 2. Sebuah data yang bersifat rahasia bisa saja dicuri, dimanipulasi dan dikorupsi apabila tidak dijaga keamanannya. Dan apabila terjadi sebuah komputer terjankit sebuah virus dan virus tersebut mengenkrip data penting tersebut maka resikonya ialah kehilangan data tersebut, olehkarena itu disamping dari pentingnya menjaga keamanan data, proses back-up data juga penting dilakukan. 3.5 Analisa Sistem Yang Sedang Berjalan Pada Fakultas Informatika Univeristas Bhayangkara pada sistem sebelumnya tidak ditemukannya pengamanan data file penting atau bersifat rahasia di divisi administrasi Fakultas Informatika Universitas Bhayangkara Jakarta Raya. Penyimpanan data file hanya bersifat standard pada suatu

device, belum di enkripsi atau diamankan untuk menjaga kerahasiaan data tersebut. Berikut penulis jabarkan sistem berjalan yang ada pada divisi administasi Fakultas Bhayangkara Jakarta Raya untuk penyimpanan data file pada tiga diagram berikut yaitu Use Case, Activity Diagram dan Sequence Diagram. Gambar 3.5.1 Use Case Sistem Berjalan Gambar 3.5.2 Activity Diagram Sistem Berjalan Gambar 3.5.3 Sequence Diagram Sistem Berjalan 3.6 Analisa Kebutuhan Sistem Sebelum melakukan perancangan sebuah aplikasi, penulis melakukan analisa terhadap kebutuhan sistem yang bertujuan untuk menyesuaikan kebutuhan user dengan aplikasi yang dirancang oleh penulis, sehingga aplikasi yang di rancang oleh penulis dapat memenuhi tujuan penelitian ini. Analisa kebutuhan sistem juga dilakukan dengan cara melakukan observasi atau pengamatan langsung kepada pihak terkait. Dalam perancangan aplikasi ini ada beberapa hal yang diperlukan dalam proses perancangannya adapun hal tersebut ialah : 1. Algoritma Advanced Encryption Standard (AES-256) dengan mode CBC Salah satu mode operasi AES yaitu CBC atau yang disebut Chiper Block Chaining. Pada algoritma blok chipper seperti AES ini, plaintext atau pesan mentah yang masuk untuk diproses dengan panjang yang tetap yaitu n, akan tetapi jika ukuran datanya terlalu panjang maka dilakukan pemecahan data data tersebut menjadi blok blok dengan ukuran yang lebih kecil. Pada CBC, rangkaian bit-bit pada plaintext dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (initialization vector) untuk menggabungkan dengan plaintext pertama dan chipertext block sebelumnya menjadi IV di block selanjutnya. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini. 2. Algoritma Secure Hash Algorithm (SHA-256) Fungsi hash adalah adalah fungsi yang menerima masukan string yang panjangnya sembarang dan dikonversikan menjadi string dengan keluaran yang panjangnya tetap. Fungsi hash yang berbeda akan menghasilkan output-output yang berbeda juga, tetapi kemungkinan ukuran output dari masing-masing algoritma hashing selalu konstan. algoritma SHA-256 hanya akan menghasilkan message digest 256 bit. Penggunaan algoritma SHA digunakan untuk proses pembuatan kunci cipher dengan cara menghitung nilai message digest. Algoritma SHA-256 tergolong dari algoritma fungsi hash

sebelumnya yaitu SHA-2. Ada beberapa series algoritma SHA yaitu SHA-0, SHA-1, SHA-2 dan terbarukan yaitu SHA-3. 3. Data Objek Data objek yang digunakan sebagai semple data untuk di enkripsi dan dekripsi ialah contoh data yang diambil dari fakultas ilmu komputer Universitas Bhayangkara Jakarta Raya. Date tersebut berjenis file dokumen, file gambar, file suara dan file video untuk pemrosesan enkripsi dan dekripsi pada aplikasi Secret Fichier. Setelah ditentukan algoritma yang digunakan dalam penelitian ini, yaitu algoritma Advanced Encryption Standard (AES) dengan panjang kunci 256 bit untuk pemrosesan enkripsi dan dekripsi file serta Secure Hash Algortihm (SHA-256) sebagai algoritma fungsi hashnya. Maka dari itu penelitian ini akan membuat sebuah aplikasi pengamanan data berbasis desktop yang dapat melakukan hal hal berikut ini : 1. Aplikasi enkripsi dan dekripsi Secret Fichier dapat berjalan pada sistem operasi windows. 2. Aplikasi Secret Fichier dapat mengenkripsi jenis file tunggal bukan folder yaitu dokumen (berformat docx, xlxs), gambar (berformat jpg, png) suara dan video (berformat mp4). 3. Aplikasi Secret Fichier dapat mengdekripsi file yang sudah di enkripsi sebelumnya pada aplikasi tersebut dengan kunci atau password yang sama dan mengembalikan file tersebut seperti semula. 4. Aplikasi Secret Fichier dapat mengamankan data bersifat penting dan mengubahnya menjadi file yang yang tidak terbaca (chipper text) dengan ekstensi file .sf. 3.6.1 Flowchart Aplikasi Secret Ficher Gambar 3.6.1 Flowchart Enkripsi dan Dekripsi File Berikut gambar diatas ialah flowchart dari aplikasi Secret Fichier, yaitu aplikasi enkripsi dan dekripsi file yang menerapkan algoritma AES-256 dan SHA-256. Seperti gambar flowchart diatas, tahapan pertama pada aplikasi enkripsi dan dekripsi file Secret Fichier, yaitu user memilih atau menginput sebuah file (bukan folder) yang ingin di proses, apabila file tersebut berformat .sf maka file tersebut akan didekripsi. setelah user memilih file dan memilih lokasi file setelah di proses, selanjutnya user harus menginputkan kembali password yang user gunakan pada proses enkripsi, password itu akan di autentifikasi apabila password yang diinputkan salah maka proses dekripsi tidak terjadi atau gagal dan jika password yang diinputkan benar, maka file akan kembali seperti semula. apabila file tersebut berjenis file dokumen, suara, gambar dan video maka file tersebut dapat di proses untuk dienkripsi.

Setelah user memilih sebuah file yang akan dienkripsi, lalu user diharuskan memilih destinasi lokasi file yang ingin ditaruh setelah proses enkripsi selesai, setelah itu user menginputkan password dan proses enkripsi terjadi. Apabila user 3.7 Alat Penelitian Demi keberlangsungnya perancangan aplikasi enkripsi dan dekripsi Secret Fichier maka dari itu dibutuhkannya perangkat yang mendukung dan memumpuni untuk kemudahan perancangan baik dari segi perangkat keras (hardware) maupun perangkat lunak (software). Berikut spesifikasi perangkat keras dan perangkat lunak yang digunakan : 1. Perangkat Keras : Processor : Intel Core i7-10510U Memory RAM : 8 GB Solid State Drive : 1000 GB VGA : Nvidia MX250 2GB 2. Perangkat Lunak : Windows 10 Home 64 bit Visual Studo .NET CORE 3.0 3.8 Jadwal Penelitian Jadwal kegiatan penulisan tugas akhir skripsi ini dirincikan sebagai berikut: Kegiatan September (Minggu) Oktober (Minggu) November (Minggu) Desember (Minggu) Januari (Minggu) Februari (Minggu) 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 Identifikasi Masalah Studi Literatur Penulisan Proposal Analisa dan Perancangan Implementasi Hasil BAB IV HASIL DAN PEMBAHASAN 4.1 Penerapan Secure Hash Algorithm (SHA-256) Pada penelitian pengamanan data file ini, penulis menerapkan algoritma SHA. Algoritma SHA yang digunakan ialah SHA-256, yang hanya menghasilkan message digest sebesar 256 bit. SHA-256 mengubah pesan masukan ke dalam message digest 256 bit, berdasarkan Secure Hash Signature Standard. Lebih mudahnya plaintext diproses dengan fungsi hash lalu keluaran tersebut menjadi hash text yang tidak dapat terbaca. Berikut ini implementasi proses SHA-256 dari sebuah pesan asli yang berisi “xyz” dengan penyelesaian tahap demi tahap proses dari SHA-256, sebagai berikut : Sebuah pesan asli disimpan kedalam variable M, maka M = “xyz”. Variable M dikonversikan kedalam bentuk binary, sehingga menjadi M = 0111 1000 0111 1001 0111 1010. Lalu dilakukan penambahan padding yaitu bit “1” pada variable M sehingga menjadi M = 0111 1000 0111 1001 0111 1010 1. Walaupum SHA-256 menghasilkan 265 bit akan tetapi Plaintext yang panjangnya kurang dari 264 bit harus dioperasikan kedalam blocksize 512 bits sehingga menjadi sebuah message digest sebesar 256 bits.. Agar

blocksize mencapai 512 bit, maka selanjutnya tambahkan panjang pesan asli ke dalam bentuk biner. karena ukuran frase “xyz” dalam hal ini terdapat delapan bit per hurufnya dan menjadi 24 bit secara keseluruhan lalu dikonversikan ke binary ialah 24 = 0001 1000 dan menjadi binary dengan ukuran 8 bit. Agar penambahan pesan asli menjadi 512 bit, maka akan ditambahkan bit ”0” sebanyak 479 buah (512-25-8 = 479) dan ditambahkan lagi hasil konversi tadi yaitu 00011000. Gambarannya seperti dibawah ini :

0111100001111001011110101000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000011000 Dari 512 bits diatas diubah menjadi variable yang memiliki 32 bit, sehingga 512/32 = 16 variable. Kemudian dilakukan parsing menjadi 16 buah variable dengan 32 bit untuk masing masing blok pesan M. M0 = 01111000 01111001 01111010 10000000 M1 = 00000000 00000000 00000000 00000000 M2 = 00000000 00000000 00000000 00000000 M3 = 00000000 00000000 00000000 00000000 M4 = 00000000 00000000 00000000 00000000 M5 = 00000000 00000000 00000000 00000000 A6 = 00000000 00000000 00000000 00000000 M7 = 00000000 00000000 00000000 00000000 M8 = 00000000 00000000 00000000 00000000 M9 = 00000000 00000000 00000000 00000000 M10 = 00000000 00000000 00000000 00000000 M11 = 00000000 00000000 00000000 00000000 M12 = 00000000 00000000 00000000 00000000 M13 = 00000000 00000000 00000000 00000000 M14 = 00000000 00000000 00000000 00000000 M15 = 00000000 00000000 00000000 00011000 Tahapan selanjutnya ialah inisialisasi nilai hash awal dalam notasi heksadesimal, inisialisasi nilai hash awal pada SHA-256 bertujuan untuk menyimpan nilai hash awal dan nilai keluarannya, oleh karena itu digunakan buffer H0, H1, H2, H3, H4, H5, H6, H7. Berikut

inisialisasi nilai hash awal dalam notasi heksadesimal ; H0 = 6a09e667 H1 = bb67ae85 H2 = 3c6ef372 H3 = a54ff53a H4 = 510e527f H5 = 9b05688c H6 = 1f83d9ab H7 = 5be0cd19 Selanjutnya ialah tahap pemrosesan dari SHA-256, pemrosesan SHA-256 sendiri terdiri atas 1 round yang mempunyai 64 operasi. Untuk memproses setiap satu blok pesan yang terdiri dari 512 bit, di perlukan sebanyak 64 operasi. Setiap blok M1, M2, …., M(n) dengan "n" adalah jumlah blok pesan. Lalu, siapkan penjadwalan pesan dari 16 buah blok hasil parsing tadi menggunakan rumus berikut ; Setelah itu, tahapan selanjutnya ialah inisialisasi working variable a, b, c, d, e, f, g dan h untuk blok pesan M tadi dari nilai hash awal. a = H0 b = H1 c = H2 d = H3 e = H4 f = H5 g = H6 h = H7 dan tahapan selanjutnya menyiapkan 64 koefisian inisialisasi array yang sudah ditetapkan pada standar SHA-2. Algoritma SHA ini menghitung 0…63 = 64 kali putaran untuk setiap perhitungan blocknya. Delapan variabel yang diberi label tadi yaitu a, b, c, d, e, f, g, dan h nilainya terus berganti selama perputaran sebanyak 64 kali putaran dan hasil dari perhitungan iterasi sebagai berikut : I = 63 cbfed63a 28e6f83f 95f9f7fb 0ad26aec 47846e35 90815365 c7f695f6 105bc569 lalu hasil dari iterasi terakhir yang tadi kemudian dijumlahkan dengan nilai hash awal, maka akan menghasilkan sebagai berikut : H0 = cbfed63a + 6a09e667 = 3608bca1 H1 = 28e6f83f + bb67ae85 = e44ea6c4 H2 = 95f9f7fb + 3c6ef372 = d268eb6d H3 = 0ad26aec + a54ff53a = b0226026 H4 = 47846e35 + 510e527f = 9892c0b4 H5 = 90815365 + 9b05688c = 2b86bbf1 H6 = c7f695f6 + 1f83d9ab = e77a6fa1 H7 = 105bc569 + 5be0cd19 = 6c3c9282 Dan selanjutnya penggabungan hasil akhir dari penjumlahan tadi merupakan perhitungan dari SHA-256, dan terbentuklah message digest variable M, dengan isi pesan asli yaitu "xyz". Maka nilah hash dari pesan M yaitu : 3608bca1e44ea6c4d268eb6db02260269892c0b42b86bbf1e77a6fa16c3c9282 4.2 Penerapan Advanced Encryption Standard (AES-256) Pada penjelasan sebelumnya disebutkan bahwa AES hanya mempunyai panjang block 128 bit dan mempunyai tiga panjang kunci yang diperbolehkan, yaitu 128 bit, 192 bit dan 256 bit. Panjang kunci tersebutlah yang akan menentukan banyaknya perputaran pada "layer" AES. Pada penelitian ini, penulis menggunakan panjang kunci 256 bit yang berarti jumlah perputaran

pada setiap layernya ialah 14 round. Untuk diketahui sebelumnya bahwa setiap teks atau file ialah sebuah aliran dari beberapa bytes. Dan setiap byte adalah delapan bits. Oleh karena itu pada penjelasan tahapan enkripsi file dengan AES-256 dan SHA-256 ini file tersebut akan di pecah menjadi kumpulan bytes. Berikut penjelasan tahapan enkripsi dan dekripsi file menggunakan metode AES-256 dan SHA-256 : 4.2.1 Enkripsi File Pada tahapan penjelasan enkripsi file dengan aes, penulis akan membagi tiap tiap tahapannya kedalam sub bab - sub bab. Ada empat tahapan yang akan dijelaskan oleh penulis, dari tahapan penurunan kunci (key derivation), padding, enkripsi blok pertama dan enkripsi blok kedua. Didalam pengenkripsian blok tersebut, ada beberapa tahapan lagi untuk menghasilkan sebuah chipper text, yaitu penambahan IV, mencari ronde kunci 2 hingga 14, pengenkripsian ronde pertama dengan tahapan SubBytes, ShiftRows, MixColumn dan AddRoundKey. Semua tahapan tersebut akan dijelaskan sebagai berikut. Siapkan file dan kunci yang akan dienkripsi pada contoh kali ini file yang akan dienkripsi ialah file text berekstensi .txt dengan size file 29 bytes, konversikan file tersebut kedalam code hexadecimal agar dapat melakukan perhitungan tahapan aes. Contoh text 1dan kunci yang akan digunakan ialah: Teks asli : “Hello this is Secret Fichier!” Kunci : “xyz” Tahapan selanjutnya ialah kunci yang kita gunakan disini harus telah di prosess terlebih dahulu dengan fungsi hash atau yang dimaksud dengan istilah Key Derivation (penurunan kunci) dengan SHA256. Pada proses sebelumnya, kunci “xyz” telah di proses dengan SHA-256 dan menghasilkan 64 hex digits atau setara dengan 32 bytes (256 bits) dengan kententuan SHA-2. Berikut hasil dari konversi teks asli kedalam heksadesimal dan kunci setelah di proses dengan SHA-256. Teks asli : 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 63 72 65 74 20 46 69 63 68 69 65 72 21 Kunci : 36 08 bc a1 e4 4e a6 c4 d2 68 eb 6d b0 22 60 26 98 92 c0 b4 2b 86 bb f1 e7 7a 6f a1 6c 3c 92 82 Dikarenakan AES hanya memiliki panjang block dengan ukuran 128 bit sedangkan file teks yang penulis gunakan disini ialah file dengan ukuran 29 bytes (29 x 8 = 232 bit), maka dari itu teks asli akan di bagi menjadi 2 block yang masing masing block tersebut mempiliki panjang 128 bit. Berikut gambaran pembagian block : 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 |Hello this is Se| 63 72

65 74 20 46 69 63 68 69 65 72 21 |cret Fichier!| Dikarenakan pada salah satu panjang block belum mencapai ukuran yang ditetapkan oleh AES, yaitu 128 bit maka di butuhkan penambahan padding atau penambahan bit dibelakangnya sebanyak 3 buah (agar menjadi block dengan ukuran 128 bit), yang ditulis dengan bit “03”. Ditulis “03” dikarenakan agar aes mengerti bahwa 2 bit yang dibelakang ialah padding, dan pada saat proses pendekripsian padding tersebut akan di hapus. Berikut gambaran penambahan padding pada block : 48 65 6c 6c 6f 20 74 68 69 73 20 69 73 20 53 65 |Hello this is Se| 63 72 65 74 20 46 69 63 68 69 65 72 21 03 03 03 |cret Fichier!| Plaintext Block 2 63 20 68 21 72 46 69 03 65 69 65 03 74 63 72 03 Agar lebih mudah penggambaran pada masing masing block, maka penulis akan membuat matriks dengan ukuran 4 x 4 untuk diisi dengan masing masing nilai heksadesimal kedalam block block yang tadi di gambarkan. Plaintext Block 1 48 6f 69 73 65 20 73 20 6c 74 20 53 48 6f 69 73 Dalam pemrosesan AES mode CBC, rangkaian bit-bit pada plain text dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (initialization vector) untuk menggabungkan dengan plain text pertama. Tahapan proses mode CBC akan ditunjukan pada gambar dibawah ini. Pada pemrosesan CBC, diperlukannya penambahan IV atau yang disebut sebagai IV (initialization vector) . Dalam pemrosesan dengan algoritma aes sudah disebutkan sebelumnya bahwa ada dua input yang dibutuhkan, yaitu input yang pertama ialah teks asli yang sudah di ubah kedalam kode heksadecimal yang akan di proses pengenkripsiannya, yang kedua ialah kunci yang telah di proses dengan fungsi hash sebagai aeskey pada aes itu sendiri. Pada penjelasan selanjutnya, penulis akan menjelaskan tahapan pengenkripsian block pertama dengan AES mode CBC. 4.2.2 Enkripsi Blok Pertama Pada pemrosesan plain text block pertama, diperlukan penambahan IV dan aeskey pada rumus CBC block pertama, yang di gambarkan sebagai mana berikut ini : IV : 0000000000000000 0000000000000000 (16 bytes = 128 bits) aeskey : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 (32 bytes = 256 bits) block1 : 48656c48 6f20746f 69732069 73205373 Berikut rumus CBC block pertama : Selanjutnya, apabila block pertama sudah di proses, maka hasil dari chiper text pertama akan menjadi IV selanjutnya

pada block kedua dan hasil tersebut juga sekaligus menjadi chiper text block pertama, seperti yang digambarkan pada gambar CBC mode diatas. Setelah penambahan IV dan aeskey dilakukan, tahapan selanjutnya ialah pengenkripsian di dalam aes. Berikut gambaran proses aes : Walaupun AES-256 mempunyai panjang kunci 256 bit, tetapi AES hanya memiliki panjang block 128 bit dalam matrik 4x4. Maka block matrik kunci dibagi menjadi 2 bagian. Block bagian pertama dijadikan kunci ronde awal atau RoundKeys 0, dan di bagian kedua dijadikan RoundKeys 1. RoundKeys 0 36 e4 d2 b0 08 4e 68 22 bc a6 eb 60 a1 c4 6d 26 RoundKeys 1 98 2b e7 6c 92 86 7a 3c c0 bb 6f 92 b4 f1 a1 82 Seperti gambar tahapan aes diatas, tahapan pertama yang akan dilakukan ialah proses initialization key atau pre round, tahapan tersebut dilakukan proses AddRoundKey kunci ronde 0. Pada tahap ini terjadi perhitungan XOR antara teks asli dengan kunci. 2XOR merupakan kepanjangan dari Exclusive OR yang mana keluarannya akan berlogika 1 apabila inputannya berbeda, namun apabila semua inputanya sama maka akan memberikan keluarannya 0. Proses XOR dilakukan pada masing masing blok pada matriks, seperti matriks plain text blok kolom [1,1] XOR dengan matriks RoundKeys 0 kolom [1,1] dan seterusnya. Berikut akan disimulasikan XOR pada matriks plain text blok 1 dengan matriks RoundKeys 0. Untuk dapat melakukan operasi XOR, nilai heksadecimal tadi harus diubah menjadi nilai biner berukuran 8 bit terlebih dahulu. Seperti contoh berikut : 48 = 0100 1000 36 = 0011 0110 Proses XOR disimulasikan seperti dibawah ini : 20 1 0 0 1 0 0 0 ⊕ 0 0 1 1 0 1 1 0 0 1 1 1 1 1 0 Nilai heksadesimal dari 0111 1110 ialah 7E, jadi 48 ⊕ 36 = 7E Hitung keseluruhan nilai XOR di kolom kolom matriks plain text blok 1 dengan RoundKeys 0 hingga mengahasilkan blok matriks berukuran 4 x 4. 65 ⊕ 08 = 6d 6c ⊕ bc = d0 6c ⊕ a1 = cd 6f ⊕ e4 = 8b 20 ⊕ 4e = 6e 74 ⊕ bc = d2 68 ⊕ c4 = ac 69 ⊕ d2 = bb 73 ⊕ 68 =1b 20 ⊕ eb = cb 69 ⊕ 6d = 04 73 ⊕ b0 = c3 20 ⊕ 22 = 02 53 ⊕ 60 = 33 65 ⊕ 26 = 43 Berikut matriks dari plaintext block 1 XOR RoundKeys 0 Plaintext ⊕ RoundKeys 0 = Hasil 48 6f 69 73 36 e4 d2 b0 7e 8b 8b c3 65 20 73 20 08 4e 68 22 6d 6e 1b 02 6c 74 20 53 bc a6 eb 60 d0 d2 cb 33 6c 68 69 65 a1 c4 6d 26 cd ac 04 43 Pada tahapan sebelumnya penambahan IV dengan operasi XOR dilakukan juga, akan tetapi karena IV awal bernilai 16 hex digit nilai “00” maka hasilnyaa

akan sama seperti aslinya. Selanjutnya pada tahap ini, terdapat perulangan sebanyak 13 kali. Setiap prosess AddRoundKey dilakukan operasi XOR terhadap RoundKeys 1 sampai dengan RoundKeys 13. Berikut merupakan tahapan dari 13 kali pengulangan : a. Proses SubBytes Pada proses SubBytes ini, hasil dari operasi XOR di pre-round akan di di subtitusikan dengan nilai yang ada di table S-Box (tabel 2.8.1 hlm.16). proses dari SubBytes ialah sebagai berikut : 7e 8b 8b c3 6d 6e 1b 02 d0 d2 cb 33 cd ac 04 43 Lakukan proses subtitusi S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (7) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (e) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil SubByte. Seperti contoh gambar berikut : Setelah dilakukan subtitusi S-box pada proses SubBytes pada nilai 7e, maka didapatkan hasil yaitu f3. Lalu lanjutkan subtitusi S-box pada kolom [2.1] matriks diatas yaitu nilai 6d. Pada proses Subtitusi S-box pada nilai 6d, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes 6d mendapatkan hasil nilai 3c. Lalu lanjutkan subtitusi S-box pada kolom [3.1] matriks diatas yaitu nilai d0. Pada proses Subtitusi S-box pada nilai d0, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes d0 mendapatkan hasil nilai 70. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai cd. Pada proses Subtitusi S-box pada nilai cd, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes d0 mendapatkan hasil nilai bd. Lalu lanjutkan subtitusi S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table S-Box proses SubBytes. Setelah dilakukan proses SubBytes pada keseluruhan nilai, maka didapatkan hasil dari proses SubBytes ialah sebagai berikut: Hasil SubBytes f3 3d ea 2e 3c 9f af 77 70 b5 1f c3 bd 91 f2 1a b. Proses ShiftRows Lalu tahapan selanjutnya ialah proses ShiftRows. Hasil dari proses SubBytes tadi akan di geser beberapa bit di dalam proses ini, berikut gambaran yang terjadi pada proses ShiftRows ini : Pada nilai yang ditandai akan di putar lebih satu byte, seperti berikut ini f3 3d ea 2e f3 3d ea 2e 3c 9f af

77 9f af 77 3c 70 b5 1f c3 70 b5 1f c3 bd 91 f2 1a bd 91 f2 1a Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut: f3 3d ea 2e f3 3d ea 2e 3c 9f af 77 9f af 77 3c 70 b5 1f c3 1f c3 70 b5 bd 91 f2 1a bd 91 f2 1a Pada nilai yang ditandai akan diputar lebih dari tiga byte, seperti berikut: f3 3d ea 2e f3 3d ea 2e 3c 9f af 77 9f af 77 3c 70 b5 1f c3 1f c3 70 b5 bd 91 f2 1a 1a bd 91 f2 Hasil dari proses ShiftRows ialah sebagai barikut : Hasil ShiftRows f3 3d ea 2e 9f af 77 3c 1f c3 70 b5 1a bd 91 f2 c. Proses MixColumns Proses selanjutnya adalah proses MixColumns, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada proses ShiftRows tadi dengan matriks yang telah di tetapkan oleh Rijndael. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial dan pergeseran beberapa bit. Berikut merupakan proses dari MixColumns : 02 03 01 01 X f3 3d ea 2e C11 C12 C13 C14 01 02 03 01 9f af 77 3c C21 C22 C23 C24 01 01 02 03 1f c3 70 b5 C31 C32 C33 C34 03 01 01 02 1a bd 91 f2 C41 C42 C43 C44 Operasi perkalian disini maksudnya adalah: 1. Perkalian dengan 01 artinya tidak berubah 2. Perkalian dengan 02 artinya mengalikan dengan perkalian polynomial dan mengubah nilai yang di kali kedalam nilai biner dan di lakukan perkalian polynomial serta modulo pada biner. 3. Perkalian dengan 03 mengalikan dengan perkalian polynomial dan mengubah nilai yang di kali kedalam nilai biner dan di lakukan perkalian polynomial serta modulo pada biner. Berikut ini merupakan perhitungan pencarian nilai untuk C11 : 02 03 01 01 X f3 01 02 03 01 9f 01 01 02 03 1f 03 01 01 02 1a C11 = {02.f3} ⊕ {03.9f} ⊕ {01.1f} ⊕ {01.1a} · {02.f3} 02 = 0000 0010 = x f3 = 1111 0011 1= x7 + x6 + x5 + x4 + x + 1 = (x)(x7 + x6 + x5 + x4 + x + 1) = x8 + x7 + x6 + x5 + x2 + x = x8 + x7 + x6 + x5 + x2 + x modulo x8 + x4 + x3 + x1 + 1 = 1 1110 0110 1 0001 1011 (XOR) 1111 1101 (fd) · {03.9f} 03 = 0000 0011 = x + 1 9f = 1001 1111 = x7 + x4 + x3 + x2 + x + 1 = (x + 1)(x7 + x4 + x3 + x2 + x + 1) = (x8 + x5 + x4 + x3 + x2 + x) + (x7 + x4 + x3 + x2 + x + 1) = x8 + x7 + x5 + 1 = x8 + x7 + x5 + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 1010 0001 1 0001 1011 (XOR) 0001 1010 (ba) · {01.1f} = 1f · {01.1a} = 1a Perhitungan : C11 = {02.f3} ⊕ {03.9f} ⊕ {01.1f} ⊕ {01.1a} C11 = {fd} ⊕ {ba} ⊕ {1f} ⊕ {1a} = 42 Berikut ini merupakan perhitungan pencarian nilai untuk C21 : 02

03 01 01 X f3 01 02 03 01 9f 01 01 02 03 1f 03 01 01 02 1a C21 = {01.f3} ⊕ {02.9f} ⊕ {03.1f} ⊕ {01.1a} · {01.f3} = f3 · {02.9f} 02 = 0000 0010 = x 9f = 1001 1111 = x7 + x4 + x3 + x2 + x + 1 = (x)(x7 + x4 + x3 + x2 + x + 1) = x8 + x5 + x4 + x3 + x2 + x = x8 + x5 + x4 + x3 + x2 + x modulo x8 + x4 + x3 + x1 + 1 = 1 0011 1110 1 0001 1011 (XOR) 0010 0101 (25) · {03.1f} 03 = 0000 0011 = x + 1 1f = 0001 1111 = x4 + x3 + x2 + x + 1 = (x + 1)(x4 + x3 + x2 + x + 1) = (x5 + x4 + x3 + x2 + x) + (x4 + x3 + x2 + x + 1) = x5 + 1 = 0010 0001 (21) · {01.1a} = 1a Perhitungan : C21 = {01.f3} ⊕ {02.9f} ⊕ {03.1f} ⊕ {01.1a} C21 = {f3} ⊕ {25} ⊕ {21} ⊕ {1a} = ed Berikut ini merupakan perhitungan pencarian nilai untuk C31 : 02 03 01 01 X f3 01 02 03 01 9f 01 01 02 03 1f 03 01 01 02 1a C31 = {01.f3} ⊕ {01.9f} ⊕ {02.1f} ⊕ {03.1a} · {01.f3} = f3 · {01.9f} = 9f · {02.1f} 02 = 0000 0010 = x 1f = 0001 1111 = x4 + x3 + x2 + x + 1 = (x)(x4 + x3 + x2 + x + 1) = x5 + x4 + x3 + x2 + x = 0011 1110 (3e) · {03.1a} 03 = 0000 0011 = x + 1 1a = 0001 1010 = x4 + x3 + x = (x + 1)(x4 + x3 + x) = (x5 + x4 + x2) + (x4 + x3 + x) = x5 + x3 + x2 + x = 0010 1110 (2e) Perhitungan : C31 = {01.f3} ⊕ {01.9f} ⊕ {02.1f} ⊕ {03.1a} = {f3} ⊕ {9f} ⊕ {3e} ⊕ {2e} = 7c Berikut ini merupakan perhitungan pencarian nilai untuk C41: 02 03 01 01 X f3 01 02 03 01 9f 01 01 02 03 1f 03 01 01 02 1a C41 = {03.f3} ⊕ {01.9f} ⊕ {01.1f} ⊕ {02.1a} · {03.f3} 03 = 0000 0011 = x + 1 f3 = 1111 0011 = x7 + x6 + x5 + x4 + x + 1 = (x + 1)(x7 + x6 + x5 + x4 + x + 1) = (x8 + x7 + x6 + x5 + x2 + x) + (x7 + x6 + x5 + x4 + x + 1) = x8 + x4 + x2 + 1 = 0000 1110 (0e) · {01.9f} = 9f · {01.1f} = 1f · {02.1a} 02 = 0000 0010 = x 1a = 0001 1010 = x4 + x3 + x = (x)(x4 + x3 + x) = x5 + x4 + x2 = 0011 0100 (34) Perhitungan : C41 = {03.f3} ⊕ {01.9f} ⊕ {01.1f} ⊕ {02.1a} = {0e} ⊕ {9f} ⊕ {1f} ⊕ {34} = ba Untuk mendapat hasil yang lainnya C12, C22, C32, C42, C13, C23, C33...C44 lakukan perhitungan polynomial dengan cara seperti diatas. Berikut hasilnya : C12 = {02.3d} ⊕ {03.af} ⊕ {01.c3} ⊕ {01.bd} = {7a} ⊕ {ea} ⊕ {c3} ⊕ {bd} = ee C22 = {01.3d} ⊕ {02.af} ⊕ {03.c3} ⊕ {01.bd} = {3d} ⊕ {45} ⊕ {5e} ⊕ {bd} = 9b C32 = {01.3d} ⊕ {01.af} ⊕ {02.c3} ⊕ {03.bd} = {3d} ⊕ {af} ⊕ {9d} ⊕ {dc} = d3 C42 = {03.3d} ⊕ {01.af} ⊕ {01.c3} ⊕ {02.bd} = {47} ⊕ {af} ⊕ {c3} ⊕ {61} = 4a C13 = {02.ea} ⊕ {03.77} ⊕ {01.70} ⊕ {01.91} = {cf} ⊕ {99} ⊕ {70} ⊕ {91} = b7 C23 = {01.ea} ⊕ {02.77} ⊕ {03.70} ⊕ {01.91} = {ea} ⊕ {ee} ⊕ {90} ⊕ {91} = 05 C33 = {01.ea} ⊕ {01.77} ⊕ {02.70} ⊕

{03.91} = {ea} ⊕ {77} ⊕ {e0} ⊕ {a8} = d5 C43 = {03.ea} ⊕ {01.77} ⊕ {01.70} ⊕ {02.91} = {25} ⊕ {77} ⊕ {70} ⊕ {39} = 1b C14 = {02.2e} ⊕ {03.3c} ⊕ {01.b5} ⊕ {01.f2} = {5c} ⊕ {44} ⊕ {b5} ⊕ {f2} = 5f C24 = {01.2e} ⊕ {02.3c} ⊕ {03.b5} ⊕ {01.f2} = {2e} ⊕ {78} ⊕ {c4} ⊕ {f2} = 60 C34 = {01.2e} ⊕ {01.3c} ⊕ {02.b5} ⊕ {03.f2} = {2e} ⊕ {3c} ⊕ {71} ⊕ {0d} = 6e C44 = {03.2e} ⊕ {01.3c} ⊕ {01.b5} ⊕ {02.f2} = {72} ⊕ {3c} ⊕ {b5} ⊕ {ff} = 04 Hasil dari proses MixColumns adalah sebagai berikut : Hasil MixColumns 42 ee b7 5f ed 9b 05 60 7c d3 d5 6e ba 4a 1b 04 d. Proses AddRoundKeys Proses selanjutnya ialah AddRoundKey, setelah hasil yang didapat dalam proses MixColumns selanjutnya akan melakukan operasi XOR dengan RoundKeys 1 terhadap hasil dari MixColumns. Seperti gambar berikut ini : Hasil MixColumn XOR KeyRound1 = Hasil Ronde 1 42 ee b7 5f 98 2b e7 6c da c5 50 33 ed 9b 05 60 92 86 7a 3c 7f 1d 7f 5c 7c d3 d5 6e c0 bb 6f 92 bc 68 ba fc ba 4a 1b 04 b4 f1 a1 82 0e bb ba 86 Setelah mendapat hasi dari AddRoundKey ronde pertama, selanjutnya pencarian nilai ronde kedua hingga ronde ketiga belas dan melalui proses proses sebelumnya, yaitu proses SubBytes, proses ShiftRows dan proses MixColumns dan pencarian kunci di tiap rondenya untuk pemrosesan XOR dengan MixColumns (proses AddRoundKey) nanti. Yang kita ketahui kunci yang kita gunakan disini ialah “xyz” yang telah di proses dengan fungsi hash SHA-256 dan menghasilkan 32 bytes atau setara dengan 256 bits, yaitu sebagai berikut : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 Karena tiap ronde kunci aes hanya berisikan 16 bytes tiap rondenya, maka penulis akan membagi dua matriks kunci tersebut dan menaruhnya kedalam matriks 4x4 atau 16 bytes, seperti sebagai berikut ; RoundKeys 0 36 e4 d2 b0 08 4e 68 22 bc a6 eb 60 a1 c4 6d 26 RoundKeys 1 98 2b e7 6c 92 86 7a 3c c0 bb 6f 92 b4 f1 a1 82 Penulis sudah mengetahui ronde kunci 0 dan 1, oleh karena itu langkah selanjutnya ialah mencari ronde kunci 2 hingga 14. Berikut tahapan pencarian ronde ronde kunci yaitu : a. RoundKeys 2 Pencarian kunci ronde 2 dengan menggunakan transformasi RotWord. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci RotWord RoundKeys 2 98 2b e7 c3 92 86 7a 92 c0 bb 6f 82 b4 f1 a1 6c ronde 1. Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys

2 98 2b e7 eb 92 86 7a 4f c0 bb 6f 13 b4 f1 a1 50 Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 0 dan kolom 1 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 2. eb XOR 36 XOR 01 = dc 4f 08 00 47 13 bc 00 af 50 a1 00 f1 RoundKeys 2 dc 38 ea 5a 47 09 61 43 af 09 e2 82 f1 35 58 7e Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 2 yang di perlu dilakukan ialah lakukan hasil RoundKeys 2 kolom 1 dengan operasi XOR Roundkeys 0 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 2 : b. RoundKeys 3 Pencarian kunci ronde 3 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini : Transformation S-box RoundKeys 3 dc 38 ea be 47 09 61 1a af 09 e2 13 f1 35 58 f3 Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 1. Hasilnya adalah kolom 1 kunci ronde 3. be XOR 98 = 26 1a 92 88 13 c0 d3 f3 b4 47 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 3 yang di perlu dilakukan ialah lakukan hasil RoundKeys 3 kolom 1 dengan operasi XOR Roundkeys 1 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk Roundkeys 3. Berikut hasil dari RoundKeys 3 : RoundKeys 3 26 0d ea 86 88 0e 74 48 d3 68 07 95 47 b6 17 95 c. RoundKeys 4 Pencarian kunci ronde 4 dengan menggunakan transformasi RotWord. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci RotWord RoundKeys 4 26 0d ea 48 88 0e 74 95 d3 68 07 95 47 b6 17 86 ronde 3. Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys 4 26 0d ea 52 88 0e 74 2a d3 68 07 2a 47 b6 17 44 52 XOR dc XOR 02 = 8c 2a 47 00 6d 2a af 00 85 44 f1 00 B5 Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 2 dan kolom 2 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 4. Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 4 yang di perlu dilakukan ialah lakukan hasil RoundKeys 4 kolom 1 dengan operasi XOR RoundKeys 2 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk

RoundKeys 4 : RoundKeys 4 8c b4 5e 04 6d 64 05 46 85 8c 6e ec b5 80 d8 a6 d. RoundKeys 5 Pencarian kunci ronde 5 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini : Transformation S-box RoundKeys 5 8c b4 5e f2 6d 64 05 5a 85 8c 6e ce b5 80 d8 24 f2 XOR 26 = d4 5a 88 d2 ce d3 1d 24 47 63 Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 3. Hasilnya adalah kolom 1 kunci ronde 5. RoundKeys 5 d4 d9 33 b5 d2 dc a8 e0 1d 75 72 e7 63 d5 c2 57 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 5 yang di perlu dilakukan ialah lakukan hasil RoundKeys 5 kolom 1 dengan operasi XOR RoundKeys 3 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 5. e. RoundKeys 6 RotWord RoundKeys 6 d4 d9 33 e0 d2 dc a8 e7 1d 75 72 57 63 d5 c2 b5 Pencarian kunci ronde 6 dengan menggunakan transformasi RotWord. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 5. Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys 6 d4 d9 33 e1 d2 dc a8 94 1d 75 72 5b 63 d5 c2 d5 e1 XOR 8c XOR 04 = 69 94 6d 00 f9 5b 85 00 de d5 b5 00 60 Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 4 dan kolom 3 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 6 RoundKeys 6 69 dd 83 87 f9 9d 98 de de 52 3c d0 60 e0 38 9e Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 6 yang di perlu dilakukan ialah lakukan hasil RoundKeys 6 kolom 1 dengan operasi XOR RoundKeys 4 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 6 : f. RoundKeys 7 Pencarian kunci ronde 7 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini : Transformation S-box RoundKeys 7 69 dd 83 17 f9 9d 98 1d de 52 3c 70 60 e0 38 0b 17 XOR d4 = c3 1d d2 cf 70 1d 6d 0b 63 68 Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan

operasi XOR dengan kolom 1 kunci ronde 5. Hasilnya adalah kolom 1 kunci ronde 7. Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 7 yang di perlu dilakukan ialah lakukan hasil RoundKeys 7 kolom 1 dengan operasi XOR RoundKeys 5 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 7. Berikut hasil dari RoundKeys 7 : RoundKeys 7 c3 1a 29 9c cf 13 bb 5b 6d 18 6a 8d 68 bd 7f 28 g. RoundKeys 8 Pencarian kunci ronde 8 dengan menggunakan transformasi RotWord. 8Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 7. RotWord RoundKeys 8 c3 1a 29 5b cf 13 bb 8d 6d 18 6a 28 68 bd 7f 9c Transformation S-Box RoundKeys 6 c3 1a 29 39 cf 13 bb 5d 6d 18 6a 34 68 bd 7f de Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). 39 XOR 69 XOR 08 = 58 5d f9 00 a4 34 de 00 ea de 60 00 be Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 6 dan kolom 4 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 8. RoundKeys 8 58 85 06 81 a4 39 a1 7f ea b8 84 54 be 5e 66 f8 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 8 yang di perlu dilakukan ialah lakukan hasil RoundKeys 8 kolom 1 dengan operasi XOR RoundKeys 6 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 8 : h. RoundKeys 9 Pencarian kunci ronde 9 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 Transformation S-box RoundKeys 9 58 85 06 0c a4 39 a1 d2 ea b8 84 20 be 5e 66 41 hlm.16). Seperti gambar dibawah ini : Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 7. Hasilnya adalah kolom 1 kunci ronde 9. 0c XOR c3 = cf d2 cf 1d 20 6d 4d 41 68 29 RoundKeys 9 cf d5 fc 60 1d 0e b5 ee 4d 55 3f b2 29 94 eb c3 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 9 yang di perlu dilakukan ialah lakukan hasil RoundKeys 9 kolom 1 dengan operasi XOR RoundKeys 7 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundkKeys 9. i. RoundKeys 10 Pencarian kunci ronde 10 dengan menggunakan transformasi RotWord.

Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 9. RotWord RoundKeys 10 cf d5 fc ee 1d 0e b5 b2 4d 55 3f c3 29 94 eb 60 Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys 10 cf d5 fc 28 1d 0e b5 37 4d 55 3f 2e 29 94 eb d0 28 XOR 58 XOR 10 = 60 37 a4 00 93 2e ea 00 c4 d0 be 00 6e Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 8 dan kolom 5 tabel RCon (hlm.17). Hasilnya adalah kolom 1 kunci ronde 10. RoundKeys 10 60 e5 e3 62 93 aa 0b 74 c4 7c f8 ac 6e 30 56 ae Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 10 yang di perlu dilakukan ialah lakukan hasil RoundKeys 10 kolom 1 dengan operasi XOR RoundKeys 8 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 10 : j. RoundKeys 11 Pencarian kunci ronde 11 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini : Transformation S-box RoundKeys 11 60 e5 e3 aa 93 aa 0b 92 c4 7c f8 91 6e 30 56 e4 Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 9. Hasilnya adalah kolom 1 kunci ronde 11. aa XOR cf = 65 92 1d 8f 91 4d dc e4 29 cd RoundKeys 11 65 b0 4c 2c 8f 81 34 da dc 89 b6 04 cd 59 b2 71 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 11 yang di perlu dilakukan ialah lakukan hasil RoundKeys 11 kolom 1 dengan operasi XOR RoundKeys 9 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 11. k. RoundKeys 12 Pencarian kunci ronde 12 dengan menggunakan transformasi RotWord. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 11. RotWord RoundKeys 12 65 b0 4c da 8f 81 34 04 dc 89 b6 71 cd 59 b2 2c Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys 12 65 b0 4c 57 8f 81 34 f2 dc 89 b6 a3 cd 59 b2 71 Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 10 dan kolom 6 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 12. 57 XOR 60 XOR 20 = 17 f2 93

00 61 a3 c4 00 67 71 6e 00 1f RoundKeys 12 17 f2 11 73 61 cb c0 b4 67 1b e3 4f 1f 2f 79 d7 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 12 yang di perlu dilakukan ialah lakukan hasil RoundKeys 12 kolom 1 dengan operasi XOR RoundKeys 10 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 12 : l. RoundKeys 13 Pencarian kunci ronde 13 tidak menggunakan transformasi RotWord. Jadi langsung saja di proses dengan transformasi s-box pada kolom terakhir matriks tersebut menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Seperti gambar dibawah ini : Transformation S-box RoundKeys 13 17 f2 11 8f 61 cb c0 8d 67 1b e3 84 1f 2f 79 0e Hasil transformasi SubBytes yang tadi dilakukan lalu kembali di proses dengan operasi XOR dengan kolom 1 kunci ronde 11. Hasilnya adalah kolom 1 kunci ronde 13. 8f XOR 65 = ea 8d 8f 02 84 dc 58 0e cd c3 RoundKeys 13 ea 5a 16 3a 02 83 b7 6d 58 d1 67 63 c3 9a 28 59 Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 13 yang di perlu dilakukan ialah lakukan hasil RoundKeys 13 kolom 1 dengan operasi XOR RoundKeys 11 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 13. m. RoundKeys 14 Pencarian kunci ronde 14 dengan menggunakan transformasi RotWord. Menggeser blok paling atas ke paling bawah pada kolom terakhir kunci ronde 13. RotWord RoundKeys 14 ea 5a 16 6d 02 83 b7 63 58 d1 67 59 c3 9a 28 3a Kemudian dilakukan operasi SubWord terhadap kolom paling kanan menggunakan tabel substitusi S-Box (tabel 2.8.1 hlm.16). Transformation S-Box RoundKeys 14 ea 5a 16 3c 02 83 b7 fb 58 d1 67 cb c3 9a 28 80 3c XOR 17 XOR 40 = 6b fb 61 00 9a cb 67 00 ac 80 1f 00 9f Hasil SubBytes tadi dilakukan operasi XOR dengan kolom 1 kunci ronde 12 dan kolom 6 tabel RCon (hlm.15). Hasilnya adalah kolom 1 kunci ronde 14. RoundKeys 14 6b 99 88 fb 9a 51 91 25 ac b7 54 1b 9f b0 c9 1e Selanjutnya mencari kolom ke 2 sampai kolom ke 4 kunci ronde 14 yang di perlu dilakukan ialah lakukan hasil RoundKeys 14 kolom 1 dengan operasi XOR RoundKeys 12 kolom 2 untuk mendapatkan isi kolom sebelahnya, lakukan begitu hingga mendapatkan hasil untuk RoundKeys 14 : Pada pengenkripsian blok pertama, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di

ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah di operasikan XOR dari RoundKeys 2 sampai 14. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan terlalu panjangnya tahapan tahapannya. Berikut hasil nilai di setiap rondenya : Hasil Ronde 2 Hasil Ronde 3 Hasil Ronde 4 35 19 1d 1b 03 8f a6 64 85 4d 97 59 00 70 e3 04 b2 1e d0 fa 3c 3f 7a 83 63 e0 14 1e d6 d9 9c be e9 ab 90 30 d0 eb 4d 44 7b 52 20 20 e1 89 6f 14 Hasil Ronde 5 Hasil Ronde 6 Hasil Ronde 7 e4 8d ae d4 79 31 1e f3 e1 45 da fe f5 64 66 e8 d3 7c 78 22 8d bf ea 8a 2a 57 d8 e0 9c d8 b5 41 b1 df 85 ea 3b de e6 d3 35 0a 4b fc 17 e7 6a 45 Hasil Ronde 8 Hasil Ronde 9 Hasil Ronde 10 52 bc 76 97 1f b3 d3 01 07 2c 69 7a 80 20 dd c5 62 a5 c8 d2 27 8f f1 93 9d 4f 81 93 c0 2a 16 19 44 7d 0c 90 ee 17 1a e9 ae 76 a0 56 0d 68 83 be Hasil Ronde 11 Hasil Ronde 12 Hasil Ronde 13 31 1d 84 df 18 9c fe 6e c1 c0 26 90 1b de 06 75 25 ff d5 a1 8d 3c 8e 3d 64 fb 6a d8 63 3e 4e 1a 3c e6 08 38 53 be ef f4 37 17 01 6b e8 6d cc 96 Hasil Ronde 14 13 23 7f 9b 71 48 b6 78 9c b0 bf 95 0f 2b f5 55 Pada ronde terakhir, yaitu ronde ke 14, process MixColumn ditiadakan, atau dilewatkan. Hasil ronde ke 14 inilah yang menjadi hasil akhir dari pengenkripsian blok pertama sekaligus menjadi chippertext pada blok pertama dan menjadi IV untuk blok selanjutnya. Hasil ronde 14 adalah: 4.2.3 Enkripsi Blok Kedua Pada pemrosesan plain text block kedua, diperlukan penambahan IV, IV yang digunakan ialah hasil dari chippertext block pertama dan aeskey yang sama. Pada enkripsi blok pertama, RoundKeys 2 sampai 14 belum diketahui oleh sebab itu kita harus mencari tahu terlebih dahulu, pada enkripsi blok kedua penulis tidak perlu mencari RoundKeys dikarenakan RoundKeys 1 sampai 14 adalah sama dengan RoundKeys pada enkripsi blok pertama. Rumus CBC blok pertama akan digambarkan sebagai mana berikut ini : IV : 13719c0f 2348b02b 7fb6bff5 9b789555 (16 bytes = 128 bits) aeskey : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 (32 bytes = 256 bits) block2 : 63726574 20466963 68696572 21030303 Berikut rumus CBC block kedua : Maka dari itu hasi akhir dari kedua enkripsi diatas yaitu enkripsi blok pertama dan enkripsi blok kedua sebagai rumus berikut : IV ⊕ Plaintext block 2 ⊕ RoundKey 0 13 23 7f 9b 63 20 68 21 36 e4 d2 b0 71 48 b6 78 72 46 69 03 08 4e 68 22 9c b0 bf 95 65 69 65 03 bc

a6 eb 60 0f 2b f5 55 74 63 72 03 a1 c4 6d 26 Tahapan selanjutnya ialah pre round atau initialization key, dimana dilakukannya operasi XOR pada IV dengan plaintext block 2 dengan RoundKeys 0 yang sudah diketahui sebelumnya agar mendapatkasih hasil untuk diproses selanjutnya. Berikut gambarannya : Hasil 46 e7 c5 0a 0b 40 b7 59 45 7f 31 f6 da 8c ea 70 a. Proses SubBytes Pada proses SubBytes ini, hasil dari operasi XOR di pre-round akan di di subtitusikan dengan nilai yang ada di table S-Box (tabel 2.8.1 hlm.16). proses dari SubBytes ialah sebagai berikut : 46 e7 c5 0a 0b 40 b7 59 45 7f 31 f6 da 8c ea 70 Lakukan proses subtitusi S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (4) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (6) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil SubByte. Seperti contoh gambar berikut : Setelah dilakukan subtitusi S-box pada proses SubBytes pada nilai 46, maka didapatkan hasil yaitu 5a. Lalu lanjutkan subtitusi S-box pada kolom [2.1] matriks diatas yaitu nilai 0b. Pada proses Subtitusi S-box pada nilai 0b, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes 0b mendapatkan hasil nilai 2b. Lalu lanjutkan subtitusi S-box pada kolom [2.3] matriks diatas yaitu nilai 45. Pada proses Subtitusi S-box pada nilai 45, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes 45 mendapatkan hasil nilai 6e. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai da. Pada proses Subtitusi S-box pada nilai cd, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi S-box proses SubBytes da mendapatkan hasil nilai 57. Lalu lanjutkan subtitusi S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table S-Box proses SubBytes. Setelah dilakukan proses SubBytes pada keseluruhan nilai, maka didapatkan hasil dari proses SubBytes ialah sebagai berikut: Hasil SubBytes 5a 94 a6 67 2b 09 a9 cb 6e d2 c7 42 57 64 87 51 b. Proses ShiftRows Lalu tahapan selanjutnya ialah proses ShiftRows. Hasil dari proses SubBytes tadi akan di geser beberapa bit di dalam proses ini, berikut gambaran yang terjadi pada proses ShiftRows ini :

Pada nilai yang ditandai akan di putar lebih 1 byte, seperti berikut ini 5a 94 a6 67 5a 94 a6 67 2b 09 a9 cb 09 a9 cb 2b 6e d2 c7 42 6e d2 c7 42 57 64 87 51 57 64 87 51 Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut: 5a 94 a6 67 5a 94 a6 67 09 a9 cb 2b 09 a9 cb 2b 6e d2 c7 42 c7 42 6e d2 57 64 87 51 57 64 87 51 Pada nilai yang ditandai akan diputar lebih dari tiga byte, seperti berikut: 5a 94 a6 67 5a 94 a6 67 09 a9 cb 2b 09 a9 cb 2b c7 42 6e d2 c7 42 6e d2 57 64 87 51 51 57 64 87 Hasil dari proses ShiftRows ialah sebagai barikut : Hasil ShiftRows 5a 94 a6 67 09 a9 cb 2b c7 42 6e d2 51 57 64 87 c. Proses MixColumns Proses selanjutnya adalah proses MixColumns, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada prose ShiftRows tadi dengan matriks yang telah di tetapkan oleh Rijndael. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial dan pergeseran beberapa bit. Berikut merupakan proses dari MixColumns : 02 03 01 01 X 5a 94 a6 67 C11 C12 C13 C14 01 02 03 01 09 a9 cb 2b C21 C22 C23 C24 01 01 02 03 c7 42 6e d2 C31 C32 C33 C34 03 01 01 02 51 57 64 87 C41 C42 C43 C44 Operasi perkalian disini maksudnya adalah : 1. Perkalian dengan 01 artinya tidak berubah. 2. Perkalian dengan 02 artinya perkalian polynomial, nilai tersebut terlebih dahulu diubah kedalam bentuk biner dan dilakukan perkalian polynomial dan modulo untuk nilai binernya. 3. Perkalian dengan 03 artinya perkalian polynomial, nilai tersebut terlebih dahulu diubah kedalam bentuk biner dan dilakukan perkalian polynomial dan modulo untuk nilai binernya. Berikut ini merupakan perhitungan pencarian nilai untuk C11 : 02 03 01 01 X 5a 01 02 03 01 09 01 01 02 03 c7 03 01 01 02 51 C11 = {02.5a} ⊕ {03.09} ⊕ {01.c7} ⊕ {01.51} · {02.5a} 02 = 0000 0010 = x 5a = 0101 1010 = x6 1+ x4 + x3 + x = (x)(x6 + x4 + x3 + x) = x7 + x5 + x4 + x2 = 1011 0100 (b4) · {03.09} 03 = 0000 0011 = x + 1 09 = 0000 1001 = x3 + 1 = (x + 1)(x3 + 1) = (x4 + x) + (x + 1) = x4 + x3 + x + 1 = 0001 1011 (1b) · {01.c7} = c7 · {01.51} = 51 Perhitungan : C11 = {02.5a} ⊕ {03.09} ⊕ {01.c7} ⊕ {01.51} C11 = {b4} ⊕ {1b} ⊕ {c7} ⊕ {51} = 39 Berikut ini merupakan perhitungan pencarian nilai untuk C21 : 02 03 01 01 X 5a 01 02 03 01 09 01 01 02 03 c7 03 01 01 02 51 C21 = {01.5a} ⊕ {02.09} ⊕ {03.c7} ⊕ {01.51} · {01.5a} = 5a · {02.09} 02 = 0000 0010 = x 09 = 0000 1001 = x3 + 1

= (x)(3 1+ 1) = x7 + x5 + x4 + x2 = 1011 0100 (12) · {03.c7} 03 = 0000 0011 = x + 1 c7 = 1100 0111 = x7 + x6 + x2 + x + 1 = (x + 1)(x7 + x6 + x2 + x + 1) = (x8 + x7 + x3 + x2 + x) + (x7 + x6 + x2 + x + 1) = x8 + x6 + x3 + 1 = x8 + x6 + x3 + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 0100 1001 1 0001 1011 (XOR) 0101 0010 (52) · {01.51} = 51 Perhitungan : C21 = {01.5a} ⊕ {02.09} ⊕ {03.c7} ⊕ {01.51} C21 = {5a} ⊕ {12} ⊕ {52} ⊕ {51} = 4b Berikut ini merupakan perhitungan pencarian nilai untuk C31 : 02 03 01 01 X 5a 01 02 03 01 09 01 01 02 03 c7 03 01 01 02 51 C31 = {01.5a} ⊕ {01.09} ⊕ {02.c7} ⊕ {03.51} · {01.5a} = 5a · {01.09} = 09 · {02.c7} 02 = 0000 0010 = x c7 = 1100 0111 = x7 + x6 + x2 + x + 1 = (x)(x7 + x6 + x2 + x + 1) = x8 + x7 + x3 + x2 + x = x8 + x7 + x3 + x2 + x modulo x8 + x4 + x3 + x1 + 1 = 1 1000 1110 1 0001 1011 (XOR) 1001 0101 (95) · {03.51} 03 = 0000 0011 = x + 1 51 = 0101 0001 = x6 + x4 + 1 = (x + 1)(x6 + x4 + 1) = (x7 + x5 + x) + (x6 + x4 + 1) = x7 + x6 + x5 + x4 + x + 1 Perhitungan : C31 = {01.5a} ⊕ {01.09} ⊕ {02.c7} ⊕ {03.51} C31 = {5a} ⊕ {09} ⊕ {95} ⊕ {f3} = 35 Berikut ini merupakan perhitungan pencarian nilai untuk C41: 02 03 01 01 X 5a 01 02 03 01 09 01 01 02 03 c7 03 01 01 02 51 C41 = {03.5a} ⊕ {01.09} ⊕ {01.c7} ⊕ {02.51} · {03.5a} 03 = 0000 0011 = x + 1 5a = 0101 1010 = x6 + x4 + x3 + x = (x + 1)(x6 + x4 + x3 + x) = (x7 + x5 + x4 + x2) + (x6 + x4 + x3 + x) = x7 + x6 + x5 + x3 + x2 + x = 1110 1110 (ee) · {01.09} = 09 · {01.c7} = c7 · {02.51} 02 = 0000 0010 = x 51 = 0101 0001 = x6 + x4 + 1 = (x)(x6 + x4 + 1) = x7 + x5 + x = 1010 0010 (a2) Perhitungan : C41 = {03.5a} ⊕ {01.09} ⊕ {01.c7} ⊕ {02.51 = {ee} ⊕ {09} ⊕ {c7} ⊕ {a2} = 82 Untuk mendapat hasil yang lainnya C12, C22, C32, C42, C13, C23, C33...C44 lakukan perhitungan perkalian polynomial dengan cara seperti diatas. Berikut hasilnya : C12 = {02.94} ⊕ {03.a9} ⊕ {01.42} ⊕ {01.57} = {33} ⊕ {e0} ⊕ {42} ⊕ {57} = c6 C22 = {01.94} ⊕ {02.a9} ⊕ {03.42} ⊕ {01.57} = {94} ⊕ {49} ⊕ {c6} ⊕ {57} = 4c C32 = {01.94} ⊕ {01.a9} ⊕ {02.42} ⊕ {03.57} = {94} ⊕ {a9} ⊕ {84} ⊕ {f9} = 40 C42 = {03.94} ⊕ {01.a9} ⊕ {01.42} ⊕ {02.57} = {a7} ⊕ {a9} ⊕ {42} ⊕ {ae} = e2 C13 = {02.a6} ⊕ {03.cb} ⊕ {01.6e} ⊕ {01.64} = {57} ⊕ {46} ⊕ {6e} ⊕ {64} = 1b C23 = {01.a6} ⊕ {02.cb} ⊕ {03.6e} ⊕ {01.64} = {a6} ⊕ {8d} ⊕ {b2} ⊕ {64} = fd C33 = {01.a6} ⊕ {01.cb} ⊕ {02.6e} ⊕ {03.64} = {a6} ⊕ {cb} ⊕ {dc} ⊕ {ac} = 1d C43 = {03.a6} ⊕ {01.cb} ⊕ {01.6e} ⊕ {02.64} = {f1} ⊕ {cb} ⊕ {6e} ⊕ {c8} = 9c C14 =

{02.67} ⊕ {03.2b} ⊕ {01.d2} ⊕ {01.87} = {ce} ⊕ {7d} ⊕ {d2} ⊕ {87} = e6 C24 = {01.67} ⊕ {02.2b} ⊕ {03.d2} ⊕ {01.87} = {67} ⊕ {56} ⊕ {6d} ⊕ {87} = db C34 = {01.67} ⊕ {01.2b} ⊕ {02.d2} ⊕ {03.87} = {67} ⊕ {2b} ⊕ {bf} ⊕ {92} = 61 C44 = {03.67} ⊕ {01.2b} ⊕ {01.d2} ⊕ {02.87} = {a9} ⊕ {2b} ⊕ {d2} ⊕ {15} = 45 Hasil MixColumn 39 c6 1b e6 4b 4c fd db 35 40 1d 61 82 e2 9c 45 Hasil dari proses MixColumn adalah sebagai berikut : d. Proses AddRoundKey Proses selanjutnya ialah AddRoundKey, setelah hasil yang didapat dalam proses MixColumns selanjutnya akan melakukan operasi XOR dengan RoundKeys 1 terhadap hasil dari MixColumns. Seperti gambar berikut ini : Hasil MixColumn XOR KeyRound1 = Hasil Ronde 1 39 c6 1b e6 98 2b e7 6c a1 ed Fc 8a 4b 4c fd db 92 86 7a 3c d9 ca 87 e7 35 40 1d 61 c0 bb 6f 92 f5 fb 72 f3 82 e2 9c 45 b4 f1 a1 82 36 13 3d c7 Setelah mendapat hasi dari AddRoundKey ronde pertama, selanjutnya pencarian nilai ronde kedua hingga ronde ketiga belas dan melalui proses proses sebelumnya, yaitu proses SubBytes, proses ShiftRows dan proses MixColumns dan pencarian kunci di tiap rondenya untuk pemrosesan XOR dengan MixColumns (proses AddRoundKey) nanti. Yang kita ketahui kunci yang kita gunakan disini ialah “xyz” yang telah di proses dengan fungsi hash SHA-256 dan menghasilkan 32 bytes atau setara dengan 256 bits, yaitu sebagai berikut : 3608bca1e44ea6c4 d268eb6db0226026 9892c0b42b86bbf1 e77a6fa16c3c9282 RoundKeys 1 98 2b e7 6c 92 86 7a 3c c0 bb 6f 92 b4 f1 a1 82 Karena tiap ronde kunci aes hanya berisikan 16 bytes tiap rondenya, maka penulis akan membagi dua matriks kunci tersebut dan menaruhnya kedalam matriks 4x4 atau 16 bytes, seperti sebagai berikut ; RoundKeys 0 36 e4 d2 b0 08 4e 68 22 bc a6 eb 60 a1 c4 6d 26 Disebutkan sebelumnya, bahwa RoundKeys enkripsi blok pertama dan blok kedua adalah sama, sehingga kita tidak perlu mencari lagi RoundKeys 2 sampai 14, dikarenakan RoundKeys nya sama dengan pengenkripsian blok pertama seperti diatas. Pada pengenkripsian blok kedua, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah di operasikan XOR oleh RoundKeys 2 sampai 14. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan terlalu panjangnya tahapan

tahapannya. Berikut hasil nilai di setiap rondenya : Hasil Ronde 2 Hasil Ronde 3 Hasil Ronde 4 a2 a3 ad d1 6d 72 a1 31 4a cf 05 69 9b 60 ae 61 66 cc 7a 19 58 ac 3f dc 38 5e 96 be 55 cc 8a e5 29 46 9c bf 04 da 1b 88 da 1d f5 de fe ed f5 5b Hasil Ronde 5 Hasil Ronde 6 Hasil Ronde 7 2c fa 84 5e f7 Ef 8b 46 b6 75 71 6e 7d 1f 75 c5 a1 1c 4b 5a c6 97 ea 19 b6 4c 31 f1 c1 24 42 e3 9b 69 87 e3 3f 40 f6 a0 2f 41 23 24 0c 4f e3 8e Hasil Ronde 8 Hasil Ronde 9 Hasil Ronde 10 49 d9 ac 8b 78 c2 6d cf ef 87 2a d9 c1 7c 09 92 a5 4b 92 b9 7f 85 8b e5 29 99 4c a5 47 e5 cf fe 27 bd ef 20 c1 93 4b 2d f6 3c 0a 6f c3 2f 29 3a Hasil Ronde 11 Hasil Ronde 12 Hasil Ronde 13 3d 40 34 f4 f7 36 a0 7f 40 89 64 35 9f 00 22 7b 81 a5 3b 2b 6b 89 d0 42 aa a4 36 e3 3b 18 4b 43 15 1b 6c e7 e4 b6 b9 d7 0c 64 40 33 53 8a 86 87 Hasil Ronde 14 62 3e Cb 6d 3d 21 bd 5a fc 23 0d b4 88 5d b7 5a Pada ronde terakhir, yaitu ronde ke 14, process MixColumn ditiadakan, atau dilewatkan. Hasil ronde ke 14 inilah yang menjadi hasil akhir dari pengenkripsian blok kedua sekaligus menjadi chippertext pada blok kedua dan menjadi akhir dari penjelasan proses pengenkripsian aes. Berikut ini hasil ronde 14 adalah: Pada rumus hasil yang telah disebutkan sebelumnya diatas bahwa : Maka dari itu, chippertext dari pesan asli “Hello this is Secret Fichier!” dengan kunci “xyz” ialah 13719c0f2348b02b 7fb6bff59b789555 623dfc883e21235d cbbd0db76d5ab45a. Berikut penulis tampilkan hasil enkripsi text tersebut dengan Secret Fichier, dan apabila di convert akan menghasilkan hexadecimal seperti diatas. Gambar 4.2.3 Hasil Enkripsi 4.2.4 Dekripsi File Sebuah file yang sudah di enkripsi tidak dapat dibaca dan dipahami oleh siapapun. Agar file tersebut dikembalikan seperti semula dan dapat terbaca lagi, maka diperlukan teknik dekripsi untuk mengembalikannya. Pada penjelasan sebelumnya, penulis telah menjelaskan tahapan tahapan algoritma SHA dan AES mode CBC untuk enkripsi file pada Secret Fichier. oleh karena itu, pada proses dekripsi file juga harus menggunakan algritma yang sama. Proses deskripsi pesan juga akan berkebalikan dari proses enkripsi, berkebalikan proses disebut sebagai inverse. Didalam pengenkripsian block tersebut, ada beberapa tahapan lagi untuk menghasilkan sebuah plain text kembali, yaitu pencarian inverse ronde kunci 2 hingga 14, pengdekripsian ronde pertama dengan tahapan Inverse SubBytes, Inverse ShiftRows, Inverse MixColumns dan AddRoundKey. Semua tahapan

tersebut akan dijelaskan sebagai berikut. Pada contoh pendekripsian kali ini, penulis akan menggunakan chiper text hasil dari pengenkripsian sebelumnya untuk menjelaskan tahap tahap algoritma AES mode CBC dan sha bekerja pada penerapan algoritma tersebut untuk penelitian pengamanan data penulis. Oleh karena siapkan chiper text dan kunci yang akan didekripsi kali ini, perlu diingat bahwa kunci yang digunakan harus sama seperti kunci yang digunakan pada proses enkripsi. Berikut chiper text dan kunci yang akan digunakan ialah : Chiper text : 13719c0f2348b02b 7fb6bff59b789555 623dfc883e21235d cbbd0db76d5ab45a Kunci : xyz Tahapan selanjutnya ialah kunci yang kita gunakan disini harus telah di prosess terlebih dahulu dengan fungsi hash atau yang dimaksud dengan istilah Key Derivation (penurunan kunci) dengan SHA256. Pada proses sebelumnya, kunci "xyz" telah di proses dengan SHA-256 dan menghasilkan 64 hex digits atau setara dengan 32 bytes (256 bits) dengan kententuan SHA-2. Berikut hasil dari kunci setelah di proses dengan SHA-256. Chipertext : 13 71 9c 0f 23 48 b0 2b 7f b6 bf f5 9b 78 95 55 62 3d fc 88 3e 21 23 5d cb bd 0d b7 6d 5a b4 5a Kunci : 36 08 bc a1 e4 4e a6 c4 d2 68 eb 6d b0 22 60 26 98 92 c0 b4 2b 86 bb f1 e7 7a 6f a1 6c 3c 92 82 Dikarenakan AES hanya memiliki panjang block dengan ukuran 128 bit sedangkan chiper text yang penulis gunakan disini ialah chiper text dengan ukuran 32 bytes (32 x 8 = 256 bit) karena adanya padding, maka dari itu chiper text akan di bagi menjadi 2 block yang masing masing block tersebut mempiliki panjang 128 bit. Berikut gambaran pembagian block : 13 71 9c 0f 23 48 b0 2b 7f b6 bf f5 9b 78 95 55 |Chipertext block 1| 62 3d fc 88 3e 21 23 5d cb bd 0d b7 6d 5a b4 5a |Chipertext block 2| Chipertext Block 1 13 23 7f 9b 71 48 b6 78 9c b0 bf 95 0f 2b f5 55 Chipertext Block 2 62 3e cb 6d 3d 21 bd 5a fc 23 0d b4 88 5d b7 5a Agar lebih mudah penggambaran pada masing masing block, maka penulis akan membuat matriks dengan ukuran 4 x 4 untuk diisi dengan masing masing nilai heksadesimal kedalam block block yang tadi di gambarkan. Dalam pemrosesan AES mode CBC, rangkaian bit-bit pada chipertext dibagi menjadi blok blok bit dengan panjang yang sama. Mode CBC memerlukan IV (initialization vector) untuk menggabungkan dengan chipertext pada tahapan akhir pengdekripsian chipertext aes. Tahapan proses mode CBC akan ditunjukan

pada gambar dibawah ini. Pada pemrosesan dekripsi CBC, diperlukannya penambahan IV atau yang disebut sebagai IV (initialization vector) pada tahap akhir chipertext dihasilkan. Dalam pemrosesan dengan algoritma aes sudah disebutkan sebelumnya bahwa ada dua input yang dibutuhkan, yaitu input yang pertama ialah chipertext yang akan di proses pendekripsiannya, yang kedua ialah kunci yang telah di proses dengan fungsi hash sebagai aeskey pada aes itu sendiri. Pada penjelasan selanjutnya, penulis akan menjelaskan tahapan pendekripsian block pertama dengan AES mode CBC. 4.2.5 Dekripsi Blok Pertama Seperti gambar dekripsi aes mode cbc diatas, pada pemrosesan chipertext block pertama apabila block pertama sudah di proses, maka hasil dari chipertext akhir ronde akan dioperasikan XOR pada IV pertama, yaitu bernilai “00” sebanyak 16 hex digit dan akan menghasilkan plaintext block pertama. Selanjutnya pada pedekripsian block kedua, chipertext block pertama akan menjadi IV block kedua yang akan dioperasikan XOR pada hasil akhir ronde cipertext block kedua dan menghasilkan plaintext block kedua. Sebelum tahapan operasi XOR IV, chipertext akan melalui tahapan AddRoundKey, Inverse MixColumn, Inverse ShiftRows, Inverse SubBytes sebanyak 14 kali putaran sebagaimana dari gambaran proses dekripsi AES dibawah ini : Chipertext Block 1 13 23 7f 9b 71 48 b6 78 9c b0 bf 95 0f 2b f5 55 setelah mendapat chipertext block pertama dan kuncinya, maka kita akan menaruh cipertext block pertama tersebut kedalam matriks 4 x 4 agar lebih mudah dalam pemrosesan dekripsinya. Untuk RoundKeys Schedule sudah didapat pada proses sebelumnya (pada tahapan enkripsi file hal. 61) yaitu RoundKeys 1 sampai 14 akan digunakan Kembali pada proses ini. Pada proses dekripsi ini penulis memakai kunci yang sama yaitu kunci “xyz” yang sudah dilakukan proses penjadwalan RoundKeys. Berikut matriks chipertext block pertama dan keseluruhan hasil RoundKeys Schedule yang telah didapat (karena pada proses inverse (terbalik) ini, penulis akan menggunakan RoundKeys 14 terlebih dahulu lalu RoundKeys 0 di akhir tahapan) : RoundKeys 14 RoundKeys 13 RoundKeys 12 6b 99 88 fb ea 5a 16 3a 17 f2 11 73 9a 51 91 25 02 83 b7 6d 61 cb c0 b4 ac b7 54 1b 58 d1 67 63 67 1b e3 4f 9f b0 c9 1e c3 9a 28 59 1f 2f 79 d7 RoundKeys 11 RoundKeys 10 RoundKeys 9 65 b0 4c 2c 60 e5 e3 62 cf d5 fc 60 8f 81 34 da 93 aa 0b 74 1d

0e b5 ee dc 89 b6 04 c4 7c f8 ac 4d 55 3f b2 cd 59 b2 71 6e 30 56 ae 29 94 eb c3 RoundKeys 8 RoundKeys 7 RoundKeys 6 58 85 06 81 c3 1a 29 9c 69 dd 83 87 a4 39 a1 7f cf 13 bb 5b f9 9d 98 de ea b8 84 54 6d 18 6a 8d de 52 3c d0 be 5e 66 f8 68 bd 7f 28 60 e0 38 9e RoundKeys 5 RoundKeys 4 RoundKeys 3 d4 d9 33 b5 8c b4 5e 04 26 0d ea 86 d2 dc a8 e0 6d 64 05 46 88 0e 74 48 1d 75 72 e7 85 8c 6e ec d3 68 07 95 63 d5 c2 57 b5 80 d8 a6 47 b6 17 95 RoundKeys 2 RoundKeys 1 RoundKeys 0 dc 38 ea 5a 98 2b e7 6c 36 e4 d2 b0 47 09 61 43 92 86 7a 3c 08 4e 68 22 af 09 e2 82 c0 bb 6f 92 bc a6 eb 60 f1 35 58 7e b4 f1 a1 82 a1 c4 6d 26 Seperti gambar tahapan dekripsi aes diatas, tahapan pertama yang akan dilakukan ialah proses ronde pertama yang diawali dengan proses AddRoundKey kunci ronde 14. Pada proses dekripsi AddRoundKey dimulai dari ronde terakhir pada proses enkripsi karena kebalikannya, yaitu dimulai dari ronde terakhir ronde 14 dengan RoundKeys terakhir 14 sampai ke ronde 1 dengan RoundKeys 0. Pada tahap AddRoundKey terjadi perhitungan XOR antara chipertext tiap blocknya dengan RoundKeys. 2XOR merupakan kepanjangan dari Exclusive OR yang mana keluarannya akan berlogika 1 apabila inputannya berbeda, namun apabila semua inputanya sama maka akan memberikan keluarannya 0. Proses XOR dilakukan pada masing masing blok pada matriks, seperti matriks chipertext block 1 kolom [1,1] XOR dengan matriks RoundKeys 14 kolom [1,1] dan seterusnya. Berikut akan disimulasikan XOR pada matriks chipertext blok 1 dengan matriks RoundKeys 14. Untuk dapat melakukan operasi XOR, nilai heksadecimal yang ada di chipertext dan di kunci tadi harus diubah menjadi nilai biner berukuran 8 bit terlebih dahulu. Seperti contoh berikut : 13 = 0001 0011 6b = 0110 1011 20 0 0 1 0 0 1 1 ⊕ 0 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 Proses XOR disimulasikan seperti dibawah ini : Nilai heksadesimal dari 0111 1000 ialah 78, jadi 13 ⊕ 6b = 78 Hitung keseluruhan nilai XOR di kolom kolom matriks chipertext blok 1 dengan RoundKeys 14 hingga mengahasilkan blok matriks berukuran 4 x 4. 71 ⊕ 9a = eb 9c ⊕ ac = 30 0f ⊕ 9f = 90
23 ⊕ 99 = ba 48 ⊕ 51 = 19 b0 ⊕ b7 = 07 2b ⊕ b0 = 9b 7f ⊕ 88 = f7
b6 ⊕ 91 = 27 bf ⊕ 54 = eb f5 ⊕ c9 = 3c 9b ⊕ fb = 60 78 ⊕ 25 = 5d
95 ⊕ 1b = 8e 55 ⊕ 1e = 4b Berikut matriks dari chipertext block 1 XOR RoundKeys 14

Chipertext block 1 ⊕ RoundKeys 14 = Hasil 13 23 7f 9b 6b 99 88 fb 78 ba f7 60 71 48 b6 78 9a 51 91 25 eb 19 27 5d 9c b0 bf 95 ac b7 54 1b 30 07 eb 8e 0f 2b f5 55 9f b0 c9 1e 90 9b 3c 4b Pada tahap selanjutnya dilakukan proses Inverse ShiftRows, Inverse SubByte pada putaran ke 14 (tahapan MixColumns dilewatkan). Lalu pindah ke putaran 13 ada tahap AddRoundKeys dengan RoundKeys 13 lalu ada Inverse MixColumns, Inverse ShiftRows dan Inverse SubBytes sampai seterusnya dan di ulang ulang sampai ke putaran 1 dan mendapatkan hasil plaintextnya. Berikut penjelasan tahapan selanjutnya a. Proses Inverse ShiftRows Hasil dari proses AddRoundKeys tadi akan di geser beberapa bit di dalam proses ini proses ini berkebalikan dari proses ShiftRows, berikut gambaran yang terjadi pada proses Inverse ShiftRows ini : Pada nilai yang ditandai akan di putar kedepan tiga bit, seperti berikut ini 78 ba f7 60 78 ba f7 60 eb 19 27 5d 5d eb 19 27 30 07 eb 8e 30 07 eb 8e 90 9b 3c 4b 90 9b 3c 4b Pada nilai yang ditandai akan diputar lebih dari dua byte, seperti berikut 78 ba f7 60 78 ba f7 60 5d eb 19 27 5d eb 19 27 30 07 eb 8e eb 8e 30 07 90 9b 3c 4b 90 9b 3c 4b Pada nilai yang ditandai akan diputar lebih dari satu byte, seperti berikut 78 ba f7 60 78 ba f7 60 5d eb 19 27 5d eb 19 27 eb 8e 30 07 eb 8e 30 07 90 9b 3c 4b 9b 3c 4b 90 Hasil Inverse ShiftRows 78 ba f7 60 5d eb 19 27 eb 8e 30 07 9b 3c 4b 90 Hasil dari Inverse ShiftRows ialah sebagai berikut : b. Proses Inverse SubBytes Pada proses Inverse SubBytes ini, hasil Inverse ShiftRows akan di subtitusikan dengan nilai yang ada di table Inverse S-Box (tabel 2.9.3 hlm.19). proses dari Inverse SubBytes ialah sebagai berikut : 78 ba f7 60 5d eb 19 27 eb 8e 30 07 9b 3c 4b 90 Lakukan proses subtitusi Inverse S-Box terhadap nilai nilai heksadecimal yang ada didalam matriks diatas, dimulai dari kolom [1.1, [2.1] dan seterusnya sampai di kolom [4.4]. Nilai heksadecimal pertama (7) dialokasikan sebagai sumbu x, sedangkan nilai heksadecimal kedua (8) dialokasikan sebagai sumbu y dan menghasilkan garis perpotongan untuk hasil Inverse SubByte. Seperti contoh gambar berikut : Setelah dilakukan subtitusi Inverse S-box pada proses Inverse SubBytes pada nilai 78, maka didapatkan hasil yaitu c1. Lalu lanjutkan subtitusi Inverse S-box pada kolom [2.1] matriks diatas yaitu nilai 5d. Pada proses Subtitusi Inverse S-box pada nilai 5d, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi

Inverse S-box proses Inverse SubBytes 5d mendapatkan hasil nilai 8d. Lalu lanjutkan subtitusi Inverse S-box pada kolom [3.1] matriks diatas yaitu nilai eb. Pada proses Subtitusi Inverse S-box pada nilai eb, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi Inverse S-box proses Inverse SubBytes eb mendapatkan hasil nilai 3c. Lalu lanjutkan subtitusi S-box pada kolom [4.1] matriks diatas yaitu nilai 9b. Pada proses Subtitusi Inverse S-box pada nilai 9b, dapat dilihat bahwa garis berpotongan pada gambar diatas. Oleh karena itu hasil dari subtitusi Inverse S-box proses Inverse SubBytes 9b mendapatkan hasil nilai e8. Lalu lanjutkan subtitusi Inverse S-box pada kolom [1.2], [2.2] dan seterusnya hingga nilai heksadesimal matriks diatas sudah di subtitusikan kedalam table Inverse S-Box proses Inverse SubBytes. Setelah dilakukan proses Inverse SubBytes pada keseluruhan nilai, maka didapatkan hasil dari proses Inverse SubBytes ialah sebagai berikut : Hasil Inverse SubBytes c1 c0 26 90 8d 3c 8e 3d 3c e6 08 38 e8 6d cc 96 Hasil Ronde 14 c1 c0 26 90 8d 3c 8e 3d 3c e6 08 38 e8 6d cc 96 Setelah mendapat hasil dari Inverse SubBytes sekaligues menjadi Hasil Ronde ke 14. Selanjutnya adalah tahap dimana proses ini melakukan 13 kali perulangan sesuai dengan proses dekripsi AES-256 (Gambar 2.9 hlm.20). Tahapan selanjutnya ialah putaran selanjutnya mencari nilai dari hasil ronde ke 13 dan seterusnya sampai ke ronde 1. a. Proses AddRoundKey (Ronde 13) Hasil Ronde 14 ⊕ RoundKeys 13 = Hasil c1 c0 26 90 ea 5a 16 3a 2b 9a 30 aa 8d 3c 8e 3d 02 83 b7 6d 8f bf 39 50 3c e6 08 38 58 d1 67 63 64 37 6f 5b e8 6d cc 96 c3 9a 28 59 2b f7 e4 cf Pada tahapan ini sama seperti tahapan AddRoundKey diatas, hasil dari ronde 14 dilakukan operasi XOR dengan RoundKeys 13. Seperti gambar berikut ini : a. Proses Inverse MixColumns (Ronde 13) Proses selanjutnya adalah proses Inverse MixColumns, pada proses ini terjadi perkalian matriks 4 x 4 antara hasil yang telah di peroleh pada proses Inverse ShiftRows tadi dengan matriks yang telah di tetapkan oleh Rijndael. Perkalian matriks ini sekilas seperti perkalian matriks 4 x 4 biasa, akan tetapi didalam setiap perkalian nilai heksadecimal nya terjadi perkalian polynomial. Berikut merupakan proses dari Inverse MixColumns : 0e 0b 0d 09 X 2b 9a 30 aa C11 C12 C13 C14 09 0e 0b 0d 8f bf 39 50 C21 C22 C23 C24 0d 09 0e 0b 64 37 6f 5b C31 C32 C33 C34 0b 0d

09 0e 2b f7 e4 cf C41 C42 C43 C44 Berikut ini merupakan perhitungan pencarian nilai untuk C11 : 0e 0b 0d 09 X 2b 09 0e 0b 0d 8f 0d 09 0e 0b 64 0b 0d 09 0e 2b C11 = {0e.2b} ⊕ {0b.8f} ⊕ {0d.64} ⊕ {09.2b} · {0e.2b} 0e = 0000 1110 = x3 + x2 + x 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x2 + x)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x7 + x5 + x3 + x2) + (x6 + x4 + x2 + x) = x8 + x7 + x5 + x = x8 + x7 + x5 + x modulo x8 + x4 + x3 + x1 + 1 = 1 1010 0010 1 0001 1011 (XOR) 1011 1001 (b9) · {0b.8f} 0b = 0000 1011 = x3 + x + 1 8f = 1000 1111 = x7 + x3 + x2 + x + 1 = (x3 + x + 1)(x7 + x3 + x2 + x + 1) = (x10 + x6 + x5 + x4 + x3) + (x8 + x4 + x3 + x2 + x) + (x7 + x3 + x2 + x + 1) = x10 + x8 + x7 + x6 + x5 + 1 = x10 + x8 + x7 + x6 + x5 + 1 modulo x8 + x4 + x3 + x1 + 1 = 101 1110 0001 100 0110 11 (XOR) 1 1000 1101 1 0001 1011 1001 1110 (9e) · {0d.64} 0d = 0000 1101 = x3 + x2 + 1 64 = 0110 0100 = x6 + x5 + x2 = (x3 + x2 + 1)(x6 + x5 + x2) = (x9 + x8 + x5) + (x8 + x7 + x4) + (x6 + x5 + x2) = x9 + x7 + x6 + x4 + x2 = x9 + x7 + x6 + x4 + x2 modulo x8 + x4 + x3 + x1 + 1 = 1011 0101 00 1000 1101 1 (XOR) 1110 0010 (e2) · {09.2b} 09 = 0000 1001 = x3 + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x5 + x3 + x + 1) = x8 + x6 + x5 + x4 + x + 1 = x8 + x6 + x5 + x4 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 0111 0011 1 0001 1011 (XOR) 0110 1000 (68) C11 = {b9} ⊕ {9e} ⊕ {e2} ⊕ {68} = ad Berikut ini merupakan perhitungan pencarian nilai untuk C21 : 0e 0b 0d 09 X 2b 09 0e 0b 0d 8f 0d 09 0e 0b 64 0b 0d 09 0e 2b C21 = {09.2b} ⊕ {0e.8f} ⊕ {0b.64} ⊕ {0d.2b} · {09.2b} 09 = 0000 1001 = x3 + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x5 + x3 + x + 1) = x8 + x6 + x5 + x4 + x + 1 = x8 + x6 + x5 + x4 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 0111 0011 1 0001 1011 (XOR) 0110 1000 (68) · {0e.8f} 0e = 0000 1110 = x3 + x2 + x 8f = 1000 1111 = x7 + x3 + x2 + x + 1 = (x3 + x2 + x)(x7 + x3 + x2 + x + 1) = (x10 + x6 + x5 + x4 + x3) + (x9 + x5 + x4 + x3 + x2) + (x8 + x4 + x3 + x2 + x) = x10 + x9 + x8 + x6 + x = x10 + x9 + x8 + x6 + x modulo x8 + x4 + x3 + x1 + 1 = 111 0100 0010 100 0110 11(XOR) 11 0010 1110 10 0011 011 1 0000 1000 1 0001 1011 0001 1011 (1b) · {0b.64} 0b = 0000 1011 = x3 + x + 1 64 = 0110 0100 = x6 + x5 + x2 = (x3 + x + 1)(x6 + x5 + x2) = (x9 + x8 + x5) + (x7 + x6 + x3)

+ (x6 + x5 + x2) = x9 + x8 + x7 + x3 + x2 = x9 + x8 + x7 + x3 + x2 modulo x8 + x4 + x3 + x1 + 1 = 11 1000 1100 10 0011 011(XOR) 1 1011 1010 1 0001 1011 1010 0001 (a1) · {0d.2b} 0d = 0000 1101 = x3 + x2 + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x2 + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x7 + x5 + x3 + x2) + (x5 + x3 + x + 1) = x8 + x7 + x6 + x4 + x2 + x + 1 = x8 + x7 + x6 + x4 + x2 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 1101 0111 1 0001 1011 (XOR) 1100 0100 (c4) C21 = {68} ⊕ {1b} ⊕ {a1} ⊕ {c4} = 16 Berikut ini merupakan perhitungan pencarian nilai untuk C31 : 0e 0b 0d 09 X 2b 09 0e 0b 0d 8f 0d 09 0e 0b 64 0b 0d 09 0e 2b C31 = {0d.2b} ⊕ {09.8f} ⊕ {0e.64} ⊕ {0b.2b} · {0d.2b} 0d = 0000 1101 = x3 + x2 + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x2 + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x7 + x5 + x3 + x2) + (x5 + x3 + x + 1) = x8 + x7 + x6 + x4 + x2 + x + 1 = x8 + x7 + x6 + x4 + x2 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 1101 0111 1 0001 1011 (XOR) 1100 0100 (c4) · {09.8f} 09 = 0000 1001 = x3 + 1 8f = 1000 1111 = x7 + x3 + x2 + x + 1 = (x3 + 1)(x7 + x3 + x2 + x + 1) = (x10 + x6 + x5 + x4 + x3) + (x7 + x3 + x2 + x + 1) = x10 + x7 + x6 + x5 + x4 + x2 + x + 1 = x10 + x7 + x6 + x5 + x4 + x2 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 100 1111 0111 100 0110 11 (XOR) 1001 1011 (9b) · {0e.64} 0e = 0000 1110 = x3 + x2 + x 64 = 0110 0100 = x6 + x5 + x2 = (x3 + x2 + x)(x6 + x5 + x2) = (x9 + x8 + x5) + (x8 + x7 + x4) + (x7 + x6 + x3) = x9 + x6 + x5 + x4 + x3 = x9 + x6 + x5 + x4 + x3 modulo x8 + x4 + x3 + x1 + 1 = 10 0111 1000 10 0011 011 (XOR) 0100 1110 (4e) · {0b.2b} 0b = 0000 1011 = x3 + x + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x6 + x4 + x2 + x) + (x5 + x3 + x + 1) = x8 + x5 + x2 + 1 = x8 + x5 + x2 + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 0010 0101 1 0001 1011 (XOR) 0011 1110 (3e) C31 = {c4} ⊕ {9b} ⊕ {4e} ⊕ {3e} = 2f Berikut ini merupakan perhitungan pencarian nilai untuk C41 : 0e 0b 0d 09 X 2b 09 0e 0b 0d 8f 0d 09 0e 0b 64 0b 0d 09 0e 2b C41 = {0b.2b} ⊕ {0d.8f} ⊕ {09.64} ⊕ {0e.2b} · {0b.2b} 0b = 0000 1011 = x3 + x + 1 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x + 1)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x6 + x4 + x2 + x) + (x5 + x3 + x + 1) = x8 + x5 + x2 + 1 = x8 + x5 + x2 + 1 modulo x8 + x4 + x3 + x1 + 1 = 1 0010 0101 1 0001 1011 (XOR) 0011 1110 (3e) · {0d.8f} 0d = 0000 1101 = x3 + x2 + 1 8f = 1000 1111 = x7 + x3

+ x2 + x + 1 = (x3 + x2 + 1)(x7 + x3 + x2 + x + 1) = (x10 + x6 + x5 + x4 + x3) + (x9 + x5 + x4 + x3 + x2) + (x7 + x3 + x2 + x + 1 = x10 + x9 + x7 + x6 + x + 1 = x10 + x9 + x7 + x6 + x + 1 modulo x8 + x4 + x3 + x1 + 1 = 110 1100 0011 100 0110 11 (XOR) 10 1010 1111 10 0011 0001 1001 0001 (91) · {09.64} 09 = 0000 1001 = x3 + 1 64 = 0110 0100 = x6 + x5 + x2 = (x3 + 1)(x6 + x5 + x2) = (x9 + x8 + x5) + (x6 + x5 + x2) = x9 + x8 + x6 + x2 = x9 + x8 + x6 + x2 modulo x8 + x4 + x3 + x1 + 1 = 11 0100 0100 10 0011 011 (XOR) 1 0111 0010 1 0001 1011 0110 1001 (69) · {0e.2b} 0e = 0000 1110 = x3 + x2 + x 2b = 0010 1011 = x5 + x3 + x + 1 = (x3 + x2 + x)(x5 + x3 + x + 1) = (x8 + x6 + x4 + x3) + (x7 + x5 + x3 + x2) + (x6 + x4 + x2 + x) = x8 + x7 + x5 + x = x8 + x7 + x5 + x modulo x8 + x4 + x3 + x1 + 1 = 1 1010 0010 1 0001 1011 (XOR) 1011 1001 (b9) C41 = {3e} ⊕ {91} ⊕ {69} ⊕ {b9} = 7f Untuk mendapat hasil yang lainnya C12, C22, C32, C42, C13, C23, C33…C44 lakukan perhitungan perkalian polynomial dengan cara seperti diatas. Berikut hasilnya : C12 = {0e.9a} ⊕ {0b.bf} ⊕ {0d.37} ⊕ {09.f7} = {cd} ⊕ {55} ⊕ {48} ⊕ {0e} = de C22 = {09.9a} ⊕ {0e.bf} ⊕ {0b.37} ⊕ {0d.f7} = {26} ⊕ {20} ⊕ {fa} ⊕ {ff} = 03 C32 = {0d.9a} ⊕ {09.bf} ⊕ {0e.37} ⊕ {0b.f7} = {78} ⊕ {30} ⊕ {11} ⊕ {fb} = a2 C42 = {0b.9a} ⊕ {0d.bf} ⊕ {09.37} ⊕ {0e.f7} = {09} ⊕ {fa} ⊕ {94} ⊕ {fd} = 9a C13 = {0e.30} ⊕ {0b.39} ⊕ {0d.6f} ⊕ {09.e4} = {3b} ⊕ {98} ⊕ {9d} ⊕ {85} = bb C23 = {09.30} ⊕ {0e.39} ⊕ {0b.6f} ⊕ {0d.e4} = {ab} ⊕ {45} ⊕ {e4} ⊕ {48} = 32 C33 = {0d.30} ⊕ {09.39} ⊕ {0e.6f} ⊕ {0b.e4} = {6b} ⊕ {ea} ⊕ {2c} ⊕ {56} = fb C43 = {0b.30} ⊕ {0d.39} ⊕ {09.6f} ⊕ {0e.e4} = {cb} ⊕ {0e} ⊕ {3a} ⊕ {0f} = f0 C14 = {0e.aa} ⊕ {0b.50} ⊕ {0d.5b} ⊕ {09.cf} = {f6} ⊕ {46} ⊕ {c2} ⊕ {ed} = 9f C24 = {09.aa} ⊕ {0e.50} ⊕ {0b.5b} ⊕ {0d.cf} = {8d} ⊕ {4d} ⊕ {03} ⊕ {fc} = 3f C34 = {0d.aa} ⊕ {09.50} ⊕ {0e.5b} ⊕ {0b.cf} = {13} ⊕ {e6} ⊕ {2f} ⊕ {68} = b2 C44 = {0b.aa} ⊕ {0d.50} ⊕ {09.5b} ⊕ {0e.cf} = {c2} ⊕ {bd} ⊕ {b5} ⊕ {b6} = 7c Hasil dari proses Inverse MixColumn adalah sebagai berikut : Hasil Inverse MixColumns ad de Bb 9f 16 03 32 3f 2f a2 fb b2 7f 9a f0 7c b. Proses Inverse ShiftRows (Ronde 13) Berikut ini merupakan hasil proses dari Inverse Shiftrows terhadap hasil dari Inverse MixColumns. Hasil Inverse ShiftRows ad de bb 9f 3f 16 03 32 fb b2 2f a2 9a f0 7c 7f c. Proses Inverse SubBytes (Ronde 13) Berikut ini merupakan hasil proses dari Inverse SubBytes terhadap hasil dari Inverse ShiftRows. Hasil

Inverse SubBytes 18 9c Fe 6e 25 ff d5 a1 63 3e 4e 1a 37 17 01 6b Setelah tahapan tahapan tersebut dilakukan, maka mendapatkan hasil dari ronde 13. Berikut hasil dari ronde 13 : Hasil Ronde 13 18 9c Fe 6e 25 ff d5 a1 63 3e 4e 1a 37 17 01 6b Pada pendekripsian blok pertama, akan menghasilkan beberapa nilai dari setiap rondenya, dimana nilai dari ronde tersebut akan menjadi input di ronde selanjutnya. Berikut penulis paparkan hasil nilai dari setiap ronde yang telah melalui proses inverse sebelumnya. Penulis tidak dapat memaparkan setiap proses untuk setiap rondernya dikarenakan mempersingkat penjelasan terhadap tahapan tahapannya. Berikut hasil nilai di setiap rondenya : Hasil Ronde 12 Hasil Ronde 11 Hasil Ronde 10 31 1d 84 df 07 2c 69 7a 1f b3 d3 01 1b de 06 75 27 8f f1 93 62 a5 c8 d2 64 fb 6a d8 44 7d 0c 90 c0 2a 16 19 53 be ef f4 0d 68 83 be ae 76 a0 56 Hasil Ronde 9 Hasil Ronde 8 Hasil Ronde 7 52 bc 76 97 e1 45 da fe 79 31 1e f3 80 20 dd c5 8d bf ea 8a d3 7c 78 22 9d 4f 81 93 b1 df 85 ea 9c d8 b5 41 ee 17 1a e9 17 e7 6a 45 35 0a 4b fc Hasil Ronde 6 Hasil Ronde 5 Hasil Ronde 4 e4 8d ae d4 85 4d 97 59 03 8f a6 64 f5 64 66 e8 3c 3f 7a 83 b2 1e d0 fa 2a 57 d8 e0 e9 ab 90 30 d6 d9 9c be 3b de e6 d3 e1 89 6f 14 7b 52 20 20 Hasil Ronde 3 Hasil Ronde 2 Hasil Ronde 1 35 19 1d 1b da c5 50 33 48 6f 69 73 00 70 e3 04 7f 1d 7f 5c 65 20 73 20 63 e0 14 1e bc 68 ba fc 6c 74 20 53 d0 eb 4d 44 0e bb ba 86 6c 68 69 65 Hasil ronde ke 11 inilah yang menjadi hasil akhir dari pendekripsian blok pertama sekaligus menjadi plaintext pada blok pertama. Berikut hasil plaintext block 1 : 48656c6c6f207468 6973206973205365 4.2.6 Dekripsi Blok Kedua Pada pemrosesan chipertext block kedua, diperlukan penambahan IV diakhir tahapan AddRoundKey, IV yang digunakan ialah hasil dari chippertext block pertama. Seperti gambaran dekripsi aes mode cbc diatas. Untuk RoundKeys yang digunakan sama seperti RoundKeys yang digunakan pada dekripsi chipertext block pertama, tahapan tahapannya pun sama. Untuk mempersingkat penjelasan penulis akan mempersingkat tahapannya dan menuliskan hasil akhirnya saja seperti dibawah ini : Chipertext Block 2 62 3e cb 6d 3d 21 bd 5a fc 23 0d b4 88 5d b7 5a a. Proses AddRoundKey Chipertext block 2 ⊕ RoundKeys 14 62 3e cb 6d 6b 99 88 fb 3d 21 bd 5a 9a 51 91 25 fc 23 0d b4 ac b7 54 1b 88 5d b7 5a 9f b0 c9 1e Hasil 09 a7 43 96 a7 70 2c 7f 50 94 59 af 17 ed 7e 44 b. Proses Inverse ShiftRows Hasil dari proses

AddRoundKeys tadi akan di geser beberapa bit di dalam proses ini proses ini berkebalikan dari proses ShiftRows. Berikut Hasil dari proses Inverse ShiftRows : Hasil Inverse ShiftRows 09 a7 43 96 7f a7 70 2c 59 af 50 94 ed 7e 44 17 c. Proses Inverse SubBytes Hasil Inverse SubBytes 40 89 64 35 6b 89 d0 42 15 1b 6c e7 53 8a 86 87 Hasil Ronde 14 40 89 64 35 6b 89 d0 42 15 1b 6c e7 53 8a 86 87 Hasil dari Inverse SubBytes diatas sekaligus hasil dari hasil ronde ke 14. Selanjutnya adalah tahap dimana proses ini melakukan 13 kali perulangan sesuai dengan proses dekripsi AES-256 (Gambar 2.9 hlm.20). Tahapan selanjutnya ialah putaran selanjutnya mencari nilai dari hasil ronde ke 13 dan seterusnya sampai ke ronde 1. Untuk mempersingkat penjelasan tahapan dikarenakan terlalu panjangnya penjelasannya, maka dari itu penulis hanya mencantumkan hasil ronde dari tiap tahapannya sebagai berikut ini : Hasil Ronde 13 Hasil Ronde 12 Hasil Ronde 11 f7 36 a0 7f 3d 40 34 f4 ef 87 2a d9 81 a5 3b 2b 9f 00 22 7b 7f 85 8b e5 3b 18 4b 43 aa a4 36 e3 27 bd ef 20 0c 64 40 33 e4 b6 b9 d7 c3 2f 29 3a Hasil Ronde 10 Hasil Ronde 9 Hasil Ronde 8 78 c2 6d cf 49 d9 ac 8b b6 75 71 6e a5 4b 92 b9 c1 7c 09 92 c6 97 ea 19 47 e5 cf fe 29 99 4c a5 9b 69 87 e3 f6 3c 0a 6f c1 93 4b 2d 0c 4f e3 8e Hasil Ronde 7 Hasil Ronde 6 Hasil Ronde 5 f7 ef 8b 46 2c fa 84 5e 4a cf 05 69 a1 1c 4b 5a 7d 1f 75 c5 58 ac 3f dc c1 24 42 e3 b6 4c 31 f1 29 46 9c bf 2f 41 23 24 3f 40 f6 a0 fe ed f5 5b Hasil Ronde 4 Hasil Ronde 3 Hasil Ronde 2 6d 72 a1 31 a2 a3 ad d1 a1 ed Fc 8a 66 cc 7a 19 9b 60 ae 61 d9 ca 87 e7 55 cc 8a e5 38 5e 96 be f5 fb 72 f3 da 1d f5 de 04 da 1b 88 36 13 3d c7 Pada ronde terakhir, yaitu ronde ke 1, Hasil Inverse SubBytes ronde ke 1 akan dilakukan operasi XOR dengan RoundKeys 1 dan XOR dengan IV, yaitu chipertext block pertama. Hasil ronde ke 1 inilah yang menjadi hasil akhir dari pendekripsian blok kedua sekaligus menjadi plaintext pada blok kedua dan menjadi akhir dari penjelasan proses pendekripsian Inverse SubBytes 1 ⊕ RoundKeys 0 ⊕ IV 46 e7 c5 0a 36 e4 d2 b0 13 23 7f 9b 0b 40 b7 59 08 4e 68 22 71 48 b6 78 45 7f 31 f6 bc a6 eb 60 9c b0 bf 95 da 8c ea 70 a1 c4 6d 26 0f 2b f5 55 aes. Berikut ini hasil ronde 14 adalah : Hasil Ronde 1 63 20 68 21 72 46 69 03 65 69 65 03 74 63 72 03 Pada rumus hasil yang telah disebutkan sebelumnya diatas bahwa : Maka dari itu, plaintext dari chipertext “13719c0f2348b02b 7fb6bff59b789555 623dfc883e21235d

cbbd0db76d5ab45a" dengan kunci "xyz" ialah 48656c6c6f207468 6973206973205365 6372657420466963 6869657221030303. Apabila di konversikan kedalam huruf alphabeth dari heksadecimal tersebut maka akan didapatkan pesan asli ialah "Hello this is Secret Fichier!" Berikut penulis tampilkan hasil dekripsi text tersebut dengan Secret Fichier, dan apabila di convert akan menghasilkan hexadecimal seperti diatas. Gambar 4.2.6 Hasil Dekripsi 4.3 Perancangan Aplikasi Pada tahap ini aplikasi mulai di rancang dengan permodelan UML (Unified Modelling Language), membuat stuktur yang ada didalam menu dan tampilan antar muka atau yang sering disebut User Interface dengan mempertimbangkan keefisiensian suatu aplikasi yang dibangun oleh peneliti. Rancangan aplikasi ini mencakup ; 1. sistem yang ada didalam aplikasi yaitu tampilan menu utama yang berisikan penginputan suatu file dengan radio button untuk pemilihan opsi proses enkripsi atau dekripsi dan menu proses untuk pemilihan destinasi file apabila telah di proses dan penginputan password atau kunci sebelum terjadinya proses enkripsi dan deksripsi. Diharapkan penulis dengan mengedepankan suatu aplikasi yang efisien tanpa mengurangi suatu kegunaan dari aplikasi tersebut. 2. Pemodelan aplikasi menggunakan Unified Modelling Language (UML), yang terdiri dari 28Use Case Diagram, Activity Diagram dan Sequence Diagram. 3. Perancangan antar muka aplikasi yaitu penggambaran tampilan menu menu yang akan dibuat dan ditampilkan didalam aplikasi yang dibangun. Berikut penjelasan penggambaran bagaimana interaksi antar user dengan sistem yang ada didalam Secret Fichier yang akan digambarkan dengan permodelan UML. 4.3.1 Use Case Use Case Diagram merupakan pengambaran rangkaian interaksi antara actor dengan sistem. Actor mewakili sebagai user yang berinteraksi kepada sistem yang dimodelkan dengan Use Case Diagram didalam diagram tersebut. Berikut pemodelan aplikasi Secret Fichier dengan UML. Gambar 4.3.1 Use Case Diagram Secret Fichier Penjelasan gambar : 1. Pada usecase tersebut, actor atau user membuka aplikasi Secret Fichier maka tampilan awal dari Secret Fichier ialah form dashboard. Lalu user dapat memilih file yang akan di enkripsi atau di dekripsi dengan choose file button dengan catatan file yang akan di enkripsi ialah file tunggal bukan sebuah folder dan file yang akan di dekripsi ialah sebuah

file dengan eksitensi format .sf. 2. Lalu, setelah user memilih file yang akan di process, ada dua buah opsi radio button yaitu encryption process dan decryption process. Dengan catatan file yang akan diproses yaitu file dokumen, gambar, suara dan video. Setelah dipilih proses mana yang akan dilakukan, lalu tombol start akan aktif dan menampilkan form process. 3. Selanjutnya di form process akan di tampilkan button file destination yang akan digunakan user untuk memilih tempat atau folder untuk file yang sudah di enkripsi atau di dekripsi. 4. Setelahnya user harus menginput sebuah password lebih dari 8 (delapan) character untuk menguci serta memproses file tersebut. Setelah file selesai di proses maka aplikasi akan balik ke halaman awal. 4.3.2 Activity Diagram Pada activity diagram, penulis menggambarkan aktifitas suatu alurkerja yang ada didalam sistem yang dibuat oleh penulis. Dalam aplikasi Secret Fichier ini terdapat dua aktifitas didalamnya, yaitu proses enkripsi suatu file dan proses dekripsi suatu file. 4.3.2.1 Activity Diagram Enkripsi File Gambar 4.3.2.1 Activity Diagram Enkripsi File Penjelasan Gambar 4.3.2.1 : 1. Pada awal masuk aplikasi Secret Fichier, user akan ditampilkan form dashboard yang berisikan tampilan penginputan file yang akan diproses yang dinamain choose file. File yang akan di enkripsi hanya suatu file tunggal (bukan folder) yaitu file yang berjenis dokumen, suara, gambar dan video. 2. Lalu pada saat file diinput, tampilan text box akan menunjukan file itu berada, dan user akan memilih opsi encryption button pada radio button yang tersedia. 3. Setelah user mengklik tombol encryption dan tombol start akan aktif, tombol tersebut menghubungkan user ke form selanjutnya yaitu form process. 4. Lalu user akan ditampilkan form process dimana user akan memilih tempat file yang akan ditaruh apabila proses pengenkripsian telah selesai dan penginputan password atau kunci untuk memulai pengenkripsian file tersebut. 5. User akan mengklik button destination dimana user memilih dimana file tersebut akan ditaruh setelah proses selesai. Dalam proses tersebut, user dapat me-rename file tersebut. Mengganti nama file yang ingin dienkrispi berbeda dengan file aslinya. 6. Setelah menentukan letak file setelah proses, user harus menginputkan password atau kunci unik yang tidak mudah ditebak sebanyak 8 (delapan) character sebagai kunci file yang akan dienkripsi, kunci tersebut berguna kembali apabila

user ingin mendekripsikan kembali file yang telah di enkrip. 7. Lalu user mengklik button process dan pengenkripsian file tersebut terjadi. 4.3.2.2 Activity Diagram Dekripsi File Gambar 4.3.2.2 Activity Diagram Dekripsi File Penjelasan Gambar 4.4.2.2 : 1. Pada awal masuk aplikasi Secret Fichier, user akan ditampilkan form dashboard yang berisikan tampilan penginputan file yang akan diproses yang dinamain choose file. File yang akan di dekripsi hanya suatu file tunggal (bukan folder) yaitu file yang berformat .sf. 2. Lalu pada saat file diinput, tampilan text box akan menunjukan file itu berada, dan user akan memilih opsi decryption button pada radio button yang tersedia. 3. Setelah itu user mengklik tombol start, dan user akan ditampilkan form process. Didalam form process tersebut process pengdekripsian terjadi. 4. Didalam form process, user akan mengklik button destination dimana user dapat memilih dimana file tersebut akan ditaruh setelah proses selesai. Dalam proses tersebut, user dapat me-rename file tersebut. Mengganti nama file yang ingin didekrispi berbeda dengan file enkripsinya. 5. Setelah menentukan letak file setelah di proses, user diharuskan menginputkan kembali password atau kunci yang telah dibuat sebelumnya pada proses pengenkripsian file. Setelah mengklik tombol proses button, terdapat decision proses dimana kunci tersebut benar atau tidak. Jika kunci tersebut benar maka proses pendekripsian selesai. 6. Apabila kunci yang di inputkan salah makan user akan diminta kembali memasukan kunci yang benar. 7. Setelah proses selesai user akan di kembalikan ke form dashboard. 4.3.3 Sequence Diagram Sequence Diagram atau diagram sekuen merupakan diagram yang menggambarkan kolaborasi dari objek objek yang berinteraksi didalam usecase dengan mendekripsikan waktu hidup objek tersebut dan massage yang dikirim dan diterima antar objek tersebut. 4.3.3.1 Sequence Diagram Enkripsi File Gambar 4.3.3.1 Sequence Diagram Enkripsi File Penjelasan Gambar : 1. Dalam sequence diagram ini terdapat 1 user atau yang disebur aktor, 2 lifeline yaitu form dashboard dan form process yang ada di secret fichier, dan 10 messages. 2. Pertama kali user membuka aplikasi, tampilan awalnya ialah form dashboard untuk menginput file yang ingin diproses, 3. Lalu user menentukan proses dan klik start button, lalu tampilan form process muncul. 4. Pilih destination file untuk file yang setelah di proses. Dan user

menginputkan password lalu proses enkripsi terjadi. 4.3.3.2 Sequence Diagram Dekripsi File Gambar 4.3.3.2 Sequence Diagram Dekripsi File Penjelasan Gambar : 1. Pada sequence diagram dekripsi file ini terdapat 1 user, 2 lifeline yaitu form dashboard dan form process yang ada di dalam aplikasi secret fichier, dan 10 messages. 2. Pertama kali user membuka aplikasi, tampilan awalnya ialah form dashboard untuk menginput file yang ingin diproses, dalam dekripsi file di aplikasi secret fichier ini, format file harus .sf. 3. Lalu user menentukan proses enkripsi dan klik start button, lalu tampilan form process muncul. 4. Pilih destination file untuk lokasi file setelah di proses lalu user harus menginput kembali password awal pada pengengkripsian file. 5. Pada proses tersebut terdapat autentifikasi password, apabila password benar maka file akan kembali seperti semula, apabila password salah maka dekripsi file tidak berhasil. 4.4 Perancangan Antar Muka Aplikasi (User Interface) Perancangan User Interface untuk aplikasi Secret Fichier adalah tampak seperti 4.4.1 Desain Tampilan Awal Aplikasi (Form Dashboard) Dalam desain tampilan awal di aplikasi Secret Fichier pada awal masuk user akan di tampilkan langsung dashboard ada beberapa tombol didalamnya yang akan terkoneksi pada form process dan form about. Berikut desain tampilan form dashboard : Gambar 4.4.1 Desain Dashboard 4.4.2 Desain Tampilan Menu Proses (Form Process) Dalam desain tampilan berikutnya di aplikasi Secret Fichier, yaitu form process dimana saat user mengklik button continue dan form process muncul. Pada form process tersebut dimana proses dari enkripsi dan dekripsi sebuah file bekerja, di form tersebut juga user harus menginputkan sebuah password untuk memproses file yang ingin di enkripsi atau di dekrispi. Berikut tampilan dari form process Gambar 4.4.2 Desain Form Process 4.4.3 Desain Tampilan Keterangan Aplikasi (Form About) Gambar 4.4.3 Desain Form About 4.5 Fase Implementasi Sebelum program diimplementasikan, maka program harus bebas dari kesalahan. Kesalahan program yang mungkin terjadi dikarenakan kesalahan dalam penulisan program (coding), kesalahan proses atau kesalahan logika. Dalam fase implementasi aplikasi pengamanan data file berbasis windows “Secret Fichier” ini, analisis kebutuhan perangkat pendukung juga sangat diperlukan. Selain itu juga, kebutuhan perangkat lunak pendukung juga harus tersedia

demi kelancaran tahap implementasi program. Dalam implementasi program ini ada beberapa proses yang dilakukan yaitu : 1. Memasukan kode program (coding), tahap ini dilakukan dengan menggunakan aplikasi Visual Studio 2019 .NET Core dengan menggunakan bahasa program C# (C Sharp). 2. Menguji aplikasi dengan mengenkripsi dan mendekripsi jenis file dengan format yang berbeda beda serta melakukan debugging atau perbaikan program jika perlu. 4.6 Implementasi Perangkat Lunak Perangkat lunak yang digunakan penulis dalam membangun aplikasi. pengamanan data file "Secret Fichier" adalah System Operasi Windows 10 64 bit dan Visual Studio 2019 (.NET Core). Aplikasi Secret Fichier merupakan aplikasi enkripsi dan dekripsi sebuah file dengan penerapan AES-256 serta SHA-256 untuk teknik pengenkripsian pada data file tersebut. Keterbatasan aplikasi ini hanya dapat mengenkripsi sebuah file tunggal bukan folder dengan file berjenis dokumen, video, text dan gambar. Instalasi aplikasi Secret Fichier akan dijelakan pada point point berikut ini : · Pada folder aplikasi Secret Fichier, user mengklik file yang bertuliskan .exe atau setup. · Lalu user menginstall aplikasi seperti biasa. Klik next. · Dan jika sudah berhasil terinstall, Secret Fichier sudah bisa dipakai untuk mengenkrip dan mengdekrip sebuah file. · User dapat mendownload resource GitHub aplikasi Secret Fichier pada link berikut : https://github.com/selfviyani/SecretFichier.git 4.7 Implementasi Perangkat Keras 12Perangkat keras yang digunakan penulis untuk membuat aplikasi ini antara lain adalah dengan sebuah laptop dengan spesifikasi mempunyai processor Intel Core i7-10510U dengan RAM 8 GB dan Solid State Drive 1000 GB serta VGA Nvidia MX250 2GB. 4.8 Implementasi Aplikasi 4.8.1 Form Dashboard Berikut tampilan awal di aplikasi Secret Fichier pada awal masuk user akan di tampilkan langsung form dashboard sebagaimana berikut ini : Gambar 4.8.1 Tampilan Form Dashboard 4.8.2 Form Process Berikut tampilan pada saat user mengklik tombol continue, pada form ini ialah form process dimana, user akan menginputkan file yang akan di proses dan password sebagai mana kuncinya. Berikut tampilan form process : Gambar 4.8.2 Tampilan Form Process 4.8.3 Form About Form ini menampilkan deklarasi bahwa aplikasi Secret Fichier benar benar dibuat oleh penulis berdasarkan referensi yang ada. Berikut tampilan dari form about : Gambar 4.8.3 Tampilan

Form About 4.9 Penerapan SHA-256 dan AES-256 Dalam Program Penerapan algoritma Advanced Encryption Standard (AES-256) mode Chiper Block Chaning (CBC) dan Secure Hash Algorithm (SHA-256) terletak pada form process di aplikasi Secret Fichier. Pada form ini proses pengenkripsian dan pendekripsian sebuah file di process, dari penginputan sebuah password yang memiliki panjang character delapan buah hingga pemilihan lokasi file setelah di process. Setelah sebuah file yang telah di input pada form dashboard selanjutnya di process pada form process, pada form inilah algoritma AES-256 mode CBC bekerja untuk memproses file tersebut, dan juga algoritma SHA-256 sebagai algortima fungsi hash dari kunci yang telah di inputkan agar merubah sebuah password yang tidak bisa dibaca. Sebuah data file yang telah dienkripsi akan menjadi sebuah file yang tidak bisa terbaca dengan format .sf pada data file tersebut. Dan apabila user ingin mengembalikan sebuah data yang sudah dienkripsi 57diperlukan proses dekripsi untuk mengembalikan data tersebut. Proses dekripsi file juga terjadi di dalam form process, algoritma AES-256 mode CBC bekerja untuk mengembalika data file yang sudah dienkripsi menjadi data file yang bisa terbaca kembali, dengan syarat format file chipertext tersebut ialah .sf dan user masih mengingat password yang sudah dibuatnya pada proses enkrispi pada file tersebut. Apabila user tidak menginput password yang benar maka algoritma AES-256 mode CBC serta fungsi hash SHA-256 tidak dapat memprosesnya dikarena password yang salah. Berdasarkan pengamanan data file yang dilakukan oleh aplikasi Secret Fichier berbasis windows, dengan sempel data file dari berbagai jenis ekstensi file yaitu file dokumen, gambar, suara serta video akan dilakukan pengamanan data file (enkripsi) sampai menjadi sebuah data file yang tidak terbaca apa informasinya. 4.10 Hasil Pengujian Dalam pengujian aplikasi Secret Fichier, penulis menerapkan pengujian blackbox. Sebagaimana disebutkan blackbox ialah 6metode yang digunakan untuk menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah input diterima dengan benar dan output yang dihasilkan telah sesuai dengan yang diharapkan. Secret Fichier diharapkan dapat menerima input yang diharapkan yaitu dapat mengenkripsi file dengan eksitensi yang berbeda dan mendapatkan output berubah

chipertext yang diharapkan tidak dapat terbaca. Secret Fichier juga diharapkan dapat mengdekripsi sebuah file yang telah di enkripsi sebelumnya dengan kunci yang tepat. Pada pengujian ini dilakukan oleh penulis pada minggu ke 2 bulan Januari 2021. Berikut tabel pengujian Secret Fichier : Tabel 4.10.1 Rancangan Pengujian Kelas Uji Tes Uji Jenis Pengujian Form Dashboard Memilih file Blackbox Menampilkan directory Blackbox Memilih opsi proses Blackbox Mengklik tombol process setelah menentukan. Blackbox Form Process Tampil form process Blackbox Memilih destinasi file Blackbox Menampilkan directory Blackbox Menginput password Blackbox Memproses enkripsi Blackbox Memproses dekripsi Blackbox Tabel 4.10.2 Pengujian Implementasi Kelas Uji Skenario Pengujian Hasil Yang Diharapkan Hasil Pengujian Form Dashboard Dapat memilih file pada button search dan menampilkan directory. Pemilihan file berjalan dengan lancar dan tidak ada kendala. Serta dapat menampilkan directory Berhasil Dapat menampilkan lokasi dari file yang ada di directory pada text box yang tersedia. Lokasi file dapat ditampilkan setelah memilih file pada text box. Berhasil Memilih radio button pada proses file yang akan dijalani, tombol enkripsi dan tombol dekripsi Dapat memilih proses dengan baik dan proses yang dijalani lancar tanpa erorr. Berhasil Form Process Memilih file destination pada tombol search untuk menyimpan hasil dari proses yang telah dijalani. Dapat bejalan dengan lancar tanpa kendala saat memilih destinasi file yang ada di directory yang ingin disimpan. Berhasil Menampilkan directory file dan mengubah nama file yang ingin disimpan. Dapat menampilkan lokasi file yang sesuai dan benar serta dapat me-rename sebuah file. Berhasil Form Process Menginput password delapan atau lebih karakter. Dapat menginput password dengan baik Berhasil Form Process Memproses enkripsi file dengan baik. Dapat mengenkripsi file dengan baik dan menghasilkan file yang tidak terbaca sama sekali Berhasil Form Process Memproses dekripsi file dengan baik. Dapat mengdekripsi file yang telah di enkripsi dengan baik dengan kunci yang tepat. Berhasil Semple data file yang di uji cobakan ialah data file yang berada di Program Studi Infomatika Fakultas Ilmu Komputer Universitas Bhayangkara Jakarta Raya. Berikut penulis jabarkan data file-nya sebagai berikut : 1. Pedoman Akademik UBJ TA

2020-21-rpur.docx (data 1) 2. Master Kurikulum dan Kurikulum Transisi TIF 2019 rpur.xlsx (data 2) 3. Rapat Persiapan Perkuliahan 2020-1.pptx (data 3) 4. 01 ND Daftar Makul dan RPS Pendukung MBKM Update (All).pdf (data 4) 5. Rpsmkwudanmkdu.zip (data 5) 6. Logo ubj.png (data 6) 7. Deklarasi anti narkoba Selviyani.mp4 (data 7) Nama File Ukuran File (bytes) Panjang Password Ukuran File Setelah Dienkripsi (bytes) Presentase Perubahan bytes (%) Waktu Proses Enkripsi (detik) Waktu Proses Dekripsi (detik) data1.docs 14.582.885 12 karakter 14.582.896 ∞ 11.26/det 6.13/det data2.xlsx 113.794 11 karakter 113.808 0.012% 2.19/det 0.56/det data3.pptx 1.625.761 10 karakter 1.625.776 0.001% 10.26/det 4.13/det data4.pdf 761.725 13 karakter 761.728 0.0004% 4.78/det 2.32/det data5.zip 14.346.667 13 karakter 14.346.672 ∞ 11.70/det 5.67/det data6.png 239.800 10 karakter 239.808 0.0033% 2.77/det 1.08/det data7.mp4 101.895.158 15 karakter 101.895.168 ∞ 12.48/det 6.32/det Tabel 4.10.3 Pengujian Enkripsi dan Dekripsi File. Berdasarkan dari tabel pengujian Secret Fichier, dapat disimpulkan semua fungsionalitas sistem dapat berjalan dengan lancar dan baik. Tidak ditemukannya sebuah bugs atau erros yang tidak diharapkan. Dan dapat disimpulkan pula setelah mengalami proses enkripsi, suatu file menghasilkan perbedaan ukuran file dengan awal file sebelum di enkripsi dengan sesudah enkripsi, dikarenakannya padding dan penyesuaian block enkripsi yang mempunyai kelipatan 16 bytes. Pada tabel pengujian diatas, rata rata penambahan byte pada file diatas ialah 8 bytes. Serta dapat disimpulkan jika ukuran file dapat mempengaruhi lamanya proses enkipsi dan dekripsi.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil berdasarkan hasil dari pengujian serta analisis penerapan algoritma Advanced Encryption Standard (AES-256) dengan mode Chiper Block Chaining (CBC) dan Secure Hash Algorithm (SHA-256) terhadap aplikasi yang telah dibuat yaitu aplikasi enkripsi dan dekripsi file berbasis Windows yaitu Secret Fichier. Dapat disimpulkan beberapa point sebagai berikut : a. Untuk Mencegah terjadinya kebocoran data yang bersifat penting dan merugikan beberapa pihak, oleh karena itu data bersifat penting harus di enkripsi terlebih dahulu dengan aplikasi Secret Fichier untuk melindungi data penting pada Univeristas Bhayangkara Jakarta Raya. b. Aplikasi Secret Fichier mampu mengenkripsi file dengan

berbagai ekstensi seperti file dokumen, file suara (voice note/mp3), file video serta file gambar dengan baik dan dapat didekripsikan kembali dengan kunci yang sama pada aplikasi Secret Fichier, aplikasi ini mampu mengamankan data bersifat penting untuk Universitas Bhayangkara Jakarta Raya sebagai penerapan baku dari aplikasi enkripsi berbasis dekstop. c. Algoritma AES mode CBC dan SHA yang diterapkan di aplikasi dapat berjalan dengan baik tanpa kendala di aplikasi Secret Fichier untuk mengenkripsi file dan mendekripsikan nya di dalam sistem operasi Windows 10 64 bit. d. Algoritma AES dan SHA dapat dibilang masih cukup aman didalam pemprosesannya dikarenakan mempunyai kunci yang panjang dan tahapan perhitungan yang cukup rumit di dalamnya. e. Pada proses pengenkripsian file terdapat perbedaan ukuran file asli dengan file yang dienkripsi dikarenkan adanya proses padding didalam proses pengenkripsian tersebut, sehingga menunjukan perbedaan ukuran file asli dengan file yang telah di enkripsi. 5.2 Saran Untuk penelitian lebih lanjut, perlu dipertimbangkan kembali berdasarkan kesimpulan yang telah dipaparkan diatas. berikut untuk 58saran saran untuk penelitian selanjutnya : a. Untuk penelitian selanjutnya, disarankan untuk dapat mengenkripsi file dengan berukuran lebih besar dan dapat dijalan di sistem operasi lainnya (tidak hanyak windows). b. Untuk penelitian selanjutnya, pertimbangkan kembali untuk menggunakan mode operasi yang lainnya, seperti CFB (Chiper FeedBack), OFB (Output FeedBack) atau GCM (Galois Counter Mode). c. Untuk penelitian selanjutnya, disarankan untuk dapat menggunakan seri SHA terbarukan yaitu SHA-3. d. Untuk penelitian selanjutnya, pertimbangkan kembali untuk menggunakan algoritma asimetris. Dimana memiliki dua kunci yang berbeda untuk keamanannya. e. Untuk penelitian selanjutnya, aplikasi Secret Fichier dapat dikembangkan dan diterapkan pada mobile atau menjadi sebuah fitur independent dalam sebuah aplikasi messenger. f. Penerapan Electronic Data Interchange (EDI) bisa 35menjadi salah satu solusi untuk membuat keamanan dalam transaksi bisnis di Internet dan dalam pertukaran file melalui handshakes files. 2

Sources

1	https://studyinformatics.blogspot.com/2012/07/aes-advanced-encryption-standard.html INTERNET 5%
2	https://www.updatenya.com/2012/12/pengertian-dan-macam-macam-gerbang.html INTERNET <1%
3	https://journal.unnes.ac.id/nju/index.php/jte/article/download/18628/9320 INTERNET <1%
4	https://devianggitanasutionabii.blogspot.com/2013/06/macam-macam-algoritma-kriptografi.html INTERNET <1%
5	https://www.slideshare.net/gustitammam/fungsi-hash-algoritma-sha256 INTERNET <1%
6	https://atikamusthafa.wordpress.com/2012/11/29/metode-blackbox-testing/ INTERNET <1%
7	https://repository.nusamandiri.ac.id/index.php/unduh/item/20883/SKRIPSI-LENGKAP.pdf INTERNET <1%
8	https://lingkarantekno.wordpress.com/2014/03/05/pembangkitan-ekspansi-kunci-aes-256-bit/ INTERNET <1%
9	http://download.garuda.ristekdikti.go.id/article.php?article=778235&val=12764&title=Aplikasi%20Pengamanan%20Data%20dengan%20Teknik%20Algoritma%20Kriptografi%20AES%20dan%20Fungsi%20Hash%20SHA-1%20Berbasis%20Desktop INTERNET <1%
1	https://www.academia.edu/21972768/ANALISA_DAN_IMPLEMENTASI_PROSES_KRIPTOGRAFI_ENCRYPTION_DECRYPTION_DENGAN_ALGORITMA_AES_128 INTERNET <1%
1	https://repository.bsi.ac.id/index.php/unduh/item/20880/File-9-Daftar-Simbol.pdf INTERNET <1%
1	https://123dok.com/document/ydmnj0ly-landasan-teori-produk-perancangan-dibuat-keputusan-keputusan-penting.html INTERNET <1%
1	https://vaskoedo.wordpress.com/tag/kriptograpi/ INTERNET <1%
1	https://123dok.com/document/ydkmvelq-bab-ii-landasan-teori.html INTERNET <1%

1 https://jurnal.likmi.ac.id/Jurnal/11_2007/Sistem_Pengkodean_Tacbir_.pdf
INTERNET
<1%

1 http://ilmusisteminfo.com/upload/file_pdf/Algoritma%20Kriptografi%201567701750.pdf
INTERNET
<1%

1 https://e-jurnalpenelitian.blogspot.com/2015/02/jurnal-algoritma-kriptografi-aes.html
INTERNET
<1%

1 https://123dok.com/document/6qmv895q-analisis-jaringan-aplikasi-bank-persero-divisi-tamansari-bandung.html
INTERNET
<1%

1 http://www.materidosen.com/2017/04/use-case-diagram-lengkap-studi-kasus.html
INTERNET
<1%

2 https://journal.amikmahaputra.ac.id/index.php/JIT/article/download/51/42/
INTERNET
<1%

2 https://lib.unnes.ac.id/35717/1/5302414027_Optimized.pdf
INTERNET
<1%

2 https://www.academia.edu/11368231/Implementasi_Kriptografi_Pada_Diary_Berbasis_Mobile_Android_Dengan_Menggunakan_Metode_AES_128_dan_SHA_1
INTERNET
<1%

2 https://www.academia.edu/26267183/Fungsi_Hash_dan_Algoritma_SHA_256
INTERNET
<1%

2 http://repo.darmajaya.ac.id/1168/3/BAB%20II.pdf
INTERNET
<1%

2 https://ejurnal.undana.ac.id/jme/article/download/1378/1127/
INTERNET
<1%

2 https://journal.amikmahaputra.ac.id/index.php/JIT/article/download/52/50/
INTERNET
<1%

2 https://widuri.raharja.info/index.php?title=SI1122469121
INTERNET
<1%

2 https://widuri.raharja.info/index.php?title=SI1414482114
INTERNET
<1%

2 http://lib.unnes.ac.id/35717/1/5302414027_Optimized.pdf
INTERNET
<1%

3 http://e-journals.unmul.ac.id/index.php/SAKTI/article/download/150/pdf
INTERNET
<1%

3 https://repository.bsi.ac.id/index.php/unduh/item/225234/File_9-Daftar-Simbol.pdf
INTERNET
<1%

3 https://skripsi-skripsiun.blogspot.com/2014/09/skripsi-teknologi-informasiimplementasi_73.html
INTERNET
<1%

3 http://agungsr.staff.gunadarma.ac.id/Downloads/files/71839/Enkripsi+dan+Dekripsi.pdf
INTERNET
<1%

3 https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Makalah-UAS/Makalah-UAS-Kripto-2020%20(3).pdf
INTERNET
<1%

3 https://wikishare27.wordpress.com/electronic-data-interchange/
INTERNET
<1%

3 https://hannymaharani23.blogspot.com/2015/05/metode-enkripsi-modern.html
INTERNET
<1%

3 https://123dok.com/document/zxn4w5oq-penerapan-algoritma-kriptografi-asimetris-untuk-keamanan-data-oracle.html
INTERNET
<1%

3 https://smksikelompok.wordpress.com/2016/01/09/konsep-kriptografi/
INTERNET
<1%

3 https://gatewaytugas.blogspot.com/2016/06/metode-penyandian-pesan-dan-menjaga.html
INTERNET
<1%

4 https://ris.uksw.edu/download/makalah/kode/M01401
INTERNET
<1%

4 http://eprints.ums.ac.id/18197/2/BAB_I.pdf
INTERNET
<1%

4 https://www.academia.edu/40645123/KEAMANAN_INFORMASI_SHA_256_UIN_SUSKA_RIAU_Oleh_Kelompok_6
INTERNET
<1%

4 https://123dok.com/document/6zkx5epy-aplikasi-indonesia-korea-menggunakan-algoritma-binary-berbasis-android.html
INTERNET
<1%

4 https://repository.dinamika.ac.id/id/eprint/1561/7/BAB_V.pdf
INTERNET
<1%

4 http://repository.unpas.ac.id/30278/4/BAB%20III.pdf
INTERNET
<1%

4 https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Makalah-UTS/Makalah1-Kripto-022-2020.pdf
INTERNET
<1%

4 https://id.scribd.com/doc/29412263/Fungsi-Hash
INTERNET
<1%

4 https://www.academia.edu/37999239/KONSEP_KRIPTOGRAFI_DAN_FUNGSI_HASH
INTERNET
<1%

4 https://eprints.akakom.ac.id/3505/10/10_135410037_BAB%20I.pdf
INTERNET
<1%

5 https://elibrary.unikom.ac.id/id/eprint/2746/6/11.%2010115477_WINDA%20PRATIWI%20SEPTIANTI_BAB%201.pdf
INTERNET
<1%

5 https://repository.dinamika.ac.id/id/eprint/1080/7/Bab_IV.pdf
INTERNET
<1%

5 http://lecturer.ukdw.ac.id/katalog/index.php?dir=&file=Prodi%20Teknik%20Informatika.xls
INTERNET
<1%

5 https://syamsurizal.staff.unja.ac.id/2020/06/01/layanan-bimbingan-tugas-akhir/
INTERNET
<1%

5 https://123dok.com/document/wq263jz1-sistem-tanda-tangan-digital-menggunakan-algoritma-kriptografi-publik.html
INTERNET
<1%

5 http://eprints.walisongo.ac.id/436/4/083811002_Bab3.pdf
INTERNET
<1%

5 https://123dok.com/document/nq7xxky6-keamanan-pesan-dengan-metode-significant-advanced-encryption-standard.html
INTERNET
<1%

5 http://e-journals.unmul.ac.id/index.php/JIM/article/download/129/pdf
INTERNET
<1%

5 https://www.researchgate.net/publication/324663274_HUBUNGAN_COPING_STRES_TERHADAP_KUALITAS_HIDUP_PENDERITA_SKIZOFRENIA_PADA_MASA_REMISI_SIMPTOM
INTERNET
<1%

BIODATA MAHASISWA

Nama Lengkap	: Selviyani
NPM	: 201710225097
Fakultas	: Ilmu Komputer
Program Studi	: Informatika
Tempat, Tgl. Lahir	: Bekasi, 30 Agustus 1999
Email	: selviyani30@gmail.com
No. HP/WA	: 0857-8280-3694
Agama	: Islam
Jenis Kelamin	: Perempuan
Kewarganegaraan	: Indonesia
Alamat Rumah	: Vila Gading Harapan 5 blok H9 No. 33 RT 01/12, ds. Satria Mekar, Kec. Tambun Utara, Kab. Bekasi, Jawa Barat. 17510
Pendidikan Format	: SDN Harapan Baru 3 tahun (2005 - 2011)
	SMPN 2 Tambun Utara tahun (2011 - 2014)
	SMAN 1 Tambun Utara tahun (2014 - 2017)

Bekasi, 19 Juli 2021

Selviyani

_____

UNIVERSITAS BHAYANGKARA JAKARTA RAYA

Jl. Harsono RM No.67, Ragunan, Pasar Minggu, Jakarta Selatan, 12550
Telepon : (021) 27808121, 27808882
Jl. Perjuangan, Bekasi Utara
Telepon : (021) 88955882, Fax : (021) 88955871
Web: *www.ubharajaya.ac.id* **Email:** *fasilkom@ubharajaya.ac.id*

KARTU BIMBINGAN SKRIPSI
PROGRAM STUDI INFORMATIKA
UNIVERSITAS BHAYANGKARA JAKARTA RAYA

Nama : Selviyani

NPM : 201710225097

Pembimbing 1 : Ahmad Fathurrozi, SE., MMSI

Judul : Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File.

PEMBIMBING MATERI

NO	TANGGAL BIMBINGAN	DESKRIPSI BIMBINGAN	PARAF
1	27 – 10 – 2020	Pengarahan bab 1	
2	03 – 11 – 2020	Pengarahan bab 2	
3	09 – 11 – 2020	Pengarahan bab 3	
4	19 – 11 – 2020	Bimbingan keseluruhan proposal skripsi	
5	26 – 11 - 2020	Revisi bab 3 setelah sempro	
6	11 – 01 - 2021	Penyerahan dan pengarahan bab 4	
7	13 – 01 - 2021	Penyerahan revisi bab 4 dan arahan bab 5	
8	18 – 01 - 2021	Penyerahan bab 5 serta point untuk di ppt	

Pembimbing 1

Ahmad Fathurrozi, SE., MMSI
NIDN. 0327117402

UNIVERSITAS BHAYANGKARA JAKARTA RAYA

Jl. Harsono RM No.67, Ragunan, Pasar Minggu, Jakarta Selatan, 12550
Telepon : (021) 27808121, 27808882
Jl. Perjuangan, Bekasi Utara
Telepon : (021) 88955882, Fax : (021) 88955871
Web: www.ubharajaya.ac.id Email: fasilkom@ubharajaya.ac.id

KARTU BIMBINGAN SKRIPSI
PROGRAM STUDI INFORMATIKA
UNIVERSITAS BHAYANGKARA JAKARTA RAYA

Nama : Selviyani

NPM : 201710225097

Pembimbing 2 : Rakhmat Purnomo, S.Pd., S.Kom., M.Kom.

Judul : Penerapan Algoritma *Advanced Encryption Standard* (AES-256) Dengan Mode CBC dan *Secure Hash Algorithm* (SHA-256) Untuk Pengamanan Data File.

PEMBIMBING TEKNIS DAN METODOLOGI

NO	TANGGAL BIMBINGAN	DESKRIPSI BIMBINGAN	PARAF
1	27 – 10 – 2020	Pengarahan judul serta bab 1	
2	03 – 11 – 2020	Pengarahan bab 2	
3	09 – 11 – 2020	Pengarahan bab 3	
4	26 – 11 - 2020	Revisi bab 3 metodologi setelah sempro	
5	11 – 01 - 2021	Penyerahan dan pengarahan bab 4	
6	13 – 01 - 2021	Penyerahan revisi bab 4 dan arahan bab 5	
7	15 – 01 - 2021	Penyerahan keseluruhan dan revisi pengujian data	
8	18 – 01 - 2021	Pengarahan Daftar Pustaka	

Pembimbing 2

Rakhmat Purnomo, S.Pd., S.Kom., M.Kom.
NIDN. 0322108201