



**PERANCANGAN APLIKASI KEAMANAN CITRA DIGITAL
MENGUNAKAN ALGORITMA AES-128 BIT BERBASIS
JARINGAN *CLIENT - SERVER***

Disusun dan Diajukan Untuk Memenuhi Persyaratan Ujian Akhir Memperoleh
Gelar Sarjana Komputer Pada Fakultas Sains dan Teknologi
Universitas Pembangunan Panca Budi
Medan

SKRIPSI

OLEH

**NAMA : ANGGRAINI
N.P.M : 1514370340
PROGRAM STUDI : SISTEM KOMPUTER**

**FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS PEMBANGUNAN PANCA BUDI
MEDAN
2019**

ABSTRAK

ANGGRAINI

PERANCANGAN APLIKASI KEAMANAN CITRA DIGITAL MENGUNAKAN ALGORITMA AES 128-BIT BERBASIS JARINGAN *CLIENT – SERVER*

2019

Kriptografi merupakan salah satu metode mengamankan data yang dapat digunakan untuk menjaga kerahasiaan data, keaslian dan serta keaslian pengirim. Metode ini bertujuan agar informasi yang bersifat rahasia yang dikirim melalui telekomunikasi umum seperti LAN atau Internet. kriptografi biasanya dalam bentuk enkripsi dan dekripsi. Untuk mengamankan keaslian citra digital biasanya menggunakan algoritma AES (*Advanced Encryption Standard*). Dalam hal ini, penulis berkeinginan mengangkat topik enkripsi dan dekripsi menjadi sebuah penulisan ilmiah skripsi dengan menggunakan visual studio yang berkembang saat ini. Diharapkan dengan adanya aplikasi ini, mahasiswa serta dosen dapat melakukan uji coba enkripsi menggunakan algoritma AES (*Advanced Encryption Standard*).

Kata Kunci: AES (*Advanced Encryption Standard*), Kriptografi.

DAFTAR ISI

	Halaman
KATA PENGANTAR	i
DAFTAR ISI	iii
DAFTAR GAMBAR	vi
DAFTAR TABEL	x
DAFTAR LAMPIRAN	xi
BAB I PENDAHULUAN	
1.1 Latar Belakang Masalah	1
1.2 Perumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	4
BAB II LANDASAN TEORI	
2.1 Defenisi Perancangan.....	6
2.2 Aplikasi	6
2.3 Kriptografi.....	7
2.3.1 Serangan Terhadap Kriptografi.....	8
2.3.2 Pengelompokkan Algoritma Kriptografi.....	8

2.4	Algoritma AES.....	10
2.4.1	Algoritma Enkripsi AES.....	11
2.4.2	Algoritma Dekripsi AES.....	15
2.4.3	Kelebihan dan Kelemahan Algoritma AES.....	18
2.5	Jaringan	19
2.5.1	Jenis – Jenis Jaringan Komputer	20
2.6	Client - Server	23
2.7	Citra Digital.....	24
2.8	Flowchart.....	25
2.9	UML (<i>Unified Modeling Language</i>)	29
2.9.1	Activity Diagram.....	31
2.9.2	Use Case Diagram.....	32
2.10	Bahasa Pemrograman Visual Studio 2010	33

BAB III METODE PENELITIAN

3.1	Tahapan Penelitian.....	39
3.2	Metode Pengumpulan Data.....	40
3.3	Analisa Sistem Sedang Berjalan.....	40
3.3.1	Kelemahan Sistem yang berjalan	45
3.3.2	Analisa Kebutuhan Siste.....	45
3.4	Rancangan Penelitian.....	46

3.4.1 Flowchart Sistem.....	47
3.4.2 Use Case Diagram.....	53
3.4.3 Activity Diagram.....	54
3.4.4 Perancangan Interface.....	48

BAB IV HASIL DAN PEMBAHASAN

4.1 Kebutuhan Sistem	65
4.1.1 Kebutuhan Perangkat Keras	65
4.1.2 Kebutuhan Perangkat Lunak.....	66
4.1.3 Proses Koneksi.....	65
4.2 Tampilan Program Aplikasi.....	70
4.2.1 Tampilan Input Program Aplikasi	70
4.2.2 Tampilan Output Program Aplikasi	76
4.3 Hasil Pengujian Program Aplikasi	127

BAB V PENUTUP

4.3 Simpulan	131
4.3 Saran.....	131

DAFTAR PUSTAKA

BIOGRAFI PENULIS

LAMPIRAN - LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Citra digital yang bersifat rahasia, sangat rentan terhadap penyadapan oleh pihak yang tidak bertanggung jawab serta penyimpanan ke dalam media *stroge* yang rawan terhadap akses orang - orang yang tidak memiliki hak. Pendistribusian citra digital melalui jaringan internet juga dapat memberikan kerugian kepada pihak yang memiliki otoritas apabila disadap oleh pihak *attacker*. Hal ini dikarenakan citra digital masih berupa citra yang dapat dikenali dan dibaca bentuk objeknya. Untuk itu diperlukanya sebuah teknik pengamanan yang dapat menjamin kerahasiaan citra digital sampai kepada pihak yang berhak menerima. Pengamanan citra digital dapat dilakukan dengan menerapkan teknik enkripsi kriptografi. Kriptografi merupakan teknik yang dapat merubah suatu pesan objek menjadi pesan yang tidak dapat dikenali dan dibaca.

Salah satu algoritma yang dapat mengamankan citra digital adalah algoritma AES (*Advance Encryption Standard*). Algoritma AES merupakan jenis algoritma kriptografi yang memiliki sifat simetri karena untuk proses enkripsi dan dekripsi hanya menggunakan kunci yang sama serta untuk *input* dan *output* nya hanya berupa blok dengan jumlah bit tertentu (Rahmawati & Rahardjo, 2016). Algoritma ini melakukan perhitungan sebanyak 10 putaran ronde, dimana setiap ronde untuk proses enkripsi dan dekripsinya memiliki 4 proses yaitu transformasi substitusi *byte* (*SubBytes*), kemudian pergeseran baris (*ShiftRows*), melakukan pencampuran kolom (*MixColumns*), dan kemudian melakukan penambahan kunci (*AddRoundKey*).

Proses enkripsi dan dekripsi citra digital menggunakan algoritma AES 128-bit dilakukan dengan merubah nilai *Red Green Blue* (RGB) pada setiap *pixel* dari citra digital *plainimage*. Hasil yang didapatkan adalah berupa citra digital *cipherimage* dengan nilai *pixel* yang sudah jauh berbeda dari nilai *pixel* awal (*plainimage*). Hal ini akan membuat citra digital tidak dapat dikenali dan dibaca objeknya dengan pandangan indra mata manusia apabila terjadi penyadapan oleh *attacker* (Winafil, Sinurat, & Zebua, 2018).

Penelitian ini akan menggunakan metode AES 128-bit untuk enkripsi dan dekripsi citra digital pada aplikasi berbasis *client - server*. Pembuatan aplikasi perangkat lunak enkripsi dan dekripsi berbasis *client - server* bertujuan untuk *transfer* data digital berupa *plainimage* yang sudah dienkripsi menggunakan algoritma AES 128-bit dari satu *client* ke *client* yang lainnya. *Client* akan berinteraksi menggunakan kabel LAN (*Local Are Network*) untuk menghubungkan satu komputer dengan komputer lainnya, dan menggunakan settingan *TCP/IP* sebagai identifikasi alamat komputer.

Berdasarkan uraian latar belakang diatas, maka penulis mengambil judul dalam penelitian ini adalah **“PERANCANGAN APLIKASI KEAMANAN CITRA DIGITAL MENGGUNAKAN ALGORITMA AES 128-BIT BERBASIS JARINGAN *CLIENT - SERVER*”**.

1.2 Perumusan Masalah

Pada kesempatan ini penulis ingin memberikan suatu rumusan dalam menganalisa penentuan lokasi. Beberapa permasalahan yang mendasar untuk melakukan perancangan dan implementasi sistem ini adalah :

1. Bagaimana mengamankan citra digital dengan teknik kriptografi ?
2. Bagaimana mengekstraksi informasi dari citra digital sebelum dienkripsi?
3. Bagaimana menerapkan algoritma AES dalam mengamankan citra digital yang sudah diekstraksi ?
4. Bagaimana merancang dan membangun sistem berbasis *client - server* yang dapat menjaga kerahasiaan informasi yang terkandung pada citra digital menggunakan algoritma AES ?

1.3 Batasan Masalah

Berikut ini beberapa batasan masalah yang akan dibahas dalam penelitian ini yaitu sebagai berikut :

1. Citra digital yang akan dienkripsi berupa gambar (*image*) dengan format *jpg*, *png*, dan *bmp*.
2. Ekstraksi citra digital dengan menentukan nilai warna RGB dari citra digital yang akan dienkripsi.
3. Algoritma kriptografi AES yang digunakan untuk mengenkripsi dan mendekripsi citra digital adalah algoritma AES 128-bit.
4. *Software* yang digunakan dalam merancang dan membangun aplikasi adalah *software* bahasa pemrograman *Microsoft Visual Studio 2010*.

5. Pengiriman citra dilakukan berbasis kabel dan nirkabel pada satu lingkup jaringan yang sama dengan hanya menentukan alamat ip *address client* (penerima).
6. Penentuan alamat ip *address* pengiriman dilakukan antara interaksi *user* dengan *user*.

1.4 Tujuan Penelitian

Berikut ini beberapa tujuan penelitian yang akan dibahas dalam penelitian ini yaitu sebagai berikut :

1. Untuk menjaga keaslian dan keterjaminan sebuah citra digital dari ancaman yang ada.
2. Untuk mengetahui bagaimana cara mengekstraksi citra digital berdasarkan warna RGB (*Red Green Blue*).
3. Untuk mengetahui bagaimana cara kerja enkripsi dan dekripsi citra digital menggunakan algoritma AES 128-bit.
4. Untuk mengetahui bagaimana merancang aplikasi kriptografi algoritma AES 128-bit menggunakan aplikasi *Microsoft Visual Studio 2010*.

1.5 Manfaat Penelitian

Berikut ini beberapa manfaat penelitian yang akan dibahas dalam penelitian ini yaitu sebagai berikut :

1. Mengurangi terjadinya manipulasi pada informasi yang terkandung pada citra digital.

2. Memberikan informasi bagaimana mengekstraksi citra digital sebelum dienkripsi menggunakan algoritma AES 128-bit.
3. Memberikan informasi bagaimana cara kerja dari algoritma AES 128-bit dalam mengenkripsi dan mendekripsi citra digital.
4. Menambah pemahaman bagaimana memanfaatkan aplikasi *Microsoft Visual Studio 2010* dalam merancang dan membangun aplikasi kriptografi untuk enkripsi dan dekripsi citra digital.

BAB II

LANDASAN TEORI

2.1 Defenisi Perancangan

Muhammad Arif (2016) mengungkapkan : “Perancangan dapat diistilahkan sebagai sebuah gambaran rencana umum suatu kegiatan rancangan proyek dan aktivitas-aktivitas khusus yaitu teknik atau metode-metode dalam merancang sesuatu.”

2.2 Aplikasi

Aplikasi merupakan perangkat lunak komputer yang memanfaatkan kinerja suatu komputer untuk membatu pengguna menyelesaikan suatu tugas yang dimiliki (Deslianti, & Muttaqin, 2016).

Aplikasi yang dirancang dipergunakan untuk praktisi khusus, klasifikasi luas ini dapat dibagi menjadi dua jenis yaitu :

1. Aplikasi spesialis, yaitu program yang dirancang untuk menyelesaikan tugas tertentu dengan menggabungkan dokumentasi yang ada.
2. Aplikasi paket, yaitu suatu program yang menggabungkan suatu dokumentasi untuk jenis masalah tertentu yang nantinya akan dirancang.

Beberapa aplikasi yang tergabung menjadi suatu paket dapat disebut sebagai *suite* aplikasi (*application suite*). Misalnya adalah *Microsoft Office* dan *OpenOffice.org* yang tergabung kedalam aplikasi pengolah kata, lembar kerja, serta

beberapa aplikasi lainnya. Aplikasi paket biasanya memiliki antar muka pengguna dan terdapat kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi yang dirancang.

2.3 Kriptografi

Miftakuil Amin (2016) menjabarkan kriptografi berasal dari bahasa negara Yunani, *kripto* dan *graphia*. Istilah *kripto* berarti *secret* (rahasia) dan istilah *graphia* berarti *writing* (tulisan). Kriptografi merupakan ilmu yang mempelajari tentang teknik matematika yang sangat berhubungan dengan aspek keamanan informasi, yaitu kerahasiaan data, integritas data, autentikasi data, serta penyangkalan data. Namun tidak semua aspek bisa terselesaikan oleh kriptografi.

Kriptografi dapat dibagi menjadi dua konsep utama, yaitu enkripsi dan dekripsi. Enkripsi merupakan proses pengiriman data/informasi yang sudah diubah menjadi bentuk yang hampir tidak dikenali sebagai data awal yang menggunakan algoritma tertentu biasa disebut *ciphertext*. Sedangkan dekripsi merupakan proses pengembalian data yang sudah diubah bentuknya menjadi data awal/asli biasa disebut *plaintext* (Ilyas & Widodo, 2014).

Untuk proses enkripsi dan dekripsi dibutuhkan suatu pengamanan yang akan menjamin data agar tetap utuh, pengamanan tersebut dinamakan *key*. Manfaat *key* adalah sebagai kunci yang gunanya untuk membuka atau mengawali tiap proses. Pada penelitian ini kunci yang digunakan adalah algoritma kunci simetri yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsi.

2.3.1 Serangan Terhadap Kriptografi

Proses enkripsi rentan terhadap serangan. Setelah pesan dienkripsi bukan berarti tidak ada pihak yang ingin menyadap hasil *cipher* yang sudah terenkripsi, dan ilmu ini disebut dengan kriptanalisis (Prabowo, 2013). Adapun jenis serangan terhadap pesan yang sudah dienkripsi, yaitu :

1. *Ciphertext only attack* merupakan penyerangan yang hanya mendapatkan pesan bersandi.
2. *Known Plaintext Attack* merupakan penyerangan yang sudah mendapatkan sandi dan pesan asli.
3. *Chosen Plaintext Attack* merupakan penyerangan yang hanya membaca sandi diperoleh dari *ciphertext* untuk *plaintext* agar mendapatkan informasi yang mengurangi keamanan skema enkripsi.

2.3.2 Pengelompokan Algoritma Kriptografi

Noura Aleisa (2015) mengungkapkan bahwa berdasarkan jenis kunci yang dipergunakan untuk proses enkripsi dan dekripsi, algoritma kriptografi dibagi menjadi tiga macam, yaitu :

1. Algoritma Kriptografi Simetrik

Kriptografi simetrik atau tunggal salah satu metode enkripsi tertua dan biasanya menggeser huruf teks dengan nomor tertentu. Algoritma ini hanya menyediakan kunci tunggal untuk proses enkripsi dan dekripsi yang akan menjadi angka, kata, atau huruf acak. Dan siapapun yang memiliki kunci tersebut dapat melakukan pendekripsian *ciphertext*. Namun kesulitan dari kunci ini harus memastikan

penerima sudah menerima kunci. Karena jika kuncinya hilang dan didapatkan oleh pihak ketiga maka data yang dienkripsi tidak dapat dibaca. Contoh dari algoritma ini adalah *Data Encryption Standard (DES)*, *Triple Data Encryption Standard (3DES)*, *International Data Encryption Algorithm (IDEA)*, *Advanced Encryption Standard (AES)*, dan sebagainya. Dalam hal ini, algoritma AES akan dibahas lebih lanjut.

2. Algoritma Kriptografi Asimetrik

Kriptografi asimetrik dikenal sebagai enkripsi kunci publik, yang lebih aman karena perlu menggunakan dua kunci yaitu kunci publik dan kunci pribadi yang digunakan untuk mengenkripsi dan mendekripsi teks. Kunci publik dapat diketahui oleh siapapun, dan penerima akan memberikan kuncinya kepada pengirim dan pengirim akan menggunakannya untuk mengenkripsi teks yang akan dikirim. Penerima akan menerima *ciphertext* dan mendekripsi menggunakan kunci pribadinya. Namun kunci pribadi tidak pernah didistribusikan, itulah sebabnya ancaman dari pihak ketiga sangat berkurang karena tanpa kunci pribadi teks tidak dapat didekripsi. Contoh dari algoritma ini adalah *Digital Signature Algoritma (DSA)*, *Rivest Shamit - Adleman (RSA)*, *Diffie - hellman (DH)*, dan sebagainya.

3. Fungsi Hash

Fungsi hash sering disebut dengan fungsi hash satu arah (*one - way function*), *message digest*, *fingerprint*, fungsi kompresi dan *message authentication code (MAC)*. Ini berupa fungsi matematika yang *input* nya panjang variabel dan diubah ke dalam urutan biner dengan panjang semula. Fungsi ini biasanya

digunakan untuk membuat sidik jari dari suatu pesan. Yang merupakan suatu tanda bahwa pesan yang benar - benar berasal dari orang yang diinginkan.

2.4 Algoritma AES

Algoritma AES (*Advanced Encryption Standard*) salah satu algoritma *block cipher* dan bersifat simetri yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Tahun 2001, NIST (*National Institute of Standard and Technology*) mengeluarkan algoritma AES untuk menggantikan algoritma DES (*Data Encryption Standard*). Algoritma AES merupakan algoritma yang untuk proses enkripsi dan dekripsinya memiliki panjang kunci bervariasi, yaitu 128 bit, 192 bit, dan 256 bit (Prameshwari & Sastra, 2018). Perbedaannya dapat dilihat dari panjang kunci yang berpengaruh terhadap jumlah *round* (perputaran) yang dapat dilihat pada tabel berikut :

Tabel 2.1 Tabel urutan data algoritma AES

Tipe	Panjang Kunci	Panjang Blok	Jumlah Putaran
AES 128-bit	4	4	10
AES 192-bit	6	4	12
AES 256-bit	8	4	14

Sumber : Prameshwari & Sastra (2018)

Proses di dalam algoritma AES (*Advanced Encryption Standard*) merupakan transformasi terhadap *state* secara berulang dalam beberapa ronde, yaitu :

1. *Plaintext* adalah *array* yang berukuran 16 byte, data masukan.
2. *Ciphertext* adalah *array* yang berukuran 16 byte, yang berisi hasil enkripsi.

3. *Key* adalah *array* berukuran 16 byte, berisi kunci *chipering* (chiper key).

2.4.1 Algoritma Enkripsi AES

Secara garis besar, enkripsi algoritma AES 128-bit adalah sebagai berikut (Adi Widarma, 2016) :

1. *AddRoundKey*

Dilakukan XOR untuk mendapatkan hasil dari *state* awal (*plaintext*) dengan *chiper key*.

2. Putaran sebanyak N_r-1 kali. Proses yang dilakukan pada setiap putaran adalah:

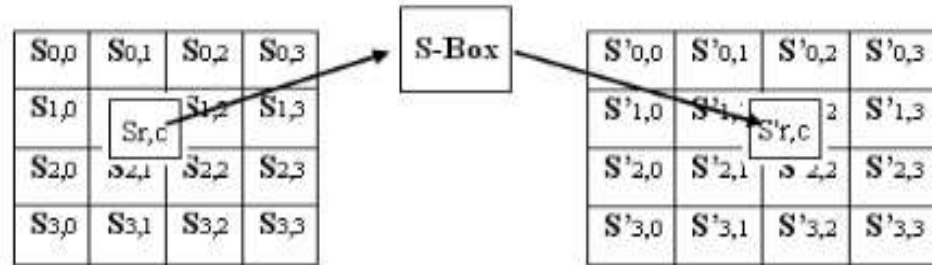
- a. *SubBytes*

Proses operasi yang akan melakukan substitusi tidak linier secara mandiri pada setiap byte dengan menggunakan S-Box yang terdiri dari 16 x 16 baris dan kolom dengan masing-masing berukuran 1 byte.

		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

Gambar 2.1 Tabel S-Box

Sumber : Asriyanik (2017)



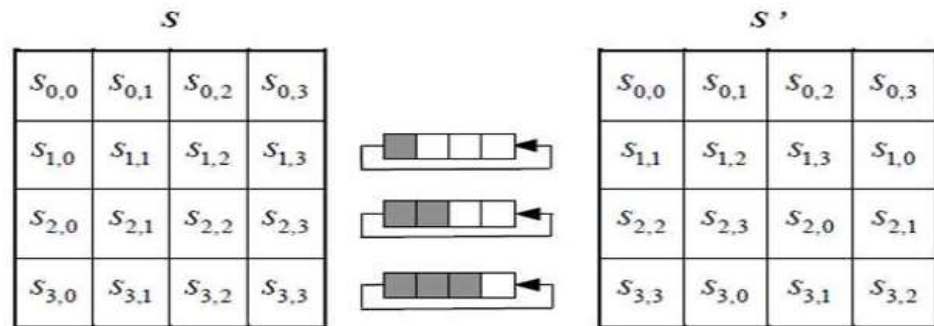
Gambar 2.2 Proses *SubBytes*

Sumber : Rahmawati & Rahardjo (2016)

Dalam pensubstitusinya adalah setiap *byte* pada *array state*, misalkan $S[r,c]=xy$, yang mana dalam hal ini xy adalah digit heksadesimal dari nilai $S[r,c]$, maka nilai substitusinya, dinyatakan dengan $S'[r,c]$, merupakan elemen dalam S-Box yaitu perpotongan baris x dan kolom y . Misalnya $S[0,0]=19$, maka $S'[0,0]=d4$.

b. *ShiftRows*

Suatu proses yang akan dilakukan *shift* atau pergeseran pada setiap elemen blok/tabel yang akan dilakukan per barisnya, yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte*, baris ketiga dilakukan pergeseran 2 *byte*, dan baris keempat dilakukan pergeseran 3 *byte*. Pergeseran tersebut dilakukan dalam sebuah blok dengan pergeseran tiap elemen ke kiri tergantung berapa *byte* tergesernya, tiap pergeseran 1 *byte* berarti bergeser ke kiri sebanyak satu kali.

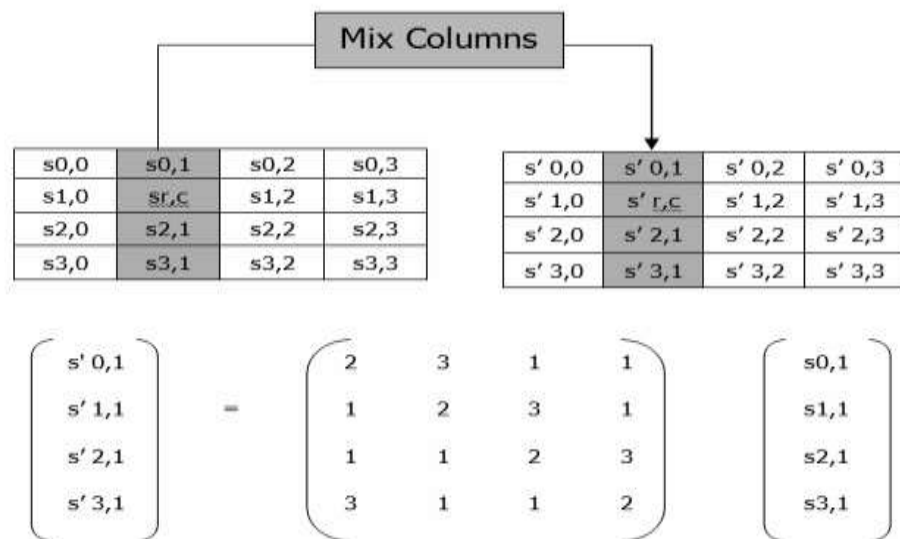


Gambar 2.3 Proses *ShiftRows*

Sumber : Rahmawati & Rahardjo (2016)

c. *MixColumn*

Tahapan yang dilakukan untuk mengalikan tiap elemen dari *blockcipher* dengan matriks yang menggunakan dot *product* lalu perkalian keduanya dimasukkan ke dalam sebuah *blockcipher* baru.



Gambar 2.4 Proses *MixColumn*

Sumber : Rahmawati & Rahardjo (2016)

d. *AddRoundKey*

Tahap ini, *subkey* digabungkan dengan *state*. Proses penggabungannya dilakukan menggunakan operasi XOR untuk setiap *byte* dari *subkey* dengan *byte* yang bersangkutan dari *state*. Setiap tahapan *subkey* akan dibangkitkan dari kunci utama dengan menggunakan proses *key schedule*. Setiap *subkey* berukuran sama dengan *state* yang bersangkutan, dan hasil dari operasi XOR akan disimpan di *array state*.

3. *Final round*

Proses untuk putaran terakhir yang meliputi tiga proses operasi, yaitu :

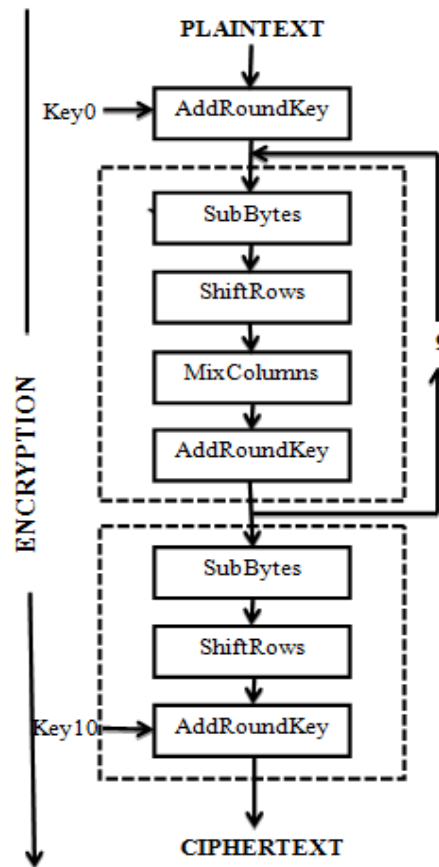
a. *SubBytes*

b. *ShiftRows*

c. *AddRoundKey*

4. *Key Expansion*

Ekspansi *cipher key* digunakan untuk membentuk *roundkey* yang akan digunakan pada langkah-langkah enkripsi dan dekripsi. Ekspansi kunci ini memiliki tahapan khusus yang dikenal sebagai *Rijndael's key schedule*. Iterasi tahapan *AddRoundKey* pada algoritma AES diulang sebanyak sebelas kali. Oleh karenanya, terdapat sepuluh kali *roundkey* yang dibutuhkan dalam satu kali enkripsi *state*. Melalui *key schedule* tersebut, *key* yang menjadi parameter masukan akan diekspansi menjadi beberapa *round*.



Gambar 2.5 Blok Diagram Enkripsi AES

Sumber : Mahajan & Sachdeva (2013)

2.4.2 Algoritma Dekripsi AES

Secara garis besar, dekripsi algoritma AES 128-bit adalah sebagai berikut (Widarma, 2016) :

1. *AddRoundKey*

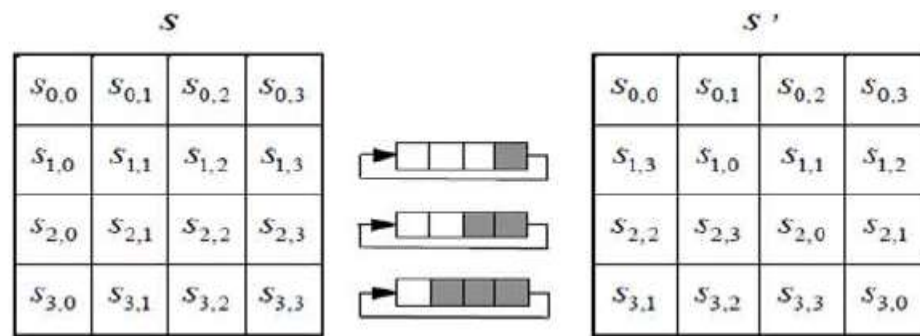
Merupakan proses melakukan XOR antara *stat* awal (*cipherteks*) dengan *cipher key*. Tahap ini disebut juga *initialround*.

2. Putaran sebanyak $Nr-1$ kali. Proses yang dilakukan pada setiap putaran, yaitu:

a. *InvShiftRow*

InvShiftRows adalah transformasi *byte* yang berkebalikan dengan

transformasi *ShiftRows*. Pada setiap transformasi *InvShiftRows*, akan dilakukan pergeseran *bit* ke kanan sedangkan pada tiap *ShiftRows* dilakukan pergeseran *bit* ke kiri.



Gambar 2.6 Proses *InvShiftRows*

Sumber : Rahmawati & Rahardjo (2016)

b. *InvSubBytes*

InvSubByte merupakan transformasi *bytes* yang berkebalikan dengan transformasi *SubBytes*. Proses *InvSubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan table *Inverse S-Box*.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

Gambar 2.7 Tabel Invers S-Box

Sumber : Rahmawati & Rahardjo (2016)

c. *AddRoundKey*

Proses pada tahap ini sama dengan proses enkripsi pada algoritma AES.

d. *InvMixColumn*

Proses yang melakukan setiap kolom dalam *state* dikalikan dengan matriks perkalian dalam AES.

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

Gambar 2.8 Perkalian Matriks *InvMixColumns*

Sumber : Rahmawati & Rahardjo (2016)

3. *Finalround*

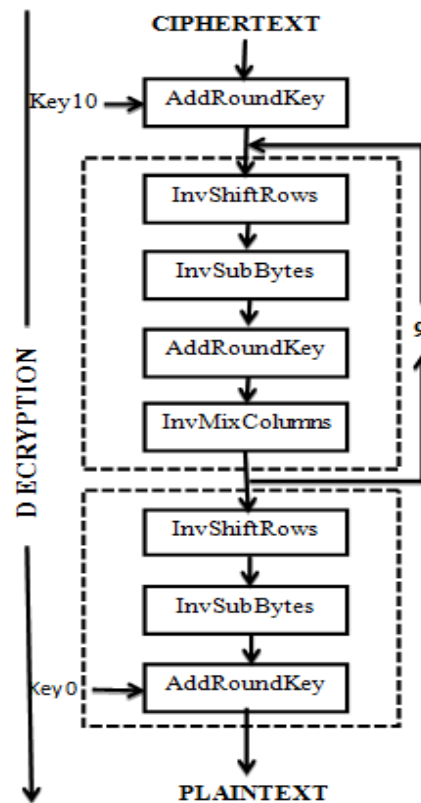
Proses untuk putaran terakhir yang meliputi tiga proses operasi, yaitu :

a. *InvShiftRow*

b. *InvSubByte*

c. *AddRoundKey*

Blok diagram algoritma dekripsi AES-128 bit akan dijelaskan setiap iterasi tahapan *rounds* dari algoritma enkripsi dan dekripsi AES-128 bit.



Gambar 2.9 Blok Diagram Dekripsi AES

Sumber : Mahajan & Sachdeva (2013)

2.4.3 Kelebihan dan Kelemahan Algoritma AES

Asriyanik (2017) menuliskan pada penelitiannya bahwa berdasarkan analisis dari beberapa referensi, algoritma AES (*Advanced Encryption Standard*) memiliki kelebihan maupun kekurangan, diantaranya :

1. Kelebihan Algoritma AES
 - a. Dapat dilihat dari segi jenis kunci yang simetri maka kecepatan operasi (komputasi) lebih tinggi bila dibandingkan dengan algoritma asimetrik sehingga dapat digunakan pada sistem *realtime* seperti GSM.

- b. AES mempunyai panjang kunci paling sedikit 128 bit, maka AES tahan terhadap serangan *exhaustive key search*, serta panjang kunci 128-bit, maka terdapat $2^{128} \approx 3,4 \times 10^{38}$ kemungkinan kunci. Jika digunakan sebuah mesin dengan semilyar prosesor paralel, masing-masing dapat menghitung sebuah kunci setiap satu pico detik, maka akan dibutuhkan waktu $10_{(10)}$ tahun untuk mencoba seluruh kemungkinan kunci.
 - c. Kekuatan AES terletak pada karakteristik sifat dari medan GF(28) di mana untuk setiap bilangan prima selalu terdapat sebuah medan tunggal terbatas yang unik sehingga seluruh representasi dari GF(28) bersifat *isomorphic* dan pemilihan polinomial *biner* berderajat 8.
2. Kelemahan Algoritma AES
- a. Jenis kunci yang simetris, sering terjadi kesulitan dalam manajemen kunci. Hal ini terjadi karena untuk setiap pengiriman dan penerimaan data dengan pengguna yang berbeda dibutuhkan kunci yang berbeda pula.
 - b. Masih dari segi jenis kunci bersifat simetris yang dapat menyebabkan kunci mudah bocor meskipun dalam waktu yang lama karena untuk proses pengiriman dan penerimaan data harus memiliki kunci yang sama.

2.5 Jaringan

Jaringan komputer merupakan gabungan beberapa peripheral yang terbagi menjadi beberapa komputer, printer, LAN card, dan peralatan lainnya yang mampu berintegrasi satu sama lain (Permana, Darmawiguna, & Kesiman, 2014).

Ada beberapa manfaat yang dapat diperoleh dari terhubungannya komputer ke jaringan, antara lain :

1. *Sharing resource*
2. Media Komunikasi
3. Integrasi Data
4. Pengembangan dan Pemeliharaan
5. Keamanan Data
6. Sumber Daya lebih Efisien dan Informasi Terkini

Jaringan komputer diistilahkan sebagai “interkoneksi” antara dua komputer *autonomous* atau lebih yang dapat terhubung menggunakan media transmisi kabel atau tanpa kabel (*wireless*). *Autonomous* dapat diistilahkan sebagai sebuah komputer yang apabila tidak melakukan kontrol pada komputer lain dengan akses penuh, maka dapat membuat komputer lain, *restart*, *shut down*, kehilangan *file* atau kerusakan sistem. Menurut *networking* defenisi *autonomous* berarti jaringan yang bersifat independent dengan manajemen sistem sendiri, juga memiliki topologi, *hardware* dan *software* sendiri. Misalnya internet yang memiliki *autonomous* yang sangat besar (Wongkar, Sinsuw, & Najoran, 2015).

2.5.1 Jenis - Jenis Jaringan Komputer

Jaringan Komputer memiliki beberapa jenis jaringan yang berbeda (Wongkar, Sinsuw, & Najoran, 2015), antara lain :

1. PAN (*Personel Area Network*)

Jaringan Komputer PAN merupakan hubungan antara dua atau lebih komputer yang berada pada jarak tidak terlalu jauh. Jaraknya berkisar 4 sampai 6 meter saja. Contohnya seperti menghubungkan jaringan hp dengan komputer.

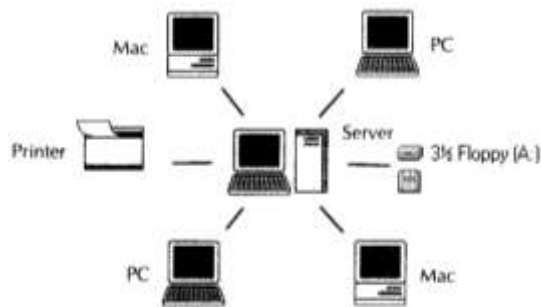


Gambar 2.10 Jaringan Personal Area Network

Sumber : Wongkar, Sinsuw, & Najoan (2015)

2. LAN (*Lokal Area Network*)

Jaringan Komputer LAN merupakan jaringan yang biasa ditemukan di area lokal seperti warnet, kampus, sekolah, atau perkantoran. Jaringan ini membutuhkan hubungan atau koneksi antara dua komputer atau lebih dalam suatu ruangan.

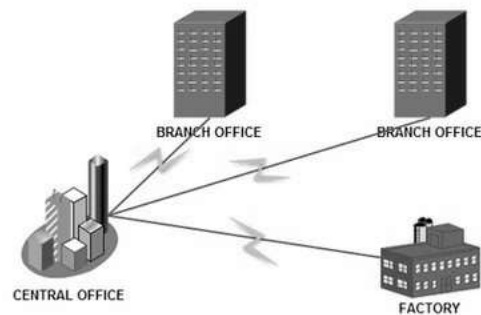


Gambar 2.11 Jaringan Lokal Area Network

Sumber : Wongkar, Sinsuw, & Najoan (2015)

3. MAN (*Metropolitan Area Network*)

Jaringan Komputer MAN merupakan jaringan yang berada dalam suatu kota yang bisa mentransfer data dengan kecepatan tinggi. Jaringan ini dapat menghubungkan suatu lokasi seperti sekolah, kampus, perkantoran dan pemerintahan. Namun faktanya jaringan MAN gabungan dari jaringan komputer LAN. Jaringan MAN mencakup jarak 10 hingga 50 kilo meter.

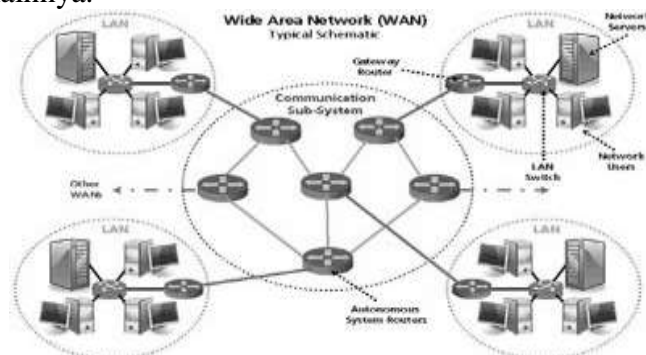


Gambar 2.12 Jaringan Metropolitan Area Network

Sumber : Wongkar, Sinsuw, & Najoa (2015)

4. WAN (*Wide Area Network*)

Jaringan Komputer WAN merupakan jaringan yang mencakup area yang cukup luas. Jaringan ini dapat menghubungkan suatu wilayah atau negara sekaligus dengan negara lainnya.

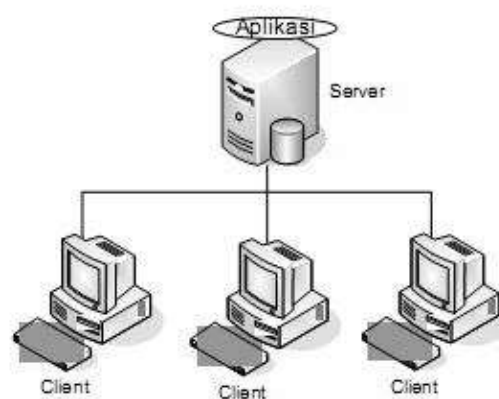


Gambar 2.13 Jaringan Wide Area Network

Sumber : Wongkar, Sinsuw, & Najoa (2015)

2.6 Client - Server

Client merupakan alat untuk meminta atau menerima layanan. Sedangkan Server adalah alat untuk memberikan atau mengirim layanan. Arsitektur ini disebut dengan sistem jaringan *client server* dan biasa digunakan pada seluruh aplikasi jaringan komputer (Varianto & Badrul, 2015).



Gambar 2.14 Jaringan Client-Server

Sumber : Mulyana & Ridwan (2017)

Pada umumnya, jaringan komputer bisa berfungsi sebagai *client* maupun *server*. Namun ada juga jaringan yang memiliki komputer khusus yang didedikasikan sebagai *server* sedangkan komputer lain sebagai *client*. Tetapi ada juga yang tidak memiliki komputer yang khusus sebagai *server* saja. Karena semua berdasarkan fungsinya, maka ada dua jenis jaringan komputer, antara lain :

1. Client Server

Jaringan komputer ini terdapat satu atau beberapa komputer server maupun dapat berfungsi sebagai client dan bisa diubah-ubah melalui *software* jaringan

pada protokolnya. Komputer client berfungsi sebagai sarana untuk mengakses data pada komputer server sedangkan komputer server untuk menyediakan informasi yang dibutuhkan oleh komputer client.

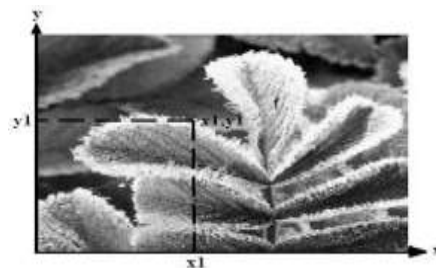
2. Peer to peer

Jaringan komputer ini tidak memiliki client maupun server. Karena semua komputer dapat berperan sebagai client sekaligus server untuk sarana pengiriman dan penerimaan informasi.

2.7 Citra Digital

Citra merupakan gambar yang memiliki bidang dua dimensi. Citra digital berarti citra yang dapat diolah oleh komputer. Pengolahan citra merupakan disiplin ilmu yang dapat diimplementasikan ke dalam suatu program aplikasi sehingga citra yang diolah dapat dirubah tingkat kualitasnya dengan cara merubah bentuk resolusi, ketajaman gambar, pencahayaan serta efek - efek lainnya (Setiawan & Firdaus, 2014).

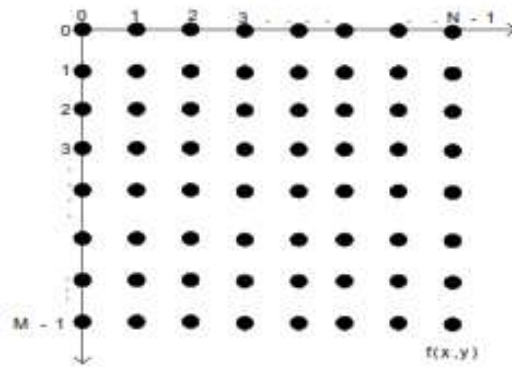
Citra digital dapat diperoleh dengan cara mengukur warna pada sebuah citra pada titik - titik di dalam citra dan mempersentasikan ke dalam bentuk digital atau angka bilangan bulat.



Gambar 2.15 Ilustrasi Citra Digital

Sumber : <https://www.temukanpengertian.com/2013/08/pengertian-citra-digital.html>

Darma Putra (2010), menuliskan citra digital memiliki fungsi $f(x,y)$ berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai (x,y) dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut merupakan citra digital.



Gambar 2.16 Posisi Koordinat Citra Digital

Sumber : Darma Putra (2010)

Format file sebuah citra harus dapat menyatukan kualitas citra, ukuran file, dan kompatibilitas dengan berbagai aplikasi. Ada beberapa jenis format file citra yang digunakan untuk menyimpan citra ke dalam sebuah file. Masing-masing format memiliki karakteristik, contohnya *Bitmap (.bmp)*, *Tagged Image Format (.tif, .tiff)*, *Portable Network Graphics (.png)*, *JPEG (.jpg)*.

2.8 Flowchart

Flowchart adalah urutan dari langkah kerja setiap proses yang digambarkan menggunakan simbol - simbol dan disusun secara sistematis (Eka Iswandy, 2015).

Edhy sutanta (2005) menuliskan bahwa flowchart disusun dengan simbol. Simbol ini digunakan sebagai alat bantu untuk menggambarkan proses didalam program. Simbol-simbol tersebut dapat dibagi menjadi tiga kelompok, antara lain :

1. Simbol Penghubung / Alur

Simbol ini berfungsi untuk menghubungkan antara simbol yang satu dengan simbol yang lain.

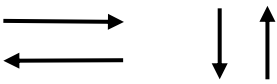

2. Simbol Proses

Simbol yang menunjukkan jenis operasi pengolahan data dalam suatu proses atau prosedur.

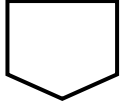
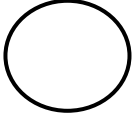
3. Simbol Input dan Output

Simbol yang menunjukkan jenis peralatan yang digunakan sebagai *input* atau *output*.

Tabel 2.2 Simbol - simbol Penghubung / Alur



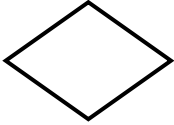


Simbol	Nama	Keterangan
	<i>Arus/Flow</i>	Simbol <i>Arus/Flow</i> untuk menyatakan jalannya arus suatu proses
	<i>Communication Link</i>	Simbol <i>Communication Link</i> untuk menyatakan bahwa adanya transaksi suatu data/informasi dari suatu lokasi ke lokasi lainnya

Tabel Lanjutan 2.2 Simbol - simbol Penghubung / Alur

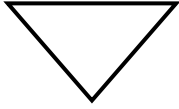

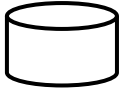

Simbol	Nama	Keterangan
	<i>Offline connector</i>	Simbol <i>Offline connector</i> untuk menyatakan sambungan dari satu proses dalam halaman/lembar yang berbeda
	<i>Connector</i>	Simbol <i>Connector</i> untuk menyatakan sambungan dari satu proses ke proses lainnyadalam halaman/lambar yang sama

Sumber : Edhy Sutanta (2005)

Tabel 2.3 Simbol - simbol Proses


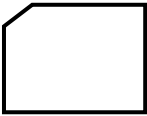

Simbol	Nama	Keterangan
	Proses	Simbol yang menunjukkan pengolahan yang dilakukan oleh computer
	<i>Manual</i>	Simbol <i>Manual</i> Untuk menunjukkan suatu tindakan (Proses) yang tidak
	<i>Decision/Logika</i>	Simbol <i>Decision/Logika</i> Untuk menunjukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban Ya/Tidak
	<i>pdefined Proses</i>	Simbol <i>pdefined Proses</i> Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal
	<i>Terminal</i>	Simbol <i>Terminal</i> Untuk menyatakan permulaan dan akhir suatu program

Tabel Lanjutan 2.3 Simbol - simbol Proses




Simbol	Nama	Keterangan
	<i>Off-Line Storage</i>	Simbol <i>Off-Line Storage</i> Untuk menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu
	<i>Manual Input</i>	Simbol <i>Manual Input</i> Untuk memasukkan data secara manual dengan menggunakan online keyboard
	<i>(Disk Storage)</i>	Data penyimpan (Disk Storage)
	<i>Keying Operating</i>	Simbol <i>Keying Operating</i> Untuk menyatakan semua jenis operasi terhadap proses yang menggunakan suatu mesin pada sebuah keyboard

Sumber : Edhy Sutanta (2005)

Tabel 2.4 Simbol - simbol Input dan Output

Simbol	Nama	Keterangan
	<i>Input-Output</i>	Simbol <i>Input-Output</i> Berfungsi sebagai proses input dan output tanpa tergantung dengan jenis peralatannya.
	<i>Punched Card</i>	Simbol <i>Punched Card</i> Berfungsi untuk menyatakan input yang berasal dari kartu atau output ditulis ke kartu.
	<i>Magnetic-Tape Unit</i>	Simbol <i>Magnetic-Tape Unit</i> Berfungsi untuk menyatakan input yang berasal dari pita magnetic atau output disimpan ke pita magnetic.

Tabel Lanjutan 2.4 Simbol - simbol Input dan Output

Simbol	Nama	Keterangan
	<i>Display</i>	Simbol Display Untuk menyatakan peralatan output yang digunakan berupa layer (Video, Komputer)
	<i>Disk Storage</i>	Simbol <i>Disk Storage</i> Berfungsi untuk menyatakan input yang berasal dari disk atau output disimpan ke disk.
	<i>Dokument</i>	Simbol <i>Dokument</i> Berfungsi untuk mencetak laporan ke Printer.

Sumber : Edhy Sutanta (2005)

2.9 UML (*Unified Modeling Language*)

UML merupakan bahasa pemodelan visual yang bermanfaat untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan rancangan pada sistem perangkat lunak. Pemodelan tersebut dapat memberikan gambaran terhadap sistem yang akan dibangun baik dari sisi struktural ataupun fungsional. UML (*Unified Modeling language*) dapat diterapkan pada semua model pengembangan, meningkatkan siklus sistem, dan berbagai domain aplikasi. Dalam UML terdapat konsep semantik, notasi, dan panduan tiap diagram. UML juga memiliki bagian yang statis, dinamis, ruang lingkup, dan organisasional. UML ini bertujuan untuk menyatukan tiap teknik pemodelan yang berorientasi objek menjadi terstandarisasi (Ibnu Akil, 2016).

Ada beberapa kategori UML yang luas, antara lain :

1. Structure Diagram

Menunjukkan struktur statis pada sistem dan bagian dari abstraksi serta level implementasi yang berbeda dalam bagian - bagian tersebut yang saling berelasi satu sama lain. Structure Diagram tidak menggunakan konsep hubungan waktu, juga tidak menunjukkan detail-detail dari tingkah laku yang dinamis. Namun Structure Diagram ini menunjukkan relasi tingkah laku dari *classifiers* yang ditunjukkan dalam Structure Diagram.

- a. Class Diagram
- b. Object Diagram
- c. Package Diagram
- d. Model Diagram
- e. Composite Structure Diagram, meliputi :
 - 1) Internal Structure Diagram
 - 2) Collaboration Diagram
- f. Component Diagram
- g. Deployment Diagram
- h. Profile Diagram

2. Behavior Diagram

Menunjukkan tingkah laku yang bersifat dinamis dari objek-objek berdasarkan sistem yang dapat dijelaskan sebagai urutan perubahan - perubahan dalam sistem untuk kurun waktu yang panjang.






- a. Use Case Diagram
 - b. Activity Diagram
 - c. State Machine Diagram
 - d. Interaction Diagram, meliputi :
 - 1) Sequence Diagram
 - 2) Communication Diagram
 - 3) Timing Diagram
 - 4) Interaction Overview Diagram
3. Unified Process

Bagian pekerjaan yang melalui proses UP, meliputi *collaboration*, *context*, dan *interaction*. *Collaboration* termasuk ke dalam elemen - elemen dari proyek, *context* termasuk ke dalam proses dari framework, dan *interaction* termasuk kedalam eksekusi dari proyek.

2.9.1 Activity Diagram

Activity Diagram merupakan *state* diagram khusus dimana sebagian besar *state* adalah *action* dan sebagian besar transisi ditrigget oleh *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behavior* internal sebuah sistem (interaksi antar sun sistem), melainkan lebih menggambarkan proses dan jalur aktivitas dari level secara umum (Sukamto & Shalahuddin, 2013).

Tabel 2.5 Simbol - simbol *Activity Diagram*

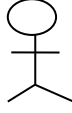

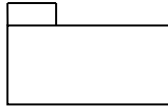
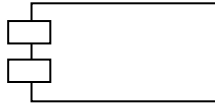
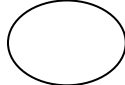
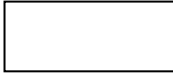

Simbol	Nama	Keterangan
	<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antar berinteraksi satu sama lain.
	<i>Action</i>	<i>State</i> dari system yang mencerminkan eksekusi dari suatu aksi
	<i>Activity final node</i>	Bagaimana objek dibentuk dan dihancurkan
	<i>Fork node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran
	<i>Initial node</i>	Bagaiman objek dibentuk atau diawali

Sumber : Sukamto & Shalahuddin (2013)

2.9.2 Use Case Diagram

Use Case Diagram merupakan suatu model untuk menggambarkan kelakuan (*behavior*) sistem yang di rancang untuk menghasilkan suatu nilai kepada *actor*. Tujuan *Use Case Diagram* ini untuk mengetahui fungsi apa saja yang berada di dalam sistem dan siapa saja yang dapat menggunakan fungsi-fungsi tersebut.

Tabel 2.6 Simbol - simbol *Use Case Diagram*

Simbol	Nama	Fungsi
	<i>Actor</i>	Menggambarkan semua objek di luar sistem (bukan hanya pengguna sistem atau perangkat lunak) yang berinteraksi dengan sistem yang dikembangkan
	<i>Use Case</i>	Mengembangkan fungsional yang dimiliki sistem
	Subsistem (<i>Subsystem</i>)	Menggambarkan konsep dasar pemodelan sistem
	Komponen (<i>component</i>)	Menggambarkan bagian-bagian fisik sistem atau perangkat lunak yang dikembangkan
	Antarmuka (<i>Interface</i>)	Menggambarkan antarmuka pengiriman pesan (<i>message</i>) antar perangkat lunak
	Simpul (<i>node</i>)	Menggambarkan sumberdaya komputasional yang digunakan oleh sistem
	Kelas (<i>class</i>)	Menggambarkan konsep dasar pemodelan sistem

Sumber : Sukamto & Shalahuddin (2013)

2.10 Bahasa Pemrograman Visual Studio 2010

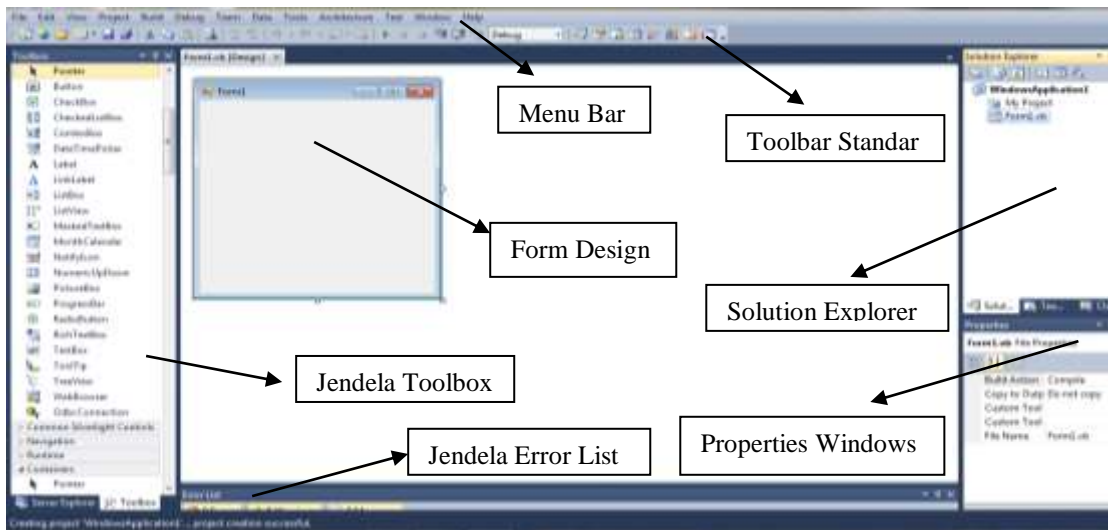
Visual Studio 2010 adalah perangkat lunak (software) yang dapat digunakan untuk pengembangan dari berbagai aplikasi yang memiliki macam-macam tipe antara lain aplikasi desktop (Windows Form, CommandLine (Console), Aplikasi

Web, dan Windows Mobile (Pocket PC). Visual Studio 2010 mempunyai lebih dari satu kompiler antara lain Visual Basic, Visual C#, Visual C++, Visual InterDev, Visual J++, Visual F#, dan Visual Source Safe. Juga memiliki SDK (Software Development Kit), dan Dokumentasi Tutorial (MSDN Library). Semua itu sudah menjadi satu paket dalam platform .Net Framework 4.0 atau versi yang lebih tinggi Rolly Yesputra (2017).

Visual Studio ini dapat bermanfaat untuk membuat suatu aplikasi yang berbasis desktop yaitu platform windows, namun juga dapat dijalankan dalam bentuk Microsoft Intermediate Language di atas .Net Framework. Selain itu Visual Studio juga dapat dijalankan di atas Windows Mobile yang berjalan di atas .Net Compact Framework. Visual studio 2010 memiliki beberapa tipe, antara lain :

1. Visual Studio 2010 Express Edition, dapat digunakan secara gratis tanpa memberikan royalti kepada Microsoft Inc
2. Visual Studio 2010 Standard Edition
3. Visual Studio 2010 Professional Edition
4. Visual Studio 2010 Ultimate Edition

Untuk membangun Aplikasi Keamanan Citra Digital, penulis menggunakan Visual Studio 2010 Ultimate Edition. Adapun tampilan dasar pada Visual Studio 2010 Ultimate Edition dapat dilihat pada gambar dibawah ini :

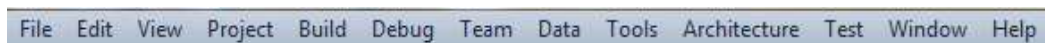


Gambar 2.17 IDE Visual Studio 2010 Ultimate Edition

Adapun penjelasan dari beberapa item yang sudah ditandai pada gambar di atas antara lain :

1. Menu Bar

Menu bar adalah suatu baris yang memiliki 11 menu utama, yang masing-masing memiliki sub menu dan perintah lengkap dengan shortcut key.



Gambar 2.18 Menu Bar Visual Studio 2010 Ultimate Edition

2. Toolbar Standar

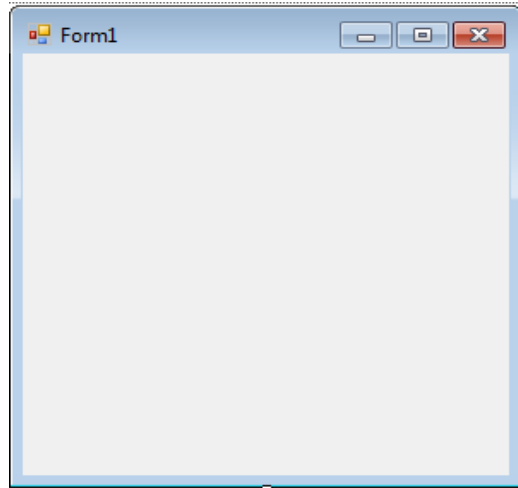
Toolbar standar adalah suatu baris menu yang memiliki fungsi sama pada setiap tool standar pada umumnya. Misalnya fungsi untuk menyimpan, meng-copy, menambah project baru, mengatur tampilan dan masih banyak lagi.



Gambar 2.19 Toolbar Standar Visual Studio 2010 Ultimate Edition

3. Form Design

Form design adalah suatu lembar form yang berguna untuk merancang tampilan aplikasi secara visual dengan menempatkan kontrol-kontrol yang diperlukan.



Gambar 2.20 Form Design Visual Studio 2010 Ultimate Edition

4. Toolbox

Toolbox adalah suatu jendela yang berguna untuk menampung komponen-komponen standard.



Gambar 2.21 Toolbox Visual Studio 2010 Ultimate Edition

5. Solution Explorer

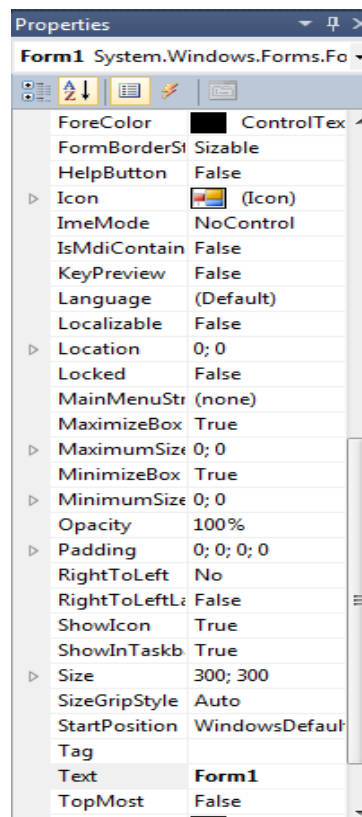
Solution explorer adalah suatu jendela yang berguna untuk menampilkan objek yang akan digunakan untuk membuat aplikasi, misalnya form, class, dan objek lainnya.



Gambar 2.22 Solution Explorer Visual Studio 2010 Ultimate Edition

6. Properties Windows

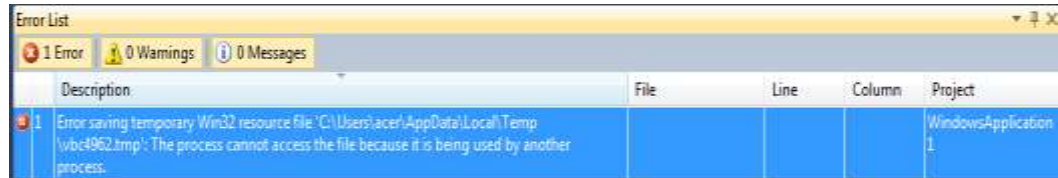
Properties windows adalah suatu jendela yang berguna untuk mengatur nilai properties dari masing-masing komponen yang akan digunakan.



Gambar 2.23 Properties Windows Studio 2010 Ultimate Edition

7. Error List

Error list adalah suatu jendela yang berguna untuk menampilkan setiap kesalahan dari pembuatan kode program suatu aplikasi.

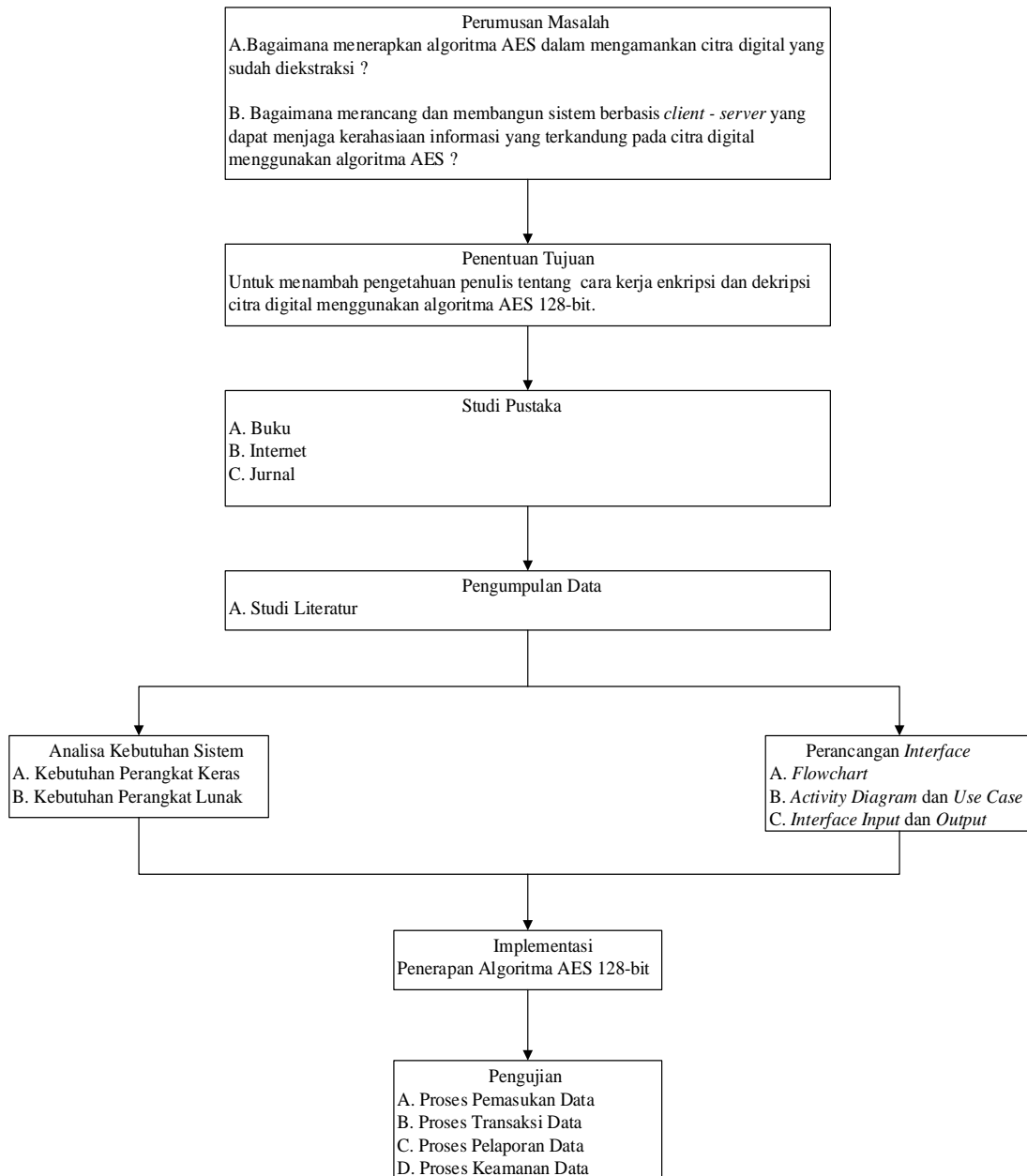


Gambar 2.24 Error List Studio 2010 Ultimate Edition

BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian



Gambar 3.1 Tahapan Penelitian

3.2 Metode Pengumpulan Data

a. Studi Literatur

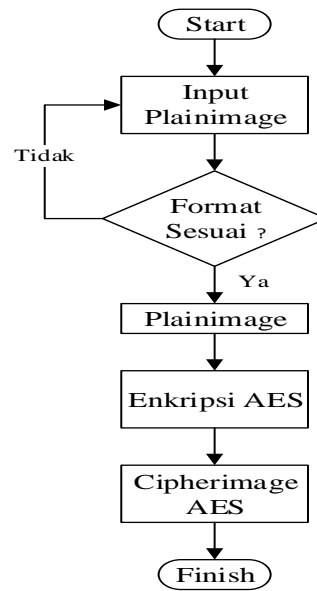
Merupakan tahap pengumpulan data dengan cara mengumpulkan literatur, jurnal, *paper*, dan buku-buku yang berkaitan dengan judul penelitian, serta mencari informasi dari berbagai sumber di internet untuk mengetahui perkembangan terbaru dari data yang diambil sebagai bahan dalam pembuatan tugas akhir.

3.3 Analisa Sistem Sedang Berjalan

Citra digital sangat rentang terhadap penyadapan maupun pencurian oleh pihak yang tidak bertanggung jawab untuk keuntungan kelompok atau pribadi. Apalagi di masa saat ini, di mana citra digital dapat didistribusikan bebas memulai jaringan internet berbasis *chatting*. Citra digital bersifat rahasia yang belum disandikan jika berada ditangan yang salah, maka dengan sangat mudah dianalisa karena citra tersebut masih berupa data yang asli sehingga dapat merugikan salah satu pihak pengirim. Hal tersebut dapat diatasi dengan teknik kriptografi.

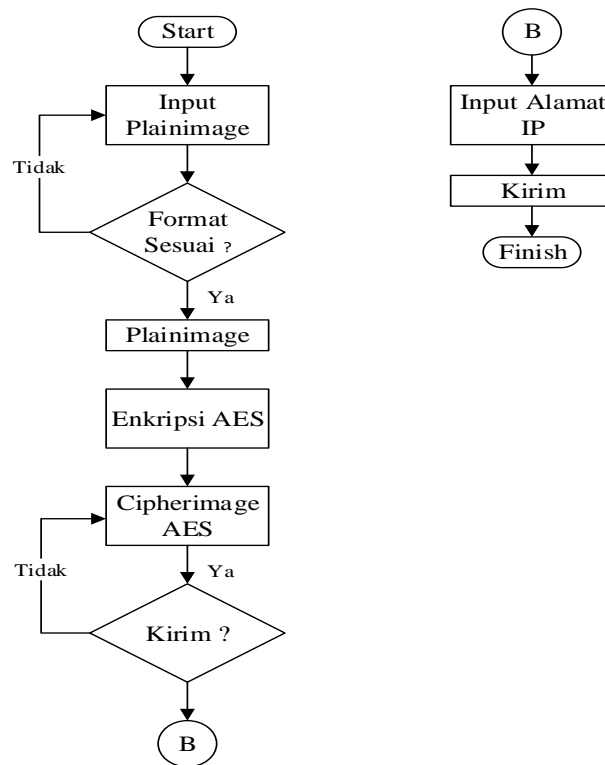
Berdasarkan rumusan masalah pada bab sebelumnya dan paparan di atas, masalah yang terjadi adalah bagaimana sebuah citra digital yang belum disandikan dapat diamankan dengan teknik kriptografi serta didistribusikan secara lokal dengan bantuan aplikasi berbasis *clien-server*. Sebelum melakukan teknik enkripsi kriptografi, terlebih dahulu menentukan pola citra yang akan dienkripsi. Pola citra yang akan dienkripsi pada pembahasan skripsi ini adalah sebuah citra warna RGB dengan format yang sudah ditentukan yaitu JPG, PNG dan BMP. Adapun proses

input dan pengaman citra dengan teknik kriptografi dapat dilihat pada gambar berikut:



Gambar 3.2 Diagram Proses Pengamanan Citra

Sedangkan proses *input* dan pengamanan citra dengan aplikasi berbasis *client-server* dapat dilihat pada gambar berikut :



Gambar 3.3 Diagram Proses Pengamanan Citra Berbasis *Client-Server*

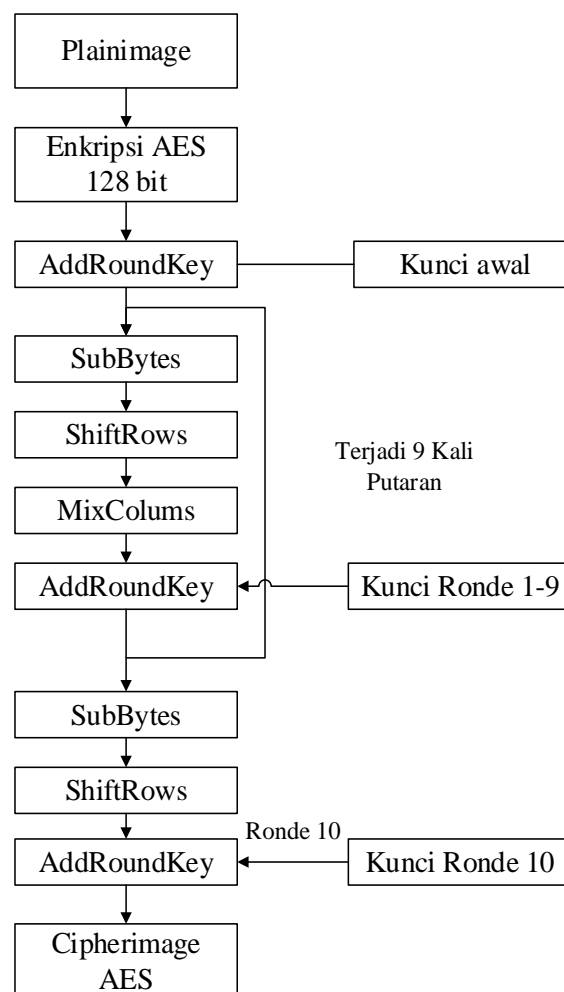
Agar algoritma dapat berjalan dengan baik terhadap enkripsi citra digital, terlebih dahulu citra diekstraksi untuk mendapatkan informasi nilai desimal pada setiap *pixel* citra. Citra yang akan diekstraksi pada kasus ini adalah citra RGB yang memiliki *bitdepth* 24 bit. Maksud dari 24 bit adalah di dalam 1 *pixel* citra digital memiliki nilai warna RGB dengan ruang sebanyak 24 bit. R (*Red*) memiliki nilai 8 bit, G (*Green*) memiliki nilai 8 bit dan B (*Blue*) memiliki nilai 8 bit. Nilai RGB disetiap *pixel* inilah yang akan dienkripsi menggunakan algoritma kriptografi.

Proses enkripsi algoritma kriptografi yang akan diterapkan adalah algoritma AES 128 bit. *Input* yang digunakan untuk penerapan algoritma AES berupa *plainimage*. Algoritma AES 128 bit pada setiap satu kali enkripsinya membutuhkan satu blok *plainimage* yang berjumlah sebanyak 128 bit atau 16 *byte* (karakter).

Enkripsi citra digital algoritma AES akan mengelompokkan setiap nilai warna *pixel plainimage* yang terdiri dari nilai RGB kedalam blok 128 bit dan kunci 128 bit untuk proses satu enkripsi. *Output* yang dihasilkan pada enkripsi berupa *cipherimage* AES. Berikut gambar proses enkripsi dan dekripsi algoritma AES 128 bit :

1. Diagram enkripsi Algoritma AES 128 bit

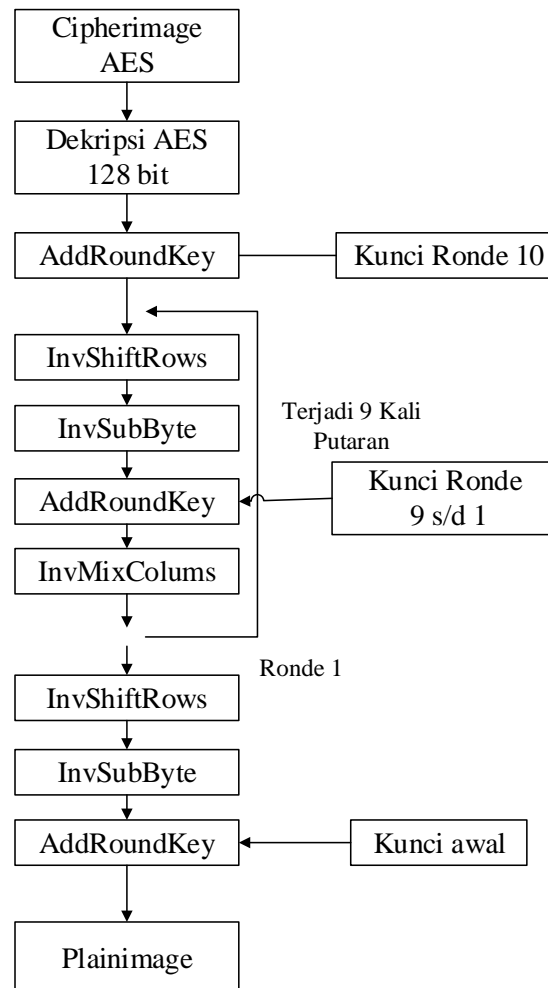
Diagram enkripsi algoritma AES 128 bit pada *plainimage* sebagai berikut:



Gambar 3.4 Diagram Enkripsi AES

2. Diagram Dekripsi Algoritma AES 128 bit

Proses dekripsi adalah kebalikan proses enkripsi. Diagram dekripsi algoritma AES 128 bit pada *cipherimage* sebagai berikut.



Gambar 3.5 Diagram Dekripsi AES

Aplikasi yang dirancang berbasis jaringan *client* dan *server* sehingga hasil enkripsi akhir berupa *cipherimage* dapat langsung dikirim melalui jaringan lokal antara *client* satu dengan yang lainnya. Pemanfaatan aplikasi berbasis *client* dan *server* ini mengurangi tingkat pencurian data karena pengiriman dapat dilakukan tanpa adanya koneksi *internet* dalam ruang lingkup jaringan lokal yang saling

berhubungan satu sama lain baik secara kabel dan tanpa kabel dengan pemanfaatan alamat *ip address* komputer.

3.3.1 Kelemahan Sistem yang berjalan

Berdasarkan hasil dari analisa yang diperoleh penulis dapat menguraikan beberapa kelemahan pada sistem yang sedang berjalan.

- a. Kurangnya keamanan pada citra digital karena sering terjadi penyadapan pada pixel.
- b. Tidak ada sistem keamanan dalam proses transfer gambar.

3.3.2 Analisa Kebutuhan Sistem

Analisa Kebutuhan sistem berfungsi untuk menentukan perangkat apa saja yang dibutuhkan dalam pembuatan aplikasi keamanan citra digital yang meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*). Dengan menggunakan analisa kebutuhan sistem maka dapat diketahui kebutuhan minimum yang diperlukan untuk membuat aplikasi keamanan citra digital tersebut. Berikut ini adalah penjabaran tentang spesifikasi *hardware* dan *software* yang dibutuhkan dalam pembuatan aplikasi keamanan citra digital.

- a. Kebutuhan Perangkat Keras (*Hardware*)

Adapun kebutuhan perangkat keras (*hardware*) yang digunakan untuk mendukung Aplikasi Keamanan Citra Digital adalah sebagai berikut :

Tabel 3.1 Kebutuhan *Hardware*

Nama Komponen	Spesifikasi
<i>Procesor</i>	Intel(R) Core(TM) i3-380M
<i>Memory</i>	4GB

Tabel Lanjutan 3.1 *Kebutuhan Hardware*

Nama Komponen	Spesifikasi
<i>Harddisk</i>	100 GB
Monitor	<i>Standard</i>
<i>Keyboard / Mouse</i>	<i>Standard</i>

b. *Kebutuhan Perangkat Lunak (Software)*

Adapun kebutuhan perangkat lunak (*software*) penulis yang digunakan untuk perancangan Aplikasi Keamanan Citra Digital adalah sebagai berikut :

- 1) *Visual Studio 2010 Ultimate Edition*
- 2) *Microsoft Visio 2010*
- 3) *Microsoft Office Exel 2007*

3.4 Rancangan Penelitian

Perancangan sistem dapat memberikan gambaran tentang sistem yang akan dibuat. Beberapa tahap dalam perancangan sistem dalam pembahasan skripsi ini yaitu :

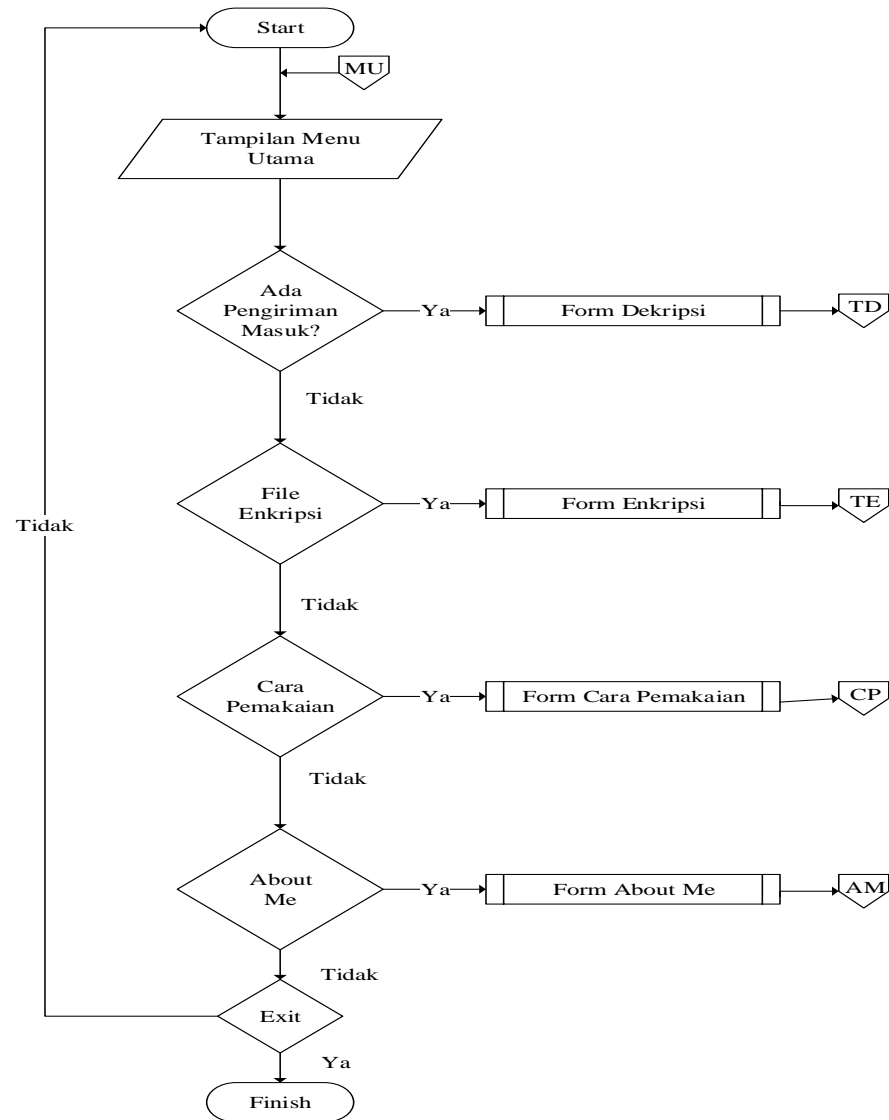
1. *Flowchart*
2. *Activity Diagram dan Use Case*
3. *Interface Input dan Output*

3.4.1 Flowchart Sistem

Flowchart sistem adalah bagian-bagian yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. Tahapan pengamanan data citra digital dilakukan dengan enkripsi dan dekripsi menggunakan algoritma AES 128-bit. Adapun langkah-langkah dari enkripsi dan dekripsi menggunakan algoritma AES dapat dilihat pada gambaran *flowchart*.

1. *Flowchart* Sistem Keseluruhan

Flowchart Sistem aplikasi keseluruhan menjelaskan tentang alur yang akan dirancang dengan antarmuka tampilan *form* pertama yang muncul ketika aplikasi dibuka oleh *user*. Adapun alur-alur dari sistem aplikasi keseluruhan dapat dijelaskan pada gambar *flowchart* dibawah ini :



Gambar 3.6 *Flowchart* Sistem

Berdasarkan pada gambar *Flowchart* Sistem di atas dapat dijelaskan sebagai berikut :

a. Tampilan Menu Utama

Merupakan *form* yang akan tampil ketika pertama kali aplikasi dibuka oleh *user* yang menggunakan.

b. Form Dekripsi

Form ini akan tampil ketika penerima berada di menu utama aplikasi dan mendapatkan kiriman file enkripsi dari pengirim.

c. Form Enkripsi

Merupakan *form* enkripsi yang digunakan *user* untuk pemilihan gambar serta proses enkripsi dan pengiriman gambar yang sudah dienkripsi.

d. Cara Pemakaian

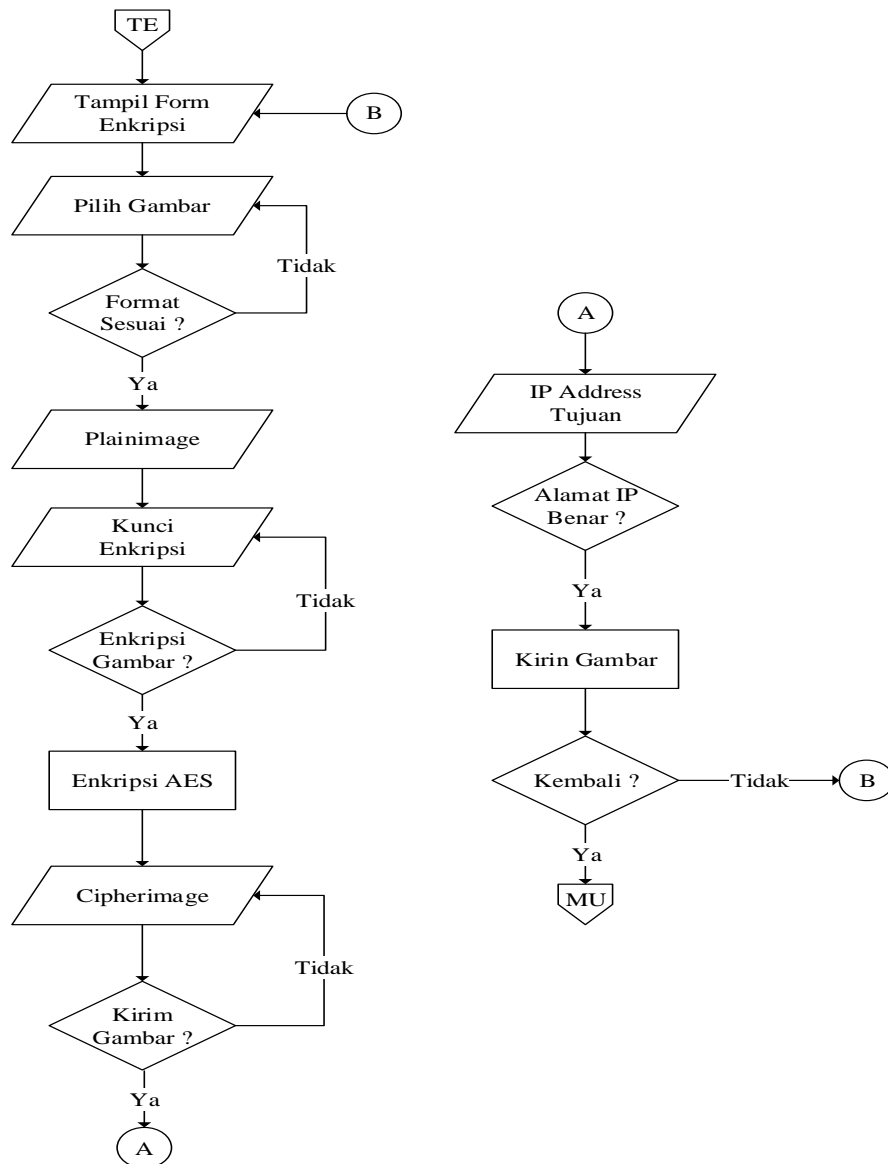
Merupakan *form* yang akan menampilkan cara pemakaian dari keseluruhan sistem aplikasi yang akan dibangun.

e. About Me

Merupakan *form* yang akan menampilkan data tentang perancang aplikasi yang akan dibangun.

2. *Flowchart* Enkripsi

Flowchart proses enkripsi pada aplikasi keamanan citra digital menggunakan algoritma AES berbasis *client-server* keseluruhannya dapat dilihat pada gambar dibawah ini :



Gambar 3.7 Flowchart Enkripsi

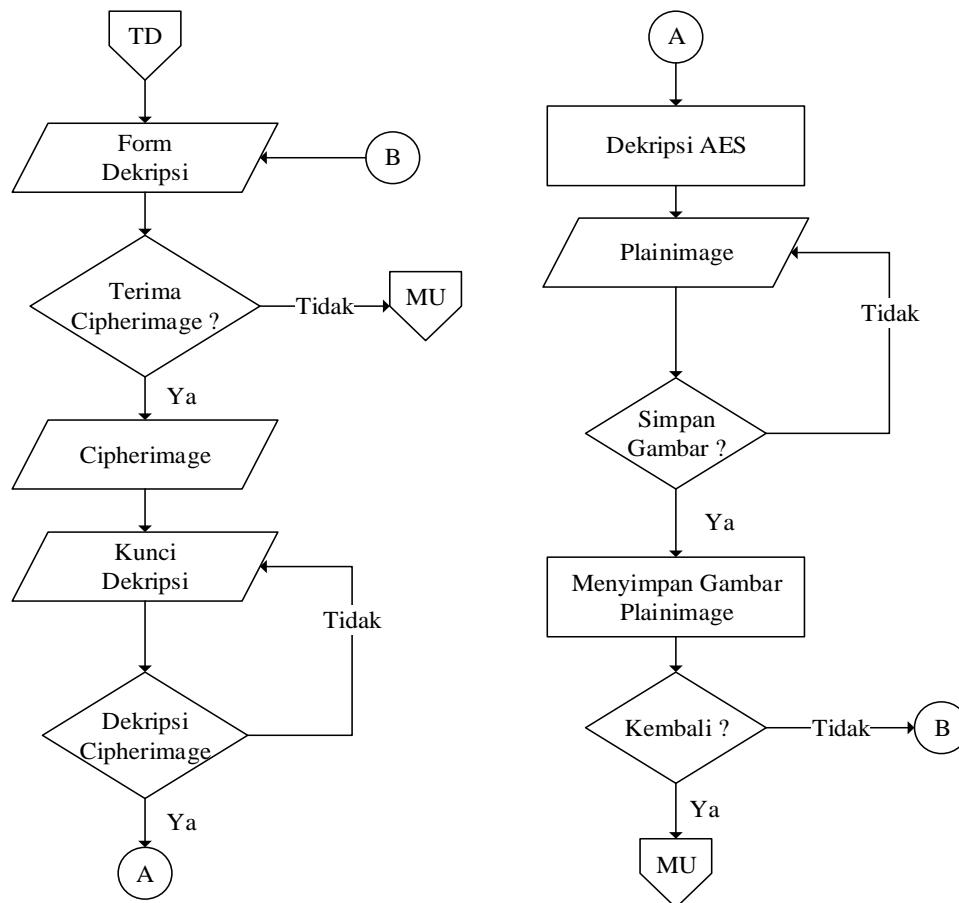
Proses enkripsi pada gambar di atas dapat dijelaskan sebagai berikut :

- Masukan gambar rahasia yang akan di enkripsi. Format gambar sudah ditentukan yaitu, jpg, png, dan bmp.
- Masukan kunci rahasia yang sudah ditentukan untuk proses enkripsi.
- Proses enkripsi menggunakan algoritma AES 128 bit terhadap *plainimage* sehingga didapat *cipherimage* AES.

- d. Kemudian untuk proses pengiriman, masukan alamat *ip address* komputer penerima yang sudah terhubung secara lokal terhadap komputer pengirim. Jika *ip address* benar maka pesan akan terkirim.

3. Flowchart Dekripsi

Flowchart proses dekripsi pada aplikasi keamanan citra digital menggunakan algoritma AES berbasis *client-server* keseluruhan dapat dilihat pada gambar berikut :



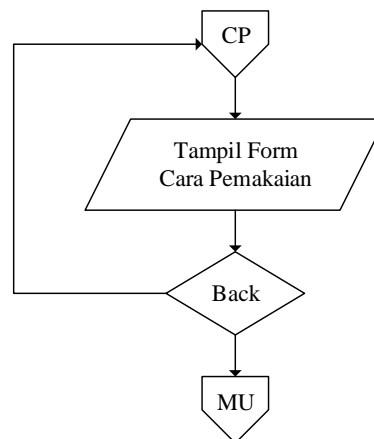
Gambar 3.8 Flowchart Dekripsi

Proses dekripsi pada gambar diatas dapat dijelaskan sebagai berikut :

- a. *Cipherimage* yang sudah dikirim oleh pengirim akan diterima oleh penerima. Penerima berhak menentukan terima atau tolak. Jika diterima maka *cipherimage* akhir hasil enkripsi akan masuk.
- b. *Cipherimage* yang sudah dikirim oleh pengirim akan diterima oleh penerima. Penerima berhak menentukan terima atau tolak. Jika diterima maka *cipherimage* akhir hasil enkripsi akan masuk.
- c. Proses dekripsi menggunakan algoritma AES 128 bit terhadap *cipherimage* sehingga didapatkan *plainimage* awal dan gambar dapat disimpan.

4. *Flowchart* Cara Pemakaian

Flowchart cara pemakaian akan menjelaskan aliran kegiatan yang dapat dilakukan oleh *user* pada saat mengakses *form* aplikasi ini. Berikut *flowchart* cara pemakaian :

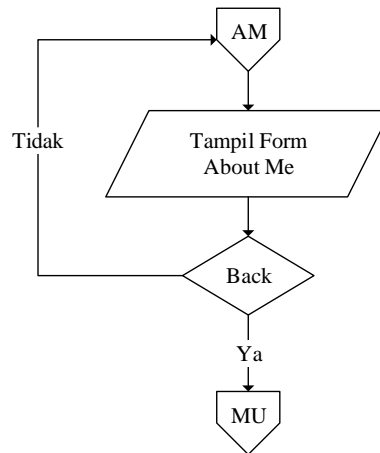


Gambar 3.9 *Flowchart* Cara Pemakaian

5. *Flowchart* About Me

Flowchart about me akan menampilkan data perancang aplikasi yang dibangun.

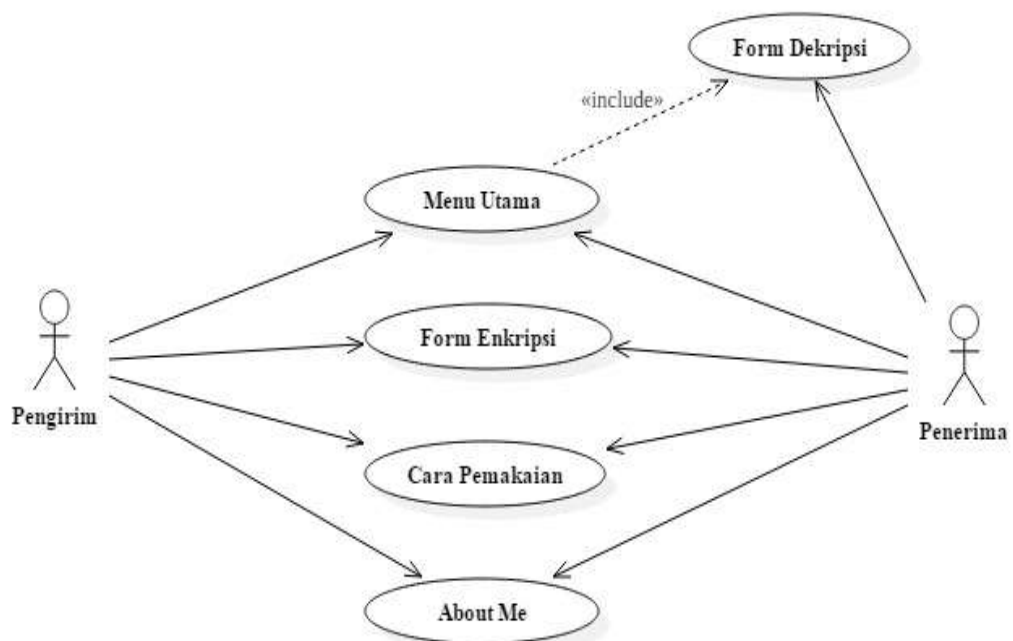
Berikut *flowchart* About Me :



Gambar 3.10 Flowchart About Me

3.4.2 Use Case Diagram

Use case diagram merupakan gambaran fungsional dari sistem aplikasi yang akan dirancang dan siapa saja yang berhak untuk menggunakannya. Adapun Use case diagram pada aplikasi yang akan dirancang dalam skripsi ini dapat dilihat pada gambar dibawah ini :



Gambar 3.11 Use Case Diagram

Berdasarkan gambar use case di atas, adapun keterangannya dapat dijelaskan sebagai berikut :

1. Menu Utama

Menu Utama adalah menu yang menampilkan sub menu untuk eksekusi sesuai keinginan *user* seperti *form* enkripsi, cara pemakaian dan about me.

2. Form Enkripsi

Menu ini adalah proses untuk melakukan enkripsi dan pengiriman terhadap citra digital.

3. Form Dekripsi

Menu ini akan terbuka apabila ada pengiriman yang masuk.

4. Cara Pemakaian

Menu ini menampilkan cara pemakaian aplikasi yang dibangun.

5. About Me

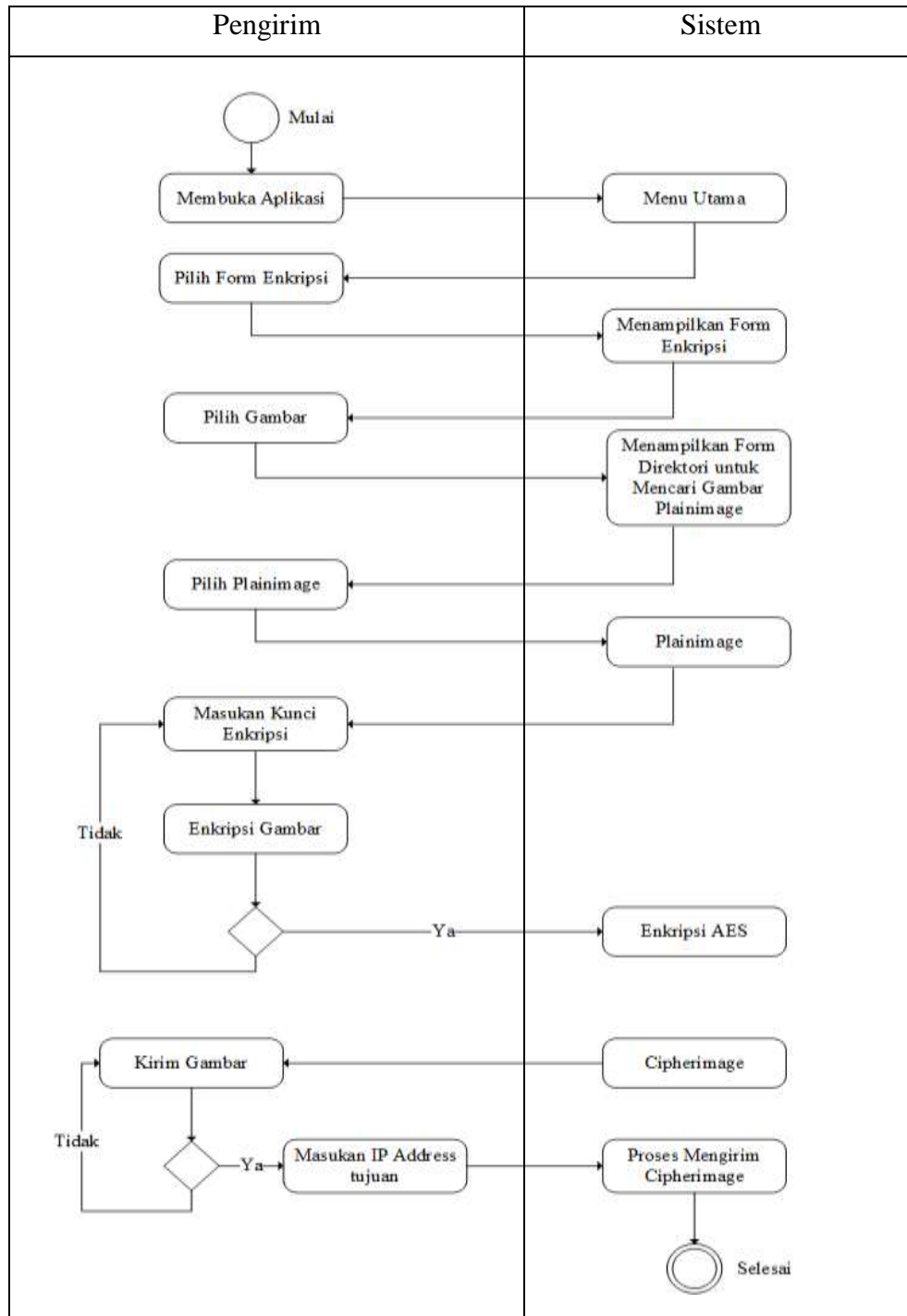
Menu ini menampilkan data diri perancang aplikasi yang dibangun.

3.4.3 Activity Diagram

Activity diagram menjelaskan alur dari aktivitas dalam sistem yang akan dirancang. Adapun *Activity diagram* pada sistem aplikasi yang akan dirancang dibagi menjadi 4 bagian yaitu, *Activity diagram* proses enkripsi serta pengiriman *file* gambar oleh *user*, *Activity diagram* proses penerima *file* gambar serta proses dekripsi oleh *user* penerima, *Activity diagram* proses cara pemakaian dan *Activity diagram* tentang perancang aplikasi.

1. Activity Diagram Sistem Proses Enkripsi dan Pengirim

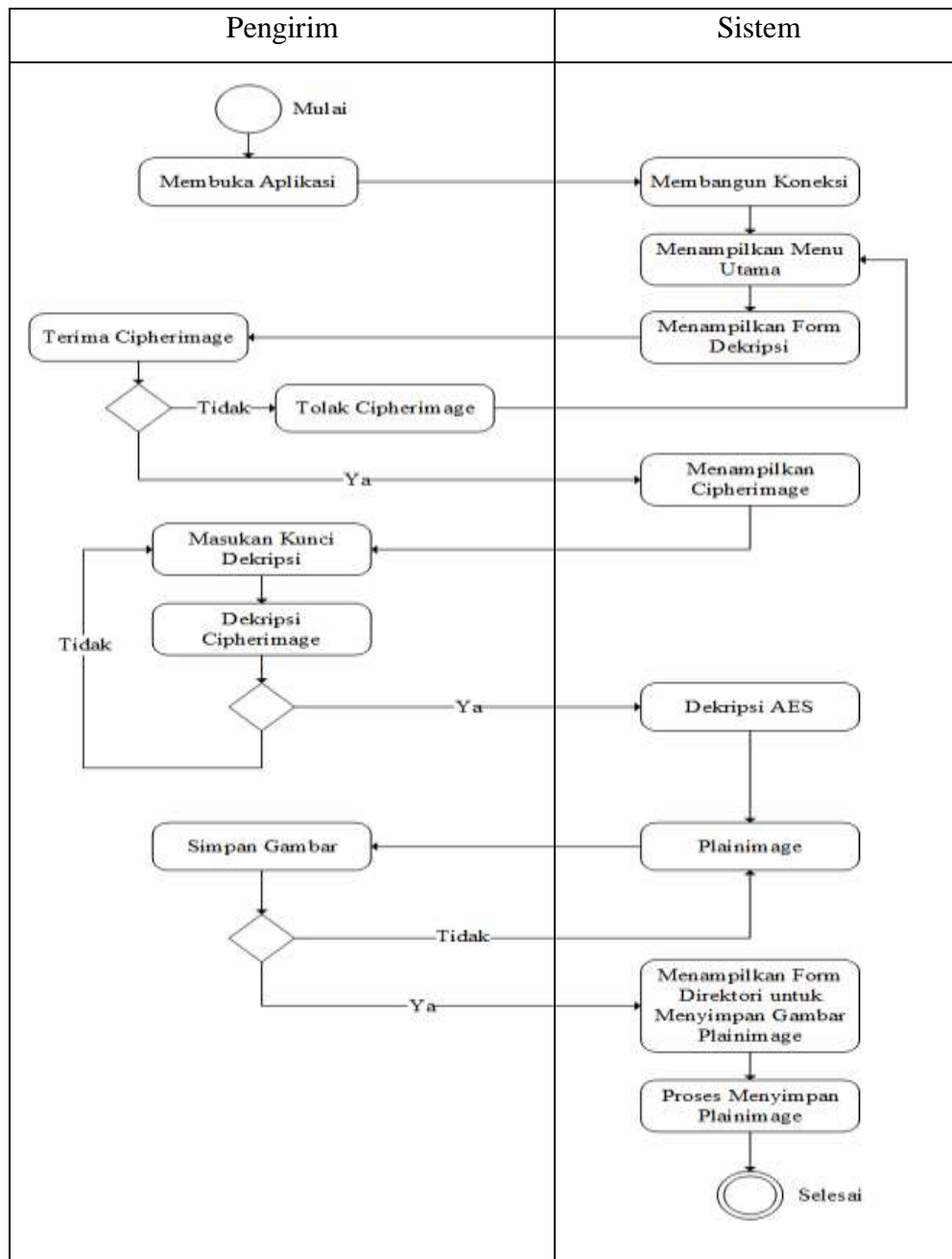
Activity diagram proses enkripsi dan pengirim pada sistem yang akan dirancang dapat dilihat pada gambar dibawah ini :



Gambar 3.12 Activity Diagram Proses Enkripsi dan Pengiriman

2. Activity Diagram Sistem Proses Penerimaan dan Dekripsi

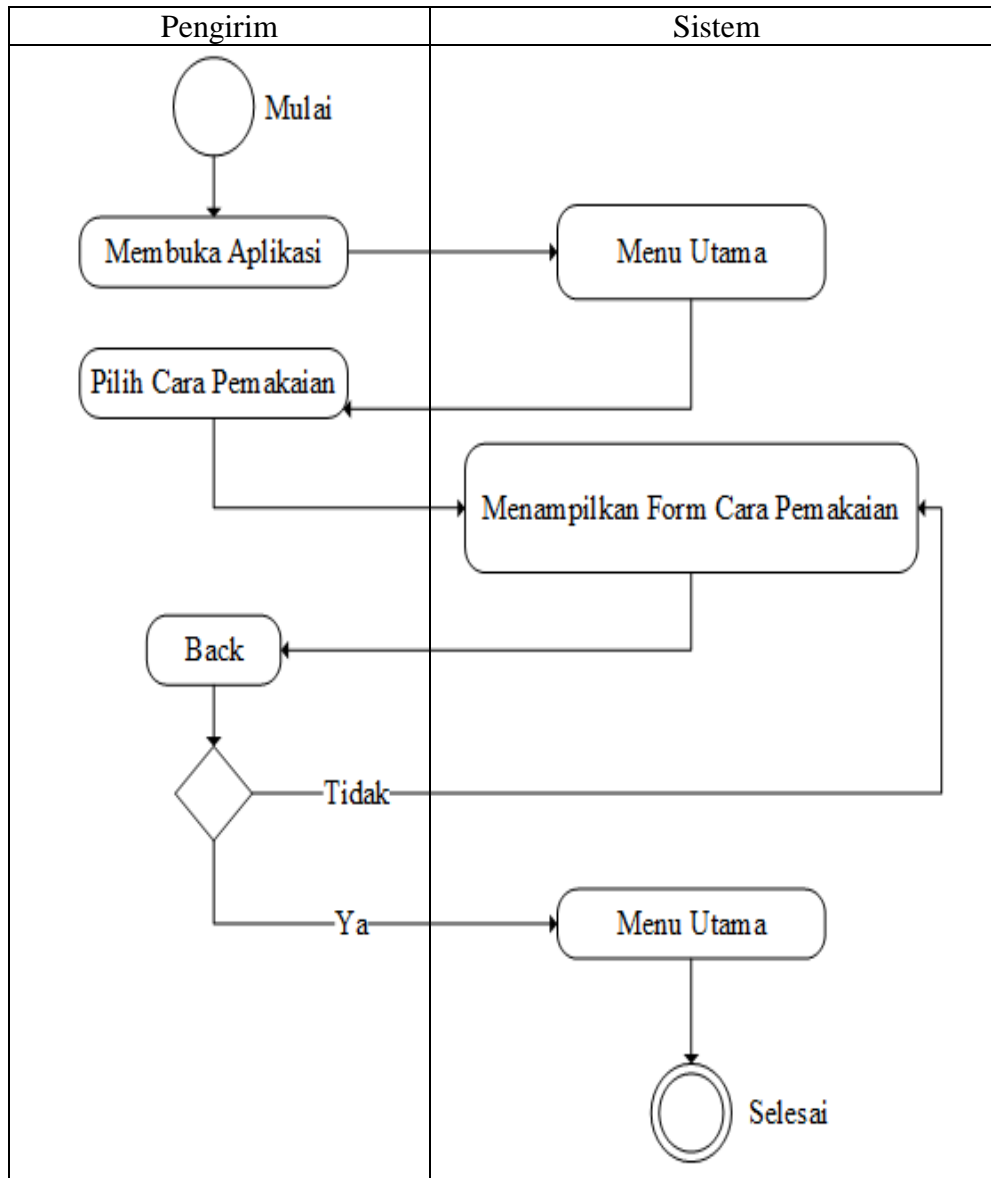
Activity diagram sistem proses penerimaan dan dekripsi pada sistem aplikasi yang akan dirancang dapat dilihat pada gambar dibawah ini :



Gambar 3.13 Activity Diagram Proses Penerimaan dan Dekripsi

3. Activity Diagram Cara Pemakaian

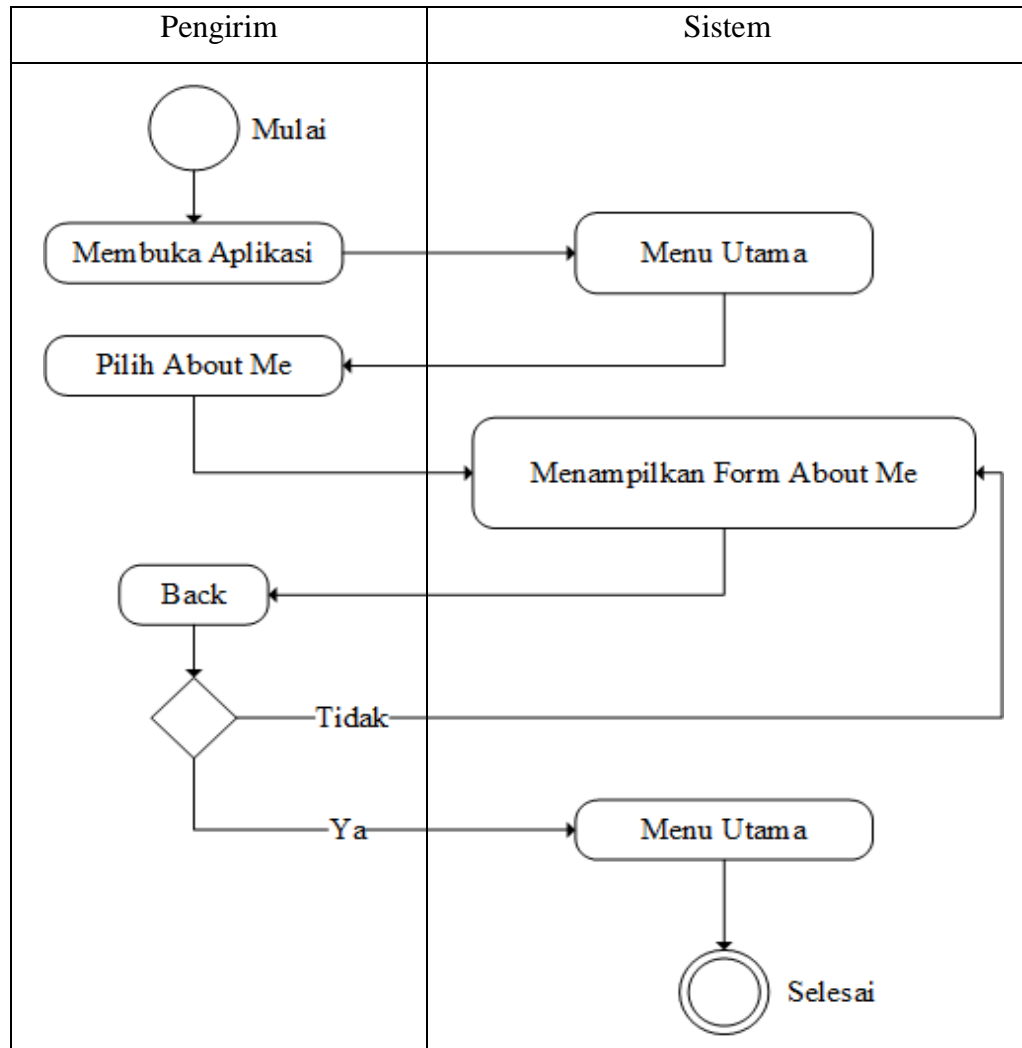
Activity diagram cara pemakaian aplikasi pada sistem aplikasi yang akan dirancang dapat dilihat pada gambar dibawah ini :



Gambar 3.14 Activity Diagram Cara Pemakaian

4. Activity Diagram Tentang Perancang Aplikasi

Activity diagram tentang perancang aplikasi pada sistem aplikasi yang akan dirancang dapat dilihat pada gambar dibawah ini :



Gambar 3.15 Activity Diagram About Me

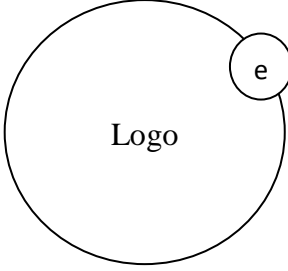
3.4.4 Perancangan Interface

Perancangan *interface input output* bertujuan untuk merancang antar muka aplikasi yang hendak dibangun kedalam sebuah perangkat lunak sehingga lebih

mudah dalam pembuatan aplikasi dan mudah dimengerti. Berikut adalah bentuk rancangan *input* dan *output* sistem yang nantinya akan diimplementasikan kedalam sebuah aplikasi.

1. Rancangan Menu Utama

Menu utama adalah *form* yang akan muncul ketika program aplikasi dibuka pertama kali. Adapun tampilan rancangan *form* menu utama adalah sebagai berikut :

Aplikasi Client-Server Kriptografi AES				X
Form Enkripsi	a	Cara Pemakaian	b	About Me
			c	Exit
<p>APLIKASI KEAMANAN CITRA DIGITAL MENGUNAKAN ALGORITMA AES 128-BIT</p>  <p>Logo</p> <p>UNIVERSITAS PEMBANGUNAN PANCA BUDI PROGRAM STUDI SISTEM KOMPUTER 2019</p>				
Host=xxx Ip=xxx		f		

Gambar 3.16 Rancangan Menu Utama

Berdasarkan gambar 3.16 di atas dapat dijelaskan keterangan sebagai berikut :

- Menu yang menampilkan *form* untuk melakukan proses enkripsi dan pengiriman sesudah enkripsi.
- Menu yang menampilkan *form* cara pemakaian aplikasi.
- Menu yang menampilkan *form* tentang penulis dan perancang aplikasi.

- d. Menu untuk menutup aplikasi.
- e. Logo gambar Universitas Pembangunan Pancabudi.
- f. Untuk menampilkan nama *host* komputer dan alamat *ip address* komputer.

2. Rancangan *Form* Enkripsi

Form enkripsi berfungsi sebagai *interface* bagi pengguna aplikasi pada saat melakukan kegiatan enkripsi dan mengirim citra digital. Melalui *form* enkripsi user dapat memilih gambar yang akan dienkripsi, memasukan kunci rahasia, memasukan *ip address* komputer penerima serta mengirim gambar yang sudah dienkripsi. Adapun rancangan *interface form* enkripsi dapat dilihat pada gambar berikut :

Gambar 3.17 Rancangan *Form* Enkripsi

Berdasarkan gambar 3.17 di atas dapat dijelaskan keterangan sebagai berikut :

- a. *Button* untuk memilih citra *plainimage* dari media penyimpanan.
 - b. *TextBox* untuk menampilkan informasi sumber *plainimage*.
 - c. *PictureBox* untuk menampilkan citra *plainimage*.
 - d. *Label* untuk menampilkan nama citra *plainimage*.
 - e. *Label* untuk menampilkan ukuran citra *plainimage*.
 - f. *TextBox* untuk menginput kunci rahasia.
 - g. *TextBox* untuk menginput *ip address* tujuan pengiriman.
 - h. *Button* untuk menghapus informasi kunci rahasia dan *ip address* tujuan pengiriman.
 - i. *Button* untuk melakukan proses enkripsi pada citra *plainimage*.
 - j. *ProgressBar* untuk menampilkan progres enkripsi.
 - k. *TextBox* untuk menampilkan waktu enkripsi.
 - l. *Label* untuk menunjukkan status proses enkripsi.
 - m. *PictureBox* untuk menampilkan citra *cipherimage* yang sudah terenkripsi.
 - n. *Button* untuk melakukan proses pengiriman citra yang sudah terenkripsi.
 - o. *Button* untuk kembali ke menu utama aplikasi.
3. Rancangan *Form* Dekripsi

Form dekripsi merupakan *interface* yang akan muncul bagi pengguna yang menerima *transfer image* berupa *cipherimage* dari pengirim. Melalui *form* ini penerima dapat mendekripsi gambar yang sudah terenkripsi dengan memasukan kunci dan menyimpan gambar yang sudah didekripsi. Adapun rancangan *interface form* dekripsi dapat dilihat pada gambar berikut :

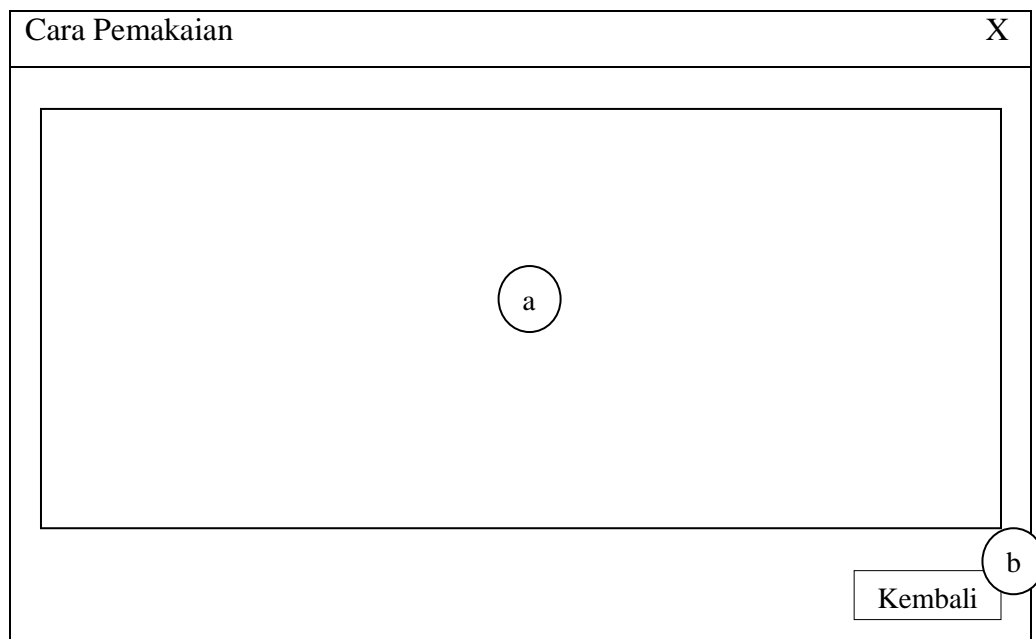
Gambar 3.18 Rancangan *Form Dekripsi*

Berdasarkan gambar 3.18 di atas dapat dijelaskan keterangan sebagai berikut :

- a. *Button* untuk menerima *cipherimage* yang dikirimkan oleh pengirim.
- b. *Button* untuk menolak *cipherimage* akhir yang dikirimkan oleh pengirim.
- c. *TextBox* untuk menampilkan informasi sumber IP Address pengirim.
- d. *Label* untuk menampilkan informasi nama citra *cipherimage*.
- e. *Label* untuk menampilkan informasi ukuran citra *cipherimage*.
- f. *PictureBox* untuk menampilkan citra *cipherimage* yang diterima.
- g. *TextBox* untuk menginput kunci rahasia dekripsi.
- h. *Button* untuk menghapus kunci rahasia dekripsi.
- i. *Button* untuk melakukan proses dekripsi *cipherimage*.

- j. *ProgressBar* untuk menampilkan proses dekripsi.
 - k. *TextBox* untuk menampilkan waktu dekripsi.
 - l. *PictureBox* untuk menampilkan *plainimage* yang sudah didekripsi.
 - m. *Button* untuk menyimpan citra digital yang sudah didekripsi.
 - n. *Button* untuk kembali ke menu utama aplikasi.
4. Rancangan *Form* Cara Pemakaian

Form cara pemakaian ini berfungsi untuk menampilkan rincian cara pemakaian dari aplikasi enkripsi dan dekripsi citra digital. Adapun rancangan *interface* dari *form* cara pemakaian dapat dilihat pada gambar berikut ini :



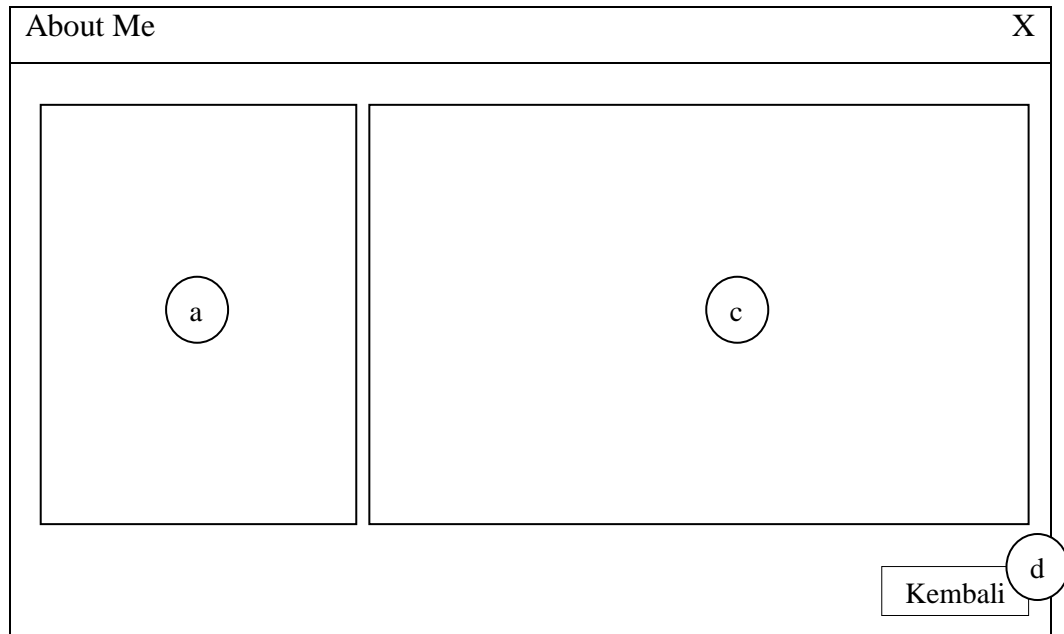
Gambar 3.19 Rancangan *Form* Cara Pemakaian

Berdasarkan gambar 3.19 di atas dapat dijelaskan keterangan sebagai berikut :

- a. *RichTextBox* untuk menampilkan informasi dari cara pemakaian aplikasi.
- b. *Button* untuk kembali pada menu utama aplikasi.

5. Rancangan *Form About Me*

Form about me berfungsi untuk menampilkan informasi tentang penulis dan perancangan aplikasi. Adapun rancangan *interface* pada *form about me* dapat dilihat pada gambar berikut :



Gambar 3.20 Rancangan *Form About Me*

Berdasarkan gambar 3.20 di atas dapat dijelaskan keterangan sebagai berikut :

- a. *PictureBox* untuk menampilkan gambar perancang aplikasi.
- b. *RichTextBox* untuk menampilkan informasi data diri perancang aplikasi.
- c. *Button* untuk kembali pada menu utama aplikasi.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Kebutuhan Sistem

Aplikasi yang telah dibuat membutuhkan beberapa kebutuhan sistem tambahan agar aplikasi dapat berjalan dengan sebagaimana mestinya. Adapun kebutuhan sistem aplikasi kriptografi berbasis *client – server* pada pembahasan ini dibagi menjadi 3 bagian yaitu, kebutuhan perangkat keras, kebutuhan perangkat lunak dan koneksi antara *server* (pengirim) dan *client* (penerima).

4.1.1 Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk menjalankan aplikasi kriptografi berbasis *client – server* menggunakan algoritma AES 128 bit harus mempunyai spesifikasi minimal. Hal ini bertujuan untuk mengoptimalkan jalannya aplikasi. Adapun kebutuhan minimal perangkat keras dibagi menjadi 2 bagian yaitu, kebutuhan minimal perangkat keras *server* (pengirim) dan kebutuhan minimal *client* (penerima) sebagai berikut :

1. *Server*

Adapun spesifikasi minimal dari perangkat keras yang akan digunakan sebagai pengirim pesan citra digital yang sudah terenkripsi adalah sebagai berikut :

- a. *Processor Intel Inside Dual Core*
- b. RAM 2 GB DDR3

- c. *Harddisk* SATA dengan kapasitas minimal 80GB
- d. LAN *Onboard* dan *Interface* Wifi

2. *Client*

Adapun spesifikasi minimal dari perangkat keras yang akan digunakan sebagai penerima pesan citra digital yang sudah terenkripsi adalah sebagai berikut :

- a. *Processor Intel Inside Dual Core*
- b. RAM 2 GB DDR3
- c. *Harddisk* SATA dengan kapasitas minimal 80GB
- d. LAN *Onboard* dan *Interface* Wifi

4.1.2 **Kebutuhan Perangkat Lunak**

Perangkat lunak sangat berperan penting dalam membantu dan mengoptimalkan jalannya aplikasi kriptografi berbasis *client – server*. Adapun perangkat lunak minimal yang harus dimiliki di dalam perangkat keras komputer *server* dan *client* untuk menjalankan aplikasi ini adalah sebagai berikut :

1. Sistem Operasi *windows* 2007
2. *Net Framework* Versi 4.5 ke atas
3. *Driver* LAN dan Wifi

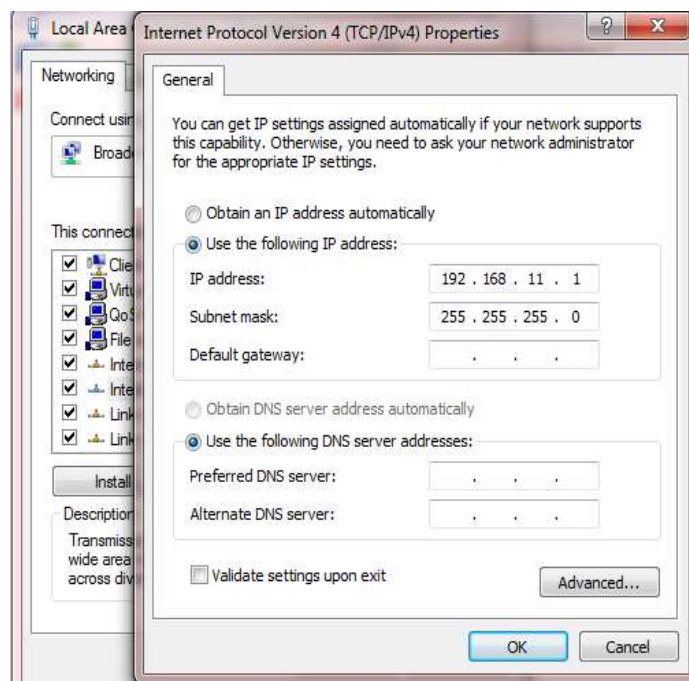
4.1.3 **Proses Koneksi**

Agar aplikasi berbasis *Client – server* dapat berjalan dengan baik, maka diperlukan koneksi antara pengirim dan penerima. Koneksi diperlukan untuk menghubungkan komputer – komputer yang akan menggunakan aplikasi. Proses

pengiriman dan penerimaan dapat dilakukan dengan koneksi secara kabel dan nirkabel (*wireless*). Adapun proses tersebut adalah sebagai berikut :

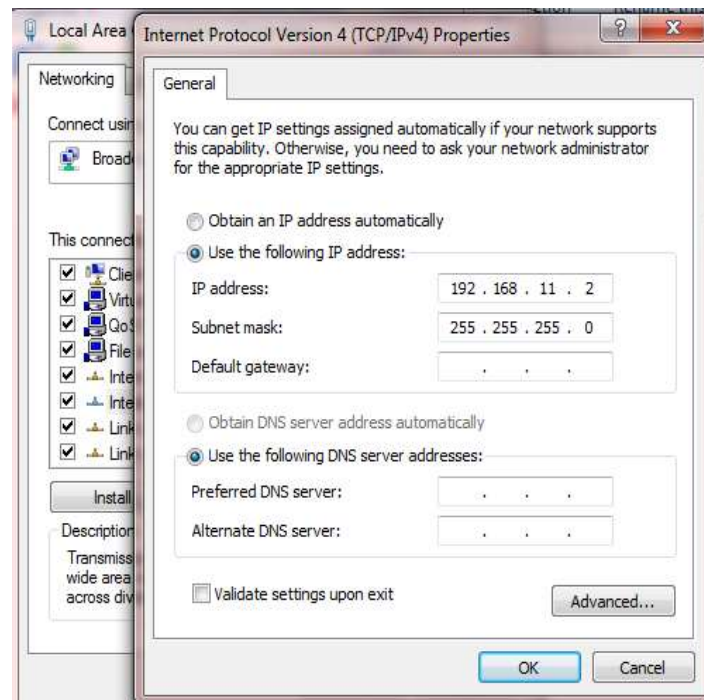
1. Proses Koneksi dengan Media Kabel

Proses koneksi menggunakan media kabel dilakukan dengan menghubungkan kabel UTP antara *server* dan *client*. Kemudian dilakukan pengaturan *ip address* yang satu subnet antara *server* dan *client*. Pengaturan *ip address* secara *static* dapat dilihat pada gambar dibawah ini :



Gambar 4.1 Pengaturan *IP address* terhadap *Server*

Berdasarkan pada gambar di atas pengaturan *ip address* terhadap *server* dilakukan dengan *ip address* kelas C yaitu 192.168.11.1. Sedangkan pengaturan *ip address* terhadap komputer *client* harus sama *Net ID* dan berbeda *Host ID* seperti gambar di bawah ini :



Gambar 4.2 Pengaturan *IP address* terhadap *Client*

Ip address inilah yang akan menentukan kemana *Cipherimage* hendak dikirim.

2. Proses Koneksi dengan Media Nirkabel (*Wireless*)

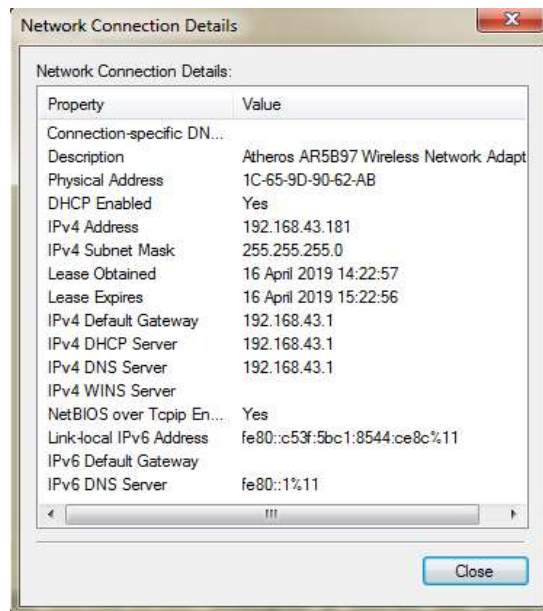
Proses koneksi dengan media *wireless* membutuhkan sinyal *wifi* yang menghubungkan antara komputer *server* dan *client*. Koneksi *wifi* umumnya memiliki pengaturan *ip address* secara dinamis. Adapun koneksi secara *wireless* dapat dilihat pada gambar di bawah ini :



Gambar 4.3 Koneksi Secara *Wireless*

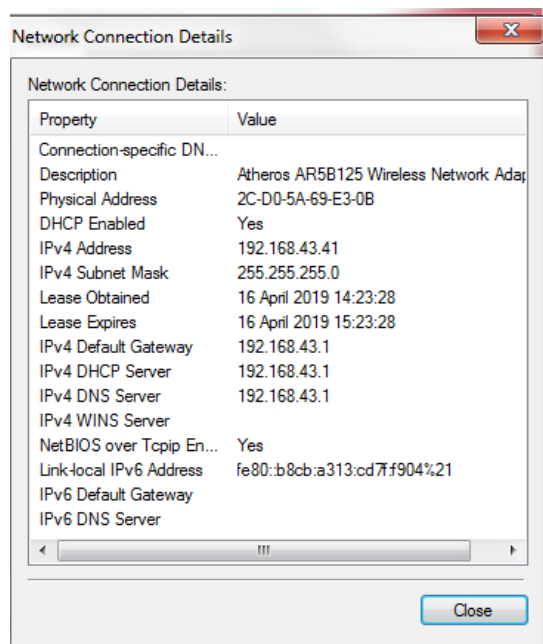
Setiap komputer *client* – *server* harus terkoneksi dengan *wifi* yang sama seperti gambar 4.3 di atas. Adapun pengaturan *ip* dilakukan secara dinamis

oleh *hostpot* penyedia layanan *Ip address* yang didapatkan oleh *server* yang terkoneksi dengan *wireless* tersebut adalah sebagai berikut :



Gambar 4.4 *IP Address Wireless Komputer Server*

Berdasarkan pada gambar diatas didapatkan *ip address* komputer *server* adalah 192.168.43.181. sedangkan *ip address* dari komputer *client* adalah sebagai berikut :



Gambar 4.5 *IP Address Wireless Komputer Client*

Berdasarkan pada gambar di atas didapatkan *ip address* komputer *client* adalah 192.168.43.41.

4.2 Tampilan Program Aplikasi

Tampilan program aplikasi dibagi menjadi 2 bagian yaitu tampilan proses *input* dan tampilan proses *output* serta proses pengiriman citra. Proses terdiri dari pengujian sistem aplikasi yang telah dibangun untuk melakukan enkripsi citra digital dengan teknik kriptografi algoritma AES.

4.2.1 Tampilan Input Program Aplikasi

Tampilan *input* pada pembahasan ini terdiri dari tampilan menu utama, tampilan *form* enkripsi, tampilan *form* dekripsi, tampilan *form* cara pemakaian, dan tampilan *form about me*. Kesimpulan yang diambil adalah apakah tampilan program sesuai dengan rancangan sebelumnya. Adapun tampilan keseluruhan menu program sebagai berikut :

1. Tampilan Menu Utama

Menu utama adalah tampilan yang akan pertama kali muncul ketika program aplikasi dibuka oleh *user*. Adapun tampilan *menu* utama aplikasi adalah sebagai berikut :



Gambar 4.6 Tampilan Menu Utama

Berdasarkan pada gambar tampilan menu utama, didapatkan beberapa tombol diantaranya :

- a. **FORM ENKRIPSI** berfungsi untuk menampilkan *form* Enkripsi yang digunakan sebagai proses enkripsi dan pengiriman sesudah enkripsi.
- b. **CARA PEMAKAIAN** berfungsi untuk menampilkan *form* Cara Pemakaian yang berguna bagi pengirim dan penerima untuk melihat cara menggunakan aplikasi.
- c. **ABOUT ME** berfungsi untuk menampilkan *form* About Me yang bertujuan memberitahu kepada pengguna aplikasi tentang biodata perancang aplikasi.
- d. **EXIT** berfungsi untuk mengakhiri pengguna aplikasi.

2. Tampilan *Form* Enkripsi

Tampilan *form* enkripsi adalah menu yang digunakan untuk proses enkripsi dan pengiriman citra digital. Adapun tampilan menu enkripsi dapat dilihat pada gambar di bawah ini :

Gambar 4.7 Tampilan *Form* Enkripsi

Berdasarkan pada gambar tampilan *form* enkripsi, didapatkan beberapa tombol diantaranya :

- a. **PILIH GAMBAR** berfungsi untuk menampilkan media penyimpanan agar pengirim dapat memilih gambar yang ingin dienkripsi.
- b. **Reset Informasi** berfungsi untuk menghapus informasi pada kolom kunci enkripsi dan Ip Address Tujuan.
- c. **ENKRIPSI PLAINIMAGE** berfungsi untuk mengenkripsi gambar Plainimage yang sudah di pilih oleh pengirim.

- d. **KIRIM CIPHERIMAGE** berfungsi untuk mengirim gambar Plainimage yang sudah dienkripsi kepada penerima.
- e. **Kembali** berfungsi untuk keluar dari *form* Enkripsi dan kembali ke Menu Utama.
3. Tampilan *Form* Dekripsi

Tampilan *form* dekripsi adalah menu yang akan tampil jika adanya proses pengiriman pesan yang masuk kepada penerima dengan syarat aplikasi terbuka dan berada pada menu utama. Adapun tampilan menu dekripsi dapat dilihat pada gambar di bawah ini :

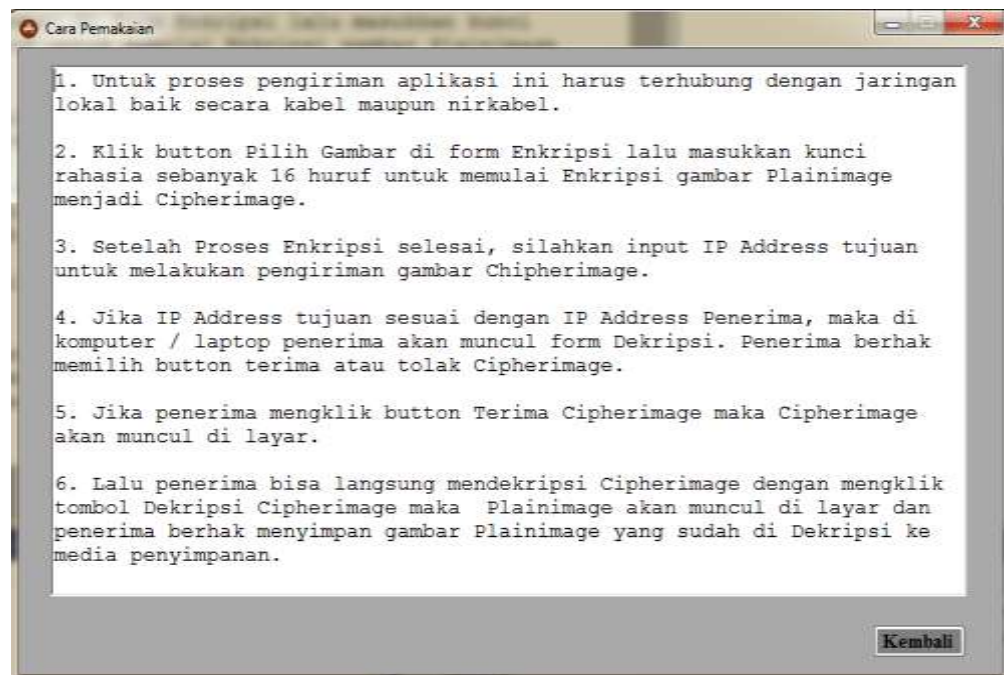
Gambar 4.8 Tampilan *Form* Dekripsi

Berdasarkan pada gambar tampilan *form* enkripsi, didapatkan beberapa tombol diantaranya :

- a. **TERIMA CIPHERIMAGE** berfungsi untuk menerima Cipherimage yang dikirim.
- b. **TOLAK CIPHERIMAGE** berfungsi untuk menolak Cipherimage yang ditolak.
- c. **Reset Kunci** berfungsi untuk menghapus informasi pada kolom kunci dekripsi.
- d. **DEKRIPSI CIPHERIMAGE** berfungsi untuk mendekripsi Cipherimage yang diterima.
- e. **SIMPAN PLAINIMAGE** berfungsi untuk menyimpan Plainimage yang sudah didekripsi ke media penyimpanan.
- f. **Kembali** berfungsi untuk keluar dari *form* Dekripsi dan kembali ke Menu Utama.

4. Tampilan Cara Pemakaian

Tampilan cara pemakaian ini berfungsi untuk memberikan informasi terhadap pengguna aplikasi bagaimana cara menggunakan aplikasi secara singkat. Adapun tampilan cara pemakaian di dalam aplikasi yang telah dibangun sebagai berikut :



Gambar 4.9 Tampilan Cara Pemakaian

Berdasarkan pada gambar tampilan *form* enkripsi, didapatkan beberapa tombol diantaranya :

- a. **Kembali** berfungsi untuk keluar dari *form* Cara Pemakaian dan kembali ke Menu Utama.

5. Tampilan *About Me*

Tampilan *about me* ini berfungsi untuk memberikan informasi terhadap pengguna aplikasi tentang penulis skripsi secara singkat. Adapun tampilan *about me* di dalam aplikasi yang telah dibangun sebagai berikut :



Gambar 4.10 Tampilan *About Me*

Berdasarkan pada gambar tampilan *form* enkripsi, didapatkan beberapa tombol diantaranya :

- b. **Kembali** berfungsi untuk keluar dari *form* about me dan kembali ke Menu Utama.

4.2.2 Tampilan Output Program Aplikasi

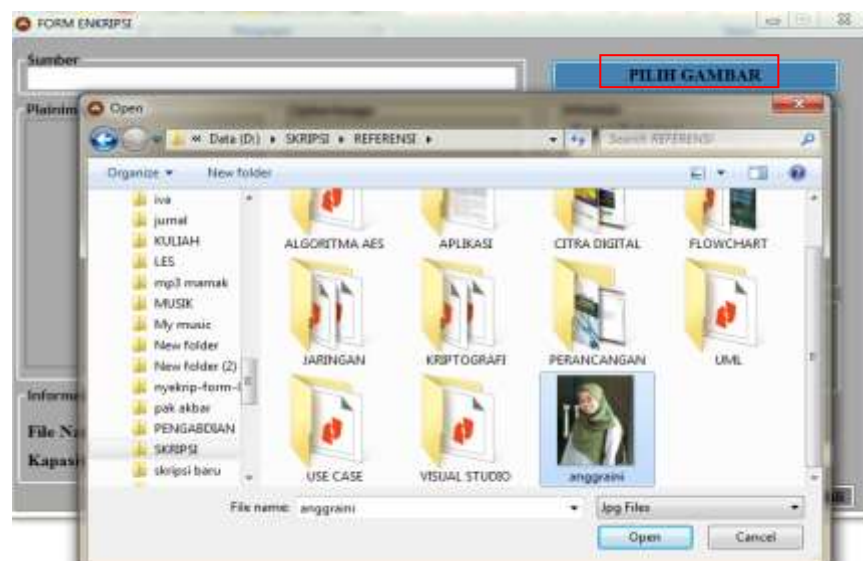
Tampilan *output* ini didapat dari proses pengujian sistem. Pengujian sistem aplikasi bertujuan untuk mengetahui apakah aplikasi berjalan dengan sesuai fungsinya, yaitu mengenkripsi dan dekripsi citra digital serta pengiriman pesan melalui jaringan. Pengujian memiliki 3 tahap yaitu proses enkripsi citra digital awal, proses pengiriman citra hasil enkripsi, dan proses dekripsi citra hasil enkripsi.

1. Proses Enkripsi Citra

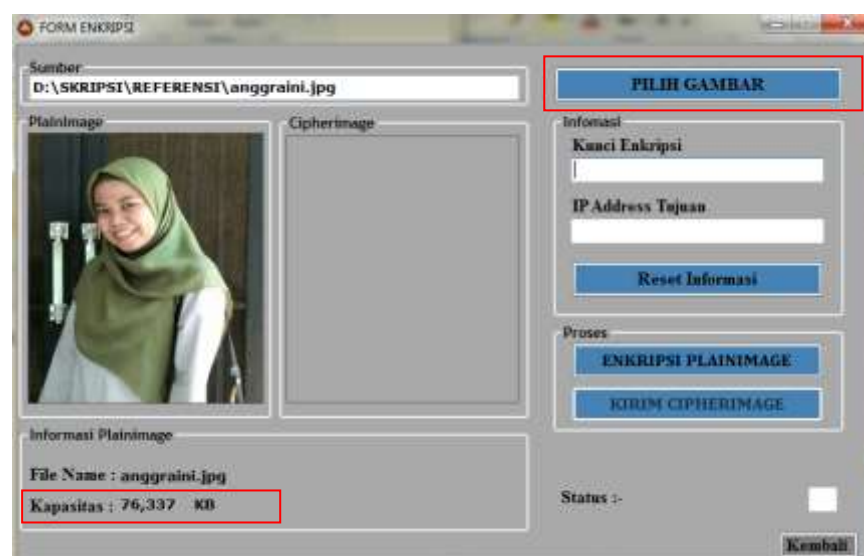
Proses enkripsi citra memiliki beberapa tahap yaitu, proses pemilihan citra digital awal (*plainimage*), proses pemasukan kunci dan proses enkripsi.

a. Pemilihan *Plainimage*.

Adapun proses pemilihan citra awal (*plainimage*) dilakukan dengan memilih *stripbutton* pilih gambar seperti pada gambar di bawah ini :



Gambar 4.11 Tampilan Pemilihan Citra *Plainimage*



Gambar 4.12 Tampilan Citra *Plainimage*

Berdasarkan pada gambar 4.12 proses pemilihan dengan memilih tombol pilih gambar. Hasil dari pemilihan berupa *plainimage* dengan informasi nama *file* adalah *anggraini* ekstensi *.jpg* dan kapasitasnya adalah 76,337 KB.

b. Memasukan Kunci

Setelah proses pemilihan citra *plainimage* proses selanjutnya adalah memasukan kunci enkripsi. Adapun kunci enkripsi dapat dilihat pada gambar di bawah ini :

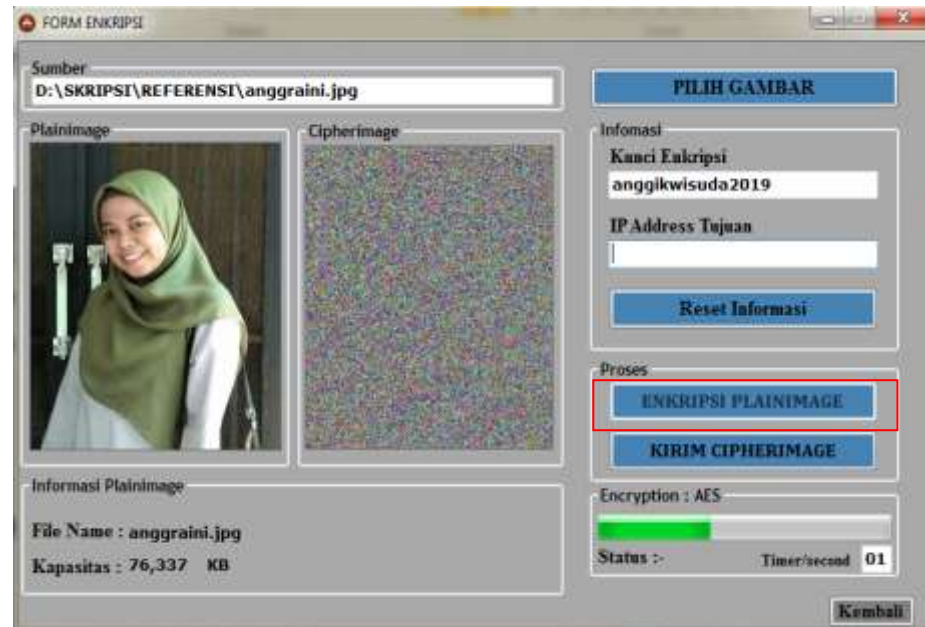
Gambar 4.13 Tampilan Proses memasukan Kunci

Berdasarkan pada gambar di atas kunci yang digunakan untuk proses enkripsi adalah *anggrkwisuda2019*.

c. Enkripsi *Plainimage*

Proses enkripsi dilakukan menggunakan algoritma AES 128 bit. Sebelum melakukan proses enkripsi, terlebih dahulu aplikasi akan membaca *byte*

dari *pixel* yang diinputkan. Adapun proses enkripsi tersebut dapat dilihat pada gambar di bawah ini :



Gambar 4.14 Tampilan Proses Enkripsi *Plainimage*

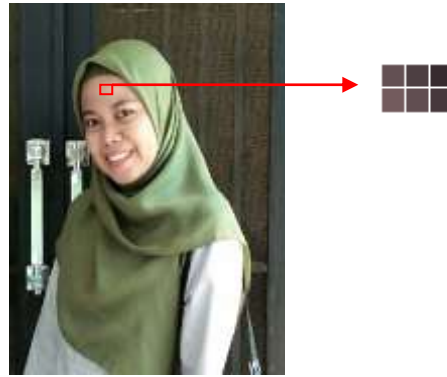
d. Perhitungan Manual Enkripsi AES 128-bit

Berikut diuraikan contoh penerapan algoritma AES dalam menyandikan sebuah citra digital berwarna dengan ekstensi jpg resolusi 259 x 339 dan *bitdepth* 24 bit.



Gambar 4.15 *Plainimage* dengan Resolusi 259 x 339

Berdasarkan pada gambar di atas akan diambil 6 *pixel* sebagai sampel dalam perhitungan manual. Enam *pixel* tersebut akan diambil nilai desimal warnanya dengan cara mengekstraksi setiap elemen *pixel*.



Gambar 4.16 Plainimage Sampel 6 Pixel

Nilai desimal elemen warna pada setiap *pixel* akan diekstraksi menggunakan *software* matlab. Perintah yang digunakan untuk menampilkan nilai RGB pixel didalam matlab adalah sebagai berikut :

```
function pushbutton1_Callback(hObject, eventdata, handles)
[nama_file1, nama_path1]=uigetfile(...
    {'*.bmp; *.jpg', *.png', 'File Citra (*.bmp, *.jpg, *.png)';
    '*.bmp', 'File Bitmap(*.bmp)';...
    '*.jpg', 'File Jpeg(*.jpg)';
    '*.png', 'File Png(*.png)';
    'Buka Citra Asli');
if ~isequal (nama_file1, 0);
    handles.data1=imread(fullfile(nama_path1, nama_file1));
    guidata(hObject, handles);
    handles.current_data1=handles.data1;
```

```

axes(handles.axes1)

imshow(handles.current_data1);

else

return

end

red=mean(mean(handles.current_data1(:,:,1)));

green=mean(mean(handles.current_data1(:,:,2)));

blue=mean(mean(handles.current_data1(:,:,3)));

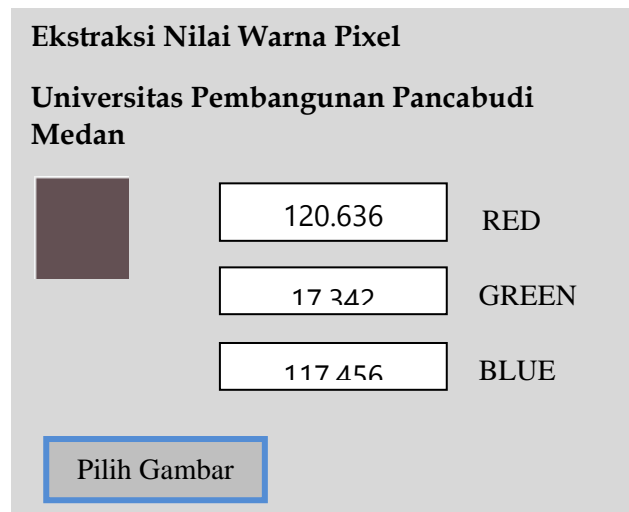
set(handles.edit1,'string',red);

set(handles.edit2,'string',green);

set(handles.edit3,'string',blue);

```

Adapun hasil proses ekstraksi setiap *pixel* pada *software* matlab sebagai berikut :



Gambar 4.17 Ekstraksi Nilai RGB *Pixel*

Berdasarkan proses ekstraksi *pixel* 1 pada gambar di atas, diperoleh nilai RGB *pixel* 1 sebagai berikut :

Tabel 4.1 Nilai RGB *Pixel* 1

Pixel	Warna	Plainimage
		Desimal
1	R	120
	G	17
	B	117

Ekstraksi *pixel* ke 2 dan seterusnya dilakukan dengan cara yang sama. Sehingga diperoleh nilai RGB dari 6 *pixel* sampel untuk perhitungan manual sebagai berikut:

Tabel 4.2 Nilai RGB *Pixel* Sampel

Pixel	Warna	Plainimage
		Desimal
1	R	120
	G	17
	B	117
2	R	130
	G	125
	B	85
3	R	37
	G	150
	B	65
4	R	56
	G	120
	B	87
5	R	75
	G	140
	B	30
6	R	120
	G	25
	B	10

Berdasarkan tabel di atas, untuk mempermudah dalam hitungan manual algoritma AES, maka penulis hanya mengambil nilai R pada *pixel* 6, sehingga nilai desimal dari *plainimage* adalah 120, 17, 117, 130, 125, 85, 37, 150, 65, 56, 120, 87, 75, 140, 30, 120. Kunci yang digunakan dalam hitungan manual adalah *anggikwisuda2019* panjang kunci 16 *byte*. Sebelum proses enkripsi, terlebih dahulu melakukan proses pembangkitan kunci. Pembangkitan kunci dilakukan sebanyak sepuluh putaran. Kunci awal diinput kedalam *array* 4x4 bernilai hexadesimal.

41	6E	67	67
69	6B	77	69
73	75	64	61
32	30	31	39

Pembangkitan kunci ronde 1 pada kolom pertama dilakukan dengan cara mengambil nilai kolom terakhir kunci sebelumnya lalu dilakukan rotasi dengan mengeser blok paling atas ke blok paling bawah seperti di bawah ini:

Sebelum Sesudah rotasi

67	69
69	61
61	39
39	67

Kemudian lakukan substitusi pada kolom yang sudah di *rotate* dengan tabel S-Box pada gambar gambar 2.1 di bab landasan teori. Sehingga mendapatkan hasil seperti di bawah ini:

F9
EF
12
85

Hasil substitusi kemudian di XOR dengan nilai kolom pertama kunci sebelumnya.

F9	XOR	41	=	B8
EF		69		86
12		73		61
85		32		B7

Kemudian di XOR kembali dengan nilai tabel *Round Constant* (Rcon)

kolom pertama untuk kunci ronde ke 1 seperti di bawah ini :

Tabel 4.3 *Round Constant*

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

B8	XOR	01	=	B9
86		00		86
61		00		61
B7		00		B7

Sehingga didapat kolom pertama kunci ronde ke 1 sebagai berikut :

B9
86
61
B7

Untuk pencarian nilai kolom kedua kunci ronde ke 1, lakukan XOR nilai kolom kedua kunci sebelumnya dengan nilai kolom pertama untuk kunci ronde berikutnya. Seperti di bawah ini :

6E	XOR	B9	=	D7
6B		86		ED
75		61		14
30		87		87

Sehingga didapat kolom kedua kunci ronde ke 1 sebagai berikut :

B9	D7
86	ED
61	14
87	87

Untuk pencarian nilai kolom ketiga kunci ronde ke 1, lakukan XOR nilai kolom ketiga kunci sebelumnya dengan nilai kolom kedua untuk kunci ronde berikutnya. Seperti di bawah ini :

67	XOR	D7	=	B0
77		ED		9A
64		14		70
31		87		86

Sehingga didapat kolom ketiga kunci ronde ke 1 sebagai berikut :

B9	D7	B0
86	ED	9A
61	14	70
87	87	86

Untuk pencarian nilai kolom keempat kunci ronde ke 1, lakukan XOR nilai kolom ketiga kunci sebelumnya dengan nilai kolom kedua untuk kunci ronde berikutnya. Seperti di bawah ini :

67	XOR	B0	=	D7
69		9A		F3
61		70		11
39		86		8F

Sehingga didapat kolom keempat dan keseluruhan kunci ronde ke 1 sebagai berikut :

B9	D7	B0	D7
86	ED	9A	F3
61	14	70	11
87	87	86	8F

Proses pembentukan kunci ronde ke 2 hingga ke 10 mengikuti langkah yang sama, sehingga didapatkan kunci keseluruhan ronde seperti berikut:

RoundKey 1 *RoundKey 2* *RoundKey 3* *RoundKey 4*

B9	D7	B0	D7	B6	61	D1	6	7F	1E	CF	C9	63	7D	B2	7B
86	ED	9A	F3	4	E9	73	80	81	68	1B	9B	3B	53	48	D3
61	14	70	11	12	6	76	67	D7	D1	A7	C0	52	83	24	E4
B7	87	B6	BF	B9	3E	88	7	D6	E8	60	67	0B	E3	83	E4

Gambar 4.18 *RoundKey 1* sampai 4 AES 128-bit

RoundKey 5 *RoundKey 2* *RoundKey 3* *RoundKey 8*

15	68	DA	A1	8D	E5	3F	9E	26	C3	FC	62	53	90	6C	0E
52	1	49	94	EE	EF	A6	3C	2	ED	4B	77	B3	5E	15	62
3B	B8	9C	78	DF	67	FB	83	49	2E	D5	56	19	37	E2	B4
2A	C9	4A	AE	18	D1	9B	35	13	C2	59	6C	B9	7B	22	4E

Gambar 4.1 *RoundKey 5* sampai 8 AES 128-bit

RoundKey 9 *RoundKey 10*

E2	72	1E	10	24	56	48	58
3E	60	75	17	65	5	70	67
36	1	E3	57	5D	5C	BF	E8
12	69	4B	5	D8	B1	FA	FF

Gambar 4.20 *RoundKey 9* sampai *10* AES 128-bit

1) *Initial Round*

Proses pertama sebelum masuk ronde 1 dinamakan *initial round*, siapkan nilai hexadesimal *plainimage* dan kunci ke dalam bentuk *array* 4x4 seperti di bawah ini :

<i>Plainimage</i> Hexadesimal					Kunci Hexadesimal			
78	11	75	82	XOR	41	6E	67	67
7D	55	25	96		69	6B	77	69
41	38	78	57		73	75	64	61
48	8C	1E	78		32	30	31	39

Selanjutnya adalah melakukan proses *AddRoundKey* dengan meng XOR setiap kolom bilangan hexadesimal *plainimage* dengan setiap kolom nilai hexadesimal kunci seperti di bawah ini :

78	41
7D	69
41	73
48	32

Lakukan XOR pada setiap kolom *state* dan kolom kunci dengan merubah nilai hexadesimal kedalam biner seperti di bawah ini :

$$\begin{array}{r}
 78 = 01111000 \\
 41 = 01000001 \\
 \hline
 \text{XOR} \\
 00111001 \text{ dalam hexadesimal adalah } 39
 \end{array}$$

$$\begin{array}{r}
 7D = 01111101 \\
 69 = 01101001 \\
 \hline
 \text{XOR} \\
 00010100 \text{ dalam hexadesimal adalah } 14
 \end{array}$$

$$\begin{array}{r}
 41 = 01000001 \\
 73 = 01110011 \\
 \hline
 \text{XOR} \\
 00110010 \text{ dalam hexadesimal adalah } 32
 \end{array}$$

$$\begin{array}{r}
 48 = 01001000 \\
 32 = 00110010 \\
 \hline
 \text{XOR} \\
 01111010 \text{ dalam hexadesimal adalah } 7A
 \end{array}$$

Sehingga didapat kolom pertama hasil *AddRoundKey* pertama sebagai di bawah ini :

39
14
32
7A

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua state di XOR dengan kolom kedua kunci dan seterusnya. Sehingga didapat hasil *AddroundKey* pertama (*initial round*) sebagai berikut :

39	7F	12	E5
14	3E	52	FF
32	4D	1C	36
7A	BC	2F	41

Hasil di atas akan menjadi *state* awal di ronde pertama algoritma AES 128 bit.

2) Enkripsi Ronde 1

Setiap ronde memiliki 4 proses yaitu *SubBytes*, *ShiftRows*, *Mixcolumns* dan *Addroundkey*, pada ronde terakhir hanya memiliki 3 proses yaitu *SubBytes*, *ShiftRows* dan *AddroundKey*. AES butuh 1 blok 128 bit *plainimage* yang didapat dari setiap nilai RGB *pixel*. Ronde pertama dapat dilakukan ketika hasil proses *initial round* pertama sudah didapat.

a) *SubBytes*

Proses ini adalah melakukan substitusi dengan menukar nilai setiap *byte state* dengan nilai *byte* yang sesuai pada tabel S-Box. Tabel S-Box dapat di lihat pada bab landasan teori gambar 2.1.

39	7F	12	E5
14	3E	52	FF
32	4D	1C	36
7A	BC	2F	41

Misal pada kolom pertama *byte* pertama adalah 39, maka elemen yang sesuai dengan tabel S-Box terletak pada baris ke-3 dan kolom ke-9 yaitu 12. Cara yang sama dilakukan sehingga didapat hasil *SubBytes* dengan tabel S-Box sebagai berikut :

12	D2	C9	D9
FA	B2	00	16
23	E3	9C	05
DA	65	15	83

b) *ShiftRows*

Proses ini melukan pergeseran pada setiap elemen tabel *array*, yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte* ke kiri, baris ketiga dilakukan pergeseran 2 *byte* ke kiri, baris ketiga dilakukan pergeseran 3 *byte* ke kiri.

Sebelum <i>ShiftRows</i>				Setelah <i>ShiftRows</i>			
12	D2	C9	D9	12	D2	C9	D9
FA	B2	00	16	B2	00	16	FA
23	E3	9C	05	9C	05	23	E3
DA	65	15	83	83	DA	65	15

c) *MixColumns*

Proses *MixColumns* adalah proses perkalian setiap kolom setelah proses *ShiftRows* dengan kolom matriks tabel *polynimial*. Operasi perkalian yang dimaksud adalah perkalian dengan 01 artinya nilai tidak berubah, perkalian dengan 02 artinya mengeser *byte* ke kiri, perkalian dengan 03 artinya mengeser *byte* ke kiri lalu melakukan XOR dengan nilai awal sebelum digeser. Setelah digeser apabila nilai lebih besar dari 0xFF dalam desimal 255

maka dilakukan XOR dengan 0x11B. Proses perkalian ini menggunakan perkalian *Galois Field* (2^8).

12	D2	C9	D9
B2	00	16	FA
9C	05	23	E3
83	DA	65	15

12	X	02	03	01	01	=	r0
B2		01	02	03	01		r1
9C		01	01	02	03		r2
83		03	01	01	02		r3

Proses mencari nilai *byte* pertama pada kolom pertama dilakukan perkalian antara kolom pertama *state* dengan baris pertama tabel.

Proses mencari nilai *byte* kedua pada kolom pertama dilakukan perkalian antara kolom pertama dengan baris kedua. Proses selanjutnya dilakukan dengan cara yang sama sesuai urutan *byte* dan baris tabel. Berikut proses hitungan *MixColumns* :

$$r0 = (12*02) \text{ XOR } (B2*03) \text{ XOR } (9C*01) \text{ XOR } (83*01)$$

$$r1 = (12*01) \text{ XOR } (B2*02) \text{ XOR } (9C*03) \text{ XOR } (83*01)$$

$$r2 = (12*01) \text{ XOR } (B2*01) \text{ XOR } (9C*02) \text{ XOR } (83*03)$$

$$r3 = (12*03) \text{ XOR } (B2*01) \text{ XOR } (9C*01) \text{ XOR } (83*02)$$

Proses perkalian didalam bentuk biner dalam *Galois Field*

$$r0 = (12*02) \text{ XOR } (B2*03) \text{ XOR } (9C*01) \text{ XOR } (83*01)$$

{12}*{02} = 00010010 >> 12 (Hexadesimal)
10 >> 02 (Desimal)

$$\begin{array}{r} \text{00010010} \\ \text{00000000} \\ \hline \text{00010010} \end{array} \begin{array}{l} * \\ \\ \text{XOR} \end{array}$$

00100100 >> Nilai lebih kecil dari 0xFF sehingga tidak dilakukan

XOR dengan 0x11B.

{B2}*{03} = 10110010 >> AD (Hexadesimal)
11 >> 03 (Desimal)

$$\begin{array}{r} \text{10110010} \\ \text{10110010} \\ \hline \text{111010110} \\ \text{100011011} \\ \hline \text{011001101} \end{array} \begin{array}{l} * \\ \\ \text{XOR} \\ \\ \text{XOR} \end{array}$$

111010110 >> Nilai lebih besar dari 0xFF sehingga dilakukan

XOR dengan nilai 100011011 (11B) dan didapat hasil 11001101

{9C}*{01} = 10011100 >> dikalikan dengan 1 tidak mengalami perubahan

{83}*{01} = 10000011 >> dikalikan dengan 1 tidak mengalami perubahan

Kemudian dilakukan XOR hasil dari semua perkalian *byte* kolom untuk mendapatkan nilai *byte* pertama pada kolom kedua.

$$\begin{array}{r} \text{00100100} \\ \text{11001101} \\ \text{10011100} \\ \text{10000011} \\ \hline \text{11101110} \end{array} \text{XOR}$$

11101110 >> dalam hexadesimal F6

Sehingga di dapat nilai *byte* pertama kolom kedua *state* seperti di bawah ini:

F6

Proses selanjutnya untuk mendapatkan nilai pada *byte* kedua (*r1*) kolom pertama dilakukan dengan cara yang sama yaitu mengalikan kolom pertama *state* dengan baris kedua tabel dan seterusnya, sehingga didapat nilai kolom pertama keseluruhan setelah *MixColumns* sebagai berikut :

F6
51
1D
05

Selanjutnya mencari nilai *MixColumns* untuk kolom kedua *state*. Proses pencarian sama dengan cara sebelumnya, yaitu untuk mencari nilai *byte* pertama pada kolom kedua *state* dikalikan dengan baris pertama tabel dan seterusnya.

D2	X	02	03	01	01	=	r0
00		01	02	03	01		r1
05		01	01	02	03		r2
DA		03	01	01	02		r3

$$r0 = (D2*02) \text{ XOR } (00*03) \text{ XOR } (05*01) \text{ XOR } (DA*01)$$

$$r1 = (D2*01) \text{ XOR } (00*02) \text{ XOR } (05*03) \text{ XOR } (DA*01)$$

$$r2 = (D2*01) \text{ XOR } (00*01) \text{ XOR } (05*02) \text{ XOR } (DA*03)$$

$$r3 = (D2*03) \text{ XOR } (00*01) \text{ XOR } (05*01) \text{ XOR } (DA*02)$$

Proses perkalian didalam bentuk biner dalam *Galois Field*

$$r_0 = (D2*02) \text{ XOR } (00*03) \text{ XOR } (05*01) \text{ XOR } (DA*01)$$

$$\{D2\}*\{02\} = 11010010 \gg D2 \text{ (Hexadesimal)}$$

$$10 \gg 02 \text{ (Desimal)}$$

$$\begin{array}{r} \text{-----}^* \\ 10101101 \\ 00000000 \\ \text{-----} \\ \text{-----} \text{XOR} \\ 101011010 \\ 100011011 \\ \text{-----} \\ \text{-----} \text{XOR} \\ 010111111 \end{array}$$

101011010 >> Nilai lebih besar dari 0xFF sehingga dilakukan XOR dengan nilai 100011011 (0x11B) dan didapat hasil 10111111

$$\{00\}*\{03\} = 00000000 \gg 00 \text{ (Hexadesimal)}$$

$$11 \gg 03 \text{ (Desimal)}$$

$$\begin{array}{r} \text{-----}^* \\ 00000000 \\ 00000000 \\ \text{-----} \\ \text{-----} \text{XOR} \\ 00000000 \end{array}$$

Setiap nilai yang dikalikan dengan 0 maka hasilnya tetap 0.

$\{05\}*\{01\} = 00000101 \gg$ dikalikan dengan 1 tidak mengalami perubahan

$\{DA\}*\{01\} = 11011010 \gg$ dikalikan dengan 1 tidak mengalami perubahan

Kemudian dilakukan XOR hasil dari semua perkalian *byte* kolom untuk mendapatkan nilai *byte* pertama pada kolom kedua.

$$\begin{array}{r} 10111111 \\ 00000000 \\ 00000101 \\ 11011010 \\ \text{-----} \\ \text{-----} \text{XOR} \end{array}$$

01100000 >> dalam hexadesimal 60

Sehingga di dapat nilai *byte* pertama kolom kedua *state* seperti di bawah ini:

60

Proses selanjutnya untuk mendapatkan nilai pada *byte* kedua (*r1*) kolom kedua dilakukan dengan cara yang sama yaitu mengalikan kolom kedua *state* dengan baris kedua tabel dan seterusnya, sehingga didapat nilai kolom kedua keseluruhan setelah *MixColumns* sebagai berikut :

F6	60
51	07
1D	AD
05	C7

Selanjutnya mencari nilai *MixColumns* untuk kolom ketiga *state*. Proses pencarian sama dengan cara sebelumnya, yaitu untuk mencari nilai *byte* pertama pada kolom ketiga *state* dikalikan dengan baris pertama tabel dan seterusnya.

C9	X	02	03	01	01	=	r0
16		01	02	03	01		r1
23		01	01	02	03		r2
65		03	01	01	02		r3

$$r0 = (C9*02) \text{ XOR } (16*03) \text{ XOR } (23*01) \text{ XOR } (65*01)$$

$$r1 = (C9*01) \text{ XOR } (16*02) \text{ XOR } (23*03) \text{ XOR } (65*01)$$

$$r2 = (C9*01) \text{ XOR } (16*01) \text{ XOR } (23*02) \text{ XOR } (65*03)$$

$$r3 = (C9*03) \text{ XOR } (16*01) \text{ XOR } (23*01) \text{ XOR } (65*02)$$

Proses perkalian didalam bentuk biner dalam *Galois Field*.

$$r0 = (C9*02) \text{ XOR } (16*03) \text{ XOR } (23*01) \text{ XOR } (65*01)$$

$$\{C9\}*\{02\} = 11001001 \gg C9 \text{ (Hexadesimal)}$$

$$\begin{array}{r}
 10 \gg 02 \text{ (Desimal)} \\
 \hline
 11001001 \\
 00000000 \\
 \hline
 \text{XOR} \\
 110010010 \\
 100011011 \\
 \hline
 \text{XOR} \\
 010001001
 \end{array}$$

110010010 \gg Nilai lebih besar dari 0xFF sehingga dilakukan XOR dengan nilai 100011011 (0x11B) dan didapat hasil 010001001.

$$\{16\}*\{03\} = 00010110 \gg 16 \text{ (Hexadesimal)}$$

$$\begin{array}{r}
 11 \gg 03 \text{ (Desimal)} \\
 \hline
 00010110 \\
 00010110 \\
 \hline
 \text{XOR} \\
 000111010
 \end{array}$$

00010110 \gg Nilai lebih kecil dari 0xFF sehingga tidak dilakukan XOR dengan 0x11B.

$\{23\}*\{01\} = 00100011 \gg$ dikalikan dengan 1 tidak mengalami perubahan

$\{65\}*\{01\} = 01100101 \gg$ dikalikan dengan 1 tidak mengalami perubahan

Kemudian dilakukan XOR hasil dari semua perkalian *byte* kolom untuk mendapatkan nilai *byte* pertama pada kolom ketiga.

```

10001001
00111010
00100011
01100101
-----XOR
11110101 >> dalam hexadesimal F5

```

Sehingga di dapat nilai *byte* pertama kolom ketiga state seperti di bawah ini:

F5

Proses selanjutnya untuk mendapatkan nilai pada *byte* kedua (*r1*) kolom ketiga dilakukan dengan cara yang sama yaitu mengalikan kolom ketiga state dengan baris kedua tabel dan seterusnya, sehingga didapat nilai kolom ketiga keseluruhan setelah *MixColumns* sebagai berikut :

F6	60	F5
51	07	E5
1D	AD	36
05	C7	BF

Selanjutnya mencari nilai *MixColumns* untuk kolom keempat *state*. Proses pencarian sama dengan cara sebelumnya, yaitu

untuk mencari nilai *byte* pertama pada kolom keempat *state* dikalikan dengan baris pertama tabel dan seterusnya.

D9	X	02	03	01	01	=	r0
FA		01	02	03	01		r1
E3		01	01	02	03		r2
15		03	01	01	02		r3

$$r0 = (D9*02) \text{ XOR } (FA*03) \text{ XOR } (E3*01) \text{ XOR } (15*01)$$

$$r1 = (D9*01) \text{ XOR } (FA*02) \text{ XOR } (E3*03) \text{ XOR } (15*01)$$

$$r2 = (D9*01) \text{ XOR } (FA*01) \text{ XOR } (E3*02) \text{ XOR } (15*03)$$

$$r3 = (D9*03) \text{ XOR } (FA*01) \text{ XOR } (E3*01) \text{ XOR } (15*02)$$

Proses perkalian didalam bentuk biner dalam *Galois Field*

$$r0 = (D9*02) \text{ XOR } (FA*03) \text{ XOR } (E3*01) \text{ XOR } (15*01)$$

$$\{D9\} * \{02\} = 11011001 \gg D9 \text{ (Hexadesimal)}$$

$$10 \gg 02 \text{ (Desimal)}$$

$$\begin{array}{r} \text{-----} \\ 11011001 \\ 00000000 \\ \text{-----} \\ \text{XOR} \\ 110110010 \\ 100011011 \\ \text{-----} \\ \text{XOR} \\ 010101001 \end{array}$$

110110010 >> Nilai lebih besar dari 0xFF sehingga dilakukan

XOR dengan nilai 100011011 (0x11B) dan didapat hasil

010101001.

$$\{FA\} * \{03\} = 11111010 \gg FA \text{ (Hexadesimal)}$$

$$10 \gg 03 \text{ (Desimal)}$$

$$\begin{array}{r} \text{-----} \\ 11111010 \\ 11111010 \\ \text{-----} \\ \text{XOR} \end{array}$$

$$\begin{array}{r} 100001110 \\ 100011011 \\ \hline \text{XOR} \end{array}$$

000010101

100001110 >> Nilai lebih besar dari 0xFF sehingga dilakukan XOR dengan nilai 100011011 (0x11B) dan didapat hasil 000010101.

{E3}*{01} = 11100011 >> dikalikan dengan 1 tidak mengalami perubahan

{15}*{01} = 00010101 >> dikalikan dengan 1 tidak mengalami perubahan

Kemudian dilakukan XOR hasil dari semua perkalian *byte* kolom untuk mendapatkan nilai *byte* pertama pada kolom keempat.

$$\begin{array}{r} 10101001 \\ 00010101 \\ 11100011 \\ 00010101 \\ \hline \text{XOR} \end{array}$$

01001010 >> dalam hexadesimal 4A

Sehingga di dapat nilai *byte* pertama kolom keempat state seperti di bawah ini:

4A

Proses selanjutnya untuk mendapatkan nilai pada *byte* kedua (r1) kolom keempat dilakukan dengan cara yang sama yaitu mengalikan kolom keempat *state* dengan baris kedua tabel dan

seterusnya, sehingga didapat nilai kolom keempat keseluruhan setelah *MixColumns* sebagai berikut :

F6	60	F5	4A
51	07	E5	1D
1D	AD	36	C1
05	C7	BF	43

d) *AddRoundKey*

Proses ini adalah melakukan XOR nilai state sekarang dengan kunci ronde berikutnya yang didapat dari proses pembangkitan kunci sebelumnya. Adapun nilai *state* sekarang dan kunci ronde berikutnya sebagai berikut:

<i>State</i> sekarang					Kunci ronde 1			
F6	60	F5	4A	XOR	B9	D7	B0	D7
51	07	E5	1D		86	ED	9A	F3
1D	AD	36	C1		61	14	70	11
05	C7	BF	43		B7	87	B6	8F

Selanjutnya adalah melakukan proses *AddRoundKey* dengan meng XOR setiap kolom *state* sekarang dengan setiap kolom nilai kunci ronde 1 seperti di bawah ini :

F6	XOR	B9
51		86
1D		61
05		B7

Lakukan XOR pada setiap *byte* kolom state dan *byte* kolom kunci dengan merubah nilai hexadesimal kedalam biner seperti di bawah ini:

$$\begin{array}{r} F6 = 11110110 \\ B9 = 10111001 \\ \hline \text{XOR} \\ 01001111 \text{ dalam hexadesimal adalah } 4F \end{array}$$

$$\begin{array}{r} 51 = 01010001 \\ 86 = 10000110 \\ \hline \text{XOR} \\ 11010111 \text{ dalam hexadesimal adalah } D7 \end{array}$$

$$\begin{array}{r} 1D = 00011101 \\ 61 = 01100001 \\ \hline \text{XOR} \\ 01111100 \text{ dalam hexadesimal adalah } 7C \end{array}$$

$$\begin{array}{r} 05 = 00000101 \\ B7 = 10110111 \\ \hline \text{XOR} \\ 10110010 \text{ dalam hexadesimal adalah } B2 \end{array}$$

Sehingga didapat kolom pertama hasil *AddRoundKey* sebagai di bawah ini :

4F
D7
7C
B2

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua *state* di XOR dengan kolom kedua kunci dan seterusnya. Sehingga didapat hasil *AddroundKey* ronde pertama sebagai berikut :

4F	B7	45	9D
D7	EA	7F	EE
7C	B9	46	D0
B2	40	09	CC

State di atas akan digunakan untuk ronde kedua dan begitu

seterusnya. Proses yang sama dilakukan untuk enkripsi ronde ke 2 hingga ronde ke 9.

3) Enkripsi Ronde 10

Setelah dilakukan iterasi dari ronde 1 sampai dengan ronde 9, maka masuk ke ronde terakhir. Proses pada ronde terakhir hanya dilakukan 3 proses dengan melewati proses *MixColumns* yaitu, *SubBytes*, *ShiftRows* dan *AddRoundkey*.

a) *SubBytes*

Proses ini adalah melakukan substitusi dengan menukar nilai setiap *byte* state dengan nilai *byte* pada gambar table S-Box 2.1.

6D	52	24	65
5B	C2	8C	93
B6	C3	47	E1
90	B7	D3	5B

Misal pada kolom pertama *byte* pertama adalah 6D, maka elemen yang sesuai dengan gambar nilai gambar tabel S-Box 2.1 pada bab landasan teori terletak pada baris ke-6 dan kolom ke-D yaitu 3C. Langkah yang sama dilakukan sehingga didapat hasil setelah proses *SubBytes* dengan tabel S-Box sebagai berikut :

3C	00	36	4D
39	25	64	DC
4E	2E	A0	F8
60	A9	66	39

b) *ShiftRows*

Proses ini melakukan pergeseran pada setiap elemen tabel *array*, yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte* ke kiri, baris ketiga dilakukan pergeseran 2 *byte* ke kiri, baris ketga dilakukan pergeseran 3 *byte* ke kiri.

Sebelum *ShiftRows*

Setelah *ShiftRows*

3C	00	36	4D		3C	00	36	4D
39	25	64	DC		25	64	DC	39
4E	2E	A0	F8		A0	F8	4E	2E
60	A9	66	39		39	60	A9	66

c) *AddRoundKey*

Proses ini adalah melakukan XOR nilai *state* sekarang dengan kunci ronde terakhir yang didapat dari proses pembangkitan kunci. Adapun nilai *state* sekarang dan kunci ronde terakhir sebagai berikut:

State sekarang					Kunci ronde 10			
3C	00	36	4D	XOR	24	56	48	58
25	64	DC	39		65	05	70	67
A0	F8	4E	2E		5D	5C	BF	E8
39	60	A9	66		D8	B1	FA	FF

Selanjutnya adalah melakukan proses *AddRoundKey* dengan meng XOR setiap kolom *state* sekarang dengan setiap kolom nilai kunci ronde 2 seperti di bawah ini :

3C	XOR	24
25		65
A0		5D
39		D8

Lakukan XOR pada setiap *byte* kolom state dan *byte* kolom kunci dengan merubah nilai hexadesimal kedalam biner seperti di bawah ini:

$$3C = 00111100$$

$$24 = 00100100$$

$$\underline{\hspace{1.5cm}} \text{ XOR}$$

00011000 dalam hexadesimal adalah **18**

$$25 = 00100101$$

$$65 = 01100101$$

$$\underline{\hspace{1.5cm}} \text{ XOR}$$

01000000 dalam hexadesimal adalah **40**

$$A0 = 10100000$$

$$5D = 01011101$$

$$\underline{\hspace{1.5cm}} \text{ XOR}$$

11111101 dalam hexadesimal adalah **FD**

39 = 00111001

D8 = 11011000

_____ XOR

11100001 dalam hexadesimal adalah E1

Sehingga didapat kolom pertama hasil *AddRoundKey* sebagai di bawah ini :

18
40
FD
E1

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua *state* di XOR dengan kolom kedua kunci dan seterusnya. Sehingga didapat hasil *AddroundKey* ronde pertama sebagai berikut :

18	56	7E	15
40	61	AC	5E
FD	A4	F1	C6
E1	D1	53	99

Hasil di atas adalah *cipherimage* AES untuk enkripsi pertama.

Adapun proses enkripsi keseluruhan ronde dapat dilihat pada gambar 4.20 Berikut :

	SubBytes				ShiftRows				MixColumns				AddRoundKey				Key Schedule							
Round 0																	39	7F	12	E5	41	6E	67	67
																	14	3E	52	FF	69	6B	77	69
																	32	4D	1C	36	73	75	64	61
																	7A	BC	2F	41	32	30	31	39
Round 1	12	D2	C9	D9	12	D2	C9	D9	F6	60	F5	4A	4F	B7	45	9D	B9	D7	B0	D7				
	FA	B2	00	16	B2	00	16	FA	51	07	E5	1D	D7	EA	7F	EE	86	ED	9A	F3				
	23	E3	9C	05	9C	05	23	E3	1D	AD	36	C1	7C	B9	46	D0	61	14	70	11				
	DA	65	15	83	83	DA	65	15	05	C7	BF	43	B2	40	09	CC	B7	87	B6	8F				
Round 2	84	A9	6E	5E	84	A9	6E	5E	90	63	BD	F9	26	02	6C	FF	B6	61	D1	06				
	0E	87	D2	28	87	D2	28	0E	34	B1	07	B9	30	58	74	39	04	E9	73	80				
	10	56	5A	70	5A	70	10	56	6A	C2	7D	FF	78	C4	0B	98	12	06	76	67				
	37	09	01	4B	4B	37	09	01	DC	2C	98	B8	65	12	10	BF	B9	3E	88	07				
Round 3	F7	77	50	16	F7	77	50	16	68	48	E3	F6	17	56	2C	3F	7F	1E	CF	C9				
	04	6A	92	12	6A	92	12	04	56	CF	62	F0	D7	A7	79	6B	81	68	1B	9B				
	BC	1C	2B	46	2B	46	BC	1C	D3	BE	61	6F	04	6F	C6	AF	D7	D1	A7	C0				
	4D	C9	CA	08	08	4D	C9	CA	53	D7	D7	AD	85	3F	B7	CA	D6	E8	60	67				
Round 4	F0	B1	71	75	F0	B1	71	75	DF	56	E4	F9	BC	2B	56	82	63	7D	B2	7B				
	0E	5C	B6	7F	5C	B6	7F	0E	FB	DA	F7	23	C0	89	BF	F0	3B	53	48	D3				
	F2	A8	B4	79	B4	79	F2	A8	43	57	6E	D0	11	D4	4A	34	52	83	24	E4				
	97	75	A9	74	74	97	75	A9	0B	32	F4	70	00	D1	77	94	0B	E3	83	E4				
Round 5	65	F1	B1	13	65	F1	B1	13	CC	9A	4A	4E	D9	F2	90	EF	15	68	DA	A1				
	BA	A7	08	8C	A7	08	8C	BA	73	AA	11	51	21	AB	58	CB	52	01	49	9A				
	82	48	D6	18	D6	18	82	48	13	6C	60	3D	28	D4	FC	45	3E	B8	9C	78				
	63	3E	F5	22	22	63	3E	F5	9A	DE	BA	36	B0	17	F0	98	2A	C9	4A	AE				
Round 6	35	89	60	DF	35	89	60	DF	3A	3E	25	7D	B7	DB	1A	E3	8D	E5	3F	9E				
	FD	62	6A	1F	62	6A	1F	FD	7C	08	F2	6A	92	E7	54	56	EE	EF	A6	3C				
	34	48	B0	6E	B0	6E	34	48	E6	0D	1C	3D	39	6A	E7	BE	DF	67	FB	83				
	E7	F0	8C	46	46	E7	F0	8C	01	51	70	CC	19	80	EB	F9	18	D1	9B	35				
Round 7	A9	B9	A2	11	A9	B9	A2	11	E3	73	48	18	C5	B0	B4	7A	26	C3	FC	62				
	4F	94	20	B1	94	20	B1	4F	A4	C4	20	60	A6	29	6B	17	02	ED	4B	77				
	12	02	94	AE	94	AE	12	02	BE	B9	7B	7A	F7	97	AE	2C	49	2E	D5	56				
	D4	CD	E9	99	99	D4	CD	E9	C9	ED	DF	B7	DA	2F	86	DB	13	C2	59	6C				
Round 8	A6	E7	8D	DA	A6	E7	8D	DA	FE	72	77	0F	AD	E2	1B	01	53	90	6C	0E				
	24	A5	7F	F0	A5	7F	F0	24	79	DD	DB	55	CA	83	CE	37	B3	5E	15	62				
	68	88	E4	71	E4	71	68	88	00	83	92	39	19	B4	70	8D	19	37	E2	B4				
	57	15	44	B9	B9	57	15	44	D9	92	3E	51	60	E9	1C	1F	B9	7B	22	4E				
Round 9	95	98	AF	7C	95	98	AF	7C	8F	20	3A	75	6D	52	24	65	E2	72	1E	10				
	74	EC	8B	9A	EC	8B	9A	74	65	A2	F9	84	5B	C2	8C	93	3E	60	75	17				
	D4	8D	51	5D	51	5D	D4	8D	80	C2	A4	B6	B6	C3	47	E1	36	01	E3	57				
	D0	1E	9C	C0	C0	D0	1E	9C	82	DE	98	5E	90	B7	D3	5B	12	69	4B	05				
Round 10	3C	00	36	4D	3C	00	36	4D					18	56	7E	15	24	56	48	58				
	39	25	64	DC	25	64	DC	39					40	61	AC	5E	65	05	70	67				
	4E	2E	A0	F8	A0	F8	4E	2E					FD	A4	F1	C6	5D	5C	BF	E8				
	60	A9	66	39	39	60	A9	66					E1	D1	53	99	D8	B1	FA	FF				

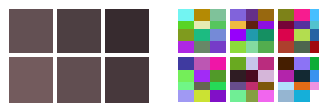
Gambar 4.21 Proses Enkripsi AES Ronde 1 sampai 10

Berikut tabel perbandingan nilai desimal *cipherimage* AES pada setiap *pixel* yang telah dienkripsi dengan algoritma AES :

Tabel 4.4 Perbandingan Nilai *Pixel* Setelah Enkripsi AES

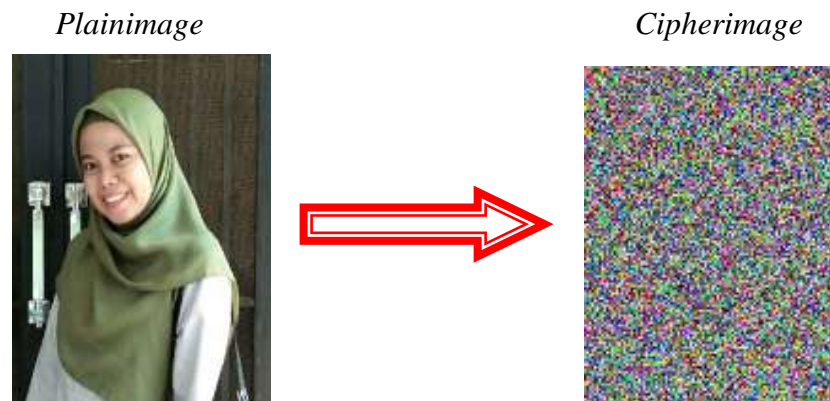
Pixel	Warna	Plainimage		Cipherimage AES	
		Desimal	Hexa	Desimal	Hexa
1	R	120	78	24	18
	G	17	11	86	56
	B	117	75	126	7E
2	R	130	82	21	15
	G	125	7D	64	40
	B	85	55	97	61
3	R	37	25	172	AC
	G	150	96	94	5E
	B	65	41	253	FD
4	R	56	38	164	A4
	G	120	78	241	F1
	B	87	57	198	C6
5	R	75	48	225	E1
	G	140	8C	209	D1
	B	30	1E	83	53
6	R	120	78	153	99
	G	25	19	25	19
	B	10	A	10	A

Berdasarkan pada tabel 4.4 di atas, pada *pixel* 6 nilai GB tidak mengalami perubahan dikarenakan AES 128 mengenkripsi blok 128 bit pertama sehingga didalam hitungan manual ini nilai GB pada *pixel* 6 tidak diikuti dalam enkripsi manual. Hasil perbandingan perubahan *plainimage* yang di enkripsi menggunakan algoritma AES 128-bit.



Gambar 4.22 *Pixel Cipherimage* Sampel

Hasil perbandingan perubahan *plainimage* yang di enkripsi menggunakan algoritma AES 128 *bit* adalah sebagai berikut :



2. Proses Pengiriman *Chiperimage*

Proses pengiriman dilakukan dengan memasukan alamat *ip address* komputer penerima. Adapun pada contoh kasus ini proses pengiriman dilakukan dengan jalur *wireless* (nirkabel) menggunakan sinyal *wifi*.

Gambar 4.23 Tampilan *Input IP Address* Penerima

Berdasarkan pada proses koneksi sebelumnya, diketahui bahwa *ip address* komputer penerima adalah 192.168.43.41. Sehingga untuk mengirimkan pesan gambar *cipherimage*, maka pengirim (*server*) harus memasukan alamat *ip address* pada kolom yang sudah disediakan aplikasi yaitu 192.168.43.41. Selanjutnya memilih tombol KIRIM CIPHERIMAGE. Jika pengiriman berhasil dan diterima maka akan menampilkan pesan seperti gambar berikut.



Gambar 4.24 Tampilan File Berhasil Dikirim

Jika ditolak oleh penerima maka akan menampilkan pesan seperti gambar di bawah ini :



Gambar 4.25 Tampilan Akses Tolak

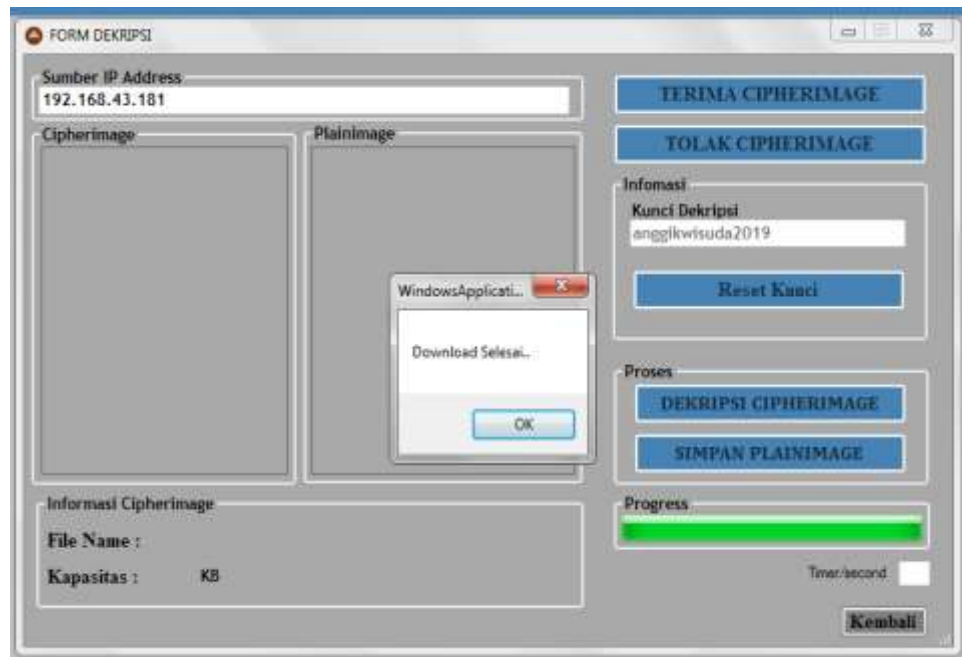
3. Proses Dekripsi

Form dekripsi pada aplikasi akan tampil jika ada pengiriman pesan masuk di menu utama sistem aplikasi. Pada kasus ini komputer *server* (pengirim) memiliki *ip address* 192.168.43.181. Sedangkan penerima (*client*) memiliki

ip address pada komputernya 192.168.43.41. User penerima (*client*) berhak menentukan menerima atau menolak pesan. Jika *user* menolak maka akan kembali pada menu utama. Sedangkan jika *user (client)* menerima pesan maka akan menampilkan *form* dekripsi seperti gambar di bawah ini :

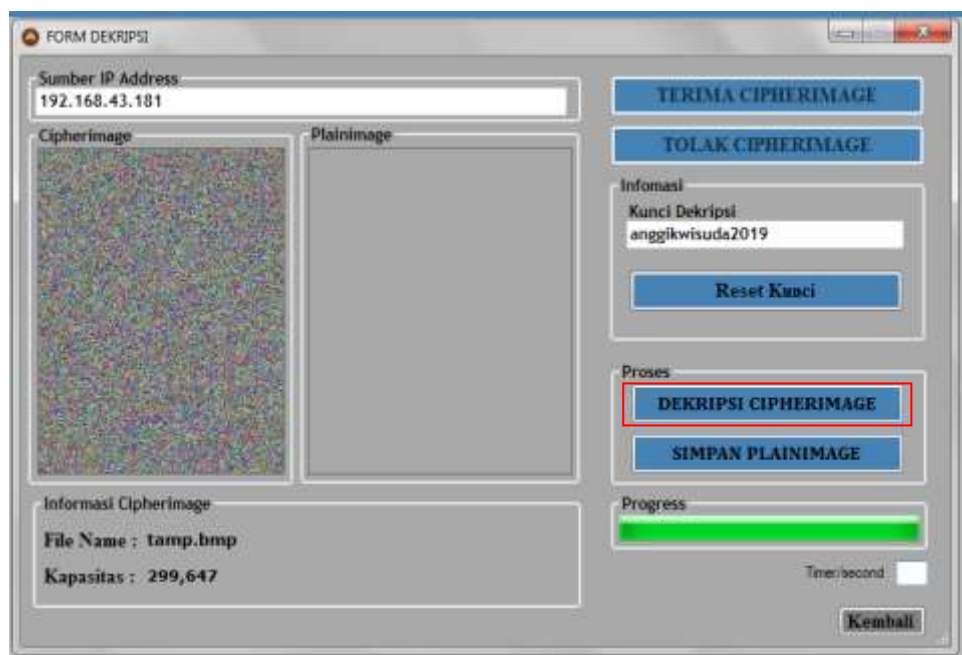
Gambar 4.26 Tampilan *Form* Dekripsi Pemberitahuan Pesan Masuk

Form dekripsi akan tampil jika ada pengiriman pesan masuk seperti pada gambar di atas. Jika *user (client)* menerima dengan memilih tombol TERIMA CIPHERIMAGE maka masuk proses *downloading chiperimage* seperti gambar di bawah ini :



Gambar 4.27 Tampilan Proses *Downloading Cipherimage*

Berdasarkan pada gambar 4.26 diatas, proses selanjutnya dapat dilihat pada gambar di bawah ini :



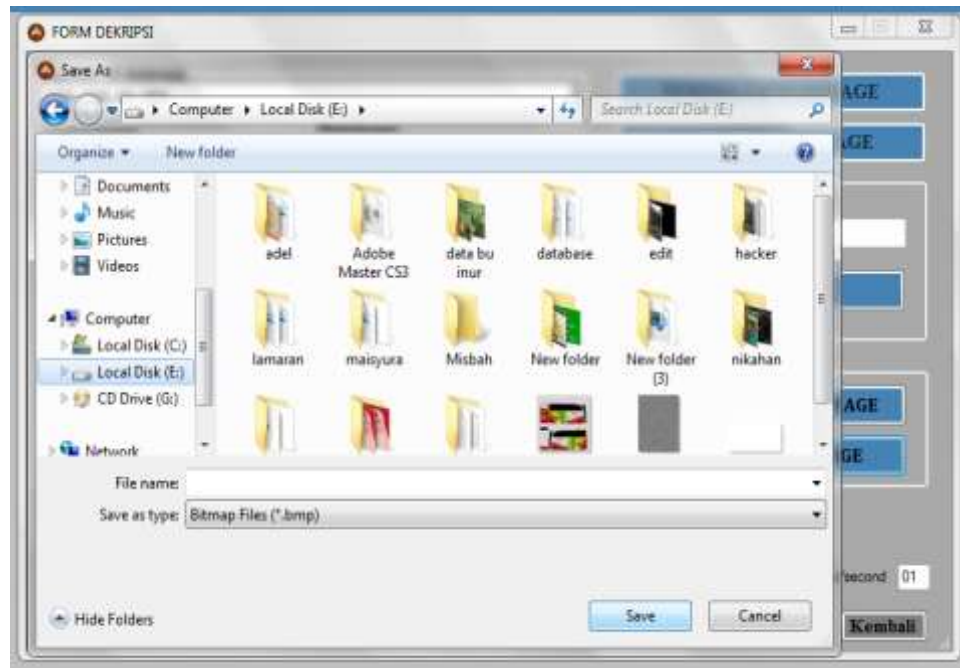
Gambar 4.28 Tampilan *Form Dekripsi Cipherimage*

Hasil *downloading* setelah proses penerimaan pesan dilakukan berupa *cipherimage* yang akan didekripsi oleh *user client*. Proses dekripsi dimulai dengan memilih *stripbutton* DEKRIPSI CIPHERIMAGE. Dekripsi dilakukan dengan algoritma AES 128-bit sehingga mendapatkan hasil berupa *plainimage* seperti gambar di bawah ini :



Gambar 4.29 Tampilan *Form* Hasil Dekripsi

Berdasarkan pada gambar di atas, proses dekripsi berhasil mengembalikan citra digital *cipherimage* ke *plainimage*. *Plainimage* mengalami perubahan ekstansi menjadi *bmp* (bitmap) serta mengalami perubahan ukuran kapasitasnya. *Plainimage* tersebut dapat disimpan kembali ke dalam *directory* komputer dengan memilih *stripbutton* SIMPAN PLAINIMAGE seperti gambar di bawah ini :



Gambar 4.30 Proses Penyimpanan *Plainimage* Hasil Dekripsi

a. Perhitungan Manual Dekripsi AES 128-bit

Proses dekripsi adalah kebalikan dari proses enkripsi. Kunci yang digunakan untuk dekripsi *cipherimage* AES sama dengan kunci saat enkripsi yaitu *anggikwisuda2019*. Rubah *cipherimage* AES biner ke nilai hexadesimal serta *input* ke dalam *array* 4x4. Rubah nilai karakter kunci ke dalam hexadesimal serta *input* dedalam *array* 4x4. Adapun hasilnya sebagai berikut:

<i>Cipherimage</i> AES					Kunci Hexadesimal AES			
18	56	7E	15	XOR	41	6E	67	67
40	61	AC	5E		69	6B	77	69
FD	A4	F1	C6		73	75	64	61
E1	D1	53	99		32	30	31	39

Sebelum melakukan proses dekripsi akan dilakukan pembangkitan kunci dilakukan dengan cara yang sama saat proses enkripsi sehingga menghasilkan nilai kunci yang sama seperti gambar 4.20 di atas. Perbedaan disini adalah penggunaan kunci saat proses dekripsi dimulai dari kunci ronde 10 hingga ronde 1.

1) Dekripsi Ronde 10

Proses dekripsi adalah kebalikan proses enkripsi setiap ronde. Ronde ke 10 memiliki 3 proses yaitu *AddRoundKey*, *InvShiftRows* dan *InvSubBytes* sedangkan pada ronde ke 9 hingga ke 1 memiliki 4 proses yaitu *AddRoundKey*, *InvMixColumns*, *InvShiftRow* dan *InvSubBytes* lalu dilanjutkan dengan proses *initial round*.

a) *AddRoundKey*

Proses ini adalah melakukan XOR nilai *state* sekarang dengan kunci ronde terakhir yang didapat dari proses pembangkitan kunci pada gambar 3.8. Adapun nilai *state* sekarang dan kunci ronde terakhir sebagai berikut:

State Sekarang					Kunci Ronde 10			
18	56	7E	15	XOR	24	56	48	58
40	61	AC	5E		65	05	70	67
FD	A4	F1	C6		5D	5C	BF	E8
E1	D1	53	99		D8	B1	FA	FF

Selanjutnya adalah melakukan proses *AddRoundKey* dengan meng XOR setiap kolom *state* sekarang dengan setiap kolom nilai kunci ronde 10 seperti di bawah ini :

18	XOR	24
40		65
FD		5D
E1		D8

Lakukan XOR pada setiap *byte* kolom state dan *byte* kolom kunci dengan merubah nilai hexadesimal kedalam biner seperti di bawah ini:

$$18 = 00011000$$

$$24 = 00100100$$

$$\underline{\hspace{2cm}} \text{ XOR}$$

00111100 dalam hexadesimal adalah **3C**

$$40 = 01000000$$

$$65 = 01100101$$

$$\underline{\hspace{2cm}} \text{ XOR}$$

00100101 dalam hexadesimal adalah **25**

$$FD = 11111101$$

$$5D = 01011101$$

$$\underline{\hspace{2cm}} \text{ XOR}$$

10100000 dalam hexadesimal adalah **A0**

$$E1 = 11100001$$

$$D8 = 11011000$$

$$\underline{\hspace{2cm}} \text{ XOR}$$

00111001 dalam hexadesimal adalah **39**

Sehingga didapat kolom pertama hasil *AddRoundKey* sebagai di bawah ini :

3C
25
A0
39

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua *state* di XOR dengan kolom kedua kunci ronde dan seterusnya. Sehingga didapat hasil *AddroundKey* ronde 10 sebagai berikut :

3C	00	36	4D
25	64	DC	39
A0	F8	4E	2E
39	60	A9	66

b) *InvShiftRows*

Proses ini adalah kebalik proses *ShiftRows* saat enkripsi. Melakukan pergeseran pada setiap elemen tabel *array*, yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte* terakhir ke kanan, baris ketiga dilakukan pergeseran 2 *byte* terakhir ke kanan, baris ketiga dilakukan pergeseran 3 *byte* terakhir ke kanan.

Sebelum *InvShiftRows*

Setelah *InvShiftRows*

3C	00	36	4D		3C	00	36	4D
25	64	DC	39		39	25	64	DC
A0	F8	4E	2E		4E	2E	A0	F8
39	60	A9	66		60	A9	66	39

c) *InvSubBytes*

Proses ini adalah melakukan substitusi dengan menukar nilai setiap *byte state* dengan nilai *byte* yang sesuai pada gambar tabel *Inverse S-Box*. Gambar tabel *Inverse S-Box* dapat di lihat pada bab landasan teori gambar 2.7.

3C	00	36	4D
39	25	64	DC
4E	2E	A0	F8
60	A9	66	39

Misal pada kolom pertama *byte* pertama adalah 3C, maka elemen yang sesuai dengan tabel *Inverse S-Box* terletak pada baris ke-3 dan kolom ke-C yaitu 6D. Cara yang sama dilakukan sehingga didapat hasil *InvSubBytes* dengan tabel *Inverse S-Box* sebagai berikut :

6D	52	24	65
5B	C2	8C	93
B6	C3	47	E1
90	B7	D3	5B

2) Dekripsi Ronde 1

Proses dekripsi ronde ke 1 memiliki 4 proses yaitu *AddRoundKey*, *InvMixColumns*, *InvShiftRow* dan *InvSubBytes*.

a) *AddRoundKey*

Proses ini adalah melakukan XOR nilai *state* sekarang dengan kunci ronde ke 9 yang didapat dari proses pembangkitan kunci. Berdasarkan pada proses dekripsi ronde sebelumnya didapat hasil *state* sebagai berikut:

State Sekarang					Kunci Ronde 1			
4F	B7	45	9D	XOR	B9	D7	B0	D7
D7	EA	7F	EE		86	ED	9A	F3
7C	B9	46	D0		61	14	70	11
B2	40	09	CC		B7	87	B6	8F

Selanjutnya adalah melakukan proses *AddRoundKey* dengan meng XOR setiap kolom *state* sekarang dengan setiap kolom nilai kunci ronde 1 seperti di bawah ini :

4F	XOR	B9
D7		86
7C		61
B2		B7

Lakukan XOR pada setiap *byte* kolom *state* dan *byte* kolom kunci dengan merubah nilai hexadesimal kedalam biner seperti di bawah ini:

4F = 01001111

B9 = 10111001

_____ XOR

11110110 dalam hexadesimal adalah F6

D7 = 11010111

86 = 10000110

_____XOR

01010001 dalam hexadesimal adalah 51

7C = 01111100

61 = 01100001

_____XOR

00011101 dalam hexadesimal adalah 1D

B2 = 10110111

B7 = 10110010

_____XOR

00000101 dalam hexadesimal adalah 05

Sehingga didapat kolom pertama hasil *AddRoundKey* sebagai di bawah ini :

F6
51
1D
05

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua *state* di XOR dengan kolom kedua kunci ronde dan seterusnya. Sehingga didapat hasil *AddroundKey* ronde 1 sebagai berikut :

F6	60	F5	4A
51	07	E5	1D
1D	AD	36	C1
05	C7	BF	43

b) *InvMixColumns*

Proses *InvMixColumns* adalah proses perkalian setiap kolom *state* dan kolom matriks tabel *InvMxColumns*. Tabel matriks *InvMixColumns* dapat di lihat pada gambar 2.12 bab landasan teori. Perkalian ini menggunakan perkalian *Galois Field*. Apabila nilai lebih besar dari 0xFF dalam desimal 255 maka dilakukan XOR dengan 0x11B.

F6	60	F5	4A
51	07	E5	1D
1D	AD	36	C1
05	C7	BF	43

F6	X	14	11	13	9	=	r0
51		9	14	11	13		r1
1D		13	9	14	11		r2
05		11	13	9	14		r3

Proses mencari nilai *byte* pertama pada kolom pertama dilakukan perkalian antara kolom pertama *state* dengan baris pertama tabel. Proses mencari nilai *byte* kedua pada kolom pertama dilakukan perkalian antara kolom pertama dengan baris kedua. Proses selanjutnya dilakukan dengan cara yang sama sesuai urutan dan baris tabel. Berikut proses hitungan *InvMixColumns* :

$$r0 = (F6*14) \text{ XOR } (51*11) \text{ XOR } (1D*13) \text{ XOR } (05*9)$$

$$r1 = (F6*9) \text{ XOR } (51*14) \text{ XOR } (1D*11) \text{ XOR } (05*13)$$

$$r2 = (F6*13) \text{ XOR } (51*9) \text{ XOR } (1D*14) \text{ XOR } (05*11)$$

$$r3 = (F6*11) \text{ XOR } (51*13) \text{ XOR } (1D*9) \text{ XOR } (05*14)$$

Perkalian di dalam bentuk biner dalam *Galois Field*

$r0 = (F6*14) \text{ XOR } (51*11) \text{ XOR } (1D*13) \text{ XOR } (05*9)$

$\{F6\}*\{14\} = 11110110 \gg F6$ (Hexadesimal)
 $1110 \gg 14$ (Desimal)

$$\begin{array}{r}
 \text{-----}^* \\
 11110110 \\
 11110110 \\
 11110110 \\
 00000000 \\
 \text{-----} \\
 \text{XOR} \\
 10110000100 \\
 100011011 \\
 \text{-----} \\
 \text{XOR} \\
 00111101000 \\
 100011011 \\
 \text{-----} \\
 \text{XOR} \\
 \mathbf{011110011}
 \end{array}$$

10110000100 >> Nilai lebih besar dari 0xFF sehingga dilakukan XOR dengan 0x11B. Proses XOR dilakukan hingga nilai *output*

<=

$\{51\}*\{11\} = 01010001 \gg 51$ (Hexadesimal)
 $1011 \gg 11$ (Desimal)

$$\begin{array}{r}
 \text{-----}^* \\
 01010001 \\
 00000000 \\
 01010001 \\
 01010001 \\
 \text{-----} \\
 \text{XOR} \\
 01001111011 \\
 100011011 \\
 \text{-----} \\
 \text{XOR} \\
 \mathbf{0001001101}
 \end{array}$$

01001111011 >> Nilai lebih besar dari 0xFF sehingga dilakukan XOR dengan 0x11B. Proses XOR dilakukan hingga nilai *output*

<= 0xFF.

{1D}*{13} = 00011101 >> 1D (Hexadesimal)

$$\begin{array}{r}
 1101 \gg 13 \text{ (Desimal)} \\
 \hline
 * \\
 00011101 \\
 00011101 \\
 00000000 \\
 00011101 \\
 \hline
 \text{XOR} \\
 0001000001
 \end{array}$$

0001000001 >> Nilai lebih kecil dari 0xFF sehingga tidak dilakukan XOR dengan 0x11B.

{05}*{9} = 00000101 >> 05 (Hexadesimal)

$$\begin{array}{r}
 1001 \gg 9 \text{ (Desimal)} \\
 \hline
 * \\
 00000101 \\
 00000000 \\
 00000000 \\
 00000101 \\
 \hline
 \text{XOR} \\
 00000101101
 \end{array}$$

0001000001 >> Nilai lebih kecil dari 0xFF sehingga tidak dilakukan XOR dengan 0x11B.

Kemudian dilakukan XOR hasil dari semua perkalian *byte* kolom untuk mendapatkan nilai *byte* pertama pada kolom pertama.

11110011
01001101
10000001
00101101

_____XOR
00010010 >> dalam hexadesimal 12

Sehingga di dapat nilai *byte* pertama kolom pertama seperti di bawah ini :

12

Proses selanjutnya untuk mendapatkan nilai pada *byte* kedua (*r1*) kolom pertama dilakukan dengan cara yang sama yaitu mengalikan kolom pertama *state* dengan baris kedua tabel dan seterusnya, sehingga didapat nilai kolom keseluruhan setelah *InvMixColumns* sebagai berikut :

12	D2	C9	D9
B2	00	16	FA
9C	05	23	E3
83	DA	65	15

c) *InvShiftRows*

Proses ini adalah kebalik proses *ShiftRows* saat enkripsi. Melakukan pergeseran pada setiap elemen tabel *array*, yaitu baris pertama tidak dilakukan pergeseran, baris kedua dilakukan pergeseran 1 *byte* terakhir ke kanan, baris ketiga dilakukan pergeseran 2 *byte* terakhir ke kanan, baris ketiga dilakukan pergeseran 3 *byte* terakhir ke kanan.

Sebelum *InvShiftRows*

Setelah *InvShiftRows*

12	D2	C9	D9		12	D2	C9	D9
B2	00	16	FA		FA	B2	00	16
9C	05	23	E3		23	E3	9C	05
83	DA	65	15		DA	65	15	83

d) *InvSubBytes*

Proses ini adalah melakukan substitusi dengan menukar nilai setiap *byte state* dengan nilai *byte* yang sesuai pada gambar tabel *Inverse S-Box*. Gambar tabel *Inverse S-Box* dapat di lihat pada bab landasan teori gambar 2.7.

12	D2	C9	D9
FA	B2	00	16
23	E3	9C	05
DA	65	15	83

Misal pada kolom pertama *byte* pertama adalah 12, maka elemen yang sesuai dengan tabel *Inverse S-Box* terletak pada baris ke-1 dan kolom ke-1 yaitu 39. Cara yang sama dilakukan sehingga didapat hasil *InvSubBytes* dengan tabel *Inverse S-Box* sebagai berikut :

39	7F	12	E5
14	3E	52	FF
32	4D	1C	36
7A	BC	2F	41

3) *Initial Round*

Initial Round ini dilakukan setelah di dapat *state* akhir ronde 1. Proses ini adalah proses *AddRoundKey* dengan men XOR nilai *state* akhir ronde 1 dengan kunci awal.

39	7F	12	E5	XOR	41	6E	67	67
14	3E	52	FF		69	6B	77	69
32	4D	1C	36		73	75	64	61
7A	BC	2F	41		32	30	31	39

XOR setiap kolom bilangan *state* akhir ronde 1 dengan setiap kolom nilai hexadesimal kunci awal seperti di bawah ini :

39	XOR	41
14		69
32		73
7A		32

$$39 = 00111001$$

$$41 = 01000001$$

_____ XOR

01111000 dalam hexadesimal adalah **78**

$$14 = 00010100$$

$$69 = 01101001$$

_____ XOR

01111101 dalam hexadesimal adalah **7D**

$$32 = 00110010$$

$$73 = 01110011$$

_____ XOR

01000001 dalam hexadesimal adalah **41**

$$7A = 01111010$$

$$32 = 00110010$$

_____ XOR

01001000 dalam hexadesimal adalah **48**

Sehingga didapat kolom pertama hasil *AddRoundKey* pertama *plainimage* sebagai di bawah ini :

78
7D
41
48

Untuk kolom selanjutnya dilakukan perhitungan XOR dengan cara yang sama yaitu kolom kedua *state* di XOR dengan kolom kedua kunci dan seterusnya. Sehingga didapat hasil keseluruhan berupa nilai hexadesimal *plainimage* setelah *AddRoundKey* (*initial round*) sebagai berikut :

78	11	75	82
7d	55	25	96
41	38	78	57
48	8c	1e	78

Hasil di atas akan merupakan nilai *plainimage*. Berikut tabel desimal *plainimage* hasil dekripsi:

Tabel 4.5 Nilai *Pixel* Citra Hasil Dekripsi

Pixel	Warna	Plainimage	
		Desimal	Hexa
1	R	120	78
	G	17	11
	B	117	75
2	R	130	82
	G	125	7D
	B	85	55
3	R	37	25
	G	150	96
	B	65	41

Tabel Lanjutan 4.5 Nilai *Pixel* Citra Hasil Dekripsi

Pixel	Warna	Plainimage	
		Desimal	Hexa
4	R	56	38
	G	120	78
	B	87	57
5	R	75	48
	G	140	8C
	B	30	1E
6	R	120	78
	G	25	19
	B	10	A



4.3 Hasil Pengujian Program Aplikasi

Hasil pengujian program meliputi dari hasil enkripsi, dan hasil dekripsi. Pengujian enkripsi dilakukan dengan 5 sampel citra yang kemudian dilakukan dekripsi. Berikut uraian masing – masing hasil pengujian :


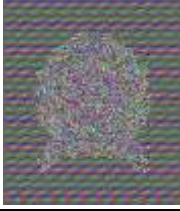






1. Hasil pengujian Enkripsi Citra *Plainimage*

Hasil pengujian enkripsi citra meliputi dari keterangan resolusi, *size* dan waktu enkripsi. Kunci yang digunakan adalah anggikwisuda2019. Contoh hasil dapat dilihat pada tabel di bawah ini :

Tabel 4.6 Pengujian Enkripsi Citra *plainimage*

No	Plainimage	Eks	Resolusi	Size KB	Cipherimage	Size KB	Waktu Enkripsi
1		Jpg	259 x 339	76		299	12 Detik

Tabel Lanjutan 4.6 Pengujian Enkripsi Citra *plainimage*

No	Plainimage	Eks	Resolusi	Size KB	Cipherimage	Size KB	Waktu Enkripsi
2		Png	300 x 330	56		88	3 Detik
3		Jpg	481 x 657	944		1273	34 Detik
4		Png	512 x 512	312		628	2 Detik
5		Jpg	541 x 717	199		1312	10 Detik

Berdasarkan pada tabel 4.6 di atas dapat diambil kesimpulan :

- Program aplikasi berhasil mengenkripsi sebuah *plainimage* menjadi sebuah *cipherimage*.
- Hasil enkripsi *cipherimage* sangat berbeda dengan *plainimage*, dimana *cipherimage* tidak dapat dikenali dengan pandangan mata manusia.
- Ukuran citra *plainimage* mengalami perubahan ketika proses enkripsi yang menjadi *cipherimage*. Ukuran *cipherimage* tersebut menjadi lebih besar dari *plainimage*.



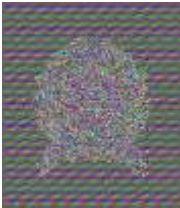



d. Waktu enkripsi citra *plainimage* memiliki variasi, hal ini disebabkan karena proses enkripsi program aplikasi dipengaruhi oleh *size*, resolusi dan spesifikasi komputer.

2. Pengujian Dekripsi *Cipherimage*





Pengujian dekripsi *cipherimage* meliputi dari keterangan resolusi, *size* dan waktu dekripsi. Kunci yang digunakan akan otomatis tersedia di *form* dekripsi.

Adapun keterangan dekripsi dapat dilihat pada tabel di bawah ini :

Tabel 4.7 Pengujian Dekripsi Citra *plainimage*

No	Plainimage	Eks	Resolusi	Size KB	Cipherimage	Size KB	Waktu Enkripsi
1		Bmp	259 x 339	299		299	15 Detik
2		Bmp	300 x 330	88		88	5 Detik
3		Bmp	481 x 657	1273		1273	38 Detik

Tabel Lanjutan 4.7 Pengujian Dekripsi Citra *plainimage*

No	Plainimage	Eks	Resolusi	Size KB	Cipherimage	Size KB	Waktu Enkripsi
4		Bmp	512 x 512	628		628	5 Detik
5		Bmp	541 x 717	1312		1312	14 Detik

Berdasarkan pada tabel 4.7 di atas diambil kesimpulan :

- a. Program aplikasi berhasil mendekripsi sebuah *cipherimage* menjadi citra *plainimage* seperti semula.
- b. Ukuran citra *plainimage* hasil dekripsi tidak mengalami perubahan dengan ukuran citra *cipherimage* sebelum proses dekripsi.
- c. Waktu dekripsi *cipherimage* memiliki sedikit perbedaan saat proses enkripsi.

Hal ini disebabkan oleh proses dekripsi menggunakan perangkat keras komputer minimal.

BAB V

PENUTUP

5.1 Simpulan

Berdasarkan pembahasan dalam merancang aplikasi keamanan citra digital menggunakan algoritma AES 128-bit berbasis jaringan Client – Server, maka dapat diambil kesimpulan sebagai berikut :

1. Dengan adanya aplikasi enkripsi dan dekripsi citra digital ini yang digunakan untuk mengamankan citra digital, penulis dapat menganalisis proses pengamanan citra digital dengan mudah dan cepat.
2. Pada proses enkripsi, sebuah *plainimage* dirubah terlebih dahulu menggunakan algoritma AES-128 bit yang melakukan proses sebanyak 4 ronde dan didapatkan sebuah *cipherimage* yang memiliki perubahan ukuran *size* dari citra *plainimage*.
3. Untuk proses enkripsi dan dekripsi menggunakan kunci yang sama.

5.2 Saran

Adapun saran - saran yang dapat dilakukan penelitian ataupun pengembangan selanjutnya adalah sebagai berikut :

1. Perangkat lunak ini dapat dikembangkan dengan menggunakan perangkat berbasis android pada aplikasi seperti WhatsApp, dan LINE.
2. Perangkat lunak ini dapat dikembangkan menggunakan Artificial Intelligence (AI).

DAFTAR PUSTAKA

- Abdul, I., I., Widodo, S. (2014). Kriptografi File Menggunakan Metode AES Dual Password. *Prosiding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen*, 8, 263-264. Diakses dari <https://media.neliti.com/media/publications/171445-ID-kriptografi-file-menggunakan-metode-aes.pdf>
- Akil, I. (2016). Rekayasa Perangkat Lunak dengan Model Unified Process Studi Kasus : Sistem Informasi Journal. *Jurnal Pilar Nusa Mandiri*. 7 (1), 2. Diakses dari <http://ejournal.nusamandiri.ac.id/ejurnal/index.php/pilar/article/viewFile/89/85>
- Aleisa, N. (2015). A Comparison of the 3DES and AES Encryption Standards. *International Journal of Security and its Applications*, 9 (7), 242-243. Diakses dari https://www.researchgate.net/publication/283178410_A_comparison_of_the_3DES_and_AES_encryption_standards
- AMIK BSI*, 1 (1), 55. Diakses dari <http://ejournal.bsi.ac.id/ejurnal/index.php/jtk/article/viewFile/236/202>
- Amin, M. (2016). Implementasi Kriptografi Klasik Pada Komunikasi Berbasis Teks. *Jurnal* 1), 3. Diakses dari <https://jurnal.unimed.ac.id/2012/index.php/cess/article/viewFile/4040/3594>
- Arif, M. (2016). Bahan Ajar Rancangan Teknik Industri. Yogyakarta: CV. Budi Utama.
- Asriyanik. (2017). Studi Terhadap Advanced Encryption Standard (AES) dan Algoritma Knapsack dalam Pengamanan Data. *Jurnal STANTIKA*, 7 (1), 553-557. Diakses dari <http://eprints.ummi.ac.id/188/1/STUDI%20TERHADAP%20ADVANCED%20ENCRYPTION%20STANDARD%20%28AES%29%20DAN%20ALGORITMA%20KNAPSACK%20DALAM%20PENGAMANAN%20DATA.pdf>
- Azmi, Fadhillah, and Winda Erika. "Analisis Keamanan Data Pada Block Cipher Algoritma Kriptografi Rsa." *CESS (Journal of Computer Engineering, System and Science)* 2.1: 27-29.
- Deslianti, D., Muttaqin, I. (2016). Aplikasi Kumpulan Hadist Nabi Muhammad SAW Berbasis Android Menggunakan Algoritma Merge Sort. *Jurnal Pseudocode*, 3 (1), 27. Diakses dari <https://media.neliti.com/media/publications/126448-ID-analisis-perbandingan-algoritma-bubble-s.pdf>

- Edhy, S. (2005). Pengantar Teknologi Informasi. Yogyakarta : Graha Ilmu.
- Erika, Winda, Heni Rachmawati, and Ibnu Surya. "Enkripsi Teks Surat Elektronik (E-Mail) Berbasis Algoritma Rivest Shamir Adleman (RSA)." *Jurnal Aksara Komputer Terapan* 1.2 (2012).
- Hafni, Layla, and Rismawati Rismawati. "Analisis Faktor-Faktor Internal Yang Mempengaruhi Nilai Perusahaan Pada Perusahaan Manufaktur Yang Terdaftar Di Bei 2011-2015." *Bilancia: Jurnal Ilmiah Akuntansi* 1.3 (2017): 371-382.
- Hamdi, Muhammad Nurul, Evi Nurjanah, and Latifah Safitri Handayani. "Community Development Based On Ibnu Khaldun Thought, Sebuah Interpretasi Program Pemberdayaan Umkm Di Bank Zakat El-Zawa." *El Muhasaba: Jurnal Akuntansi (e-journal)* 5.2 (2014): 158-180.
- Hartanto, S. (2017). Implementasi fuzzy rule based system untuk klasifikasi buah mangga. *TECHSI-Jurnal Teknik Informatika*, 9(2), 103-122.
- Harumy, T. H. F., & Sulistianingsih, I. (2016). Sistem penunjang keputusan penentuan jabatan manager menggunakan metode mfep pada cv. Sapo durin. In *Seminar Nasional Teknologi Informasi dan Multimedia* (pp. 6-7).
- Herdianto, H. (2018). Perancangan Smart Home dengan Konsep Internet of Things (IoT) Berbasis Smartphone. *Jurnal Ilmiah Core IT: Community Research Information Technology*, 6(2).
Informatika, 3 (2), 81-82. Diakses dari <https://ejournal.undiksha.ac.id/index.php/janapati/article/viewFile/9808/6231>
Informatika.
- Iswandy, E. (2015). Sistem Penunjang Keputusan untuk Menentukan Penerimaan Dana Santunan Sosial Anak Nagari dan Penyalurannya bagi Mahasiswa dan Pelajar Kurang Mampu di Kenagarian Barung – Barung Balantai Timur. *Jurnal TEKNOIF*,
- Khairul, K., Haryati, S., & Yusman, Y. (2018). Aplikasi Kamus Bahasa Jawa Indonesia dengan Algoritma Raita Berbasis Android. *Jurnal Teknologi Informasi dan Pendidikan*, 11(1), 1-6.
- Mahajan, P., Sachdeva, A. (2013). A Study of Encryption Algorithms AES, DES and RSA for Security. *Global Journal of Computer Science and Technology, Network Web & Security*, 13. Diakses https://globaljournals.org/GJCST_Volume13/4-A-Study-of-Encryption-Algorithms.pdf

- Muttaqin, Muhammad. "Analisa Pemanfaatan Sistem Informasi E-Office Pada Universitas Pembangunan Panca Budi Medan Dengan Menggunakan Metode Utaut." *Jurnal Teknik dan Informatika* 5.1 (2018): 40-43.
- Permana, H., T., Darmawiguna, M., Kesiman, A., W. (2014). JA-KO Balinese Pizza: Game Edukasi Interaktif Jaringan Komputer. *Jurnal Nasional Pendidikan Teknik*
- Perwitasari, I. D. (2018). Teknik Marker Based Tracking Augmented Reality untuk Visualisasi Anatomi Organ Tubuh Manusia Berbasis Android. *INTECOMS: Journal of Information Technology and Computer Science*, 1(1), 8-18.
- Prabowo, S. (2013). Kriptografi-Jenis jenis Serangan dalam Kriptografi. Diakses 1 Maret 2019, dari <http://www.sigiprabowo.id/2013/01/kriptografi-jenis-jenis-serangan-dalam.html?m=1>
- Prameshwari, A., Sastra, P., N. (2018). Implementasi Algoritma Advanced Encryption Standard (AES) 128 untuk Enkripsi dan Dekripsi File Dokumen. *Eksplora Informatika*, 8 (2), 53. DOI: 10.30864/eksplora.v8i1.139
- Pseudocode*, 3 (2), 130. Diakses dari <https://media.neliti.com/media/publications/126448-ID-analisis-perbandingan-algoritma-bubble-s.pdf>
- Putra, D. (2010). *Pengolahan Citra Digital*. Yogyakarta: Andi.
- Putri, R. E., & Siahaan, A. (2017). Examination of document similarity using Rabin-Karp algorithm. *International Journal of Recent Trends in Engineering & Research*, 3(8), 196-201.
- Rahmawati, R., Rahardjo, D. (2016). Aplikasi Pengamanan Data Menggunakan Algoritma Steganografi Discrete Cosine Transform dan Kriptografi AES 128 BIT pada SMK PGRI 15 Jakarta. *Jurnal Teknik Informatika dan Sistem Informasi*, 2 (1), 67-69. Diakses dari <https://media.neliti.com/media/publications/141018-ID-aplikasi-pengamanan-data-menggunakan-alg.pdf>
- Ramadhani, S., Suherman, S., Melvasari, M., & Herdianto, H. (2018). Perancangan Teks Berjalan Online Sebagai Media Informasi Nelayan. *Jurnal Ilmiah Core IT: Community Research Information Technology*, 6(2).
- Rizal, Chairul. "Pengaruh Varietas dan Pupuk Petroganik Terhadap Pertumbuhan, Produksi dan Viabilitas Benih Jagung (*Zea mays L.*)."
ETD Unsyiah (2013).
- Setiawan, S., Firdaus, A. (2014). Pembuatan Aplikasi Penajaman Gambar Untuk Pengolahan Citra Digital. *Jurnal Techno Nusa Mandiri*, 6 (1), 53. Diakses dari <http://ejournal.nusamandiri.ac.id/ejurnal/index.php/techno/article/download/109/105>

- Sukamto, R., A., Shalahuddin, M. (2013). *Rekayasa Perangkat Lunak*. Bandung : Syahputra, Rizki, and Hafni Hafni. "Analisis Kinerja Jaringan Switching Clos Tanpa Buffer." *Journal Of Science And Social Research* 1.2 (2018): 109-115.
- Temukan Pengertian. (2013). <https://www.temukanpengertian.com/2013/08/pengertian-citra-digital.html>
- Varianto, E., Badrul, M. (2015). Implementasi Virtual Private Network dan Proxy Server Menggunakan Clear Os pada PT.Valdo International. *Jurnal Teknik Komputer*
- Widarma, A. (2016). Kombinasi Algoritma AES, RC4 dan Elgamal dalam Skema Hybrid untuk Keamanan Data. *Journal Of Computer Engineering System and Science*, 1
- Winafil, M., Sinurat, S., Zebua, T. (2018). Implementasi Algoritma Advanced Encryption Standard dan Triple Data Encryption Standard Untuk Mengamankan Citra Ditigal. *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, 2 (1), 450. Diakses dari <http://ejurnal.stmik-budidarma.ac.id/index.php/komik>
- Wongkar, S., Sinsuw, A., Najoran, X. (2015). Analisa Implementasi Jaringan Internet dengan Menggabungkan Jaringan LAN dan WLAN di Desa Kawangkoan bawah Wilayah Amurang II. *E-Journal Teknik Elektro dan Komputer*, 4 (6), 63-64. Diakses dari <https://ejournal.unsrat.ac.id/index.php/elekdankom/article/viewFile/10400/9986>
- Yesputra, R. (2017). *Belajar Visual Basic.NET dengan Visual Studio 2010*. Kisaran : Royal Asahan Press.