BAB IV
ANALISA DAN PERANCANGAN

Pada bab ini berisi tentang analisa dan perancangan program kripto-stego dengan menggunakan *JavaScript*. Analisa ini bertujuan untuk mengetahui cara kerja proses kriptografi dan steganografi, serta keluaran yang dihasilkan. Dalam perancangan akan membahas mengenai analisa algoritma AES dan steganografi metode PIT.

4.1 Analisa Algoritma AES

AES adalah algoritma *block cipher*, yaitu proses data masukan dibagi terlebih dahulu ke dalam blok-blok, kemudian dilakukan proses enkripsi dan dekripsi secara terpisah pada setiap blok. AES-128 terdiri atas 4x4 *byte* yang artinya terbagi dalam 128 pit per blok.

Secara umum, analisa proses enkripsi AES dijelaskan sebagai berikut :

1. Sisipkan *array* 4x4 (kunci)
2. Sisipkan *array* 4x4 *(state)*
3. Masukkan kunci (yang sudah dikonversi ke dalam bilangan heksadesimal) berukuran 16 bit
4. Masukkan pesan yang akan dienkripsi
5. Konversi teks ke dalam heksadesimal
6. Bagi kelompok teks dalam 128 bit per bagian
7. Ambil 128 bit pertama untuk diproses
8. Kelompokkan bit teks ke dalam 16 bagian (8 bit per bagiannya)
9. Lakukan proses *AddRoundKey*
10. Lakukan proses *SubBytes*
11. Lakukan proses *ShiftRows*
12. Lakukan proses *MixColumns*
13. Ulangi langkah 9 sampai 12 hingga iterasi ke-9

14. Pada iterasi ke-10, proses *MixColumns* tidak dilakukan lagi, hanya *SubBytes, ShiftRows,* dan *AddRoundKey*
15. Selesai.

Adapun proses dekripsi secara umum dijelaskan sebagai berikut :

1. Pesan yang sudah dienkripsi dibaca dalam *byte stream*
2. *Byte stream* dibagi 16 *byte* per blok
3. Lakukan proses *AddRoundKey*
4. Lakukan proses *InvShiftRow*
5. Lakukan proses *InvSubBytes*
6. Lakukan proses *AddRoundKey*
7. Lakukan proses *InvMixColumns*
8. Ulangi langkah 3 sampai 7 hingga iterasi ke-9
9. Pada iterasi ke-10, lakukan langkah 3 sampai 6 saja.
10. Selesai.

4.1.1 Analisa Enkripsi

Berikut dijelaskan contoh perhitungan manual dari proses enkripsi AES-128 :

Plaintext : 2e cc c8 50 | 01 40 9e 62 | 62 f6 51 a7 | 1a 37 23 2b

Key : 3e ab dd 11 | 14 67 83 fc | 11 23 af 64 | cc da 19 aa

Hitung telebih dahulu *key expansion,* yang mana hasil dari perhitungan akan digunakan pada proses enkripsi selanjutnya.

Key Expansion :

Cipher Key : 3e ab dd 11 | 14 67 83 fc | 11 23 af 64 | cc da 19 aa

3e	14	11	cc
ab	67	23	da
dd	83	af	19
11	fc	64	aa

RotWord : da 19 aa cc

S-Box : 57 d4 ac 4b

- *Round Key 1* : 68 7f 71 5a | 7c 18 f2 a6 | 6d 3b 5d c2 | a1 e1 44 68
- *Round Key 2* : 92 64 34 68 | ee 7c c6 ce | 83 47 9b 0c | 22 a6 df 64
- *Round Key 3* : b2 fa 77 fb | 5c 86 b1 35 | df c1 2a 39 | fd 67 f5 5d
- *Round Key 4* : 3f 1c 3b af | 63 9a 8a 9a | bc 5b a0 a3 | 41 3c 55 fe
- *Round Key 5* : c4 e0 80 2c | a7 7a 0a b6 | 1b 21 aa 15 | 5a 1d ff eb
- *Round Key 6* : 40 f6 69 92 | e7 8c 63 24 | fc ad c9 31 | a6 b0 36 da
- *Round Key 7* : e7 f3 3e b6 | 00 7f 5d 92 | fc d2 94 a3 | 5a 62 a2 79
- *Round Key 8* : cd c9 88 08 | cd b6 d5 9a | 31 64 41 39 | 6b 06 e3 40
- *Round Key 9* : b9 d8 81 77 | 74 6e 54 ed | 45 0a 15 d4 | 2e 0c f6 94
- *Round Key 10* : 71 9a a3 46 | 05 f4 f7 ab | 40 fe e2 7f | 6e f2 14 eb

a. *Initial Round*

Lakukan operasi XOR antara *state* dan *key*.

- *Initial state* : 2e cc c8 50 | 01 40 9e 62 | 62 f6 51 a7 | 1a 37 23 2b
- *Cipher key* : 3e ab dd 11 | 14 67 83 fc | 11 23 af 64 | cc da 19 fa
- *After AddRoundKey* : 10 67 15 41 | 15 27 1d 9e | 73 d5 fe c3 | d6 ed 3a 81

Setelah proses inisialisasi selesai, mulai proses *Round 1*. *State* pada *Round 1* merupakan *After AddRoundKey* pada *initial round*.

b. *Round 1*

- *State* : 10 67 15 41 | 15 27 1d 9e | 73 d5 fe c3 | d6 ed 3a 81
- *Round Key* : 68 7f 71 5a | 7c 18 f2 a6 | 6d 3b 5d c2 | a1 e1 44 68

Langkah pertama yang dilakukan pada setiap *round* adalah Transformasi SubBytes, yaitu substitusi *state* menggunakan tabel S-Box. Jika diketahui *byte* pertama *state* pada *round 1* adalah 1c (x = 1, y = 0), maka :

		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

Lakukan proses diatas hingga semua matriks tersubstitusi.

- *After SubBytes* : ca 85 59 83 | 59 cc a4 0b | 8f 03 bb 2e | f6 55 80 0c

Langkah kedua adalah Transformasi *ShiftRows*.

ca	59	8f	f6	➢	ca	59	8f	f6
85	cc	03	55		cc	03	55	85
59	a4	bb	80		**59**	**a4**	bb	80
83	0b	2e	0c		83	0b	2e	0c

➢	ca	59	8f	f6	➢	ca	59	8f	f6
	cc	03	55	85		cc	03	55	85
	bb	80	59	a4		bb	80	59	a4
	83	**0b**	**2e**	0c		0c	83	0b	2e

- *After ShiftRows* : ca cc bb 0c | 59 03 80 83 | 8f 55 59 0b | f6 85 a4 2e

Kemudian, tahap ketiga adalah Transformasi *MixColumns*, dimana hasil dari *After ShiftRows* dikalikan dengan perkalian matriks dalam ketentuan Galois Field (GF (2^8)) :

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} S'_0 \\ S'_1 \\ S'_2 \\ S'_3 \end{bmatrix}$$

Langkah perhitungan Mix Columns :

$\boldsymbol{S'_0}$:

ca = 11001010

02 = 00000010

ca = $x^7 + x^6 + x^3 + x$

02 = x

= $\boldsymbol{x^8 + x^7 + x^4 + x^2}$

cc = 11001100

03 = 00000011

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

03 = $x + 1$

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

$= \boldsymbol{x^8 + x^6 + x^4 + x^2}$

bb = 10111011

$= \boldsymbol{x^7 + x^5 + x^4 + x^3 + x + 1}$

0c = 00001100

$= \boldsymbol{x^3 + x^2}$

$S'_0 = x^6 + x^5 + x^4 + x^2 + x + 1$

= 01110111 = **77**

- *After MixColumns* : 77 93 7f 2a | b4 47 df 75 | a8 c5 75 90 | e9 3e 52 7c

Setelah hasil *MixColumns* didapatkan, lakukan *AddRoundKey* dengan cara perhitungan XOR antara *round key* dengan hasil *MixColumns*.

68 = 01101000 XOR 77 = 01110111 = 00011111 = 1f

- *After AddRoundKey* : 1f ec 0e 70 | c8 5f 2d d3 | c5 fe 28 52 | 48 df 16 14

c. *Round 2*

- *After SubBytes* : c0 ce ab 51 | e8 cf d8 66 | a6 bb 34 00 | 52 9e 47 fa
- *After ShiftRows* : c0 cf 34 fa | e8 bb 47 51 | a6 9e ab 66 | 52 ce d8 00
- *After MixColumns* : 1f e3 72 4f | 0b 1d 2e 7d | 23 01 df 08 | 35 a6 37 e0
- *After AddRoundKey* : 8d 87 46 27 | e5 61 e8 b3 | a0 46 44 04 | 17 00 e8 84

d. *Round 3*

- *After SubBytes* : 5d 17 5a cc | d9 ef 9b 6d | e0 5a 1b f2 | f0 63 9b 5f
- *After ShiftRows* : 5d ef 1b 5f | d9 5a 9b cc | e0 63 5a 6d | f0 17 9b f2
- *After MixColumns* : d4 ea 65 ad | 10 17 e1 32 | 49 a5 80 d8 | ab 9a c7 78
- *After AddRoundKey* : 66 10 12 56 | 4c 91 50 07 | 96 64 aa e1 | 56 fd 32 25

e. *Round 4*

- *After SubBytes* : 33 ca c9 b1 | 29 81 53 c5 | 90 43 ac f8 | b1 54 23 3f
- *After ShiftRows* : 33 81 ac 3f | 29 43 23 b1 | 90 54 c9 c5 | b1 ca 53 f8
- *After MixColumns* : 6d fa b0 06 | 05 7b e4 62 | cb bd 19 a7 | 97 33 ce ba
- *After AddRoundKey* : 52 e6 8b a9 | 66 e1 6e f8 | 77 e6 b9 04 | d6 0f 9b 44

f. *Round 5*

- *After SubBytes* : 00 8e 3d d3 | 33 f8 9f 41 | f5 8e 56 f2 | f6 76 14 1b
- *After ShiftRows* : 00 f8 56 1b | 33 8e 14 d3 | f5 76 3d 41 | f6 8e 9f f2

- *After MixColumns* : 5e 0a 79 98 | 28 db fb 72 | 17 1f 3a cd | 13 b9 50 ef
- *After AddRoundKey* : 9a ea f9 b4 | 8f a1 f1 c4 | 0c 3e 90 d8 | 49 a4 af 04

g. *Round 6*

- *After SubBytes* : b8 87 99 8d | 73 32 a1 1c | fe b2 60 61 | 3b 49 79 f2
- *After ShiftRows* : b8 32 60 f2 | 73 b2 79 8d | fe 49 99 1c | 3b 87 a1 61
- *After MixColumns* : af 8e 47 7e | df 0a bf 5f | b9 c0 ba f1 | 24 b7 46 a9
- *After AddRoundKey* : ef 78 2e ec | 38 86 dc 7b | 45 6d 73 c0 | 82 07 70 73

h. *Round 7*

- *After SubBytes* : df bc 31 ce | 07 44 86 21 | 6e 3c 8f ba | 13 c5 51 8f
- *After ShiftRows* : df 44 8f 8f | 07 3c 51 ce | 6e c5 31 21 | 13 bc 86 ba
- *After MixColumns* : 69 52 14 b4 | d5 42 d0 e3 | 98 8d aa 04 | c5 5b 6d 60
- *After AddRoundKey* : 8e a1 2a 02 | d5 3d 8d 71 | 64 5f 3e a7 | 9f 39 cf 19

i. *Round 8*

- *After SubBytes* : 19 32 e5 77 | 03 27 5d a3 | 43 cf b2 5c | db 12 8a d4
- *After ShiftRows* : 19 27 b2 d4 | 03 cf 8a 77 | 43 12 e5 a3 | db 32 5d 5c
- *After MixColumns* : 3d 4e 26 0d | b1 74 5a ae | f6 f0 7e 6f | fa 04 b7 a1
- *After AddRoundKey* : f0 87 ae 05 | 7c c2 8f 34 | c7 94 3f 56 | 91 02 54 e1

j. *Round 9*

- *After SubBytes* : 8c 17 e4 6b | 10 25 73 18 | c6 22 75 b1 | 81 77 20 f8
- *After ShiftRows* : 8c 25 75 f8 | 10 22 20 6b | c6 77 e4 18 | 81 17 73 b1
- *After MixColumns* : e1 a1 50 34 | 0d 5f cf e4 | f2 07 4a f2 | e2 8b b8 b5
- *After AddRoundKey* : 58 79 d1 43 | 79 31 9b 09 | b7 0d 5f 26 | cc 87 4e 11

k. *Round 10 (Final Round)*

- *After SubBytes* : 6a b6 3e 1a | b6 c7 14 01 | a9 d7 cf f7 | 4b 17 2f 82
- *After ShiftRows* : 6a c7 cf 82 | b6 d7 2f 1a | a9 17 3e 01 | 4b b6 14 f7
- *After AddRoundKey* : 1b 5d 6c c4 | b3 23 d8 b1 | e9 e9 dc 7e | 25 44 00 1c

Hasil *Cipher Text* : **1b 5d 6c c4 | b3 23 d8 b1 | e9 e9 dc 7e | 25 44 00 1c**

4.1.2 Analisa Dekripsi

- *Cipher Text* : 1b 5d 6c c4 | b3 23 d8 b1 | e9 e9 dc 7e | 25 44 00 1c
- *Key* : 3e ab dd 11 | 14 67 83 fc | 11 23 af 64 | cc da 19 aa

Key Expansion

- *Round Key 1* : e1 a1 50 34 | 0d 5f cf e4 | f2 07 4a f2 | e2 8b b8 85
- *Round Key 2* : 3d 4e 26 0d | b1 74 5a ae | f6 f0 7e 6f | fa 04 b7 a1
- *Round Key 3* : 69 52 14 b4 | d5 42 d0 e3 | 98 8d aa 04 | c5 5b 6d 60
- *Round Key 4* : af 8e 47 7e | df 0a bf 5f | b9 c0 ba f1 | 24 b7 46 a9
- *Round Key 5* : 5e 0a 79 98 | 28 db fb 72 | 17 1f 3a cd | 13 b9 50 ef
- *Round Key 6* : 6d fa b0 06 | 05 7b e4 62 | cb bd 19 a7 | 97 33 ce ba
- *Round Key 7* : d4 ea 65 ad | 10 17 e1 32 | 49 a5 80 d8 | ab 9a c7 78
- *Round Key 8* : 1f e3 72 4f | 0b 1d 2e 7d | 23 01 df 08 | 35 a6 37 e0
- *Round Key 9* : 77 93 7f 2a | b4 47 df 75 | a8 c5 75 90 | e9 3e 52 7c
- *Round Key 10* : 3e ab dd 11 | 14 67 83 fc | 11 23 af 64 | cc da 19 aa

a. *Initial Round*

- *After AddRoundKey* : 6a c7 cf 82 | b6 d7 2f 1a | a9 17 3e 01 | 4b b6 14 f7

b. *Round 1*

Di *Round 1* proses dekripsi, pertama lakukan proses *Inverse ShiftRows*. Proses ini merupakan kebalikan dari *ShiftRows.*

6a	b6	a9	4b		6a	b6	a9	4b
c7	d7	17	**b6**	➢	b6	c7	d7	17
cf	2f	3e	14		cf	2f	**3e**	**14**
82	1a	01	f7		82	1a	01	f7

	6a	b6	a9	4b		ca	59	8f	f6
➢	b6	c7	d7	17	➢	cc	03	55	85
	3e	14	cf	2f		bb	80	59	a4
	82	**1a**	**01**	**f7**		1a	01	f7	82

- *After InvShiftRows* : 6a b6 3e 1a | b6 c7 14 01 | a9 d7 cf f7 | 4b 17 2f 82

Langkah kedua, hasil *Inverse ShiftRows* di-*inversesubstitusi*-kan dengan mengacu pada tabel *InvSubBytes*

- *After InvSubBytes* : 58 79 d1 43 | 79 31 9b 09 | b7 0d 5f 26 | cc 87 4e 11

Kemudian, lakukan perhitungan XOR antara hasil *InvSubBytes* dengan *round key.*

58 = 01011000 XOR e1 = 11100001 = 10111001 = b9.

- *After AddRoundKey* : b9 d8 81 77 | 74 6e 54 ed | 45 0a 15 d4 | 2e 0c f6 94
- *After InvMixColumns* : 8c 25 75 f8 | 10 22 20 6b | c6 77 e4 18 | 81 17 73 b1

c. *Round 2*

- *After InvShiftRows* : 8c 17 e4 6b | 10 25 73 18 | c6 22 75 b1 | 81 77 20 f8
- *After InvSubBytes* : f0 87 ae 05 | 7c c2 8f 34 | c7 94 3f 56 | 91 02 54 e1
- *After AddRoundKey* : cd c9 88 08 | cd b6 d5 9a | 31 64 41 39 | 6b 06 e3 40
- *After InvMixColumns* : 19 27 b2 d4 | 03 cf 8a 77 | 43 12 e5 a3 | db 32 5d 5c

d. *Round 3*

- *After InvShiftRows* : 19 32 e5 77 | 03 27 5d a3 | 43 cf b2 5c | db 12 8a d4
- *After InvSubBytes* : 8e a1 2a 02 | d5 3d 8d 71 | 64 5f 3e a7 | 9f 39 cf 19
- *After AddRoundKey* : e7 f3 3e b6 | 00 7f 5d 92 | fc d2 94 a3 | 5a 62 a2 79
- *After InvMixColumns* : df 44 8f 8f | 07 3c 51 ce | 6e c5 31 21 | 13 bc 86 ba

e. *Round 4*

- *After InvShiftRows* : df bc 31 ce | 07 44 86 21 | 6e 3c 8f ba | 13 c5 51 8f
- *After InvSubBytes* : ef 78 2e ec | 38 86 dc 7b | 45 6d 73 c0 | 82 07 70 73
- *After AddRoundKey* : 40 f6 69 92 | e7 8c 63 24 | fc ad c9 31 | a6 b0 36 da
- *After InvMixColumns* : b8 32 60 f2 | 73 b2 79 8d | fe 49 99 1c | 3b 87 a1 61

f. *Round 5*

- *After InvShiftRows* : b8 87 99 8d | 73 32 a1 1c | fe b2 60 61 | 3b 49 79 f2
- *After InvSubBytes* : 9a ea f9 b4 | 8f a1 f1 c4 | 0c 3e 90 d8 | 49 a4 af 04
- *After AddRoundKey* : c4 e0 80 2c | a7 7a 0a b6 | 1b 21 aa 15 | 5a 1d ff eb
- *After InvMixColumns* : 00 f8 56 1b | 33 8e 14 d3 | f5 76 3d 41 | f6 8e 9f f2

g. *Round 6*

- *After InvShiftRows* : 00 8e 3d d3 | 33 f8 9f 41 | f5 8e 56 f2 | f6 76 14 1b
- *After InvSubBytes* : 52 e6 8b a9 | 66 e1 6e f8 | 77 e6 b9 04 | d6 0f 9b 44
- *After AddRoundKey* : 3f 1c 3b af | 63 9a 8a 9a | bc 5b a0 a3 | 41 3c 55 fe
- *After InvMixColumns* : 33 81 ac 3f | 29 43 23 b1 | 90 54 c9 c5 | b1 ca 53 f8

h. *Round 7*

- *After InvShiftRows* : 33 ca c9 b1 | 29 81 53 c5 | 90 43 ac f8 | b1 54 23 3f
- *After InvSubBytes* : 66 10 12 56 | 4c 91 50 07 | 96 64 aa e1 | 56 fd 32 25

- *After AddRoundKey* : b2 fa 77 fb | 5c 86 b1 35 | df c1 2a 39 | fd 67 f5 5d
- *After InvMixColumns* : 5d ef 1b 5f | d9 5a 9b cc | e0 63 5a 6d | f0 17 9b f2

i. *Round 8*

- *After InvShiftRows* : 5d 17 5a cc | d9 ef 9b 6d | e0 5a 1b f2 | f0 63 9b 5f
- *After InvSubBytes* : 8d 87 46 27 | e5 61 e8 b3 | a0 46 44 04 | 17 00 e8 84
- *After AddRoundKey* : 92 64 34 68 | ee 7c c6 ce | 83 47 9b 0c | 22 a6 df 64
- *After InvMixColumns* : c0 cf 34 fa | e8 bb 47 51 | a6 9e ab 66 | 52 ce d8 00

j. *Round 9*

- *After InvShiftRows* : c0 ce ab 51 | e8 cf d8 66 | a6 bb 34 00 | 52 9e 47 fa
- *After InvSubBytes* : 1f ec 0e 70 | c8 5f 2d d3 | c5 fe 28 52 | 48 df 16 14
- *After AddRoundKey* : 68 7f 71 5a | 7c 18 f2 a6 | 6d 3b 5d c2 | a1 e1 44 68
- *After InvMixColumns* : ca cc bb 0c | 59 03 80 83 | 8f 55 59 0b | f6 85 a4 2e

k. *Round 10 (Final Round)*

- *After InvShiftRows* : ca 85 59 83 | 59 cc a4 0b | 8f 03 bb 2e | f6 55 80 0c
- *After InvSubBytes* : 10 67 15 41 | 15 27 1d 9e | 73 d5 fe c3 | d6 ed 3a 81
- *After AddRoundKey* : 2e cc c8 50 | 01 40 9e 62 | 62 f6 51 a7 | 1a 37 23 2b

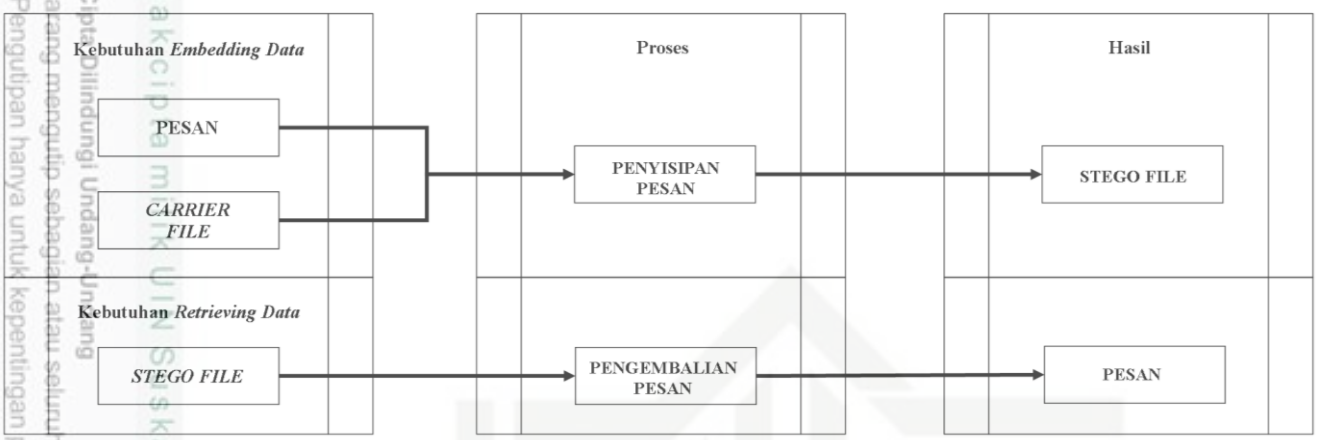
Hasil *Plain Text* : **2e cc c8 50 | 01 40 9e 62 | 62 f6 51 a7 | 1a 37 23 2b**

4.2 Analisa Steganografi PIT

4.2.1 Gambaran Umum

Pada umumnya, program steganografi digunakan untuk keperluan penyisipan pesan atau informasi ke dalam suatu media *(carrier file)* agar keberadaan pesan sulit dideteksi. Pesan yang sudah disisipkan harus bisa dikembalikan *(retrieving)* dari dalam media penyisipan. Dalam penelitian ini, media yang digunakan adalah file gambar JPG, PNG, dan BMP.

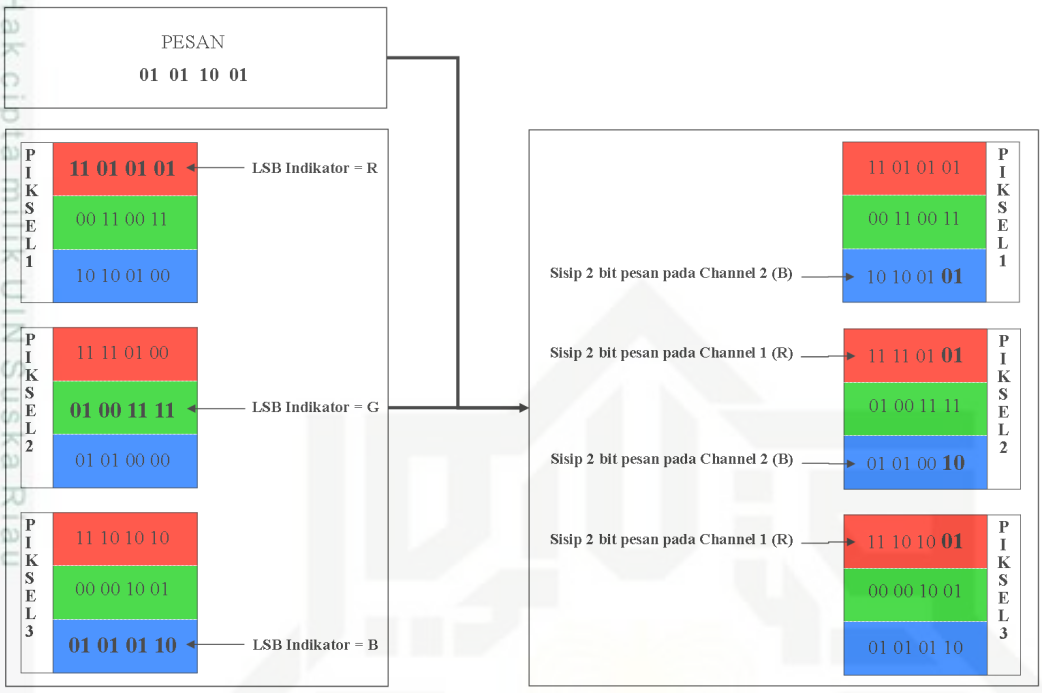
Secara garis besar dalam sistem steganografi memiliki tiga bagian penting, yakni : Kebutuhan data, proses, dan hasil. Seperti yang terlihat pada Gambar 4.1 berikut :



Gambar 4.1 Gambaran Umum Sistem Steganografi

1. Kebutuhan data terdiri dari dua bagian, kebutuhan data penyisipan *(embedding)* dan kebutuhan data pengekstraksian *(retrieving)*. Data penyisipan yang dibutuhkan yaitu pesan itu sendiri dan media penyisipan pesan *(carrier file)*. Sedangkan data pengekstraksian yang dibutuhkan tentunya yaitu media hasil penyisipan pesan *(stego file)*.
2. Proses, terbagi ke dalam dua bagian yaitu proses penyisipan dan pengekstraksian pesan. Setiap algoritma yang ada digunakan dalam kedua proses tersebut.
3. Hasil, terdapat dua hasil atau keluaran dari kedua proses yang dilakukan, hasil dari proses penyisipan yaitu *Stego File* (media yang sudah berisi pesan rahasia), kemudian hasil dari proses pengekstraksian yaitu pesan (yang sebelumnya disisipkan pada *carrier file.*

Pada penelitian ini, pesan akan disisipkan pada 2 bit LSB terakhir pada media penyisipan *(carrier file)* agar perbedaan yang terjadi antara gambar asli dan gambar hasil penyisipan tidak terlihat signifikan.



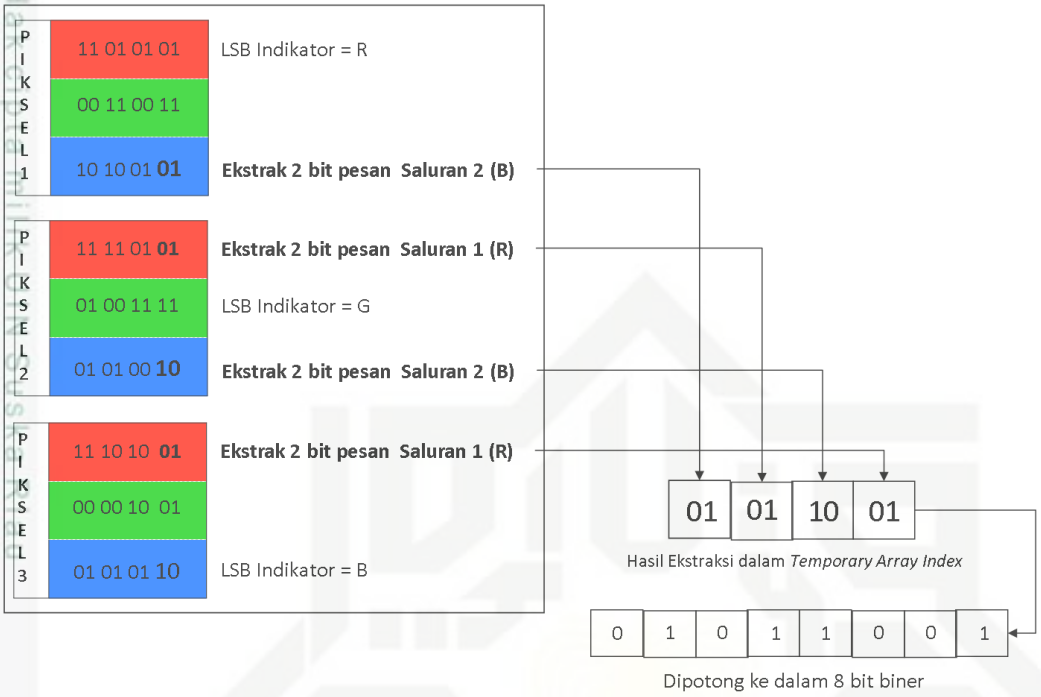
Gambar 4.2 Diagram Proses Embedding PIT

Dari Gambar 4.2 dapat dijelaskan langkah-langkah dalam proses *embedding* sebagai berikut :

1. Umpamakan terdapat *carrier image* yang terdiri dari 3 piksel dengan 3 saluran warna, yaitu *Red, Green*, dan *Blue*. Kemudian pesan karakter dengan nilai biner Y = 01011001 yang akan disisipkan pada piksel-piksel di dalam *carrier image.*
2. Pada piksel pertama, saluran indikator adalah saluran *Red* yang nilai pikselnya = 11 01 01 01 (LSB indikator “01”). Piksel kedua, saluran indikator adalah *Green* yang nilai pikselnya = 01 00 11 11 (LSB indikator “11”). Sedangkan piksel ketiga, saluran indikator adalah *Blue* yang nilai pikselnya = 01 01 01 10 (LSB indikator “10”).
3. Dalam proses penyisipan, pada piksel 1, LSB indikator di saluran *Red* bernilai “01” yang menandakan ada 2 bit pesan yang disisipkan, yakni pada saluran *Blue.* Dengan begitu, nilai *biner* pada saluran *Blue* yang sebelumnya adalah “10 10 01 00” berubah menjadi “10 10 01 **01**”. Antrian pesan saat ini tersisa “01 10 01”.

4. Pada piksel 2, LSB indikator di saluran *Green* bernilai "11" yang menandakan ada 4 bit pesan yang disisipkan, yakni masing-masing 2 bit pada saluran *Red* dan 2 bit pada saluran *Blue.* Sehingga, nilai *biner* pada saluran *Red* yang sebelumnya adalah "11 11 01 00" berubah menjadi "11 11 01 **01**", dan nilai biner pada saluran *Blue* yang sebelumnya adalah "01 01 00 00" berubah menjadi "01 01 00 **10**". Antrian pesan saat ini tersisa "01".
5. Selanjutnya pada piksel 3, LSB indikator di saluran *Blue* bernilai "10" yang menandakan ada 2 bit pesan yang disisipkan, yakni pada saluran *Red.* Sehingga, nilai *biner* pada saluran *Red* yang sebelumnya adalah "11 10 10 10" berubah menjadi "11 10 10 **01**". Antrian pesan sudah habis, artinya stego file sudah berhasil disisipi pesan.
6. Apabila antrian pesan masih ada, maka penyisipan dilanjutkan ke 4,5, dan seterusnya (sampai antrian habis) dengan urutan indikator kembali ke *Red, Green,* dan *Blue*.

Selanjutnya pada proses ekstraksi pesan dapat dilihat pada Gambar 4.3 :



Gambar 4.3 Diagram Proses Ekstraksi PIT

Dari Gambar 4.3 dapat dijelaskan sebagai berikut :

1. Pada proses ekstraksi *stego image*, saluran indikator piksel 1 adalah *Red* yang nilai binernya "11 01 01 01". Dengan LSB indikator "01" dari saluran *Red*, menandakan ada 2 bit pesan yang tersembunyi pada saluran *Blue* yang binernya "10 10 01 01" yakni bit "01". Ekstrak 2 bit pesan dan posisikan pada *temporary array index.*
2. Pada piksel 2, saluran indikator adalah saluran *Green* yang nilai binernya "01 00 11 11". Dengan LSB indikator "11" dari saluran *Green*, menandakan ada 4 bit pesan yang tersembunyi masing-masing 2 bit pada saluran *Red* dan 2 bit pada saluran *Blue.* Nilai biner saluran *Red* adalah "11 11 01 01" dengan bit pesan tersembunyi adalah "01". Nilai biner saluran *Blue* adalah "01 01 00 10" dengan bit pesan tersembunyi adalah "10". Ekstrak masing-masing 2 bit pesan dari saluran *Red* dan 2 bit pesan dari saluran *Blue*, kemudian posisikan pada *temporary array index*.
3. Kemudian pada piksel 3, saluran indikator adalah saluran *Blue* yang nilai binernya "01 01 01 10". Dengan LSB indikator "10" dari

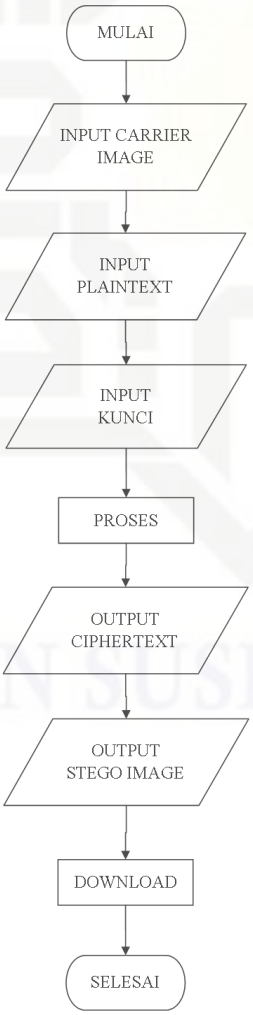
saluran *Blue,* menandakan ada 2 bit pesan yang tersembunyi pada saluran *Red* yang binernya "11 10 10 01" yakni bit "01". Ekstrak 2 bit pesan dan posisikan pada *temporary array index.*

4. Bit-bit pesan yang sudah diposisikan pada *temporary array index,* dipisah menjadi 8 bit biner dan dikonversi ke dalam pesan karakter kembali yakni 01011001 = Y.

4.3 *Flowchart* Sistem

Sistem terbagi ke dalam dua *flowchart, flowchart encrypt and embed,* serta *flowchart retrieve and decrypt*.

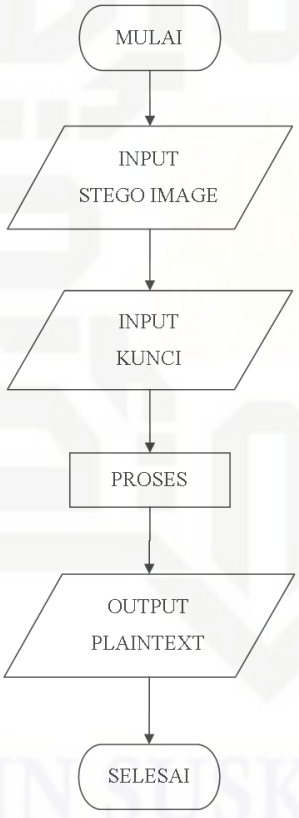
4.3.1 *Flowchart Encrypt and Embed*



Gambar 4.4 *Flowchart Encrypt and Embed*

Alur kerja sistem pada menu *Encrypt and Embed* bermula saat pengguna menginput terlebih dahulu *carrier image* yang akan disisipi pesan, kemudian input pesan teks yang akan dienkripsi. Lalu pengguna menginput kunci enkripsi. Sistem memproses hasil enkripsi sekaligus kemudian menyisipkan pesan hasil enkripsi ke dalam *carrier image*. Setelah selesai, sistem akan menampilkan hasil ciphertext dan *stego image*. Terakhir, pengguna men-*download stego image*.

4.3.2 *Flowchart Retrieve and Decrypt*



Gambar 4.5 *Flowchart Retrieve and Decrypt*

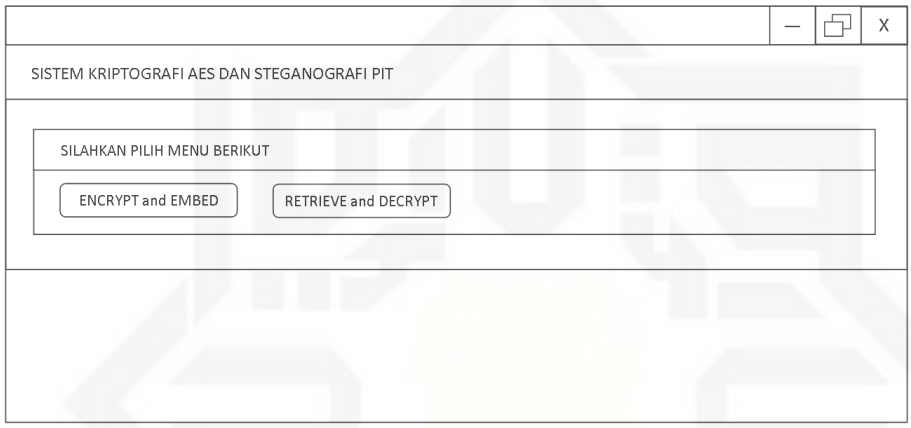
Pada menu *retrieve and decrypt*, alur kerja sistem dimulai bermula saat pengguna menginput *stego image* yang sudah disisipi pesan, kemudian pengguna menginput kunci dekripsi yang sama dengan kunci saat proses enkripsi. Sistem memproses hasil ektraksi pesan dan dekripsi

pesan. Setelah selesai, sistem akan menampilkan pesan asli *(plaintext)* yang telah diproses sebelumnya.

4.4 Perancangan Antarmuka Sistem

4.4.1 Perancangan Antarmuka Beranda

Berikut adalah rancangan antarmuka beranda sistem :

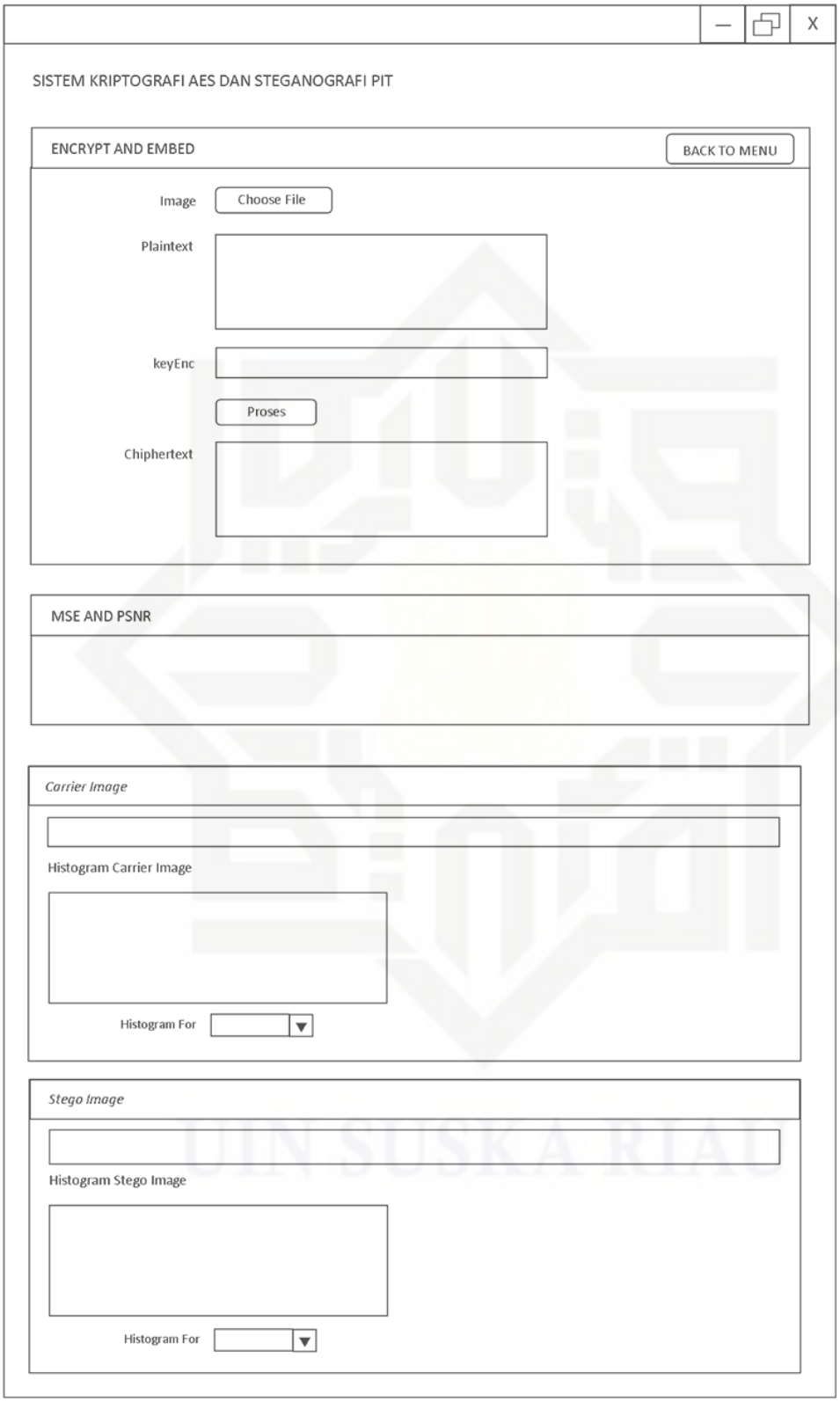


Gambar 4.6 Antarmuka Beranda Sistem

Dari Gambar 4.4 dapat dijelaskan dalam sistem ini, pengguna dihadapkan pada dua pilihan *button, button* 1 adalah ”*Encrypt and Embed”* dan *button* 2 adalah “*Retrieve and Decrypt*”. Pada *button* 1, pengguna nantinya akan dibawa ke halaman untuk proses pengenkripsian dan penyisipan pesan. Sedangkan pada *button* 2, pengguna akan dibawa pada halaman untuk proses pengekstraksian dan pendekripsian pesan.

4.4.2 Perancangan Antarmuka Menu *Encrypt and Embed*

Berikut adalah tampilan rancangan antarmuka dari menu pengenkripsian dan penyisipan :

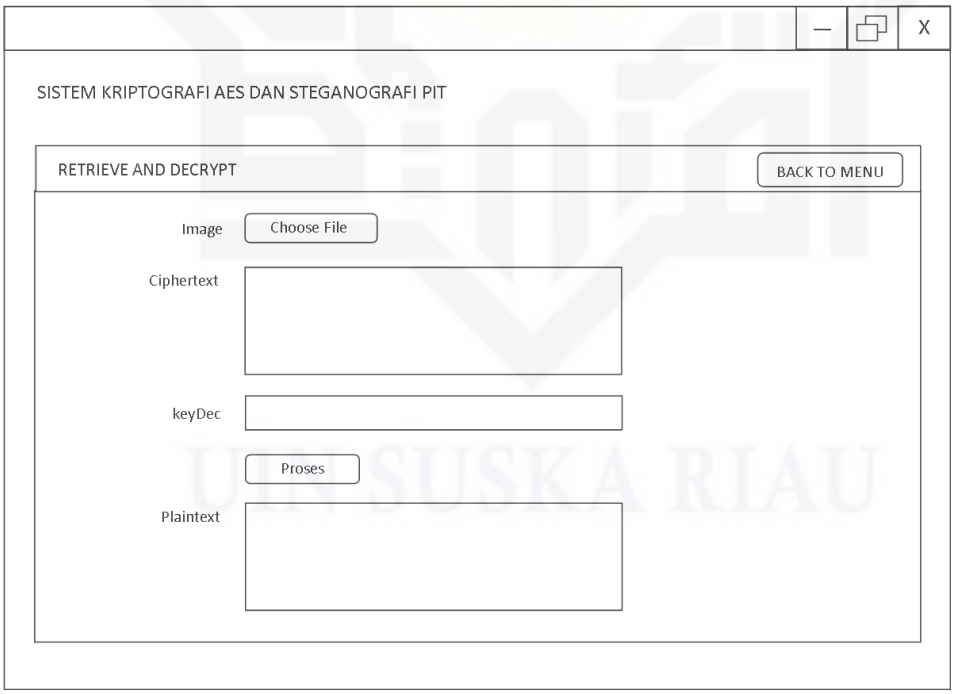


Gambar 4.7 Antarmuka Menu *Encrypt and Embed*

Dari Gambar 4.5 dijelaskan bahwa setelah pengguna masuk ke dalam menu *Encrypt and Embed,* pada kolom pertama akan terlihat *upload button* untuk mulai menginput *carrier image* yang akan disisipi pesan. Sesaat setelah *carrier image* diinput, gambar yang diinput langsung muncul pada kolom *Carrier Image* bersamaan dengan grafik histogramnya. Setelah itu, masukkan pesan teks pada kolom *text* yang nantinya akan dienkrip dan kemudian langsung disisipi ke dalam *carrier image*.

Tekan *button* Proses untuk memulai pengenkripsian dan penyisipan pesan ke dalam gambar. Setelah proses selesai, *stego image* muncul di kolom *Stego Image* bersamaan dengan grafik histogramnya. Terakhir, *download stego image* untuk disimpan ke dalam komputer pengguna.

4.4.3 Perancangan Antarmuka Menu *Retrieve and Decrypt*



Gambar 4.8 Antarmuka Menu *Retrieve and Decrypt*

Pada proses *retrieve and decrypt*, pengguna menginputkan *stego image* dengan mengklik *button* pada kolom *Image*. Kemudian masukkan *key* pada

kolom *keyDec. Key* yang dimasukkan harus sama dengan saat pengguna melakukan proses enkripsi sebelumnya. Klik *button* Proses, *ciphertext* akan muncul pada kolom *Ciphertext* bersamaan dengan *plaintext* di kolom *Plaintext.*