BAB III
METODE PENELITIAN

3.1 Metodologi Penelitian

Algoritma penyematan yang digunakan merupakan modifikasi dari teknik-teknik sebelumnya dimana perbedaan terbesarnya terletak pada proses penyematan pesan. Pada metode yang digunakan, nilai komponen piksel (*RED, GREEN, BLUE*) yang dijadikan media untuk menyisipkan pesan tidak berubah.

Pesan terlebih dahulu dikonversi menjadi data biner. Setelah didapat data biner dari pesan, kemudian dilakukan proses pencarin lokasi nilai bit identik antara bit pesan dengan bit piksel.

Proses pencarian dilakukan dengan membandingkan bit biner pertama dari pesan/ bit paling kiri (MSB) dengan bit biner hasil konversi nilai komponen warna *RED* yang terdapat pada piksel *cover-image* dimulai dari indeks terkecil/ bit paling kiri (MSB). Apabila telah didapat nilai yang sama, koordinat piksel serta indeks lokasi bit biner komponen warna *RED* yang memiliki nilai sama dengan pesan dicatat. Pada penelitian ini digunakan kesepakatan jika indeks terkecil *array* adalah indeks ke-1 (mengacu pada sistem penomoran indeks pada Matlab).

Pada proses kedua, bit selanjutnya dari pesan dilakukan proses pencarian yang sama namun pembandingan dilakukan dengan nilai komponen warna *GREEN*. Kemudian pada proses ketiga, bit selanjutnya dari pesan dibandingkan dengan nilai komponen warna *BLUE*. Proses kemudian diulang kembali dengan cara yang sama dengan pembandingan dimulai kembali dari komponen warna *RED*, *GREEN*, kemudian *BLUE* sampai didapat semua catatan lokasi bit pada *cover-image* yang memiliki nilai yang sama dengan bit pesan.

Sebagai contoh apabila bit {0,0,1} akan disematkan ke dalam sebuah piksel, pertama buat *array* yang berkapastas sebanyak jumlah data, kemudian masukkan data kedalam *array*, A[]= {0,0,1}. Setelah itu bit pertama dari *array* A={0} dibandingkan dengan komponen warna merah (*RED*) pada piksel, bit kedua dari *array* A={0} dibandingkan dengan komponen warna hijau pada piksel (*GREEN*), dan bit ketiga dari *array* A={1} dibandingkan dengan komponen warna biru pada piksel (*BLUE*).

Apabila pada saat pembandingan ditemukan kesamaan nilai antara bit pesan dan bit pada piksel, lokasi kolom, baris, komponen warna, serta indeks letak bit piksel tersebut

dicatat sebagai data lokasi. Proses ini berlangsung sampai semua bit pesan telah berhasil menemukan kesamaan nilai dengan bit pada piksel, serta data lokasinya.

Misalkan setelah proses pembandingan didapat hasil seperti gambar berikut :

		[0]	[1]
Piksel (0,0)	11111111	11110110	10111010
	RED	GREEN	BLUE

	[0]		
Piksel (0,1)	10111111	10001101	10111011
	RED	GREEN	BLUE

Gambar 3.1 Hasil Pencarian Nilai Bit Identik.

Maka akan didapatkan data lokasi :

1) bit pertama *array* A sesuai dengan bit pada *cover-image* koordinat piksel baris ke 0 kolom ke 1, koordinat piksel = (0,1) komponen warna *RED* (R) indeks ke 2 (indeks terkecil *array* adalah 1).
2) bit kedua *array* A sesuai dengan bit pada *cover-image* koordinat piksel = (0,0) komponen warna *GREEN* (G) indeks ke 5.
3) bit ketiga *array* A sesuai dengan bit pada *cover-image* koordinat piksel = (0,0) komponen warna *BLUE* (B) indeks ke 1.

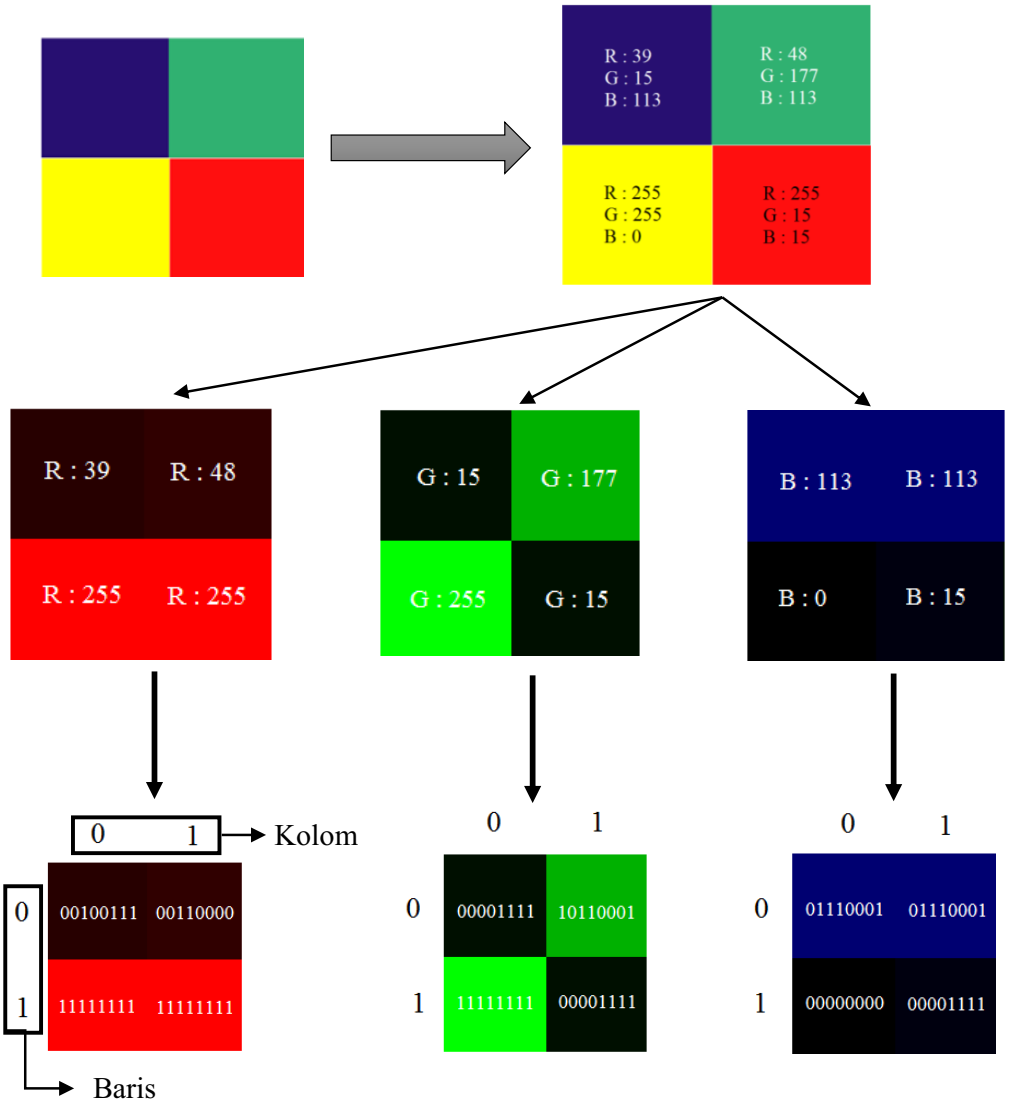
Sehingga didapat data lokasi R(1,0) 2 ; G(0,0)5 ; B(0,0)1. Data lokasi yang didapat kemudian disimpan pada bagian *Comment Segment* pada citra.

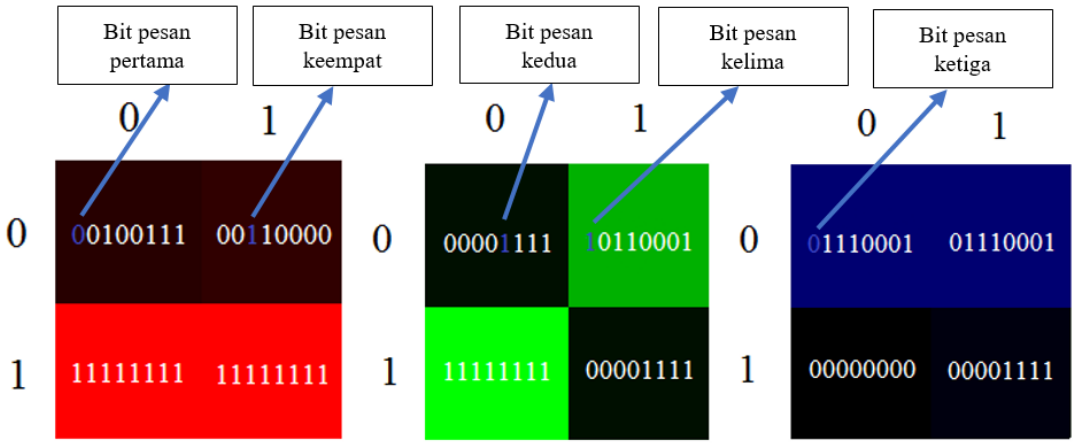
Penggunaan algoritma ini memiliki kesempatan yang lebih baik untuk mempersulit para *steganalysis* yang menggunakan metode deteksi visual untuk mendeteksi adanya *stego-image*. Hal ini dikarenakan algoritma yang digunakan tidak mengakibatkan perubahan nilai piksel yang dimiliki citra penampung (*cover-image*) sama sekali.

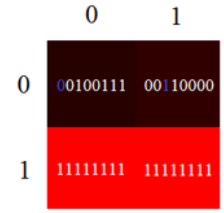
Berikut merupakan simulai penyematan pesan pada citra secara singkat :

Misalkan penyematan pesan dengan bit 01011, A = {0,1,0,1,1} pada citra JPEG ukuran 2x2 piksel dibawah ini :

Citra JPEG ukuran 2x2 piksel

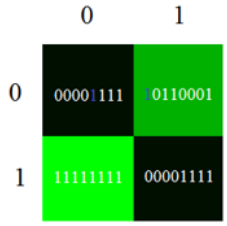






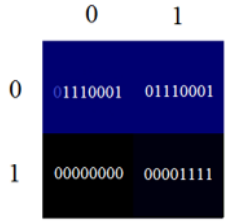
KOMPONEN WARNA RED

(baris,kolom)	Indeks							
	1	2	3	4	5	6	7	8
0,0	0	0	1	0	0	1	1	1
0,1	0	0	1	1	0	0	0	0
1,0	1	1	1	1	1	1	1	1
1,1	1	1	1	1	1	1	1	1



KOMPONEN WARNA GREEN

(baris,kolom)	Indeks							
	1	2	3	4	5	6	7	8
0,0	0	0	0	0	1	1	1	1
0,1	1	0	1	1	0	0	0	1
1,0	1	1	1	1	1	1	1	1
1,1	0	0	0	0	1	1	1	1



KOMPONEN WARNA BLUE

(baris,kolom)	Indeks							
	1	2	3	4	5	6	7	8
0,0	0	1	1	1	0	0	0	1
0,1	0	1	1	1	0	0	0	1
1,0	0	0	0	0	0	0	0	0
1,1	0	0	0	0	1	1	1	1

Data lokasi pesan : R(0,0)1 ; G(0,0)5 ; B(0,0)1 ; R(0,1)3; G(0,1)1
Setelah data lokasi didapat, kemudian data lokasi disisipkan ke bagian *Comment Segment cover-image*.

3.2 Studi Literatur

Pada studi literatur dijelaskan mengenai teori pendukung yang relevan dan dapat menjadi panduan dalam melaksanakan penelitian. Studi literatur ini sudah dijelaskan pada bab 2. Dengan adanya studi literatur diharapkan memperlancar implementasi sistem.

3.3 Analisa Kebutuhan

Perangkat yang digunakan untuk menunjang perancangan dan pengujian sistem antara lain :

A. Perangkat Keras

1) Laptop Lenovo G400s

Prosesor : Intel® Core™ i5-3230M CPU 2.60 GHz

RAM : 4.00 GB

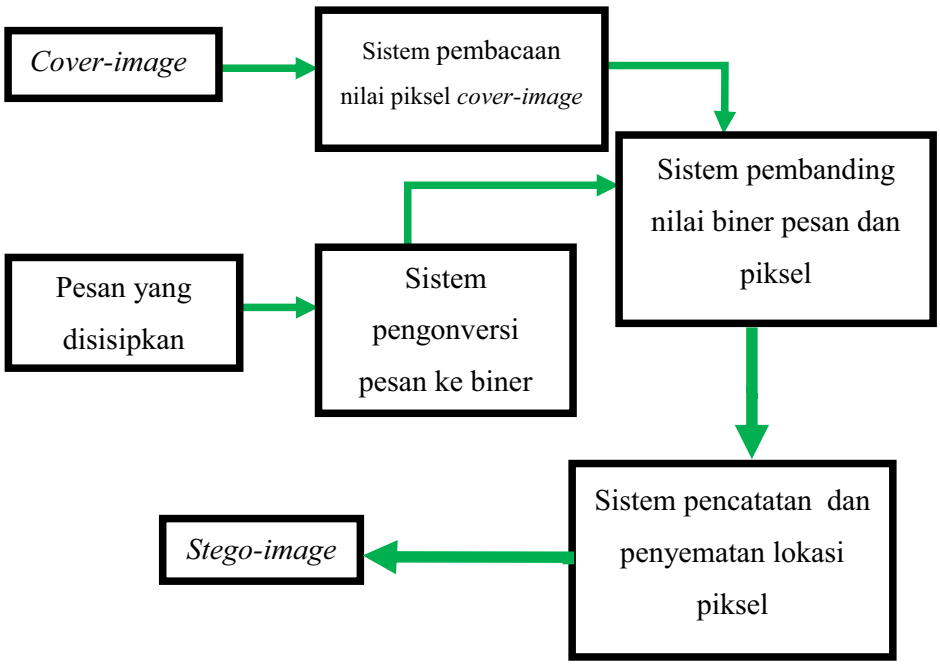
VGA : Nvidia GeForce 720M 2 GB

B. Perangkat Lunak

1) OS Windows 10 Pro 64-bit
2) Matlab R2016a 64-bit

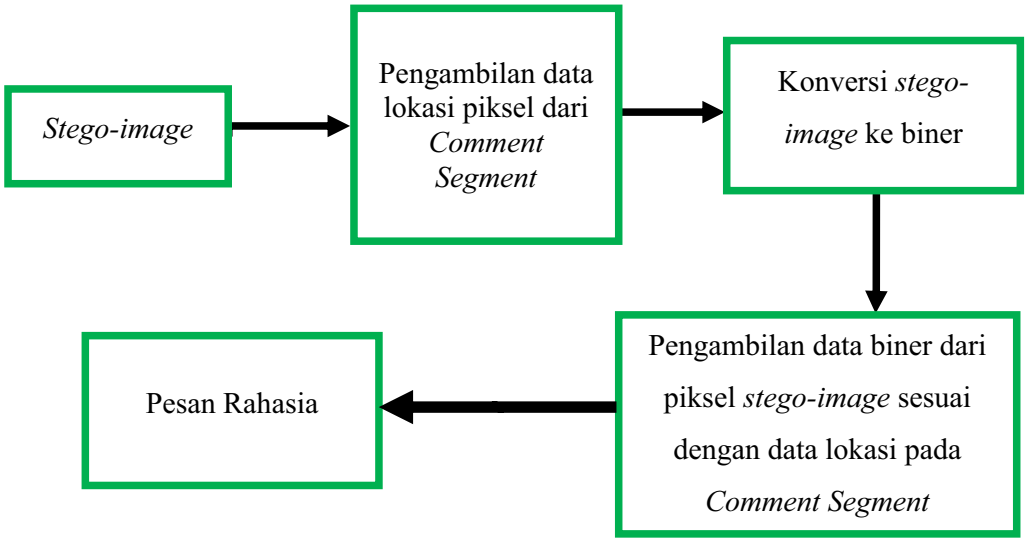
3.4 Metode Penyematan Secara Umum

Metode penyematan pesan rahasia yang diusulkan dapat digambarkan dengan model pada gambar 3.3



Gambar 3.2 Blok Diagram Sistem Penyisipan Pesan

Proses ekstraksi pesan dari *stego-image* digambarkan dengan model pada gambar berikut :



Gambar 3.3 Blok Diagram Proses Ekstraksi Pesan

Berikut merupakan penjelasan konfigurasi sistem :

1) Sistem pembacaan nilai piksel :

   Pada bagian ini, nilai piksel *cover-image* dibaca. Pembacaan meliputi pembacaan komponen *RED*, komponen *GREEN*, Komponen *BLUE*. Kemudian nilai yang terbaca dikonversikan ke dalam bentuk biner.

2) Sistem pengonversi pesan rahasia ke dalam bentuk biner :

   Pada bagian ini pesan rahasia yang akan disematkan dirubah kedalam bentuk biner. Bit-bit biner tersebut nantinya dijadikan sebagai salah satu masukan sistem pembandingan nilai piksel dengan pesan rahasia.

3) Sistem pembandingan nilai piksel denga pesan rahasia

   Pada bagian ini bit-bit biner antara *cover-image* dan pesan rahasia dibandingkan. Pembandingan nilai bertujuan untuk mencari lokasi bit biner pada piksel yang memiliki nilai bit biner sama dengan pesan rahasia. Proses pembandingan ini dilakukan dengan cara pembandingan per bit. Hasil keluaran dari proses pembandingan adalah data lokasi sebaran nilai-nilai bit biner pesan yang memiliki nilai identik dengan nilai bit biner komponen warna piksel pada *cover-image*.

4) Pencatatan dan penyematan data lokasi piksel

   Setelah ditemukan nilai bit-bit identik antara bit pesan dan bit komponen warna piksel melalui proses pembandingan, data lokasi sebaran bit-bit biner identik yang didapatkan dicatat. Data lokasi tersebut meliputi data lokasi kolom, baris, komponen warna, serta indeks lokasi. Data lokasi yang didapat kemudian disematkan pada bagian *Comment Segment* pada *cover-image* citra JPEG. Setelah proses penyematan data lokasi akan didapat citra keluaran berupa *stego-image*. Data lokasi yang terdapat pada bagian *Comment Segment* (*key*) nantinya digunakan dalam proses ekstraksi pesan.

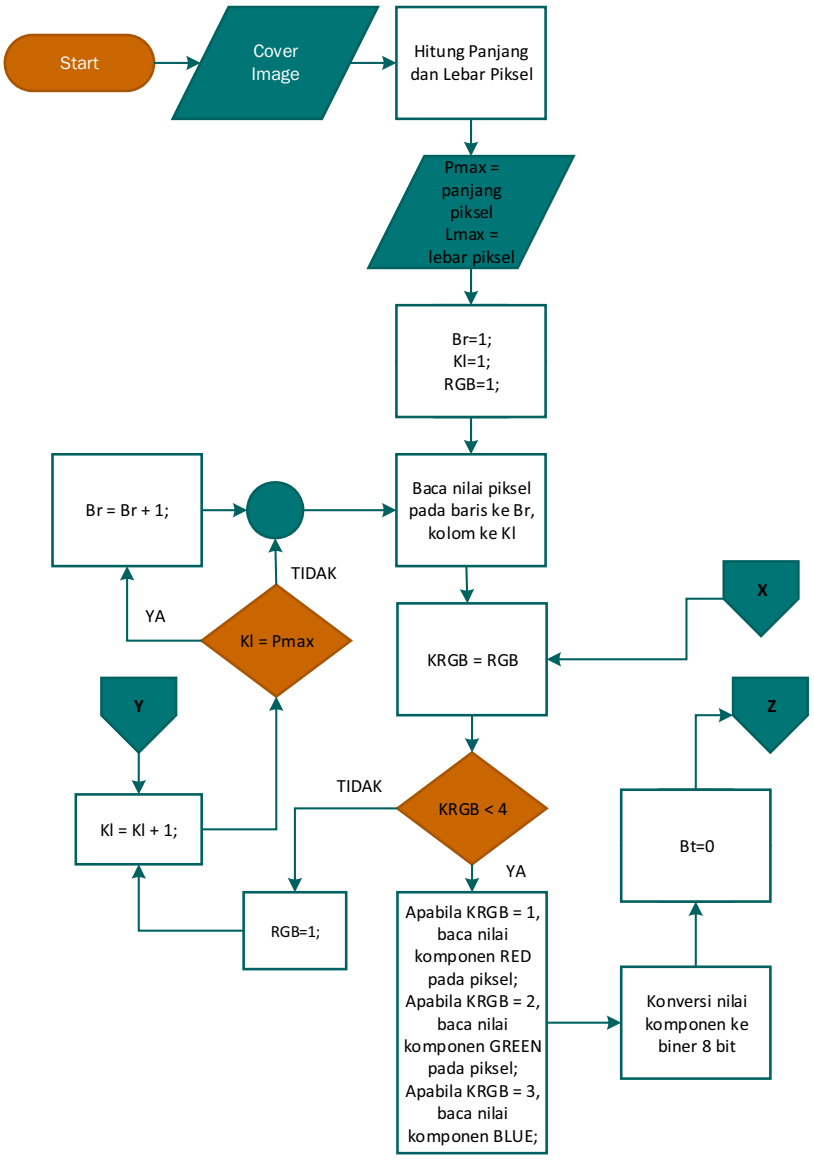
5) Ekstraksi pesan dari *stego-image*

   Pada proses ekstraksi, data lokasi (*key*) yang berada pada *Comment Segment stego-image* diambil dan diproses yang kemudian menghasilkan keluaran berupa pesan rahasia.

3.5 Perancangan Sistem

Pada bagian ini dilakukan perancangan sistem lebih mendetail. Metode penyematan secara umum yang telah dibuat pada tahap sebelumnya akan dirancang labih lanjut. Perancangan sistem meliputi perancangan proses pembacaan nilai piksel, perancangan proses konversi pesan rahasia ke biner, perancangan proses pembanding dan penyematan, perancangan proses ekstraksi pesan.

3.5.1 Perancangan Proses Pembacaan Nilai Piksel



Gambar 3.4 Flowchart Proses Pembacaan Nilai Piksel

Keterangan :

Br : Variabel yang menunjukkan posisi baris piksel.

Kl : Variabel yang menunjukkan posisi kolom piksel.

Pmax : Panjang maksimum piksel.

Lmax : Lebar maksimum piksel.

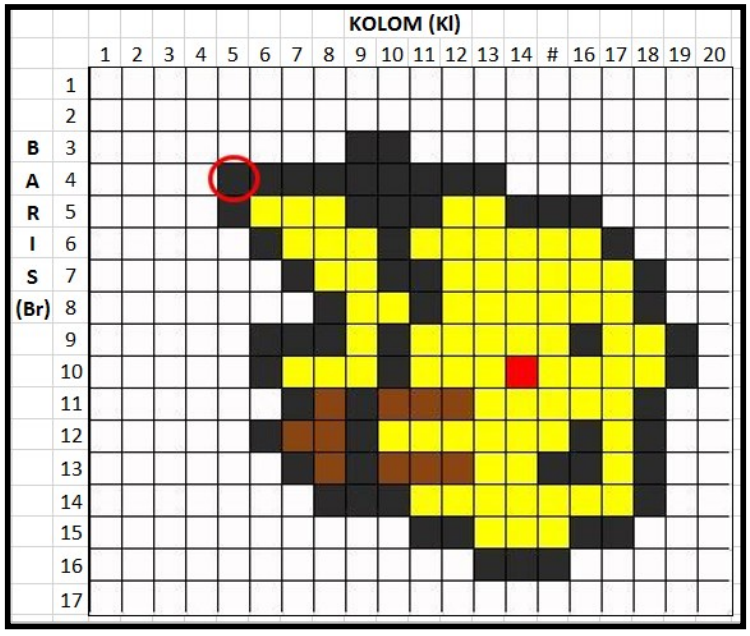
RGB : Variabel untuk pengaturan supaya pembacaan komponen piksel dimulai dari komponen *RED*.

KRGB : Variabel yang menentukan komponen piksel mana yang akan dibaca. Apakah komponen *RED*, *GREEN*, atau *BLUE*.

Bt : Variabel yang menentukan pada indeks keberapa bit pada pesan dibaca oleh proses.

Penjelasan urutan proses :

1) Proses pembacaan nilai piksel dimulai dari load *cover-image*, kemudian menghitung ukuran panjang dan lebar piksel pada *cover-image* untuk menentukan Pmax dan Lmax. Proses pencarian nilai bit biner identik antara bit pesan dan bit piksel dimulai dari piksel baris ke-1 kolom ke-1 (berdasarkan sistem pemetaan koordinat piksel yang dipakai Matlab), kemudian dilanjutkan ke piksel baris ke 1 kolom ke 2. Ketika proses pencarian bit biner telah sampai pada baris ke n dan kolom ke Pmax, maka proses pencarian selanjutnya akan dimulai kembali dari baris ke (n+1) dan kolom ke 1.
2) Pemberian nilai awal Br, Kl, dan RGB=1. Br dan Kl diberi nilai awal 1 supaya pencarian bit biner identik dimulai dari piksel baris (Br) ke 1 dan kolom (Kl) ke 1. RGB diberi nilai awal 1 supaya pencarian bit biner identik dimulai dari komponen warna *RED* terlebih dahulu.
3) Proses selanjutnya adalah pembacaan nilai piksel yang lokasinya ditentukan oleh nilai variabel Br dan Kl. Misalkan nilai Br adalah 4 dan Kl adalah 5, maka nilai piksel yang harus dibaca oleh proses adalah piksel yang berada pada urutan baris ke 4 dan kolom ke 5 pada *cover-image*.



Gambar 3.5 Piksel Br ke 4 Kl ke 5

4) Memasukkan nilai variabel RGB ke KRGB, lihat apakah nilai KRGB < 4

Kondisi *false* :

- Ubah nilai variabel RGB = 1. Tambah nilai Kl, Kl = Kl +1 (Nilai variabel Kl ditambah dengan 1, masukkan hasilnya ke variabel Kl).
- Lihat apakah nilai variabel Kl sama dengan nilai variabel Pmax

- Jika tidak langsung kembali ke proses baca nilai piksel pada baris ke Br kolom ke Kl
- Jika ya tambahkan terlebih dahulu nilai variabel Br dengan 1 (Br = Br + 1) , kemudian kembali ke proses baca nilai piksel pada baris ke Br kolom ke Kl.

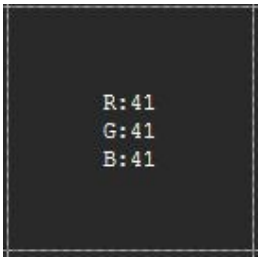
Kondisi *true* :

- Lihat nilai KRGB

- Apabila nilai KRGB = 1, maka komponen warna piksel yang dibaca proses adalah komponen *RED*
- Apabila nilai KRGB = 2, maka komponen warna piksel yang dibaca proses adalah komponen *GREEN*
- Apabila nilai KRGB = 3, maka komponen warna piksel yang dibaca proses adalah komponen *BLUE.*

Setelah nilai komponen warna dibaca, proses selanjutnya adalah mengonversi nilai komponen warna ke biner 8 bit. Kemudian set nilai Bt = 0 (nilai indeks terkecil pada *array*) kemudian masuk ke proses C. Misalkan nilai KRGB = 1, maka baca nilai komponen piksel *RED*.

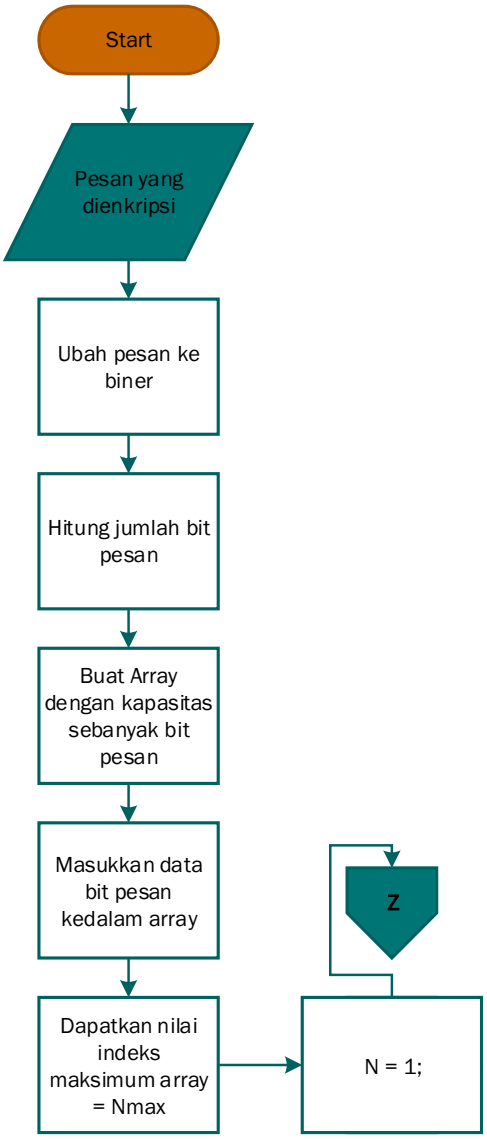
Pada baris ke Br = 4 kolom ke Kl = 5



Gambar Piksel baris ke Br = 4 kolom ke Kl = 5

Didapatkan nilai komponen *RED* (R) = 41, kemudian nilai yang didapat dikonversikan ke biner 8 bit. Maka didapat *array* biner 00101001. Kemudian masuk ke proses **Z**.

3.5.2 Perancangan Proses Konversi Pesan Rahasia ke Biner



Gambar 3.6 Flowchart Proses Konversi Pesan Rahasia ke Biner

Penjelasan urutan proses :

1) Proses konversi pesan rahasia ke biner dimulai dari pengubahan pesan yang telah dienkripsi ke dalam bentuk biner, proses konversi tersebut beracuan pada tabel ASCII. Misalnya sebuah pesan yang dienkripsi adalah H5, setelah proses pengubahan ke biner akan menjadi :

   H ke desimal = 72, 72 ke biner = 0100 1000

   5 ke desimal = 5, 5 ke biner = 0000 0101

   Setelah melalui proses konversi, akan didapatkan pesan dalam bentuk biner = 0100 1000 0000 0101

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciichars.com

Gambar 3.7 ASCII Tabel

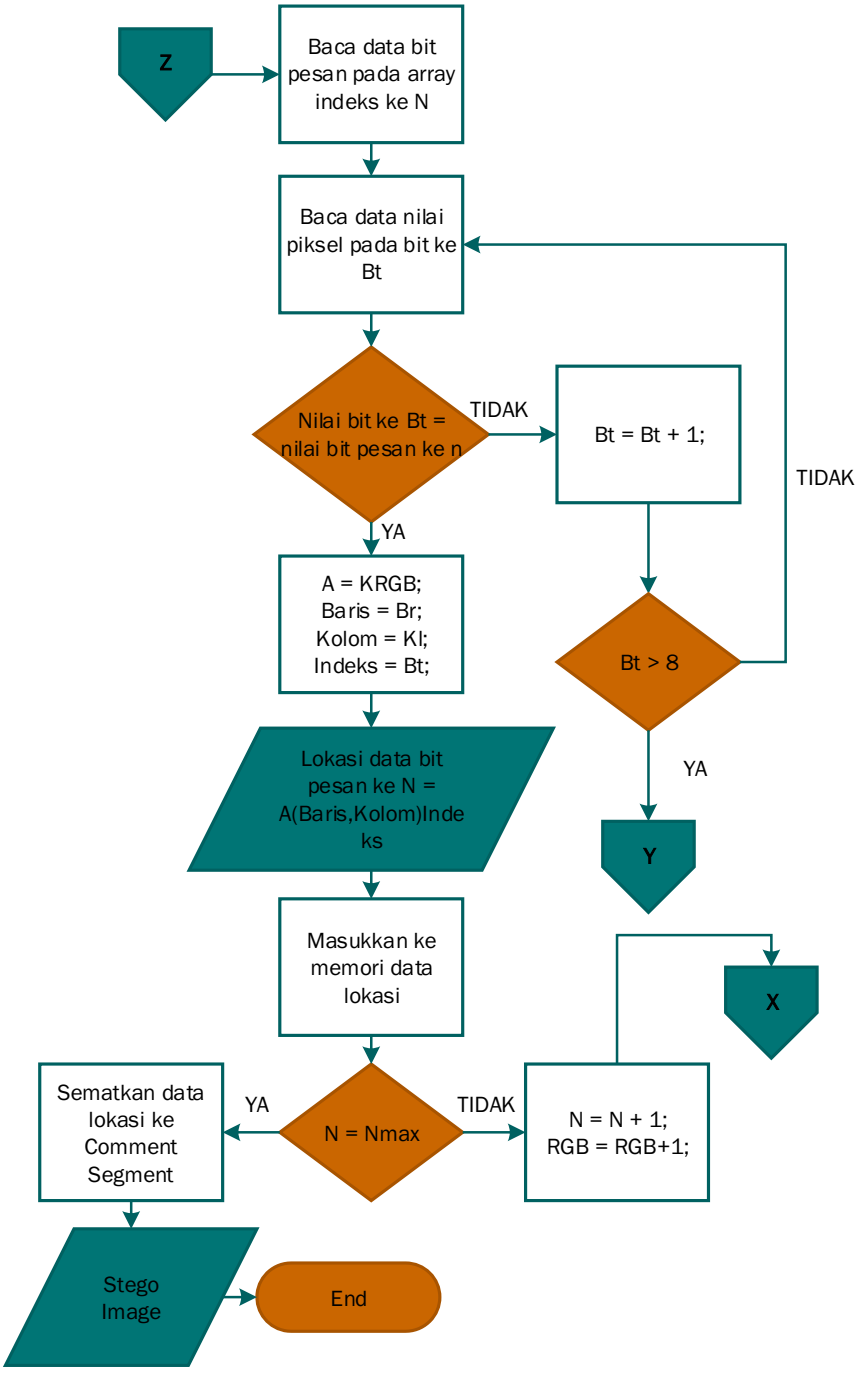
2) Pesan yang telah dirubah ke dalam bentuk biner kemudian dihitung jumlah bit nya. Dari pesan 0100 1000 0000 0101 didapatkan hasil 16 bit.
3) Pembuatan *array* yang berukuran sesuai bit pesan.

   Pesan [ ] = { 0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,1}.

Pesan	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1
Indeks ke-	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

   Nilai indeks maksimum = 15
4) Beri nilai variabel N = 1, pemberian nilai ini berfungsi untuk setting supaya pembacaan *array* dimulai dari indeks terkecil (pada Matlab indeks terkecil adalah indeks ke-1).
5) Masuk ke proses **Z**.

3.5.3 Perancangan Proses Pembandingan dan Penyematan



Gambar 3.8 Flowchart Proses Pembandingan dan Penyematan

Penjelasan urutan proses :

1) Dari proses pembacaan nilai piksel dan proses konversi pesan rahasia ke biner, selanjutnya masuk ke proses pembandingan dan penyematan. Proses pembandingan dan

penyematan dimulai dari baca data bit pesan pada *array* indeks ke N, kemudian baca data nilai piksel pada bit ke Bt.

2) Selanjutnya nilai bit pesan pada *array* indeks ke N dan nilai piksel pada bit ke Bt dibandingkan. (Apakah Nilai bit pesan pada *array* indeks ke N = nilai piksel pada bit ke Bt ?)

Kondisi *True*

1) Apabila nilai bit pesan pada *array* indeks ke N = nilai piksel pada bit ke Bt. Salin nilai KRGB ke A (A = KRGB), nilai Br ke Baris (Baris = Br), nilai Kl ke Kolom (Kolom = Kl), nilai Bt ke Indeks (Indeks = Bt).
2) Didapat lokasi data bit pesan ke N = A(Baris, Kolom) Indeks, kemudian masukkan ke memori data lokasi. Misalkan didapat lokasi data bit pesan ke 1 = 2(6,7)2 maka, bit pesan ke 1 sama dengan nilai piksel baris ke 6, kolom ke 7, komponen warna *RED*, indeks ke 2.
3) Lokasi data bit pesan ke N kemudian disimpan ke memori data lokasi.
4) Apakah nilai N=Nmax

   Kondisi *True*

   1) Data lokasi disematkan ke *Comment Segment cover-image* JPEG
   2) Didapatkan keluaran *stego-image*. Akhiri seluruh proses.

   Kondisi *False*

   1) Nilai N ditambah 1 (N = N+1), nilai RGB ditambah 1 (RGB = RGB+1).
   2) Kembali ke proses **X.**

Kondisi *False*

1) Nilai Bt ditambah 1 (Bt = Bt+1), hal ini menunjukkan jika proses pembandingan dilanjutkan dengan data bit piksel pada indeks selanjutnya.
2) Bandingkan apakah nilai Bt > 8

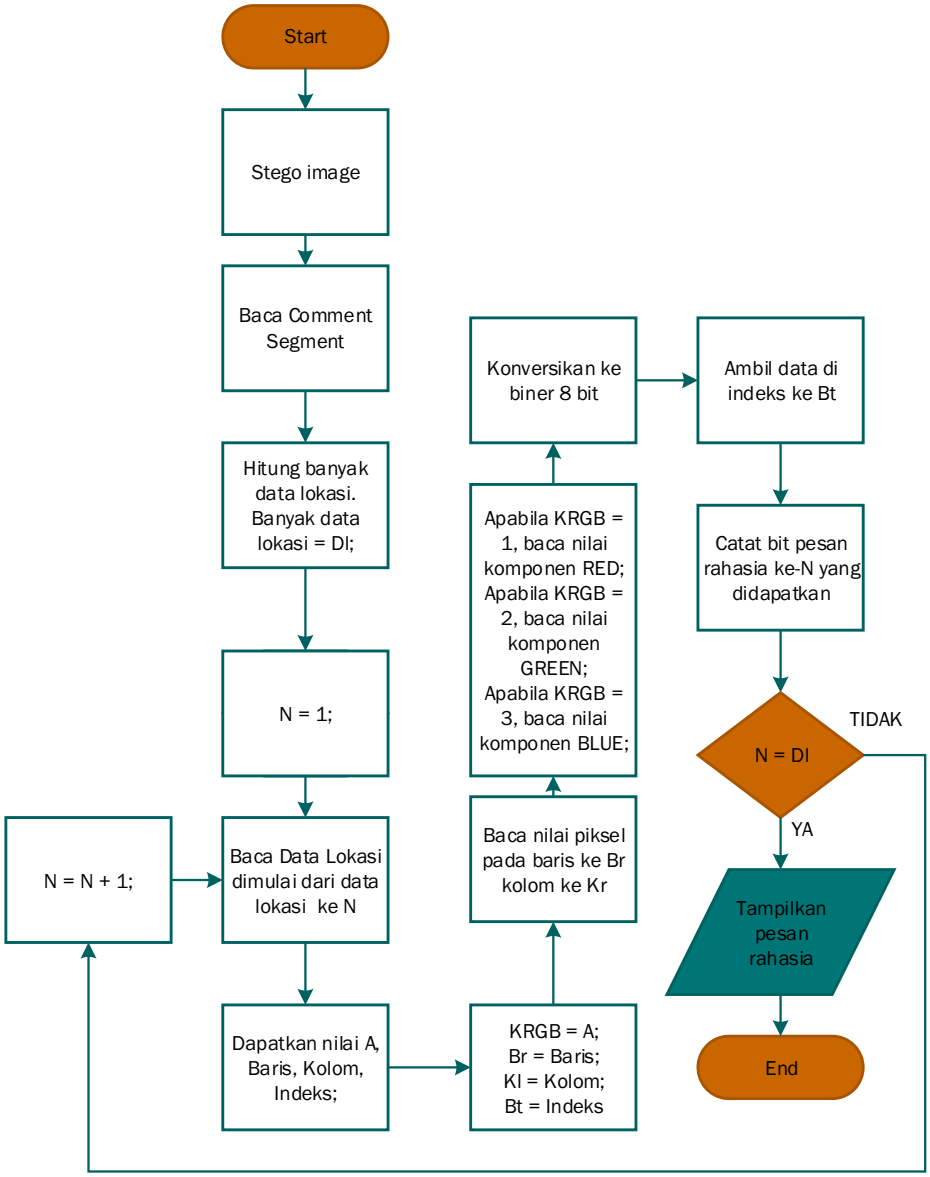
   Kondisi *True*

   Kembali ke proses **Y**

   Kondisi *False*

   Kembali ke proses baca data nilai piksel pada bit ke Bt.

3.5.4 Perancangan Proses Ekstraksi Pesan Rahasia dari *Stego-image*



Gambar 3.9 Flowchart Ekstraksi Pesan Rahasia dari *Stego-image*

Penjelasan urutan proses :

1) Load *stego-image*, kemudian baca *Comment Segment* yang memuat data lokasi piksel yang memiliki nilai identik dengan pesan.
2) Dapatkan nilai Dl dengan cara menghitung banyak data lokasi, banyak data lokasi = Dl.
3) Set nilai awal N = 1. Nilai awal N diset sama dengan nol supaya pembacaan data lokasi dimulai dari indeks terkecil.

4) Dapatkan nilai A, Baris, Kolom, dan Indeks. Salin nilai A ke KRGB (KRGB = A), nilai Baris ke Br (Br = Baris), nilai Kolom ke Kl (Kl = Kolom), Indeks ke Bt (Bt = Indeks).
5) Kemudian baca nilai piksel *cover-image* pada baris ke Br kolom ke Kl
6) Apabila nilai KRGB = 1, maka komponen warna piksel yang dibaca proses adalah komponen *RED*.

   Apabila nilai KRGB = 2, maka komponen warna piksel yang dibaca proses adalah komponen *GREEN*.

   Apabila nilai KRGB = 3, maka komponen warna piksel yang dibaca proses adalah komponen *BLUE*.
7) Mengonversikan nilai komponen yang dibaca kedalam biner 8 bit.

   Misalkan komponen warna yang dibaca oleh sistem adalah komponen warna *RED* dengan nilai 233. Maka akan didapat data biner 8 bit hasil dari konversi = 1101 1111
8) Dari data biner 8 bit yang telah didapat, lihat nilai Bt kemudian ambil
   data pada indeks ke Bt. Kemudian catat data tersebut sebagai data pesan rahasis ke N.
9) Apakah nilai N = Dl ?

   Kondisi *True*

   Tampilkan seluruh pesan rahasia yang telah dicatat. Akhiri semua proses.

   Kondisi *False*

   Tambah nilai N dengan 1 (N = N+1), kemudian kembali ke proses baca data lokasi pada urutan ke N.

3.6 Pengujian Sistem

Pengujian sistem dilakukan untuk mengukur tingkat keberhasilan sistem yang telah dibuat, apakah bisa berjalan sesuai rancangan yang telah dibuat dan mencapai tujuan yang ingin dicapai dalam penelitian. Maka dari itu diperlukan serangkaian pengujian diantaranya

1) Pengujian fungsionalitas sistem.
2) Analisis kompleksitas algoritma
3) Pengujian karakteristik sistem.
4) Pengujian autokorelasi.

Pada pengujian fungsionalitas sistem, akan diuji apakah program yang telah dibuat dapat berjalan sesuai yang diharapkan. Analisis kompleksitas algoritma dilakukan untuk mengetahui pengaruh perubahan ukuran data terhadap tingkat pertumbuhan waktu eksekusi program. Pada pengujian karakteristik sistem akan diamati pengaruh variasi ukuran pesan serta ukuran citra terhadap *ciphertext* yang dihasilkan. Pada pengujian autokorelasi akan dilihat tingkat keacakan *key* yang dihasilkan sistem.

3.7 Analisa Hasil dan Kesimpulan

Pada tahap ini diambil kesimpulan dari hasil pengujian dan analisa terhadap sistem yang dibangun. Dari kesimpulan yang didapat, akan dihasilkan saran yang dapat digunakan pada penelitian selanjutnya sehingga dapat menyempurnakan kekurangan yang ada. Yang nantinya sangat bermanfaat dalam pengembangan pada tingkat pokok kajian yang lebih lanjut.