BAB III
METODOLOGI DAN PENELITIAN

Pada bab metodologi dan perancangan ini akan dibahas langkah-langkah metode perancangan yang akan digunakan dalam skripsi ini. Langkah-langkahnya adalah:

1. Mencari dan mempelajari literatur-literatur yang terkait dengan masalah enkripsi menggunakan metode *rijndael* dari buku maupun sumber lain dari internet.
2. Merancang sistem perangkat lunak dengan metode yang akan digunakan
3. Mempersiapkan data uji berupa data *plain text.*
4. Mengimplementasikan metode *rijndael* untuk mengenkripsi data *plain text* atau mendekripsi hasil enkripsi *rijndael*.
5. Melakukan uji ketahanan terhadap hasil enkripsi.
6. Mengevaluasi dan menganalisis hasil enkripsi yang dihasilkan oleh sistem.

Alur penelitian ditunjukkan pada gambar 3.1.

Gambar 3.1 Alur Penelitian

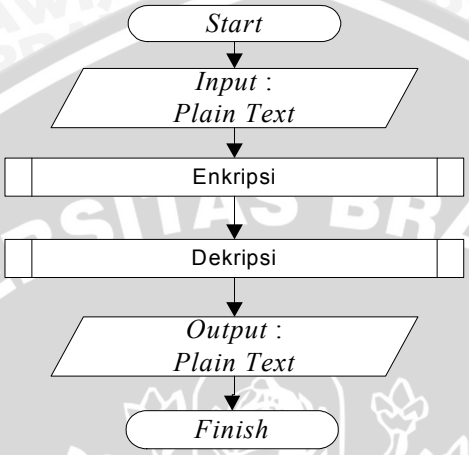
Sumber : perancangan

3.1 Deskripsi Sistem

3.1.1 Deskripsi Umum Sistem

Sistem ini dibangun untuk menjaga kerahasiaan data dengan cara menyisipkan sebuah kunci dan dienkripsi dengan menggunakan metode rijndael. Pada penelitian ini dibagi menjadi 2 bagian, yakni enkripsi dan dekripsi.

Berikut flowchart umum tentang Algoritma rijndael di tunjukan pada gambar di bawah



gambar

3.1.1 *Flowchart*

system

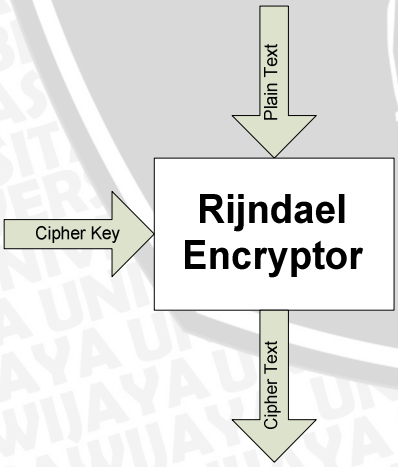
sumber :

perancangan

3.2 Desain Sistem

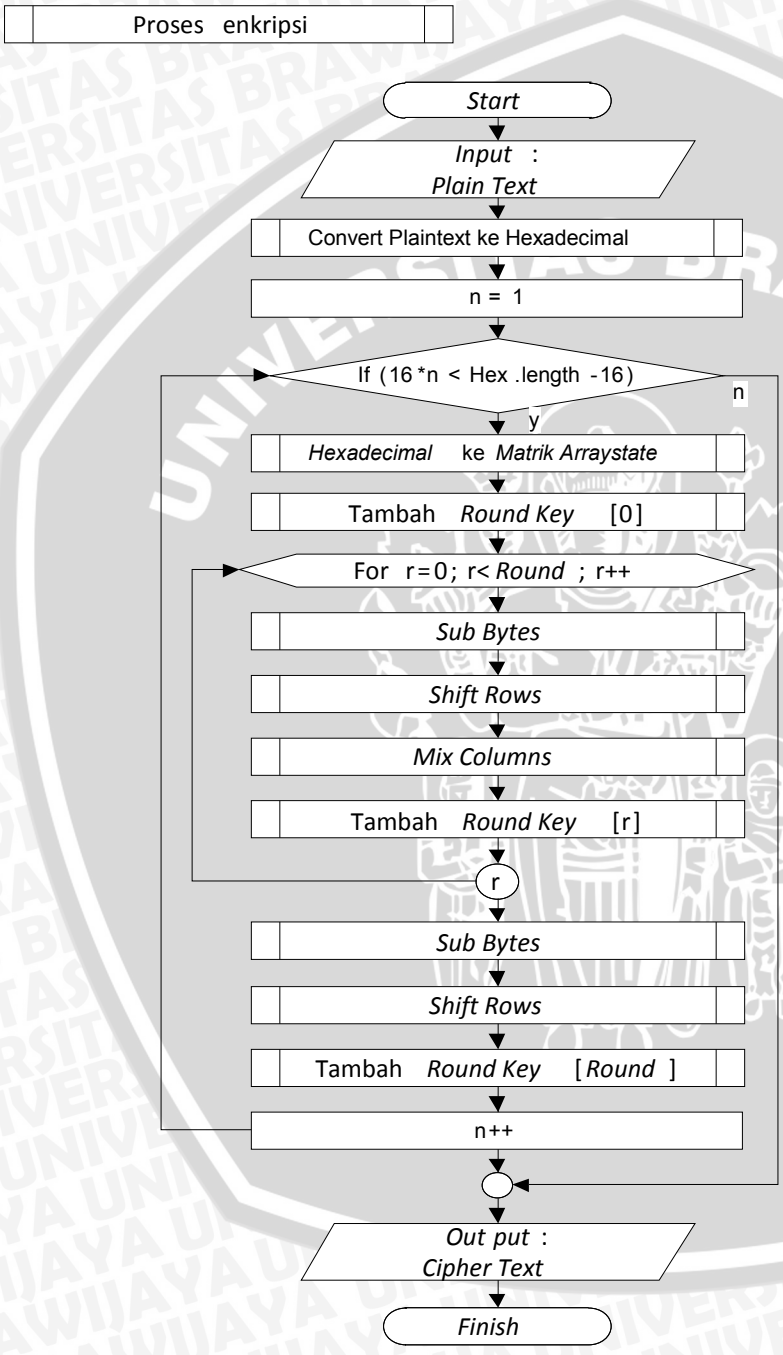
3.2.1 Enkripsi

Pada sub bab desain sistem ini akan dijelaskan mengenai tahapan atau proses-proses dalam membangun sistem enkripsi dengan menggunakan algoritma *rijndael*. Pada proses enkripsi dilakukan penyisipan *chiperkey* seperti pada gambar 3.2 sebagai kunci rahasia agar hasil enkripsi bisa dibaca kembali.



Gambar 3.2 Rijndael Encryptor
Sumber : perancangan

Enkripsi pada algoritma *rijndael* dilakukan dengan beberapa metode yakni, penambahan *round key, sub bytes, shift rows,* dan *mix columns* serta *key schedule* untuk membangkitkan key baru. Alur proses enkripsi ditunjukkan pada gambar 3.3.

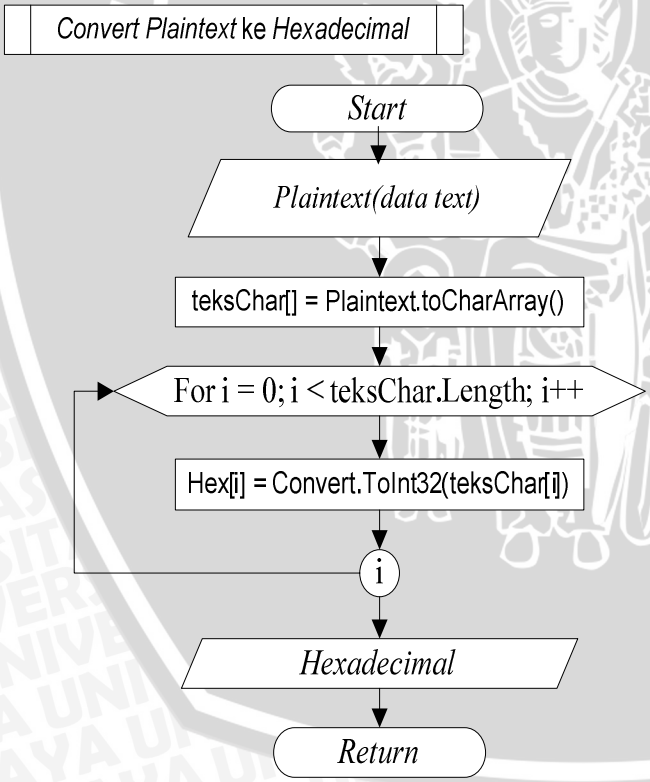


Gambar 3.3 Flowchart Enkripsi

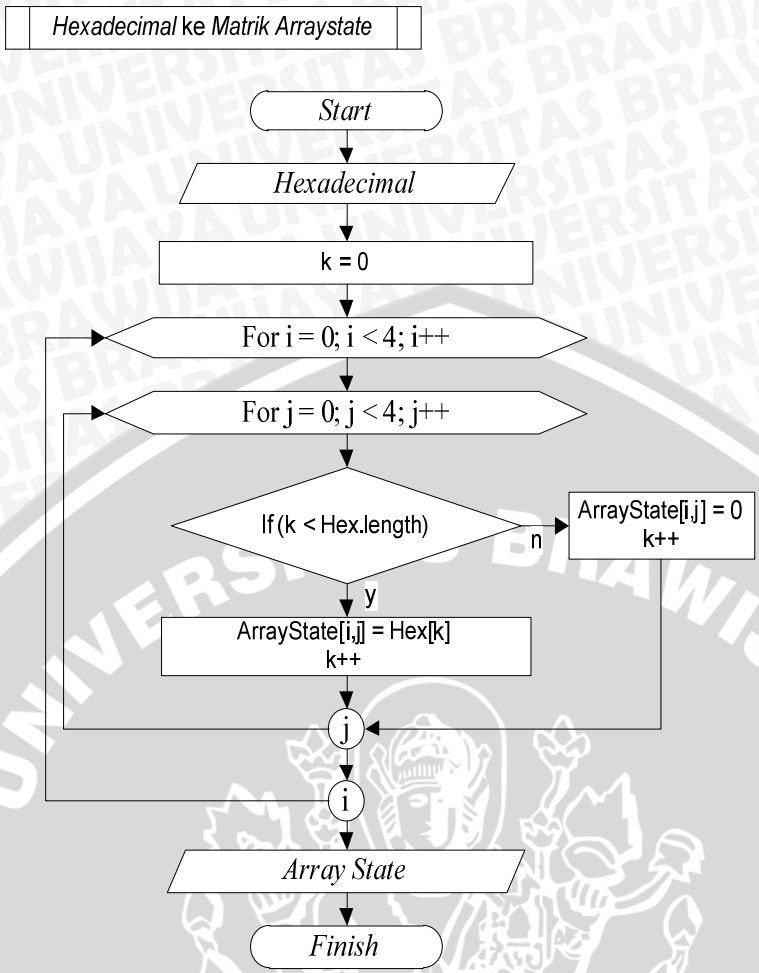
Sumber : Perancangan

Secara umum deskripsi alur proses enkripsi berdasarkan gambar 3.3 adalah:

1. Diambil data *plain text.*
2. Data *plain text* dikonvert ke bentuk *hexadecimal*. (flowchart du tunjukan pada gambar 3.4
3. Dilakukan pembentukan dari *hexadecimal* ke matrik *arraystate* (bentuk matrik 4x4)
4. Dilakukan proses penambahan *round key* menggunakan *chiper key*.
5. Dilakukan 9 kali perulangan untuk proses *sub bytes, shift rows, mix columns,* dan penambahan *round key*. Pada proses penambahan *round key* menggunakan *chiper key* baru yang sudah dibangkitkan dari proses *key schedule*.
6. Pada iterasi ke-10 dilakukan proses *sub bytes, shift rows*, dan penambahan *round key* ke-10.
7. Dihasilkan teks hasil enkripsi.



Gambar 3.4 convert*plaintext* ke *hexadecimal*
Sumber : Perancangan



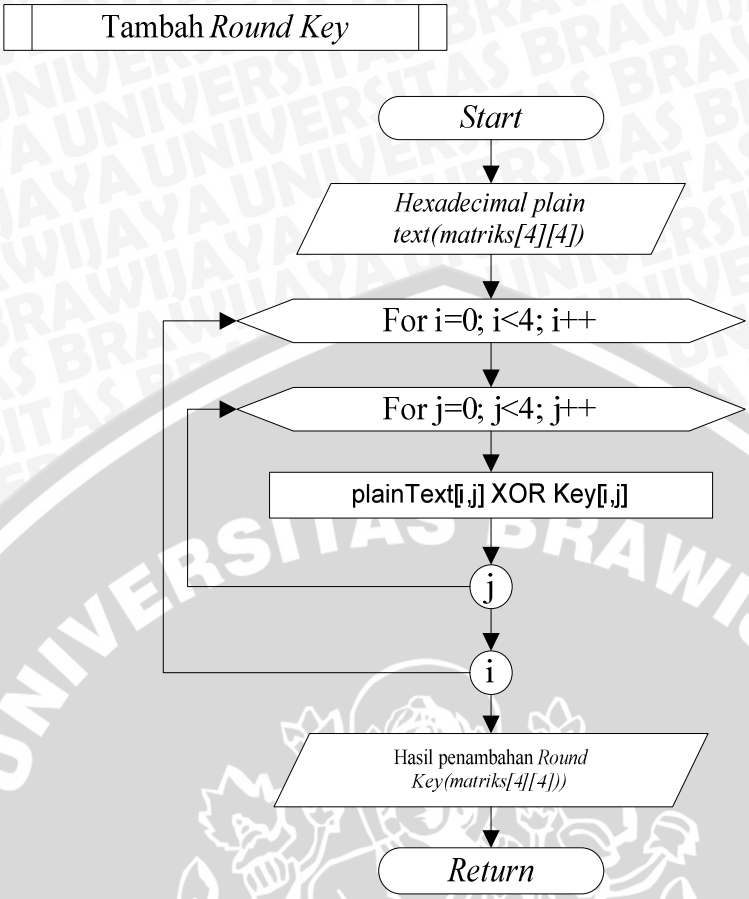
Gambar pembentukan hexadecimal ke arraystate (matrik 4x4)

Sumber : perancangan

3.2.1.1 Tambah *Round Key*

Pada proses penambahan *round key* dilakukan penyisipan *chiper key*. Perancangan proses penambahan *round key* terdapat pada *flowchart* gambar 3.5.

1. Diambil data teks yang sudah diubah ke bentuk *hexadecimal(gambar 3.4)*.
2. Dijadikan matrik 4x4.
3. Dilakukan perhitungan XOR dengan *chiper key*.
4. Dihasilkan *arraystate* hasil penambahan *chiper key*.

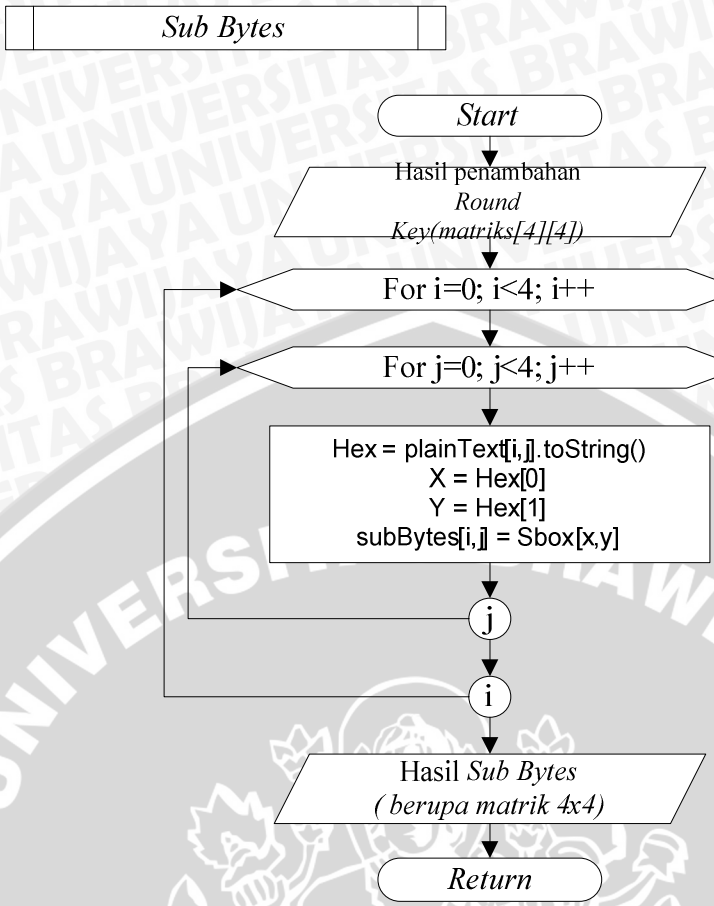


Gambar 3.5 Proses Penambahan *Round Key*
Sumber : Perancangan

3.2.1.2 *Sub Bytes*

Pada proses *sub bytes* dilakukan pemetakan setiap *byte* dari *arraystate* dengan menggunakan tabel substitusi S-*Box*. Perancangan proses *sub bytes* terdapat pada *flowchart* gambar 3.6.

1. Diambil hasil penambahan *round key*.
2. Dilakukan perulangan untuk mengganti matrik hasil *roundkey* ke matrik baru hasil *subbytes* dengan cara mengambil *hexadecimal*, untuk karakter pertama dijadikan parameter X dan karakter kedua dijadikan parameter Y. Parameter X dan Y dijadikan sebagai indeks untuk S-*Box*. Untuk tabel S-Box ditunjukkan pada lampiran I.
3. Dihasilkan *arraystate* hasil *sub bytes*.

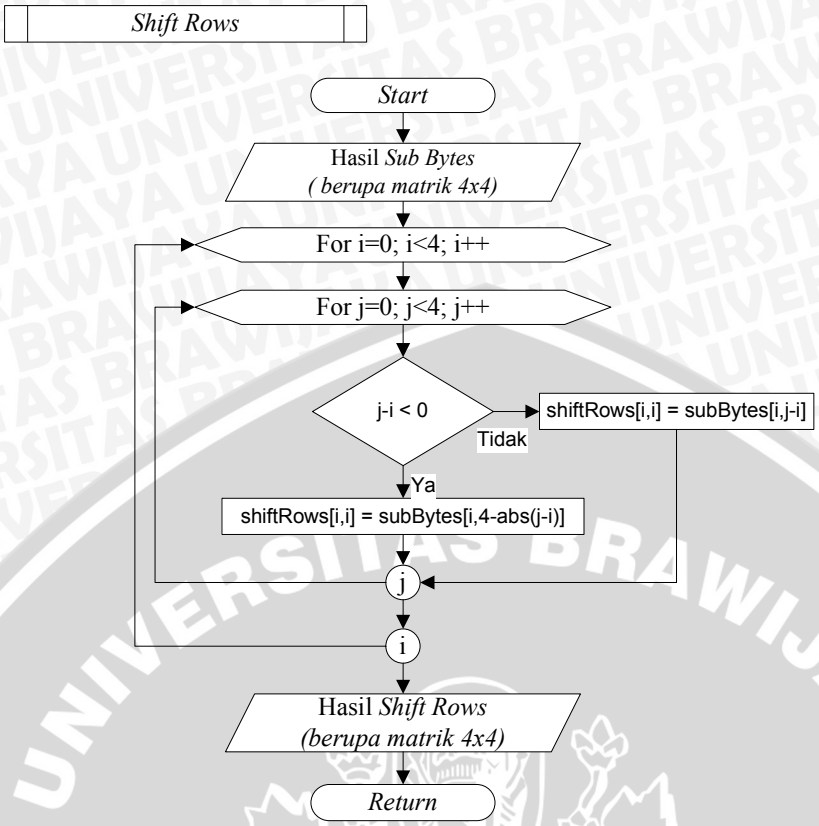


Gambar 3.6 *Sub Bytes*
Sumber : perancangan

3.2.1.3 *Shift Rows*

Pada proses *shiftrows* dilakukan pergeseran kekiri secara *wrapping* pada 3 baris terakhir *arraystate*. Perancangan proses *shiftrows* terdapat pada *flowchart* gambar 3.7.

1. Diambil *arraystate* dari hasil *subbytes*.
2. Dilakukan perulangan untuk menggeser *arraystate* mulai dari baris 1 sebanyak 0 pergeseran (tidak ada pergeseran), baris 2 sebanyak 1 pergeseran, baris 3 sebanyak 2 pergeseran, dan baris 4 sebanyak 3 kali pergeseran. Pergeseran dilakukan ke kiri menurut indeks *j*, sehingga sebesar *j-i*. Apabila *j-i* bernilai minus, maka dilakukan pergeseran sebesar 4-*abs(j-i)*.
3. Dihasilkan *array state* hasil *shiftrows*.

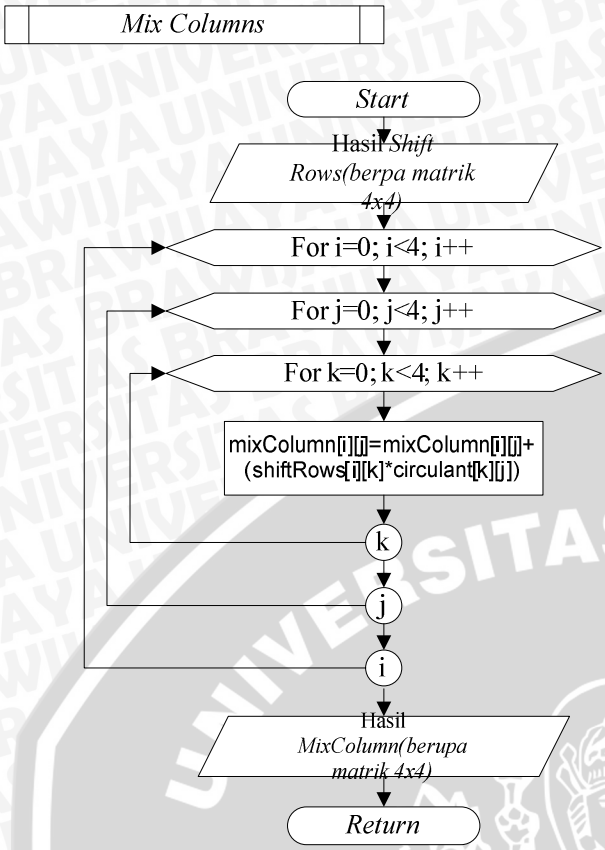


Gambar 3.7 Shift Rows
Sumber : Perancangan

3.2.1.4 *Mix Columns*

Pada proses *mixcolumns* dilakukan perkalian setiap kolom *arraystate* dengan polinom. Perancangan proses *mixcolumns* terdapat pada *flowchart* gambar 3.8.

1. Diambil *arraystate* hasil *shiftrows*.
2. Dilakukan perkalian matriks antara *arraystate* dengan *circulant* matrik dengan cara 3 kali perulangan.
3. Dihasilkan *arraystate* hasil *mixcolumns*.



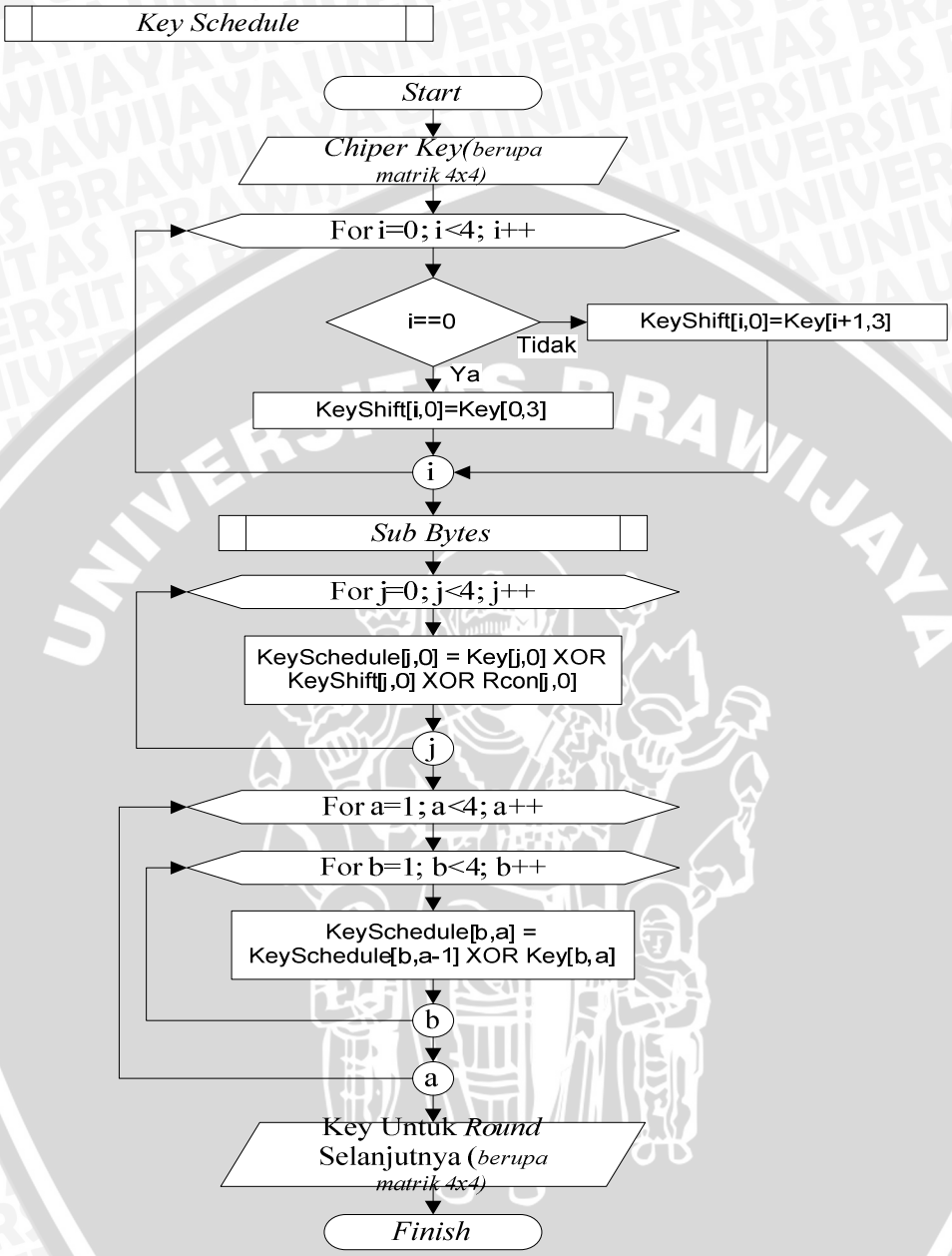
Gambar 3.8 Mix Columns
Sumber : perancangan

3.2.1.5 *Key Schedule*

Pada key *schedule* dilakukan proses untuk membangkitkan *key* baru dari *chiperkey*. Perancangan proses *keyschedule* terdapat pada *flowchart* gambar 3.9.

1. Diambil *chiperkey*.
2. Diambil kolom terakhir pada *arraystate* untuk digeser ke atas. Indeks *i* menunjukkan baris, sehingga pergeseran dihitung dengan *i*+1 dan pada *i*sama dengan 0 digeser pada indeks 3.
3. Dilakukan proses *subbytes* pada kolom hasil pergeseran.
4. Dilakukan XOR hasil *subbytes* tersebut dengan *Rcon* untuk kolom pertama pada key *schedule*.
5. Untuk kolom kedua diambil berdasarkan kolom pertama. Dilakukan XOR untuk kolom pertama key *schedule* dengan kolom kedua pada *chiperkey*.
6. Untuk kolom selanjutnya mengikuti *point* langkah nomor 5 sampai dengan kolom keempat.
7. Dihasilkan *key* baru.

8. *Key* baru tersebut nantinya akan dilakukan proses yang sama untuk mendapatkan *key* baru selanjutnya.

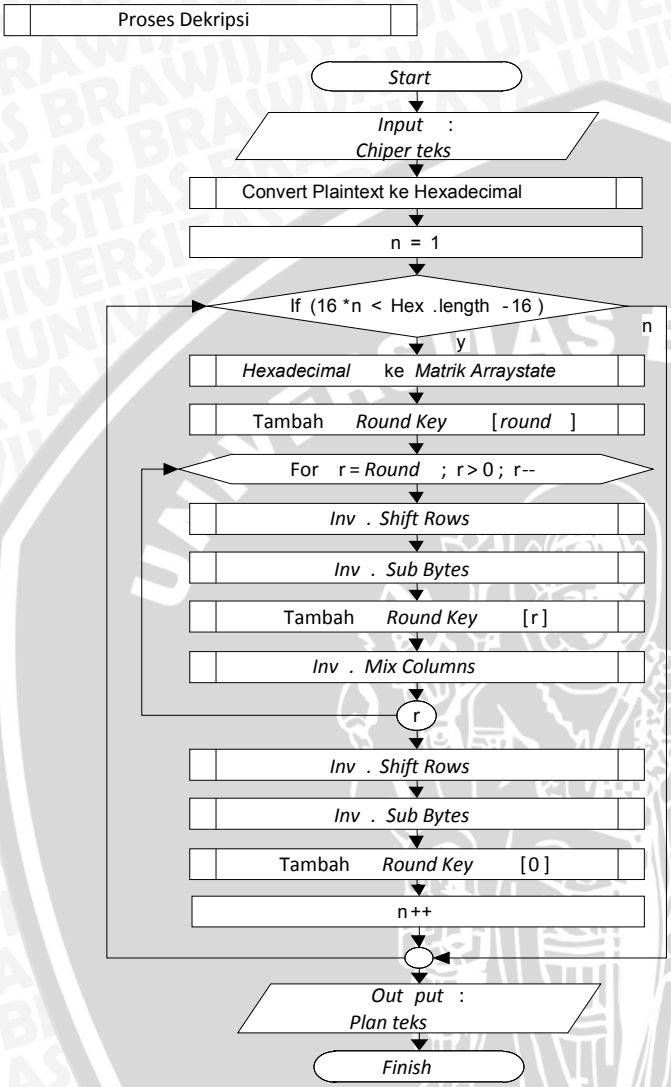


Gambar 3.9 Key Schedule
Sumber : Perancangan

3.2.2 Dekripsi

Pada sub bab desain sistem ini akan dijelaskan mengenai tahapan atau proses-proses dalam membangun sistem dekripsi dengan menggunakan algoritma *rijndael*. Pada proses dekripsi dilakukan penyisipan *chiperkey*sama seperti pada proses enkripsi.

Dekripsi pada algoritma *rijndael* dilakukan dengan beberapa metode yakni, penambahan *inverseround key, inverse sub bytes,inverse shift rows,* dan *inversemix columns*. Alur proses Dekripsi ditunjukkan pada gambar 3.10.



Gambar 3.10 Proses Dekripsi
Sumber : Perancangan

Secara umum deskripsi alur proses dekripsi berdasarkan gambar 3.10 adalah:

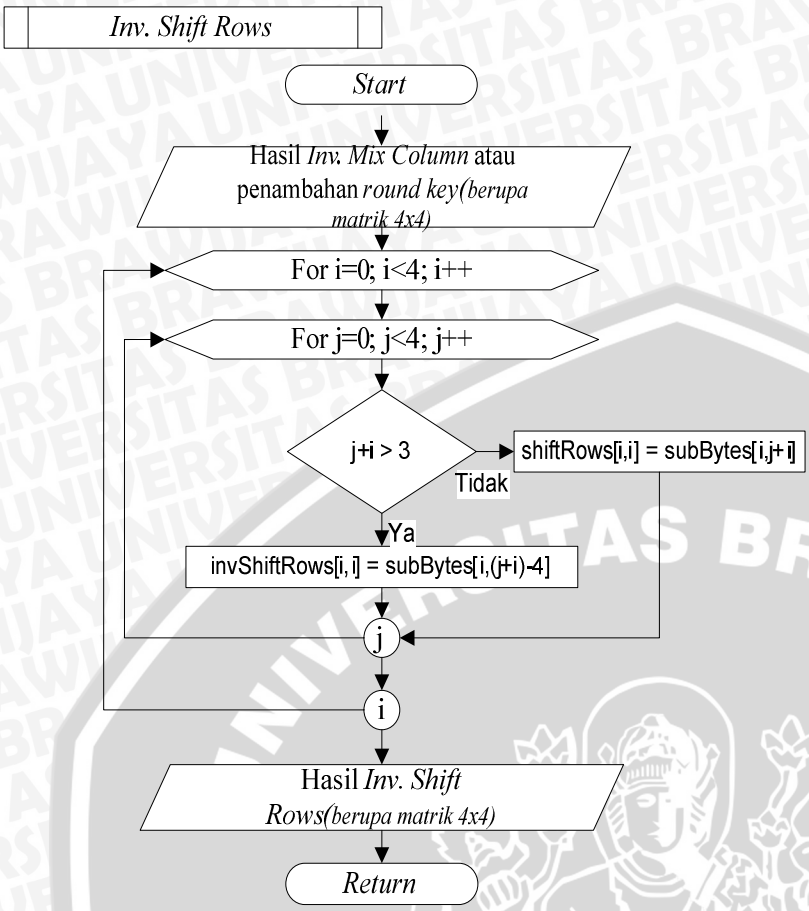
1. Diambil data *chipper text*
2. Data diubah ke bentuk *hexadecimal*.
3. Dilakukan proses penambahan *roundkey* dengan *chiperkey* atau *keyschedule* terakhir.

4. Dilakukan perulangan sebanyak 9 kali untuk proses *inverseshift rows, inverse sub bytes,* penambahan *roundkey*, dan *inversemixcolumns*.
5. Pada proses penambahan *roundkey* dilakukan penambahan *chiperkey* dengan *key* baru kebalikan dari proses enkripsi yakni dari *keyschedule* terakhir ke awal.
6. Pada iterasi atau ronde terakhir dilakukan proses *inverseshiftrows*, *inversesubbytes*, dan penambahan *roundkey*.
7. Dihasilkan hasil dekripsi.

3.2.2.1 *Inverse Shift Rows*

Pada proses *inverseshiftrows* dilakukan pergeseran kekanan secara *wrapping* pada 3 baris terakhir *arraystate*. Perancangan proses *inverseshiftrows* terdapat pada *flowchart* gambar 3.11.

1. Diambil *arraystate* hasil *inversemixcolumns* atau hasil penambahan *roundkey*.
2. Dilakukan perulangan untuk menggeser *arraystate* mulai dari baris 1 sebanyak 0 pergeseran, baris 2 sebanyak 1 pergeseran, baris 3 sebanyak 2 pergeseran, dan baris 4 sebanyak 3 kali pergeseran. Pergeseran dilakukan ke kanan menurut indeks *j*, sehingga sebesar *j+i*. Apabila *j+i* bernilai lebih dari 3, maka dilakukan pergeseran sebesar *(j+i)-4*.
3. Dihasilkan *arraystate* hasil *inverseshiftcolumns*.

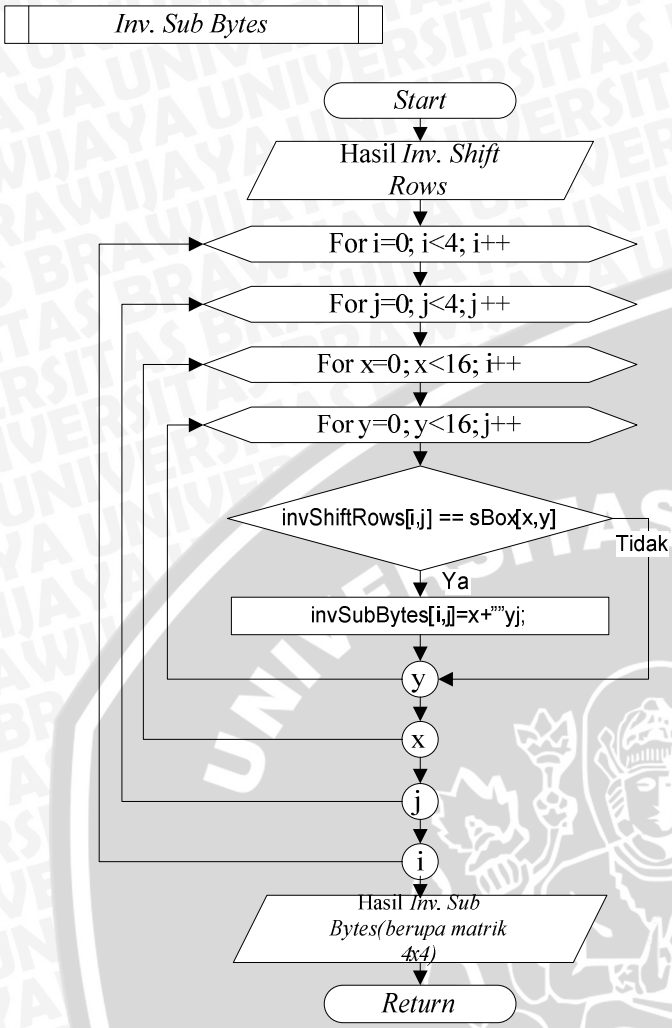


Gambar 3.11 Inverse Shift Column
Sumber : Perancangan

3.2.2.2 *Inverse Sub Bytes*

Pada proses *inversesub bytes* dilakukan pemetakan setiap *byte* dari *arraystate* dengan mengambil indeks dari tabel substitusi S-*Box* untukdisubtitusi. Perancangan proses *inversesub bytes* terdapat pada *flowchart* gambar 3.12.

1. Diambil hasil *inverse shift rows*.
2. Dilakukan perulangan untuk mengganti matrik hasil *inverse shift rows* ke matrik baru hasil *inversesubbytes* dengan cara mengambil *hexadecimal.* Jika *arraystateinverseshiftrows*merupakan nilai dari S-*Box*, maka diambil indeks X dan Y pada S-*Box* untuk nilai *inversesubbytes*.
3. Dihasilkan *arraystate* hasil *inversesub bytes*.

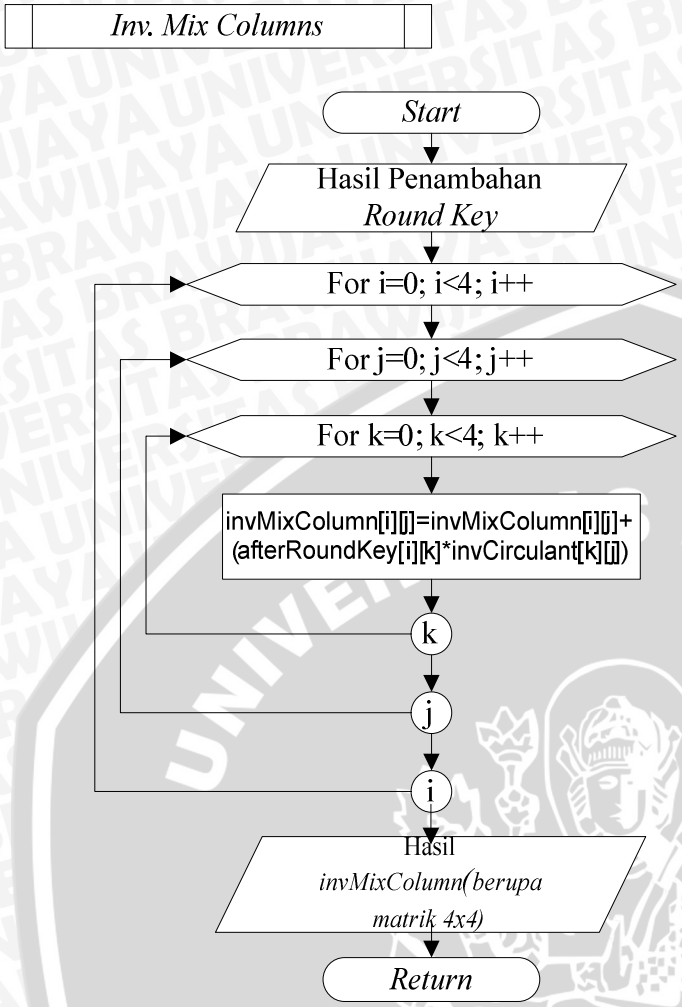


Gambar 3.12 Inverse Sub Bytes
Sumber : perancangan

3.2.2.3 *Inverse Mix Columns*

Pada proses *inversemixcolumns* dilakukan perkalian setiap kolom *arraystate* dengan *inverse* polinom. Perancangan proses *inversemixcolumns* terdapat pada *flowchart* gambar 3.13.

1. Diambil *arraystate* hasil penambahan *round key*.
2. Dilakukan perkalian matriks antara *arraystate* dengan *circulant* matriks dengan cara 3 kali perulangan.
3. Dihasilkan *arraystate* hasil *inversemixcolumns*.



Gambar 3.13 Inverse Mix Columns
Sumber : perancangan

3.3 Perhitungan Manual

3.3.1 Enkripsi

3.3.1.1 Data *Plain Text*

Data yang digunakan dalam perhitungan manual yakni *plaintext* "ilmu komputer 07". Dari data *plaintext* tersebut dikonvert menjadi bentuk *hexadecimal* yakni menjadi "69 6C 6D 75 20 6B 6F 6D 70 75 74 65 72 20 30 37". Pada penelitian ini digunakan AES 128 dengan ukuran blok 4, panjang kunci 4 blok dan terdapat 10 ronde. Hasil dari *hexadecimalplainttext* tersebut dipindahkan ke *arraystate* sebesar 4 blok seperti pada gambar 3.14.

69	6C	6D	75

20	6B	6F	6D
70	75	74	65
72	20	30	37

Gambar 3.14*Array State* Awal

3.3.1.2 Tambah *RoundKey*

Dalam proses penambahan *roundkey* dilakukan perhitungan XOR *arraystate* dengan *chiperkey*. *Chiperkey* yang digunakan pada penelitian ini ditunjukkan pada gambar 3.15.

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Gambar 3.15 Chiper Key
Sumber : perancangan

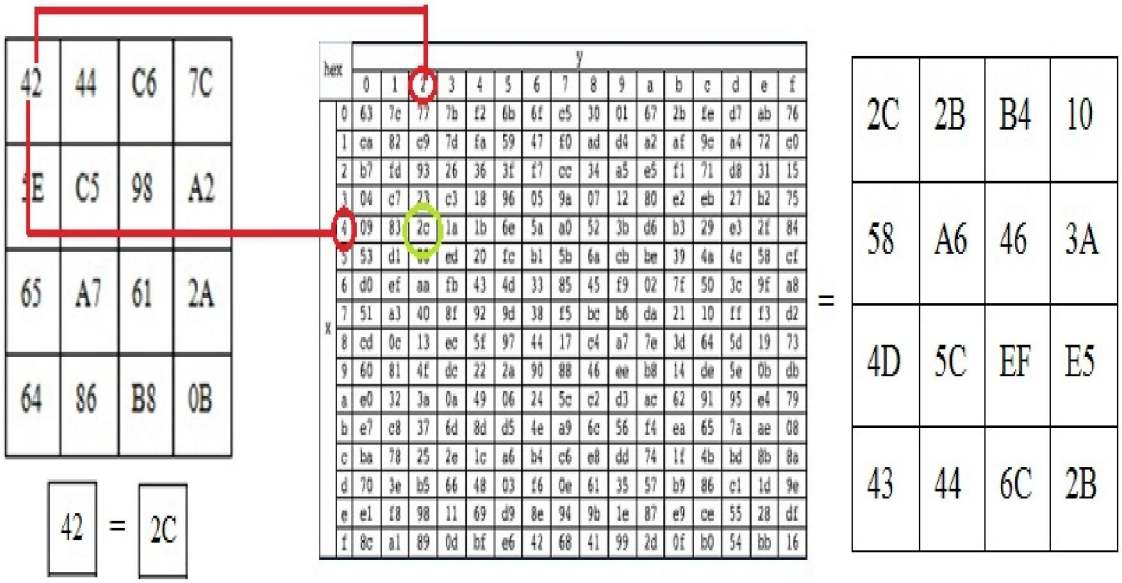
Dilakukan perhitungan XOR untuk *arraystate* dengan *chiperkey* terdapat pada gambar 3.16.

69	6C	6D	75	⊕	2B	28	AB	09	=	42	44	C6	7C
20	6B	6F	6D		7E	AE	F7	CF		5E	C5	98	A2
70	75	74	65		15	D2	15	4F		65	A7	61	2A
72	20	30	37		16	A6	88	3C		64	86	B8	0B

Gambar 3.16 PerhitunganTambah *Round Key*
Sumber : perancangan

3.3.1.3 *Round* 1 *SubBytes*

Pada langkah *subbytes* pada *round* ke-1 dilakukan substitusi hasil *arraystate* pada penambahan *roundkey* pada sub bab 3.3.1.2 dengan S-*Box*. Hasil substitusi dari *subbytes* terdapat pada gambar 3.16.

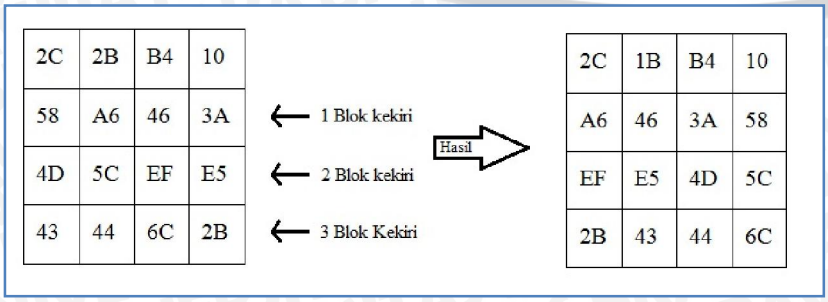


Gambar 3.17 Hasil *Round* 1 *Sub Bytes*
Sumber : jurnal S.box

Nilai *hexadecimal* pada *arraystate* dijadikan indeks pada S-*Box*. pada*hexadecimal* terdapat 2 digit, digit pertama sebagai indeks x dan digit kedua sebagai indeks y pada S-*Box*.

3.3.1.4 *Round* 1 *ShiftRows*

Pada langkah *shiftrows* dilakukan pergeseran blok*array* pada hasil *subbytes*. Pergeseran dilakukan pada blok*array* baris ke-2 sampai dengan 4. Pergeseran dilakukan sebanyak 1 blok ke kiri pada baris ke 2, 2 blok kekiri pada baris ke 3, dan 3 blok kekiri pada baris ke 4. Hasil dari *shiftrows* dapat dilihat pada gambar 3.18.



Gambar 3.18 Hasil *Round* 1 *Shift Rows*

3.3.1.5 *Round* 1 *MixColumns*

Pada langkah *mixcolumns* dilakukan perhitungan *multiplication* untuk *arraystate* hasil *shiftrows* dengan *circulant* matriks. Hasil *mixcolumns* dapat dilihat pada gambar 3.19.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 2C & 1B & B4 & 10 \\ A6 & 46 & 3A & 58 \\ EF & E5 & 4D & 5C \\ 2B & 43 & 44 & 6C \end{bmatrix} = \begin{bmatrix} 6D & 5A & 34 & F8 \\ 7A & E0 & 53 & 28 \\ 32 & 49 & D8 & 44 \\ 6B & 08 & 38 & EC \end{bmatrix}$$

Gambar 3.19 Hasil *MixColumns*

{02}.{2C}+{03}.{A6}+{01}.{EF}+{01}.{2B}={6D}

{02}.{1B}+{03}.{46}+{01}.{E5}+{01}.{43}={5A}

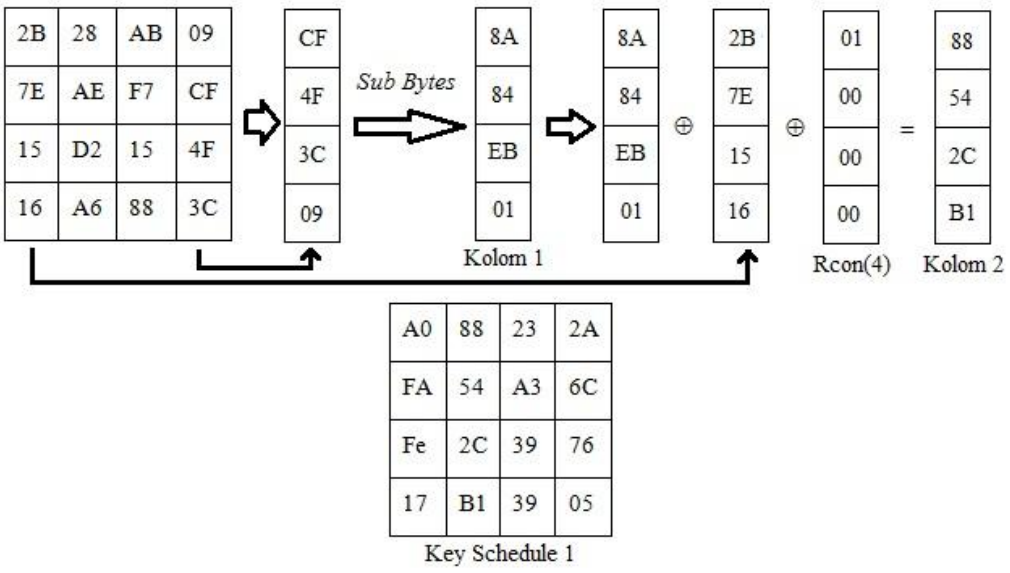
{02}.{B4}+{03}.{3A}+{01}.{4D}+{01}.{44}={34}

{02}.{10}+{03}.{58}+{01}.{5C}+{01}.{6C}={F8}

Perhitungan dilakukan pada semua sampai pada *hexadecimal* terakhir sehingga diperoleh hasil sesuai pada gambar 3.18.

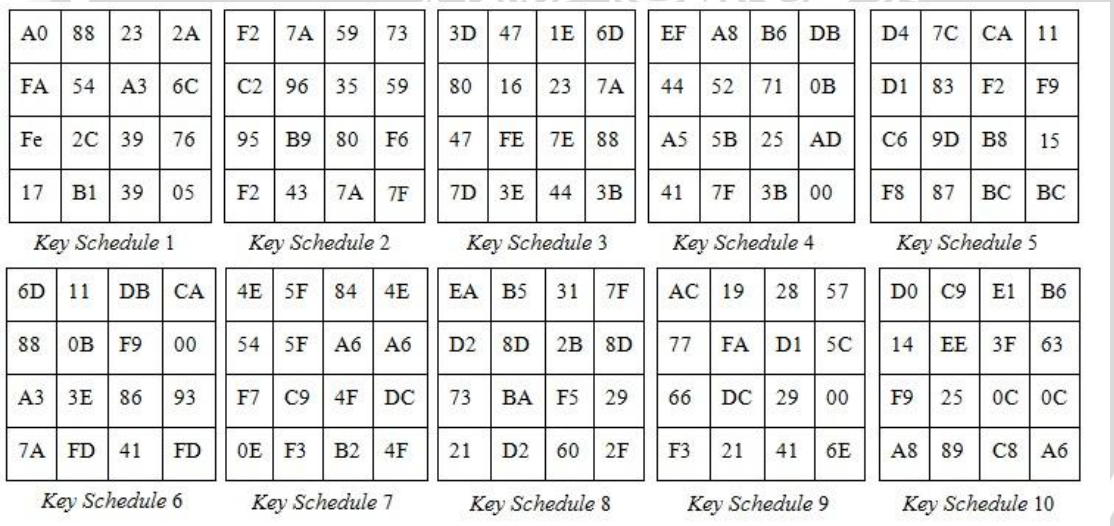
3.3.1.6 *KeySchedule*

Pada langkah ini dilakukan pembangkitan *key* baru dari *chiperkey* yang nantinya akan digunakan para proses penambahan *keyschedule* dari *round* 1 sampai dengan 10. Pembangkitan *key* baru dapat dilihat pada gambar 3.20.



Gambar 3.20 Hasil *KeySchedule* 1

Setelah terbentuk *keyschedule* ke-1, dihitung untuk *keyschedule* ke-2 sampai dengan ke-10 dengan proses yang sama dengan gambar 3.20. Hasil pembangkitan *key* ke-1 sampai dengan ke-10 dapat dilihat pada gambar 3.21.



Gambar 3.21*Key Schedule*

3.3.1.7 *Round* 1 Tambah *RoundKey*

Dalam proses penambahan *roundkey* pada ronde 1 dilakukan perhitungan XOR *arraystate* dengan *key schedule ke-1*. Hasil penambahan round key pada *round* 1 dapat dilihat pada gambar 3.22.

CD	D2	17	D2
80	B4	F0	44
CC	65	E1	32
7C	B9	01	E9

Gambar 3.22 Hasil*Round* 1 Tambah *Round Key*

3.3.1.8 *Round* 10

Pada perhitungan ronde ke-2 sampai dengan ke-9 dilakukan perhitungan yang sama untuk *sub bytes, shift rows, mix columns*, dan tambah *roundkey*. Perhitungan sampai dengan ronde ke-10 dapat dilihat pada gambar 3.23.

	SubBytes				*ShiftRows*				*MixColumns*				*AddRoundKey*			
Round 2	dD	b5	F0	B5	BD	B5	F0	B5	0B	CD	CB	0C	F9	B7	92	7F
	CD	8D	8C	1B	8D	8C	1B	CD	B1	C3	4D	9F	73	55	78	C6
	4B	4D	F8	23	F8	23	4B	4D	F9	4F	87	66	6C	F6	07	90
	10	56	7C	1E	1E	10	56	7C	95	4B	F7	BC	67	08	8D	C3
Round 3	99	A9	4F	D2	99	A9	4F	D2	DD	73	39	2A	E0	34	27	47
	8F	FC	BC	B4	FC	BC	B4	8F	00	EF	FC	4C	80	F9	DF	36
	50	42	C5	60	C5	60	50	42	86	41	0B	3E	C1	BF	75	B6
	85	30	5D	2E	2E	85	30	5D	D5	2D	55	1A	A8	13	11	21
Round 4	E1	18	CC	A0	E1	18	CC	A0	09	05	89	9D	E6	AD	3F	46
	CD	99	9E	05	99	9E	05	CD	89	2F	33	BB	CD	7D	42	B0

78	08	9D	FD
C2	7D	82	FD

9D	FD	78	08
FD	C2	7D	82

45	57	BE	E0
DD	67	C8	21

E0	1C	9B	4D
9C	18	F3	21

	Sub Bytes				Shift Rows				Mix Columns				Add Round Key			
Round 5	8E	95	75	5A	8E	95	75	5A	F4	78	94	F9	20	04	5E	E8
	BD	FF	2C	E7	FF	2C	E7	BD	AA	2D	35	89	7B	AE	C7	70
	E1	9C	14	E3	14	E3	E1	9C	45	1D	A7	D3	83	80	1F	C6
	DE	AD	0D	FD	FD	DE	AD	0D	83	CC	D8	D5	7B	4B	64	69
Round 6	B7	F2	58	9B	B7	F2	58	9B	7B	3B	1C	C0	16	2A	C7	0A
	21	E4	C6	51	E4	C6	51	21	C6	83	66	D6	4E	88	9F	D6
	EC	CD	C0	B4	C0	B4	EC	CD	D8	24	04	FE	7B	1A	82	6D
	21	B3	43	F9	F9	21	B3	43	0F	3D	28	DC	75	C0	69	21
Round 7	47	E5	C6	67	47	E5	C6	67	37	06	0D	E4	79	59	89	AA
	2F	C4	DB	F6	C4	DB	F6	2F	1C	91	E8	3D	48	CE	4E	9B
	21	A2	13	3C	13	3C	21	A2	B9	FA	A7	07	4E	33	E8	DB
	9D	BA	F9	FD	FD	9D	BA	F9	FF	F2	E9	CD	F1	01	5B	82

Round 8	B6	CB	A7	AC	B6	CB	A7	AC	79	E4	3A	4F	93	51	0B	30
	52	8B	2F	14	8B	2F	14	52	1E	E4	82	6F	CC	69	A9	E2
	2F	C3	9B	B9	9B	B9	2F	C3	25	75	69	28	56	CF	9C	01
	A1	7C	39	13	13	A1	7C	39	F7	89	31	0C	D6	5B	51	23
Round 9	DC	D1	2B	04	DC	D1	2B	04	4B	5D	6D	8E	E7	44	45	D9
	4B	F9	D3	98	F9	D3	98	4B	6A	1E	F1	C6	1D	E4	20	9A
	B1	8A	DE	7C	DE	7C	B1	8A	E8	FB	81	28	8E	27	A8	28
	F6	39	D1	26	26	F6	39	D1	14	30	26	74	E7	11	67	1A
Round 10	94	1B	6E	35	94	1B	6E	35					44	D2	8F	83
	A4	69	B7	B8	69	B7	B8	A4					7D	59	87	C7
	19	CC	C2	34	C2	34	19	CC					3B	11	15	C0
	94	82	85	A2	A2	94	82	85					0A	1D	4A	23

Gambar 3.23 Hasil *Round* 10

Pada gambar 3.22 dihasilkan *hexadecimal* hasil enkripsi dari *rijndael* adalah “44 D2 8F 83 7D 59 87 C7 3B 11 15 C0 0A 1D 4A 23”, kemudian di konvert ke *plaintext* menjadi text:

DÒ *f*}Y‡Ç;__À

.I#

3.3.2 Dekripsi

3.3.2.1 Data *Plain Text*

Ciphertext hasil enkripsi pada sub bab 3.3.1 akan didekripsi untuk dikembalikan ke bentuk semula. *Hexadecimal* yang terbentuk hasil dari enkripsi adalah "44 D2 8F 83 7D 59 87 C7 3B 11 15 C0 0A 1D 4A 23", selanjutnya dikonvert ke bentuk *arraystate* 4 blok.

3.3.2.2 Tambah *RoundKey*

Pada proses penambahan *roundkey* dilakukan perhitungan XOR untuk *arraystate* awal dengan hasil *keyschedule* ke-10 pada sub bab 3.3.1.6. Hasil dari penambahan *roundkey* terdapat pada gambar 3.24.

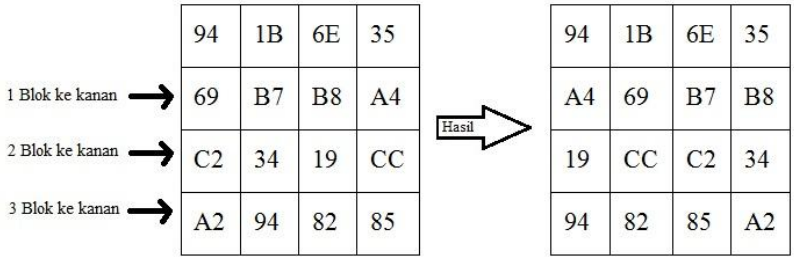
94	1B	6E	35
69	B7	B8	A4
C2	34	19	CC
A2	94	82	85

Gambar 3.24 Hasil Penambahan *RoundKey*

Setelah didapatkan hasil penambahan *roundkey*, selanjutnya akan dilakukan iterasi sebanyak 9 ronde. Pada setiap ronde dilakukan proses *inverseshiftrows*, *inversesubbytes*, tambah *roundkey*, dan *inversemixcolumns*.

3.3.2.3 *Round* 1 *InverseShiftRows*

Pada langkah *inverseshiftrows* dilakukan pergeseran blok*array* pada hasil penambahan *round key*. Pergeseran dilakukan pada blok*array* baris ke-2 sampai dengan 4. Pergeseran dilakukan sebanyak 1 blok ke kanan pada baris ke 2, 2 blok kekanan pada baris ke 3, dan 3 blok kekanan pada baris ke 4. Hasil dari *inverseshiftrows* dapat dilihat pada gambar 3.25.



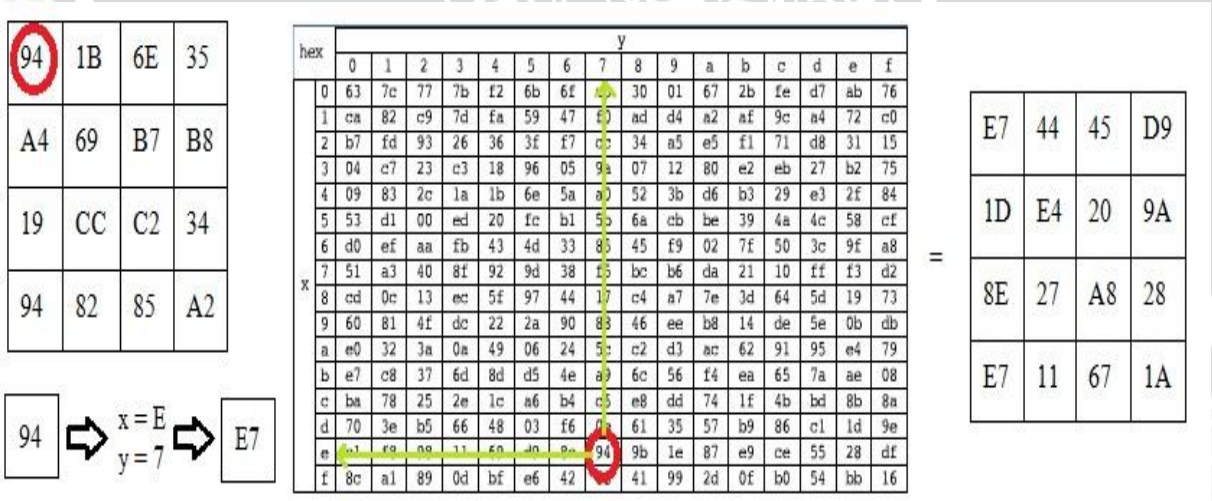
94	1B	6E	35
69	B7	B8	A4
C2	34	19	CC
A2	94	82	85

94	1B	6E	35
A4	69	B7	B8
19	CC	C2	34
94	82	85	A2

Gambar 3.25 Hasil *Round* 1 *InverseShiftRows*

3.3.2.4 *Round* 1 *InverseSubBytes*

Pada langkah *subbytes* pada *round* ke-1 dilakukan substitusi hasil *arraystate* pada hasil inverse shift rows pada sub bab 3.3.2.3 dengan S-*Box*. Hasil substitusi dari *subbytes* terdapat pada gambar 3.26.



E7	44	45	D9
1D	E4	20	9A
8E	27	A8	28
E7	11	67	1A

Gambar 3.26 Hasil *Round* 1 *InverseSub Bytes*

Nilai hexadecimal pada array state dijadikan sebagai nilai pada S-Box, kemudian dari nlai tersebut diambil sumbu X dan sumbu Y. Indeks yang merupakan sumbu X dan Y digabung, sehingga hasil substitusi bernilai XY.

3.3.2.5 *Round* 1 Tambah *RoundKey*

Dalam proses penambahan *roundkey* pada *round* 1 dilakukan perhitungan XOR untuk *arraystate* hasil *inversesubbytes* dengan *key schedule ke-9*. Hasil penambahan round key pada *round* 1 dapat dilihat pada gambar 3.27.

4B	5D	6D	8E
6A	1E	F1	C6
E8	FB	81	28
14	30	26	74

Gambar 3.27 Hasil*Round* 1 Tambah *Round Key*

3.3.2.6 *Round* 1 *InverseMixColumns*

Pada langkah *inversemixcolumns* dilakukan perhitungan *multiplication* untuk *arraystate* hasil penambahan *round key* dengan *circulant* matriks. Hasil *mixcolumns* dapat dilihat pada gambar 3.28.

$$\begin{bmatrix} 14 & 11 & 13 & 09 \\ 09 & 14 & 11 & 13 \\ 13 & 09 & 14 & 11 \\ 11 & 13 & 09 & 14 \end{bmatrix} \begin{bmatrix} 4B & 5D & 6D & 8E \\ 6A & 1E & F1 & C6 \\ E8 & FB & 81 & 28 \\ 14 & 30 & 26 & 74 \end{bmatrix} = \begin{bmatrix} DC & D1 & 2B & 04 \\ F9 & D3 & 98 & 4B \\ DE & 7C & B1 & 8A \\ 26 & F6 & 39 & D1 \end{bmatrix}$$

Gambar 3.28 Hasil *Round* 1 *InverseMixColumns*

{14}.{4B}+{11}.{6A}+{13}.{E8}+{09}.{14}={DC}

{14}.{5D}+{11}.{1E}+{13}.{FB}+{09}.{30}={D1}

{14}.{6D}+{11}.{F1}+{13}.{81}+{09}.{26}={2B}

{14}.{8E}+{11}.{C6}+{13}.{28}+{09}.{74}={04}

Perhitungan dilakukan pada semua sampai pada *hexadecimal* terakhir sehingga diperoleh hasil sesuai pada gambar 3.28

3.3.2.7 *Round* 10

Pada perhitungan *round* ke-2 sampai dengan ke-10 dilakukan perhitungan yang sama untuk *inverse shift rows, inversesub bytes,* tambah *roundkey*, dan *inverse mix columns*. Perhitungan sampai dengan *round* ke-10 dapat dilihat pada gambar 3.29.

	Inv. Shift Rows				*Inv. Sub Bytes*				*Add Round Key*				*Inv. Mix Columns*			
Round 2	DC	D1	2B	04	93	51	0B	30	79	E4	3A	4F	B6	CB	A7	AC
	4B	F9	D3	98	CC	69	A9	E2	1E	E4	82	6F	8B	2F	14	52
	B1	8A	DE	7C	56	CF	9C	01	25	75	69	28	9B	B9	2F	C3
	F6	39	D1	26	D6	5B	51	23	F7	89	31	0C	13	A1	7C	39
Round 3	B6	CB	A7	AC	79	59	89	AA	37	06	0D	E4	47	E5	C6	67
	52	8B	2F	14	48	CE	4E	9B	1C	91	E8	3D	C4	DB	F6	2F
	2F	C3	9B	B9	4E	33	E8	DB	B9	FA	A7	07	13	3C	21	A2
	A1	7C	39	13	F1	01	5B	82	FF	F2	E9	CD	FD	9D	BA	F9
Round 4	47	E5	C6	67	16	2A	C7	0A	7B	3B	1C	C0	B7	F2	58	9B
	2F	C4	DB	F6	4E	88	9F	D6	C6	83	66	D6	E4	C6	51	21
	21	A2	13	3C	7B	1A	82	6D	D8	24	04	FE	C0	B4	EC	CD
	9D	BA	F9	FD	75	C0	69	21	0F	3D	28	D	F9	21	B3	43

			C

Round 5

B7	F2	58	9B
21	E4	C6	51
EC	CD	C0	B4
21	B3	43	F9

20	04	5E	E8
7B	AE	C7	70
83	80	1F	C6
7B	4B	64	69

F4	78	94	F9
AA	2D	35	89
45	1D	A7	D3
83	CC	D8	D5

8E	95	75	5A
FF	2C	E7	BD
14	E3	E1	9C
FD	DE	AD	0D

Round 6

8E	95	75	5A
BD	FF	2C	E7
E1	9C	14	E3
DE	AD	0D	FD

E6	AD	3F	46
CD	7D	42	B0
E0	1C	9B	4D
9C	18	F3	21

09	05	89	9D
89	2F	33	BB
45	57	BE	E0
DD	67	C8	21

E1	18	CC	A0
99	9E	05	CD
9D	FD	78	08
FD	C2	7D	82

Inv. Shift Rows *Inv. Sub Bytes* *Add Round Key* *Inv. Mix Columns*

Round 7

E1	18	CC	A0
CD	99	9E	05
78	08	9D	FD
C2	7D	82	FD

E0	34	27	47
80	F9	DF	36
C1	BF	75	B6
A8	13	11	21

DD	73	39	2A
00	EF	FC	4C
86	41	0B	3E
D5	2D	55	1A

99	A9	4F	D2
FC	BC	B4	8F
C5	60	50	42
2E	85	30	5D

Round 8	99	A9	4F	D2	F9	B7	92	7F	0B	CD	CB	0C	BD	B5	F0	B5
	8F	FC	BC	B4	73	55	78	C6	B1	C3	4D	9F	8D	8C	1B	CD
	50	42	C5	60	6C	F6	07	90	F9	4F	87	66	F8	23	4B	4D
	85	30	5D	2E	67	08	8D	C3	95	4B	F7	BC	1E	10	56	7C
Round 9	BD	B5	F0	B5	CD	D2	17	D2	6D	5A	34	F8	2C	1B	B4	10
	CD	8D	8C	1B	80	B4	F0	44	7A	E0	53	28	A6	46	3A	58
	4B	4D	F8	23	CC	65	E1	32	32	49	D8	44	EF	E5	4D	5C
	10	56	7C	1E	7C	B9	01	E9	6B	08	38	EC	2B	43	44	6C
Round 10	2C	1B	B4	10	42	44	C6	7C	69	6C	6D	75				
	58	A6	46	3A	5E	C5	98	A2	20	6B	6F	6D				
	4D	5C	EF	E5	65	A7	61	2A	70	75	74	65				
	43	44	6C	2B	64	86	B8	0B	72	20	30	37				

Gambar 3.29 Hasil *round* ke-10

Pada gambar 3.29 dihasilkan *hexadecimal* hasil dekripsi dari *rijndael* adalah “69 6C 6D 75 20 6B 6F 6D 70 75 74 65 72 20 30 37”, kemudian di konvert ke bentuk plain text menjadi:

ilmu komputer 07

Metode pengujian

Setelah dilakukan keseluruhan tahapan sistem, maka selanjutnya dilakukan pengujian ketahanan enkripsi dengan menggunakan exhaustive attack, yakni menghitung peluang keadaan terburuk peluang memecahkan key dan keadaan terbaik pemecahan key. Tabel testing echaustive attack dapat dilihat pada tabel 3.1

Tabel 3.1 exhaustive attack

Panjang	Karakter yang mungkin	Total

Plaintext		
1		
…		
16		

Pengukuran avalanche effect

dilakukan dengan menghitung jumlah yang berbeda pada dua cipher teks yang telah dienkripsi, tabel perhitungan avallance effect terdapat pada tabel 3.2

Tabel 3.2 Pengujian *Avalanche Effect*

Key	**Bit Berubah**	**Persentase (%)**
….		
Rata – rata		