J-SISKO TECH
Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD
P:ISSN : 2621-8976 E-ISSN : 2615-5133 ■ 36
Vol.1, No.2, Juli 2018, pp.36-50

Pengamanan Citra Digital Menggunakan Kombinasi Antara Algoritma Aes Dan Metode Lsb

Muhammad Zunaidi, Suharsil
STMIK Triguna Dharma
info@trigunadharma.ac.id

Abstrak

Pesatnya perkembangan teknologi sekarang ini membuat proses komunikasi menjadi lebih mudah dan memiliki jangkauan yang sangat luas untuk mengirimkan suatu pesan atau informasi. Penyampaian pesan melalui media internet merupakan sarana komunikasi yang mudah dan efesien. Akan tetapi, banyak orang yang kini telah meragukan keamanan pada sarana komunikasi yang ada. Hal ini tidak terlepas dari terjadinya berbagai tindakan penyadapan dan pemantauan oleh pihak-pihak yang tidak berkepentingan atau tidak bertanggung jawab sehingga kerahasiaannya kurang terjaga dalam mengamankan suatu pesan.Banyak cara untuk mengamankan data salah satunya dengan menggunakan teknik Kriptografi dan Steganografi. Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan pesan. Sedangkan Steganografi adalah seni untuk menyembunyikan pesan di dalam media citra digital sedemikian rupa sehingga orang lain tidak menyadari ada sesuatu pesan di dalam media tersebut. Adapun salah satu metode yang dapat digunakan pada teknik Kriptografi adalah algoritma AES sedangkan pada teknik Steganografi dengan menggunakan metode LSB.Dengan adanya aplikasi sistem pengamanan pesan rahasia menggunakan teknik kriptografi dengan algoritma AES dan steganografi dengan metode LSB, diharapkan dapat mengamankan suatu pesan sehingga tidak dapat dimengerti oleh pihak-pihak yang tidak berkepentingan.

Kata kunci : Kriptografi, Steganografi, Algoritma AES,Metode LSB, Citra Digital, pesan

1. Pendahuluan

Pesatnya perkembangan teknologi sekarang ini membuat proses komunikasi menjadi lebih mudah dan memiliki jangkauan yang sangat luas untuk mengirimkan suatu pesan atau informasi. Penyampaian pesan melalui media *internet* merupakan sarana komunikasi yang mudah dan efesien. Akan tetapi, banyak orang yang kini telah meragukan keamanan pada sarana komunikasi yang ada. Hal ini tidak terlepas dari terjadinya berbagai tindakan penyadapan dan pemantauan oleh pihak-pihak yang tidak berkepentingan atau tidak bertanggung jawab sehingga kerahasiaannya kurang terjaga dalam mengamankan suatu pesan.

Banyak cara untuk mengamankan data salah satunya dengan menggunakan metode Kriptografi dan Steganografi. Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan pesan. Sedangkan Steganografi adalah seni untuk menyembunyikan pesan di dalam media digital sedemikian rupa sehingga orang lain tidak menyadari ada sesuatu pesan di dalam media tersebut.

Dalam Kriptografi dan Steganografi maka data yang dianggap rahasia akan disamarkan dengan sedemikian rupa sehingga jika data itu bisa didapatkan maka tidak akan bisa dimengerti oleh pihak yang tidak berhak. Adapun salah satu metode yang dapat digunakan pada teknik Kriptografi adalah algoritma *Advanced Encryption Standard* sedangkan pada teknik Steganografi dengan menggunakan metode *Least Significant Bit*.

Received February 12, 2018; Revised May 10, 2018; Accepted June 20, 2018

AES merupakan sistem penyandian blok yang bersifat non-Feistel karena AES menggunakan komponen yang selalu memiliki invers dengan panjang blok 128 bit. Kunci AES dapat memiliki panjang kunci bit 128, 192 dan 256 bit. Penyandian AES menggunakan proses yang berulang yang disebut dengan ronde. Jumlah ronde yang digunakan oleh AES tergantung dengan panjang kunci yang digunakan. Setiap ronde membutuhkan kunci dan masukan dari ronde berikutnya. Kunci ronde dibangkitkan berdasarkan kunci yang diberikan. Sedangkan LSB merupakan teknik substitusi pada steganografi, biasanya arsip 24 bit atau 8 bit digunakan untuk menyimpan citra digital. Representasi warna dari *pixel-pixel* bisa diperoleh dari warna-warna primer, yaitu merah, hijau, dan biru. Citra 24 bit menggunakan 3 *byte* untuk masing-masing *pixel*, dimana setiap warna primer direpresentasikan dengan ukuran 1 *byte*. Penggunaan citra 24 bit memungkinkan setiap pixel dipresentasikan dengan nilai warna sebanyak 16.777.216 macam. Dua bit dari saluran warna tersebut bisa digunakan untuk menyembunyikan data, yang akan mengubah jenis warna *pixel* nya menjadi 64 warna.

Dengan kombinasi Kriptografi dan Steganografi secara bersamaan dimaksudkan untuk memberikan keamanan berlapis dalam pengamanan suatu pesan. Berdasarkan uraian diatas maka pembahasan ini mengangkat judul "Pengamanan Citra Digital Menggunakan Kombinasi Antara Algoritma AES Dan Metode LSB".

2. Metode Penelitian

Dalam melakukan penulisan skripsi ini ada beberapa hal untuk mendapatkan atau mengumpulkan data yang diperlukan,antara lain adalah:

1. Pengumpulan Data yaitu :
   a. *Surfing (Field Research)*
      Yaitu pencarian atau pengumpulan data yang diperlukan melalui media internet serta sumber lainnya.
   b. Studi Pustaka *(Library Research)*
      Yaitu dengan membaca literatur-literatur dan referensi yang berhubungan dengan permasalahan yang dibahas.
2. Analisa Data.
   Yaitu melakukan analisa terhadap data yang diperoleh untuk digunakan dalam menyelesaikan masalah yang diangkat.
3. Perancangan
   Yaitu melakukan pengkodean terhadap rancangan-rancangan yang telah didefenisikan.
4. Implementasi
   Yaitu mengevaluasi kemampuan program dengan menggunakan aplikasi yang telah di instalasi sebagai pengujian, untuk menemukan hasil enkripsi, dekripsi, penyisipan pesan dan pengekstrakan.

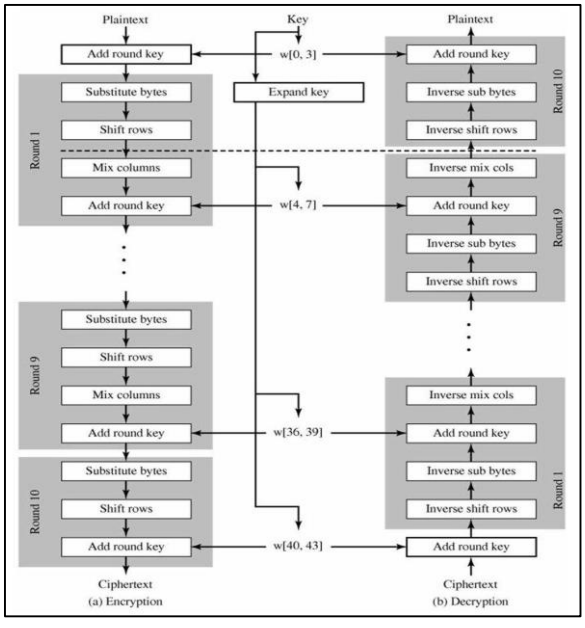
3. ANALISA DAN HASIL

A. Algoritma Enkripsi Aes-128

Sebelum pesan rahasia disisipkan kedalam media citra digital, pesan akan dilakukan proses enkripsi menggunakan algoritma AES. Proses enkripsi ini membutuhkan kunci yang sama panjangnya dengan teks pesan, apabila panjang kunci yang dimasukkan kurang dari panjang teks pesan maka kunci akan dulakukan proses perulangan sepanjang jumlah teks pesan. Proses enkripsi ini akan menghasilkan *ciphertext* yang siap untuk disisipkan kedalam media citra digital.Sesuai dengan batasan masalah, file citra yang digunakan adalah berformat bitmap 24 bit. Pesan rahasia yang dapat ditampung di dalam

media citra digital tergantung kapasistas citra yang dimiliki, misalkan pada berkas citra digital 24 bit berukuran 400 x 267 *pixel* terdapat 40050 *pixel*, setiap pixel (titik) pada citra tersebut terdiri dari susunan tiga warna dasar yaitu, merah, hijau dan biru (RGB), dimana setiap warna dasar dipresentasikan dengan ukuran 8 bit (1 *byte*). Sehingga jumlah seluruh kapasitas media penampung adalah 40050 x 3 = 120150 *byte*. Karena setiap *byte* hanya bisa menyisipkan 1 bit LSB-nya, maka ukuran data yang disisipkan di dalam citra tersebut maksimum 40050 : 8 = 5006,25 *byte* (1 *byte* = 1 karakter). Ukuran data ini harus dikurangi dengan panjang nama *file*-nya karena penyembunyian pesan rahasia tidak hanya menyembunyikan isi pesan tersebut, tetapi juga nama *file*-nya.

Misalkan pesan rahasia yang akan disisipkan adalah “MuhammadAjiPraye” dengan menggunakan kunci “A5r1S4m5UARrT1I1”. Pesan tersebut akan disisipkan kedalam media citra digital ”mobil.bmp” dengan ukuran 400 x 267 *pixel*.



Dalam hal ini sebelum proses penyisipan pesan dilakukan, terlebih dahulu pesan dienkripsikan dengan algoritma AES. Berikut skema algoritma AES ditunjukkan pada gambar diatas.Dalam algoritma AES untuk mengenkripsi sebuah teks, teks terlebih dahulu diorganisir sebagai *state*. *State* adalah

blok-blok *array* dengan berukuran tetap yaitu 128 bit dimana tiap blok menyimpan bilangan biner 8 bit yang dikonversi dalam bentuk heksadesimal. Berikut ini akan dijelaskan tahapan-tahapan dalam proses enkripsi algoritma AES dengan sebuah *plaintext* dan kunci :

***Plaintext* (P) = MuhammadAjiPraye**
Kunci (K) = A5r1S4m5UARrT1I1

Langkah pertama yang dilakukan adalah mencari bilangan heksadesimal dari P dan K dengan mengkonversi P dan K dalam bentuk bilangan biner. Berikut adalah bilangan biner 8 bit dari P dan K :

P = 01001101 01110101 01101000 01100001 01101101 01101101 01100001 01100100 01000001 01000001 01101010 01101001 01110010 01100001 01111001 01100101
K = 01000001 00110101 01110010 00110001 01010011 00110100 01101101 00110101 01010101 01000001 01010010 01110010 01010100 00110001 01001001 00110001

Pada AES *state* dibentuk menjadi blok *array* 4x4 (matriks 4x4) yang berarti terdapat 16 blok *array*. Berdasarkan bilangan biner yang didapat dari P dan K maka pembentukan *state* dapat dilakukan dengan membagi bilangan biner menjadi 16 bagian. Pembagian tersebut dikarenakan untuk memudahkan langkah selanjutnya yaitu mencari bilangan heksadesimal dari masing-masing bilangan biner hasil pembagian, bilangan heksadesimal untuk P dan K adalah bilangan heksadesimal dua dimensi. Setelah bilangan heksadesimal didapat maka selanjutnya penyusunan *state* blok *array* dilakukan secara vertikal. Untuk bilangan heksadesimal dari P dan K dapat dilihat pada gambar berikut:

4D	75	68	61
6D	6D	61	64
41	6A	69	50
72	61	79	65

P

⊕

41	35	72	31
53	34	6D	35
55	41	52	72
54	31	49	31

K

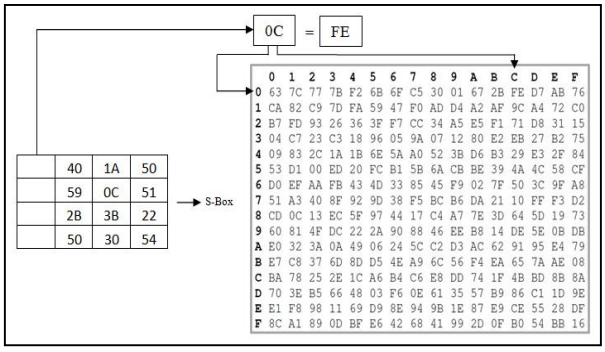
Proses enkripsi algoritma AES menggunakan 4 jenis transformasi, yaitu *SubBytes, ShiftRows, MixColumns* dan *AddRoundKey*. Berdasarkan *plaintext* pada *state* akan dijelaskan untuk setiap transformasi. Berikut transformasi lagoritma enkripsi AES.

1. Transformasi *SubBytes*

Pada proses transformasi *SubBytes*, *plaintext* yang telah disusun berbentuk matriks 4x4 pada *state* melakukan pencarian bilangan heksadesimal pada *S-Box SubBytes*. Namun sebelum melakukan transformasi *SubBytes*, *plaintext* pada *state* terlebih dahulu melakukan operasi XOR dengan kunci sebagai proses pencampuran *plaintext* dan kunci *(initial Round)*. Berikut adalah gambar untuk proses operasi XOR dengan plainteks pada *state* dan kunci sebagai nilai masukan.

State					Key					Output			
4D	75	68	61	⊕	41	35	72	31	=	0C	40	1A	50
6D	6D	61	64		53	34	6D	35		3E	59	0C	51
41	6A	69	50		55	41	52	72		14	2B	3B	22
72	61	79	65		54	31	49	31		16	50	30	54

Pada gambar diatas, *output* adalah hasil dari operasi XOR yang kemudian dianggap sebagai *state* yang baru setelah pencampuran *plaintext* dengan kunci. Proses tersebut dikatakan *initial round.*
Setelah proses XOR *plainteks* dan key, *state* keluaran kemudian di transformasikan dengan *S-Box SubBytes*. Pada transformasi ini nilai pertama bilangan heksadesimal *state* mengarah pada kolom *S-Box SubBytes*. Bilangan heksadesimal kedua *state* mengarah pada kolom *S-Box* SubBytes. Bilangan heksadesimal yang didapat antara baris dan kolom adalah hasil dari transformasi. Setelah proses transformasi, keluaran proses transformasi selanjutnya, yaitu transformasi *ShiftRows*. Untuk transformasi *SubBytes* pada *state* terhadap *S-Box* dapat dilihat pada gambar berikut:

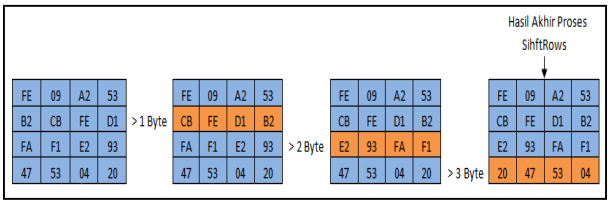


Dari gambar di atas proses transformasi *SubBytes* maka hasil keseluruhannya ditunjukkan pada gambar berikut:

FE	09	A2	53
B2	CB	FE	D1
FA	F1	E2	93
47	53	04	20

2. Transformasi ShiftRows

Setelah melakukan transformasi *SubBytes*, selanjutnya adalah transformasi permutasi atau dikenal dengan transformasi *ShiftRows*. *ShiftRows* dilakukan dengan menjalankan operasi *circular left*. Transformasi ini mengubah posisi blok pada *state* tanpa merubah nilai blok. Perubahan posisi dilakukan sebanyak satu kali pada baris kedua, sebanyak dua kali pada baris ketiga dan sebanyak tiga kali pada baris keempat. Berdasarkan hasil transformasi *SubBytes*, maka gambar berikut adalah dari transformasi *ShiftRows*.



3. Transformasi MixColumns

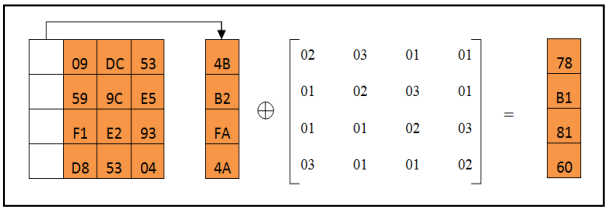
Transformasi *MixColumns* adalah proses ketiga dalam satu ronde enkripsi. Transformasi *MixColumns* bertujuan untuk mencampur blok pada kolom-kolom *state* terhadap blok keluaran. Agar dapat melakukan perhitungan matriks dengan melakukan perkalian dan penjumlahan. Disini, blok-blok pada kolom *state* akan diperlukan sebagai suatu *plynomial* yang berada dalam GF(2^8) dan akan dikalikan dengan modulo x4 + 1. Berikut adalah rumus perhitungan transformasi *MixColumns*.

$$S'_{0,0} = (S_{0,0} * 02) \oplus (S_{1,0} * 03) \oplus (S_{2,0} * 01) \oplus (S_{3,0} * 01)$$

$$S'_{0,1} = (S_{0,1} * 01) \oplus (S_{1,1} * 02) \oplus (S_{2,1} * 03) \oplus (S_{3,1} * 01)$$

$$S'_{0,2} = (S_{0,2} * 01) \oplus (S_{1,2} * 01) \oplus (S_{2,2} * 02) \oplus (S_{3,2} * 03)$$

$$S'_{0,3} = (S_{0,3} * 03) \oplus (S_{1,3} * 01) \oplus (S_{2,3} * 01) \oplus (S_{3,3} * 02)$$

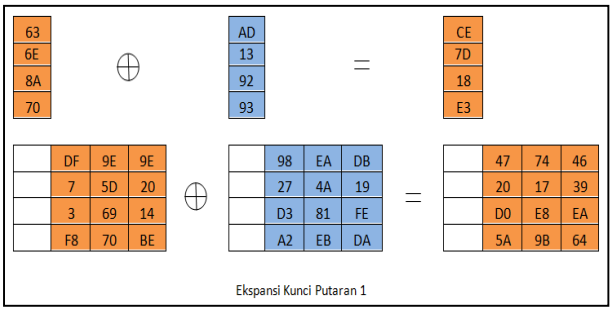


Dari gambar diatas skema rumus transformasi *MixColumns* maka hasil keseluruhannya ditunjukkan pada gambar berikut:

63	DF	9E	9E
6E	07	5D	20
8A	03	69	14
70	F8	70	BE

4. Transformasi AddRoundKey

Transformasi *AddRoundKey* pada satu ronde enkripsi adalah proses pencampuran *state* keluaran pada proses transformasi *MixColumns* dengan kunci. Dalam ronde pertama dan seterusnya yang menjadi kunci adalah hasil dari proses ekspansi kunci awal *(Key Schedule)*. Berikut adalah gambar yang menunjukkan proses *AddRoundKey* antara keluaran *MixColumns* dengan hasil kunci ekspansi.



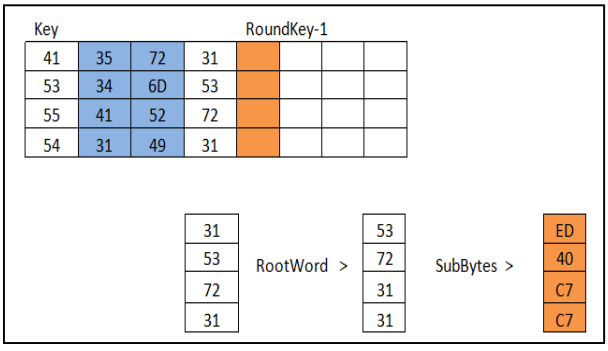
Berikut hasil transformasi *AddRoundKey :*

CE	47	74	46
7D	20	17	39
18	D0	E8	EA
E3	5A	9B	64

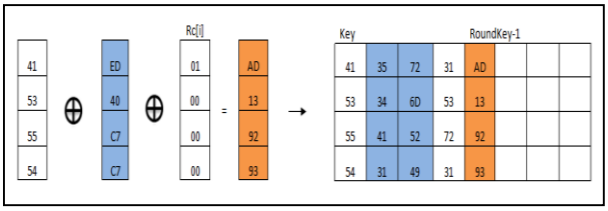
Pada gambar diatas hasil transformasi *AddRoundKey* ronde-1 yang juga merupakan *ciphertext* yang dihasilkan untuk ronde-1 (*state* keluaran ronde-1). Untuk selanjutnya *state* keluaran ronde-1 akan menjadi *state* masukan untuk ronde-2, proses tersebut akan berlangsung hingga *state* keluaran ronde-9 menjadi *state* masukan untuk ronde ke-10 dan *state* keluaran ronde-10 merupakan *ciphertext* yang dihasilkan. Untuk menghasilkan *ciphertext* dalam penyandian algoritma AES membutuhkan kunci ronde untuk setiap ronde transformasi. Kunci ronde ini dibangkitkan (diekspansi) dari kunci algoritma AES yang disebut dengan *Key Shedule*. Berikut adalah dimana proses *Key Shedule* berlangsung.

5. Key Schedule

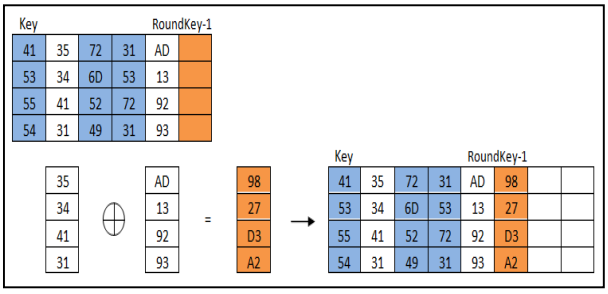
Dalam ekspansi atau pembangkitan kunci dikenal dengan istilah *Key Schedule*. Pada setiap ronde kunci yang digunakan berbeda satu dengan yang lainnya. Kunci ronde pertama adalah hasil dari ekspansi dari kunci awal, kunci ronde kedua adalah hasil dari ekspansi dari kunci ronde pertama dan seterusnya. Dalam hal ini, pembahasan *Key Schedule* akan memperlihatkan bagaimana kunci dibangkitkan pada setiap ronde hingga ronde terakhir (sesuai dengan tipe AES). Berikut ini adalah gambar dari ekspansi kunci.



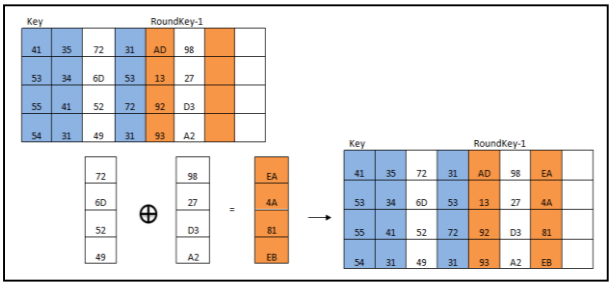
Pada gambar diatas langkah yang pertama untuk mencari kunci pada kolom pertama *round key-1* adalah dengan mencari nilai kunci kolom keempat setelah *RootWord. RootWord* merupakan proses pergeseran posisi nilai kunci teratas bergeser kebawah. Setelah pergeseran, maka melakukan transformasi *SubBytes*. Untuk langkah kedua dapat dilihat pada gambar berikut:



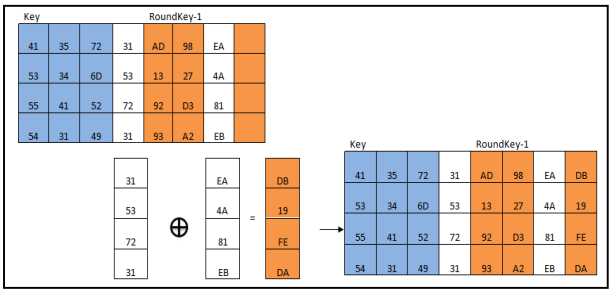
Pada langkah kedua ekspansi kunci, yaitu melakukan perhitungan XOR antara kolom pertama *key*, kolom ketiga *key* yang terlebih dahulu melalui proses *RootWord* dan *SubBytes* dengan Rc[i]. langkah ketiga dapa dilihat pada gambar berikut:



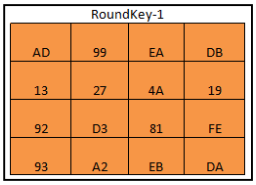
Pada proses langkah ketiga ekspansi kunci, maka melakukan perhitungan XOR antara kolom kedua *key* dengan kolom pertama *Round Key-1* tanpa harus ada proses *RootWord, SubWord* atau perhitungan XOR dengan Rc[i]. untuk langkah selanjutnya dapat dilihat pada gambar berikut:



Proses pada langkah keempat sama dengan proses langkah ketiga, namun pada langkah keempat kolom yang dihitung adalah kolom ketiga pada kunci dan kolom kedua pada *Round Key-1*. Untuk langkah kelima dapat dilihat pada gambar berikut:

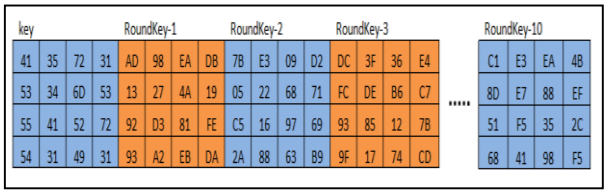


Berikut ini gambar yang menampilkan hasil enkripsinya:

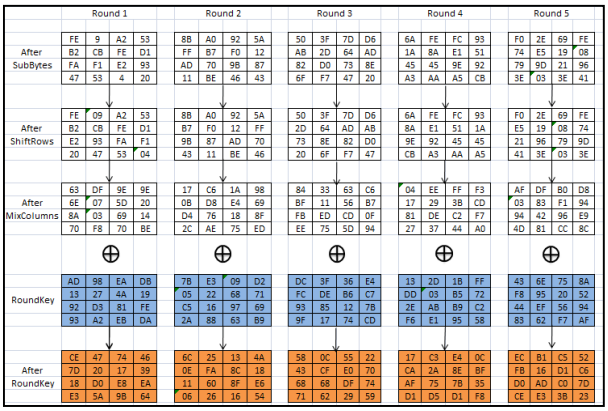


Proses pada langkah kelima juga sama dengan langkah ketiga dan keempat, namun pada langkah keempat kolom yang dihitung adalah kolom keempat pada *key* dengan kolom ketiga *RoundKey-1.*

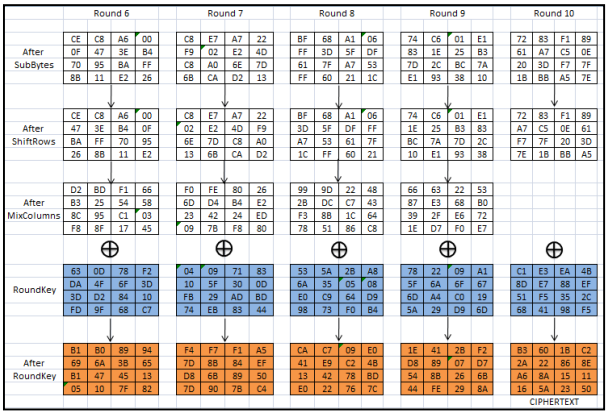
Untuk mencari nilai kunci pada *RoundKey-2,* lakukan langkah yang sama, namun perhitungannya dimulai dari *Round Key-1,*dengan langkah pertama yaitu melakukan proses *RootWord* dan *SubBytes* pada kolom ketiga *Round Key-1*, setelah hasil didapat maka lakukan perhitungan XOR dengan kolom pertama *Round Key-1* dan Rc[i]. untuk Rc[i] dapat dilihat pada tabel 2.6 pada bab II. Berikut ini hasil ekspansi kunci putaran 10.



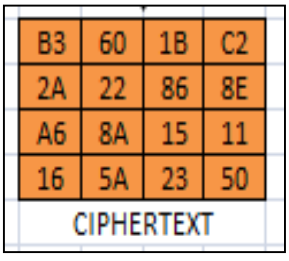
Untuk melihat gambaran enkripsi Ronde 1-5 dapat dilihat pada tampilan berikut:



Sedangkan berikut ini adalah gambaran enkripsi ronde 6-10.



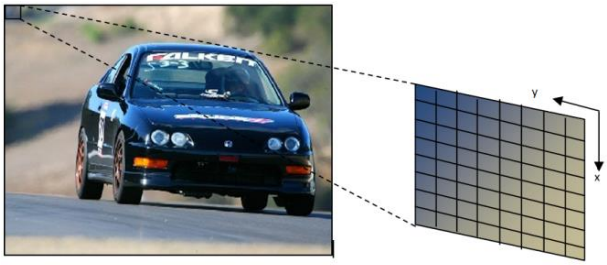
Berikut ini adalah tampilan *ciphertext* yang dihasilkan :



6. Analisis Proses Steganografi Atau Penyisipan Pesan

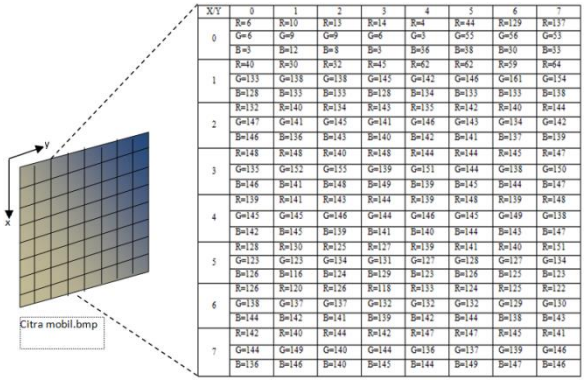
Setelah proses pengenkripsian selesai maka tahap selanjutnya melakukan proses penyisipan pesan (*chipertext*) kedalam media citra “contrast.bmp” berukuran 640 x 246 *pixel* dengan metode LSB. Karena ukuran pesan (*chipertext*) adalah 16 *byte* maka (16 *byte* x 8 = 128 bit) dimana 1 karakter = 1 *byte*, mka

jumlah *pixel* pada media citra tersebut yang dibutuhkan adalah 128 bit/3 bit = 8 *pixel* pertama. Sebagai contoh dari citra 640 x 246 *pixel* ini akan diambil 8 x 8 *pixel* pertama, seperti yang terlihat pada gambar berikut ini:



Citra mobil.bmp (400 x 267 *pixel)* **8 x 8 *pixel* pertama**

Dari pengambilan 8 x 8 *pixel* pertama pada citra contrast.bmp (400 x 267 *pixel*) tersebut, diketahui nilai warnanya sebagai berikut:



X/Y	0	1	2	3	4	5	6	7
0	R=6	R=10	R=13	R=14	R=4	R=44	R=129	R=137
	G=6	G=9	G=9	G=6	G=3	G=55	G=56	G=53
	B=3	B=12	B=8	B=3	B=36	B=38	B=30	B=33
1	R=40	R=30	R=32	R=45	R=62	R=62	R=59	R=64
	G=133	G=138	G=138	G=145	G=142	G=146	G=161	G=154
	B=128	B=133	B=133	B=128	B=134	B=133	B=133	B=138
2	R=132	R=140	R=134	R=143	R=135	R=142	R=140	R=144
	G=147	G=141	G=145	G=141	G=146	G=143	G=134	G=142
	B=146	B=136	B=143	B=140	B=142	B=141	B=137	B=139
3	R=148	R=148	R=140	R=148	R=144	R=144	R=145	R=147
	G=135	G=152	G=155	G=139	G=151	G=144	G=138	G=150
	B=146	B=141	B=148	B=149	B=139	B=145	B=144	B=147
4	R=139	R=141	R=143	R=144	R=139	R=148	R=139	R=148
	G=145	G=145	G=146	G=144	G=146	G=145	G=149	G=138
	B=142	B=145	B=139	B=141	B=140	B=144	B=143	B=147
5	R=128	R=130	R=125	R=127	R=139	R=141	R=140	R=151
	G=123	G=123	G=134	G=131	G=127	G=128	G=127	G=134
	B=126	B=116	B=124	B=129	B=123	B=126	B=125	B=123
6	R=126	R=120	R=126	R=118	R=133	R=124	R=125	R=122
	G=138	G=137	G=137	G=132	G=132	G=132	G=129	G=130
	B=144	B=142	B=141	B=139	B=142	B=144	B=138	B=143
7	R=142	R=140	R=144	R=142	R=147	R=147	R=145	R=141
	G=144	G=149	G=140	G=144	G=136	G=137	G=139	G=146
	B=136	B=146	B=140	B=145	B=144	B=149	B=147	B=146

Setelah itu konversikan nilai dari representasi warna pada gambar 3.23 kedalam biner, sehingga akan terbentuk nilai citra biner sebagai berikut.

Y / X	0	1	2	3	4	5	6	7
0	00000110	00001010	00001101	00001110	00000100	00101100	10000001	10001001
	00000110	00001001	00001001	00000110	00000011	00110111	00111000	00110101
	00000011	00001100	00001000	00000011	00100100	00100110	00011110	00100001
1	00101000	00011110	00100000	00101101	00111110	00111110	00111011	01000000
	10000101	10001010	10001010	10010001	10001110	10010010	10100001	10011010
	10000000	10000101	10000101	10000000	10000110	10000101	10000101	10001010
2	10000100	10001100	10000110	10001111	10000111	10001110	10001100	10010000
	10010011	10001101	10010001	10001101	10010010	10001111	10000110	10001110
	10010010	10001000	10001111	10001100	10001110	10001101	10001001	10001011
3	10010100	10010100	10001100	10010100	10010000	10010000	10010001	10010011
	10000111	10011000	10011011	10001011	10010111	10010000	10001010	10010110
	10010010	10001101	10010100	10010101	10001011	10010001	10010000	10010011
4	10001011	10001101	10001111	10010000	10001011	10010100	10001011	10010100
	10010001	10010001	10010010	10010000	10010010	10010001	10010101	10001010
	10001110	10010001	10001011	10001101	10001100	10010000	10001111	10010011
5	10000000	10000010	01111101	01111111	10001011	10001101	10001100	10010111
	01111011	01111011	10000110	10000011	01111111	10000000	01111111	10000110
	01111110	01110100	01111100	10000001	01111011	01111110	01111101	01111011
6	01111110	01111000	01111110	01110110	10000101	01111100	01111101	01111010
	10001010	10001001	10001001	10000100	10000100	10000100	10000001	10000010
	10010000	10001110	10001101	10001011	10001110	10010000	10001010	10001111
7	10001110	10001100	10010000	10001110	10010011	10010011	10010001	10001101
	10010000	10010101	10001100	10010000	10001000	10001001	10001011	10010010
	10001000	10010010	10001100	10010001	10010000	10010101	10010011	10010010

Lalu sisipkan nilai biner *chipertext* dari "**B3601BC22A22868EA68A1511165A 2350**" yaitu "10110011 01100000 00011011 11000010 00101010 00100010 10000110 10001110 10100110 10001010 00010101 00010001 00010110 01011010 00100011 01010000" kedalam *pixel* biner pada media citra contrast.bmp dengan menggunakan metode LSB, yaitu setiap 1 nilai bit *chipertext* dimasukkan untuk menggantikan posisi akhir dari citra RGB. Seperti yang terlihat pada tabel 3.2, yang merupakan hasil dari penyisipan nilai biner *chipertext* ke *pixel* biner pada citra contrast.bmp, dimana pada bit terakhir merupakan rangkaian bit-bit *ciphertext* yang telah disisipkan dan terdapat beberapa bit yang mengalami perubahan dengan bit-bit aslinya. Berikut tabel hasil penyisipan nilai biner *Ciphertext* ke pixel biner pada citra Mobil.bmp:

X/Y	0	1	2	3	4	5	6	7
0	00000111	00001011	00001101	00001111	00000100	00101100	10000000	10001000
	00000110	00001000	00001001	00000111	00000010	00110110	00111001	00110101
	00000011	00001100	00001000	00000010	00100100	00100110	00011111	00100001
1	00101001	00011110	00100001	00101100	00111111	00111110	00111011	01000000
	10000101	10001010	10001010	10010001	10001110	10010010	10100000	10011011
	10000000	10000100	10000100	10000000	10000111	10000100	10000100	10001010
2	10000101	10001100	10000111	10001110	10000111	10001110	10001101	10010001
	10010010	10001100	10010000	10001100	10010011	10001111	10000110	10001111
	10010010	10001001	10001111	10001100	10001111	10001100	10001000	10001010
3	10010101	10010100	10001101	10010100	10010000	10010001	10010000	10010010
	10000110	10011001	10011010	10001010	10010111	10010000	10001011	10010110
	10010010	10001100	10010100	10010101	10001010	10010000	10010000	10010011
4	10001010	10001101	10001111	10010001	10001011	10010100	10001011	10010100
	10010000	10010000	10010010	10010000	10010010	10010000	10010100	10001011
	10001110	10010001	10001010	10001101	10001101	10010000	10001110	10010011
5	10000000	10000011	01111100	01111111	10001011	10001101	10001100	10010111
	01111011	01111010	10000110	10000011	01111111	10000000	01111111	10000110
	01111110	01110100	01111100	10000001	01111011	01111110	01111101	01111011
6	01111110	01111000	01111110	01110110	10000101	01111100	01111101	01111010
	10001010	10001001	10001001	10000100	10000100	10000100	10000001	10000010
	10010000	10001110	10001101	10001011	10001110	10010000	10001010	10001111
7	10001110	10001100	10010000	10001110	10010011	10010011	10010001	10001101
	10010000	10010101	10001100	10010000	10001000	10001001	10001011	10010010
	10001000	10010010	10001100	10010001	10010000	10010101	10010011	10010010

Setelah penyisipan selesai dilakukan, konversikan kembali nilai biner citra tersebut ke nilai RGB pada citra digital. Dari proses penyisipan ini, maka akan terbentuk susunan nila raster baru. Berikut adalah tabel raster baru yang sudah disisipkan pesan.

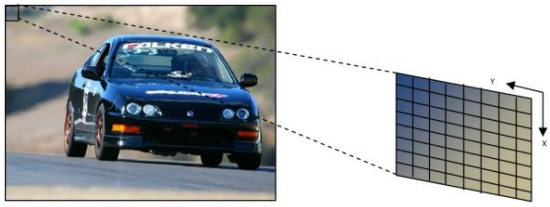
X/Y	0	1	2	3	4	5	6	7
0	R = 7	R = 11	R= 13	R= 15	R= 4	R= 44	R= 128	R= 136
	G = 6	G = 8	G= 9	G = 7	G = 2	G = 54	G = 57	G = 53
	B = 3	B = 12	B= 8	B = 2	B = 36	B = 38	B = 31	B = 33
1	R = 41	R = 30	R = 33	R = 44	R = 63	R = 62	R = 59	R = 64
	G =133	G = 138	G = 138	G = 145	G = 142	G = 146	G = 160	G = 155
	B = 128	B = 132	B = 132	B = 128	B = 135	B = 132	B = 132	B = 138
2	R = 133	R = 140	R = 135	R = 142	R = 135	R = 142	R = 141	R = 145
	G = 146	G = 140	G = 144	G = 140	G = 147	G = 143	G = 134	G = 143
	B = 146	B = 137	B = 143	B = 140	B = 143	B = 140	B = 136	B = 138
3	R = 149	R = 148	R = 141	R = 148	R = 144	R = 145	R = 144	R = 146
	G = 134	G = 153	G = 154	G = 138	G = 151	G = 144	G = 139	G = 150
	B = 146	B = 140	B =148	B = 149	B = 138	B = 144	B = 144	B = 147
4	R = 138	R = 141	R = 143	R = 145	R = 139	R = 148	R = 139	R = 148
	G = 144	G = 144	G = 146	G = 144	G = 146	G = 144	G = 148	G = 139
	B = 142	B = 145	B = 138	B = 141	B = 141	B = 144	B = 142	B = 147
5	R = 128	R = 131	R = 124	R = 127	R = 139	R = 141	R = 140	R = 151
	G = 123	G = 122	G = 134	G = 131	G = 127	G = 128	G = 127	G = 134
	B = 126	B = 116	B = 124	B = 129	B = 123	B = 126	B = 125	B = 123
6	R = 126	R = 120	R = 126	R = 118	R = 133	R = 124	R = 125	R = 122
	G = 138	G = 137	G = 137	G = 132	G = 132	G = 132	G = 129	G = 130
	B = 144	B = 142	B = 141	B = 139	B = 142	B = 144	B = 138	B = 143
7	R = 142	R = 140	R = 144	R = 142	R = 147	R = 147	R = 145	R = 141
	G = 144	G = 149	G = 140	G = 144	G = 136	G = 137	G = 139	G = 146
	B = 136	B = 146	B = 140	B = 145	B = 144	B = 149	B = 147	B = 146

7. Proses Ekstraksi Pesan

Proses ekstraksi pesan merupakan proses pengungkapan atau pengambilan pesan rahasia yang terdapat di dalam citra stego. Proses pengambilan pesan rahasia di dalam citra stego ini merupakan kebalikan dari proses penyisipan pesan dengan LSB, dimana bit-bit LSB yang sekarang berada pada citra stego diambil satu persatu dan disatukan kembali sehingga menjadi sebuah informasi. Proses ini disebut dengan istilah *extraction* sehingga pesan akan dilakukan proses enkripsi dengan menggunakan kunci yang sama pada saat enkripsi pesan.

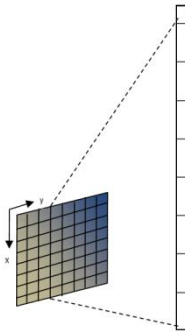
Misalkan gambar merupakan merupakan citra stego contrast.bmp dari hasil proses penyisipan pesan rahasia. Citra ini akan dilakukan proses ekstraksi pesan untuk mendapatkan kembali informasi yang disisipkan di dalamnya, dimana tahap-tahap proses ekstrasinya adalah sebagai berikut :

1. Citra stego tersebut akan dilakukan proses pengungkapan terhadap pesan rahasia di dalamnya. Sebagai contoh dari citra contrast.bmp ukuran 640 x 246 *pixel* ini, akan diambil 8 x 8 *pixel* pertama seperti yang terlihat pada gambar berikut:



Citra stego mobil.bmp (400 x 267 *pixel*)

2. Dari 8 x 8 *pixel* pertama pada citra stego mobil.bmp (400 x 267 *pixel*) tersebut, diketahui nilai representasi warnanya sebagai berikut ditunjukkan pada gambar citra mobil.bmp diatas.



X/Y	0	1	2	3	4	5	6	7
0	R=7	R=11	R=13	R=15	R=4	R=44	R=128	R=136
	G=6	G=8	G=9	G=7	G=2	G=54	G=57	G=53
	B=3	B=12	B=8	B=2	B=36	B=38	B=31	B=33
1	R=41	R=30	R=33	R=44	R=63	R=62	R=59	R=64
	G=133	G=138	G=138	G=145	G=142	G=146	G=160	G=155
	B=128	B=132	B=132	B=128	B=135	B=132	B=132	B=138
2	R=135	R=140	R=135	R=142	R=135	R=142	R=141	R=145
	G=146	G=140	G=144	G=140	G=147	G=143	G=134	G=143
	B=146	B=137	B=143	B=140	B=143	B=140	B=136	B=138
3	R=149	R=148	R=141	R=148	R=144	R=145	R=144	R=146
	G=134	G=153	G=154	G=138	G=151	G=144	G=139	G=150
	B=146	B=140	B=148	B=149	B=138	B=144	B=144	B=147
4	R=138	R=141	R=143	R=145	R=139	R=148	R=139	R=148
	G=144	G=144	G=146	G=144	G=146	G=144	G=148	G=139
	B=142	B=145	B=138	B=141	B=141	B=144	B=142	B=147
5	R=128	R=131	R=124	R=127	R=139	R=141	R=140	R=151
	G=125	G=122	G=134	G=131	G=127	G=128	G=127	G=134
	B=126	B=116	B=124	B=129	B=123	B=126	B=125	B=123
6	R=126	R=120	R=126	R=118	R=133	R=124	R=125	R=122
	G=138	G=137	G=137	G=132	G=132	G=132	G=129	G=130
	B=144	B=142	B=141	B=139	B=142	B=144	B=138	B=143
7	R=142	R=140	R=144	R=142	R=147	R=147	R=145	R=141
	G=144	G=149	G=140	G=144	G=136	G=137	G=139	G=146
	B=136	B=146	B=140	B=145	B=144	B=149	B=147	B=146

3. Konversi nilai dari representasi warna tersebut kedalam biner, sehingga akan terbentuk nilai yang ditunjukkan pada citra sebagai berikut :

X/Y	0	1	2	3	4	5	6	7
0	00000111	00001011	00001101	00001111	00000100	00101100	10000001	10001000
	00000110	00001000	00001001	00000111	00000010	00110110	00111001	00110101
	00000011	00001100	00001000	00000010	00100100	00100110	00011111	00100001
1	00101001	00011110	00100001	00101100	00111111	00111110	00111011	01000000
	10000101	10001010	10001010	10010001	10001110	10010010	10100000	10011011
	10000000	10000100	10000100	10000000	10000111	10000100	10000100	10001010
2	10000101	10001100	10000111	10001110	10000111	10001110	10001101	10010001
	10010010	10001100	10010000	10001100	10010011	10001111	10000110	10001111
	10010010	10001001	10001111	10001100	10001111	10001100	10001000	10001010
3	10010101	10010100	10001101	10010100	10010000	10010001	10010000	10010010
	10000110	10011001	10011010	10001010	10010111	10010000	10001011	10010110
	10010010	10001100	10010100	10010101	10001010	10010000	10010000	10010011
4	10001010	10001101	10001111	10010001	10001011	10010100	10001011	10010100
	10010000	10010001	10010010	10010000	10010010	10010000	10010100	10001011
	10001110	10010001	10001010	10001101	10001101	10010000	10001110	10010011
5	10000000	10000011	01111100	01111111	10001011	10001101	10001100	10010111
	01111011	01111010	10000110	10000011	01111111	10000000	01111111	10000110
	01111110	01110100	01111100	10000001	01111011	01111110	01111101	01111011
6	01111110	01111000	01111110	01110110	10000101	01111100	01111101	01111010
	10001010	10001001	10001001	10000100	10000100	10000100	10000001	10000010
	10010000	10001110	10001101	10001011	10001110	10010000	10001010	10001111
7	10001110	10001100	10010000	10001110	10010011	10010011	10010001	10001101
	10010000	10010101	10001100	10010000	10001000	10001001	10001011	10010010
	10001000	10010010	10001100	10010001	10010000	10010101	10010011	10010010

4. Dari data citra biner pada tabel 3.4, maka akan di ambil kembali bit-bit LSB yang disisipkan di dalamnya, nilai bit-bit LSB ini akan disatukan dan disusun kembali menjadi bentuk biner, sehingga terbentuk nilai-nilai LSB (*ciphertext*) yaitu “10110011 01100000 00011011 11000010 00101010 00100010 10000110 10001110 10100110 10001010 00010101 00010001 00010110 01011010 00100011 01010000“

5. Nilai biner yang terbentuk akan dikonversikan kedalam bilangan ASCII sehingg dapatlah *ciphertext* "**B3601BC22A22868EA68A1511165A2350** "

B. Algoritma Dekripsi Aes-128

Proses dekripsi AES tidak jauh berbeda dengan proses enkripsi, untuk *state* yang dibentuk adalah hasil dari proses enkripsi dan melakukan proses dari kelebihan enkripsi. Untuk melakukan proses dekripsi empat transformasi yang dilakukan adalah *AddRoundKey, InvMixColumns, InvShiftRows* dan *InvSubBytes*. Berikut adalah hasil dari keseluruhan proses enkripsi.

B3	60	1B	C2
2A	22	86	8E
A6	8A	15	11
16	5A	23	50

CIPHERTEXT

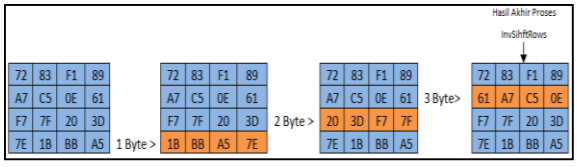
1. Transformasi AddRoundKey

Pada proses dekripsi *AddRoundKey*, *state* melakukan operasi XOR dengan kunci masukan yang sama pada saat melakukan proses enkripsi, dengan catatan untuk dekripsi kunci untuk pra ronde masukan adalah hasil ekspansi kunci ronde-10 pada proses enkripsi.

State					Key					Output			
B3	60	1B	C2	⊕	C1	E3	EA	4B	=	72	83	F1	89
2A	22	86	8E		8D	E7	88	EF		A7	C5	0E	61
A6	8A	15	11		51	F5	35	2C		F7	7F	20	3D
16	5A	23	50		68	41	98	F5		7E	1B	BB	A5

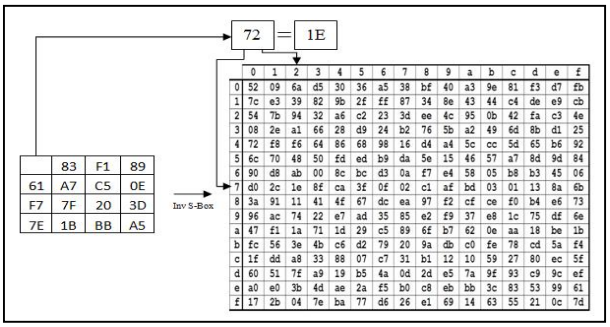
2. Transformasi InvShiftRows

Pada proses dekripsi *InvShiftRows*, *state* melakukan proses pergeseran posisi dari nilai *array* yang terdapat pada *state* seperti pada proses enkripsi *ShiftRows*, namun proses pergeseran *InvShiftRows* yaitu dimulai dari baris keempat dengan melakukan pergeseran satu blok nila *array* dan baris kedua melakukan pergeseran sebanyak tiga blok *array*, baris pertama tidak ada pergeseran.



3. Transformasi InvSubBytes

Pada proses dekripsi, *InvSubBytes* sama dengan proses *SubBytes* pada enkripsi, akan tetapi proses *InvSubBytes* menggunakan tabel transformasi *InvSubBytes* (dapat dilihat pada pembahasan sebelumnya).



	83	F1	89
61	A7	C5	0E
F7	7F	20	3D
7E	1B	BB	A5

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

Berikut ini hasil Transformasi InvSubBytes:

1E	41	2B	F2
D8	89	07	D7
54	8B	26	6B
44	FE	29	8A

Selanjutnya hasil dari transformasi *InvSubBytes* di XOR kan dengan *AddRoundKey* lalu dilakukan proses trasnformasi *InvMixColumns.*

4. Transformasi InvMixColumns

Pada transformasi *InvMixColumns* dihitung dengan rumus penjumlahan dan perkalian serta menggunakan tabel E dan L *Galois Field Multiplication*. Transformasi InvMixCoulumns sama dengan Mix-Columns, dimana perbedaannya adalah a(x) yang digunakan adalah inversnya (a-1) yaitu :

$$a^{-1}(x) = \{0b\}x^3 | + \{0d\}x^2 + \{09\}x + \{0e\}$$

Sehingga :

$s'(x) = a^{-1}(x) \oplus s(x)$:

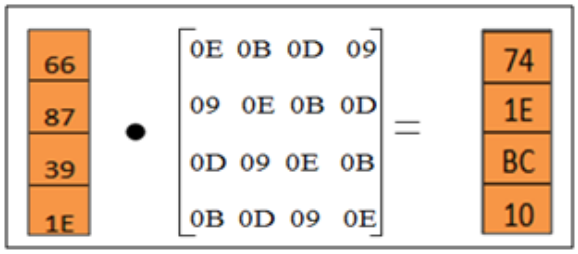
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

$S'_{0,c} = (\{0e\} * S_{0,c}) \oplus (\{0b\} * S_{1,c}) \oplus (\{0d\} * S_{0,c}) \oplus (\{09\} * S_{0,c})$

$S'_{1,c} = (\{09\} * S_{0,c}) \oplus (\{0e\} * S_{1,c}) \oplus (\{0b\} * S_{0,c}) \oplus (\{0d\} * S_{0,c})$

$S'_{2,c} = (\{0d\} * S_{0,c}) \oplus (\{09\} * S_{1,c}) \oplus (\{0e\} * S_{0,c}) \oplus (\{0b\} * S_{0,c})$

$S'_{3,c} = (\{0b\} * S_{0,c}) \oplus (\{0d\} * S_{1,c}) \oplus (\{09\} * S_{0,c}) \oplus (\{0e\} * S_{0,c})$



Berikut ini hasil transformasi InvMixColumns:

74	C6	01	E1
1E	25	B3	83
BC	7A	7D	2C
10	E1	93	38

Rumus di atas adalah rumus yang digunakan secara umum untuk mencari nilai invers dari *state* untuk proses *InvMixColumns*. Secara detail rumus dapat dijelaskan sebagai berikut dengan menggunakan tabel E dan L pada *Galois Field Multiplication*. Misal : Input = 66 87 39 1E
Output:

= (66 * 0E)⊕(87*0B)⊕(39*0D)⊕(1E*09)
= E(L(66)+L(0E))⊕E(L(87)+L(0B))⊕E(L(39)+L(0D))⊕E(L(1E)+L(09))
= E(1E+DF)⊕E(74+68)⊕E(F0+EE)⊕E(1C+C7)
= E(FD)⊕E(DC)⊕E(1DE-FF)⊕E(E3)
= E(FD)⊕E(DC)⊕E(DF)⊕E(E3)
= 52⊕C6⊕0E⊕EE
= **74**

Setelah itu lakukan hal yang sama sampai iterasi ke-10 namun tidak dilakukan *MixColumns* lagi, melainkan langsung melakukan XOR hasil dari Inv*ShiftRows* dengan *RoundKey*.

4.Kesimpulan

1. Aplikasi yang digunakan untuk melakukan enkripsi dan dekripsi pesan yaitu menggunakan teknik kriptografi dengan algoritma AES.
2. Untuk melakukan proses penyisipan pesan, teknik steganografi yang digunakan dalam menyisipkan pesan rahasia yang sudah dienkripsi kedalam citra bitmap yaitu menggunakan teknik steganografi dengan metode LSB.
3. Pada proses ekstraksi Pesan, data yang sudah disisipkan kedalam media citra bitmap dapat diperoleh kembali dengan secara utuh.
4. Semakin besar ukuran citra digital yang digunakan maka semakin besar pula kapasitas pesan yang adapat disisipkan pada citra tersebut.

Daftar Pustaka

[1] Rifki Sadikin, 2012, ***"Kriptografi untuk Keamanan Jaringan"***, Yogyakarta, Penerbit Andi
[2] Dony Ariyus, 2009, ***"Keamanan Multimedia"***, Yogyakarta, Penerbit Andi
[3] T. Sutoyo, S.Si., M.Kom., Edy Mulyanto, S.Si., M.Kom., Dr. Vincent Suhartono, Oky Dwi Nurhayati, [4]
[4] M.T., Wijanarto, M.Kom, 2009, ***"Teori Pengolahan Citra Digital"***, Yogyakarta, Penerbit Andi
[5] Wahana Komputer, 2009, ***" Visual Basic 2008",*** Yogyakarta, Penerbit Andi