BAB II

LANDASAN TEORI

2.1 Surat Izin Mengemudi

Di Indonesia, Surat Izin Mengemudi (SIM) adalah bukti registrasi dan identifikasi yang diberikan oleh Polri kepada seseorang yang telah memenuhi persyaratan administrasi, sehat jasmani dan rohani, memahami peraturan lalu lintas dan terampil mengemudikan kendaraan bermotor. Setiap orang yang mengemudikan kendaraan bermotor di jalan wajib memiliki Surat Izin Mengemudi sesuai dengan jenis kendaraan bermotor yang dikemudikan. Surat Izin Mengemudi di Indonesia terdapat dua (2) jenis (Pasal 77 ayat (2) UU No. 22 Tahun 2009) [5]:

1. Surat Izin Mengemudi Kendaraan Bermotor perseorangan

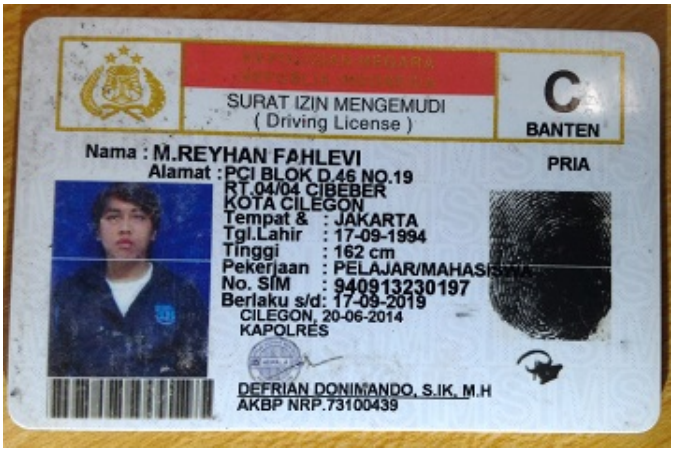
    (a) SIM A, untuk mengemudikan mobil penumpang dan barang perseorangan dengan jumlah berat yang diperbolehkan tidak melebihi 2.500 kg.

    (b) SIM B1, untuk mengemudikan mobil penumpang dan barang perseorangan dengan jumlah berat yang diperbolehkan lebih dari 3.500 kg.

    (c) SIM B2, untuk mengemudikan Kendaraan alat berat, Kendaraan penarik, atau Kendaraan Bermotor dengan menarik kereta tempelan atau gandengan perseorangan dengan berat yang diperbolehkan untuk kereta tempelan atau gandengan lebih dari 1.000 kg.

    (d) SIM C, untuk mengemudikan Sepeda Motor.

    (e) SIM D, untuk mengemudikan kendaraan khusus bagi penyandang cacat.



Gambar 2.1: SIM C

2. Surat Izin Mengemudi Kendaraan Bermotor Umum

   (a) SIM A Umum, untuk mengemudikan kendaraan bermotor umum dan barang dengan jumlah berat yang diperbolehkan tidak melebihi 3.500 kg.

   (b) SIM B1 Umum, untuk mengemudikan mobil penumpang dan barang umum dengan jumlah berat yang diperbolehkan lebih dari 3.500 kg.

   (c) SIM B2 Umum, untuk mengemudikan Kendaraan penarik atau Kendaraan Bermotor dengan menarik kereta atau gandengan dengan berat yang diperbolehkan untuk kereta tempelan atau gandengan lebih dari 1.000 kg.

2.2 Arduino

Arduino merupakan nama dari salah satu famili papan mikrokontroler. Papan ini berupa kombinasi antara mikroprosesor ATMEL termasuk RAM, *flash memory*, dan kanal I/O. Arduino ini memiliki struktur umum seperti komputer, akan tetapi kemampuannya tidak sekompleks yang dimiliki oleh komputer [12].

Arduino memiliki berbagai macam pilihan model [7] tergantung kebutuhan, diantaranya:

1. Arduino Mini dan Pro Mini, digunakan apabila kita membutuhkan ukuran yang cukup kecil untuk proyek yang ingin dibuat.

2. Arduino Mega, digunakan jika kita perlu rangkaian yang lumayan banyak, prosesor yang lebih cepat, dan memiliki kapasitas pin yang lebih banyak.

3. Arduino LilyPad, biasanya LilyPad digunakan untuk perangkat *wearable*. Misalnya, baju dengan hiasan LED.

4. Arduino Fio, biasa digunakan untuk proyek yang membutuhkan koneksi nirkabel (*wireless*).

5. Arduino Pro, digunakan untuk *embedded application* (proyek mandiri yang menggunakan baterai)

6. Arduino Uno, merupakan arduino yang paling banyak digunakan oleh pemula karena sudah tersedia USB dan *Power Slot*. Menggunakan mikrokontroler tambahan untuk mengatur komunikasi USB. Bisa dibilang model inilah yang paling lengkap.

Pada penelitian ini penulis menggunakan Arduino Uno R3, karena memiliki spesifikasi yang mencukupi untuk digunakan dalam penelitian ini.

Arduino Uno memiliki 32 kB memori *flash*, beroperasi pada kecepatan *clock* 16 MHz, dan memiliki 14 pin I/O digital dan 6 pin I/O analog. Bagian lain dari Arduino adalah *programming environtment*, dimana kode pemrograman yang dibuat, ditulis dalam bahasa C yang disederhanakan, dan ditransfer ke Arduino menggunakan kabel USB. Setelah pemrograman, Arduino dapat bekerja ketika sedang terhubung ke PC (mengirimkan data) , atau beroperasi secara mandiri menggunakan sumber tenaga lain seperti adaptor 5V-12V.

Berikut ini adalah spesifikasi Lengkap dari Arduino Uno R3 yang digunakan dalam penelitian ini[13]:

Tabel 2.2.1: Spesifikasi Arduino UNO R3

Tegangan Operasi	5 VDC
Rekomendasi Tegangan	7-12 VDC
Batas Tegangan	6-20 VDC
Jumlah I/O Digital	14
Jumlah I/O Analog	6
DC Current setiap pin I/O	40mA
DC Current untuk pin 3.3V	50mA.
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

Gambar 2.2: Mikrokontroler Arduino UNO R3

2.3 RFID

Radio Frequency Identification (RFID) adalah teknologi yang memanfaatkan gelombang radio sebagai media transmisi data untuk mendeteksi objek secara otomatis. Teknologi ini terdiri dari dua komponen utama, yaitu RFID *reader* dan RFID *tag*. RFID *reader* berfungsi sebagai *transmitter* yang memancarkan sinyal sehingga RFID tag dideteksi dan pentransferan data dapat dilakukan[11].

RFID *tag* (Gambar 2.3 Kiri) mengandung informasi dan *reader* akan membaca *tag* untuk informasi. *Tag* disebut juga sebagai *transponder* dimana kata ini diambil dari kata *transmitter* dan *responder*. *Tag* merupakan tanda pengenal yang dipasang atau ditempel pada suatu benda atau objek yang menyimpan informasi. *Tag* merespon permintaan *reader* untuk mengirimkan informasi yang dimiliki. *Tag* terdiri dari *microchip* yang terhubung ke antena atau baterai. *Microchip* ini memiliki memori yang dapat menampung data hingga 128 KB[3].

Gambar 2.3: RFID *tag* dan RFID *reader*

RFID *reader* (Gambar 2.3 Kanan), atau bisa disebut juga sebagai *interrogator* adalah sebuah alat yang digunakan untuk membaca satu atau lebih tag yang berada di jangkauannya dan berkomunikasi dengannya. *Reader* ini terdiri dari satu atau lebih antena yang memancarkan gelombang radio dan menerima sinyal dari satu atau lebih

tag. Alat ini mengirim permintaan sebagai sinyal yang menginterogasi informasi identifikasi dari tag. Tag akan merespon dan mengirimkan informasi yang telah di *encode* kepada *reader*, informasi yang dikirimkan akan di *decode* oleh *reader*.

Secara singkat RFID *tag* dan RFID *reader* dapat disimpulkan sebagai berikut:

1. RFID *tag* (*transponder*) yang terdiri dari sebuah perangkat kecil yang tertanam di dalam sebuah benda atau objek seperti label, *smart card* dan lainya yang memiliki identifikasi yang unik dan memori yang dapat ditulis.

2. RFID *reader* merupakan sebuah device yang dapat berkomunikasi tanpa kontak langsung dengan suatu tag untuk mengidentifikasikannya apabila terhubung di dalam suatu asosiasi data komunikasi tanpa kontak langsung (*wireless*) pada radio frekuensi.

Frekuensi yang digunakan oleh sistem RFID dibuat pada frekuensi tertentu dan ada 4 macam yaitu *Band* LF (*Low Frequency*), *Ba*nd HW (*High Frequency*), UHF (*Ultra High Frequency*) dan Gelombang mikro 2,4 GHz. Pada penelitian ini digunakan *Band* HW. *Band* HW (*High Frequency*) yang beroperasi pada frekuensi 13.56 MHz dengan jarak pembacaan secara teori hingga kurang lebih 3 m, frekuensi ini cocok digunakan untuk pembacaan pada tingkat *item* dan banyak digunakan untuk pencocokan barang-barang di toko, gedung atau pelacakan yang memerlukan dengan kecepatan baca 10 hingga 100 *tag* per detik[11].

2.4 Kartu RFID

Penulis menggunakan kartu RFID *Mifare Classic S50* yang menawarkan penyimpanan sebesar 1Kb atau 1024 *bytes*. Penyimpanannya dibagi ke dalam 16 sektor, tiap-tiap sektor dilindungi oleh kunci yang disebut kunci A dan kunci B. Kartu ini memiliki spesifikasi sebagai berikut:

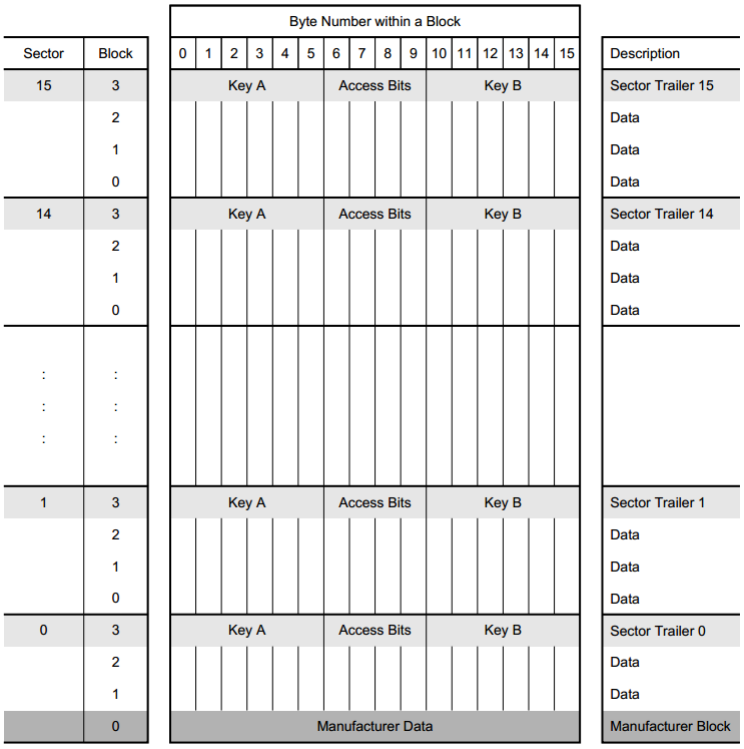
Tabel 2.4.1: Spesifikasi *Mifare Classic S50*

Fungsi	baca/tulis
Chip	Philips MF1 IC S50
Memory	1Kbytes EEPROM
Frekuensi	13.56 MHz
Suhu Operasional	-10 s/d 50 ^{o}C
Suhu Penyimpanan	-10 s/d 60 ^{o}C
Jenis Bahan	PVC/PET
Dimensi	85,6 x 54,0 x 0,8 (mm) ISO CR80

2.4.1 Struktur Memori Kartu

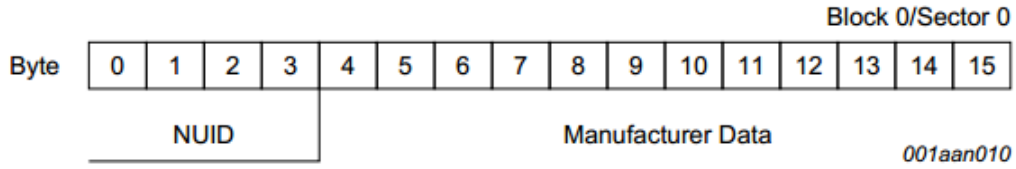
Kartu RFID Mifare S50 mempunyai 1 *Kbytes* lokasi memori yang dibagi menjadi 16 sektor [8]. Setiap sektor terdiri dari 4 blok masing-masing berukuran 16 *byte* dengan nomor (0-3) yang berisi:

Blok 0 - Data*
Blok 1 - Data
Blok 2 - Data
Blok 3 - *Sector Trailer*



Gambar 2.4: Struktur Memori Mifare S50

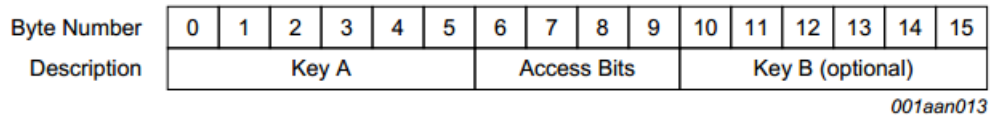
Khusus pada sektor 0, blok 0 berisi ID yang sudah terprogram dari pabrik. Data pada blok ini tidak bisa dimodifikasi (*write protected*)[8]. Pada sektor lain, blok 0 dapat diisi dengan data.



Gambar 2.5: Blok ID Kartu RFID

Blok 3 berisi kunci (*keys*) dan kondisi akses (*access condition*) untuk seluruh 4 blok yang ada dalam sektor termasuk blok itu sendiri[8]. Hanya ada satu pasangan kunci untuk sektor ini, tapi terdapat kondisi akses yang unik untuk kunci pada setiap blok:

Security Key A
Acces Condition
For Block 0
For Block 1
For Block 2
For Block 3
Security Key B



Gambar 2.6: Blok *Sector Trailer*

2.5 Algoritma Kriptografi AES (*Advanced Encryption Standard*)

Dalam kriptografi, *Advanced Encryption Standard* (AES) merupakan standar enkripsi dengan kunci-simetris yang diadopsi oleh pemerintah Amerika Serikat. AES menggunakan algoritma Rijndael (Vincent Rijmen dan Joan Daemen) yang telah memenangkan sayembara terbuka yang diselenggarakan oleh NIST (*National Institute of Standard and Technology*). Sayembara ini diselenggarakan untuk menemukan algoritma enkripsi baru pengganti algoritma DES (*Data Encryption Standard*) yang dianggap sudah tidak aman lagi.

Algoritma Rijndael (telah menjadi AES) terbagi menjadi AES-128, AES-192, dan AES-256. Pengelompokan jenis AES ini berdasarkan panjang kunci yang digunakan. Angka-angka di belakang kata AES menggambarkan panjang kunci yang digunakan pada tipa-tiap AES. Selain itu, hal yang membedakan dari masing-masing AES ini adalah banyaknya round yang dipakai. AES-128 menggunakan 10 *round*, AES-192 sebanyak 12 *round*, dan AES-256 sebanyak 14 *round*.

Karena AES mempunyai panjang kunci paling sedikit 128-bit, AES mampu bertahan terhadap serangan *exhaustive key search*. Dengan panjang kunci 128-bit,

Tabel 2.5.1: Properti AES

AES	Panjang Kunci (N_k words)	Ukuran Blok (N_b blok)	Jumlah Round (N_r)
128	4	4	10
192	6	4	12
256	8	4	14

maka terdapat sebanyak $2^{128} = 3,4 \times 10^{38}$ kemungkinan kunci yang dapat dibuat. Jika digunakan komputer tercepat saat ini yaitu komputer *Taihulight* dari Cina dengan kecepatan 93 petaflops[14], maka akan membutuhkan waktu:

$$\begin{aligned} 1 \text{ Petaflops} &= 10^{15} \\ \text{Kombinasi Kunci / Tahun} &= 93 \times 10^{15} \times 60 \times 60 \times 24 \times 365 \\ &= 2,9 \times 10^{24} \\ ETA &= \frac{3,4 \times 10^{38}}{2,9 \times 10^{24}} = 1,16 \times 10^{14} \text{ Tahun} \end{aligned}$$

2.5.1 Algoritma Rijndael

Seperti pada DES, Rijndael menggunakan substitusi dan permutasi, dan sejumlah putaran (cipher berulang), pada setiap putaran menggunakan kunci internal yang berbeda (kunci setiap putaran disebut *round key*). Tidak seperti DES yang berorientasi bit, Rijndael beroperasi dalam orientasi byte (untuk mengefektifkan implementasi algoritma ke dalam *software* dan *hardware*).

Berikut ini adalah garis besar algoritma Rijndael yang beroperasi pada blok 128-bit dengan kunci 128-bit (di luar proses pembangkitan *round key*)[10]:

1. *AddRoundKey*: melakukan operasi XOR antara kondisi sebelumnya (plainteks) dengan cipher key. Tahap ini disebut sebagai putaran inisiasi (initial round).

2. Putaran sebanyak $N_r - 1$ kali dimana N_r adalah banyaknya putaran. Proses

yang dilakukan pada setiap 1 kali putaran adalah:

(a) *SubBytes*: mensubtitusi byte dengan menggunakan tabel substitusi (S-box).

(b) *ShiftRows*: pergeseran baris-baris array state dengan menggunakan metode wrapping.

(c) *MixColumns*: mengacak data di masing-masing kolom *array state*.

(d) *AddRoundKey*: melakukan operasi XOR antara kondisi sekarang dari *round key*.

3. *Final Round*: proses untuk putaran akhir hanya terdiri adari 3 operasi:

(a) *SubBytes*

(b) *ShiftRows*

(c) *AddRoundKey*

Pada subbab berikutnya akan dijelaskan lebih lanjut mengenai proses enkripsi dan proses dekripsi yang dilakukan oleh algoritma AES-128/Rijndael.

2.5.2 Proses Enkripsi

Dalam penggunaannya proses enkripsi AES-128 menerima tiga parameter yaitu *plaintext*, *ciphertext*, dan *key* dengan deskripsi masing-masing sebagai berikut:

1. *PlainText* : berupa *array* berukuran 16-*byte* sebagai data masukan atau pesan yang ingin dienkripsi.

2. *CipherText* : berupa *array* berukuran 16-*byte* sebagai *storage* yang digunakan untuk menyimpan hasil enkripsi.

3. *Key* : berupa *array* berukuran 16-*byte* sebagai kunci yang digunakan untuk mengenkripsi *PlainText*.

Algoritma 1.1 Fungsi Enkripsi

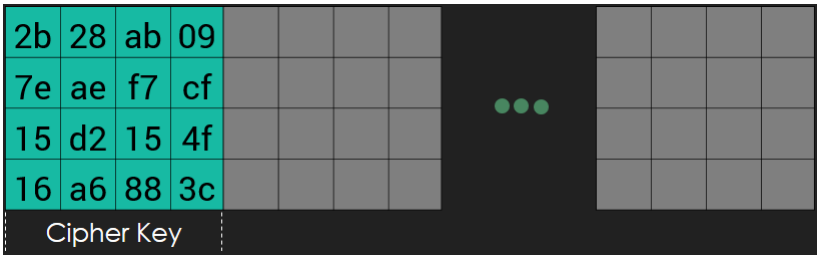
```
KeyExpansion(key, rk);
AddRoundKey(state, rk[0]);
for (r = 1; r <= ROUNDS - 1; r++) {
  SubBytes(state);
  ShiftRows(state);
  MixColumns(state);
  AddRoundKey(state, rk[r]);
}
SubBytes(state);
ShiftRows(state);
AddRoundKey(state, rk[ROUNDS]);
```

Pada proses enkripsi kode di atas, status plainteks saat ini disimpan pada *array* 2 dimensi *state* dengan ukuran $NROWS \times NCOLS$ dimana $NROWS$ adalah jumlah baris dan $NCOLS$ jumlah kolom dari array. Jika data berukuran 128-bit maka ukuran *state* adalah 4×4. Elemen *array state* diacu sebagai $S[r, c]$, dengan $0 \leq r < 4$ dan $0 \leq c < Nb$ dimana *Nb* adalah panjang blok dibagi dengan 32.

2.5.2.1 Ekspansi Kunci

Sebelum memasuki blok utama pengenkripsian, terlebih dahulu dilakukan ekspansi kunci. Proses ini bertujuan untuk membangkitkan sejumlah *round key* yang disimpan pada array *rk* dimana *rk* adalah variabel tempat penyimpanan *round key*. Banyaknya *round key* bergantung pada jumlah putaran atau *round* (Tabel 2.5.1) yang dilakukan.

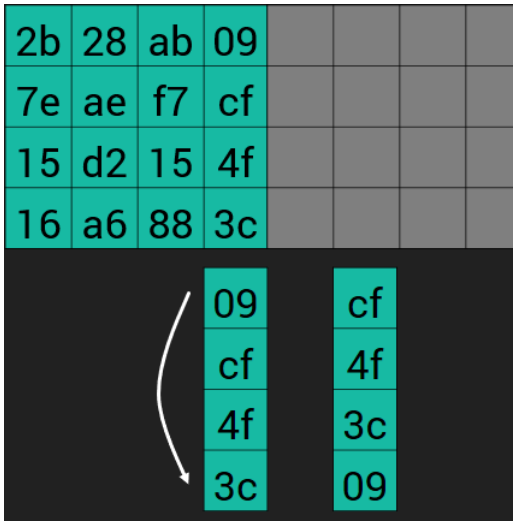
Ekspansi kunci dilihat sebagai *array* 32 bit *words* (*columns*), memiliki indeks dari 0 - 43 dengan 4 indeks pertama berisi *array chiper key*.



Gambar 2.7: Inisialisasi rk

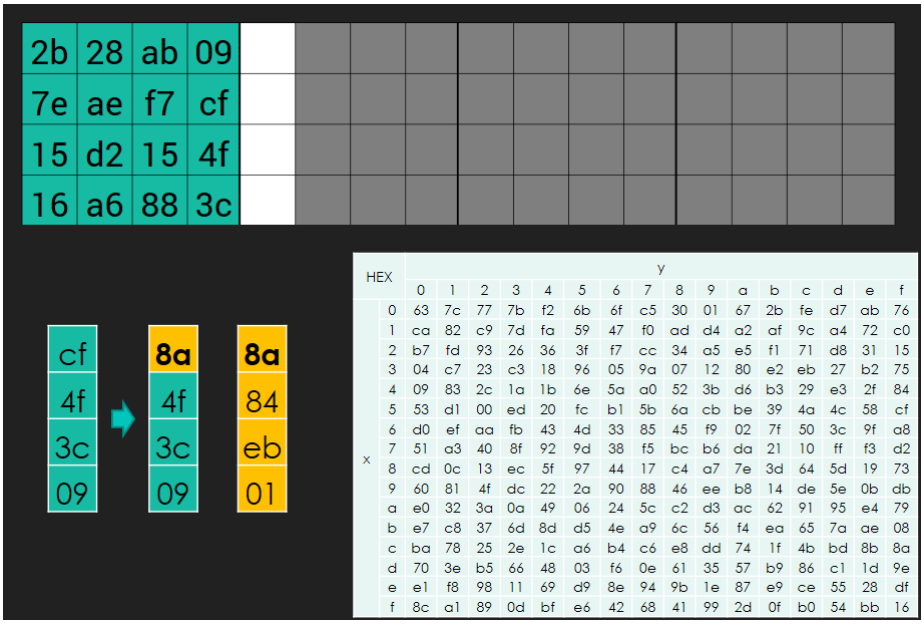
Kolom *rk* (W_i) dengan i kelipatan 4 ($W_4, W_8, ..., W_{40}$) dikalkulasi dengan cara sebagai berikut:

1. ***RotWord***, melakukan rotasi pada array W_{i-1}.



Gambar 2.8: Operasi *RotWord*

2. ***SubBytes***, melakukan transformasi *SubBytes* terhadap array W_{i-1} dengan melihat tabel subtitusi *S-Box*.

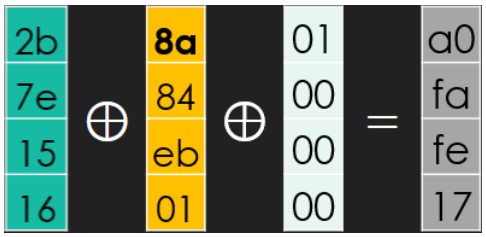


Gambar 2.9: Operasi *SubBytes*

3. ***XOR***, melakukan operasi XOR terhadap W_{i-4} dengan hasil perhitungan sebelumnya, dan terakhir dilakukan operasi XOR dengan *round constant* ($rcon(i)$).

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

Gambar 2.10: *Rcon lookup*

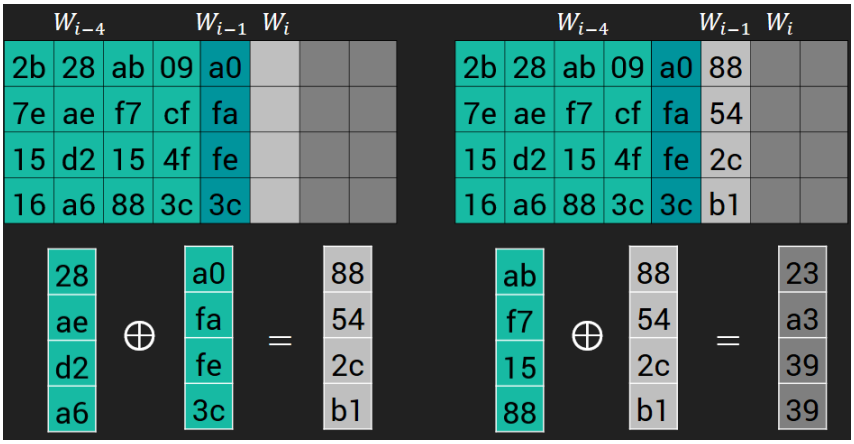


Gambar 2.11: Operasi *XOR*

Gambar 2.12: Hasil Kalkukasi W_4

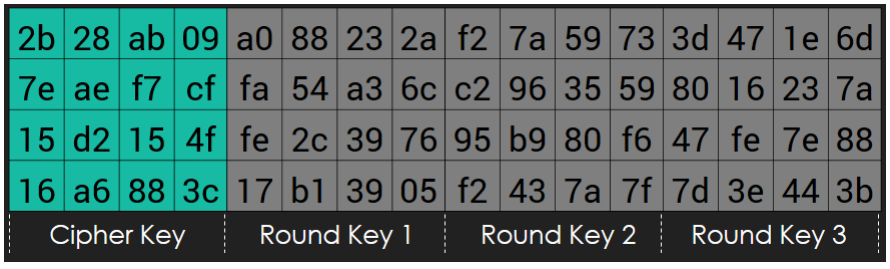
Kolom rk (W_i) dengan i selain kelipatan 4 dikalkulasikan dengan melakukan operasi XOR kolom sebelumnya (W_{i-1}) dengan 4 kolom sebelumnya (W_{i-4}).

$$W_i = W_{i-1} \oplus W_{i-4} \tag{2.1}$$



Gambar 2.13: Hasil Kalkukasi W_i bukan kelipatan 4

Langkah-langkah di atas diiterasi hingga sehingga didapatkan keseluruhan kalkulasi *round key* telah didapatkan.



Gambar 2.14: Hasil ekspansi kunci

2.5.2.2 Transformasi *SubBytes*

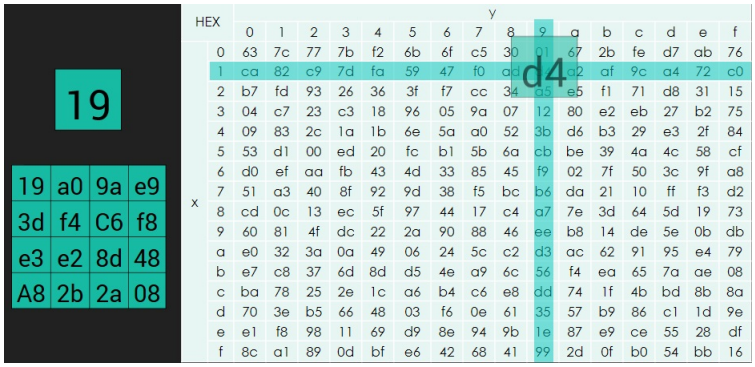
Transformasi *SubBytes()* memetakan tiap *byte* pada *array state* menggunakan tabel subtitusi *S-Box*. Berbeda dengan *DES* yang menggunakan tabel *S-Box* berbeda untuk setiap putaran (*round*), *AES* hanya menggunakan satu tabel *S-Box*.

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

x

Gambar 2.15: Tabel S-Box

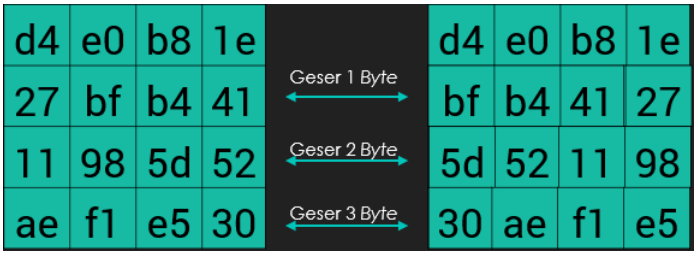
Cara pensubtitusiannya dengan melihat tiap byte array state yang ingin dipetakan, misalkan $S[r, c] = XY$ dimana XY merepresentasikan digit heksadesimal dari nilai $S[r, c]$. Hasil subtitusinya merupakan perpotongan antara digit X dengan digit Y pada tabel S-Box.



Gambar 2.16: Contoh hasil subtitusi

2.5.2.3 Transformasi *ShiftRows*

Transformasi ini melakukan pergeseran pada 3 baris terakhir pada *array state* dengan cara *wrapping* (siklik) ke kiri. Banyaknya pergeseran bergantung pada r, misal baris $r = 1$ pergeseran dilakukan sejauh 1 *byte*, $r = 2$ digeser sejauh 2 *byte*, dan $r = 3$ digeser sejauh 3 *byte*.



Gambar 2.17: Proses *ShiftRows()*

2.5.2.4 Transformasi *MixColumns*

Transformasi *MixColumns* mengalikan setiap kolom dalam *array state* dengan polinom $a(x)$ mod (x^4+1). Setiap kolom diperlakukan sebagai polinom 4-suku pada *Rijndael Galois Field* $GF(2^8)$.

Polinom $a(x)$ yang ditetapkan pada proses *MixColumns* adalah:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{2.2}$$

Transformasi *MixColumns* dinyatakan sebagai perkalian matriks:

$$s'(x) = a(x) \otimes s(x) \tag{2.3}$$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{0,c} \\ S'_{0,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{2.4}$$

$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \tag{2.5}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c} \tag{2.6}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c}) \tag{2.7}$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c}) \tag{2.8}$$

Dikarenakan proses perkalian $a_{r,c} \bullet S_{r,c}$ dalam *MixColumns* cukup kompleks (2.5 - 2.8), maka kita bisa menggunakann tabel *Galois Multiplication* (Lampiran A) untuk mendapatkan hasilnya. Caranya yaitu dengan melihat pengali dari $S_{r,c}$ yaitu $a_{r,c}$, kemudian petakan nilai heksadesimal $S_{r,c}$ dengan tabel *Galois Multiplication* $M = a_{r,c}$.

Sebagai contoh, dari hasil transformasi *ShiftRows* diperoleh *array state*:

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Gambar 2.18: Hasil transformasi *ShiftRows*

Operasi *MixColumns* pada kolom pertama *array state* adalah sebagai berikut:

$$S'_{0,c} = (\{02\} \bullet \texttt{0xD4}) \oplus (\{03\} \bullet \texttt{0xBF}) \oplus \texttt{0x5D} \oplus \texttt{0x30}$$

$$S'_{1,c} = \texttt{0xD4} \oplus (\{02\} \bullet \texttt{0xBF}) \oplus (\{03\} \bullet \texttt{0x5D}) \oplus \texttt{0x30}$$

$$S'_{2,c} = \texttt{0xD4} \oplus \texttt{0xBF} \oplus (\{02\} \bullet \texttt{0x5D}) \oplus (\{03\} \bullet \texttt{0x30})$$

$$S'_{3,c} = (\{03\} \bullet \texttt{0xD4}) \oplus \texttt{0xBF} \oplus \texttt{0x5D} \oplus (\{02\} \bullet \texttt{0x30})$$

Dengan menggunakan tabel *Galois Multiplication*, maka didapatkan:

Untuk $S'_{0,0}$:

$$\begin{aligned} S'_{0,0} &= (\{02\} \bullet \texttt{0xD4}) \oplus (\{03\} \bullet \texttt{0xBF}) \oplus \texttt{0x5D} \oplus \texttt{0x30} \\ &= \texttt{0xD4} \oplus \texttt{0xDA} \oplus \texttt{0x5D} \oplus \texttt{0x30} \\ &= \texttt{0x04} \end{aligned}$$

Untuk $S'_{1,0}$:

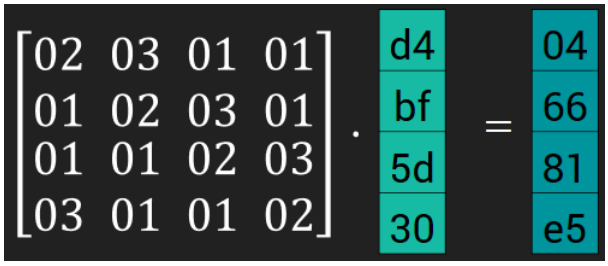
$$\begin{aligned} S'_{1,0} &= \texttt{0xD4} \oplus (\{02\} \bullet \texttt{0xBF}) \oplus (\{03\} \bullet \texttt{0x5D}) \oplus \texttt{0x30} \\ &= \texttt{0xD4} \oplus \texttt{0x65} \oplus \texttt{0xE7} \oplus \texttt{0x30} \\ &= \texttt{0x66} \end{aligned}$$

Untuk $S'_{2,0}$:

$$\begin{aligned} S'_{2,0} &= \texttt{0xD4} \oplus \texttt{0xBF} \oplus (\{02\} \bullet \texttt{0x5D}) \oplus (\{03\} \bullet \texttt{0x30}) \\ &= \texttt{0xD4} \oplus \texttt{0xBF} \oplus \texttt{0xBA} \oplus \texttt{0x50} \\ &= \texttt{0x81} \end{aligned}$$

Untuk $S'_{3,0}$:

$$\begin{aligned} S'_{3,0} &= (\{03\} \bullet \texttt{0xD4}) \oplus \texttt{0xBF} \oplus \texttt{0x5D} \oplus (\{02\} \bullet \texttt{0x30}) \\ &= \texttt{0x67} \oplus \texttt{0xBF} \oplus \texttt{0xBA} \oplus \texttt{0x60} \\ &= \texttt{0xE5} \end{aligned}$$



Gambar 2.19: Hasil perkalian Galois *MixColumns*

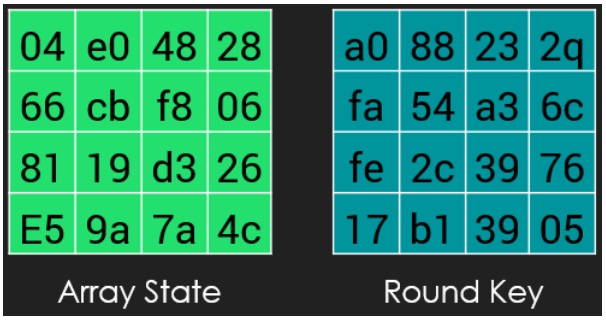
Operasi dilakukan pada kolom-kolom berikutnya pada *array state*. Sehingga *array state* yang baru adalah:

04	e0	48	28
66	cb	f8	06
81	19	d3	26
E5	9a	7a	4c

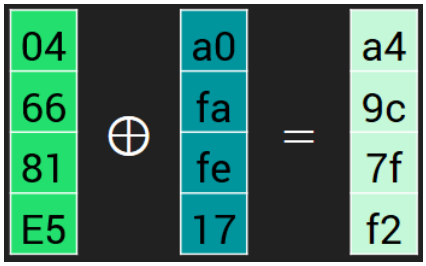
Gambar 2.20: Hasil transformasi *MixColumns*

2.5.2.5 Transformasi *AddRoundKey*

Transformasi ini melakukan operasi XOR antara *array state* dengan *round key* yang telah dibangkitakan pada proses ekspansi. Hasil dari transformasi ini disimpan kembali ke dalam *array state*.



Gambar 2.21: *Array State* dan *Round Key*



Gambar 2.22: Operasi XOR Transformasi *AddRoundKey*

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

Gambar 2.23: Hasil Transformasi *AddRoundKey*

2.5.3 Proses Dekripsi

Proses pendekripsian ini dilakukan menggunakan operasi invers atau membalikkan proses yang terjadi pada proses enkripsi yang telah dijelaskan sebelumnya. Sehingga, proses invers yang dilakukan adalah sebagai berikut:

1. *AddRoundKey*: yaitu dengan melakukan operasi XOR antara kondisi sebelumnya (*ciphertext*) dengan *round key* indeks terakhir yang telah dibangkitkan. Tahap ini disebut sebagai putaran inisiasi (initial round).

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

   (a) *invShiftRows*: pergeseran baris-baris array state dengan menggunakan metode wrapping secara invers.

(b) *invSubBytes*: mensubtitusi byte dengan menggunakan tabel invers substitusi (S-box).

(c) *AddRoundKey*: melakukan operasi XOR antara *array state* dengan *round key*.

(d) *invMixColumns*: mengacak data di masing-masing kolom *array state*.

3. *Final Round*: proses untuk putaran akhir hanya terdiri adari 3 operasi:

(a) *invShiftRows*

(b) *invSubBytes*

(c) *AddRoundKey*

Algoritma 1.2 Fungsi Dekripsi

```
KeyExpansion(key, rk);
AddRoundKey(state, rk[ROUNDS]);
for (r = 1; r <= ROUNDS - 1; r++) {
  invShiftRows(state);
  invSubBytes(state);
  AddRoundKey(state, rk[ROUNDS-i]);
  invMixColumns(state);
}
invShiftRows(state);
invSubBytes(state);
AddRoundKey(state, rk[0]);
}
```

Proses ekspansi kunci (*round key*) masih sama seperti pada proses enkripsi.

2.5.3.1 Transformasi *InvShiftRows*

Transformasi ini melakukan pergeseran pada 3 baris terakhir pada *array state*

dengan cara *wrapping* (siklik) ke kanan. Banyaknya pergeseran bergantung pada r, misal baris $r = 1$ pergeseran dilakukan sejauh 1 *byte*, $r = 2$ digeser sejauh 2 *byte*, dan $r = 3$ digeser sejauh 3 *byte*.

Gambar 2.24: Proses *InvShiftRows()*

2.5.3.2 Transformasi *InvSubBytes*

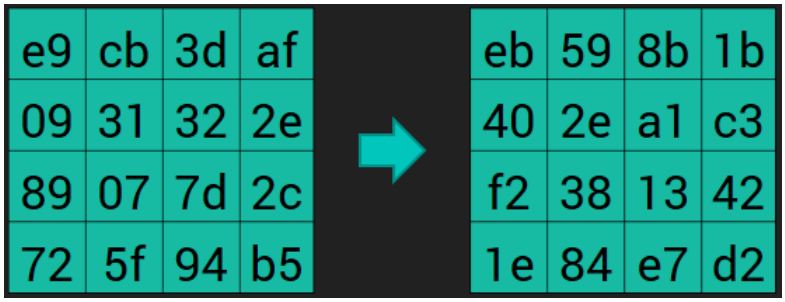
Transformasi *SubBytes()* memetakan tiap *byte* pada *array state* menggunakan tabel subtitusi. Karena merupakan operasi inverse, maka tabel subtitusi yang digunakan berbeda yaitu inverse dari tabel *S-Box*.

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

Gambar 2.25: Tabel Invers S-Box

Cara pensubtitusiannya tetap sama yaitu dengan melihat tiap byte array state

yang ingin dipetakan, misalkan $S[r, c] = XY$ dimana XY merepresentasikan digit heksadesimal dari nilai $S[r, c]$. Hasil subtitusinya merupakan perpotongan antara digit X dengan digit Y pada tabel invers *S-Box* [4].



Gambar 2.26: Hasil Subtitusi dengan tabel Inv S-Box

2.5.3.3 Transformasi *InvMixColumns*

Transformasi ini merupakan invers dari *MixColumns* pada proses enkripsi yang sudah dijelaskan sebelumnya, sehingga polinom dan perkalian matriksnya juga berbeda.

Polinom $a(x)$ yang ditetapkan pada proses *InvMixColumns* adalah:

$$a^{-1}(x) = \{11\}x^3 + \{13\}x^2 + \{09\}x + \{14\} \tag{2.9}$$

Dan transformasi *InvMixColumns* dinyatakan sebagai perkalian matriks:

$$s'(x) = a^{-1}(x) \otimes s(x) \tag{2.10}$$

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{0,c} \\ S'_{0,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{2.11}$$

$$S'_{0,c} = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c}) \quad (2.12)$$

$$S'_{1,c} = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c}) \quad (2.13)$$

$$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c}) \quad (2.14)$$

$$S'_{3,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c}) \quad (2.15)$$

Dikarenakan proses perkalian $a_{r,c} \bullet S_{r,c}$ dalam *InvMixColumns* cukup kompleks (2.12 - 2.15), maka kita bisa menggunakann tabel *Galois Multiplication* untuk mendapatkan hasilnya. Caranya yaitu dengan melihat pengali dari $S_{r,c}$ yaitu $a_{r,c}$, kemudian petakan nilai heksadesimal $S_{r,c}$ dengan tabel *Galois Multiplication* $M = a_{r,c}$.

Sebagai contoh, dari hasil transformasi *AddRoundKey* diperoleh *array state*:

47	40	a3	4c
37	d4	70	9f
94	e4	3a	42
ed	a5	a6	bc

Gambar 2.27: Hasil transformasi *AddRoundKey*

Operasi *MixColumns* pada kolom pertama *array state* adalah sebagai berikut:

$$S'_{0,c} = (\{0e\} \bullet \texttt{0x47}) \oplus (\{0b\} \bullet \texttt{0x37}) \oplus (\{0d\} \bullet \texttt{0x94}) \oplus (\{09\} \bullet \texttt{0xED})$$

$$S'_{1,c} = (\{09\} \bullet \texttt{0x47}) \oplus (\{0e\} \bullet \texttt{0x37}) \oplus (\{0b\} \bullet \texttt{0x94}) \oplus (\{0d\} \bullet \texttt{0xED})$$

$$S'_{2,c} = (\{0d\} \bullet \texttt{0x47}) \oplus (\{09\} \bullet \texttt{0x37}) \oplus (\{0e\} \bullet \texttt{0x94}) \oplus (\{0b\} \bullet \texttt{0xED})$$

$$S'_{3,c} = (\{0b\} \bullet \texttt{0x47}) \oplus (\{0d\} \bullet \texttt{0x37}) \oplus (\{09\} \bullet \texttt{0x94}) \oplus (\{0e\} \bullet \texttt{0xED})$$

Dengan menggunakan tabel *Galois Multiplication*, maka untuk kolom perta-

ma didapatkan:

$$s'(x) = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} 47 \\ 37 \\ 94 \\ ed \end{bmatrix} = \begin{bmatrix} 87 \\ 6e \\ 46 \\ a6 \end{bmatrix}$$

Operasi dilakukan pada kolom-kolom berikutnya pada *array state*. Sehingga *array state* yang baru adalah:

87	f2	4d	97
6e	4c	90	ec
46	e7	4a	c3
a6	8c	d8	95

Gambar 2.28: Hasil transformasi *InvMixColumns*