# BAB 2
# TINJAUAN PUSTAKA

## 2.1 VoIP (Voice Over Internet Protocol)

*Voice Over Internet Protocol* atau disingkat VoIP, dikenal juga dengan sebutan IP *Telephony*. VoIP didefinisikan sebagai suatu sistem yang menggunakan jaringan internet untuk mengirimkan data paket suara dari suatu tempat ke tempat yang lain menggunakan perantara protokol IP. Sehingga perbedaan VoIP dengan telepon tradisional adalah masalah infastrukturnya, jika VoIP menggunakan internet sedangkan telepon tradisional menggunakan infrastruktur telepon yang sudah dibangun lebih awal. Prinsip kerja VoIP secara sederhana adalah dengan mengubah sinyal *analog* yang didapatkan *microphone* menjadi sinyal *digital*, kemudian sinyal *digital* tersebut dikirimkan melalui jaringan *internet* menuju *server* VoIP kemudian diteruskan dari pihak pengirim ke pihak penerima, sehingga komunikasi suara terjadi antara kedua belah pihak [2].

## 2.2 Manfaat Voip

Manfaat berkomunikasi dengan VoIP dengan menggunakan jaringan data antara lain efesiensi alokasi *bandwidth*, kemampuan untuk menggunakan metode kompresi suara, menekan biaya penggunaan, kemampuan menggunakan *single interface*, meningkatkan keandalan *(reliability)* jaringan komputer [2].

### 2.2.1 Unsur Pembentuk Voip

Ada empat unsur pembentuk jaringan VoIP, yaitu: *User Agent, Proxy, Protocol,* dan CODEC *(Coder-Decoder).* Dari unsur-unsur tersebut jaringan VoIP terbentuk [2].

#### 2.2.1.1 *User Agent*

*User Agent* ada yang berupa *software*, ada pula yang berupa *hardware*. *User agent* seperti layaknya telepon yang kita kenal yang berfungsi untuk melakukan pemanggilan atau menerima telepon, baik berasal dari sambungan komputer dengan komputer, komputer dengan IP-phone, komputer dengan PSTN (perlu ditambah alat ATA).

#### 2.2.1.2 *Proxy*

Karena VoIP akan dijalankan di internet maka perlu dibentuk jembatan penghubung, jembatan tersebut biasa disebut *proxy*, seperti halnya *proxy-proxy server* pada umumnya, tetapi ini khusus untuk kebutuhan VoIP. Untuk versi *softswitch open source* dan cukup terkenal dan telah teruji keandalannya adalah Asterisk.

#### 2.2.1.3 *Protocol*

*Protocol* merupakan sebuah aturan atau *rule* yang harus dipenuhi agar akses komunikasi dalam hal ini komunikasi VoIP dapat melewati jaringan, dalam hal ini internet. Di dalam komunikasi VoIP mengenal tiga macam *protocol* tambahan selain protokol standar internet, yaitu TCP/IP, yaitu:

##### 2.2.1.3.1 H.323

H.323 merupakan protokol yang dikembangkan oleh ITU-T (International Telecomunications Union-Telecomunication). VoIP dapat berkomunikasi dengan sistem lain yang beroperasi pada jaringan packet-switch. Standar H.323 terdiri dari komponen, protokol, dan prosedur yang menyediakan komunikasi multimedia melalui jaringan packet-based. Bentuk jaringan packet-based yang dpat dilalui antara lain jaringan internet, Internet Packet Exchange (IPX)-based, Local Area Network (LAN), dan Wide Area Network (WAN). H.323 dapat digunakan untuk layanyan – layanan multimedia seperti komunikasi suara (IP telephony), komunikasi video dengan suara (video telephony), dan gabungan suara, video dan data.

##### 2.2.1.3.2 SIP (*Session Initiation Protocol*)

SIP merupakan protokol yang didesain untuk dapat melakukan pembangunan sesi antar dua titik (user agent) sehingga kedua titik dapat berbagi resource. SIP yang merupakan protokol pada layer aplikasi dapat digunakan bersama dengan proxy *server*. SIP dapat menangani registrasi user, undangan sesi, dan permintaan lainnya yang bertujuan untuk membangun, memodifikasi, atau menutup sesi. SIP bukanlah

media transfer protocol, tapi signalling protocol. Sehingga paket voice dan video tidak dibawa oleh protokol ini, SIP hanya melakukan signaling [5].

Jenis SIP *Message* terdri dari dua macam yaitu *Request* (Dikirim dari *client* ke *server* dan mendifinisikan operasi yang dibutuhkan oleh *client*) dan *Response* (Dikirim dari *server* ke *client* serta memberikan status dari *request* yang dikirim *client*). SIP *Request* tercantum dalam Tabel 2.1, yang paling dasar pada SIP *Request* adalah INVITE yang digunakan untuk mengatur atau memulai sesi. SIP *Response* menggunakan kode tiga digit, dan dibagi menjadi enam kategori: *Informational*, *Success, Redirection, Client Error, Server Error,* dan *Global* Error. SIP *Response* ditunjukkan pada Tabel 2.2 [6].

**Tabel 2.1 SIP *Request***

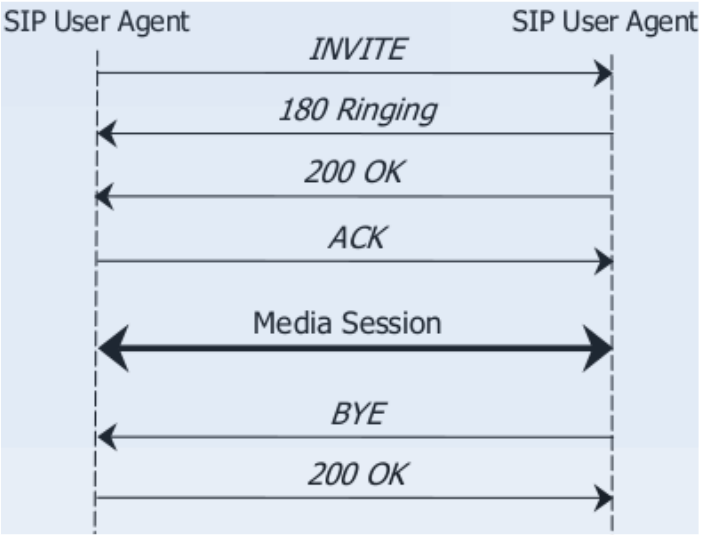
| SIP Request | Keterangan |
|---|---|
| REGISTER | Digunakan oleh user mengindikasikan alamat IP sekarang dan URL yang akan menerima panggilan. |
| INVITE | Digunakan untuk membuat sesi antar user agent. |
| ACK | Konfirmasi untuk menyetujui pesan INVITE. |
| CANCEL | Menghentikan request yang menunggu |
| BYE | Menghentikan antara dua user agent dalam saat conference. |
| OPTIONS | Meminta informasi tentang kapabilitas pemanggil tanpa membuat panggilan. |

**Tabel 2.2 SIP *Response***

| SIP Response | Contoh |
|---|---|
| Informational or Provisional (1xx) | 100 *Trying*<br>180 *Ringing*<br>183 *Session in Progress* |
| Success (2xx) | 200 *Ok*<br>202 *Accepted* |
| Redirection (3xx) | 300 *Moved*<br>302 *Multiple Choices*<br>205 *Use Proxy* |
| Client Error (4xx) | 401 *Unauthorized* |

| | |
|---|---|
| | 403 *Forbidden*<br>404 *Not Found*<br>415 *Unsupported Media Type*<br>486 *Busy Here*<br>428 *Use Identity Header* |
| *Server* Error (5xx) | 501 *Not Implemented*<br>503 *Service Unavailable* |
| Global Error (6xx) | 600 *Busy Everywhere*<br>603 *Decline* |

Berikut merupakan aliran panggilan atau *Callflow* pada komunikasi VoIP berlangsung pada sebuah jaringan :




**Gambar 2.1 *Callflow* komunikasi VoIP berlangsung**

Berdasarkan Gambar 2.1 Sebuah UA ingin membangun sebuah sesi menciptakan sebuah *Invite* dan mengirimkannya ke UA lain. misalnya sebuah 180 *Ringing* untuk menunjukkan bahwa sedang terjadi pesan *Invite*. Jika permintaan sesi berhasil, sebuah respon 200 OK dikirimkan. Untuk menyetujui UA mengirimkan respon 200 OK.

#### 2.2.1.3.3 IAX (*The Inter-Asterisk Exchange*)

Iax merupakan protokol dari asterisk. Sekarang ini, dengan dimotori di Indonesia melalui VoIP Rakyat dimasyarakatkan protokol IAX2, untuk *user agent, software idefisk* menggunakan IAX2.

#### 2.2.1.4 *Codec*

Pada prinsipnya pengodean suara merupakan pengalihan kode analog menjadi kode digital agar suara dapat dikirim dalam jaringan komputer. Pengodean ini dikenal dengan istilah codec (*compressor – decompressor*). Berbagai jenis dikembangkan untuk memampatkan / mengompresikan suara agar dapat menggunakan bandwidth secara hemat tanpa mengorbankan kualitas suara (didengar relatif baik).

### 2.2.2 RTP (Realtime Treansfer Protocol)

RTP adalah protokol standar diperkenalkan oleh IETF pada tahun 1996. Tujuan utamanya adalah untuk memberikan media yang real-time. Data *audio* atau *video* melalui jaringan IP seperti LAN atau WAN. IP telephony dan *video streaming* adalah contoh sistem menggunakan RTP. Namun RTP tidak bisa memberikan data dengan sendirinya. Untuk alasan ini, RTP bekerja sama dengan lapisan transport ptotocol dalam rangka untuk memberikan data yang *real-time*. Diberikan pembatasan, RTP tidak dapat dianggap sebagai protokol lapisan transport karena bekerja diatas protokol lapisan transport. Maka RTP terbaik dilihat sebagai protokol lapisan aplikasi. di sisi lain tidak memiliki mekanisme untuk memastikan real-time pengiriman paket atau pengiriman halus. RTP menyediakan informasi yang diperlukan seperti timestamp dan urutan nomor yang digunakan oleh mekanisme aplikasi *real-time* untuk memastikan pengiriman tepat waktu dan lancar [7].

### 2.2.3 SRTP (Secure Real Time Transport Protocol)

*Secure Real Time Transport Protocol* (SRTP) menyediakan fitur enkripsi pada profil RTP. Enkripsi disini dimaksudkan untuk menyediakan sistem keamanan data dengan otentikasi dan integritas pesan, dan perlindungan terhadap *playback* dengan data RTP dalam aplikasi baik *unicast* maupun *multicast*. SRTP dikembangkan oleh sebuah tim kecil dari Cisco dan Ericsson yang merupakan ahli protokol IP dan kriptografi. SRTP menggunakan algoritma *AES (Advanced Encryption Standart)* sebagai metode enkripsi dalam pengiriman data. Pada aplikasi, SRTP memiliki 2 buah mode, yaitu *Segmented Integer Counter*, dan AES di f8-mode. Selain itu SRTP juga dapat berjalan pada mode *null cipher*, Mode ini adalah mode dimana pengiriman data tidak dilindungi dengan algoritma enkripsi. SRTP pada kenyataannya hanya mengenkripsi *payload* (audio dan video) untuk kerahasiaan [5].

## 2.3 Kriptografi

Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan [8]. Kriptografi juga berarti ilmu yang mempelajari teknik – teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, keabsahan, integritas, serta otentikasi data [9]. Pada kriptografi pesan atau informasi yang dapat dibaca disebut *plauntext* atau *cleartext.* Teknik untuk membuat pesan menjadi tidak dapat dibaca disebut sebagai enkripsi. Pesan yang tidak dapat dibaca disebut *chipertext*, proses yang merupakan kebalikan dari enkripsi disebur sebagai dekripsi. Jadi dekripsi akan membuat *chipertext* menjadi *plaintext* [10]. Kriptografi memiliki empat aspek keamanan yang terdiri dari [8]:

1. **Kerahasiaan *(Confidentiality)***

   Layanan untuk menjaga data agar tidak dapat dibaca oleh pihak yang tidak berhak membacanya. Pada umumnya kerahasiaan direalisasikan menggunakan penyandian.

2. **Integritas data *(Data Integrity)***

Layanan yang menjamin bahwa data masih asli/utuh atau belum pernah dimanipulsi selama pengiriman, untk menjaga integritas data sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan seperti penyisipan, penghapusan, dan pensubtitusian data kedalam pesan yang sebenarnya. Integritas data pada umumnya direalisasikan dengan tanda tangan digital *(digital signature).*

3. **Otentikasi *(Authentication)***

   Layanan untuk mengidentifikasi pihak – pihak yang berkomunikasi *(user authentication* atau *entity authentication)* maupun kebenaran sumber data *(data origin authentication).* Dua pihak yang saling berkomunikasi harus dapat saling mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran komunikasi juga harus diotentikasi asalnya. Otentikasi pada umumnya direalisasikan dengan tanda tangan digital *(digital signature).*

4. **Nirpenyangkalan *(Non-repudiatin)***

   Layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal menerima pesan.

### 2.3.1 Macam – Macam Algoritma Kriptografi

Berdasarkan kuncinya algoritma kriptografi dibagi menjadi dua [8], yaitu:

1. **Algoritma kunci simetris *(symmetric-key)***

   Pada algoritma kunci simetris, kunci yang digunakan untuk mengenkripsi dan mendekripsi sama, atau setidaknya terdapat perbedaan namun berhubungan sehingga dapat dikomputasi dengan mudah.

2. **Algoritma kunci asimetris *(asymmetric-key)***

   Pada algoritma kunci asimetris, kunci terbagi menjadi dua jenis, yaitu kunci privat dan kunci publik. Kunci privat merupakan kunci yang disimpan, sementara kunci publik adalah kunci yang didistribusikan. Apabila suatu teks

dienkripsi menggunakan suatu kunci publik, maka teks sandi yang dihasilkannya hanya dapat didekripsi dengan menggunakan kunci privat yang merupakan pasangan kunci publik tersebut. Begitu juga sebaliknya, jika kunci privat digunakan untuk proses enkripsi maka proses dekripsi harus menggunakan kunci publik pasangannya.

## 2.4 Algoritma Enkripsi AES

Rijndael termasuk dalam jenis algoritma kriptografi yang sifatnya simetri dan *cipher block*. Dengan demikian algoritma ini mempergunakan kunci yang sama saat enkripsi dan dekripsi serta masukan dan keluarannya berupa blok dengan jumlah bit tertentu.

Rijndael mendukung berbagai variasi ukuran blok dan kunci yang akan digunakan. Namun Rijndael mempunyai ukuran blok dan kunci yang tetap sebesar 128, 192, 256 bit. Pemilihan ukuran blok data dan kunci akan menentukan jumlah proses yang harus dilalui untuk proses enkripsi dan dekripsi. Berikut adalah perbandingan jumlah proses yang harus dilalui untuk masing – masing masukan.

**Tabel 2.3 Jumlah proses berdasarkan bit blok dan kunci**

| Panjang Kunci (Nk) Dalam *words* | Ukuran Blok Data (Nb) Dalam *words* | Jumlah Proses (Nr) |
|---|---|---|
| 4 | 4 | 10 |
| 6 | 4 | 12 |
| 8 | 4 | 14 |

Blok – blok data masukan dan kunci dioperasikan dalam bentuk *array*. Setiap anggota *array* sebelum menghasilkan keluaran *ciphertext* dinamakan dengan *state*. Setiap *state* akan mengalami proses yang secara garis besar terdiri dari empat tahap yaitu, *AddRoundKey*, *SubBytes, ShiftRows,* dan *MixColumns.* Kecuali tahap *MixColumns,* ketiga tahap lainnya akan diulang pada setiap proses sedangkan tahap

*MixColumns* tidak akan dilakukan pada tahap terakhir. Proses dekripsi adalah kebalikkan dari enkripsi.

Karena terjadi beberapa tahap dalam proses enkripsi, maka diperlukan *subkey – subkey* yang akan dipakai pada tiap tahap. Pengembangan jumlah kunci yang akan dipakai diperlukan karena kebutuhan *subkey – subkey* yang akan dipakai dapat mencapai ribuan bit, sedangkan kunci yang disediakan secara *default* hanya 128-256 bit. Jumlah total kunci yang diperlukan sebagai *subkey* adalah sebanyak Nb(Nr+1), dimana Nb adalah besarnya blok data dalam satuan *word*. Sedangkan Nr adalah jumlah tahapan yang harus dilalui dalam satuan *word*. Sebagai contoh, bilamana digunakan 128 bit (4 *word*) blok data dan 128 bit (4 *word*) kunci maka akan dilakukan 10 kali proses (lihat Tabel 2.1). Dengan demikian dari rumus didapatkan 4(10+1)=44 word=1408 bit kunci. Untuk melakukan pengembangan jumlah kunci yang akan dipakai dari kunci utama maka dilakukan *key schedule* [11].

### 2.4.1 Rijndael Key Schedule

AES menggunakan *key schedule* (Penjadwalan kunci) untuk menghasilkan sebuah kunci baru yang digunakan pada setiap *round* agar kunci pada setiap *round* memiliki nilai yang berbeda. Penjadwalan kunci dihasilkan dari kunci master dengan melalui tahapan – tahapan sebagai berikut [11] :

1. ***Rotate***
   *Rotate* digunakan untuk memindahkan awal dari kunci menjadi akhir dari kunci. Misalkan kunci master memiliki 32 bit kata yang di bagi masing – masing menjadi 8 bit (Hexadecimal) yaitu 1D 2C 3A 4F yang kemudian bit awal 1D tersebut akan dipindahkan menjadi bit yang terakhir menjadi 2C 3A 4F 1D.
2. **RCON**
   RCON merupakan sebuah metode yang digunakan untuk menghasilkan kunci yang baru dengan perhitungan polinomial dengan rumus : rcon (i) = $x^{i-1}$ mod $x^8 + x^4 + x^3 + x^1$ RCON memiliki tabel yang sudah ditetapkan untuk

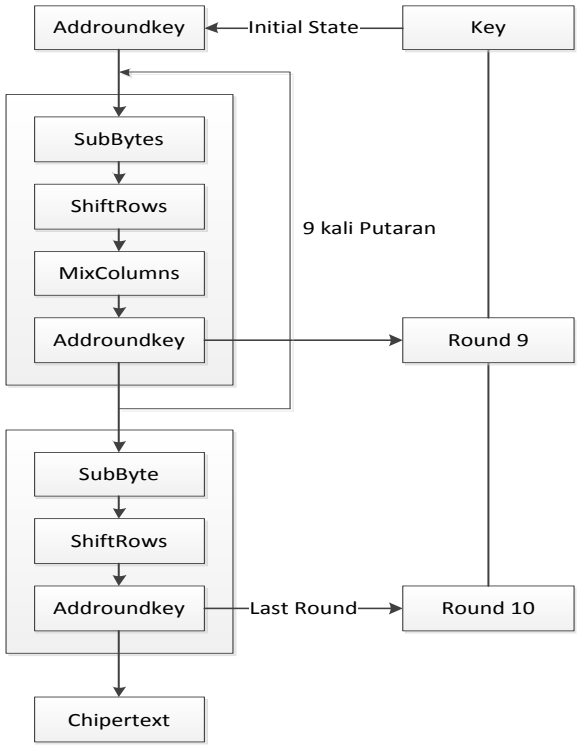
melakukan perhitungan polinomial antara kunci dengan tabel berikut sehingga menghasilkan sebuah kunci yang baru. Berikut tabel dari RCON :

**Tabel 2.4 *RCon***

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **RCon** | 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 |

### 2.4.2 Proses Enkripsi AES

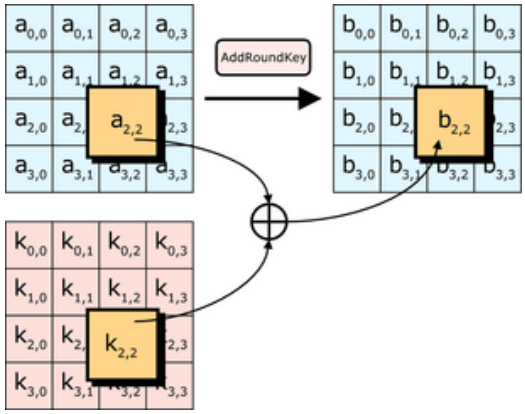
Proses enkripsi pada Aes adalah sebagai berikut:




**Gambar 2.2 Proses Enkripsi**

1. ***AddRoundKey***

Pada proses ini subkey digabungkan dengan state. Proses penggabungan ini menggunakan operasi XOR untuk setiap byte dari subkey dengan byte yang bersangkutan dari state. Untuk setiap tahap, subkey dibangkitkan dari kunci utama dengan menggunakan proses key schedule. Setiap subkey berukuran sama dengan state yang bersangkutan. Proses AddRoundKey diperlihatkan pada Gambar 2.3.



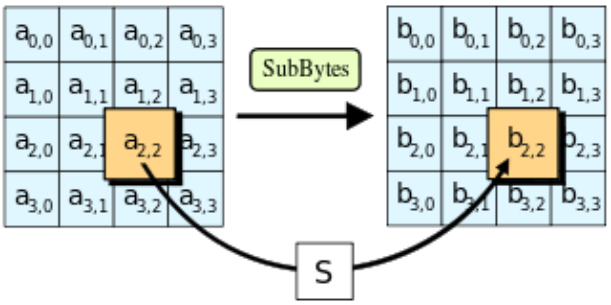

**Gambar 2.3 Proses A*ddRoundKey***

2. ***SubBytes***

Proses SubBytes adalah operasi yang akan melakukan substitusi tidak linear dengan cara mengganti setiap byte state dengan byte pada sebuah tabel yang dinamakan tabel S-Box. Sebuah tabel S-Box terdiri dari 16x16 baris dan kolom dengan masing-masing berukuran 1 byte. Tabel S-Box diperlihatkan pada Gambar 2.3 sedangkan proses SubBytes diperlihatkan pada Gambar 2.4.

| | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xa | xb | xc | xd | xe | xf |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 1x | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 2x | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 3x | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 4x | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 5x | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 6x | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 7x | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 8x | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 9x | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| ax | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| bx | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| cx | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| dx | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| ex | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| fx | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

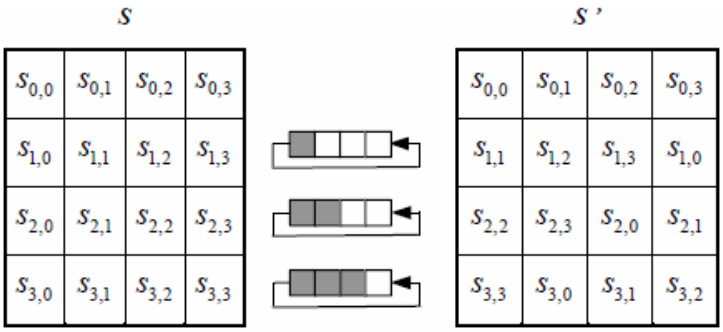
**Gambar 2.4 Tabel *S-Box***




**Gambar 2.5 Proses *subbyte***

### 3. *ShiftRows*

Proses *ShiftRows* akan beroperasi pada tiap baris dari tabel *state*. Proses ini akan bekerja dengan cara memutar *byte-byte* pada 3 baris terakhir (baris 1, 2, dan 3) dengan jumlah perputaran yang berbeda-beda. Baris 1 akan diputar sebanyak 1 kali, baris 2 akan diputar sebanyak 2 kali, dan baris 3 akan diputar sebanyak 3 kali. Sedangkan baris 0 tidak akan diputar. Proses *ShiftRows* diperlihatkan pada Gambar 2.5.




**Gambar 2.6 Proses *ShiftRows***

**4. *MixColumns***

Proses *MixColumns* akan beroperasi pada tiap kolom dari tabel *state*. Operasi ini menggabungkan 4 *bytes* dari setiap kolom tabel *state* dan menggunakan transformasi linier. Operasi *MixColumns* memperlakukan setiap kolom sebagai polinomial 4 suku dalam Galois field dan kemudian dikalikan dengan c(x) modulo (x4+1), dimana c(x)=3x3+x2+x+2. Operasi *MixColumns* juga dapat dipandang sebagai perkalian matrix. Langkah *MixColumns* dapat ditunjukkan dengan mengalikan 4 bilangan di dalam Galois field oleh matrix berikut ini.

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Atau bila dijabarkan:

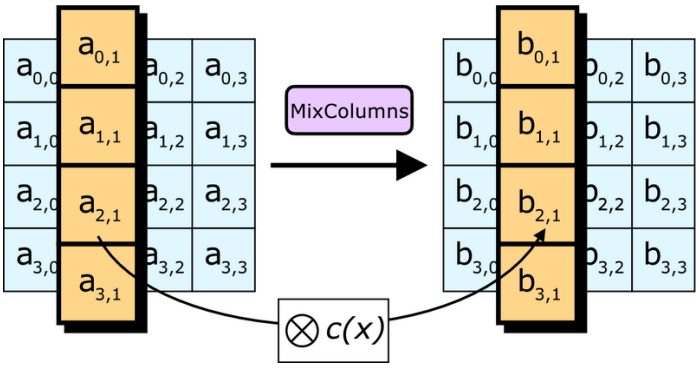
Ro = 2a0+3a1+1a2+1a3

R1 = 1a0+2a1+3a2+1a3

R3 = 1a0+1a1+2a2+3a3

R4 = 3a0+1a1+1a2+2a3

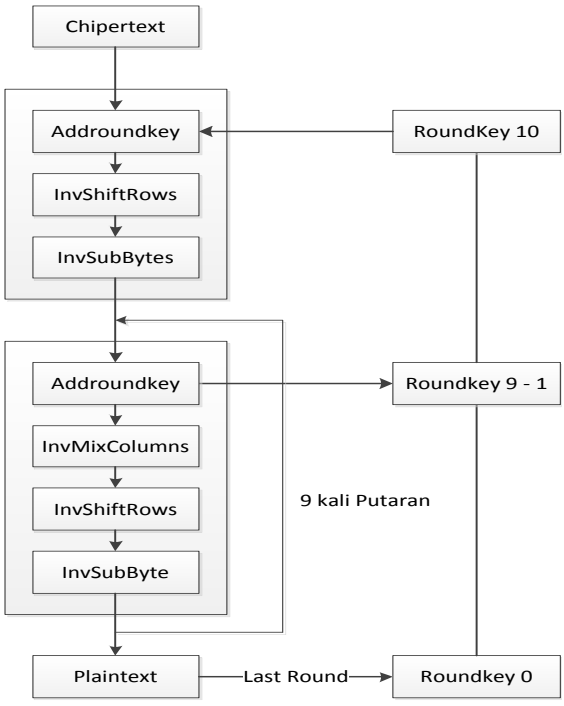
Operasi penjumlahan di atas dilakukan dengan operasi XOR, sedangkan operasi perkalian dilakukan dalam Galois field.




**Gambar 2.7 Proses *MixColumn***

### 2.4.3 Proses Dekripsi AES

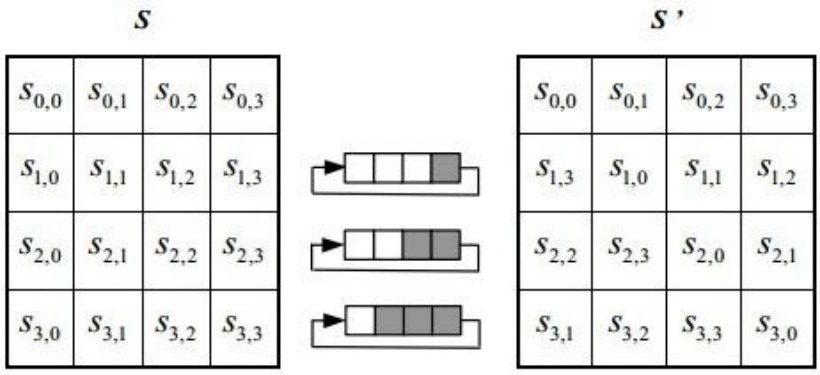
Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi *byte* yang digunakan pada *invers cipher* adalah *InvShiftRows, InvSubBytes, InvMixColumns*, dan *AddRoundKey*.




**Gambar 2.8 Proses Dekripsi**

**1. *Inverse ShiftRows***

*InvShiftRows* adalah transformasi *byte* yang berkebalikan dengan transformasi *ShiftRows*. Pada transformasi *InvShiftRows* dilakukan pergeseran *bit* ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran *bit* ke kiri. Ilustrasi transformasi *InvShiftRows* dapat pada Gambar 2.8



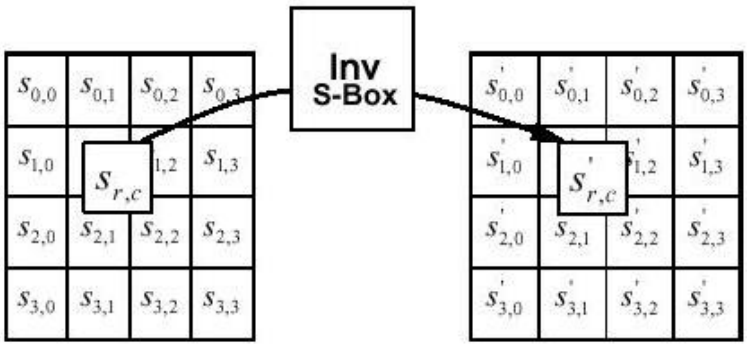

**Gambar 2.9 Proses *InvShiftRows***

**2. *Inverse SubBytes***

*InvSubBytes* juga merupakan transformasi *bytes* yang berkebalikan dengan transformasi *SubBytes*. Pada *InvSubBytes* tiap elemen pada *state* dipetakan dengan menggunakan table *Inverse S-Box*. Tabel *Inverse S-Box* akan ditunjukkan dalam gambar 2.9 berikut:

| | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xa | xb | xc | xd | xe | xf |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | 52 | 09 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| 1x | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| 2x | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| 3x | 08 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| 4x | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| 5x | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| 6x | 90 | d8 | ab | 00 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 05 | b8 | b3 | 45 | 06 |
| 7x | d0 | 2c | 1e | 8f | ca | 3f | 0f | 02 | c1 | af | bd | 03 | 01 | 13 | 8a | 6b |
| 8x | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| 9x | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| ax | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1b |
| bx | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| cx | 1f | dd | a8 | 33 | 88 | 07 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| dx | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |
| ex | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| fx | 17 | 2b | 04 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

**Gambar 2.10 Tabel *InvS-Box***




**Gambar 2.11 Proses *InvSubByte***

### 3. *Inverse MixColumns*

Setiap kolom dalam *state* dikalikan dengan matriks perkalian dalam AES. Operasi *InvMixColumns* memperlakukan setiap kolom sebagai polinomial 4 suku dalam Galois field dan kemudian dikalikan dengan c(x) modulo (x4+1) c(x)=11x3+13x2+9x+14. Operasi *InvMixColumns* juga dapat dipandang sebagai perkalian matrix. Langkah *InvMixColumns* dapat ditunjukkan dengan mengalikan 4 bilangan di dalam Galois field oleh matrix berikut ini:

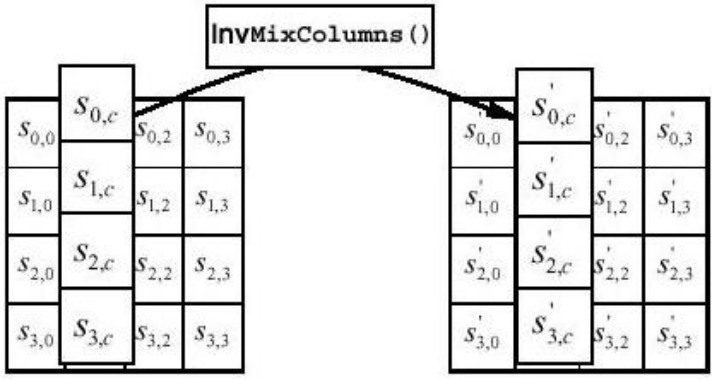
$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 14 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Atau

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Dimana r0, r1, r2 dan r3 adalah hasil setelah transformasi. a0 - a3 dapat diperoleh dari matriks setelah data mengalami proses substitusi dalam *InvSbox*. Untuk memperoleh nilai r0, rumusnya seperti dibawah ini:

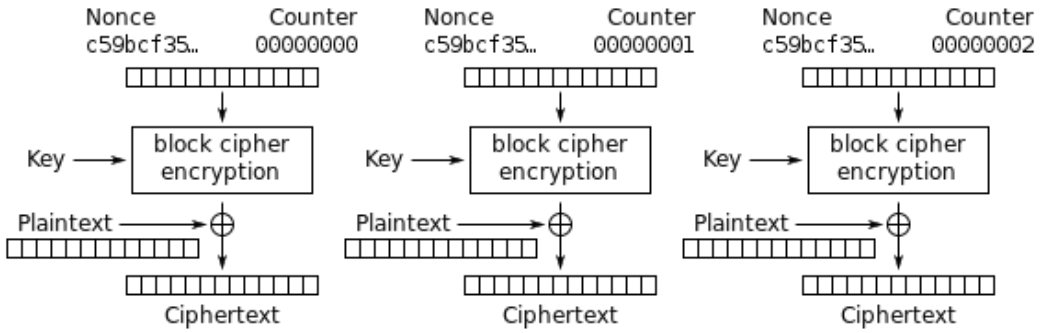
$$r0 = \{0e.a0\}xor\{0b.a1\}xor\{0d.a2\}xor\{09.a3\}$$
$$r1 = \{09.a0\}xor\{0e.a1\}xor\{0b.a2\}xor\{0d.a3\}$$
$$r2 = \{0d.a0\}xor\{09.a1\}xor\{0e.a2\}xor\{0b.a3\}$$
$$r3 = \{0b.a0\}xor\{0d.a1\}xor\{09.a2\}xor\{0e.a3\}$$



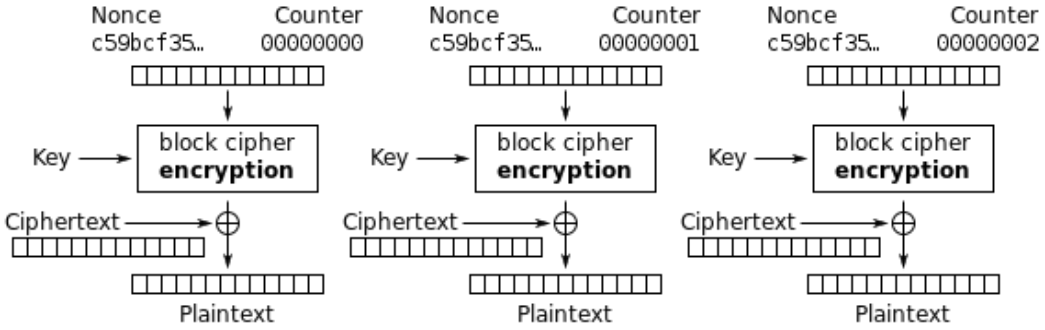

**Gambar 2.12 Proses *InvMixColumns***

### 2.4.4 Counter Mode

*Counter Mode* merupakan salah satu metode pada AES yang berfungsi untuk mengubah *Block Cipher* menjadi *Stream Cipher*. Pada *Counter Mode* terdapat sebuah nilai yang memiliki angka unik yang digunakan untuk menyamarkan *key*. Nilai tersebut di namakan *Nonce* yang nantinya akan di masukkan ke dalam sebuah *Counter*. Pada *Counter Mode*, kunci yang digunakan haruslah berbeda pada setiap *counter* dimana kunci tersebut di bangkitkan secara acak untuk proses enkripsi dan kunci yang sama pada saat enkripsi di gunakan kembali untuk proses dekripsinya.




**Gambar 2.13 Alur Proses Enkripsi *Counter Mode***




**Gambar 2.14 Alur Proses Dekripsi *Counter Mode***

Pada gambar diatas menjelaskan bahwa nilai *Nonce* bisa di pakai berulang-ulang pada setiap *Counter* tetapi kunci yang digunakan untuk enkripsi haruslah berbeda. Kunci *Master* akan di lakukan proses *schedule* untuk menghasilkan kunci

yang berbeda pada setiap *Counter*. Pada *Counter* Pertama Nilai *Nonce* dan *Counter* harus memiliki panjang sebanyak 16 bit dan dibagi menjadi masing – masing 8bit untuk setiap nilai *Nonce* dan *Counter* nya.

Setelah itu kita melakukan proses *Key Schedule* untuk memastikan kunci yang digunakan oleh setiap *Counter* berbeda. Setelah semua siap *Counter* pertama di enkripsi dengan menggunakan *key master* dengan menggunakan metode AES dan menghasilkan sebuah *Key Stream*. *Key Stream* tersebut kemudian akan di Xor kan dengan *Plaintext* untuk mendapatkan *Ciphertext* dan proses enkripsi akan berlanjut sampai pesan yang dikirimkan berakhir.

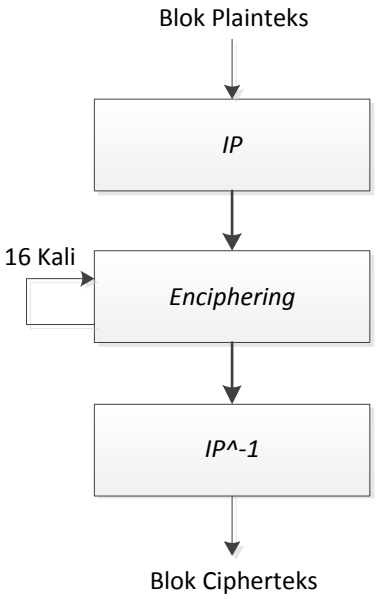
Pada dekripsi *Counter Mode* melalui proses yang sama dengan enkripsi yang sama dengan enkripsinya yaitu memasukkan *Nonce* dan nilai *Counter* yang sama dan nantinya akan di enkripsi menggunakan *master key* yang sama dengan proses enkripsi untuk menghasilkan sebuah *Key Stream* dan *Key Stream* yang dihasilkan tersebut akan di Xor kan dengan *Ciphertext* nya sehingga menghasilkan *plaintext*.

## 2.5 Algoritma DES (*Data Encryption Standart*)

DES (*Data Encryption Standart*) adlah algoritma *cipher* blok yang populer karena dijadikan standar algoritma enkripsi kunci-simetri, meskipun saat ini standar tesebut telah digantikan dengan algoritma yang baru yaitu AES, karena DES sudah dianggap tidak aman lagi. Algoritma DES dikembangkan di IBM dibawah pimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma Lucifer yang dibuat oleh Horst Feisel. Algoritma ini telah disetujui oleh *National Bureau of Standart* (NBS) setelah penilaian kekuatannya oleh *National Security Agency* (NSA) Amerika Serikat.

DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis *cipher* blok. DES beroperasi pada ukuran blok 6 bit. DES mengenkripsi 64 bit *plaintext* menjadi 64 bit *ciphertext* dengan menggunakan 56 bit kunci internal (*internal key*)

atau upa-kunci (*subkey*). Kunci internal dibangkitakan dari kunci eksternal (*ekternal key*) yang panjangnya 64 bit [8].




**Gambar 2.15 Skema Global Algoritma DES**

Skema global dari algoritma DES adalah sebagai berikut (lihat gambar 2.15):

1. Blok plainteks dipermutasi dengan matriks permutasi awal (*Initial Permutation* atau *IP*)
2. Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali atau putaran. Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^-1) menjadi blok cipherteks.

## 2.6 Algoritma *Triple* DES

*Triple* DES atau TDES menggunakan DES tiga kali. Penggunaantiga langgkah ini penting untuk mencegah *meet-in-the-midle attack* pada *Double DES.* Bentuk sederhana *Triple* DES adalah:

Enkripsi $C = Ek3(Ek2(Ek1(P)))$

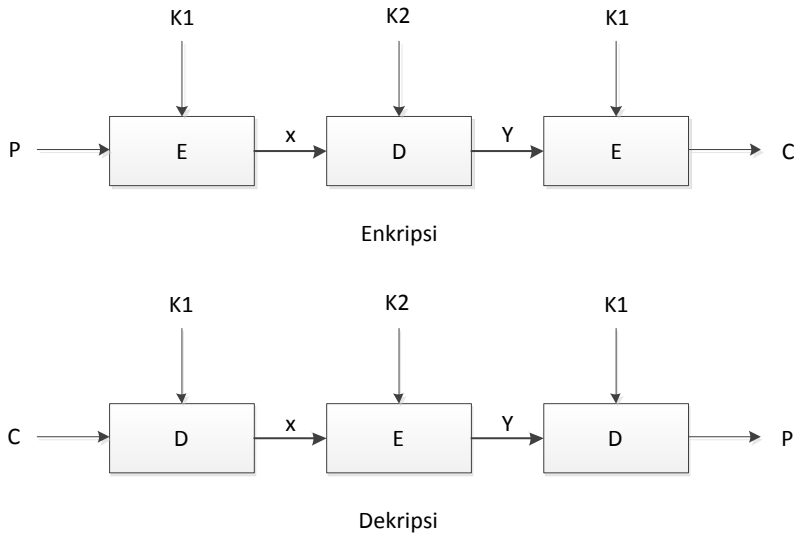
Dekripsi $P = Dk1(Dk2(Dk3(C)))$

Varian ini umum dikenal sebagai mode EEE (untuk enkripsi) karena pada proses enkripsi semuanya menggunakan enkripsi. Untuk menyederhanakan *interoperability* antara DES dan TDES, maka langkah di tengah (pada proses enkripsi TDES) diganti dengan dekripsi (mode EDE). Dengan pengubahan ini, maka dibuat beberapa versi TDES. Versi pertama TDES menggunakan 2 buah kunci, *K1* dan *K2:*

Enkripsi $C = Ek1(Dk2(Ek1(P)))$

Dekripsi $P = Dk1(Ek2(Dk1(C)))$

Enkripsi DES tunggal dengan kunci k dapat dinyatakan sebagai TDES-EDE dengan $k1 = k2 = k$. Gambar memperlihatkan versi TDES yang menggunakan 2 buah kunci. Penggunaan enkripsi pada langkah ditengah tidak mempengaruhi keamanan algoritma.

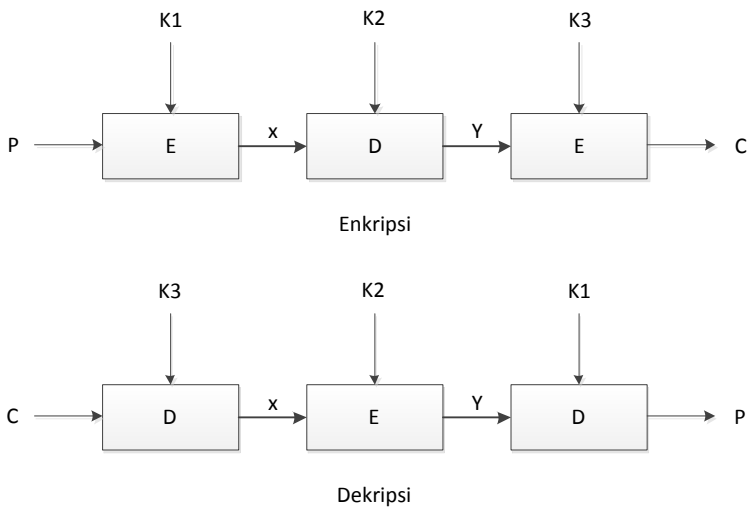



**Gambar 2.16 Diagram Enkripsi dan Dekripsi TDES Dengan 2 Buah Kunci**

Sedangkan versi TDES kedua menggunakan tiga buah kunci yaitu *k1, k2* dan *k3:*

Enkripsi $C = Ek3(Dk2(Ek1(P)))$

Dekripsi $P = Dk1(Ek2(Dk1(C)))$




**Gambar 2.17 Diagram Enkripsi Dan Dekripsi TDES Dengan 3 Buak Kunci**

Secara umum, *Triple* DES dengan dua buah kunci mempunyai panjang kunci 2 x 56 = 112 bit. Jauh lebih pendek daripada *Triple* DES dengan tiga buah kunci yang mempunyai panjang kunci 168 bit [8].

## 2.7 Diffie-Hellman *key exchange*

Diffie-Hellman (DH) dianggap merupakan algoritma kunci publik yang pertama kali ditemukan pada tahun 1976, meskipun NSA telah menemukan algoritma asimetrik jauh – jauh hari sebelumnya, algoritma ini memperoleh keamanannya dari sulitnya menghitung logaritma diskrit dari bilangan yang sangat besar. Algoritma DH hanya dapat digunakan untuk pertukaran kunci (simetris) dan tidak dapat digunakan untuk enkripsi / dekripsi maupun tanda tangan digital.

Matematika DH cukup sederhana. Pertama – tama A dan B menetapkan dua bilangan prima yang sangat besar (misalnya berorde $2^{2048}$) q dan a untuk digunakan bersama, sedemikian sehingga a < q dan a adalah akar primitif dari q. bila a adalah akar primitif dari bilangan p, maka nilai – nilai :

a mod p, $a^2$ mod p,…, $a^{p-1}$ mod p

akan berbeda semua, dan terdiri dari bilangan integer dari 1 sampai p-1 yang sangat tidak mungkin berurutan.

Sebagai contoh ambil p = 19, maka bila kita mengambil nilai a = 2 maka akan diperoleh $a^i$ mod p berturut – turut: 2,4,8,16,13,7,14,9,18,17,15,11,3,6,12,5,10,1 bila i bernilai dari 1 hingga 18. Jadi $2^2$ mod 19 = 4, $2^4$ mod 19 = 16 dan seterusnya. Perhatikan bahwa tidak ada nilai yang sama, ini berarti 2 adalah akar primitif dari 19.

Bila kita ambil p = 19 dan a = 7, maka akan kita dapat $7^1$ mod 19 = 7, $7^2$ mod 19 = 11, $7^3$ mod 19 = 1, $7^4$ mod 19 = 7. Jadi terdapat kesamaan nilai antara $7^1$ mod 19 dengan $7^4$ mod 19, sehingga 7 bukan akar primitif dari 19. kemudian proses berlanjut:

1. A membangkitkan bilangan acak x < q. nilai x ini merupakan kunci privat bagi A. tak ada orang lain yang boleh mengetahunya. Kemudia A menghitung x = $a^x$ mod q dan mengirim nilai x ke B, x merupakan kunci publik bagi A.
2. B membangkitkan bilangan acak y < q. nilai y ini merupakan kunci privat bagi B. tak ada orang lain yang boleh mengetahunya. Kemudia B menghitung y = $a^y$ mod q dan mengirim nilai y ke A, y merupakan kunci publik bagi B.
3. A membangkitkan kunci simetri yang akan digunakan sebagai kunci IDEA (pada PGP)

   K = $y^x$ mod q

   Perhatikan bahwa A menerima kunci publik B, y dan memiliki kunci privat x.
4. B juga membangkitkan kunci simetri yang akan digunakan sebagai kunci IDEA.

   K = $x^y$ mod q

   Kedua perhitungan pada langkah 3 dan 4 akan menghasilkan nilai yang sama:

   K = $y^x$ mod q (langkah 3)

   = $(a^y mod\ q)^x$ mod q = $(a^y)^x$ mod q

   = $a^{yx}$ mod q = $(a^x)^y$ mod q

   = $(a^x mod\ q)^y$ mod q

= $x^y$ mod q (langkah 4)

Lawan dapat menemukan a, q, x dan y dengan mudah karena memang tidak diamankan, namun untuk mendapatkan k yang akan dijadikan kunci penyandi data, diperlukan pemecahan x dari persamaan x = $a^x$ mod q. masalah ini disebut logaritma diskrit yang dipercaya sangata sulit dilakukan bila nilai x,y,a, dan q sangat besar misalnya berorde $2^{2048}$ [10].

## 2.8 Tools

Berikut adalah perangkat lunak pendukung dalam penunjang pembangunan aplikasi yang akan di bangun :

### 2.8.1 Android

Dibawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* perangkat Android adalah sebuah sistem operasi perangkat *mobile* berbasis linux yang mencangkup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel atau *smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode – kode Android seluler. Android mempunyai beberapa versi, adapun versi android yang pernah dirilis adalah sebagai berikut [12]:

1. **Android versi 1.1**

   Pada 9 maret 2009, Google merilis android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, ja, *alarm*, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. **Android versi 1.5 (*Cupcake*)**

   Pada pertengahan mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*). Terdapat banyak pembaharuan yakni mampu merekan dan menonton video dengan modus kamera, mengupload video ke youtube dan gambar ke picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung otomatis ke *headset Bluetooth*.

3. **Android versi 1.6 (Donut)**

   Dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan *control applet* VPN. Fitur lainnya adalah *camera, camcorder, VPN, Gestures, CDMA/EVDO, Text-to-Speech engine, resolusi VWGA*.

4. **Android versi 2.0/2.1 (Eclair)**

   Pada 3 Desember 2009 kembali diluncurkan ponsel android dengan mengoptimalkan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan kamera flash, dan *Bluetooth* 2.1.

5. **Android versi 2.2 (Froyo : Frozen Yoghurt)**

   Pada bulan mei 2010 Android versi 2.2 diluncurkan. Android yg lebih *complex* dengan penambahan *fitur – fitur* terbaru seperti *Global Positioning System* (GPS), kompas, kamera, dan Accelerometer.

6. **Android versi 2.3 (*Gingerbread*)**

   Diluncurkan Desember 2010. Hal-hal yang direvisi dari versi sebelumnya adalah kemampuan seperti berikut :

   1. SIP-*based* VoIP

2. *Near Field Communication* (NCF)
3. *Gyroscope* dan sensor
4. *Multiple cameras support*
5. *Mixable audio effects*
6. *Download manager*

**7. Android versi 3.0 (*Honeycomb*)**

Dirilis februari 2011 sebagai android 3.0 yang terdapat fitur baru setelah versi sebelumnya.

**8. Android versi 4.0 (*Ice Cream Sandwich*)**

Dirilis pada 19 Oktober 2011, petinggi Goolge, Gabe Cohen menyatakan bahwa android 4.0 secara teoritis kompatibel dengan perangkat android 2.3x yang diproduksi pada saat itu.

**9. Android versi 4.1 (*Jelly Bean*)**

Google mengumumkan Android 4.1 dalam konferensi Google I/O pada tanggal 27 juni 2012, *jelly bean* adalah pembaharuan penting yang bertujuan meningkatkan fungsi dan kinerja antarmuka pengguna (UI).

**10. Android versi 4.4 (Kitkat)**

Google mengumumkan Android 4.4 Kitkat dinamai dengan izin dari Nestle dan Herskey pada 3 september 2013 dengan penambahan fitur UI yang lebih menarik.

**11. Android versi 5.0 (Lollipop)**

Versi terbaru saat ini dengan tampilan UI yang lebih menarik, dengan meningkatkan daya hidup baterai 30% lebih tahan lama.

### 2.8.2 Java

Java merupakan bahasa pemrograman berorientasi objek dan bebas *platform*, dikembangkan oleh SUN Micro Sistem dengan sejumlah keunggulan yang memungkinkan java dijadikan sebagai bahasa pengembangan enterprise. Java lahir karena ketidakpuasan seseorang insinyur SUN Micro Sistem bernama james Gosling. Ia tidak puas dengan kompiler C++ yang ia gunakan untuk membuat software yang di-*embed* pada peralatan elektronik) karena dinilai terlalu banyak menghasilkan bug, berbiaya besar, sangat bergantung terhadap *platform*. Gosling merasa perlu membuat kompiler baru sebagai solusi terhadap sejumlah kelemahan pada C++ tersebut.

Kompiler baru tersebut diberi nama dengan Oak. Kompiler ini mirip dengan C++ tetapi dengan sejumlah pengurangan fitur yang dianggap kurang menguntungkan dala pengembangan, seperti *multiple inheritance*, konversi tipe secara otomatis, penggunaan *pointer managemen* memori.

Pada tahun 1994, Oak dubah namanya menjadi java. Pada era ini, java divisikan sebagai bahasa yang memiliki dukungan baik terhadap web. Berikut adalah keunggulan – keunggulan java [13]:

1. **Berbasis GUI**

   Kita bisa membuat tampilan berbasis grafik *(Graphic User Interface/*GUI*)* untuk memudahkan pemakai berinteraksi degan program.

2. **Berorientai Objek**

   Konsep pemrograman berorientasi objek tak lain dirancang agar kita dapat memandang pemrograman sebagai suatu kehidupan nyata. Ini membuat pengembangan software menjadi lebih mudah karena kita solah – olah berhubungan dengan kehidupan nyata, bukan lagi kehidupan ‘dunia lain’. Nah, java merupakan salah satu bahasa yang memiliki dukungan penuh terhadap konsep pemrograman berorientasi objek ini.

3. **Aplikasi Web**

   Saat ini web merpakan sarana yang tidak dipisahkan dari IT. Mungkin ini disebabkan fitur web sebagai sarana komunikasi dihubungkan dengan konsep

bahwa manusia itu adalah makhluk sosial yang perlu komunikasi. Java merupakan bahasa pemrograman yang memiliki dukungan sangat baik terhadap aplikasi web. Hal ini wajar sebab memang pada awalnya java dilahirkan sebagai solusi untuk menjawab kebutuhan bahasa pengembangan yang mendukung aplikasi berbasis jaringan.

4. **Multiplatform**

   Java bisa dijalankan pada komputer dengan platform yang berbeda, asalkan JVM telah terinstal unuk platform itu. Java juga bisa digunakan untuk membuat program yang berjalan diatas berbagai *mobile device* (dengan J2ME), PDA, *embeded sistem,* dan lain – lain.

5. **Keamanan**

   Salah satu alasan mengapa java tepat digunakan untuk menangani kebutuhan aplikasi enterprise adalah keampuhannya dalam soal keamanan. Fitur keamanan java ada 2 paket, yaitu pada JDK dan pada *Java Cryptography Extension* (JCE). Fitur keamanan yang tersedia pada kedua paket itu meliputi *signature* (untuk menandatangani dokumen), *message digest*, pembangkitan kunci, autentikasi, enkripsi dan bilangan besar *(big number).*

6. **Bisa digunakan pada aplikasi jaringan terdistribusi**

   Aplikasi yang berjalan pada jaringan terdistribusi melibatkan sejumlah komputer yang berkomunikasi secara transparan, yaitu seolah sejumlah komputer itu merupakan sebuah komputer yang bersatu untuk menjalankan sebuah perintah. Dengan fitur ini bisa membuat program untuk menyelesaikan kasus tertentu dengan melibatkan beberapa komputer. Ini akan membuat penyelesaian masalah menjadi cepat.

7. **Mendukung software mission-critical**

   Java bisa digunakan untuk pengembangan software dimana tingkat error yang terjadi sangant diperhatikan, dimana sebuah error pada eksekusi program bisa mengakibatkan kerusakan fatal pada sistem secara keseluruhan. Sebagai dukungan terhadap software seperti ini, java menghilangkan sejumlah fitur

C++ yang berpotensi menghasilkan error yang fatal seperti pointer, konversi tipe tanpa dicek, dan lain – lain.

8. ***Multithread***

   Fitur *Multithread* digunakan untuk menjalankan sejumlah proses secra bersamaan. Dengan menggunakan fasilitas ini bisa membuat program java ynag menjalankan beberpa perintah sekaligus, sehingga tidak perlu menunggu sebuah perintah selesai dikerjakan untuk mengerjakan perintah lain.

### 2.8.3 Software Development Kit (SDK)

*Software Development Kit* (SDK) adalah suatu kit atau *library* dari bahasa perograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan *tools* yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis java [14].

### 2.8.4 Linux

Linux merupakan sistem operasi yang bersifat Unix-like. Linux pertama kali diperkenalkan oleh Linus Trovalds pada tahun 1991. Sistem operasi dari Linux sejak dari awalnya memang sengaja dijadikan sistem operasi terbuka dimana semua pengunanya tidak perlu membayar lisensi untuk mendapatkannya. Linux memiliki logo seekor pinguin bernama Tux yang ditunjukan pada Gambar 2.14.



**Gambar 2.18 Tux Logo Linux**

Linux juga dapat dikembangkan oleh siapa saja, dimana seluruh kode program yang ada di Linux disebarluaskan dan dapat dimodifikasi atau diganti oleh siapa saja [15].

### 2.8.5 Asterisk

Asterisk adalah sebuah proyek *open source* yang disponsori oleh Digium. Maintainer utama adalah Mark Spencer, tapi banyak patch telah memberikan kontribusi bagi masyarakat. Asterisk ini hanya berjalan pada Linux untuk Intel, meskipun ada beberapa keberhasilan di masa lalu dengan Linux PPC, dan upaya sedang berlangsung untuk pelabuhan Asterisk untuk * BSD. Digium juga menjual berbagai komponen hardware yang beroperasi dengan Asterisk. Komponen ini adalah kartu PCI yang menghubungkan saluran telepon analog standar untuk komputer. Hardware lain didukung juga, seperti perangkat keras dari dialogis dan Quicknet. Asterisk memiliki protokol VoIP sendiri, yang disebut IAX, tetapi juga mendukung SIP dan H.323. Hal ini membawa kita ke salah satu fitur Asterisk yang paling kuat: kemampuannya untuk menghubungkan teknologi yang berbeda dalam lingkungan yang kaya fitur yang sama. Misalnya, Anda bisa memiliki IAX, SIP, H.323 dan saluran telepon biasa yang terhubung melalui Asterisk [16].

### 2.8.6 Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *open source* atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 mei 2013 pada event Google I/O *Conference* untuk tahun 2013. Sejak saat itu, Android Studio mengantikan Eclipse sebagai IDE resmi untuk mengembangkan aplikasi Android [14].




**Gambar 2.19 Android Studio**

Android studio sendiri dikembangkan berdasarkan IntelliJ IDEA yang mirip dengan Eclipse disertai dengan ADT *plugin* (*Android Development Tools*). Android studio memiliki fitur :

a. Projek berbasis pada *Gradle Build*
b. *Refactory* dan pembenahan *bug* yang cepat
c. *Tools* baru yang bernama "Lint" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
d. Mendukung *Proguard And App-signing* untuk keamanan.
e. Memiliki GUI aplikasi android lebih mudah
f. Didukung oleh Google Cloud Platfrom untuk setiap aplikasi yang dikembangkan.

### 2.8.7 *Unified Modelling Language* (UML)

*Unified Modeling Language* (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML juga sebagai dasar bagi perangkat desain berorientasi objek dari IBM.

UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.

Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

Berikut akan dijelaskan empat macam diagram yang akan digunakan dalam penelitian skripsi ini, yaitu *use case* diagram, *activity* diagram, *class* diagram dan *sequence* diagram [17].

#### 2.8.7.1 *Use Case Diagram*

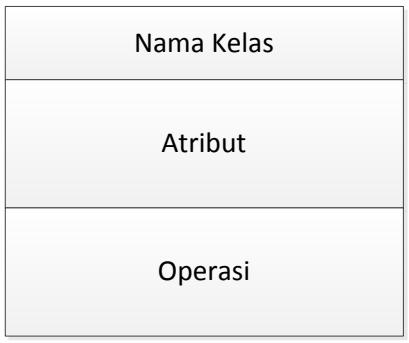
*Use Case* diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use Case* diagram terdiri atas diagram untuk *use case* dan *actor*. *Actor* mempresentasikan orang yang akan mengoperasikan atau orang yang berinterkasi dengan sistem aplikasi. *Use Case* mempresentasikan operasi-operasi yang dilakukan oleh aktor. *Use Case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

#### 2.8.7.2 *Activity Diagram*

*Activity* diagram merupakan diagram yang menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Diagram ini memiliki struktur diagram yang mirip dengan *flowchart* atau data *flow* diagram pada perancangan terstruktur. Manfaat dari diagram ini yaitu dapat membantu dalam memahami proses secara keseluruhan. *Activity* diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram.

#### 2.8.7.3 *Class Diagram*

*Class* diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. *Class* seperti juga objek adalah sesuatu yang membungkus (*encapsulate*) informasi (baca: atribut) dan perilaku (baca: operasi) dalam dirinya.




**Gambar 2.20 Notasi Kelas Dalam UML**

Pada gambar 2.20 memperlihatkan notasi kelas dalam UML, berikut ini adalah penjelasannya:

a. Bagian paling atas memuat nama kelas dan (jika ada) *stereotyoe*-nya.
b. Bagian tengah mendaftarkan atribut-atribut yang dimiliki sebuah kelas, atribut seperti yang kita ketahui, adalah informasi-informasi yang berkaitan dengan suatu kelas. Sebagai contoh, kelas Mahasiswa mungkin memiliki NIM, Nama Mahasiswa, Tgl Lahir, Alamat dan sebagainya, yang merupakan atribut dari kelas Mahasiswa tersebut.
c. sedangkan bagian paling bawah mendaftarkan operasi-operasi yang dimiliki oleh sebuah kelas. Operasi berhubungan dengan perilaku yang berhubungan dengan suatu kelas. Operasi biasanya memuat 3 bagian, yaitu: nama operasi itu sendiri, parameter-parameter operasi, serta tipe kembalian. Dalam hal ini, parameter-parameter merupakan argumen-argumen yang diterima oleh suatu operasi sebagai asupannya (*input*), sedangkan tipe kembalian adalah luaran (*output*) operasi yang bersangkutan.

Pada *class diagram,* kita dapat melihat baik nama operasi, parameter-parameter, serta tipe luarannya. Untuk mengurangi kerumitan saat menghasilakan

kode-kode dalam bahasa pemrograman kelak pada *class diagram,* adalah sangat membantu jika *class diagram* dilengkapi dengan ketiga bagian operasi di atas [18].

#### 2.8.7.4 *Sequence Diagram*

*Sequence* diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi. Masing-masing *sequence diagram* akan menggambarkan aliran-aliran pada suatu *use case.* Kita dapat membaca diagram ini dengan melihat pada objek-objek dan pesan-pesan (*message*). Objek-objek yang berperan dalam aliran diperlihatkan pada kotak empat persegi panjang yang melintas pada bagian atas diagram.

Setiap objek memiliki garis hidup (*lifeline*), yang digambarkan sebagai garis vertikal dibawah nama suatu objek. Garis hidup dimulai saat suatu objek terbentuk (diinstansiasi) dan berakhir saat objek yang bersangkutan dihancurkan. Pesan-pesan digambarkan diantara garis hidup yang dimiliki dua objek untuk memperlihatkan bagaimana objek-objek itu saling berkomunikasi. Masing-masing pesan menggambarkan suatu objek yang membuat pemanggilan fungsi dari objek lainnya. Dikemudian hari, saat kita mendefinisikan operasi-operasi untuk suatu kelas, masing-masing pesan akan menjadi operasi. Pesan-pesan dapat juga bersifat refleksif; suatu objek memanggila operasi yang dimilikinya sendiri [18].