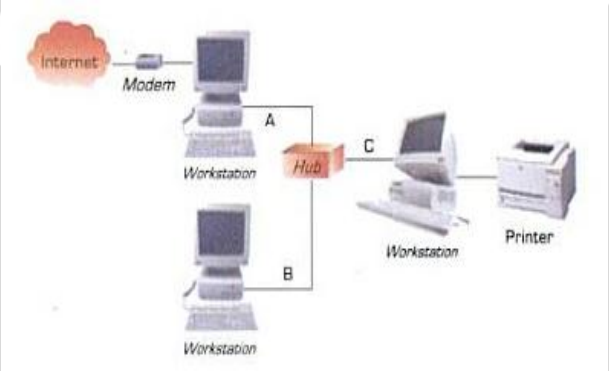
BAB II
LANDASAN TEORI

2.1 Jaringan Komputer

Jaringan komputer adalah himpunan "interkoneksi" antara 2 komputer *autonomous* atau lebih yang terhubung dengan media transmisi kabel atau tanpa kabel (*wireless*). Bila sebuah komputer dapat membuat komputer lainnya *restart, shutdown,* atau melakukan kontrol lainnya, maka komputer-komputer tersebut bukan *autonomous* (tidak melakukan kontrol terhadap komputer lain dengan akses penuh). [SYA-05 : 2]



Gambar 2.1 Optimalisasi Jaringan Komputer
Sumber : [KUS-05 : 2]

2.1.1 Model Hubungan Pada Jaringan

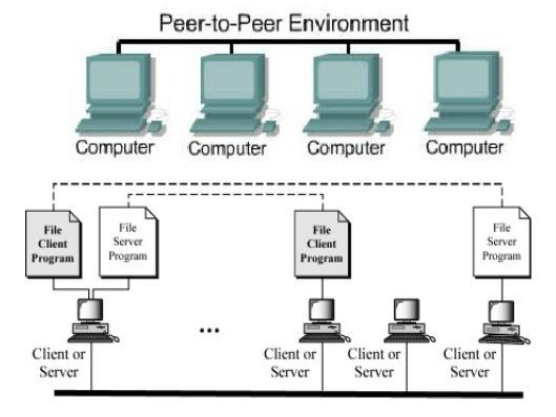
Tiap komputer atau *peripheral* yang terhubung dalam jaringan disebut dengan node. Ada dua node dalam jaringan komputer, antara lain:

2.1.1.1 Jaringan *Peer To Peer*

Peer to peer adalah suatu model di mana tiap PC dapat memakai *resource* pada PC lain atau memberikan *resource*nya untuk dipakai PC lain. Dengan kata lain dapat berfungsi sebagai *client* maupun *server* pada periode yang sama.

Kelebihan model hubungan *peer to peer* adalah tidak terlalu mahal karena tidak membutuhkan *server*, mudah dalam konfigurasinya. Kekurangan model *peer to peer* diantaranya tidak terpusat untuk penyimpanan file dan aplikasi. [SYA-05 : 2]

Mode jaringan *peer to peer* dapat dilihat pada gambar 2.2.



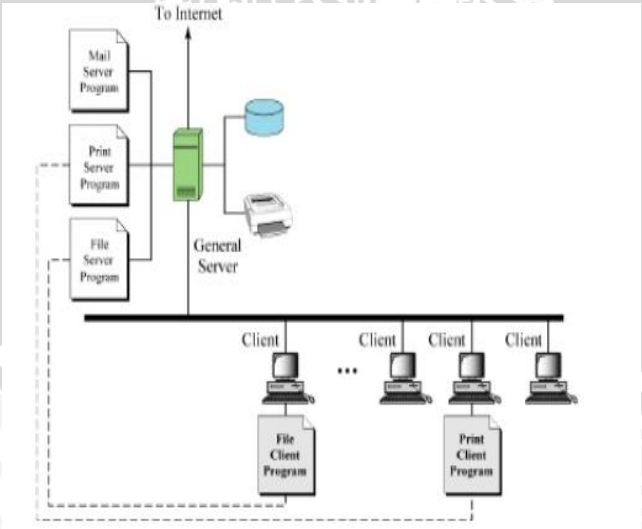
Gambar 2.2 Mode Jaringan *Peer to Peer*

Sumber : [SYA-05 : 3]

2.1.1.2 *Client-Server*

Server akan mengatur mekanisme akses *resource* yang boleh digunakan, serta mekanisme komunikasi antar node dalam jaringan. *Client* hanya bisa menggunakan *resource* yang disediakan *server* sesuai dengan otoritas yang diberikan oleh administrator. Aplikasi yang dijalankan pada sisi *client*, bisa saja merupakan *resource* yang tersedia di server. Namun hanya bisa dijalankan setelah terkoneksi ke *server*. [SYA-05 : 3]

Mode jaringan *client-server* dapat dilihat pada gambar 2.3.



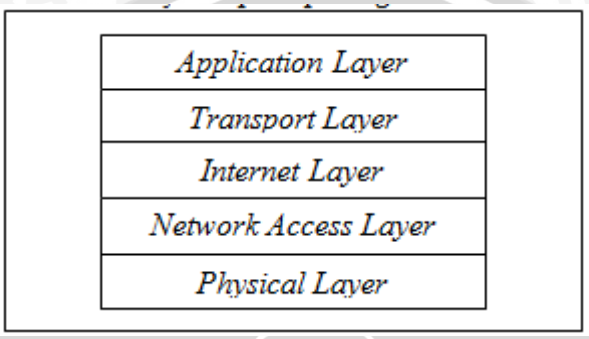
Gambar 2.3 Mode Jaringan *Client-Server*

Sumber : [SYA-05 : 4]

2.2 TCP/IP

TCP/IP (*Transport Control Protocol/Internet Protocol*) adalah sekumpulan protokol yang mengatur komunikasi data komputer dan memungkinkan komputer berbagai jenis dan vendor serta berbeda system operasi untuk berkomunikasi bersama dengan baik. [ANO-11 : 3]

Model ini memiliki 5 layer seperti pada gambar 2.4 berikut ini :



Gambar 2.4 TCP/IP *Layer*
(Sumber : SUA-09: 2)

Adapun layer-layer pada TCP/IP dan fungsi dari masing-masing lapisan adalah sebagai berikut : [SUA-09 : 2]

1. ***Physical Layer* (Lapisan Fisik)**

   Lapisan ini merupakan lapisan terbawah yang mendefinisikan besaran fisik seperti media komunikasi, rata-rata persinyalan serta skema pengkodean sinyal dan sarana system pengiriman data ke device yang terhubung ke *network.*

2. ***Network Access Layer* (*Data Link*)**

   Pada layer ini mengatur penyaluran data pada frame-frame data pada media fisik yang digunakan secara handal. Memberikan servis untuk mendeteksi dan koreksi kesalahan dari data yang ditransmisikan.

3. ***Internet Layer* (*Network*)**

   a. Lapisan ini mendefinisikan bagaimana hubungan dapat terjadi antara dua pihak yang berada pada jaringan yang berbeda seperti *network layer* pada OSI.

   b. Menjamin agar suatu paket yang dikirimkan dapat menemukan tujuannya dimanapun berada. Oleh karena itu, lapisan ini memiliki

peranan penting terutama mewujudkan *internetworking* yang meliputi wilayah luas (*worldwide* Internet).

4. ***Transport Layer***

   Lapisan ini terbagi menjadi 2 bagian, yaitu UDP dan TCP:

   a. *User Datagram Protocol* (UDP) adalah protokol *process-to-process* yang menambahakan hanya alamat port, *check-sum error control*, dan panjang informasi data dari lapisan di atasnya. (*Connectionless*)

   b. *Transmission Control Protocol* (TCP) TCP menyediakan layanan penuh lapisan transpor untuk aplikasi. TCP juga dikatakan protokol transport untuk stream yang reliabel. Dalam konteks ini artinya TCP bermakna *connectionoriented*, dengan kata lain: koneksi *end-to-end* harus dibangun dulu di kedua ujung terminal sebelum kedua ujung terminal mengirimkan data. (*Connection Oriented*)

5. ***Application Layer***

   Merupakan layer program aplikasi yang menggunakan protokol TCP/IP. beberapa diantaranya adalah Telnet, FTP (*File Transport Protocol*), SMTP (*Simple Mail Transport Protocol*), SNMP (*Simple Network Management Protocol*), HTTP (*Hypertext Transport Protocol).*

2.2.1 IP *Address*

IP *address* adalah pengidentifikasian dengan angka yang diberikan ke setiap mesin di dalam jaringan IP. IP *address* berupa bilangan biner 32 bit dan ditulis sebagai 4 urutan bilangan desimal yang dipisahkan dengan tanda titik. Format penulisan IP adalah : xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx, dengan x adalah bilangan biner 0 atau 1. Dalam implementasinya IP address ditulis dalam bilangan desimal dengan bobot antara 0 – 255 (nilai desimal mungkin untuk 1 byte). IP *address* terdiri dari bagian jaringan dan bagian host, tapi format dari bagian-bagian ini tidak sama untuk setiap IP *address*.

Jumlah bit alamat yang digunakan untuk mengidentifikasi jaringan, dan bilangan yang digunakan untuk mengidentifikasi *host* berbeda-beda tergantung kelas alamat yang digunakan. Ada tiga kelas alamat utama, yaitu kelas A, kelas B, dan kelas C. Dengan memeriksa beberapa bit pertama dari suatu alamat,

software IP bisa dengan cepat membedakan kelas IP *address* dan strukturnya. [BAY-12: 7]

Kelas dalam IP *address* dapat dilihat pada table 2.1 berikut ini:

Tabel 2.1 Kelas IP *Address*

Kelas	Oktet Pertama	Subnet	Jumlah Host	Catatan
Kelas A	1 – 126	Net.H.H.H	16.777.214	Umum
Kelas B	128 – 191	Net.Net.H.H	65.534	Umum
Kelas C	192 – 223	Net.Net.Net.H	128	Umum
Kelas D	224 – 239	Unknown	Unknown	Multicast
Kelas E	240 – 255	Unknown	Unknown	InterNIC

Sumber : [BAY-12 : 8]

Keterangan :

1. Kelas A : bit pertama dari kelompok 8 bit bernilai 0.
   Contoh 0xxxxxxx . ---- . ---- , ----
2. Kelas B : dua bit pertama kelompok 8 bit paling kiri bernilai 10.
   Contoh 10xxxxxx . ---- . ---- , ----
3. Kelas C : tiga bit pertama dari kelompok 8 bit paling kiri bernilai 110.
   Contoh 110xxxxx . ---- . ---- , ----
4. Kelas D : empat bit pertama dari kelompok 8 bit paling kiri bernilai 1110.
   Contoh 1110xxxx . ---- . ---- , ----
5. Kelas E : lima bit pertama dari kelompok 8 bit paling kiri bernilai 11110.
   Contoh 11110xxx . ---- . ---- , ----

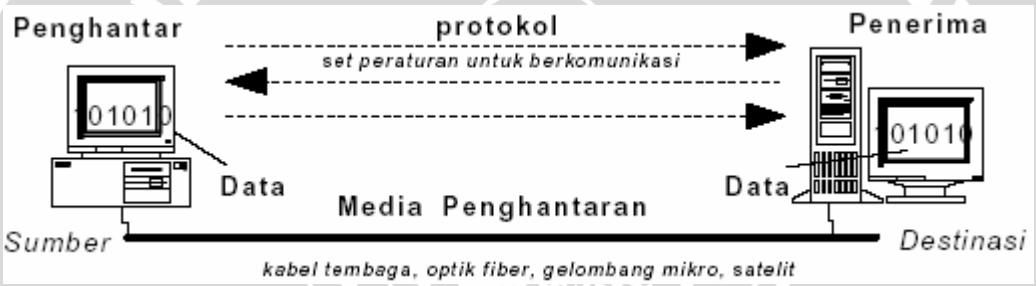
2.3 Komunikasi Data

Komunikasi data adalah merupakan bagian dari telekomunikasi yang secara khusus berkenaan dengan transmisi atau pemindahan data dan informasi diantara komputer-komputer dan piranti-piranti yang lain dalam bentuk digital

yang dikirimkan melalui media komunikasi data. Data berarti informasi yang disajikan oleh isyarat digital. Komunikasi data merupakan bagian vital dari suatu masyarakat informasi karena sistem ini menyediakan infrastruktur yang memungkinkan komputer-komputer dapat berkomunikasi satu sama lain.

Adapun komponen komunikasi data, antara lain:

1. **Pengirim** adalah piranti yang mengirimkan data.
2. **Penerima** adalah piranti yang menerima data.
3. **Data** adalah informasi yang akan dipindahkan.
4. **Media pengiriman** adalah media atau saluran yang digunakan untuk mengirimkan data.
5. **Protokol**, adalah aturan-aturan yang berfungsi untuk menyelaraskan hubungan.



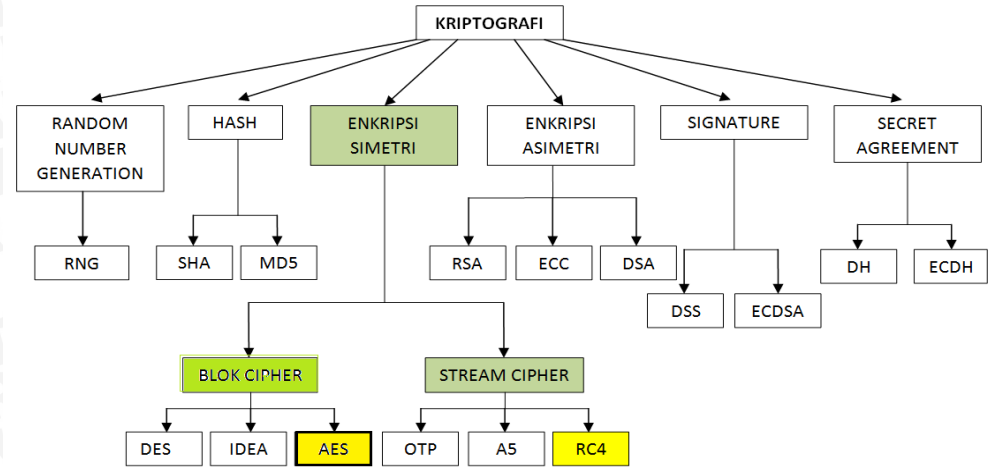
Gambar 2.5 Komunikasi Data

[RIY-04:1]

2.4 Algoritma Kriptografi

Kriptografi adalah ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan ditransfer dari suatu tempat ke tempat lain, isi dari pesan tersebut kemungkinan dapat disadap oleh pihak lain. Untuk menjaga keamanan pesan, pesan tersebut dapat di *scramble* atau diubah menjadi kode yang tidak dapat dimengerti oleh orang lain. [ARI-08 : 10]

Klasifikasi kriptografi dapat dilihat pada gambar 2.6 berikut ini.



Gambar 2.6 Klasifikasi Kriptografi

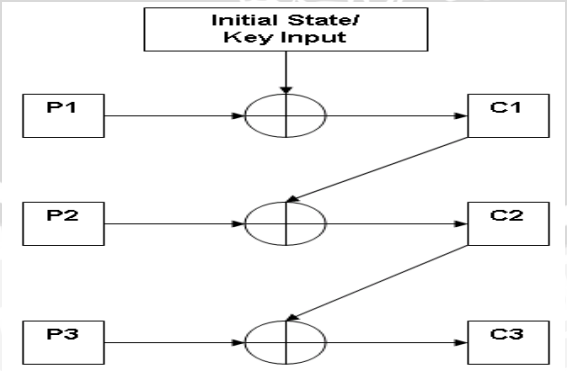
(Sumber : RIZ-11 : 144)

Secara umum berdasarkan kesamaan kuncinya, algoritma sandi dibedakan menjadi : [SUT 10 : 2]

1. Kunci simetris/*symmetric-key*, sering disebut juga dengan kunci pribadi/*private key*.
2. Kunci asimetri/*asymmetric-key*, sering disebut juga dengan kunci publik/*public key*.

Berdasarkan enkripsi dan dekripsinya, dibagi menjadi 2 antara lain: [ANO-10 : 6]

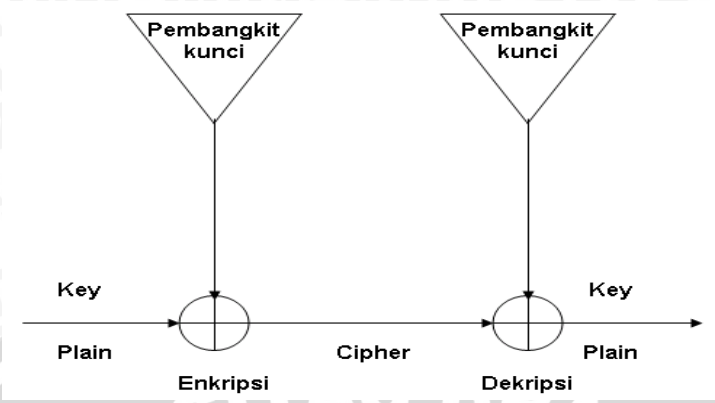
1. *Block Cipher* adalah proses enkripsi dan dekripsi ini biasanya dilakukan dalam ukuran blok tertentu



Gambar 2.7 Algoritma *Block Cipher*

(Sumber: NUG-09 : 1)

2. *Stream Cipher* adalah suatu sistem dimana proses enkripsi dan dekripsinya dilakukan dengan cara bit per bit.



Gambar 2.8 Algoritma *Stream Cipher*

(Sumber: NUG-09 : 1)

Berdasarkan kerahasiaan kuncinya dibedakan menjadi : [SUT 10 : 2]

1. Algoritma sandi kunci rahasia (*secret key*).
2. Algoritma sandi kunci publik (*public key*).

- Komponen dalam kriptografi, antara lain: [ARI-08 : 10]

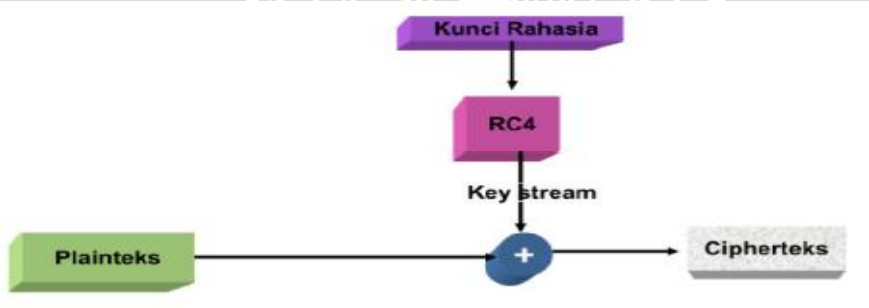
1. *Enkripsi* adalah mengubah pesan asli (*plaintext*) menjadi kode-kode yang tidak dimengerti (*ciphertext*).
2. *Dekripsi* adalah mengubah pesan yang telah dienkripsi (*ciphertext*) menjadi bentuk asalnya (*plaintext*).
3. Kunci adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Terbagi menjadi 2 bagian, yaitu kunci rahasia (*private key*) dan kunci umum (*public key*).
4. *Ciphertext* adalah suatu pesan yang telah melalui proses enkripsi. Pesan yang ada pada teks kode ini tidak bisa dibaca, karena berupa karakter-karakter yang tidak mempunyai makna (arti).
5. *Plaintext* sering disebut juga dengan *cleartext.* Teks ini merupakan pesan yang ditulis yang memiliki makna. Teks inilah yang kemudian diroses untuk menjadi *ciphertext*.

- Tujuan kriptografi antara lain sebagai berikut : [SUT-10 : 1]

1. *Confidentiality* untuk melindungi identitas pemakai atau isi pesan agar tidak dapat dibaca oleh orang lain.
2. *Data Integrity* untuk melindungi pesan agar tidak dirubah oleh orang lain.
3. *Authentication* untuk menjamin keaslian pesan.
4. *Non Repudiation* membuktikan suatu pesan berasal dari seseorang, apabila ia menyangkal mengirimkan pesan tersebut.

2.4.1 Algoritma Enkripsi Rivest Cipher 4 (RC4)

RC4 merupakan salah satu jenis *stream chiper*, yaitu memproses unit atau input data pada suatu saat. Unit atau data pada umumnya sebuah byte atau bahkan kadang bit (byte dalam hal RC4). Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variable. RC4 adalah penyandian *stream cipher* yang dibuat oleh Ron R. Iverst pada tahun 1987 untuk pengamanan RSA. Algoritmanya didasarkan pada permutasi acak. RC4 adalah algoritma yang digunakan pada metode keamanan WPA-PSK. [ANO-11:20]



Gambar 2.9 Diagram Blok Algoritma RC4 Secara Umum
(Sumber : PRA-10 : 3)

2.4.1.1 Transformasi RC4

Pada algoritma RC4, data input atau *plaintext* diproses melalui serangkaian transformasi, yang terdiri dari inisialisasi *S-Box*, menyimpan *key* dalam *key bit array*, pengacakan table *S-Box*, *generate pseudorandom byte stream*. [WIB-10 : 56]

1. Inisialisasi *S-Box*

Pada tahapan ini, *S-Box* akan diisi dengan nilai sesuai indeksnya untuk mendapatkan *S-Box* awal. Isi secara berurutan, yaitu S_0=0, S_1=1, …, S_{255} =255.

2. Menyimpan *key* dalam *key byte array*

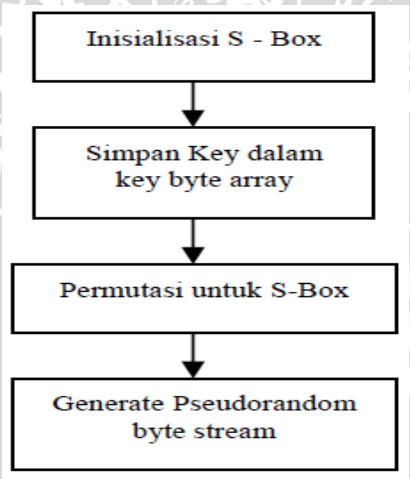
   Pada tahapan ini, kunci (*key*) yang akan digunakan untuk mengenkripsi atau dekripsi akan dimasukkan ke dalam *array* berukuran 128 *byte* secara berulang sampai seluruh *array* terisi.

3. Permutasi pada tabel *S-Box*

   Pada tahapan ini, akan dibangkitkan sebuah nilai yang akan dijadikan aturan untuk permutasi pada S-Box.

4. *Generate Pseudorandom byte stream*

   Pada tahapan ini akan dihasilkan nilai *pseudorandom* yang akan dikenakan operasi XOR untuk menghasilkan *chipertext* ataupun sebaliknya yaitu untuk menghasilkan *plaintext*.



Gambar 2.10 Proses Transformasi RC4

(Sumber : IRF-07 : 2)

2.4.1.2 Langkah-langkah RC4

Algoritma RC4 sangat sederhana bila dibandingkan dengan algoritma enkripsi lain. Secara umum tahap-tahap enkripsi RC4 adalah sebagai berikut : [PRA-10 : 3]

1. Inisialisasi larik *State* sepanjang 128 byte.

2. Memilih kunci rahasia yang ingin digunakan (panjang kunci bebas dalam jangkauan antara 1 byte sampai 128 byte, semakin panjang semakin baik).
3. Menjalankan algoritma KSA (*Key Scheduling Algorithm*) dan PRGA (*Pseudocode Random Generation Algorithm*) untuk membangkitkan bilangan acak semu 8 bit yang menyusun aliran kunci (*key stream*).
4. *Key stream* yang dihasilkan dioperasikan dengan operasi logika XOR dengan *plaintext.*

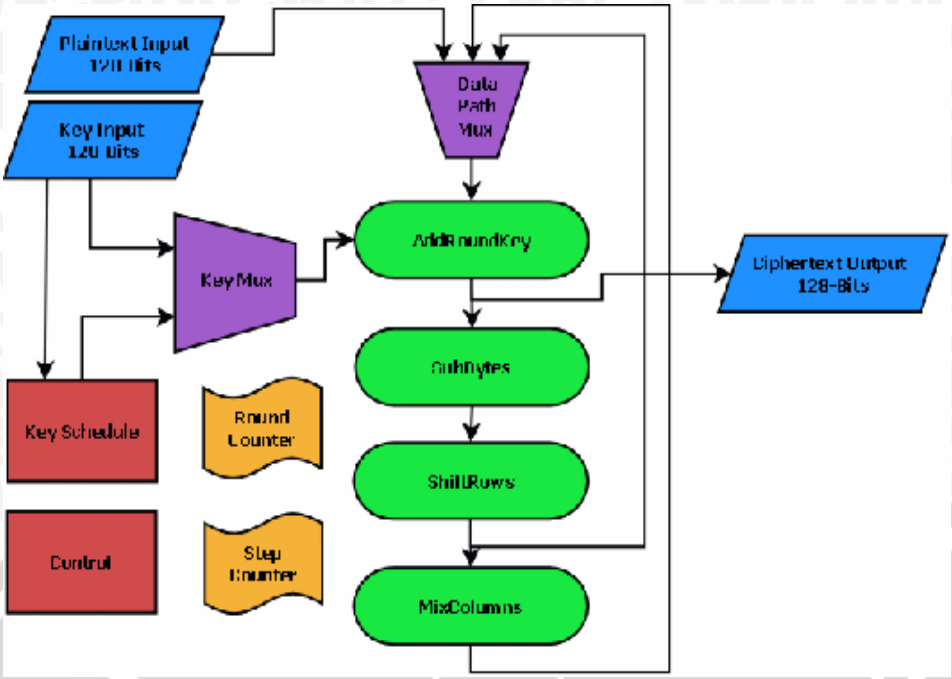
Sedangkan untuk tahap dekripsi pada RC4 adalah sebagai berikut : [PRA-10 : 3]

1. Inisialisasi larik *state* sepanjang 128 byte.
2. Gunakan kunci rahasia yang sama seperti yang digunakan pada proses enkripsi.
3. Bangkitkan *key stream* dengan algoritma KSA dan PRGA.
4. *Key stream* yang dihasilkan dioperasikan dengan operasi logika XOR dengan *ciphertext*.
5. *Plainteks* didapatkan kembali.

2.4.2 Algoritma Enkripsi *Advanced Encryption Standart* (AES)

Advanced Encryption Standart (AES) adalah algoritma yang digunakan pada metode keamanan WPA-Radius. Seperti halnya RC4, AES merupakan jenis blok *cipher,* yaitu algoritma yang memproses enkripsi per blok (8 byte atau 16 byte). [ANO-10 : 25]

Algoritma *Rijndael*, karya peneliti dari universitas di Belgia ditetapkan menjadi AES. *Rijndael* merupakan algoritma yang dapat menerima masukan data 128 bit dan menghasilkan data 128 bit pula. Bila digunakan dengan kunci 128 bit, maka kita menyebutnya sebagai AES-128. [KUR-06 : 75]



Gambar 2.11 Diagram Blok Algoritma AES Secara Umum

(Sumber : GIR-04 : 3)

Jumlah *round* / putaran (Nr) yang harus diimplementasikan pada masing-masing panjang kunci dapat dilihat pada table 2.2.

Tabel 2.2 Perbandingan Jumlah *Round* dan *Key*

	Jumlah Key (Nk)	Ukuran Block (Nb)	Jumlah Putaran (Nr)
AES – 128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

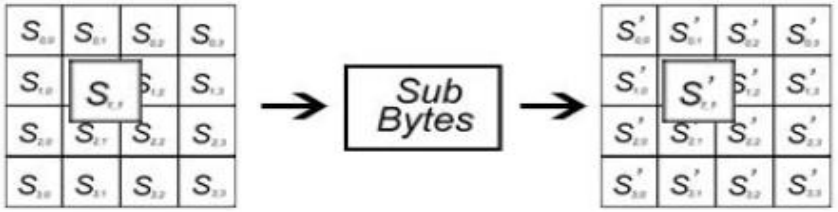
Sumber : [YUN-09 : 23]

2.4.2.1 Transformasi AES

Pada algoritma AES, data input atau *plaintext* diproses melalui serangkaian transformasi, yang terdiri dari transformasi *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey*, dengan menggunakan kunci rahasia yaitu *cipher key*.

1. *SubByteS*

Subtitusi *byte* dengan menggunakan table subtitusi (*S-box*).

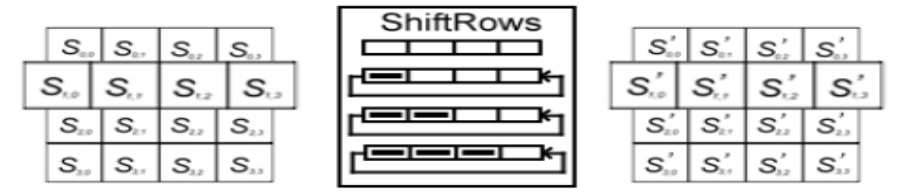


Gambar 2.12 Proses *SubBytes*

(Sumber : GIR-04 : 3)

2. *ShiftRow*

Pergeseran baris-baris *array state*.



Gambar 2.13 Proses *ShiftRows*

(Sumber : GIR-04 : 3)

3. *MixColoumn*

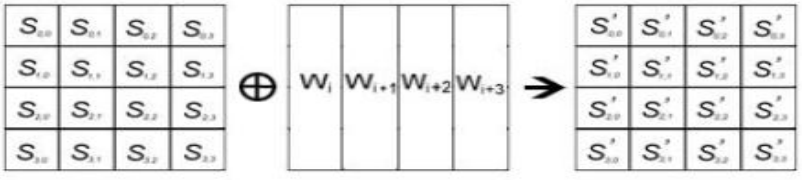
Mengacak data di masing-masing kolom *array State.*

Gambar 2.14 Proses *MixColoumn*

(Sumber : GIR-04 : 3)

4. *AddRoundKey*

Melakukan XOR antara *state* sekarang dengan *round key*.



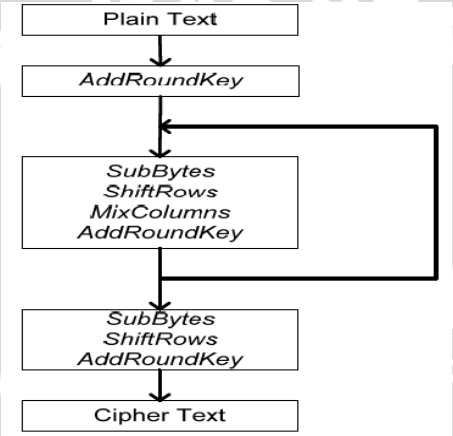
Gambar 2.15 Proses *AddRoundKey*

(Sumber : GIR-04 : 3)

2.4.2.2 Langkah-langkah AES

Secara umum tahap-tahap enkripsi AES adalah sebagai berikut : [WIB-10 : 61]

1. Inisialisasi *round.*
2. Subtitusi *byte* dengan menggunakan table subtitusi (S-*box*).
3. Geser baris-baris *array state.*
4. Mengacak data di masing-masing kolom *array State.*
5. Melakukan XOR antara *state* sekarang dengan *round key.*
6. Proses untuk putaran akhir.



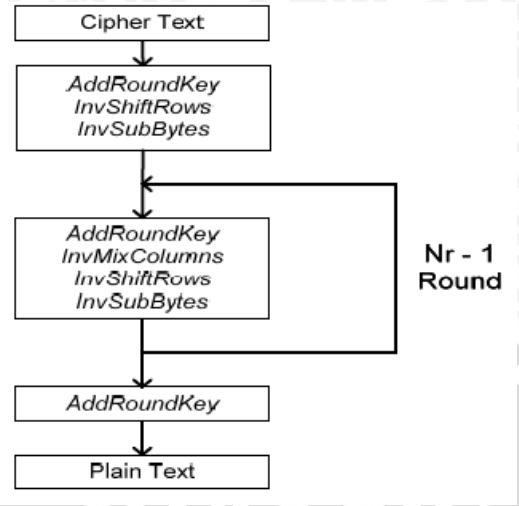
Gambar 2.16 Transformasi Enkripsi AES

(Sumber : DAR-11 : 322)

Sedangkan untuk tahap dekripsi pada AES memiliki urutan proses dan transformasi yang berbeda. Selain itu transformasi yang digunakan merupakan kebalikan dari proses transformasi penyusun setiap *round* pada proses enkripsi. Adapun langkah-langkahnya, antara lain : [DAR-11 : 322]

1. Melakukan XOR antara state awal (*ciphertext).*
2. *InvShiftRows*
3. *InvSubbytes*
4. *AddRoundKey*
5. *InvMixColumns*

Transformasi Dekripsi AES dapat dilihat pada gambar 2.17 berikut ini.



Gambar 2.17 Transformasi Dekripsi AES

(Sumber : DAR-11 : 322)

2.5 Parameter QoS (*Quality Of Service)*

QoS (*Quality of Service*) adalah kemampuan jaringan untuk menyediakan *services* yang lebih baik pada suatu trafik tertentu mulai berbagai macam teknologi meliputi jaringan IP, *frame relay* ATM dan SDH. [DIK-11 :2]

Secara khusus QoS mampu memberikan layanan yang lebih baik, dengan cara : [DIK-11 :2]

1. Mendukung *dedicated bandwidth.*
2. Memperbaiki atau memperkecil timbulnya *packetloss* dalam jaringan.
3. Menghindari timbulnya *congestion.*
4. Memberikan prioritas pada suatu tradik tertentu melalui jaringan.

Adapun parameter yang mempengaruhi QoS dalam jaringan, antara lain sebagai berikut : [DIK-11 :2]

2.5.1 *Availability*

Persentase hidupnya sistem atau subsistem telekomunikasi. Idealnya, *availability* harus mencapai 100 %. Nilai *availability* yang diakui cukup baik adalah 99,9999 % (*six nines*), yang menunjukkan tingkat kerusakan sebesar 2,6 detik per bulan.

2.5.2 *Jitter*

Jitter merupakan masalah khas dari *connectionless network* atau *packet switched network*. Jitter didefinisikan sebagai variasi *delay* yang diakibatkan oleh panjang *queue* data suatu pengolahan data dan *reassemble* paket-paket data di akhir pengiriman akibat kegagalan sebelumnya.

Secara umum *jitter* merupakan masalah dalam *slow speed links*. Diharapkan bahwa peningkatan QoS dengan mekanisme *priority buffer*, *bandwidth reservation* (RSVP, MPLS dll) dan *high speed connections* dapat mereduksi masalah *jitter* di masa yang akan datang. *Jitter* diantara titik awal dan akhir komunikasi seharusnya kurang dari 150 ms sedangkan untuk *wireless* kurang dari 5 ms.

Standart *jitter* dapat dilihat pada table 2.3 berikut ini :

Tabel 2.3 Standart *Jitter*

Kategori Degradasi	*Peak Jitter*
Sangat bagus	0 ms
Bagus	75 ms
Sedang	125 ms
Jelek	225 ms

Sumber : [DIK-11 : 4]

2.5.3 *Throughput*

Kecepatan (rate) transfer data efektif, yang diukur dalam bps. *Header* dalam paket data mengurangi nilai ini. *Throughput* dapat dihitung dengan melihat jumlah paket yang datang terhadap yang dikirim.

$$Troughput\,(Bps) = \frac{Ukuran\,File\,yang\,diterima\,(B)}{Waktu\,Aktual\,Pengiriman\,(s)}$$

Biasanya *throughput* selalu dikaitkan dengan *bandwidth*. Karena *throughput* memang bisa disebut juga dengan *bandwidth* dalam kondisi yang sebenarnya. *Bandwidth* lebih bersifat *fix* sementara *throughput* sifatnya adalah dinamis tergantung trafik yang sedang terjadi.

Contoh aplikasi dan *throughput* dapat dilihat pada table 2.4 berikut ini.

Tabel 2.4 Aplikasi dan *Throughput*

No.	Aplikasi	*Bandwidth* / Pengguna	CATATAN
1.	Web	50 – 100+ kbps	Web browser hanya menggunakan jaringan bila ada data yang diminta. Komunikasi adalah asinkron, sehingga *lag* sampai jumlah tertentu masih dapat ditolerir.
2.	*Streaming audio*	96 – 160 kbps	Setiap pengguna layanan *streaming audio* akan menggunakan *bandwidth* yang relative besar secara konstan selama di*request. Latency* dapat ditolerir sementara dengan menggunakan *buffer* yang besar pada *client.*
3.	*Voice over IP (VoIP)*	24 – 100 kbps	VoIP menggunakan *bandwidth* yang konstan untuk setiap pengguna selama panggilan. *Bandwidth* yang digunakan adalah 2 arah dan sama besarnya. *Latency* pada VoIP akan langsung terasa pada pengguna. *Lag* yang lebih besar dari beberapa ms tidak dapat diterima oleh pengguna VoIP.
4.	*Streaming video*	64-200 kbps	Seperti *streaming audio*, beberapa *delay latency* harus dihindari dengan menggunakan *buffer* pada *client. Streaming* video memerlukan *throughput* yang tinggi dan *latency* rendah untuk bekerja dengan benar.
5.	*Peer to peer* aplikasi *file sharing*	0 – tidak terbatas	Aplikasi ini cenderung menghabiskan semua *bandwidth* yang tersedia dengan cara mengirim ke sebanyak mungkin *client*, dengan secepat mungkin.

Sumber : [BAY-11 : 19]

2.5.4 *Packet Loss*

Packet Loss, merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena *retransmisi* akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut.

Packet loss untuk aplikasi *voice* dan multimedia dapat ditoleransi sampai dengan 20% untuk *single Access Point.* Performansi *packet loss* dapat dilihat pada tabel 2.5 berikut ini.

Tabel 2.5 Performansi *Packet Loss*

KATEGORI DEGREDASI	***PACKET LOSS***
Sangat bagus	0
Bagus	3 %
Sedang	15 %
Jelek	25 %

Sumber : [DIK-11 : 5]

Beberapa penyebab terjadinya *paket loss* yaitu: [DIK-11 : 5]

1. *Congestion*, disebabkan terjadinya antrian yang berlebihan dalam jaringan.
2. Node yang bekerja melebihi kapasitas *buffer*.
3. Memory yang terbatas pada node.
4. *Policing* atau kontrol terhadap jaringan untuk memastikan bahwa jumlah trafik yang mengalir sesuai dengan besarnya *bandwidth*. Jika besarnya trafik yang mengalir didalam jaringan melebihi dari kapasitas *bandwidth* yang ada maka *policing* kontrol akan membuang kelebihan trafik yang ada.

2.5.5 *Delay*

Delay adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan.

$$T_{total} = t_e + t_d + 3t_p + 2t_{prop} + 2t_t + t_w$$

Dengan,

t_e : *Delay* Enkripsi (ms)

t_d : *Delay* Dekripsi (ms)

t_p : *Delay* Proses (ms)

t_{prop} : *Delay* Propagasi (ms)

t_t : *Delay* Transmisi (ms)

t_w : *Delay* Antrian (ms)

T_{total} : *Delay* Total (ms)

Tabel 2.6 Target Performansi *Delay* Untuk Aplikasi Data

No	Medium	Application	Degree of symmetry	Typical amount of data	Key performance parameters and target values		
					One-way delay (Note)	Delay variation	Information loss
1	Data	Web-browsing – HTML	Primarily one-way	~10 KB	Preferred < 2 s /page Acceptable < 4 s /page	N.A.	Zero
2	Data	Bulk data transfer/retrieval	Primarily one-way	10 KB-10 MB	Preferred < 15 s Acceptable < 60 s	N.A.	Zero
3	Data	Transaction services – high priority e.g. e-commerce, ATM	Two-way	< 10 KB	Preferred < 2 s Acceptable < 4 s	N.A.	Zero
4	Data	Command/control	Two-way	~ 1 KB	< 250 ms	N.A.	Zero
5	Data	Still image	One-way	< 100 KB	Preferred < 15 s Acceptable < 60 s	N.A.	Zero
6	Data	Interactive games	Two-way	< 1 KB	< 200 ms	N.A.	Zero
7	Data	Telnet	Two-way (asymmetric)	< 1 KB	< 200 ms	N.A.	Zero
8	Data	E-mail (server access)	Primarily one-way	< 10 KB	Preferred < 2 s Acceptable < 4 s	N.A.	Zero
9	Data	E-mail (server to server transfer)	Primarily one-way	< 10 KB	Can be several minutes	N.A.	Zero
10	Data	Fax ("real-time")	Primarily one-way	~ 10 KB	< 30 s/page	N.A.	$<10^{-6}$ BER
11	Data	Fax (store & forward)	Primarily one-way	~ 10 KB	Can be several minutes	N.A.	$<10^{-6}$ BER
12	Data	Low priority transactions	Primarily one-way	< 10 KB	< 30 s	N.A.	Zero
13	Data	Usenet	Primarily one-way	Can be 1 MB or more	Can be several minutes	N.A.	Zero
	NOTE – In some cases, it may be more appropriate to consider these values as response times.						

Sumber [ITU-TG 1010 : 9]

Ada beberapa penyebab terjadinya *delay*, antara lain:

1. *Congestion*, disebabkan terjadinya antrian yang berlebihan dalam jaringan.
2. Penggunaan paket-paket data yang besar pada jaringan berkecepatan rendah.
3. Adanya paket-paket data dengan ukuran berbeda-beda.
4. Perubahan kecepatan antar jaringan WAN.
5. Pemadatan *bandwidth* secara tiba-tiba.
6. Jarak antara PC dengan *access point.*

Adapun macam-macam delay, antara lain:

2.5.5.1 *Delay* Enkripsi

Delay enkripsi merupakan waktu yang dibutuhkan untuk mengenkripsi data. *Delay* ini dipengaruhi oleh *prosessor* yang digunakan, semakin cepat kinerjanya, semakin kecil *delay* enkripsinya. *Delay* enkripsi dapat dirumuskan sebagai berikut :

$$T_{enkripsi} = \frac{L}{V}$$

dengan,

$T_{enkripsi}$ = *Delay* enkripsi (s)

L = Panjang data yang dienkripsi (bit)

V = kecepatan *prossesor* (bps)

2.5.5.2 *Delay* Dekripsi

Delay dekripsi merupakan waktu yang dibutuhkan untuk mendekripsi data *Delay* ini dipengaruhi oleh *prosessor* yang digunakan, semakin cepat kinerjanya, semakin kecil *delay* dekripsinya. Sedangkan rumus *delay* dekripsi adalah :

$$T_{dekripsi} = \frac{L}{V}$$

dengan,

$T_{dekripsi}$ = *Delay* dekripsi (s)

L = Panjang data yang didekripsi (bit)

V = kecepatan *prossesor* (bps)

2.5.5.3 Delay Propagasi

Delay propagasi adalah waktu perambatan atau perjalanan yang dibutuhkan oleh data bit, *byte* atau paket dari suatu *node* ke *node* yang lain melalui sebuah media transmisi. Kecepatan transmisi tergantung pada karakteristik fisik koneksi antara pengirim dan penerima. Besarnya *delay* dapat dirumuskan sebagai berikut[ANA-08:33]:

$$t_{prop} = \frac{d}{c}$$

dengan,

t_{prop} = waktu propagasi (s)

d = jarak antara *base station* ke *subscriber station* (m)

c = kecepatan gelombang elektromagnetik (m/s)

2.5.5.4 Delay Proses

Delay proses adalah waktu yang dibutuhkan untuk memproses paket data dari pengirim ke penerima. *Delay* proses pada WLAN merupakan *delay* enkapsulasi dan *delay* dekapsulasi. Enkapsulasi adalah sebuah proses menambah *header* atau melakukan pemaketan pada sebuah data. Dekapsulasi adalah proses pemisahan header IP terluar pada paket yang datang, sehingga datagram yang ditumpangkan dapat dikirimkan dan diakses pada tujuan yang sebenarnya.

Berikut ini adalah format data pada WLAN saat terjadi pemrosesan data dari pengirim ke penerima.

a. Format Data Pada *Network Layer*

Data yang berasal dari layer aplikasi pada *layer network* ditambah *header* IP dan *header* TCP, sehingga format datanya berubah. Besarnya *header* ini adalah 20 *byte* dan *header* TCP adalah 20 *byte*.

IP *Header* (20 *byte*)	TCP *Header* (20 *byte*)	Application Data (1460 *byte*)

Gambar 2.18 Format Data Pada *Layer Network*

(Sumber : WIB-10 : 20)

Ketika data dikirim dari sumber melewati *layer* aplikasi menuju *layer network*, data akan diubah menjadi segmen. Pada layer ini, data akan

mengalami penambahan *header* TCP dan *header* IP. Jika data yang dikirimkan sebesar 1460 *byte*, maka besar *message* data ($W_{message}$) dapat diperoleh sebagai berikut :

$W_{message} = W_{data} + Header_{TCP} + Header_{IP}$

$= 1460\ byte + 20\ byte + 20\ byte$

$= 1500\ byte$

b. Format Data Pada Data *Link Layer*

Setelah melewati protokol TCP/IP, data diteruskan ke layer data *link.* Pada layer ini, data yang berasal dari protocol TCP/IP ditambah *header* MAC sebesar 30 *byte* dan ditambahkan FCS (*Frame Check Squence)* sebesar 4 *byte*. Sehingga format data berubah menjadi :

MAC *Header* (30 *byte*)	IP *Header* (20 *byte*)	TCP *Header* (20 *byte*)	Application Data (1460 *byte*)	FCS (4 *byte*)

Gambar 2.19 Format Data Pada *Layer Data Link*

(Sumber : WIB-10 : 20)

Pada *layer* ini *message* data diubah menjadi *frame.* Sehingga besar datanya menjadi :

$W_{frame} = W_{message} + Header\ \text{MAC} + \text{FCS}$

$= 1500\ byte + 30\ byte + 4\ byte$

$= 1534\ byte$

c. Format Data Pada *Physical Layer*

Setelah melewati *layer data link,* data diteruskan ke *layer physical.* Dalam protocol ini data ditambahkan PLCP (*Physical Layer Convergence Protocol*) *Preamble* dan PLCP *Header* yang besar 18 *byte* dan 6 *byte.* Sehingga format datanya menjadi sebagai berikut :

PLCP *Preamble* (18 *byte*)	PLCP *Header* (6 *byte*)	MAC *Header* (30 *byte*)	IP *Header* (20 *byte*)	TCP *Header* (20 *byte*)	Application Data (1460 *byte*)	FCS (4 *byte*)

Gambar 2.20 Format Data Pada *Layer Data Link*

(Sumber : WIB-10 : 20)

Dari *data link*, data dikirimkan ke *layer physical* dan mengalami penambahan PLCP *Preamble* sebanyak 18 *byte* dan PLCP *Header* sebanyak 6 *byte* menjadi :

$W_{total} = W_{frame} + Header_{TCP} + Header_{IP}$

$= 1534\ byte + 18\ byte + 6\ byte$

$= 1558\ byte$

Maka besar *delay* enkapsulasi adalah :

$$t_{enc} = \frac{W_{frametotal}}{C_{pros}} \text{ x 8 bit/byte}$$

Sedangkan *delay* dekapsulasi dirumuskan :

$$t_{dec} = \frac{W_{frametotal}}{C_{prosz}} \text{ x 8 bit/byte}$$

Dengan demikian delay prosesnya :

$$t_{pros} = t_{enc} + t_{dec}$$

2.5.5.5 *Delay* Transmisi

Delay transmisi adalah waktu yang dibutuhkan untuk meletakkan semua data pada medium, dipengaruhi oleh ukuran paket dan kapasitas media transmisi. Berikut rumus *delay* transmisi [WIB-10:22]:

$$t_{trans} = \frac{W_{frame\ total}}{C_{trans}}$$

dengan,

t_{trans} = *delay* transmisi (s)

$W_{frame\ total}$ = besar frame total (bit)

C_{trans} = kecepatan transmisi (bps)

2.5.5.6 Delay Antrian

Delay antrian adalah waktu dimana paket data tersebut berada dalam antrian untuk ditransmisikan. Selama waktu ini paket data menunggu sampai selesainya paket lain ditransmisikan. Jika antrian kosong dan tidak ada paket lain yang ditransmisikan, maka *delay* antrian tidak terjadi.

Besarnya total waktu antrian t_w adalah :

$$t_w = t_{queue} + t_s$$

dengan,

t_w = total waktu antrian (s)

t_{queue} = waktu tunggu paket (s)

t_s = rata-rata kecepatan pelayanan (s)

dengan t_s adalah rata-rata kecepatan pelayanan, dimana sama dengan

$$t_s = \frac{1}{\mu}$$

dengan kecepatan kedatangan = λ (paket/detik), dan kecepatan pelayanan adalah μ (paket/detik). Performansi sistem antrian ditunjukkan dalam bentuk ρ (utilization)

$$\rho = \frac{\lambda}{\mu}$$

dimana besar rata-rata panjang antrian paket (N) adalah

$$N = \sum_{m=1}^{\infty} pm = \frac{\rho}{\mu(1-\rho)}$$

waktu tunggu dari paket data dapat dirumuskan

$$t_{queue} = T - t_s$$

$$t_{queue} = \frac{1/\mu}{1-\rho} - \frac{1}{\mu} = \frac{1/\mu^2}{1-\rho} = \frac{\lambda}{\mu(\mu-\lambda)}$$

maka total *delay* antrian yang terjadi dengan memasukkan persamaan t_{queue} di atas adalah

$$t_w = \frac{\lambda-\mu}{\mu-\lambda} + \frac{1}{\mu}$$

dengan

λ = kecepatan kedatangan (paket/detik)

μ = kecepatan pelayanan (paket/detik)

2.5.6 *Mean Opinion Score (MOS)*

Metode ini digunakan untuk menentukan kualitas suara dalam jaringan IP berdasar kepada standart ITU-T P.800. Rekomendasi nilai ITU-T P.800 untuk nilai MOS dapat dilihat pada table 2.7 berikut ini.

Tabel 2.7 Rekomendasi MOS

Nilai MOS	Opini
5	Sangat baik
4	Baik
3	Cukup baik
2	Tidak baik
1	Buruk

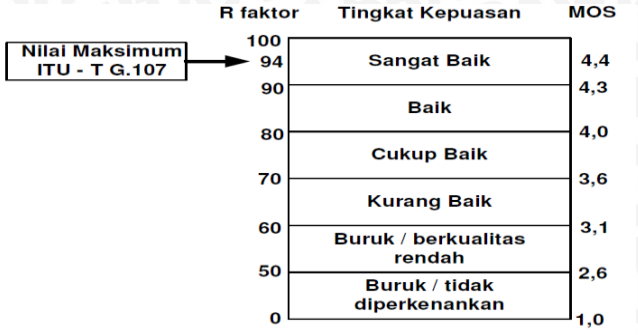
Sumber : [DIK-11 : 7]

Metode MOS dirasakan kurang efektif untuk mengestimasi kualitas layanan suara untuk VoIP, hal ini dikarenakan :

1. Tidak tedapatnya nilai yang pasti terhadap parameter yang mempengaruhi kualitas layanan suara dalam VoIP.
2. Setiap orang memiliki standar yang berbeda-beda terhadap suara yang mereka dengar dengan hanya melalui percakapan.

2.5.7 *Estimate* E-Model

Di dalam jaringan VoIP, tingkat penurunan kualitas yang diakibatkan oleh transmisi data memegang peranan penting terhadap kualitas suara yang dihasilkan, hal yang menjadi penyebab penurunan kualitas suara ini diantaranya adalah *delay* , *paket loss* dan *echo*. Pendekatan matematis yang digunakan untuk menentukan kualitas suara berdasarkan penyebab menurunnya kualitas suara dalam jaringan VoIP dimodelkan dengan E – Model yang distandardkan kepada ITU–T G.107 .



Gambar 2.21 Korelasi E–Model (ITU G.107) dengan MOS (ITU P.800)

(Sumber : DIK-11 : 9)