BAB 2
TINJAUAN PUSTAKA

2.1 Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan komputer, printer dan peralatan lainnya yang terhubung. Informasi dan data bergerak melalui kabel-kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada printer yang sama dan bersama sama menggunakan *hardware*/*software* yang terhubung dengan jaringan. Tiap komputer, printer atau perangkat keras yang terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan *node*. Sebuah jaringan biasanya terdiri dari dua atau lebih komputer yang saling berhubungan diantara satu dengan yang lain, dan saling berbagi sumber daya misalnya CDROM, Printer, pertukaran data, atau memungkinkan untuk saling berkomunikasi secara elektronik.

2.1.1 Jenis Jaringan Komputer

Ada 3 macam jenis jaringan[1], yaitu :

1. *Local Area Network* (LAN)
   LAN adalah jaringan yang dibatasi oleh area yang relative kecil, umumnya dibatasi oleh area lingkungan seperti sebuah perkantoran di sebuah gedung, atau sebuah sekolah, dan tidak jauh dari sekitar 1 km persegi.
2. *Metropolitan Area Network* (MAN)
   MAN meliputi area yang lebih besar dari LAN, misalnya antar wilayah dalam satu provinsi. Dalam hal ini jaringan menghubungkan beberapa buah jaringan-jaringan kecil ke dalam lingkungan area yang lebih besar, sebagai contoh yaitu jaringan Bank dimana beberapa kantor cabang sebuah Bank di dalam sebuah kota besar dihubungkan antara satu dengan lainnya.

3. *Wide Area Network* (WAN)

   Wide Area Networks adalah jaringan yang lingkupnya biasanya sudah menggunakan media satelit atau kabel bawah laut sebagai contoh keseluruhan jaringan Bank yang ada di Indonesia atau yang ada di Negara-negara lain.

2.1.2 Arsitektur Jaringan Komputer

Arsitektur sebuah jaringan komputer dibedakan menjadi arsitektur fisik dan arsitektur logik. Arsitektur fisik berkaitan dengan susunan fisik sebuah jaringan komputer, bisa juga disebut dengan topologi fisik jaringan yaitu menjelaskan hubungan perkabelan dan lokasi *node* (simpul) atau *workstation*. Sedangkan arsitektur logik berkaitan dengan logika hubungan masing-masing komputer dalam jaringan atau menjelaskan aliran data dari satu user ke user lainnya dalam jaringan.

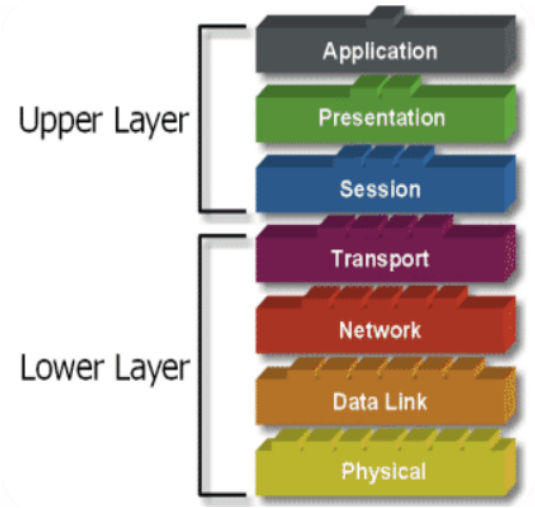
Topologi adalah bentuk hubungan dari suatu jaringan (*map of network*). Topologi fisik dari sebuah jaringan adalah merujuk pada konfigurasi kabel, komputer, dan perangkat lainnya. Terdapat tiga topologi fisik dasar yaitu: topologi *bus,*topologi *star, dan* topologi *ring*.

2.2 Protokol Jaringan Komputer

Pada saat network baru muncul, kebanyakan komputer hanya dapat berkomunikasi dengan komputer yang dibuat oleh perusahaan yang sama. Sebagai contoh, perusahaan harus menggunakan seluruh solusi dari *DECnet* atau seluruh solusi dari IBM, tapi tidak bisa kedua – duanya. Pada akhir tahun 1970, *International Organization for Standarization* (ISO) membuat model referensi *Open System Interconnection* (OSI) sebagai solusi untuk mengatasi masalah kompatibilitas ini. Model OSI dimaksudkan untuk membantu para *vendor* (vendor di sini adalah perusahaan pembuat perangkat keras atau pembuat perangkat lunak) agar bisa membuat alat – alat dan perangkat lunak yang bisa saling bekerja sama, dalam bentuk protokol – protokol sehingga *network* dengan *vendor – vendor* yang berbeda bisa saling bekerja sama.[1]

2.2.1 Model Referensi OSI

The International Organization for Standarization (ISO) mengembangkan model referensi *Open System Interconnection* (OSI) sebagai pemandu untuk mendefinisikan sekumpulan protokol terbuka. Walaupun minat pada protokol OSI telah berkurang, model referensi OSI tetap merupakan standar yang paling umum untuk menggambarkan dan membandingkan rangkaian protokol. Model OSI memiliki tujuh *layer* (lapisan).



Gambar 2.1 OSI Layer

a. *Physical Layer* (Lapisan Fisik)

Berfungsi untuk mendefinisikan media transmisi jaringan, metode pensinyalan, sinkronisasi bit, arsitektur jaringan (seperti halnya *Ethernet* atau *Token Ring*), topologi jaringan dan pengabelan. Selain itu, level ini juga mendefinisikan bagaimana *Network Interface Card* (NIC) dapat berinteraksi dengan media kabel atau radio.

b. *Data Link Layer* (Lapisan Data Link)

Befungsi untuk menentukan bagaimana bit-bit data dikelompokkan menjadi format yang disebut sebagai frame. Selain itu, pada level ini terjadi koreksi kesalahan, *flow control*, pengalamatan perangkat keras (seperti halnya *Media*

Access Control Address (MAC Address)), dan menetukan bagaimana perangkat-perangkat jaringan seperti *hub, bridge, repeater*, dan *switch layer* 2 beroperasi. Spesifikasi IEEE 802, membagi level ini menjadi dua level anak, yaitu lapisan *Logical Link Control* (LLC) dan lapisan *Media Access Control* (MAC).

c. *Network Layer* (Lapisan Jaringan)

Network layer berfungsi untuk pengendalian operasi subnet dengan meneruskan paket-paket dari satu node ke node lain dalam jaringan. Masalah desain yang penting adalah bagaimana caranya menentukan route pengiriman paket dari sumber ke tujuannya. Route dapat didasarkan pada table statik yang dihubungkan ke network. Route juga dapat ditentukan pada saat awal percakapan misalnya session terminal. Route juga sangat dinamik, dapat berbeda bagi setiap paketnya, dan karena itu, route pengiriman sebuah paket tergantung beban jaringan saat itu. Bila pada saat yang sama dalam sebuah subnet terdapat terlalu banyak paket,maka ada kemungkinan paket-paket tersebut tiba pada saat yang bersamaan. Hal ini dapat menyebabkan terjadinya *bottleneck* (penyempitan di bagian ujung, seperti leher botol).Pengendalian kemacetan seperti itu juga merupakan tugas *network layer*.

d. *Transport Layer* (Lapisan Transpor)

Fungsi dasar transport layer adalah menerima data dari session layer, memecah data menjadi bagian-bagian yang lebih kecil bila perlu, meneruskan data ke *network layer*, dan menjamin bahwa semua potongan data tersebut bisa tiba di sisi lainnya dengan benar. Selain itu, semua hal tersebut harus dilaksanakan secara efisien, dan bertujuan dapat melindungi layer-layer bagian atas dari perubahan teknologi hardware yang tidak dapat dihindari.

Dalam keadaan normal, *transport layer* membuat koneksi jaringan yang berbeda bagi setiap koneksi *transport* yang diperlukan oleh *session layer*. Bila koneksi transport memerlukan *throughput* yang tinggi, maka transport layer dapat membuat koneksi jaringan yang banyak. Transport layer membagi-bagi pengiriman data ke sejumlah jaringan untuk meningkatkan *throughput*.

e. *Session Layer* (Lapisan Sesi)

Session layer mengijinkan para pengguna untuk menetapkan session dengan pengguna lainnya. Layer ini membuka, mengatur dan menutup suatu session antara aplikasi-aplikasi. Sebuah session selain memungkinkan transport data biasa, seperti yang dilakukan oleh *transport layer*, juga menyediakan layanan yang istimewa untuk aplikasi-aplikasi tertentu. Sebuah session digunakan untuk memungkinkan seseorang pengguna log ke *remote timesharing system* atau untuk memindahkan file dari satu mesin ke mesin lainnya.

Sebuah layanan *session layer* yang lain adalah untuk melaksanakan pengendalian dialog. Session dapat memungkinkan lalu lintas bergerak dalam bentuk dua arah pada suatu saat, atau hanya satu arah saja. Jika pada satu saat lalu lintas hanya satu arah saja (analog dengan rel kereta api tunggal), *session layer* membantu untuk menentukan giliran yang berhak menggunakan saluran pada suatu saat. Layanan *session* di atas disebut manajemen token. Untuk sebagian protokol, adalah penting untuk memastikan bahwa kedua pihak yang bersangkutan tidak melakukan operasi pada saat yang sama. Untuk mengatur aktivitas ini, *session layer* menyediakan token-token yang dapat digilirkan. Hanya pihak yang memegang token yang diijinkan melakukan operasi kritis.

f. *Presentation Layer* (Lapisan Presentasi)

Presentation layer melakukan fungsi-fungsi tertentu yang diminta untuk menjamin penemuan sebuah penyelesaian umum bagi masalah tertentu. Selain memberikan sarana-sarana pelayanan untuk konversi, format dan enkripsi data, presentation layer juga bekerja dengan file berformat ASCII, EBCDIC, JPEG, MPEG, TIFF, PICT, MIDI, dan Quick Time.

Presentation layer tidak mengijinkan pengguna untuk menyelesaikan sendiri suatu masalah. Tidak seperti layer-layer di bawahnya yang hanya melakukan pemindahan bit dari satu tempat ke tempat lainnya, *presentation layer* memperhatikan syntax dan semantik informasi yang dikirimkan.

Satu contoh layanan presentation adalah *encoding data*. Kebanyakan pengguna tidak memindahkan *string bit biner* yang random. Para pengguna saling

bertukar data seperti nama orang, tanggal, jumlah uang, dan tagihan. Item-item tersebut dinyatakan dalam bentuk string karakter, bilangan interger, bilangan floating point, struktur data yang dibentuk dari beberapa item yang lebih sederhana. Terdapat perbedaan antara satu komputer dengan komputer lainnya dalam memberi kode untuk menyatakan string karakter (misalnya, ASCII dan Unicode), integer (misalnya komplemen satu dan komplemen dua), dsb.

Untuk memungkinkan dua buah komputer yang memiliki presentation yang berbeda untuk dapat berkomunikasi, struktur data yang akan dipertukarkan dapat dinyatakan dengan cara abstrak, sesuai dengan ”*encoding standard*” yang akan digunakan “pada saluran”. *Presentation layer* mengatur data struktur abstrak ini dan mengkonversi dari *representation* yang digunakan pada sebuah komputer menjadi ”*representation standard*” jaringan, dan sebaliknya.

g. *Application Layer* (Lapisan Aplikasi)

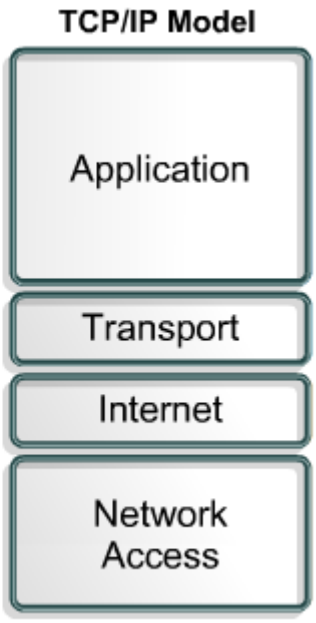
Lapisan ini bertugas memberikan sarana pelayanan langsung ke user, yang berupa aplikasi-aplikasi dan mengadakan komunikasi dari program ke program. Jika kita mencari suatu file dari file server untuk digunakan sebagai aplikasi pengolah kata, maka proses ini bekerja melalui layer ini. Demikian pula jika kita mengirimkan e-mail, browse ke internet, chatting, membuka *telnet session*, atau menjalankan FTP, maka semua proses tersebut dilaksanakan di layer ini.

Application layer terdiri dari bermacam-macam protokol. Misalnya terdapat ratusan jenis terminal yang tidak kompatibel di seluruh dunia, kemudian kita memerlukan aplikasi yang diharapkan bekerja pada jaringan dengan bermacam-macam terminal, yang masing-masing memiliki layout layar yang berlainan, mempunyai cara urutan penekanan tombol yang berbeda untuk penyisipan dan penghapusan teks, memindahkan sensor dan sebagainya. Suatu cara untuk mengatasi masalah seperti di atas, adalah dengan menentukan terminal virtual jaringan abstrak, sehingga editor dan program-program lainnya dapat ditulis agar saling bersesuaian. Untuk menangani setiap jenis terminal, satu bagian software harus ditulis untuk memetakan fungsi terminal virtual jaringan ke terminal sebenarnya. Fungsi lainnya adalah pemindahan file. Sistem file yang satu dengan

lainnya memiliki konvensi penamaan yang berbeda, cara menyatakan baris-baris teks yang berbeda, dan sebagainya. Perpindahan file dari sebuah sistem ke sistem lainnya yang berbeda memerlukan penanganan untuk mengatasi adanya ketidak-kompatibel-an ini. Tugas tersebut juga merupakan pekerjaan *application layer*, seperti pada surat elektronik, *remote job entry*, *directory lookup*, dan berbagai fasilitas bertujuan umum dan fasilitas bertujuan khusus lainnya.

2.2.2 Protokol TCP/IP

TCP/IP adalah sekumpulan protokol yang terdapat di dalam jaringan komputer (network) yang digunakan untuk berkomunikasi atau bertukar data antar komputer.



Gambar 2.2 TCP/IP Model

TCP/IP merupakan standard protokol pada jaringan internet yang menghubungkan banyak komputer yang berbeda jenis mesin maupun system operasinya agar dapat berinteraksi satu sama lain.

2.2.2.1 Layanan TCP/IP

Berikut ini adalah layanan “tradisional” yang dilakukan TCP/IP:

1. Pengiriman File (*file transfer*). *File Transfer Protokol* (FTP) memungkinkan pengguna komputer yang satu untuk dapat mengirim ataupun menerima file ke komputer jaringan. Karena masalah keamanan data, maka FTP seringkali memerlukan nama pengguna (user name) dan *password*, meskipun banyak juga FTP yang dapat diakses melalui *anonymous*, alias tidak ber-*password*.
2. *Remote Login*. *Network Terminal Protokol* (telnet) memungkinkan pengguna komputer dapat melakukan login ke dalam suatu komputer di dalam suatu jaringan. Jadi hal ini berarti bahwa pengguna menggunakan komputernya sebagai perpanjangan tangan dari komputer jaringan tersebut.
3. *Computer Mail*. Digunakan untuk menerapkan sistem e-mail (*elektronik mail*).
4. *Network File System* (NFS). Pelayanan akses file-file jarak jauh yang memungkinkan klien-klien untuk mengakses file-file pada komputer jaringan jarak jauh walaupun file tersebut disimpan secara lokal.
5. *Remote Execution* memungkinkan pengguna komputer untuk menjalankan suatu program dari komputer yang berbeda. Biasanya berguna jika pengguna menggunakan komputer yang terbatas, sedangkan ia memerlukan sumber yang banyak dalam suatu system komputer. Ada beberapa jenis *remote execution*, ada yang berupa perintah-perintah dasar saja, yaitu yang dapat dijalankan dalam system komputer yang sama dan ada pula yang menggunakan "*procedure remote call system*", yang memungkinkan program untuk memanggil subroutine yang akan dijalankan di system komputer yang berbeda. (sebagai contoh dalam *Berkeley UNIX* ada perintah "*rsh*" dan "*rexec*").

2.2.2.2 Arsitektur TCP/IP

Protocol-protocol TCP/IP memberikan dukungan jaringan untuk menghubungkan seluruh tempat dan host serta mengikuti aturan standar tentang bagaimana komputer-komputer berkomunikasi dan bagaimana jaringan-jaringan diinterkoneksikan. Protocol-protocol TCP/IP mengikuti model konsep empat layer yang dikenal sebagai model *Department of Defense* (*DOD*): *Application, Transport, Internet, dan Network Interface*[2].

1. ***Layer Application***

   Layer Application terletak di atas pada model TCP/IP konsep empat layer dan merupakan tempat program software memperoleh akses ke jaringan. Layer ini sesuai dengan layer Session, layer Presentation, dan *layer application* pada model OSI. Beberapa layanan dan utiliti TCP/IP beroperasi pada Layer Application. Layanan dan *utility* tersebut meliputi:

   a. *HyperText Transfer Protocol* (HTTP). HTTP adalah *protocol* yang dipakai untuk mayoritas komunikasi *World Wide Web*. *Windows* 2003 menghadirkan *Internet Explorer* sebagai *client HTTP* dan *Internet Information Services* (IIS) sebagai *server* HTTP.

   b. *File Transfer Protocol* (FTP). FTP adalah suatu layanan Internet yang mentransfer file-file dari satu komputer ke komputer lain. *Internet Explorer* dan FTP utility yang termasuk command-line bertindak sebagai client FTP. IIS menyediakan *server* FTP.

   c. *Simple Mail Transfer Protocol* (SMTP). SMTP merupakan suatu *protocol* yang dipakai *server* mail untuk mentransfer e-mail. IIS dapat mengirimkan pesan-pesan dengan memakai *protocol* SMTP.

   d. *Telnet*. *Telnet* adalah suatu *protocol* yang menyamai terminal yang dapat dipakai untuk me-logon ke host jaringan yang jauh. *Telnet* menawarkan para pemakai suatu kapabilitas dalam mengoperasikan program- program secara jauh dan memudahkan administrasi yang jauh. *Telnet* secara praktis memang disediakan untuk semua sistem

operasi dan mengurangi integrasi dalam lingkungan jaringan yang heterogen. Windows 2003 memberikan baik *server* maupun *client Telnet*.

e. *Domain Name System* (DNS). DNS merupakan seperangkat *protocol* dan layanan pada suatu jaringan TCP/IP yang membolehkan para pemakai jaringan untuk mempergunakan nama-nama hierarki yang sudah dikenal ketika meletakkan host ketimbang harus mengingat dan memakai alamat IP-nya. DNS sangat banyak dipakai di Internet dan pada kebanyakan perusahaan pribadi dewasa ini. Saat Anda memakai *Web browser*, aplikasi *Telnet*, *utility* FTP, atau *utility* TCP/IP mirip lainnya di Internet, maka Anda mungkin sedang memakai sebuah server DNS. Windows 2003 menyediakan juga server DNS.

f. *Simple Network Management Protocol* (SNMP). SNMP memungkinkan untuk mengelola node jaringan seperti *server*, *workstation*, *router*, *bridge*, dan hub dari host sentral. SNMP dapat dipakai untuk mengonfigurasi *device* yang jauh, memantau unjuk kerja jaringan, mendeteksi kesalahan jaringan atau akses yang tidak cocok, dan mengaudit pemakaian jaringan.

2. *Network Application* API

TCP/IP *Microsoft* menyediakan dua interface bagi aplikasi-aplikasi jaringan untuk memakai layanan-layanan stack *protocol* TCP/IP:

a. *WinSock*. Implementasi Windows 2003 sangat banyak memakai *Socket* API (*application programming interface*). *Socket* API adalah mekanisme standar untuk mengakses layanan-layanan datagram dan session pada TCP/IP.

b. *NetBIOS*. Suatu API standar yang dipakai sebagai suatu mekanisme *inter-process communication* (IPC) di lingkungan Windows. Meskipun *NetBIOS* dapat dipakai untuk menyediakan koneksi standar ke protocol-protocol yang mendukung layanan-layanan penyampaian pesan dan penamaan *NetBIOS*, misalnya TCP/IP dan *NetBEUI*, namun

NetBIOS disertakan di dalam Windows 2003 terutama sekali untuk mendukung aplikasi-aplikasi peninggalan.

3. *Layer Transport*

Protocol-protocol *transport* menghadirkan sesi komunikasi di antara komputer-komputer dan menentukan tipe layanan transport entah berorientasi koneksi (TCP) ataupun berorientasi datagram yang tanpa koneksi (UDP). TCP menyajikan komunikasi berorientasi koneksi yang dapat diandalkan untuk aplikasi-aplikasi yang secara khusus mentransfer sejumlah besar data pada suatu waktu. TCP juga dipakai untuk aplikasi-aplikasi yang memerlukan pengakuan bagi data yang diterima. Namun demikian, UDP menyediakan komunikasi tanpa koneksi dan tidak menjamin mampu mengirimkan paket. Aplikasi-aplikasi yang memakai UDP secara khusus mentransfer sejumlah kecil data pada suatu waktu. Pengiriman data yang dapat diandalkan merupakan tanggung jawab aplikasi. *Layer transport* pada model DOD sesuai dengan *layer transport* pada model OSI.

4. *Layer Internet*

Protocol-protocol *internet* meng-encapsulate paket-paket menjadi datagram internet dan mengoperasikan semua algoritma routing yang diperlukan. Fungsi-fungsi routing yang dikerjakan *layer Internet* memang dibutuhkan untuk memungkinkan host meng-interoperate dengan jaringan-jaringan lain. *Layer Internet* sesuai dengan *Layer Network* pada model OSI. Lima protocol yang diimplementasikan pada layer ini:

a. *Address Resolution Protocol* (ARP), yang menentukan alamat hardware pada host.

b. *Reverse Address Resolution Protocol* (RARP), yang menyediakan resolusi alamat kebalikan pada host yang menerima.

c. *Internet Control Message Protocol* (ICMP), yang mengirimkan pesan-pesan kesalahan ke IP ketika terjadi problem.

d. *Internet Group Management Protocol* (IGMP), yang menginformasikan router tentang ketersediaan anggota-anggota pada grup multicast.

e. *Internet Protocol* (IP), yang mengalamatkan dan meroute paket-paket

5. *Layer Interface Network*

Model yang paling mendasar adalah *Layer Interface Network*. Masing-masing tipe *local area network* (LAN), tipe *metropolitan area network* (MAN), tipe WAN, dan tipe *dial-up*, misalnya *Ethernet*, *Token Ring*, *Fiber Distributed Data Interface* (FDDI), dan *ARCnet*, mempunyai persyaratan yang berbeda untuk pengkabelan, pembuatan sinyal, dan penyandian data. *Layer Interface Network* menetapkan persyaratan yang ekuivalen dengan *Layer Data Link* dan *Layer Physical* pada model OSI. *Layer Interface Network* bertanggung jawab untuk mengirimkan dan menerima frame, yang merupakan paket-paket informasi yang ditransmisikan pada suatu jaringan sebagai suatu unit tunggal. *Layer Interface Network* meletakkan frame pada jaringan dan mengeluarkan frame dari jaringan.

2.2.2.3 Pengalamatan IP

1. IPv4 *Address*

Agar memungkinkan bagi dua buah sistem untuk saling berkomunikasi, keduanya harus bisa saling mengidentifikasi dan mengetahui letak masing-masing sistem. Tiap *interface* atau *connection point* yang terdapat di peralatan yang terhubung ke jaringan memiliki alamat. Ini memungkinkan komputer lain untuk dapat menemukannya pada jaringan tempat peralatan tersebut berada. Kombinasi dari *network address* dan *host address* membentuk alamat unik untuk setiap peralatan yang terhubung ke jaringan. Tiap komputer (atau antarmuka dalam jaringan IP) yang berada dalam TCP/IP harus diberikan sebuah *identifier* unik atau bisa disebut sebagai IP *address* (Edwards et al, 2005, p38). IP

address ini beroperasi pada layer 3 dan memungkinkan sebuah komputer untuk menemukan komputer lainnya dalam sebuah network. Selain IP address, ada sebuah *physical address* yang unik dalam setiap peralatan yang dinamakan MAC *address*. MAC *address* ini dikeluarkan oleh perusahaan pembuat NIC pada peralatan tersebut dan ditanamkan pada NIC tersebut. MAC *address* beroperasi pada *layer* 2.

IP *Address* terdiri dari 32 bit angka biner (1 dan 0) yang terdiri dari *host numbers* dan *network numbers* (Goncalves, 1999, p4). Pada umumnya, dalam penulisan IP address dibagi empat bagian di mana masing-masing bagian terdiri atas 8 angka biner (disebut juga *octet*s) dan menggunakan titik sebagai pemisah. Contohnya adalah : 11000000.00010000.00010010.00000010. Ini dinamakan sebagai bentuk notasi biner. Sebenarnya, IP *address* lebih sering dan umum ditulis dalam bentuk empat kelompok angka desimal (0-255). Sebagai contoh : 192.16.18.2. Bentuk seperti ini dinamakan *dotted decimal*. IP *address* yang terdiri dari 32 bit angka dikenal sebagai IP versi 4 (IPv4).

Seluruh IP *address* terdiri atas dua bagian. Bagian pertama mengidentifikasi jaringan tempat sistem itu terhubung dan bagian kedua mengidentifikasikan sistem itu sendiri. Bagian pertama dapat disebut juga sebagai *network* ID dan bagian kedua disebut sebagai *host* ID (Edwards et al, 2005, p40). Tiap *octets* dapat berisi nilai antara 0-255. Ini berarti tiap satu *octet* dapat dipecah menjadi 256 *subgroups* kemudian dapat dipecah menjadi 256 *subgroups* lagi dengan masing-masing memilik 256 *address.* IP *address* dibagi dalam kelas-kelas untuk membedakan jaringan besar, sedang, dan kecil.

2. Kelas Kelas IP *Address*

Para pendiri Internet memutuskan untuk membagi kelas-kelas jaringan berdasarkan ukuran jaringan (Edwards et al, 2005, p41). Untuk mengakomodasi berbagai ukuran jaringan dan membantu dalam

pengelompokkan jaringan, IP *address* dapat dibagi menjadi lima kelas, yaitu A,B,C,D, dan E.

Kelas A diperuntukkan bagi jaringan yang sangat besar dengan jumlah host lebih dari 16 juta. Bit pertama dari alamat Kelas A selalu 0. Artinya nomor terkecil adalah 0000 0000 yaitu angka desimal 0 dan nomor terbesar yang mungkin adalah 0111 1111 yaitu angka desimal 127. Nomor 0 dan 127 tidak dapat digunakan sebagai alamat *network*. Jaringan 127.0.0.0 digunakan untuk *loopback testing* oleh *router* atau mesin lokal untuk mengirim paket ke dirinya sendiri.

Kelas B digunakan pada jaringan berukuran sedang ke besar. Kelas B menggunakan 2 *octets* pertama dari 4 *octets* yang ada sebagai alamat network. 2 bit pertama pada alamat Kelas B selalu 10. Artinya, nomor terkecil yang terdapat dalam Kelas B adalah 1000 0000 yaitu angka desimal 128 dan nomor terbesar yang mungkin adalah 1011 1111 yaitu angka desimal 191.

Kelas C adalah kelas alamat asli yang paling sering dipergunakan. Kelas C digunakan untuk jaringan kecil dengan jumlah *host* maksimal sebanyak 254 *hosts*. Alamat kelas C selalu dimulai dengan angka biner 110. Artinya nomor terkecil yang berada dalam Kelas C adalah 1100 0000 yaitu angka desimal 192. Dan nomor terbesar yang mungkin adalah 1101 1111 yaitu angka desimal 223.

Jika Kelas A,B, dan C umum digunakan untuk pengalamatan biasa, Kelas D dirancang untuk memungkinan *multicasting* dalam alamat IP. Alamat m*ulticast* adalah alamat unik yang mengarahkan paket dengan alamat tujuan tersebut ke grup IP *address* yang sudah didefinisikan sebelumnya. Kelas D berada dalam jangkauan 224.0.0.0 sampai dengan 239.255.255.255 dengan empat *bit* pertama selalu dimulai dengan 1110.

Kelas E yang berada dalam jangkauan alamat 240.0.0.0 sampai dengan 247.255.255.255 dicadangkan oleh *Internet Engineering and Task Force (IETF)* dan belum digunakan. Empat *bit* pertama dalam Kelas E selalu dimulai dengan 1111.

Pada Kelas A,B,dan C ini dikenal juga istilah *network mask*. *Network mask* ini dapat digunakan untuk mengenali di dalam kelas yang mana suatu IP *address* berada. Secara *default*, *network mask* dari kelas-kelas tersebut : Kelas A adalah 255.0.0.0 , Kelas B adalah 255.255.0.0 , dan Kelas C adalah 255.255.255.0.

3. Subnetting

Subnetting adalah salah satu metode untuk mengelola IP *addresses*. Metode pembagian alamat *network* ini mencegah terjadinya kekurangan akan IP *address* yang tersedia. Bagi jaringan yang kecil tidak terlalu dibutuhkan *subnetting*, namun pada jaringan yang besar diperlukan. Jaringan yang lebih kecil yang merupakan hasil pembagian dari jaringan yang lebih besar disebut juga dengan *subnets*. Beberapa komputer dalam sebuah *subnet* tidak dapat berkomunikasi dengan komputer pada *subnet* berbeda tanpa sebuah *router* (Odom, 2004, p275).

Dalam membagi jaringan besar ke dalam jaringan-jaringan yang lebih kecil, diperlukan adanya *subnet mask*. Dalam membagi-bagi jaringan diperlukan informasi mengenai berapa banyak jumlah *subnets* yang diinginkan juga berapa banyak jumlah *host* yang dikehendaki untuk berada dalam satu *subnet* atau jaringan.

Untuk membuat sebuah alamat *subnet*, beberapa *bit* dipinjam dari *host field* kemudian *bit* yang dipinjam ini dinamakan sebagai *subnet field*. Misal, sebuah jaringan 192.168.1.0 dengan *default network mask* Kelas C yaitu 255.255.255.0 dipinjam 4 *bit* dari *octet* terkahir *host portion*-nya sebagai *subnet field*. Maka akan menghasilkan jaringan yang lebih kecil dengan jumlah *host* yang memungkinkan 14. Untuk mendapatkan jumlah *host* tadi dapat menggunakan rumus 2n-2, di mana n adalah jumlah *bit* yang dipinjam. *Subnet mask* dari jaringan yang baru ini bukan lagi 255.255.255.0 melainkan 255.255.255.240. Hal ini terjadi karena *octet* terakhirnya berubah dari 0000 0000 menjadi 1111 0000 sesuai banyaknya *bit* yang dipinjam.

4. *Public* dan *Private* IP *Address*

Stabilitas *internet* sangat bergantung pada keunikan alamat jaringan yang digunakan secara *public.* Oleh karena itu diciptakanlah lembaga internasional bernama *Internet Assigned Number Authority* (IANA) yang mengatur persediaan IP *address* agar tidak terjadi duplikasi dalam pemakaian IP *address* secara *public.*

Public IP *address* sifatnya unik. IP *address* yang sama tidak boleh dimiliki oleh dua perangkat atau lebih yang terhubung ke jaringan *public*. Untuk mendapatkan *public* IP *address* bisa dilakukan melalui *Internet Service Provider* (ISP).

Private IP *address* dapat digunakan pada jaringan yang bersifat pribadi sebagai solusi atas semakin meningkatnya kebutuhan akan *public* IP *address*. Jaringan pribadi yang tidak terhubung ke *Internet* dapat menggunakan IP *address* apapun pada masing-masing *host*-nya asalkan masing-masing *host* tersebut memiliki alamat yang unik pada jaringan pribadi tersebut. Walau demikian, penggunaan IP *address* secara bebas pada jaringan pribadi dapat menimbulkan masalah ketika jaringan pribadi tersebut terhubung ke *Internet.*

RFC 1918 bertemakan "*Address Allocation for Private Internet*" yang membahas tentang penggunaan jaringan / operasional jaringan menggunakan TCP/IP mengemukakan permasalahan penggunaan *public* dan *private* yang harus dicermati berkenaan dengan *global address space* yang semakin berkurang setiap harinya. Berikut ini adalah set IP *private* yang direkomendasikan dalam RFC 1918, yang dapat digunakan dalam jaringan pribadi / internal.

Tabel 2.1 Rekomendasi IP *Private* dalam RFC 1918

Class	RFC 1918 *internal address range*
A	10.0.0.0 to 10.255.255.255

Class	RFC 1918 *internal address range*
B	172.16.0.0 to 172.31.255.255
C	192.168.0.0 to 192.168.255.255

5. Cara Mendapatkan IP *Address*

Ada beberapa cara yang dapat digunakan untuk mengkonfigurasi IP *address*.

a. Pengalamatan Statis

Pengalamatan statis sangat cocok diterapkan pada jaringan kecil yang jarang mengalami perubahan. Pencatatan *record* IP oleh seorang *network administrator* penting untuk dilakukan agar tidak terjadi duplikasi alamat IP. *Server* adalah salah satu perangkat yang biasanya diberikan alamat statis. Hal ini dimaksudkan untuk memudahkan akses pada layanan yang disediakan oleh *server* tersebut. Untuk *user* pada sistem operasi *Windows* 2000/NT, kita dapat membuka *path* :

Start → Control Panel → Network and Dial-up Connections → Local Area Connection → Internet Protocol (TCP/IP) → Properties → Use the following IP address, lalu isi dengan alamat IP yang diinginkan.

b. Pengalamatan dengan *Dynamic Host Configuration Protocol* (DHCP)

DHCP merupakan penerus dari BOOTP. Pengalamatan IP dengan DHCP sering disebut juga pengalamatan IP secara dinamis. Sebab, tidak seperti pada BOOTP, *network administrator* tidak perlu membuat profil individual pada tiap perangkat. Pengaturan yang diperlukan saat menggunakan DHCP hanyalah mendefinisikan jangkauan alamat IP pada DHCP *server*. Saat sebuah *host* terhubung secara *online* ia menghubungi DHCP server untuk meminta IP *address*. Lalu, DHCP server akan memilihkan satu alamat dan memesankan alamat tersebut bagi *host* yang memintanya.

DHCP juga memiliki kemampuan untuk menggunakan kembali IP *address* yang tadinya dipesan oleh sebuah *host* manakala *host* tersebut sudah tidak menggunakannya lagi. Sehingga IP *address* tadi dapat dipakai kembali bagi *host* lain yang memintanya. Hal ini berarti DHCP menawarkan rasio penggunaan IP *address one-to-many.* Sehingga sebuah alamat tersedia bagi siapa saja yang terhubung ke jaringan. Proses yang berjalan dalam mendapatkan IP *address* dengan DHCP sama dengan proses BOOTP yaitu dengan menggunakan *broadcast* IP *address* dan MAC *address.*

2.3 Keamanan Jaringan Komputer

Keamanan jaringan secara umum adalah computer yang terhubung ke jaringan, mempunyai ancaman keamanan yang lebih besar dibandingkan computer yang tidak terhubung ke jaringan. Namun keamanan komputer biasanya bertentangan dengan *network access*, dimana bila *network access* semakin mudah, maka *network security* semakin rawan, dan bila network semakin baik maka *network access* semakin tidak nyaman. Suatu *network* didesain sebagain komunikasi data dengan tujuan meningkatkan akses ke sistem komputer, sementara *security* didesain untuk mengontrol akses. Sebagai analogi, keamanan di rumah dilakukan dengan memberi kunci di pintu rumah, tidak dengan cara memblokir jalan di depan rumah. Hal seperti ini juga diterapkan pada *network security*. Keamanan dijaga untuk setiap host-host tertentu, tidak langsing pada networknya.

Langkah awal dalam mengembangkan rencana *network security* yang efektif adalah dengan mengenali ancaman yang mungkin dating. Dalam RFC 1244, *Site security Handbook*, dibedakan tiga tipe ancaman, yaitu :

1. Akses tidak sah oleh orang yang tidak memiliki wewenang.
2. Kesalahan informasi, segala masalah yang dapat menyebabkan diberikannya informasi yang penting atau *sensitive* kepada orang yang salah, yang seharusnya tidak boleh mendapatkan informasi tersebut.

3. Penolakan terhadap *service*, segala masalah mengenai *security* yang menyebabkan system mengganggu pekerjaan-pekerjaan yang produktif.

2.3.1 Tujuan Keamanan Jaringan Komputer

Pada dasarnya tujuan dari keamanan komputer yaitu :

1. *Confidentiality*, merupakan usaha untuk menjaga informasi dari orang yang tidak berhak mengakses. *Confidentiality* biasanya berhubungan dengan informasi yang diberikan ke pihak lain.
2. *Integrity*, Keaslian pesan yang dikirim melalui sebuah jaringan dan dapat dipastikan bahwa informasi yang dikirim tidak dimodifikasi oleh orang yang tidak berhak dalam perjalanan informasi tersebut.
3. *Availability*, Aspek *availability* atau ketersediaan berhubungan dengan ketersediaan informasi ketika dibutuhkan.

Tabel 2.2 CIA

	Definisi	***Tools***
Confidentiality	a. Penyembunyian informasi dan *resource* b. Sembunyikan adanya info dan *resource*	a. Enkripsi *b. Access control*
Integrity	*a.* Kepercayaan dari info dan sumber daya *(Authentication)* *b. Kebenaran data (data integrity)*	a. Pencegahan b. Deteksi
Availability	Kemampuan untuk menggunakan informasi dan sumber daya	*a. System design* *b. Statstical models of use*

2.3.2 Tipe Ancaman Keamana Sistem Komputer

Fungsi sistem komputer dapat digunakan sebagai dasar untuk menentukan model tipe ancaman dari suatu sistem komputer. Berdasarkan fungsi system

komputer sebagai penyedia informasi, ancaman terhadap sistem computer dikategorikan menjadi empat, yaitu:

1. *Interruption*, merupakan suatu ancaman terhadap *availability*, informasi atau data yang ada dalam sistem komputer dirusak, dihapus, sehingga jika dibutuhkan maka sudah tidak ada lagi .
2. *Interception*, merupakan ancaman terhadap kerahasiaan (*secrecy*). Informasi yang ada di dalam sistem disadap oleh orang yang tidak berhak.
3. *Modification*,merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil menyadap lalu-lintas informasi yang sedang dikirim lalu mengubahnya sesuai keinginan orang itu.
4. *Fabrication*, merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil meniru atau memalsukan suatu informasi sehingga orang yang menerima informasi tersebut menyangka informasi tersebut berasal dari orang yang dikehendaki oleh si penerima informasi tersebut.

2.3.3 Bentuk Ancaman Jaringan Komputer

Berikut beberapa ancaman yang umum ditemui pada jaringan nirkabel:

1. ***MAC Spoofing***

   Penyerang berusaha mendapatkan koneksi ke dalam jaringan dengan mengambil alamat NIC dari suatu perangkat komputer pada jaringan tersebut.

2. ***ARP Spoofing***

   Penyerang menangkap penyebaran paket ARP dari *access point* dan kemudian mengirimkan balasan ARP fiktif sehingga informasi perangkat dari penyerang akan terpetakan ke dalam tabel ARP untuk kemudian mendapatkan hak akses kedalam jaringan.

3. ***Man in the Middle Attack***

Metode serangan ini biasanya didahului dengan *ARP spoofing* kemudian penyerang menempatkan perangkat yang dimilikinya sebagai sebuah komputer fiktif yang akan terlihat resmi dari sisi *access point*.

4. ***Denial of Service***

   Metode serangan dengan mengirimkan paket data dalam jumlah yang sangat besar terhadap jaringan yang menjadi targetnya secara terus-menerus. Hal ini dapat mengganggu lalu-lintas data bahkan kerusakan sistem jaringan.

2.4 Konsep Dasar Algoritma

Pemrograman sudah menjadi kegiatan yang sangat penting di era teknologi informasi saat ini. Program yang berjalan di berbagai *device* seperti komputer (*personal computer*), *netbook*, *handheld*, web (berbasis internet) pada dasarnya tidak dibangun begitu saja, melainkan ada suatu proses yang menjadi suatu pola kerja dari program itu sendiri yaitu algoritma.

2.4.1 Sejarah Algoritma

Algoritma mempunyai sejarah yang panjang. Jika dilihat dari asal kata nya yaitu "algoritma", kata ini tidak muncul dalam kamus Webster pada tahun 1957.

Menurut Rinaldi Munir (2011:10), Para ahli bahasa menemukan kata algorism berasal dari nama cendikiawan muslim yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarijmi (Al-Khuwarijmi dibaca oleh orang Barat menjadi algorism) dalam bukunya yang berjudul Kitab Aljabar Wal-muqabala, yang artinya "Buku Pemugaran dan Pengurangan" (*The book of restoration and reduction*). Dari judul buku itu kita memperoleh kata "aljabar" (algebra). Perubahan dari kata algorism menjadi algorithm muncul karena kata algorism sering dikelirukan dengan arithmetic sehingga akhiran –sm berubah menjadi –thm.

Pada tahun 1950 algoritma yang lebih condong ke arah aritmatika terbukti dengan dipakainya kata algoritma tersebut dalam "*Algoritma Euclidean*" yaitu

algoritma yang mencari pembagi bersama terbesar (*Great Common Divisor*) diantara kedua bilangan. Dalam algoritma ini sangat membantu dalam mencari nilai *enciphering* pada algoritma RSA.

Kemudian dari hal tersebut, algoritma dikembangkan ke arah prosedur komputasi sehingga komputer dapat bekerja seperti yang diharapkan seperti saat ini.

2.4.2 Definisi Algoritma

Algoritma berusaha melakukan langkah-langkah seefisien mungkin untuk mencapai tujuan semaksimal mungkin. Algoritma sebenarnya implementasi dari kehidupan sehari-hari misalnya algoritma *stack* dan algoritma *queue* yang merupakan implementasi dari antrian dan tumpukan yang terjadi dalam aktifitas sehari-hari.

Menurut *Thomas H. Cormen* (2009:5), Algoritma adalah prosedur komputasi yang mengambil beberapa nilai atau kumpulan nilai sebagai input kemudian di proses sebagai output sehingga algoritma merupakan urutan langkah komputasi yang mengubah input menjadi output.

2.5 Konsep Dasar Kriptografi

Kriptografi menjadi salah satu unsur penting dalam dunia informasi. Hal ini terkait dengan semakin canggihnya teknologi dan kemudahan dalam komunikasi sehingga memberikan efek samping yaitu semakin mudahnya orang untuk mendapatkan informasi.

2.5.1 Sejarah Kriptografi

Kriptografi mempunyai sejarah yang sangat menarik dan panjang. Kriptografi sudah digunakan 4000 tahun yang lalu yang diperkenalkan oleh orang-orang Mesir untuk mengirimkan pesan kepasukan Militer yang berada dilapangan dan supaya pesan tersebut tidak dapat dibaca oleh pihak musuh jika kurir pembawa pesan tersebut tertangkap.

Pada zaman Romawi kuno, ketika Julius Caesar ingin mengirimkan satu pesan rahasia kepada seorang Jendral di medan perang. Pesan tersebut harus dikirimkan melalui kurir, tetapi karena pesan tersebut mengandung sifat rahasia maka *Julius Caesar* tidak ingin pesan tersebut dibuka di tengah jalan. Di sini *Julius Caesar* memikirkan bagaimana cara mengatasinya yaitu dengan cara mengacak pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapa pun kecuali hanya dapat dipahami oleh Jenderalnya saja. Tentu Sang Jenderal sudah diberitahu sebelumnya bagaimana cara membaca pesan yang teracak tersebut karena telah mengetahui kuncinya. Caesar mengganti semua susunan alphabet dari a, b, c yaitu a menjadi d, b menjadi e, c menjadi f dan seterusnya.

Pada perang dunia kedua, Jerman menggunakan enigma atau juga disebut dengan mesin rotor yang digunakan Hitler untuk mengirimkan pesan ketentaranya. Jerman sangat percaya pesan yang di kirim melalui Enigma tidak terpecahkan kode-kode enkripsinya. Tapi anggapan itu keliru, setelah bertahun-tahun sekutu dapat memecahkan kode-kode tersebut setelah mempelajarinya. Setelah Jerman mengetahui kode-kode tersebut terpecahkan, maka enigma yang digunakan pada perang dunia kedua mengalami beberapa kali perubahan.

Selama bertahun-tahun kriptografi menjadi bidang khusus yang hanya dipelajari oleh pihak militer, seperti agen Keamanan Nasional Amerika (National Security Agency), Uni Soviet, Inggris, Perancis, Israel dan Negara-negara lainnya yang telah membelanjakan miliaran dolar untuk mengamankan komunikasi mereka dari pihak luar, tapi mereka selalu mempelajari kode-kode rahasia Negara lain, dengan adanya persaingan ini maka kriptografi terus berkembang sesuai dengan perkembangan zaman.

Namun pada 30 tahun terakhir ini, kriptografi tidak hanya dimonopoli oleh pihak militer saja, hal yang sama juga dilakukan oleh individu- individu yang menginginkan pesan dan komunikasi mereka tidak di ketahui oleh pihak lain. Apalagi pada zaman sekarang ini persaingan yang begitu tinggi, mereka rela mengeluarkan sekian dolar hanya untuk menjaga privasi mereka.

2.5.2 Pengenalan Algoritma Kriptografi

Secara terminologi Algoritma merupakan urutan langkah-langkah logis untuk penyelseian masalah yang disusun secara sistematis. Menurut Ariyus (2006:36), Algoritma kriptografi merupakan langkah- langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut dengan melakukan pembangkitan kunci, enkripsi dan dekripsi.

Algoritma Kriptografi terdiri dari tiga fungsi dasar yaitu :

1. Kunci

   Kunci yang di pakai untuk melakukan enkripsi dan dekripsi, kunci terbagi dua bagian yaitu kunci publik (*public key*) dan kunci privat (*private key*).

   Keamanan dari kriptografi modern hanya dengan merahasiakan kunci yang dimiliki oleh orang lain tanpa harus merahasiakan algoritma itu sendiri

2. Enkripsi

   Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga kerahasiaannya. Pesan asli disebut *plaintext* yang dirubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa di artikan dengan cipher atau kode. Beda halnya dengan enkripsi, untuk merubah plaintext ke *ciphertext* kita menggunakan algoritma yang dapat mengkodekan data.

3. Dekripsi

   Dekripsi merupakan kebalikan dari enkripsi, pesan telah di enkripsi dikembalikan kebentuk asalnya (*plaintext*) disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi.

2.5.3 Macam Macam Algoritma Kriptografi

Menurut Ariyus (2006:44) Algoritma kriptografi terbagi menjadi tiga bagian berdasarkan kunci yang dipakainya yaitu :

1. Algoritma Simetri (menggunakan kunci yang sama saat enkripsi dan dekripsi)
2. Algoritma Asimetri (menggunakan kunci yang berbeda saat enkripsi dan dekripsi)
3. Fungsi Hash

2.5.3.1 Algoritma Simetris (Konvensional)

Algoritma simetri disebut juga sebagai algoritma konvensional adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya. Disebut konvensional karena algoritma yang biasa digunakan orang sejak berabad-abad yang lalu adalah jenis ini. Keamanan algoritma simetri tergantung pada kunci. Membocorkan kunci berarti bahwa orang lain dapat mengenkrip dan mendekrip pesan tersebut. Agar komunikasi tetap aman, kunci harus tetap dirahasiakan. Yang termasuk algoritma kunci simetri adalah OTP, DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5, Kasumi dan lain-lain.

2.5.3.2 Algoritma Asimetris

Algoritma asimetris di desain sedemikian rupa sehingga kunci yang digunakan untuk enkripsi berbeda daru kunci yang digunakan untuk dekripsi. Lebih jauh lagi, kunci dapat dihitung dari kunci enkripsi. Algoritma ini disebut algoritma kunci publik karena kunci enkripsi dapat dibuat publik sehingga orang lain dapat mengetahuinya, namun hanya orang tertentu dan sekaligus pemilik kunci dekripsi yang sekaligus dapat melakukan dekripsi pesan tersebut. Dalam hal ini kunci enkripsi sering disebut dengan kunci publik dan kunci dekripsi sering disebut dengan kunci privat. Kunci privat kadang-kadang sering disebut kunci rahasia.

Adapun yang termasuk ke dalam algoritma ini diantaranya adalah *Digital Signature Algorithm* (DSA), RSA (*Rivest Shamir Adleman*), *Diffe-Helman* (DH), *Ellitic Curve Cryptography* (ECC), Kriptografi Quantum dan lain-lain.

2.5.3.3 Fungsi Hash

Fungsi Hash sering disebut dengan fungsi Hash satu arah (*one-way function*), *message digest*, *fingerprint*, fungsi kompresi dan *message authentication code* (MAC), merupakan suatu fungsi dimana pesan yang sudah diubah menjadi message digest tidak dapat dikembalikan lagi menjadi pesan semula. Dua pesan berbeda akan menghasilkan nilai hash yang berbeda pula.

2.6 Konsep Dasar Algoritma RSA

RSA merupakan algoritma kriptografi kunci publik (*public key cryptography*) dengan penggunaan kunci yang berbeda pada proses enkripsi dan dekripsi. Algoritma RSA dijabarkan pada tahun 1997 oleh tiga orang : Ron Rivest, Adi Shamir dan Len Adleman dari MIT (*Massachusetts Institute of Technology*). Hurup RSA sendiri berasal dari inisial nama mereka (*Rivest-Shamir-Adleman*).

Algoritma RSA melakukan pemfaktoran bilangan yang sangat besar, oleh karena alasan tersebut RSA dianggap aman. Untuk membangkitkan kedua kunci, yang dipilih dua bilangan acak yang besar. Skema yang dikembangkan oleh Rivest, Shamir dan Adleman yang mengekspresikan bahwa plaintext dienkripsi menjadi blok-blok yang setiap blok memiliki nilai bilangan biner yang diberi simbol “n”, *plaintext* blok “M” dan ciphertext blok “C”. Untuk melakukan enkripsi pesan “M” dibagi ke dalam blok-blok numeric yang lebih kecil dari pada “n” (data biner dengan pangkat terbesar), jika bilangan prima yang panjangnya 200 digit dan dapat menambah beberapa bit 0 di kiri bilangan untuk menjaga agar pesan tetap kurang dari nilai “n”.

Menurut Ir. Rinaldi Munir, M.T (2004:2), rumus pembentukan algoritma RSA didasarkan pada persamaan matematika dan didasarkan pada teorema Euler sehingga didapat rumus untuk enkripsi. Adapun rumus untuk enkripsi yaitu :

$$C = M^e \bmod n$$

keterangan :

C : *ciphertext* (blok *plaintext* yang sudah dienkripsi)

M : *message* (blok pesan yang akan dienkripsi)

e : *enciphering*

n : nilai modulus

dan rumus untuk melakukan proses enkripsi yaitu :

$$M = C^d \bmod n$$

Keterangan:

C : *ciphertext* (blok *plaintext* yang sudah dienkripsi)

M : message (blok pesan yang akan dienkripsi)

d : *deciphering*

n: nilai modulus

2.7 Voice Over IP

Internet Telephony adalah pengantaran suara (dan mungkin juga pengantaran tipe data multimedia lainnya) secara real-time, antar dua atau lebih partisipan, melalui satu atau lebih jaringan, menggunakan protokol-protokol yang lazim digunakan dalam ber-'Internet', dan protokol-protokol lainnya yang diperlukan untuk mengontrol pengantaran tersebut.

International Engineering Consortium (IEC) mendefinisikan Internet Telephony sebagai berikut [INT04]:

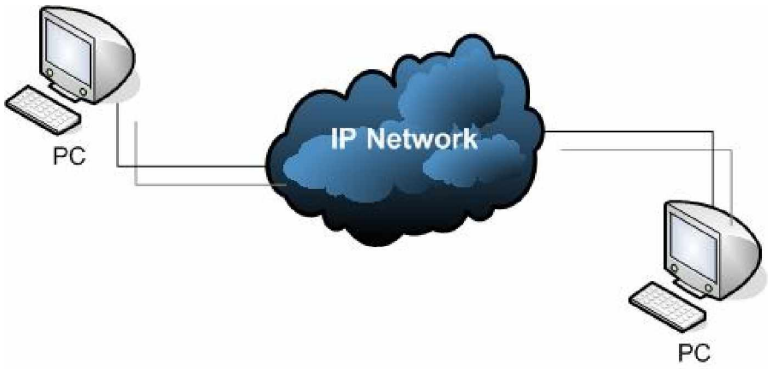
Internet telephony refers to communications services—voice, facsimile, and/or voice-messaging applications—that are transported via the Internet, rather than the public switched telephone network (PSTN). The basic steps involved in originating an Internet telephone call are conversion of the analog voice signal to digital format and compression/translation of the signal into Internet protocol

(IP) packets for transmission over the Internet; the process is reversed at the receiving end.

Penggunaan istilah Internet Telephony dan Voice-over-IP sering digunakan secara bergantian. Secara umum, kedua istilah tersebut mengacu kepada satu hal yang sama.

Skenario penggunaan Voice-over-IP pada umumnya diklasifikasikan menurut tipe terminal / perangkat komunikasi yang digunakan. Ada tiga buah skenario umum penggunaan Voice-over-IP, yaitu :

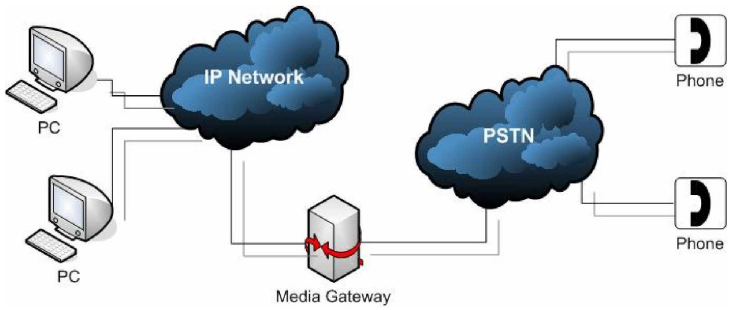
1. PC to PC



Gambar 2. 1 Skenario PC to PC

Pada skenario ini, komunikasi dilakukan antara dua PC atau lebih sebagai end-point, melalui jaringan IP. Interaksi suara dapat dilakukan melalui microphone dan speaker / headset ataupun handset telpon yang disambungkan ke modem.

2. *PC to Phone* dan *Phone to PC*

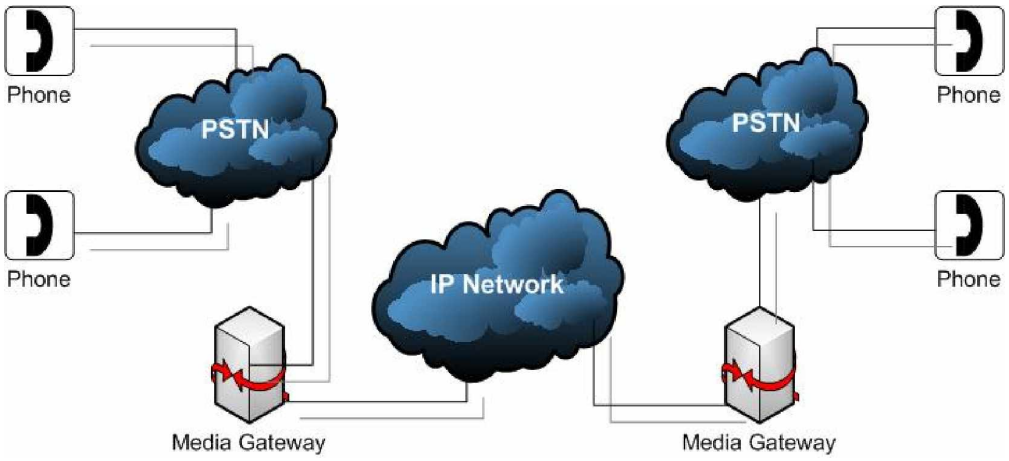


Gambar 2. 2 Skenario PC to Phone dan Phone to PC

Pada skenario ini, komunikasi dilakukan antara PC yang tersambung dengan jaringan IP, dengan telpon biasa yang tersambung dengan jaringan PSTN, dan vice versa. Hubungan antar dua jaringan yang berbeda dimungkinkan dengan menggunakan sebuah *Media Gateway* yang akan melakukan proses konversi sinyal analog menjadi paket (dan juga sebaliknya) yang dibutuhkan, dan juga melakukan proses translasi antara protokol signaling yang digunakan pada masing – masing jaringan IP dan jaringan PSTN. Jaringan PSTN yang dimaksud dapat berupa jaringan telpon publik, atau jaringan *Private Branch Exchange* (PBX).

3. *Phone to Phone*

Pada skenario ini, komunikasi dilakukan antara dua atau lebih telpon biasa yang masing – masing tersambung ke jaringan PSTN.



Gambar 2. 3 Skenario *Phone to Phone*

Masing – masing jaringan PSTN tersebut kemudian saling berkomunikasi melalui jaringan IP, dengan bantuan *Media Gateway* yang akan melakukan proses konversi sinyal analog menjadi paket (dan juga sebaliknya) yang dibutuhkan, dan juga melakukan proses translasi antara protokol signaling

yang digunakan pada masing – masing jaringan IP dan jaringan PSTN. Jaringan PSTN yang dimaksud dapat berupa jaringan telpon publik, atau jaringan *Private Branch Exchange* (PBX).

2.7.1 Protokol VoIP

Ada berbagai macam protokol yang digunakan dalam IP *telephony.* Berikut adalah protokol-protokol tersebut :

2.7.1.1 *H.323*

H.323 dikembangkan oleh ITU-T yang digunakan untuk mengirimkan *voice, video, data,* dan komunikasi *fax* melalui jaringan berbasis IP selama menjaga konektivitas dengan PSTN (Meggelen et al, 2005, p142). Bentuk jaringan berbasis paket yang termasuk di dalamnya antara lain *Internet*, *Internet Packet Exchange* (IPX-*based)*, *Local Area Network* (LAN), dan *Wide Area Network* (WAN). H.323 banyak digunakan dalam layanan – layanan yang berhubungan dengan multimedia seperti komunikasi suara (IP *Telephony*), komunikasi video dengan suara (*video telephony*), dan gabungan suara, video dan data. H.323 juga menjelaskan tentang penggunaan beberapa protokol ITU-T yang lain seperti :

A. H.225.0, terdiri atas dua protokol yaitu *Call Signaling* (digunakan untuk menghubungkan dua *endpoints* H.323) dan *Registration, Administration, and Status* (RAS) yang digunakan untuk menghubungkan antara *endpoint* dengan *gatekeeper*.
B. H.245, mengatur protokol komunikasi multimedia, menjelaskan prosedur yang digunakan untuk membuka dan menutup *channel* logis untuk *audio, video,* data, kemampuan melakukan pertukaran dan kontrol dan indikasi.
C. H.450, berisi rekomendasi tentang beberapa layanan tambahan
D. H.235, berisi tentang rekomendasi keamanan pada H.323 termasuk di dalamnya *signaling* dan keamanan medium.
E. H.239, menjelaskan tentang *dual stream* yang digunakan pada *teleconferencing.*

F. H.460, berisi rekomendasi yang menjelaskan tentang *extension* opsional yang dapat digunakan oleh suatu *endpoint* atau *gatekeeper.* Termasuk di dalamnya rekomendasi NAT/FW *traversal* dari H.460.17, H.460.18, dan H.460.19.

2.7.1.2 *Real Time Protocol* (RTP)

Real-time Transport Protocol atau RTP adalah sebuah protokol transport yang dirancang untuk digunakan oleh aplikasi yang membutuhkan sarana pengantaran data secara real-time, seperti data audio, data video atau data simulasi.

RTP mendefinisikan format paket standar yang digunakan untuk membawa data *audio* dan *video* melewati *Internet.* Ia berperan dalam mengkompensasi *jitter* dan *desequencing* yang terjadi pada jaringan IP. RTP banyak digunakan untuk sistem media *streaming, videoconferencing,* dan *push-to-talk.* RTP menggunakan *sequence numbers* untuk pengurutan paket data dan mendeteksi adanya paket yang hilang.

Salah satu keunggulan dari RTP adalah kemampuannya untuk mengantarkan data secara multicast. Kemampuan ini telah mengantarkan RTP sebagai protokol yang sering dipakai untuk aplikasi *multimedia conference*. Meskipun RTP dirancang untuk independen terhadap protokol pada layer yang lebih rendah, namun pada umumnya RTP dijalankan di atas UDP, agar dapat memanfaatkan kemampuan *multiplexing* dan *checksum service* dari UDP.

Definisi dari RTP tidak mencakup masalah resource reservation, dan juga tidak menjamin QoS dari layanan real-time yang menggunakan RTP. Kedua hal tersebut ditangani oleh protokol – protokol pada layer yang lebih rendah. RTP memiliki dua buah komponen: yang pertama adalah RTP itu sendiri, digunakan untuk membawa data melintasi network; yang kedua adalah RTP Control Protocol atau RTCP, berfungsi sebagai control protocol yang memonitor kualitas pelayanan dari RTP, dan mengumpulkan informasi mengenai partisipan dari sebuah session yang sedang berjalan.

2.7.1.2.1 Fitur yang Dimiliki

Berikut ini adalah beberapa fitur yang dimiliki oleh RTP:

a. *Sequencing Data* ,dikirim dalam bentuk paket. RTP menyediakan kemampuan untuk mendeteksi adanya paket yang hilang atau kesalahan urutan penerimaan paket pada receiver.
b. Intra-*media synchronization*. Sinkronisasi antara pengiriman paket dengan percakapan asli harus tetap terjaga. Pada sebuah percakapan, jeda waktu antara suatu ucapan dengan ucapan berikutnya harus bisa direkonstruksi oleh aplikasi dengan tepat, padahal biasanya tidak ada ada data yang dikirimkan pada saat jeda tersebut. RTP memiliki kemampuan untuk melakukan sinkronisasi tersebut.
c. *Inter-media synchronization*. Apabila ada beberapa jenis media (misalkan audio dan video) yang digunakan dalam sebuah percakapan, perlu adanya mekanisme untuk melakukan sinkronisasi antar media – media tersebut (proses ini biasa disebut sebagai lip-sync).
d. *Payload identification*. Ada kalanya aplikasi harus mengganti encoding terhadap data yang dikirimkan (atau disebut juga sebagai payload) di tengah percakapan, misalnya untuk menyesuaikan terhadap perubahan ketersediaan bandwidth, atau untuk menyesuaikan terhadap kemampuan user yang baru bergabung. RTP menyediakan mekanisme untuk meng-identifikasi encoding dari tiap – tiap paket.
e. *Multicast-friendly RTP* dan *RTCP* dirancang untuk men-support multicast. Kedua protokol tersebut dikembangkan dengan sedemikian rupa sehingga dapat beroperasi pada grup multicast kecil (seperti *three-way phone call*) maupun pada grup yang sangat besar (seperti *event broadcasting*).
f. *Media independent RTP* menyediakan layanan generik untuk komunikasi multimedia secara real-time, seperti audio dan video. Tidak ada pembatasan terhadap jenis codec yang harus digunakan untuk komunikasi multimedia tersebut.

g. *Mixer* dan *translator*. *Mixer* adalah alat yang menggabungkan media dari beberapa user menjadi satu media stream baru. *Translator* adalah alat yang mengubah format atau *encoding* dari sebuah *media stream*. *Translator* berguna untuk menurunkan kebutuhan *bandwidth* dari sebuah *stream*, tanpa mengharuskan sender untuk menurunkan *bitrate*-nya. *Mixer* membatasi besar bandwidth yang dibutuhkan apabila beberapa sender mengirimkan data secara bersamaan. Proses ini memenuhi fungsi sebagai *conference bridge*. RTP menyediakan support bagi *mixer* dan *translator*.
h. *QoS feedback* RTCP memungkinkan *receiver* untuk mengirimkan *feedback* kepada semua anggota dari grup mengenai kualitas penerimaan. Sender dapat menggunakan informasi ini untuk menyesuaikan data rate pengiriman, sementara anggota grup lainnya dapat menggunakan informasi tersebut untuk mencari tahu apakah masalah QoS yang dihadapi bersifat lokal atau mencakup keseluruhan jaringan. Pengamat dapat menggunakan informasi tersebut sebagai dasar dari manajemen *quality-of-service*.
i. *Loose session control* . Melalui RTCP, partisipan dapat saling bertukar informasi identifikasi, seperti nama, email, nomer telpon, dan pesan singkat.
j. *Encryption Media stream* RTP dapat di-enkripsi menggunakan key yang bisa dipertukarkan melalui metode non-RTP, seperti melalui SIP atau SDP.

2.7.1.3 *Real-Time Control Protocol (RTCP)*

RTCP merupakan protokol yang digunakan bersamaan dengan RTP dalam hal penyampaian dan penyusunan data multimedia namun tidak ikut serta dalam proses pengirimannya sendiri. Fungsi utamanya ialah memberikan *feedback* atas QoS yang diberikan oleh RTP. Dua komponen penting pada paket RTCP ialah *sender report* yang berisi informasi banyaknya data yang dikirimkan dan yang kedua adalah *receiver report* (dikirimkan oleh penerima panggilan) yang berisi informasi mengenai jumlah paket yang hilang selama sesi percakapan, dan *delay* sejak pengiriman *sender report* yang terakhir.

2.7.1.4 *Resource Reservation Protocol (RSVP)*

RSVP adalah protokol *layer transport* dan dirancang untuk menyediakan *resource* atau sumber daya jaringan (Goncalves, 1999, p40). Ia merupakan *signalling protocol* tambahan pada yang mempengaruhi QoS. RSVP biasanya digunakan untuk menyediakan *bandwidth* agar transmisi data suara tidak mengalami *delay* ataupun kerusakan saat mencapai tujuan. Cara kerjanya ialah dengan mengirimkan *request* pada setiap *node* dalam jaringan yang akan dilakukan pengiriman data *stream,* lalu pada untuk *node* tadi, RSVP membuat *resource reservation* untuk pengiriman data.

2.7.1.5 *Session Initiation Protocol* (SIP)

Session Initiation Protocol atau SIP adalah sebuah protokol signaling yang dikembangkan oleh IETF; berfungsi untuk menciptakan, mengatur, dan mengakhiri session antar *Internet end-points*. Yang dimaksud dengan *session* adalah pertukaran data antar *end-point* yang terkait, baik data teks, audio, video, ataupun jenis data lainnya.

SIP dibuat berdasarkan beberapa protokol IETF lainnya yang telah lebih dahulu populer seperti HTTP dan SMTP (*Simple Mail Transfer Protocol*). SIP banyak menggunakan *syntax* dan semantik dari HTTP, termasuk arsitektur *response code*, beberapa *message header*, dan operasi – operasi dasar. Seperti kedua protokol tersebut, SIP berbasis teks dan memiliki arsitektur *client-server*.

SIP dirancang mengikuti Internet model. SIP memiliki orientasi *end-to-end*, dalam artian semua logic pengoperasian disimpan pada *end device* (kecuali operasi routing dari SIP *message*).

SIP bukan merupakan sebuah *general purpose protocol*. Agar sebuah komunikasi (*audio*, *video*, atau *text*) dapat berlangsung, dibutuhkan protokol lainnya, antara lain RTP (untuk mengantarkan data multimedia), RTSP (untuk mengontrol *streaming session*), dan SDP (untuk mendeskripsikan session).

SIP dapat menggunakan protokol *transport* apa saja. Meski pada umumnya SIP menggunakan TCP atau UDP sebagai protokol *transport*-nya, namun secara

teoritis SIP dapat dijalankan di atas IPX, *Frame Relay*, ATM, ataupun X.25. SIP memiliki mekanisme sendiri untuk menjaga reliability, yang membuatnya dapat menggunakan protokol yang tidak reliable, seperti UDP.

2.7.1.5.1 *Addressing*

Dalam meng-identifikasi user, SIP menggunakan identifier berupa *Uniform Resource Identifier* (URI) yang disebut SIP URI. Bentuk dari SIP URI tersebut mirip dengan bentuk alamat email, yaitu memiliki sebuah username dan hostname. Bentuk dari SIP URI tersebut adalah ”sip:user@domain”, ”sip:user@host”, ”sip:user@ip_address”, atau ” sip:phone-number@gateway”. URI dengan bentuk ”sip:phone-number@gateway” ditujukan untuk nomer telpon PSTN yang dapat dihubungi melalui gateway yang disebutkan.

2.7.1.5.2 *SIP Network Elements*

Sebuah sistem SIP memiliki empat buat komponen dasar, yaitu *user agent*, *proxy server*, *registrar server*, dan *redirect server*. Komponen – komponen tersebut merupakan *logical entity*. Sebuah implementasi SIP dapat menggabungkan komponen – komponen tersebut ke dalam satu aplikasi yang terintegrasi.

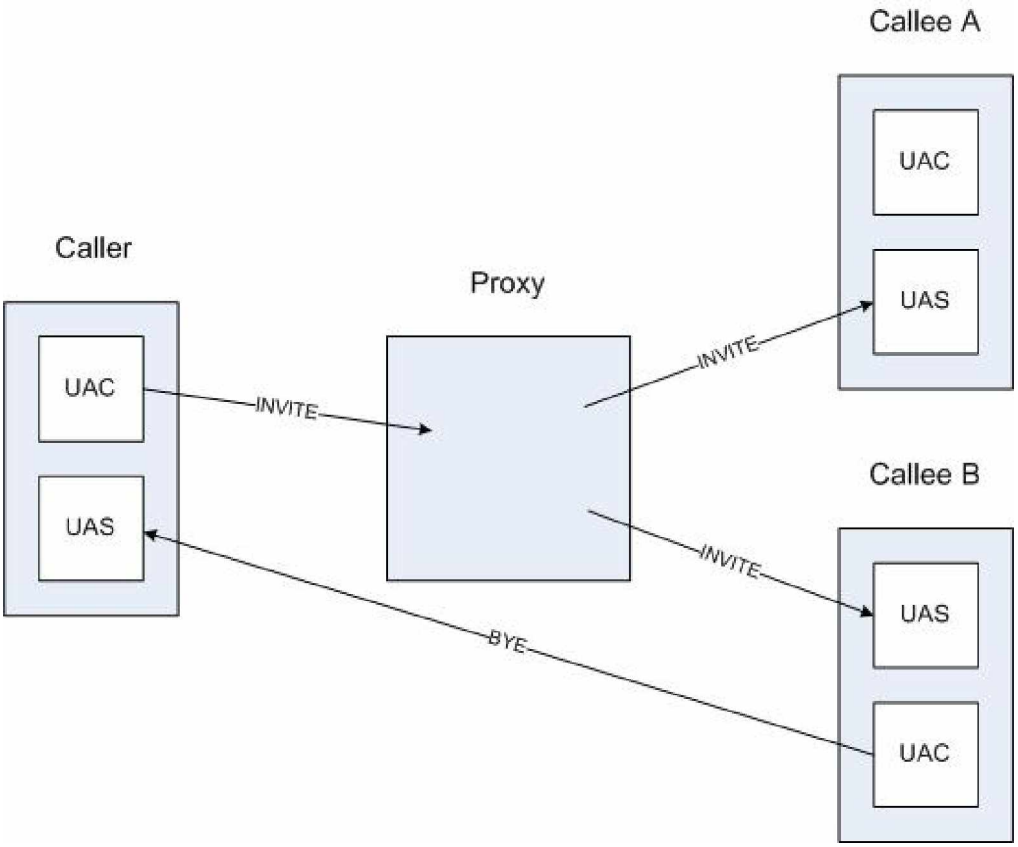
2.7.1.5.2.1 *User Agent*

User agent adalah sebutan bagi Internet end-points yang menjadi asal dan tujuan dari sebuah SIP *request*.

User agent biasanya berbentuk sebuah aplikasi yang berada pada komputer user. Namun, user agent dapat direpresentasikan dalam bentuk – bentuk lainnya, seperti berupa telpon selular, PSTN *gateway*, PDA, dan lain sebagainya.

Berdasarkan fungsionalitasnya, sebuah user agent dapat dibagi dua menjadi *User Agent Server* (UAS) dan *User Agent Client* (UAC). UAC adalah bagian dari *user agent* yang mengirim *request* dan menerima *response*, sedangkan UAS adalah bagian dari *user agent* yang menerima *request* dan mengirim *response*. UAS dan UAC merupakan *logical entity*. Pada umumnya, sebuah *user agent*

memiliki baik fungsi UAS maupun fungsi UAC.



Gambar 2. 4 *User Agent*

Gambar 2.4 menunjukkan tiga buah *user agent* dan sebuah *proxy server*. Setiap *user agent* memiliki komponen UAC dan UAS. Pada saat *caller* meng-inisiasikan panggilan, maka UAC dari *caller* mengirim SIP *message INVITE* ke *proxy server*, yang kemudian melanjutkan request tersebut ke callee A dan callee B. Komponen UAS dari *user agent* masing – masing *callee* menerima *INVITE* tersebut, dan mengirimkan *response* yang diperlukan untuk memulai sebuah panggilan. Pada saat callee B ingin mengakhiri panggilan, maka UAC dari *callee* B akan mengirimkan SIP *message* BYE ke *caller* yang diterima oleh komponen UAS-nya.

2.7.1.5.2.2 *Proxy Server*

Proxy server (atau SIP *proxy*) adalah entity yang memiliki peran cukup

penting di dalam sebuah sistem SIP. Fungsi utama dari sebuah SIP *proxy* adalah untuk men-route sebuah SIP request dari pengirim ke penerima, dan men-route response dari request tersebut dari penerima ke pengirim. SIP *proxy* menerima sebuah SIP *request*, lalu menentukan ke server mana *request* tersebut akan di-*forward*, dan kemudian men-*forward request* itu (biasanya setelah melakukan beberapa perubahan di *header*-nya). SIP *proxy* tidak bisa mengetahui apakah *next-hop* dari request akan berupa sebuah SIP *proxy* juga, sebuah *redirect server*, atau sebuah UAS. Oleh karena itu, sebuah request biasanya dapat men-traverse beberapa buah SIP *proxy* hingga akhirnya mencapai SIP *proxy* yang mengetahui lokasi aktual dari penerima, dimana SIP *proxy* tersebut kemudian akan men-*forward request* tersebut langsung ke *user agent* dari penerima. *Response* dari *request* akan melalui path yang sama dengan yang dilalui oleh *request*, namun dengan urutan yang terbalik.

Selain melakukan fungsi routing, SIP *proxy* juga dapat memiliki fungsi – fungsi lainnya, antara lain *authentication*, *accounting*, dan lain sebagainya. Ada dua jenis SIP *proxy*, yaitu *stateleess* dan *stateful*.

2.7.1.5.2.3 *Registrar Server*

Registrar server adalah sebuah SIP *network entity* yang bertugas untuk menerima *registration* dari user, dan kemudian meng-ekstrak informasi lokasi sekarang (*current location*) dari user (*IP address*, *port*, *username*, dsb.) dan menyimpan informasi tersebut di sebuah *location database*. Kegunaan dari location database adalah untuk memetakan misalnya ”sip:cica@ui.edu” ke dalam bentuk ”sip:cica@152.118.31.85:5060”, dimana pada waktu yang berbeda, hasil pemetaan dari sebuah user bisa berbeda – beda, tergantung dari lokasi sekarang user.

Location database digunakan oleh *proxy server*. Pada saat *proxy server* menerima INVITE *request* untuk ”sip:cica@ui.edu”, *proxy* akan mencari *username* tersebut di *location database*, dan kemudian akan mendapatkan ”sip:cica@152.118.31.85:5060” sebagai hasil pemetaannya. *Proxy server* lalu mengirimkan INVITE *request* ke alamat tersebut.

Karena keterkaitannya yang sangat erat dengan *proxy server*, *registrar server* biasanya berada di dalam aplikasi dan / atau mesin yang sama dengan *proxy*.

Setiap *registration* memiliki masa berlaku yang terbatas. *Field* '*expires'* pada *header* atau *parameter* '*expires'* di *field* '*Contact'* pada header menentukan berapa lama sebuah *registration* tetap valid. *User agent* harus me-*refresh registration* sebelum masa berlaku habis jika tidak ingin *registration* tersebut kadaluarsa dan user menjadi tidak dapat dihubungi.

2.7.1.5.2.4 *Redirect Server*

Sesuai dengan namanya, *redirect server* adalah sebuah SIP network entity yang bertugas untuk me-redirect SIP *request* yang diterimanya ke suatu tujuan baru. Cara kerjanya adalah sebagai berikut. Pertama – tama, *redirect server* menerima SIP *request*. Setelah itu *redirect server* mencari *username* dari *callee* di dalam *location database* (yang dibuat oleh *registrar*), lalu men-*generate* sebuah list yang berisi satu atau lebih lokasi sekarang dari user yang dipanggil, dan mengarahkan *user agent caller* ke lokasi – lokasi tersebut menggunakan 3xx *response*. *User agent caller* kemudian menerima informasi tersebut dan mengirimkan request berikutnya langsung ke lokasi – lokasi yang dimaksud.

2.7.1.5.2.5 SIP Messages

Berkomunikasi menggunakan SIP (atau yang sering disebut sebagai *proses signaling*) dilakukan dengan cara saling bertukar message antar *user agent*. SIP *message* memiliki dua jenis, yaitu *request*, yang dikirimkan oleh client ke *server*, atau *response*, yang dikirimkan oleh *server* ke *client*.

Sebuah SIP message terdiri dari (secara berurutan) start-line, message *header*, dan *message body*. *Start-line* menunjukkan identitas/ enis dari *message* tersebut, apakah merupakan jenis *request* atau *response*. *Message header* berisikan informasi mengenai panggilan, alamat (baik asal maupun tujuan), fitur dari protokol, dan informasi lainnya. Message body bisa berisi apa saja. Isi dari message body dideskripsikan menggunakan *Multipurpose*

Internet Mail Extension (MIME). Berikut contoh dari sebuah SIP message:

Tabel 2. 1 *SIP Header*

INVITE sip:	adhit@uab.edu SIP/2.0
Via:	SIP/2.0/UDP 152.118.31.85:5040;rport
Max-Forwards:	10 From: "Cica" <sip:cica@ui.edu>;tag=76ff7a07-c091-4192-84a0- d56e91fe104f
To:	<sip:adhit@uab.edu>
Call-ID:	d10815e0-bf17-4afa-8412-d9130a793d96@152.118.31.85
CSeq:	2 INVITE
Contact:	<sip:cica@152.118.31.85:9315>
User-Agent:	Windows RTC/1.0
Content-Type:	application/sdp
Content-Length:	451
v=0 o=jku2 0 0 IN IP4 152.118.31.85 s=session c=IN IP4 152.118.31.85 b=CT:1000 t=0 0 m=audio 54742 RTP/AVP 97 a=rtpmap:97 red/8000 a=rtpmap:111 SIREN/16000 a=fmtp:111 bitrate=16000 a=rtpmap:112 G7221/16000 a=fmtp:112 bitrate=24000 a=rtpmap:6 DVI4/16000 a=rtpmap:0 PCMU/8000 a=rtpmap:4 G723/8000 a=rtpmap: 3 GSM/8000 a=rtpmap:101 telephone-event/8000 a=fmtp:101 0-16	

Baris pertama dari *message* adalah yang disebut sebagai *start-line*; dalam contoh di atas, *start-line* memberitahu bahwa *message* merupakan sebuah request ber- tipe INVITE dengan *request* URI "adhit@uab.edu". *Request* URI umumnya memiliki *value* yang sama dengan *field* 'To' pada *header*.

Field '*Via'* pada *header* mencatat *path* dari perjalanan yang telah dilalui oleh *message* pada suatu saat. Sebuah SIP *message* dapat memiliki satu atau lebih *field* '*Via'*, tergantung dari panjangnya perjalanan yang telah ditempuh oleh *message* tersebut. *Response* dari sebuah *request* akan dapat sampai ke pengirim *request* melalui *path* yang sama dengan *request* dengan menggunakan informasi yang terdapat pada *field* '*Via'*. Pada contoh di atas, hanya terdapat satu *field* '*Via'* yang ditulis oleh *user agent* dari *caller*. Dari *field* tersebut, dapat kita ketahui bahwa *user agent* dari *caller* memiliki IP *address* 152.118.31.85 dan *port* 5040.

Field '*From'* dan '*To'* membawa informasi identitas dari *caller* dan *callee*. Identitas direpresentasikan oleh sebuah URI dan display name yang optional. *Field* '*From'* dan '*To'* mengandung *parameter* 'tag' yang berfungsi sebagai dialog *identifier*

Field '*Call-ID'* adalah sebuah dialog *identifier*. Selain itu, *field* '*Call-ID'* berfungsi sebagai *identifier* bagi message – message yang dimiliki oleh sebuah panggilan tertentu. Message – message tersebut akan memiliki value 'Call-ID' yang sama.

Kombinasi dari '*To tag'*, '*From tag'*, dan '*Call-ID'* mengidentifikasikan sebuah relasi *peer-to-peer* antara caller dan callee; biasanya disebut sebagai dialog.

Field '*CSeq'* berfungsi sebagai penanda urutan dari *message*. Karena message dapat dikirimkan melalui protokol transport yang tidak reliable dan dapat mengacaukan urutan message, dibutuhkan sebuah nomer urut yang dapat membantu untuk mengidentifikasi *retransmission* dan message yang sampainya tidak berurutan.

Field '*Contact'* berisi IP *address* dan *port number* yang merepresentasikan suatu alamat dimana *caller* dapat dihubungi secara langsung oleh *callee*.

Field '*Max-Forwards'* digunakan untuk membatasi jumlah hop yang dapat dilakukan oleh sebuah *request* untuk dapat sampai ke tujuannya. *Field* ini terdiri dari sebuah *integer* yang berkurang nilainya untuk setiap hop yang dilakukan.

Field '*Content-Type'* berisi deskripsi dari *message body*. *Field* '*Content-Length'* berisi besarnya *message body* dalam *ukuran byte*.

Message header dipisahkan dengan *message body* oleh sebuah baris kosong. *Message body* berisi deskripsi dari *session*, seperti tipe dari *codec* yang digunakan, *bitrate*, atau *sampling rate*, dan dideskripsikan menggunakan SDP.

2.7.1.5.2.6 *SIP Request*

SIP *request* adalah SIP *message* yang dikirimkan oleh UAC ke UAS dengan maksud untuk melakukan sebuah operasi tertentu. Ada banyak jenis dari SIP *request*. Beberapa yang penting antara lain [RFC05e]:

A. *INVITE*, digunakan untuk memulai sebuah panggilan / *session*.
B. *ACK*, digunakan untuk memberi konfirmasi terhadap diterimanya sebuah *final response*.
C. *BYE*, digunakan untuk mengakhiri sebuah panggilan / *session*.
D. *CANCEL*, digunakan untuk membatalkan sebuah *request* yang belum mendapatkan final *response*.
E. *OPTIONS*, digunakan untuk meminta informasi mengenai fitur – fitur yang dimiliki oleh *callee*.
F. *REGISTER*, digunakan untuk melakukan registration ke *registrar server*.

2.7.1.5.2.7 *SIP Response*

SIP response adalah *SIP message* yang dikirimkan oleh UAS ke UAC dengan maksud untuk mengindikasikan status dari sebuah request

yang sebelumnya telah dikirimkan oleh UAC ke UAS. Ada dua jenis *response*, dibedakan menurut pengaruhnya terhadap request yang bersangkutan, yaitu [RFC05b]:

a. *Provisional response* – yang digunakan oleh UAS untuk memberitahu penerimanya mengenai sedang berlangsungnya pemrosesan dari *request* yang bersangkutan – merupakan jenis *response* yang tidak mengakhiri sebuah SIP *transaction*. *Response* kelas 1xx adalah yang termasuk ke dalam jenis ini.
b. *Final response*, adalah *response* yang menginformasikan hasil pemrosesan dari sebuah *request*, dan mengakhiri sebuah SIP *transaction*. *Response* kelas 2xx, 3xx, 4xx, 5xx, dan 6xx adalah yang termasuk ke dalam jenis ini.

   Ada enam kelas dari SIP *response*, yaitu [JAN04]:

   a. 1xx *Provisional.* Kelas 1xx disebut sebagai *provisional response*. Pengirim request harus menghentikan melakukan *retransmission* terhadap request tersebut jika menerima *provisional response*. Sebagai contoh penggunaan, SIP *proxy* mengirimkan 100 *Trying response* ketika mulai memproses SIP *request INVITE*.
   b. 2xx *Successful*. Kelas ini mengindikasikan kepada penerimanya bahwa *request* yang bersangkutan berhasil diproses oleh UAS
   c. 3xx *Redirection Response*. Kelas 3xx memberikan informasi mengenai lokasi baru dari user yang dituju. Pengiriman *response* kelas ini biasanya menandakan bahwa *user* tidak dapat dihubungi pada alamat yang terdapat pada *Request* URI.
   d. 4xx *Request Failure Response* . Kelas 4xx menandakan adanya kegagalan pemrosesan *request* oleh UAS yang diakibatkan oleh kesalahan pada sisi *caller*. Kesalahan biasanya terdapat pada request yang dikirimkan. Sebagai contoh penggunaan, SIP *proxy* mengirimkan *404 Not Found response* jika user yang dituju tidak ditemukan pada domain yang terdapat pada *Request* URI.

e. 5xx *Server Failure.* Kelas 5xx menandakan adanya kegagalan pemrosesan request oleh UAS yang diakibatkan oleh kesalahan pada sisi *server*. *Caller* disarankan untuk mencoba mengirimkan request di lain waktu.

f. 6xx *Global Failure.* Kelas 6xx memberikan informasi bahwa request yang bersangkutan tidak dapat dipenuhi di server manapun. Server biasanya mengirimkan *response* ini jika memiliki informasi definitif mengenai user yang dituju. Sebagai contoh penggunaan, SIP *proxy* mengirimkan *603 Decline response* jika user menolak untuk berpartisipasi dalam sebuah *session*.

Berikut adalah contoh dari SIP *response code*:

Tabel 2. 2 *SIP Response Code*

Provisional	*Succesful*	*Redirection*
100 Trying *180 Ringing* *181 Call Is Being Forwarded* *182 Queued* *183 Session Progress*	*200 OK*	*300 Multiple Choices* *301 Moved Permanently* *302 Moved Temporarily* *380 Alternative Service*
Request Failure	*Server Failure*	*Global Failure*
400 Bad Request *401 Unauthorized* *403 Forbidden* *404 Not Found* *405 Method Not Allowed* *415 Unsupported Media Type* *420 Bad Extension* *486 Busy Here*	*500 Server Internal Error* *501 Not Implemented* *502 Bad Gateway* *503 Service Unavailable* *504 Server Time-out*	*600 Busy Everywhere* *603 Decline* *604 Does Not Exist Anywhere* *606 Not Acceptable*

2.7.1.6 SRTP

SRTP adalah sebuah protocol yang dikembangkan oleh kelompok kecil yang berasal dari CISCO dan Ericson dan dipublikasikan pada Maret 2004. SRTP –

Secure Real-time Transport Protocol adalah pengembangan dari *protocol* RTP yang menyediakan fitur kerahasiaan melalui fasilitas enkripsi, autentikasi dan integritas pesan, dan relay protection untuk RTP traffic dan untuk mengontrol *traffic* untuk RTP. SRTP berfungsi untuk menjamin pengamanan pengiriman data khususnya multimedia file.

SRTP menggunakan Algoritma AES sebagai metode enkripsi dalam pengiriman data. Pada aplikasinya SRTP memiliki 2 buah mode, yaitu *Segmented Integer Counter*, dan *AES* di *f8-mode*. Selain itu SRTP juga dapat berjalan dengan mode null cipher. Mode ini adalah mode dimana pengiriman data tidak dilindungi dengan algoritma enkripsi.

SRTP pada kenyataannya hanya mengenkripsi payload (Audio atau video) untuk kerahasiaan. Algoritma Autentikasi melindungi integritas dari seluruh paket RTP Packet. SRTP MKI (*Master Key Identification*) akan mengidentifikasikan kunci master mana yang digunakan untuk mendapatkan session keys yang sekarang ini digunakkan dalam proses enkripsi dan dekripsi. Walau kadang tidak digunakkan, sebuah MKI yang *typical* berukuran 4 byte dan digunakkan di system yang membutuhkan *multiple key exchange*.

SRTP adalah sebuah *protocol* keamanan internet yang efisien dan singkat, berjalan dengan baik, dan telah tercapai layanan *interoperability* yang baik. Namun, SRTP kekurangan *protocol* untuk bertukar kunci yang lebih diterima.

Setiap SRTP *stream* membutuhkan baik pengirim amupun penerima untuk menjaga informasi status *cryptographic*, yang bernama "*cryptograpic context*". SRTP membutuhkan dua buah jenis kunci yaitu *master key* dan *session key*. *Session key* artinya kunci yang secara langsung digunakan dalam *cryptographic transform*. *Master key* adalah sebuah bilangan *random bit string*.

2.7.2 Komponen dari *Voice-over-IP*

Voice-over-IP bukanlah sebuah protokol atau servis/layanan yang berdiri sendiri, melainkan istilah yang menjelaskan sebuah sistem yang menggabungkan

penggunaan berbagai macam teknologi dan protokol untuk memberikan layanan komunikasi suara. Teknologi dan protokol tersebut dapat dikelompokkan menjadi empat komponen utama (*network transport*, *media transport*, *signaling*, *directory services*), dan dua komponen pendukung (*media encoding*, *gateway control*).

2.7.2.1 *Network Transport*

Voice-over-IP menggunakan protokol – protokol jaringan yang juga umum digunakan oleh layanan berbasis Internet lainnya, seperti email atau web. Protokol yang digunakan tersebut diantaranya adalah IP sebagai *network layer*, TCP dan UDP sebagai *transport layer*, dan DNS untuk name-to-address mapping.

Untuk mendapatkan jaminan kualitas suara yang baik, maka VoIP bisa memanfaatkan fitur *Quality of Service* (QoS) dari sebuah jaringan. QoS itu sendiri adalah kemampuan sebuah jaringan untuk memberikan prioritas layanan terhadap network traffic tertentu, relatif terhadap network traffic lainnya. Pada VoIP, fitur QoS yang sering dimanfaatkan antara lain bandwidth *reservation*, *traffic priority*, dan *smart routing*.

2.7.2.2 *Media Transport*

Komponen media transport berfungsi untuk mengantarkan data multimedia dari satu *end-system* ke *end-system* lainnya. Data yang dibawa bisa berupa data audio, video, ataupun data multimedia lainnya.

Implementasi VoIP pada umumnya menggunakan *Real-time Transport Protocol* (RTP) sebagai protokol media transport pilihan.

2.7.2.3 *Signaling*

Komponen *signaling* berfungsi untuk menciptakan, mengatur, dan mengakhiri session antara dua end-system atau lebih. Yang dimaksud dengan session adalah pertukaran data antar *end-system* yang terkait, baik *data teks*, audio, video, ataupun jenis data lainnya.

Dua protokol signaling yang paling populer pada saat ini adalah *Session Initiation Protocol* (SIP) dan H.323.

2.7.2.3.1 *Dual Tone Multi Frequency (DTMF)*

DTMF adalah sebuah *signalling* yang menggunakan rentang frekuensi suara tertentu kepada pusat *call switching*. DTMF telah distandarisasi oleh ITU-T Rekomendasi Q.23. Telepon yang memiliki kemampuan untuk mengirim sinyal DTMF (*tone dialing*) disebut dengan Touch-Tone. Selain dalam teknologi telepon, aplikasi DTMF juga digunakan dalam menandakan awal dan akhir dari iklan komersial pada siaran televisi kabel di Amerika Serikat dan di negara lain.

Pada tombol telepon, DTMF disusun dalam matriks 4x4, di mana setiap tombol akan mengirim sinyal DTMF sebanyak dua frekuensi. Baris pada matriks ini merepresentasikan frekuensi rendah, sementara kolom untuk frekuensi tinggi. Apabila sebuah tombol pada telepon ditekan, maka akan dihasilkan sebuah bunyi sinusoidal dari dua jenis frekuensi tersebut. Bunyi sinusoidal ini akan diterjemahkan kembali oleh *switching center*, sehingga program seperti IVR dapat mengetahui tombol yang ditekan oleh *user*. Untuk tombol A, B, C, dan D sudah mulai banyak ditinggalkan dan tidak dipakai pada telepon jaman sekarang.

Tabel 2. 3 *Frekuensi DTMF* untuk Tombol Telephone

	1209Hz	1336Hz	1477Hz	163Hz
697Hz	1	2	3	A
770Hz	4	5	6	B
852Hz	7	8	9	C
941Hz	*	0	#	D

Untuk *decode* frekuensi DTMF, digunakan algoritma Goertzel.

2.7.2.4 *Directory Services*

Directory service menyediakan layanan yang mengasosiasikan sebuah nama (nama orang, lokasi, dsb.) dengan informasi mengenai nama tersebut. Nama yang dimaksud seringkali merupakan nama dari sebuah *resource* jaringan / terletak pada jaringan, seperti *user*, *server*, *file*, *printer*, *shared-directory*, dan lain – lain.

VoIP menggunakan layanan *directory service* untuk meng-asosiasikan nama dengan identifier pada protokol di layer lebih rendah (SIP URL, *proxy address*, *mac address*, dsb.), dan juga untuk menyediakan layanan phone book bagi pengguna-nya.

Ada berbagai macam implementasi dari *directory service*. X.500 merupakan standar *directory service* yang dibuat oleh *International Telecommunication Union* (ITU) dan *International Organization for Standardization* (ISO). *Lightweight Directory Access Protocol* (LDAP) merupakan pengembangan dari X.500 yang menggunakan protokol TCP/IP dalam implementasinya. *Protokol Network Information Service* (NIS) merupakan implementasi *directory service* ciptaan *Sun Microsystems* untuk sistem berbasis UNIX, sedangkan *Active Directory* adalah implementasi ciptaan *Microsoft*.

ITU *Recommendation* H.350 merupakan sebuah usulan *arsitektur directory service* untuk *multimedia conferencing* dengan menggunakan LDAP. ITU *Recommendation* H.350.4 secara spesifik adalah usulan *arsitektur directory service* untuk *multimedia conferencing* berbasis SIP dengan menggunakan LDAP.

2.7.2.5 Komponen Pendukung

2.7.2.5.1 *Media Encoding*

Di dalam dunia *telekomunikasi*, dikenal istilah *codec* (kependekan dari *coder*/*decoder*). *Codec* adalah sebutan bagi sebuah *hardware* / *software* yang bisa mengubah sinyal analog (seperti suara atau gambar) ke dalam bentuk sinyal digital agar bisa ditransmisikan melalui jaringan digital, dan pada saat penerimaan, mengubah sinyal digital tersebut ke dalam bentuk analog kembali.

Pada sebuah sistem VoIP, *codec* digunakan untuk mengubah sinyal suara dari format analog ke *format digital* (dan *vice-versa*), agar suara dapat ditransmisikan melewati jaringan IP. *Codec* yang digunakan pada VoIP sering disebut juga sebagai *vocoder* (*voice encoder*).

Ada banyak jenis *vocoder* yang tersedia pada saat ini. Beberapa yang populer antara lain ITU G.711, ITU G.723, ITU G.729, dan GSM.

Pada umumnya (namun tidak selalu), ada *trade-off* antara kualitas suara dari sebuah *vocoder* dengan besar *bandwidth* yang digunakan. Semakin baik kualitas suara, semakin besar *bandwidth* yang digunakan, dan begitu juga sebaliknya. *Bit-rate* untuk *narrowband vocoder* yang ada sekarang berkisar antara 1.2 hingga 64 kbps. *Vocoder* paling efisien yang ada pada saat ini mampu mengantarkan suara dengan *quasi-toll quality* (*toll quality* setara dengan kualitas suara pada komunikasi telpon biasa) dengan menggunakan *bandwidth* hanya 5 kbps saja. Seiring dengan berkembangnya teknologi komunikasi, diharapkan pada nantinya angka tersebut bisa menjadi lebih efisien lagi.

Untuk mendapatkan kualitas suara yang baik dan dengan penggunaan *bandwidth* yang seminimal mungkin, maka dibutuhkan pemilihan *codec* yang tepat, sesuai dengan kebutuhan dan kondisi infrastruktur yang ada.

Pada tugas akhir ini, penulis tidak memilih codec secara spesifik. Pemilihan *codec* diserahkan kepada user agent yang dipakai oleh user. Sistem yang diimplementasikan oleh penulis mendukung semua codec yang umum dipakai. Alasan pembebasan ini antara lain dilatarbelakangi oleh masih tersedianya bandwidth yang cukup besar, dan tidak dibatasinya user agent yang dapat digunakan oleh user.

2.7.2.5.2 *Gateway Control*

Seperti yang telah dijelaskan, pada skenario penggunaan *VoIP PC* to *Phone* dan *Phone to PC*, untuk menghubungkan jaringan IP dengan jaringan PSTN, digunakan sebuah *Media Gateway* (MG) yang berfungsi untuk melakukan proses konversi sinyal analog menjadi paket (dan juga sebaliknya), dan juga melakukan

proses translasi antara protokol signaling yang digunakan pada masing – masing jaringan IP dan jaringan PSTN.

Media Gateway Controller (MGC) atau sering disebut juga sebagai "*softswitch*" adalah sebutan bagi device yang melakukan proses pengontrolan terhadap sebuah *media gateway*.

H.248 (atau lebih populer dengan sebutan *Megaco*) adalah sebuah protokol pengontrolan yang digunakan oleh MGC untuk mengatur MG. *Megaco* merupakan hasil kolaborasi antara ITU dengan *Internet Engineering Task Force* (IETF). Sebelum *Megaco* di-spesifikasikan, sudah ada beberapa protokol lainnya, seperti *Media Gateway Control Protocol* (MGCP) dan *Media Device Control Protocol* (MDCP). *Megaco* merupakan hasil pengembangan dari MGCP.

Protokol *Megaco* dipandang sebagai komplemen dari H.323 dan SIP, dalam artian bahwa sebuah MGC akan mengontrol MG dengan menggunakan Megaco, namun berkomunikasi dengan sesama MGC menggunakan H.323 atau SIP.

2.7.3 Standar Kompresi Data Suara

ITU-T (*International Telecommunication Union – Telecommunication Sector*) mengeluarkan beberapa standar untuk *voice coding* yang direkomendasikan untuk digunakan pada implementasi VoIP. Beberapa standar yang dipakai dalam implementasi IP *Telephony* ini adalah :

2.7.3.1 G.711

G.711 adalah suatu standar Internasional untuk melakukan kompresi terhadap data *audio* dengan menggunakan teknik *Pulse Code Modulation* (PCM) dalam pengiriman suara. PCM melakukan konversi sinyal analog ke bentuk digital dengan melakukan sampling sinyal analog tersebut 8000 kali/detik dan dikodekan dengan kode angka. Jarak antara sampel adalah 125 μ detik.. Sinyal tersampel lalu dikonversikan ke bentuk diskrit. Sinyal diskrit ini direpresentasikan dengan kode yang disesuaikan dengan amplitudo dari sinyal sampel. Format PCM menggunakan 8 bit untuk pengkodeannya. Laju transmisi diperoleh dengan mengkalikan 8000 sampel/detik dengan 8 bit/sampel, menghasilkan 64.000

bit/detik. *Bit rate* 64 kbps ini merupakan standar transmisi untuk satu kanal telepon digital.

Pada VoIP *gateway*, di bagian *terminal*, terdapat *audio codec* yang melakukan proses *framing* (pembentukan *frame datagram* IP yang dikompresi) dari sinyal suara terdigitasi (hasil PCM G.711) dan juga melakukan rekonstruksi pada sisi *receiver*. *Frame – frame* yang merupakan paket – paket informasi ini lalu di transmisikan melalui jaringan IP dengan suatu standar komunikasi jaringan *packet – based*. Standar G.711 merupakan teknik kompresi yang tidak efisien, karena akan memakan *bandwidth* 64Kbps unutk kanal pembicaraan.

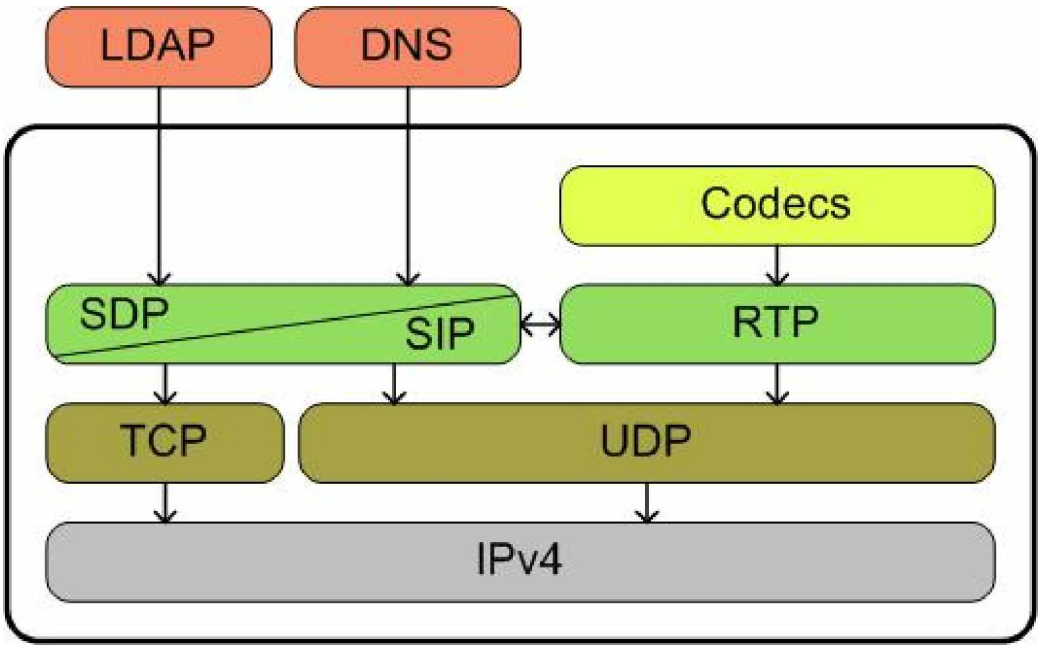
2.7.3.2 G.723.1

G.723.1 adalah jenis pengkode suara yang direkomendasikan untuk terminal multimedia dengan *bit rate* rendah.G.723.1 memiliki dual *rate speech coder* yang dapat di-switch pada batas 5.3 kbit/s dan 6.3 kbit/s. Dengan memiliki dual *rate speech coder* ini maka G.723.1 memiliki fleksibilitas dalam beradaptasi terhadap memperbagus sinyal suara hasil sintesis. Pada bagian *encoder* G.723.1 dilengkapi dengan *formant perceptual weighting filter* dan *harmonic noise shaping filter* sementara di bagian *decoder*-nya G.723.1 memiliki *pitch postfilter* dan *formant postfilter* sehingga sinyal suara hasil rekonstruksi menjadi sangat mirip dengan aslinya. Sinyal eksitasi untuk *bit rate* rendah dikodekan dengan *Algebraic Code Excited Linier Predicition* (ACELP) sedangkan untuk *rate* tinggi dikodekan dengan menggunakan *Multipulse Maximum Likelihood Quantization* (MP-MLQ). Rate yang lebih tinggi menghasilkan kualitas yang lebih baik. Masukan bagi G.723.1 adalah sinal suara digital yang di-sampling dengan frekuensi sampling 8.000 Hz dan dikuantisasi dengan PCM 16 bit. *Delay* algoritmik dari G.723.1 adalah 37.5 msec (panjang *frame* ditambah *lookahead*), *delay* pemrosesnya sangat ditentukan oleh prosesor yang mengerjakan perhitungan.

2.7.4 Protocol Stack

Berikut ini adalah building block dari protokol – protokol (atau lazim disebut

sebagai protocol stack) yang digunakan dalam tugas akhir ini, digambarkan dalam kerangka VoIP:



Gambar 2. 5 Protocol Stack VoIP

2.7.5 VoIP *Gateway*

VoIP *Gateway* adalah sebuah perangkat jaringan yang melakukan konversi terhadap panggilan telepon atau fax antara PSTN dengan jaringan berbasis IP. Fungsi utamanya meliputi kompresi dan dekompresi suara dan fax, *packeting, routing* panggilan, dan *control signaling*. *Gateway* ini memungkinkan terjadinya koneksi antara IP ke perangkat *legacy* PBX dan antara PSTN dengan sistem IP PBX.

2.7.6 IP-PBX Server

IP-PBX (*Internet Protocol - Private Branches eXchange*) adalah teknologi yang memungkinkan sambungan antara beberapa telepon dalam lingkungan privat dan hubungan ke lingkungan luar (misalnya PSTN). IP-PBX *server* adalah perangkat lunak atau mesin yang mengatur dan menjalankan tugas PBX. Beberapa IP-PBX *server* gratis berbasis perangkat lunak di antaranya adalah Asterisk, OpenSER, dan SIP *Express Router* (SER).

Asterisk, produk dari Digium, adalah salah satu IP-PBX *server* yang bersifat *open-source*, dan berarti dapat digunakan dengan gratis dan dapat dimodifikasi sesuai keperluan. Asterisk mampu berjalan pada beberapa *platform* yang ada, namun paling baik berjalan pada *platform* Linux. Secara umum, Asterisk memiliki fitur-fitur dasar seperti *voice mail, conference calling, interactive voice response*, dan *automatic call distribution*. Untuk menghubungkan PBX dengan PSTN, Asterisk membutuhkan PCI *device* tertentu (dijual oleh Digium) yang berfungsi sebagai penghubung, apakah itu ke PSTN, jalur T1, jalur E1, atau lainnya.

Asterisk mendukung beberapa protocol VoIP terkenal, di antaranya SIP dan H.323. Namun Digium juga mengembangkan protokol baru yang dinamakan IAX (*Inter-Asterisk eXchange*), yaitu protokol yang khusus didesain untuk kepentingan *trunking* antar Asterisk.

2.8 *Advanced Encryption Standard* (AES)

Advanced Encrption Standar merupakan algoritma kriptografi modern yang menggantikan algoritma DES (*Data Encryption Standard*). Sebagai standard enkripsi kritografi simetri.

AES merupakan hasil sayembara pencarian algoritma kriptografi yang baru. Pada kenyatannya, AES menggunakan algoritma *Rijndael* yang diusulkan oleh *Vincent Rijmen* dan *Joan Daemen*. Sebenarnya ada fitur tambahan yang tidak diimplementasikan dari algoritma *Rijndael* yaitu penanganan untuk ukuran block dan ukuran key masukkan yang berlebihan.

AES memiliki 3 buah versi berdasarkan panjang kunci yaitu: AES128, AES192, dan AES256. Tetapi algoritma AES192 sangat jarang untuk digunakan, sehingga algoritma AES128 dan AES256, lebih lazim digunakan.

Algoritma Rijndael, secara garis besar terbagi menjadi 3 buah bagian, yaitu: *AddRoundKey*, putaran sebanyak Nr, dan *Final Round*.

AddRoundkey yaitu merupakan sebuah tahapan inisialisasi dari AES. Add roundkey melakukan operasi xor dengan key. Putaran sebanyak Nr -1 terdiri dari

4 buah proses sederhana yaitu: *subbytes* (subtitusi byte dengan menggunakan sbox), *ShiftRows* (pergeseran baris-baris array state secara *wrapping*), *MixColumns* (mengacak data di masing-masing kolom *array state*), *AddRoundkey* (melakukan SOR state sekarang dengan *round key*). Keempat Subproses diatas akan diulangi sebanyak Nr -1 kali. Proses yang terakhir terdiri dari 3 buah *subprose* yaitu *subbytes*, *Shiftrows*, dan *addRoundKey*.

2.9 ZRTP

Zimmerman Realtime Transport Protocol (ZRTP) adalah *key agreement protocol* yang menggunakan *Diffie Hellman* (DH) sebagai *key exchange* saat melakukan inisiasi panggilan VoIP dan menggunakan *Realtime Transport Protocol* (RTP) sebagai *media transport* nya yang telah diinisiasikan oleh *Signaling Protocol* seperti *Session Initiation Protocol* (SIP).

Zimmerman Realtime Tranport Protocol berperan sebagai pengaman komunikasi pada Voice over Internet Protocol. Penerapan ZRTP pada VoIP diperuntukan agar VoIP mampu menahan penyadapan melalui serangan *Man in The Middle*. ZRTP mengenkripsi data komunikasi menggunakan AES 128.

ZRTP memiliki fitur kriptografi yang menarik dibandingkan dengan metode pada *media session encryption* yang lain. Walaupun ZRTP menggunakan algoritma *public key,* tetapi proses pada ZRTP tidak berdasarkan pada *Public Key Infrastrucure* (PKI). ZRTP menggunakan hasil hash dari DH lalu menampilkannya berupa short authentication string (SAS) kepada pengguna agar dapat dibandingkan melalui suara pada saat panggilan dilakukan. Pada panggilan berikutnya pengguna tidak akan direpotkan dengan SAS, ZRTP akan menyimpan *Key Material* pada panggilan sebelumya dan mengkalkalkulasikan kembali dengan *DH shared secret.* Proses *Key Management* pada ZRTP tidak berdasar kepada SIP dan pada VoIP server. *Key Management* dilakukan secara *peer-to-peer* melalui paket RTP.

2.9.1 *Key Agreement*

Sebelum melakukan enkripsi terhadap data yang dikirim, masing masing

endpoint mengirim semacam pesan pembuka atau *Hello Messages* untuk mengkonfirmasi protokol yang digunakan dari setiap endpoint dan menanyakan apakah algoritma yang digunakan sama. Pada *Hello Messages* tersebut terdapat juga informasi konfigurasi SRTP dan ZRTP ID atau ZID. Setiap *instance* dari ZRTP memiliki 96-bit kode unik ZID yang acak. ZID di*generate* pada saat instalasi. ZID yang diterima pada saat proses pertukaran *Hello Message* digunakan untuk mengecek apakah endpoint tersebut sudah memiliki shared secret dari sesi ZRTP yang sebelumnya.

Setelah pertukaran antar *endpoint* terhadap *Hello Message* tersebut, selanjutnya dilakukan *Key Agreement* untuk menentukan *key exchange* yang akan digunakan. Dalam ZRTP terdapat 3 mode *Key Agreements* yaitu *Diffie Helman*, *Preshared* dan *Multistream*. Setelah *Key Agreement* disetujui, proses enkripsi menggunakan AES 128 dimulai dan komunikasi sudah bisa dilakukan.

2.9.1.1 *Diffie Hellman* (DH)

Diffie-Hellman (DH) *Key Exchange* adalah protokol kriptografik yang memungkinkan kedua pihak yang bertukar informasi, walaupun mereka tidak mengenal satu sama lainnya, dapat secara bersama menciptakan sebuah kunci rahasia bersama melalui sebuah jalur komunikasi yang tak aman sekalipun. Kunci ini kemudian dapat digunakan untuk mengenkripsi pesan menggunakan kunci cipher simetris. DH *Key Exchange* mempunyai beberapa sinonim. Di antaranya adalah:

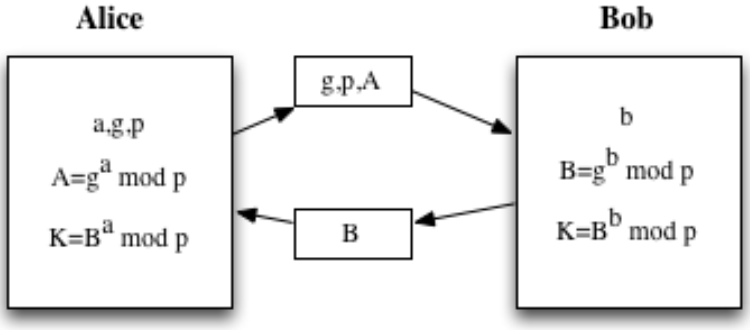
a. *Diffie-Hellman key agreement*
b. *Diffie-Hellman key establishment*
c. *Diffie-Hellman key negotiation*
d. *Exponential key exchange*

DH Key Exchange ditemukan pada tahun 1976 atas hasil kerjasama antara Whitfield Diffie dan Martin Hellman. Metode ini merupakan metude praktikal pertama untuk menciptakan sebuah rahasia bersama antara dua belah pihak melalui sebuah jalur komunikasi yang tidak terjaga. Metode ini amat dipengaruhi

oleh karya-karya Ralph Merkle dalam bidang distribusi kunci publik. Sedangkan John Gill juga turut berkontribusi dalam penciptakan metode ini dengan menyarankan pengaplikasian masalah logaritma diskrit.

Sebenarnya, metode yang sama telah lebih dulu diciptakan oleh Malcolm Williamson dari GCHQ (Government Communications Headquarters) di Inggris. Namun GCHQ memilih untuk menyembunyikannya dari publik sampai pada akhirnya mempublikasikannya pada tahun 1997. Kemudian pada tahun yang sama, metode ini dijadikan sebagai dasar oleh tiga orang dari MIT (Massachusetts Institute of Technology) yaitu Ron Rivest, Adi Shamir, dan Len Adleman untuk menciptakan algoritma RSA. Metode DH *Key Exchange* dipatenkan dengan U.S Patent no. 4,200,770 dengan mengkreditkan Whitfield Diffie, Martin Hellman, dan Ralph Merkle sebagai penemunya [5]. Secara umum metode DH dapat digambarkan oleh skema diagram berikut:

Gambar 2. 6 Skema Diffie-Hellman Key Exchange



Berikut adalah penjelasan deskriptif mengenai diagram di atas:

1. Alice dan Bob sepakat untuk menggunakan bilangan prima p dan suatu bilangan g yang akan digunakan sebagai basis.

2. Alice memilih suatu bilangan natural acak a dan mengirim (ga mod p) ke Bob.

3. Bob memilih suatu bilangan natural acak b dan mengirim (gb mod p) ke Alice.

4. Alice menghitung (gb mod p)a.

5. Bob menghitung (ga mod p)b.

Sekarang Alice dan Bob memiliki nilai dari $(g^a \bmod p)^b$ dan $(g^b \bmod p)^a$ yang notabene memiliki nilai yang sama berdasarkan sifat asosiatif perpangkatan, yaitu: $(g^b)^a = (g^b)^a$

Nilai tersebut adalah nilai dari kunci yang dapat digunakan oleh Alice dan Bob untuk mengenkripsi dan mendekripsi pesan mereka. Perlu dicatat bahwa hanya nilai a, b, $(g^a \bmod p)^b$,dan,$(g^b \bmod p)^a$ yang perlu dirahasiakan.Tentu saja dengan menggunakan nilai a, b , dan p yang besar, maka akan didapatkan nilai kunci yang lebih sulit untuk dipecahkan oleh pihak ketiga

Protokol DH dinilai aman dari pihak ke tiga yang bertindak sebagai pengamat yang tidak diinginkan (eavesdropper) apabila variabel g, a, b, dan p dipilih secara tepat. Bila angka-angka tersebut dipilih dengan benar, maka pengamat tersebut akan kesulitan untuk memecahkan apa yang dikenal dengan istilah DH *Problem* untuk menemukan kunci publik yang diinginkan.

Kelemahan dari DH *Key Exchange* akan terjadi apabila terpenuhinya syarat sehungga pengamat dapat menggunakan Algoritma Pohlig- Hellman (yang dapat diatasi dengan bilangan prima Sophie-Germain) dan adanya *Man in the Middle Attack*.

2.9.1.2 *Preshared Mode*

Pada *Preshared Mode*, *endpoint* dapat melewati proses kalkulasi pada DH jika endpoint telah memiliki shared secret dari sesi ZRTP yang sebelumnya. Perbedaan antara *Preshared Mode* dan *Multistream Mode* adalah *Preshared Mode* menggunakan *shared secret* yang telah disimpan daripada menggunakan *session key* ZRTP yang sedang aktif sebagai *keying material.* [3]

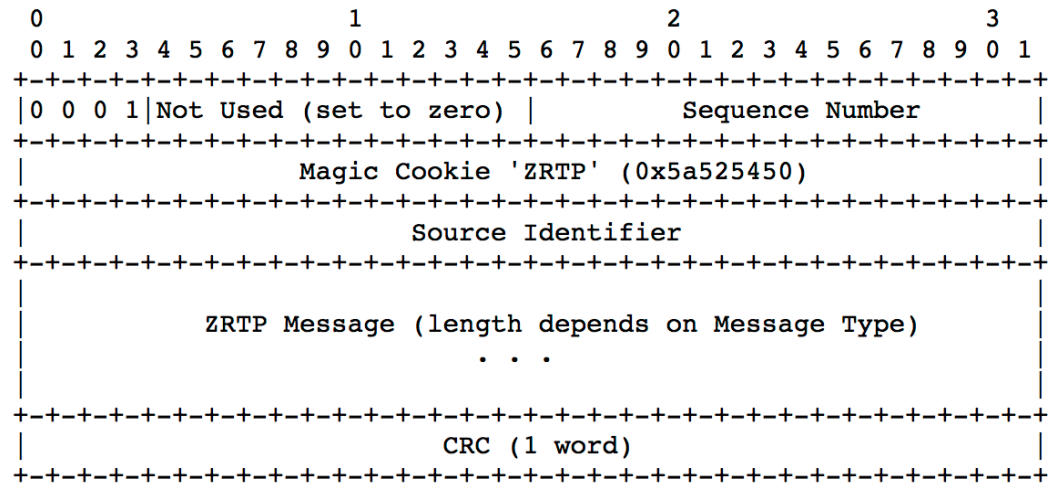
Pada *Preshared Mode,* Kalkulasi DH tidak harus dilakukan setiap memulai session sehingga mode ini akan sangat membantu *endpoint* yang memiliki prosessor rendah. Atau, mode ini dapat digunakan untuk mengembalikan *session* yang terputus sehingga tidak perlu mengulang proses kalkulasi DH.

2.9.1.3 Multistream Mode

Multistream mode adalah alternatif metode key agreement ketika dua endpoint telah menyelesaikan SRTP media stream dan memiliki ZRTP session key yang aktif. ZRTP dapat mengambil banyak SRTP key menggunakan satu DH exchange.

2.9.2 ZRTP Messages

Semua ZRTP Messages menggunakan format pesan seperti pada gambar 2.7. Panjang karakter pada gambar 2.7 adalah 32 bits atau 4 oktet. [3]



Gambar 2. 7 Format Paket pada ZRTP

Sequence Number adalah penomeran dari setiap paket ZRTP yang dikirim. Penomeran dimulai dari nilai yang diacak. Metode tersebut digunakan untuk memperkirakan packet ZRTP yang hilang dan mendeteksi ketika paket ZRTP diterima tidak berdasarkan urutan.

ZRTP Magic Cookie adalah string dengan panjang 32 bit yang secara unik mengidentifikasi paket ZRTP dan memiliki nilai 0x5a525450.

Source Identifier adalah nomer SSRC dari paket RTP Stream yang berhubungan dengan paket ZRTP. Pada kasus *Forking* (Percabangan) dan *Forwarding*, RTP dan ZRTP mungkin tiba pada port yang sama dengan sumber

yang berbeda. Setiap sumber tersebut akan memiliki SSRC yang berbeda, bahkan bisa menunjukan sesi ZRTP yang berbeda pula.

ZRTP menggunakan 32 bit *Cyclic Redundancy Check* (CRC) untuk mendeteksi transmisi yang *error.*

2.9.2.1 ZRTP *Message Format*

ZRTP Message didesain untuk menyederhanakan kebutuhan parsing dari *endpoint* dan mengurangi celah penyerangan melalui *buffer overflow attacks.* ZRTP menggunakan 8 oktet blok (2 kata) untuk meng-*encode Message Type.* 4 oktet digunakan untuk meng-*encode Hash Type, Cipher Type, Key Agreement Type*, dan *Authentication Tag Type*. Nilai pada blok adalah berupa ASCII *strings* yang diperluas dengan spasi (0x20).

2.9.2.1.1 *Message Type Block*

Terdapat 16 *Message Type Block* yang merepresentasikan awal dari *ZRTP Messages* [3]. Berikut 16 *Message Type Block* tersebut;

Tabel 2. 4 Nilai *Message Type Block*

Message Type Block	**Definisi**
“Hello”	Pesan Hello
“HelloACK”	Pesan HelloACK
“Commit”	Pesan Commit
“DHPart1”	Pesan DHPart1
“DHPart2”	Pesan DHPart2
“Confirm1”	Pesan Confirm1
“Confirm2”	Pesan Confirm2
“Conf2ACK”	Pesan Conf2ACK
“Error”	Pesan Error

Message Type Block	**Definisi**
“ErrorACK”	Pesan ErrorACK
“GoClear”	Pesan GoClear
“ClearACK”	Pesan ClearACK
“SASrelay”	Pesan SASrelay
“RelayACK”	Pesan RelayACK
“Ping”	Pesan Ping
“PingACK”	Pesan PingACK

2.9.2.1.2 *Hash Type Block*

Algoritma hash dan semua hal yang berhubungan dengan algoritma MAC dinegosiasikan menggunakan *Hash Type Block* yang terdapat pada *Hello Message* dan *Commit Message.* Seluruh ZRTP *endpoint* harus dapat mendukung *Hash* dengan type SHA-256. *Hash Type* menentukan *Hash Algorithm* yang akan digunakan pada proses *ZRTP key exchange*, bukan menentukan *Hash Algorithm* yang akan digunakan pada *SRTP Authentication Tag*[3].

Tabel 2. 5 Nilai *Hash Type Block*

Hash Type Block	Pengertian
“S256”	SHA-256 Hash
“S384”	SHA-284 Hash
“N256”	NIST SHA-3 256-bit hash
“N384”	NIST SHA-3 384-bit hash

2.9.2.1.3 *Cipher Type Block*

Blok *Cipher Algorithm* dinegosiasikan melalui *Cipher Type Block* yang terdapat pada *Hello Message* dan *Commit Message*.[3]

Tabel 2. 6 Nilai *Cipher Type Block*

Cipher Type Block	Pengertian
"AES1"	AES dengan kunci 128-bit
"AES2"	AES dengan kunci 198-bit
"AES3"	AES dengan kunci 256-bit
"2FS1"	*TwoFish* dengan kunci 128-bit
"2FS2"	*TwoFish* dengan kunci 198-bit
"2FS3"	*TwoFish* dengan kunci 256-bit

2.9.2.1.4 *Auth Tag Type Block*

Seluruh ZRTP *endpoint* harus mendukung *HMAC-SHA1 authentication* tags untuk SRTP dengan panjang tag 32-bit dan 80 bit. ZRTP *endpoint* juga dapat menggunakan *SRTP authentication tags* dengan panjang 32-bit dan 64-bit yang berdasar kepada *Skein Hash Function (Skein)* [3].

Tabel 2. 7 Nilai *Auth Tag Type*

Auth Tag Type Block	Pengertian
"*HS32*"	*32-bit authentication tag* berdasarkan *HMAC-SHA1*
"*HS80*"	*80-bit authentication tag* berdasarkan *HMAC-SHA1*
"*SK32*"	*32-bit authentication tag* berdasarkan *Skein-512-MAC* dgn panjang kunci 256-bit dan 32-bit panjang MAC
"*SK64*"	*64-bit authentication tag* berdasarkan *Skein-512-MAC* dgn panjang kunci 256-bit dan 64-bit panjang MAC

2.9.2.1.5 *Key Agreement Type Block*

Seluruh ZRTP endpoint harus mendukung DH3k, *Preshared* dan EC25,EC38 dan DH2k. *Hash Algorithm* yang dipilih berhubungan dengan *Key Agreement Type*. Jika EDCH-384 (EC38) dipilih sebagai key agreement, maka hash algorithm –nya adalah SHA-384 atau yang berhubungna dengan SHA-3 [3].

DH2k digunakan agar dapat mendukung aplikasi yang memiliki spesifikasi hardware yang rendah, serta untuk aplikasi yang memerlukan kecepatan dalam menegosiasikan kunci.

Tabel 2. 8 Nilai *Key Agreement Type Block*

Key Agreement Type Block	Pv words	Message words	Pengertian
"DH3k"	96	117	DH mode dengan p=3072 bit prime
"DH2k"	64	85	DH mode dengan p=2048 bit prime
"EC25"	16	37	Elliptic Curve DH dengan P-256
"EC38"	24	45	Elliptic Curve DH dengan P-384
"EC52"	33	54	Elliptic Curve DH dengan P-521
"Prsh"	-	-	*Preshared Non-DH mode*
"Mult"	-	-	*Multistream Non-DH mode*

2.9.2.1.6 SAS *Type Block*

SAS Type menentukan bagaimana SAS ditampilkan ke pengguna sehingga pengguna dapat membandingkannya secara verbal melalui komunikasi suara. Metode ini dapat mendeteksi serangan *MiTM*. Seluruh ZRTP *endpoint* harus mendukung base 32 dan base256 untuk me-*render Short Authentication String.*

Tabel 2. 9 Nilai SAS *Type Block*

SAS Type Block	Pengertian
"B32"	SAS menggunakan encoding base32
"B256"	SAS menggunakan encoding base256

2.9.2.1.7 *Signature Type Block*

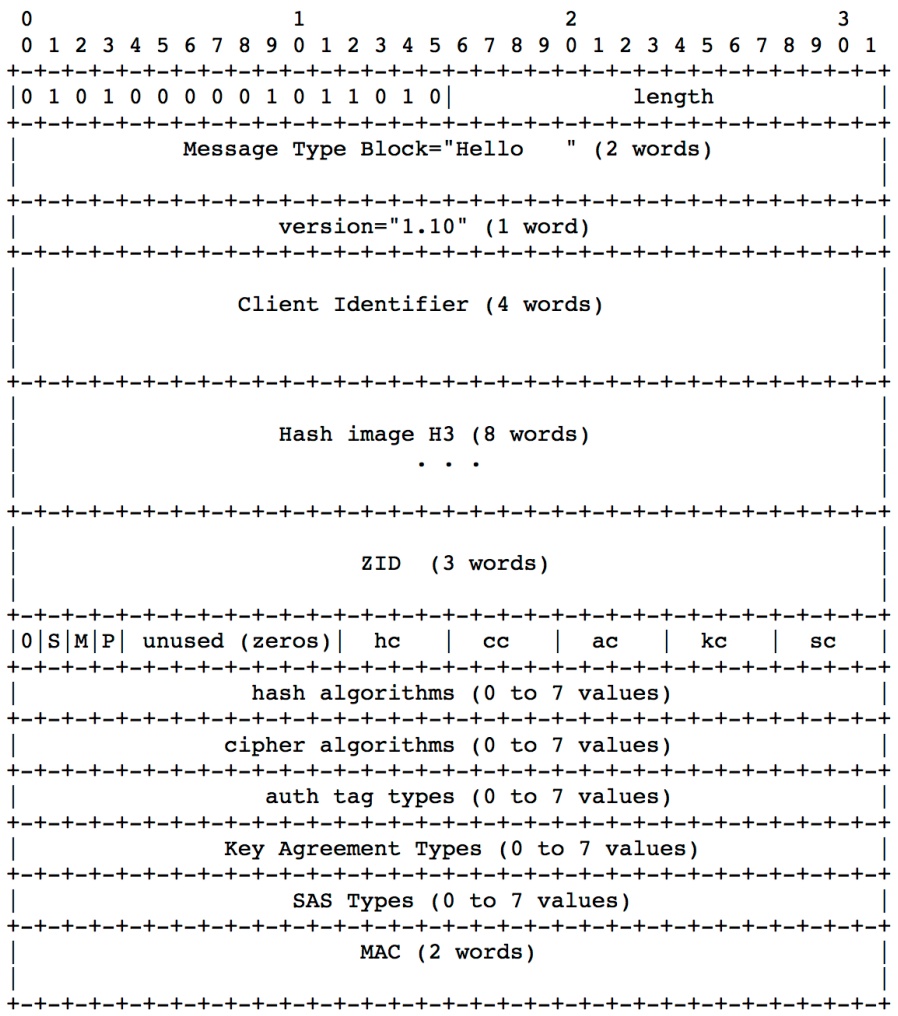
Signature Type Block menentukan siganture algorithm yang akan digunakan untuk SAS [3]. Berikut *siganture type* yang dapat digunakan;

Tabel 2. 10 Nilai *Signature Type Block*

Signature Type Block	*Meaning*
"PGP"	*OpenPGP Signature*
"X509"	ECDSA dengan sertifikat X.509v3

2.9.2.2 *Hello Message*

Seluruh ZRTP messages dimulai dengan nilai awal 0x505a, lalu 32-bit kata dengan panjang 16-bit. Didalamnya hanya terdapat *ZRTP message*, bukan header dari paket ZRTP dan CRC [3].



Gambar 2. 8 Format Hello Message

Message Type memiliki panjang 8 oktet, selanjutnya terdapat 4 karakter string yang menunjukan versi dari ZRTP protocol. *Client Identifier* (cid) menunjukan vendor dan versi dari ZRTP *software.* Selanjutnya terdapat 4 bits yang berisi tiga *flag bits*, yaitu;

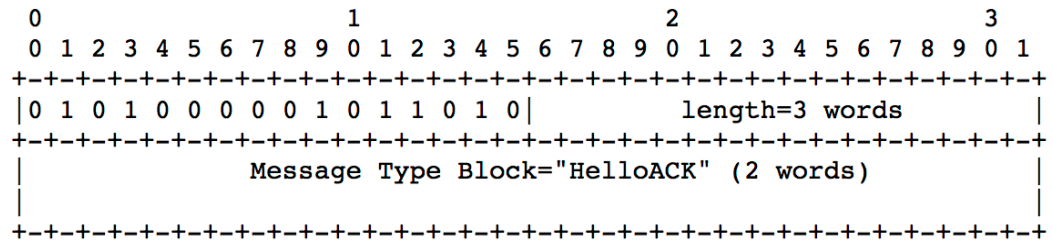
1. *Signature-capable flag* (S) menunjukan bahwa paket *Hello Message* dikirim dari ZRTP *endpoints* yang telah memiliki fungsi untuk mem-*parsing* dan verifikasi *digital signature*. Jika *signature* tidak didukung, maka nilai (S) adalah nol (0).

2. *MiTM flag* (M), memiliki tipe *Boolean* yang berisi *true* ketika *Hello Message* dikirim dari perangkat yang memiliki kemampuan untuk mengirim *SASrelay message*. (PBX).
3. *Passive Flag* (P) adalah sebuah *Boolean* yang secara default bernilai *false* dan bernilai *true* hanya ketika *Hello Message* dikirim dari perangkat yang telah terkonfigurasi untuk tidak mengirim *Commit Message*. Artinya perangkat tersebut tidak bisa memulai *secure session*, tetapi dapat bertindak sebagai *responder*.

8 bits selanjutnya tidak pernah dipakai dan harus bernilai nol (0) ketika dikirim dan harus diabaikan ketika diterima. Selanjutnya adalah daftar dari *Hash Algorithm* yang didukung, yaitu *Cipher algorithm, SRTP Auth Tag Type, Key Agreement Type*, dan *SAS Type.* Keterangan kode dari daftar tersebut diantaranya *hc=hash-count, cc=cipher count, ac=auth tag count, kc=key agreement count* dan *sc=sas count.*

2.9.2.3 *HelloACK Message*

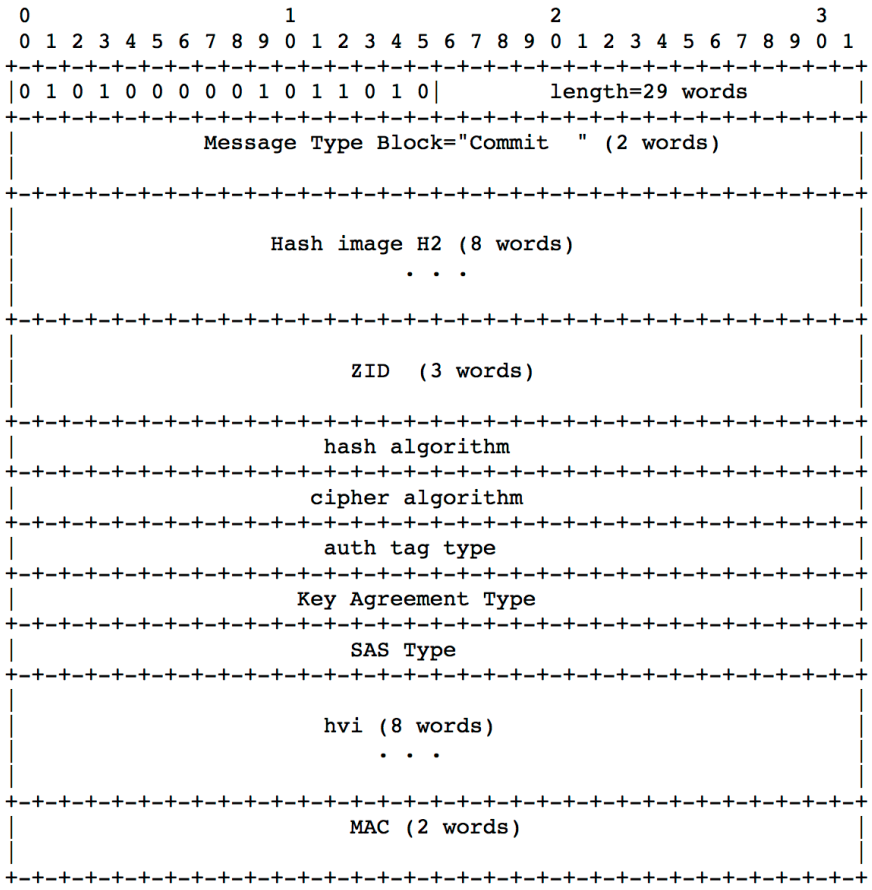
HelloACK Message berfungsi untuk menghentikan transmisi ulang dari *Hello message*. *HelloACK* dikirim untuk memberitahu bahwa *version number* pada *Hello* message didukung atau algorithma yang diminta dapat dilakukan. Penerima dari *HelloACK* akan menghentikan pengiriman *Hello message* [3]. Berikut format dari *HelloACK*;



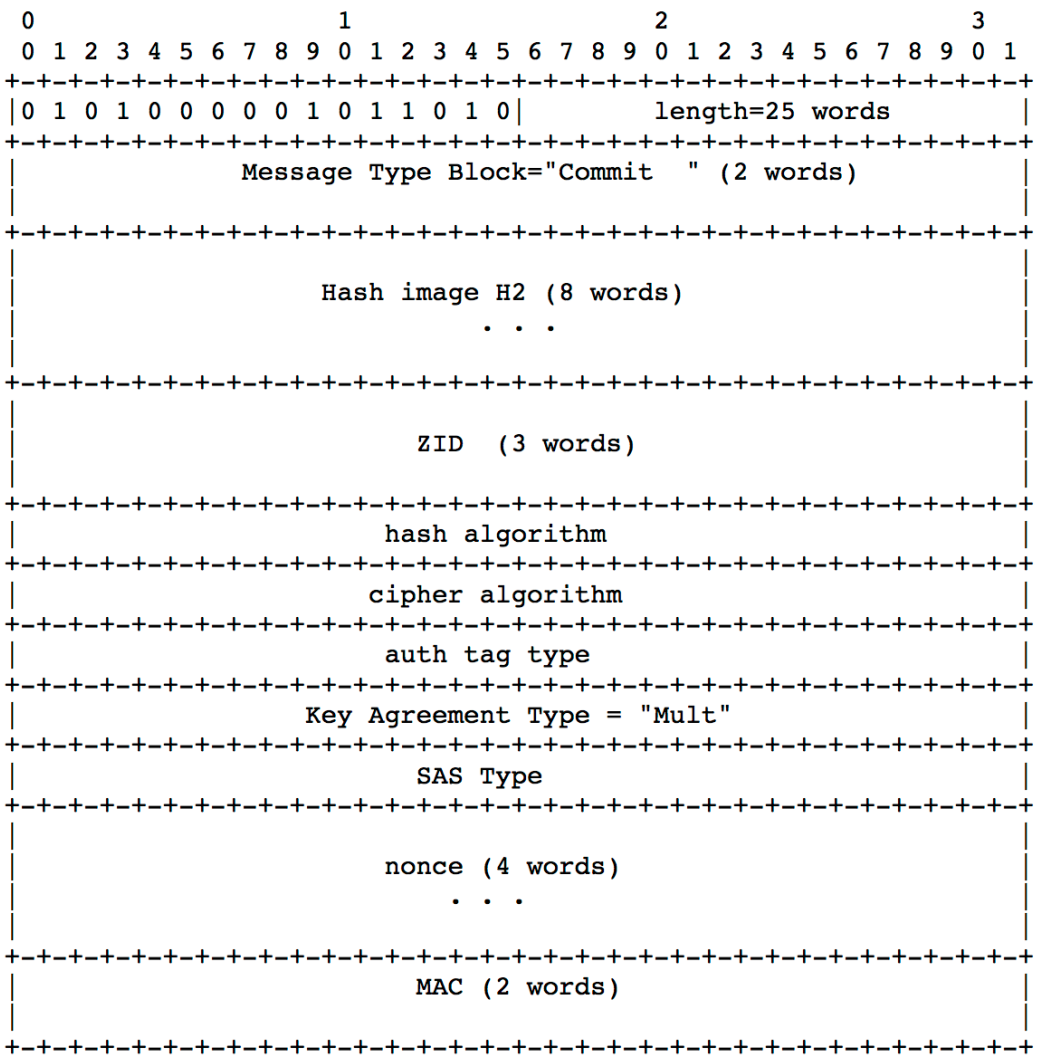
Gambar 2. 9 Format *HelloACK*

2.9.2.4 *Commit Message*

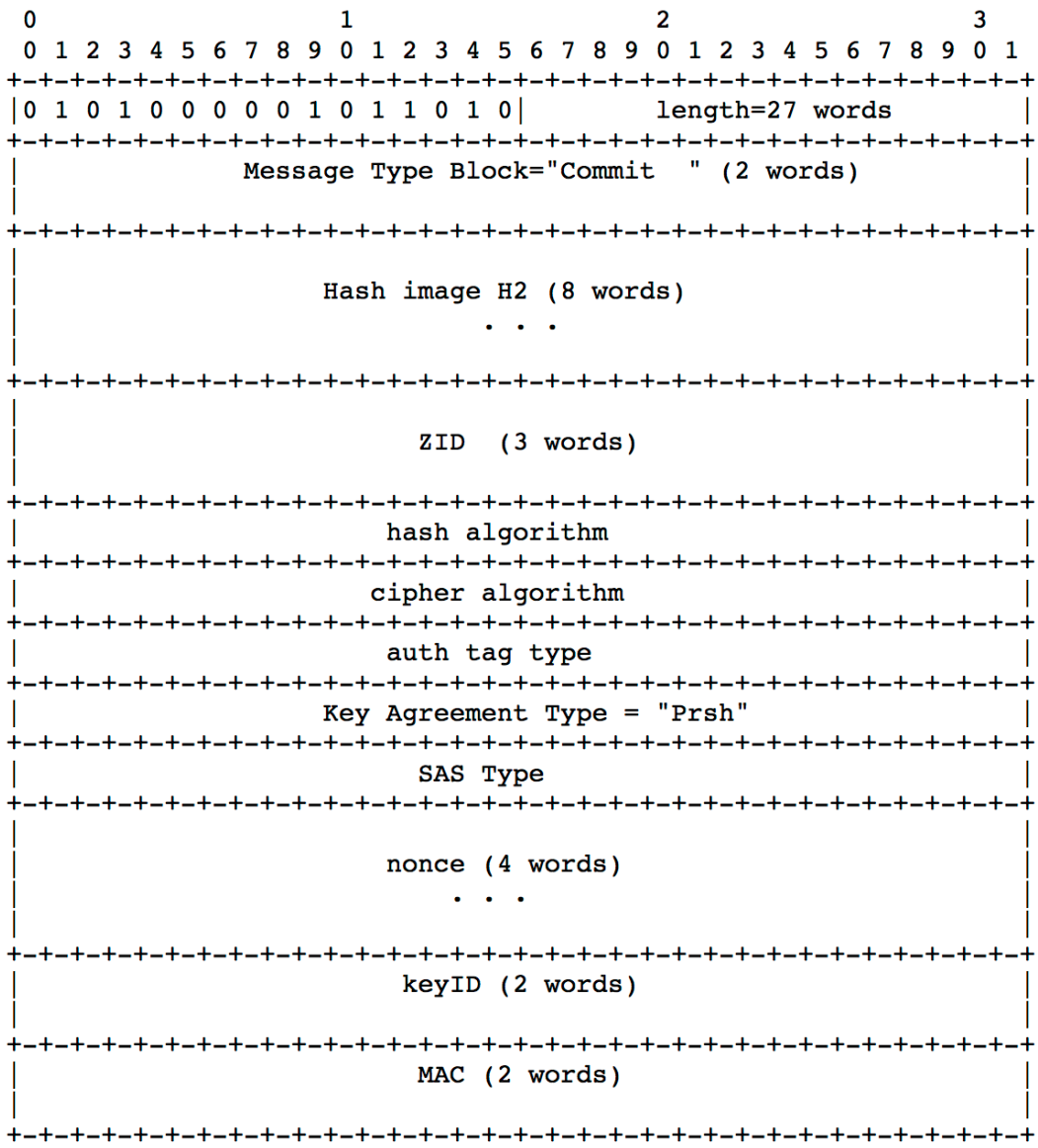
Commit Message dikirim untuk memulai proses *key agreement* setelah kedua *endpoint* menerima *Hello message* dan *HelloACK message*. Terdapat tiga subtipe dari *Commit Message* yang format nya ditampilkan pada Gambar 2.9, 2.10 dan 2.11. Tiga subtipe tersebut berdasar kepada tiga mode key agreement yaitu DH, *multistream* dan *preshared* [3].



Gambar 2. 10 DH Commit Message



Gambar 2. 11 Multistream Commit Message Format



Gambar 2. 12 Preshared Commit Message Format

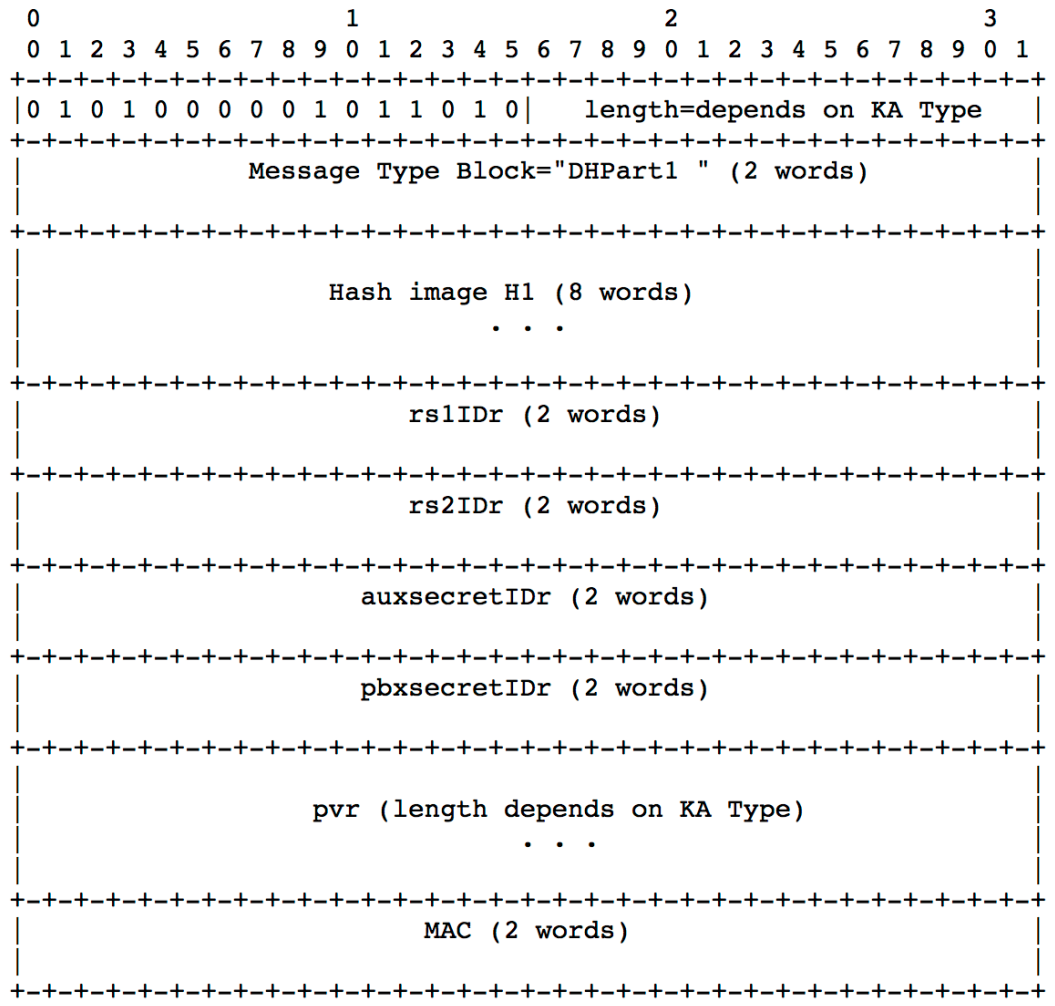
Commit Message terdiri dari Message Type Block, lalu 256-bit hash image H2. Parameter selanjutnya adalah ZID *initiator* (perangkat yg memulai komunikasi), memiliki panjang 96-bit *unique identifier* untuk ZRTP *endpoint*, nilai ZID tersebut harus sama dengan yang digunakan pada *Hello message*.

Selanjutnya terdapat daftar algoritma yang dipilih oleh *initiator* (*hash, cipher, auth tag type, key agreement, sas type*). Untuk yang menggunakan DH,

hash value hvi adalah hash dari *Dhpart2* milik *initiator* dan *Hello message* dari *responder*.

2.9.2.5 *DHPart1 Message*

Pengiriman *DHPart 1 message* menandakan dimulainya pertukaran kunci menggunakan DH atau DH *exchange*. *Message* ini dikirim oleh *responder* ketika telah menerima *Commit Message* dari *initiator*. Panjang dari nilai *pvr* dan panjang dari *DHPart1* message bergantung kepada *Key Agreement Type* yang dipilih. Sebagai pengingat, ketika menggunakan *Multistream* dan *Preshared Mode*, message *DHPart1* dan *DHPart2* tidak akan dikirim [3].

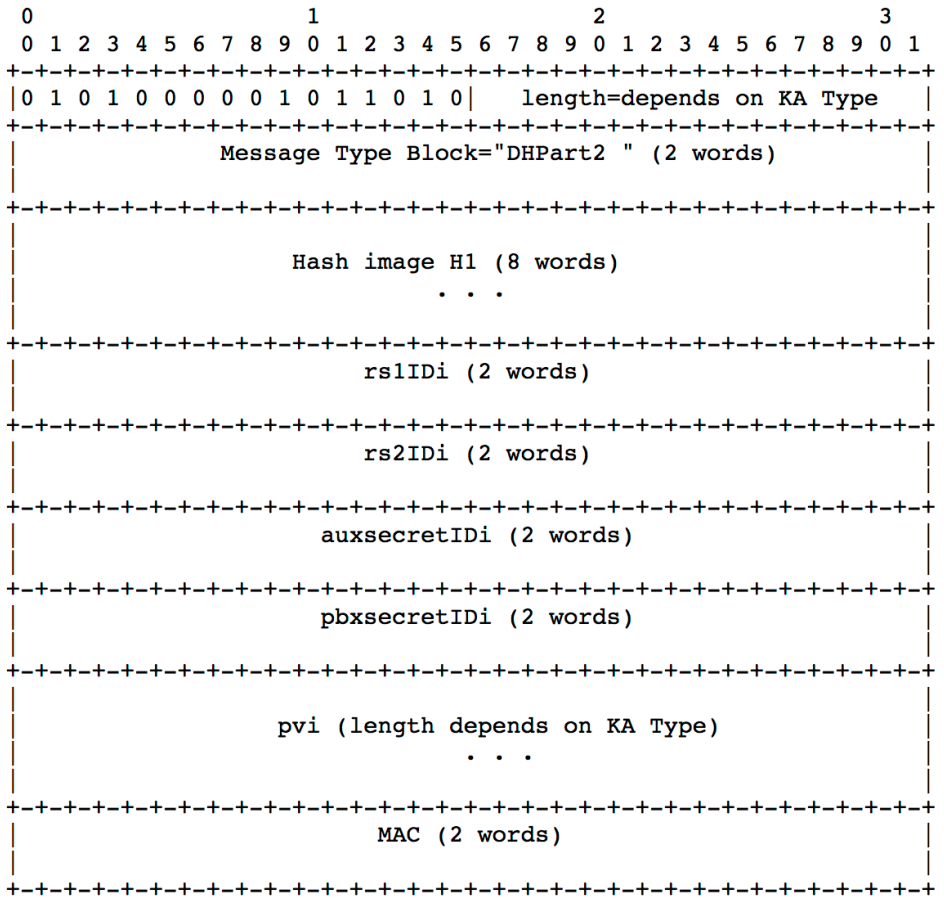


Gambar 2. 13 DHPart1 Message Format

Terdapat empat parameter (*rs1IDr,rs2IDr,auxsecretIDr, dan pbxsecretIDr*) yang merupakan *non-invertible hashes* berisi *shared secret* yang digunakan untuk membuat ZRTP *secret*. *rs1IDr* dan *rs2IDr* merupakan hashes dari dua buah *reatained shared secret* milik *responder*. Sedangkan *auxsecretIDr* adalah hash dari *auxsecret* milik *responder*. Yang terakhir, *pbxsecretIDr* merupakan hash dari *shared secret pbxsecret* milik PBX.

2.9.2.6 *DHPart2 Message*

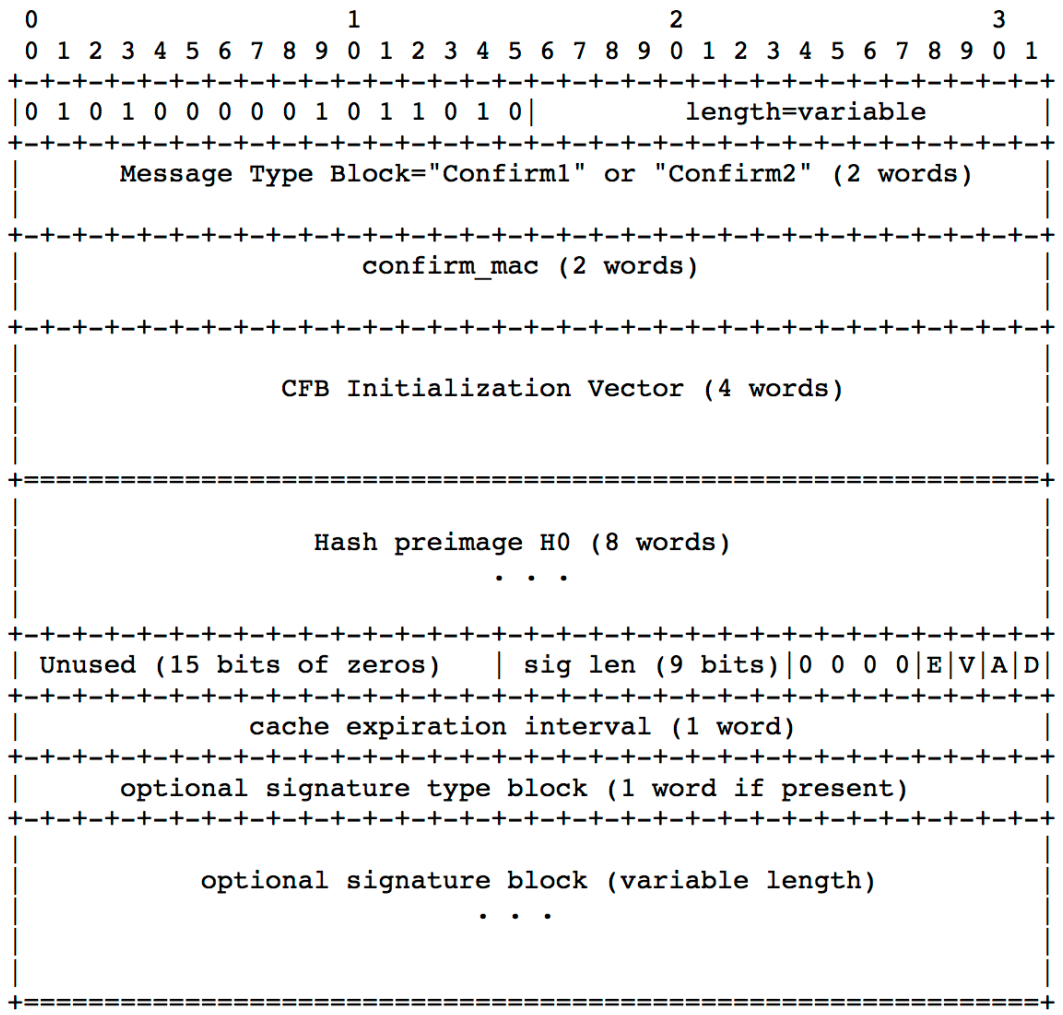
DHPart2 Message melengkapi dan mengakhiri dari proses DH *exchange*. Message ini dikirim oleh *initiator* ketika DHPart1 *message* telah diterima dari responder. Panjang dari nilai *pvi* dan panjang dari *DHPart2* message bergantung kepada *Key Agreement Type* yang dipilih [3].



Gambar 2. 14 Format *DHPart2 Message*

2.9.2.7 Confirm1 dan Confirm2 Messages

Confirm1 message dikirim oleh responder sebagai respon dari DHPart2 message setelah SRTP session key dan paramternya dinegosiasikan. Confirm2 message dikirim oleh intiator sebagai respon dari Confirm1 message [3]. Berikut adalah format dari Confirm1 dan Confirm2 message;



Gambar 2. 15 Format Confirm1 dan Confirm2 Message

Message berisi *Message Type Block* "*Confirm1*" dan "*Confirm2*". Selanjutnya terdapat *confirm_mac*, adalah MAC hasil komputasi dari seluruh bagian *message* yang dienkripsi. Lalu terdapat 16-oktet berisi *CFB Initialization Vector*. Sisa dari isi pesan selanjutnya di enkrisi oleh CFB dan dilindungi oleh *confirm_mac*.

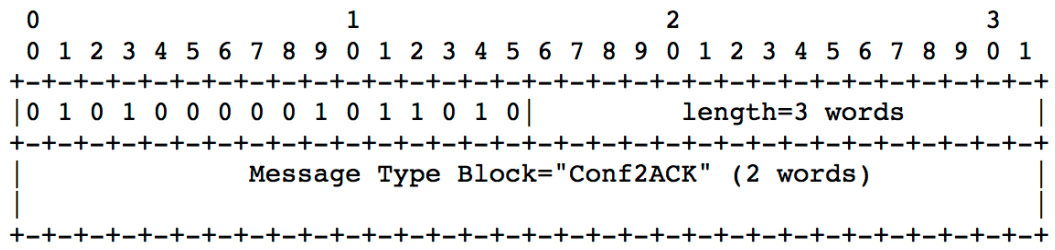
Bagian pertama dari wilayah yang dienkripsi adalah *hash preimage H0.* Lalu terdapat 15 bit yang tidak boleh digunakan dan harus bernilai nol (0) ketika dikirim dan diabaikan ketika menerima *Confirm1* dan *Confirm2 message*.

9 bit selanjutnya berisi panjang dari *signature*. Jika tidak terdapat *SAS signature* maka seluruh bit bernilai nol. Jika tidak terdapat *signature*, maka panjang keseluruhan dari *Confirm1* atau *Confirm2 message* akan bernilai 19 *words*.

Setelah itu akan terdapat *flag* yang memiliki panjang 8 bit. *Flag* yang tidak didefinisikan akan bernilai nol. Terdapat 4 flag yaitu *flag* (E) adalah *PBX enrollment*, *flag* (V) adalah *SAS verified*, *flag* (A) adalah *Allow Clear*, dan *flag* (D) adalah *Disclosure*. Seluruh *flag* tersebut memiliki tipe *Boolean*.

2.9.2.8 *Conf2ACK Message*

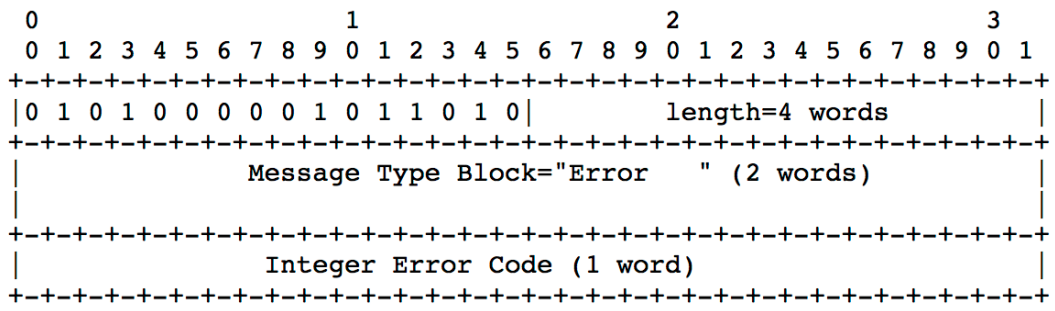
Conf2ACK Message dikirim oleh responder sebagai response dari *Confirm2 message* yang valid. Penerima dari *Conf2ACK* menghentikan transmisi dari *Confirm2 message*. SRTP media (dengan *SRTP auth tag* yang telah divalidasi) pertama dari *responder* juga dapat menghentikan transmisi *Confirm2 message*. [3]



Gambar 2. 16 Format Conf2ACK Message

2.9.2.9 *Error Message*

Error message dikirim untuk mengakhiri proses ZRTP *key agreement exchange* yang error [3]. Berikut adalah format dari error message;



Gambar 2. 17 Format *Error Message*

Pada Tabel akan dijelaskan nilai hexadecimal dari *Error Code*;

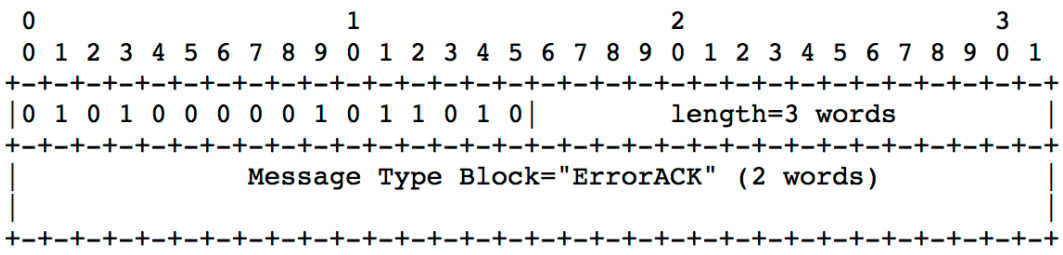
Tabel 2. 11 Kode Error ZRTP

Error Code	Pengertian
0x10	*Malformed Packet* (CRC Ok, tapi struktur nya salah)
0x20	*Critical Software Error*
0x30	Versi ZRTP tidak didukung
0x40	Komponen *Hello* tidak cocok
0x51	*Hash Type* tidak didukung
0x52	*Cipher Type* tidak didukung
0x53	*Public Key Exchange* tidak didukung
0x54	*SRTP auth tag* tidak didukung
0x55	*SAS rendering scheme* tidak didukung
0x56	Tidak tersedia *shared secret*, diperlukan DH *mode*
0x61	*DH error: bad pvi or pvr (== 1, 0, or p-1)*
0x62	*DH Error: hvi != hashed data*
0x63	Menerima relay SAS dari MiTM yg tidak dikenal
0x70	*Auth Error: Bad Confirm pkt MAC*
0x80	Tidak dapat digunakan kembali

Error Code	Pengertian
0x90	Nilai ZIDs sama pada Hello
0x91	*SSRC collision*
0xA0	Service tidak tersedia
0xB0	*Protocol Timeout error*
0x100	*GoClear message* diterima, tapi tidak diperbolehkan

2.9.2.10 *ErrorACK Message*

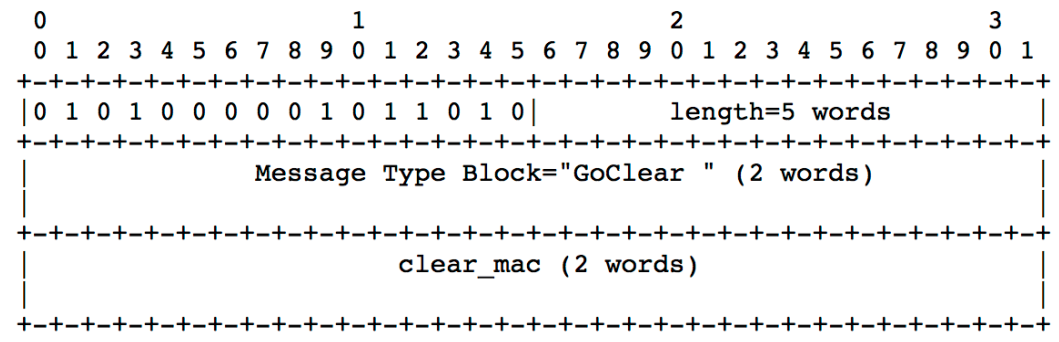
ErrorACK Message dikirim sebagai respon dari *Error Message*. Penerima dari *ErrorACK* menghentikan transmisi *Error message* [3]. Format nya sebagai berikut;



Gambar 2. 18 Format *ErrorACK message*

2.9.2.11 *GoClear Message*

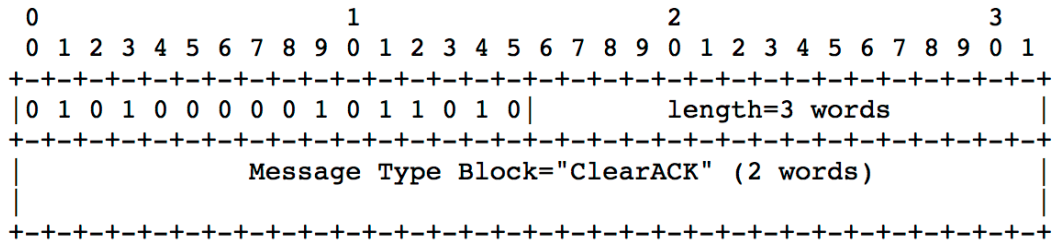
Penggunaan dari *GoClear Message* adalah opsional, *GoClear Message* berfungsi untuk memberitahukan perpindahan dari SRTP ke RTP. *clear_mac* digunakan untuk meng-authentikasi *GoClear Message* sehingga *GoClear messages* palsu yang dikirimkan oleh penyerang dapat dideteksi dan diabaikan.



Gambar 2. 19 Format *GoClear Message*

2.9.2.12 *ClearACK Message*

Penggunaan *ClearACK message* adalah opsional, *ClearACK message* berfungsi untuk mengirim konfirmasi bahwa *GoClear* telah diterima. *ClearACK* dikirim hanya ketika *clear_mac* dari *GoClear message* telah diauthentikasi [3].



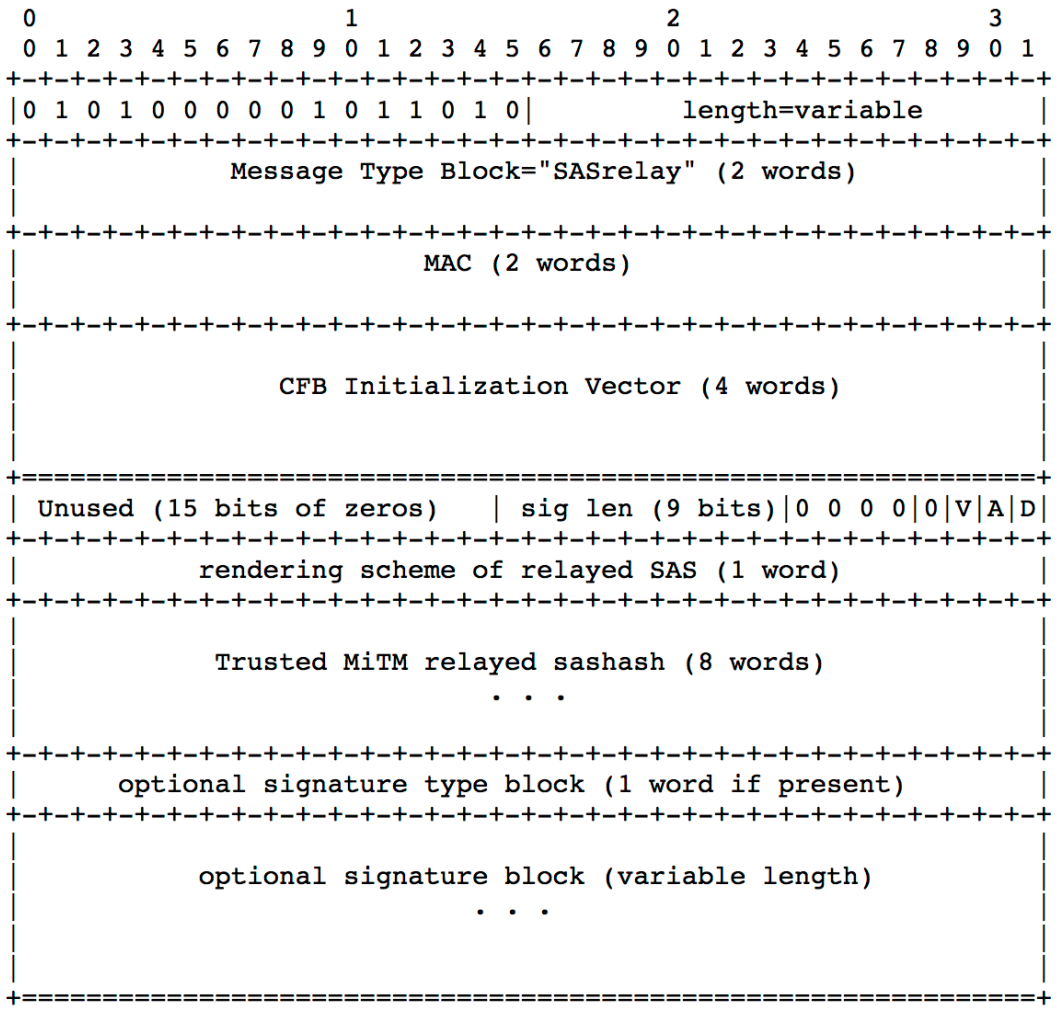
Gambar 2. 20 Format *ClearACK message*

2.9.2.13 *SASrelay Message*

SASrelay message dikirim oleh *MiTM* yang telah dipercaya, biasanya PBX. Message ini tidak dikirim sebagai respon dari paket, tapi dikirim sebagai *self_initiated packet* yg dikirim oleh *MiTM* yg telah dipercaya. Message ini hanya akan dikirim ketika seluruh ZRTP *key negotiation* telah selesai. Tepatnya setelah *Confirm message* dan ACK-nya. Message ini juga hanya akan dikirim setelah *MiTM* menyelesaikan *key negotiation* dengan kelompok yang lain.

Ketika *MiTM* (PBX) mengirim *SASrelay message*, *MiTM* sebelumnya harus mendeklarasikan dirinya sebagai *MiTM device* dengan memberi nilai MiTM *flag* (M) pada *Hello message*. Jika penerima dari *SASrelay message* sebelumnya belum menerima *Hello message* dengan *flag* M, maka SAS yg telah di-relay tidak

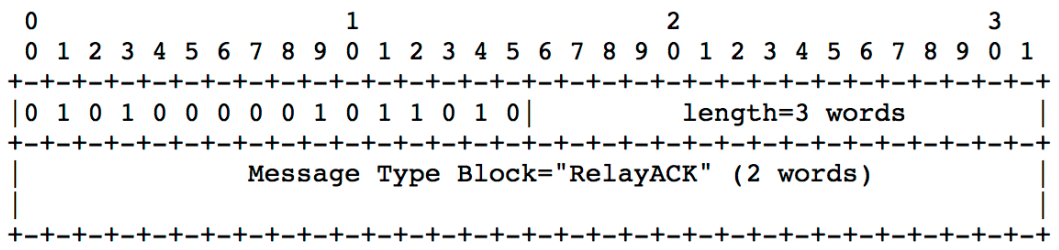
akan di-render. *RelayACK* tetap dikirim, tetapi tidak ada *error message* yang akan dikirim[3].



Gambar 2. 21 Format *SASrelay Message*

2.9.2.14 *RelayACK Message*

RelayACK message dikirim sebagai respon dari *SASrelay message* yg valid. Penerima dari *RelayACK* menghentikan transmisi dari *SASrelay message*.



Gambar 2. 22 Format *RelayACK Message*

2.9.3 *Short Authentication String (SAS)*

Mekanisme ini digunakan untuk mendeteksi jika terjadi sesuatu saat proses *key negotiation* dan memberikan fungsi agar pengguna dapat mengecek integritas dari suatu sesi. Pada implementasinya, SAS ditampilkan berupa text yang harus dikonfirmasi oleh kedua pengguna melalui suara pada saat memulai percakapan. Cara terbaik untuk mengkonfirmasinya yaitu pemanggil membaca dua karakter awal dari SAS, penerima membaca dua karakter akhir dari SAS. Jika hasilnya cocok, maka komunikasi telah diamankan, informasi ini akan disimpan untuk Sesi ZRTP selanjutnya.

SAS didapatkan dari hasil hash DH *exchange,* sehingga kata pada SAS akan pendek. Untuk menembus SAS, penyerang harus mendapatkan hash dari banyak kemungkinan. Sebagai contoh 16 bit SAS, akan menghasilkan 65536 kemungkinan untuk bisa ditebak oleh penyerang.

Hanya akan terdapat satu SAS yg dikomputasi dalam setiap panggilan. SAS harus di-render kepada user sebagai authentikasi. Rendering nilai SAS yg ditampilkan pada kedua endpoint tergantung dari SAS type yang disetujui pada Commit Message.

2.9.3.1 *SAS Verified flag*

SAS Verified flag (V) diberikan nilai ketika user telah berhasil melakukan perbandingan nilai *SAS*. *SAS verified flag* ditukar secara aman melalui *Confirm1* dan *Confirm2 message* pada sesi berikutnya. Lebih jelasnya, setiap *endpoint* mengirim *SAS verified flag* dari sesi sebelumnya dan dikirim melalui *Confirm*

message pada sesi yang sedang terjadi. Terlepas dari ada atau tidaknya ketersediaan user interface untuk memberi nilai pad *SAS verified flag*, *shared secret* tetap perlu disimpan untuk mengurangi kemungkinan penyerangan pada sesi panggilan selanjutnya.

2.9.3.2 Signing SAS

ZRTP memperbolehkan menggunakan metode *signature* lainya, yang berarti memperbolehkan SAS untuk diverifikasi tanpa partisipasi langsung dari pengguna. *Signature* tersebut dapat dikirim bersamaan dengan SAS pada *Confirm1, Confirm2*, atau *SASrelay messages*. *Signature type, panjang signature*, dan kunci untuk membuat *signatur*e dikirim bersamaan saat *signature* dikirim. *Signature* dikalkulasi berdasarkan seluruh hasih *SAS hash (sashash)*. Pertukaran *signature* terjadi pada *Confirm1, Confirm2 atau SASrelay message* yang telah dienkripsi. Proses tersebut dapat digunakan untuk mengautentikasi *ZRTP exchange*. *Signature* hanya dapat dikirim pada *DH mode*, bukan pada *multistream mode*.