BAB 2

LANDASAN TEORI

2.1 Kriptografi

Kata kriptografi berasal dari bahasa Yunani, yaitu *crypto*s yang berarti tersembunyi atau rahasia dan *graphien* yang berarti tulisan. Kriptografi merupakan ilmu dan seni dalam matematika yang dipelajari dengan tujuan untuk menjaga keamanan dan kerahasiaan suatu informasi (Schneier, 1996). Adapun aspek layanan keamanan informasi yang dapat diberikan oleh kriptografi adalah *confidentiality* (kerahasiaan), *data integrity* (integritas data), *authentication* (otentikasi), dan *non-repudiation* (nir-penyangkalan) (Menezes, et al., 1997). Aspek-aspek tersebut dijelaskan lebih lanjut sebagai berikut:

1. *Confidentiality* (kerahasiaan) yaitu layanan yang memberikan jaminan agar pesan yang dikirimkan tetap rahasia dan tidak dapat diketahui oleh pihak lain yang tidak berhak. Umumnya hal ini dilakukan dengan menggunakan teknik enkripsi, yaitu mengubah bentuk pesan semula (*plaintext*) menjadi pesan pesan tersandi (*ciphertext*).
2. *Data integrity* (keutuhan data) yaitu layanan yang mampu mengenali atau mendeteksi adanya modifikasi terhadap informasi yang dikirimkan (penghapusan, pengubahan atau penambahan) oleh pihak lain.

3. *Authentication* (otentikasi) yaitu layanan yang berhubungan dengan identifikasi. Kriptografi memiliki kemampuan untuk mengidentifikasi pihak-pihak yang terlibat dalam pengiriman informasi (*entity authentication*) maupun mengidentifikasi keaslian sumber informasi (*data origin authentication*).
4. *Non-repudiation* (nir-penyangkalan) yaitu layanan yang dapat mencegah suatu entitas melakukan penyangkalan terhadap aksi yang dilakukan dalam komunikasi.

Dalam peneltian ini, algoritma kriptografi yang akan digunakan adalah stream cipher Rabbit, namun untuk melihat hasil perbedaan dari algoritma yang sudah menjadi default cipherlist OpenSSL, maka akan dilakukan perbandingan dengan algoritma RC4 dan AES. Berikut ini penjelasan dari algoritma yang terkait dalam penelitian ini:

2.1.1 Algoritma Rabbit

Algoritma Rabbit dipublikasikan pertama kali pada *Fast Software Encryption Workshop* 2003. Algoritma ini merupakan algoritma stream cipher yang menggunakan kunci (*secret key*) dengan panjang 128-bit dan 64- bit IV sebagai input. Hasil kombinasi *internal state*-nya menghasilkan output berupa rangkaian bit semi acak (*pseudorandom bit*) dengan panjang 128-bit per blok. Proses enkripsi/dekripsi dilakukan dengan men-XOR-kan rangkaian bit semi acak tersebut dengan plaintext/ciphertext. *Internal state*-nya berjumlah 513 bit dibagi ke dalam delapan 32-bit variabel *state*, delapan 32-bit *counter* dan

satu *counter carry bit.* Kedelapan variabel *state* di-*update* oleh fungsi non-linear masing-masing pasangan variabel *state* tersebut. (Boesgard, 2003)

Algoritma Rabbit didesain agar efisien dalam implementasi pada software serta mengimbangi ukuran kunci sepanjang 128 bit untuk mengenkripsi sampai dengan 2^{64} blok plaintext. Ini berarti bahwa untuk melakukan *attack* tanpa mengetahui rangkaian kunci, maka pihak penyerang harus menentukan sampai dengan 2^{64} blok output *ciphertext* atau melakukan *exhaustive key search* sebanyak 2^{128} kombinasi kunci.

Spesifikasi Algoritma

a. Notasi

Notasi yang digunakan adalah sebagai berikut : $\oplus$

melambangkan logika XOR, & melambangkan logika AND, << dan >> melambangkan operasi pergeseran bit ke kiri dan kanan, <<< dan >>> melambangkan operasi rotasi bit ke kiri dan kanan, dan $\lozenge$ melambangkan penggabungan (*concatenation*) antara dua buah rangkaian bit. $A^{g...h}$ melambangkan bit urutan ke *g* sampai dengan *h* dari variabel *A*.

Internal state dari algoritma ini berjumlah 513 bit. 513 bit dibagi ke dalam delapan variabel *state* masing-masing sebanyak 32 bit $x_{j,i}$ dan delapan variabel 32 bit *counter* $c_{j,i}$,

dimana $x_{j,i}$ merupakan variabel *state* dari subsistem *j* pada iterasi *i*, dan $c_{j,i}$ merupakan variabel *counter* yang berkorespondensi. Terdapat bit carry *counter* $ø_{,7,i}$, dimana variabel *counter carry* ini digunakan untuk menyimpan bit *carry* antar iterasi. Bit *counter carry* ini mempunyai nilai awal 0. Delapan variabel *state* dan *counter* didapatkan dari kunci pada saat inisialisasi.

b. Skema Key Setup

Proses inisialisasi dimulai dengan mengekspansi 128-bit kunci menjadi delapan variabel *state* dan delapan *counter* sehingga terdapat korespondensi satu-satu antara kunci dengan variabel *initial state* $x_{j,i}$ dan *initial counter* $c_{j,i}$

Kunci $K^{[127..0]}$, dibagi ke dalam delapan *subkey*: $k_0 = K^{[15..0]}$,

$$k_1 = K^{[31..16]}, \ldots, k_7 = K^{[127..112]}.$$

Variabel *state* dan *counter* diinisialisasikan dari *subkey* sebagai berikut :

$k_{(j+1 \bmod 8)} \lozenge k_j$	untuk	j	bilangan genap
$k_{(j+5 \bmod 8)} \lozenge k_{(j+4 \bmod 8)}$	untuk	j	bilangan ganjil
	dan		
$k_{(j+4 \bmod 8)} \lozenge k_{(j+5 \bmod 8)}$	untuk	j	bilangan ganjil
$k_j \lozenge k_{(j+1 \bmod 8)}$	untuk	j	bilangan ganjil

Iterasi ini dilakukan sebanyak empat kali menggunakan *next-state function* untuk menghilangkan korelasi antara bit-bit pada kunci dan bit-bit pada variabel *internal state*. Kemudian, variabel *counter* diinisialisasi ulang berdasarkan persamaan berikut :

$$c_{j,4} = c_{j,4} \oplus x(j+4 \bmod 8), 4$$

untuk mencegah didapatkannya kembali kunci dengan memasukkan variabel *counter*.

c. Skema *Initial Value* (IV) Setup

Internal state setelah proses *key setup* dilambangkan sebagai *master state*, dan variabel *master state* ini diubah berdasarkan skema IV. Skema IV *Setup* bekerja dengan men-XOR 64 bit IV pada seluruh 256 bit *counter state*. Ke-64 bit IV dinotasikan dalam $IV^{[63..0]}$. *Counter* diubah berdasarkan persamaan berikut :

$$c_{0,4} = c_{0,4} \oplus IV^{[31\ldots0]} \qquad c_{1,4} = c_{1,4} \oplus IV^{[63\ldots48]} \lozenge IV^{[31\ldots16]}$$

$$c_{2,4} = c_{2,4} \oplus IV^{[63\ldots32]} \qquad c_{3,4} = c_{3,4} \oplus IV^{[47\ldots32]} \lozenge IV^{[15\ldots0]}$$

$$c_{4,4} = c_{4,4} \oplus IV^{[31\ldots0]} \qquad c_{5,4} = c_{5,4} \oplus IV^{[63\ldots48]} \lozenge IV^{[31\ldots16]}$$

$$c_{6,4} = c_{6,4} \oplus IV^{[63\ldots32]} \qquad c_{7,4} = c_{7,4} \oplus IV^{[47\ldots32]} \lozenge IV^{[15\ldots0]}$$

Iterasi dilakukan sebanyak empat kali untuk membuat seluruh bit yang bergantung secara nonlinear pada seluruh IV. Pengubahan variabel pada *counter* oleh IV menjamin bahwa seluruh 2^{64} IV akan menghasilkan rangkaian kunci yang unik.

d. *Next-state Function*

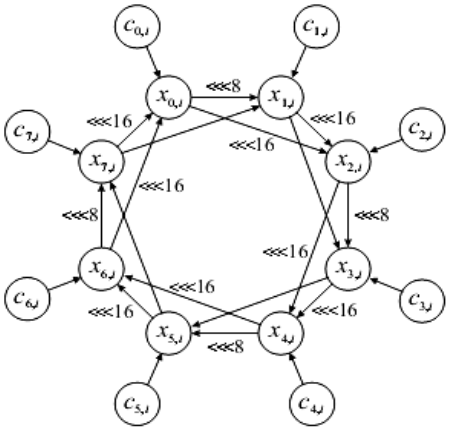
Fungsi inti dari algoritma Rabbit adalah fungsi *g* dengan persamaan sebagai berikut :

$$g_{j,i} = ((x_{j,i} + c_{j,i})^2 \oplus ((x_{j,i} + c_{j,i+1})^2 >> 32)) \bmod 2^{32}$$

dengan

$$\begin{aligned}
x_{0,i+1} &= g_{0,i} + (g_{7,i} <<< 16) + (g_{6,i} <<< 16) \\
x_{2,i+1} &= g_{1,i} + (g_{0,i} <<< 8) + g_{7,i} \\
x_{2,i+1} &= g_{2,i} + (g_{1,i} <<< 16) + (g_{0,i} <<< 16) \\
x_{3,i+1} &= g_{3,i} + (g_{2,i} <<< 8) + g_{1,i} \\
x_{4,i+1} &= g_{4,i} + (g_{3,i} <<< 16) + (g_{2,i} <<< 16) \\
x_{5,i+1} &= g_{5,i} + (g_{4,i} <<< 8) + g_{3,i} \\
x_{6,i+1} &= g_{6,i} + (g_{5,i} <<< 16) + (g_{4,i} <<< 16) \\
x_{7,i+1} &= g_{7,i} + (g_{6,i} <<< 8) + g_{5,i}
\end{aligned}$$

dimana seluruh operasi penambahan merupakan modulo 2^{32}. Ilustrasi dari operasi-operasi fungsi di atas ditunjukkan pada gambar di bawah ini.



Gambar 2. 1. Skema Next-state Function pada Rabbit

(Sumber : Boesgard, M. et al. 2003. *The Stream Cipher Rabbit)*

e. Counter System

$$c_{0,i+1} = c_{0,i} + a_0 + \phi_{7,i} \bmod 2^{32}$$

$$c_{1,i+1} = c_{1,i} + a_1 + \phi_{0,i+1} \bmod 2^{32}$$

$$c_{2,i+1} = c_{2,i} + a_2 + \phi_{7,i+1} \bmod 2^{32}$$

$$c_{3,i+1} = c_{3,i} + a_3 + \phi_{7,i+1} \bmod 2^{32}$$

$$c_{4,i+1} = c_{4,i} + a_4 + \phi_{7,i+1} \bmod 2^{32}$$

$$c_{5,i+1} = c_{5,i} + a_5 + \phi_{7,i+1} \bmod 2^{32}$$

$$c_{6,i+1} = c_{6,i} + a_6 + \phi_{7,i+1} \bmod 2^{32}$$

$$c_{7,i+1} = c_{7,i} + a_7 + \phi_{7,i} \bmod 2^{32}$$

dimana bit *carry counter* $\phi_{j,i+1}$ adalah sebagai berikut :

$$\phi_{j,i+1} = 1 \begin{cases} 1 \text{ jika } c_{0,i} + a_0 + \phi_{7,i} \geq 2^{32} \wedge j = 0 \\ \text{jika } c_{j,i} + a_0 + \phi_{j-1,i+1} > 2^{32} \wedge j > 0 \\ 0 \text{ jika lainya} \end{cases}$$

dengan konstanta a_j didefinisikan sebagai berikut :

a_0 = 0x4D34D34D $\qquad$ a_1 = 0xD34D34D3

a_2 = 0x34D34D34 $\qquad$ a_3 = 0x4D34D34D

a_4 = 0x4D34D34D a_5 = 0x34D34D34

a_6 = 0x4D34D34D a_7 = 0xD34D34D3

f. Skema Ekstraksi

Setelah proses tiap iterasi, outputnya diekstrak sesuai dengan persamaan berikut :

$$s_i^{[15..0]} = x_{0,i}^{[15..0]} \oplus x_{5,i}^{[31..16]} \qquad s_i^{[31..16]} = x_{0,i}^{[31..16]} \oplus x_{3,i}^{[15..0]}$$

$$s_i^{[47..32]} = x_{2,i}^{[15..0]} \oplus x_{7,i}^{[31..16]} \qquad s_i^{[63..48]} = x_{2,i}^{[31..16]} \oplus x_{5,i}^{[15..0]}$$

$$s_i^{[79..64]} = x_{4,i}^{[15..0]} \oplus x_{1,i}^{[31..16]} \qquad s_i^{[95..80]} = x_{4,i}^{[31..16]} \oplus x_{7,i}^{[15..0]}$$

$$s_i^{[111..96]} = x_{6,i}^{[15..0]} \oplus x_{3,i}^{[31..16]} \qquad s_i^{[127..112]} = x_{6,i}^{[31..16]} \oplus x_{1,i}^{[15..0]}$$

dimana

s_i merupakan blok rangkaian kunci 128 bit pada iterasi ke-*i*.

g. Skema Enkripsi/dekripsi

Rangkaian bit yang telah diekstrak sebelumnya di-XOR dengan *plaintext/ciphertext*.

$$c_i = p_i \oplus s_i$$

$$p_i = c_i \oplus s_i$$

dimana c_i dan p_i merupakan rangkaian bit *ciphertext/plaintext* pada iterasi ke-*i*.

2.1.2 Algoritma AES

Advanced Encryption Standard. AES[1] adalah algoritma simetrik berbasis *block cipher* yang mengenkripsi/mendekripsi blok berukuran 128 bit dengan panjang kunci yang beragam, yaitu 128, 192, dan 256 bit. Sehingga dikenal tiga tipe AES berdasarkan panjang kuncinya, yaitu AES-128, AES-192, dan AES-256. AES tidak berorientasi bit melainkan berorientasi byte sehingga implementasi AES ke hardware maupun software menjadi efektif. AES adalah teknik enkripsi kunci simetris yang menggantikan Data Encryption Standard (DES). Algoritma Rijndael,dibuat oleh dua kriptologis Belgia, Vincent Rijmen and Joan Daemen, memenangkan kompetisi yang diadakan oleh NIST sehingga algoritma tersebut ditetapkan sebagai AES.

AES menyediakan enkripsi yang kuat dan telah dipilih oleh NIST sebagai Federal Information Processing Standard pada November 2001 (FIPS-197), dan pada Juni 2003 Pemerintah A.S. (NSA) mengumumkan bahwa AES cukup aman untuk melindungi informasi rahasia sampai pada tingkat TOP SECRET, yang merupakan tingkat kemanan tertinggi dan didefinisikan sebagai informasi yang akan menyebabkan "kerusakan yang sangat fatal

[1] Comparation And Analysis Of Performance Of Encryptiondecryption Of Text Using Aes Algorithm And Modified Aes Based On Android

(*exceptionally grave damage*)" terhadap keamanan nasional jika informasi tersebut sampai ke publik/ masyarakat.

Algoritma menggunakan salah satu dari tiga kekuatan kunci cipher: 128-, 192-, atau 256-bit kunci enkripsi (password). Setiap ukuran kunci enkripsi menyebabkan algoritma berperilaku sedikit berbeda dengan yang lainnya, jadi ukuran kunci yang lebih besar tidak hanya menawarkan jumlah bit yang lebih banyak dimana dengan hal itu anda dapat mengacak data, tetapi juga meningkatkan kompleksitas dari algoritma cipher.

Mekanisme Algoritma AES (Advanced Encryption Standard) 256 bit[2]

Garis besar Algoritma Rijndael yang beroperasi pada blok 128-bit dengan kunci 128- bit adalah sebagai berikut (di luar proses pembangkitan round key):

a. AddRoundKey

melakukan XOR antara state awal (plainteks) dengan cipher key. Tahap ini isebut juga initial round.

b. Putaran sebanyak Nr – 1 kali. Proses yang dilakukan pada setiap putaran adalah :

- ***Bytes***: substitusi byte dengan menggunakan tabel substitusi (S-box).

[2] Penerapan Algoritma Aes : Rijndael Dalam Pengenkripsian Data Rahasia (Dedi Alyanto, 2016)

- ***ShiftRows***: pergeseran baris-baris array state secara wrapping.
- ***MixColumns***: mengacak data di masing-masing kolom array state.
- ***AddRoundKey***: melakukan XOR antara state sekarang round key.

c. *Final round*

proses untuk putaran terakhir:

- *SubBytes*
- *ShiftRows*
- *AddRoundKey*

Dari penjelasan secara umum diatas, bahwasanya algoritma AES ini merupakan block cipher yang memiliki banyak operasinya dimana *plaintext* yang menjadi input pada algoritma ini diproses melalui beberapa *round* (putaran), dan tiap-tiap *round* tersebut terdiri dari beberapa operasi, yaitu substitusi S-Box, pergeseran, multiplikasi dan operasi XOR. Keseluruhan proses ini tentunya berdampak pada performa algoritma ini sehingga relatif lebih lambat dibandingkan dengan algoritma yang akan diusulkan oleh penulis, yaitu *stream cipher* Rabbit.

Peneliti melakukan tinjauan keamanan dan kelebihan algoritma Rabbit serta mengapa algoritma Rabbit digunakan dalam penelitian ini? Karena Desain algoritma Rabbit merupakan stream cipher dengan desain baru.

Desain algoritma ini menggunakan proses pencampuran (*mixing*) *inner state* secara non-linear antara dua iterasinya. Berbeda dengan desain-desain algoritma pada umumnya, algoritma Rabbit tidak menggunakan *linear feedback shift register* ataupun S-box. Desain ini memiliki beberapa konsekuensi penting.

a. Desain yang Kompak

Desain algoritma Rabbit sangat kompak. Seluruh operasi aritmatika yang digunakan pada algoritma ini didukung oleh prosesor modern, sehingga algoritma ini memiliki kecepatan pemrosesan yang sangat tinggi pada berbagai *platform*. Berbeda dengan desain algoritma stream cipher pada umumnya, algoritma ini juga memiliki kecepatan IV serta *key setup* yang tinggi.

Algoritma Rabbit tidak menggunakan S-box atau operasi pada *GF* (2^n). Oleh karena itu tidak dibutuhkan *look up table,* sehingga ruang memori yang dibutuhkan baik dalam implementasi pada *hardware* maupun *software* sangat kecil. Tabel 2.1 merupakan performa (dalam *clock cycles* atau *clock cycles per byte*), ukuran kode (*code size*) dan jumlah memori (dalam *byte*) yang dibutuhkan masing-masing untuk proses enkripsi, *key setup* dan IV *setup* pada prosesor Intel Pentium[3] (Boesgaard et al, 2003).

[3] Pengujian dilakukan pada prosesor Intel Pentium III 1,0 GHz dan Intel Pentium 4 1,7 GHz. Kode implementasi Rabbit diprogram menggunakan kombinasi bahasa *assembly* dengan C, dikompilasi menggunakan Intel C 7.1 *compiler*.

Tabel 2. 1. Performa, ukuran kode dan jumlah memori yang dibutuhkan oleh Rabbit pada prosesor Pentium.

Prosesor	Performa	Ukuran Kode	Memori
Pentium III	3,7/278/253	440/617/720	40/36/44
Pentium IV	5,1/480/749	698/516/762	16/36/28

Selain itu, yang perlu dijadikan catatan, pada prosesor modern jumlah *inner state*-nya tidak melebihi ruang pada register prosesor tersebut, sehingga tidak membutuhkan akses ke memori utama yang kurang efisien secara komputasi (membutuhkan waktu dan perhitungan tambahan).

b. Keamanan

Algoritma Rabbit telah dievaluasi secara ekstensif terhadap seluruh kemungkinan *attack* yang dapat dilakukan, baik stream maupun block cipher kriptanalisis. Desain Rabbit yang berbeda membuat jenis *attack* umum yang dapat dilakukan terhadap stream cipher menjadi tidak aplikatif. Baik *algebraic attack* maupun *correlation attack* tidak dapat diaplikasikan terhadap Rabbit. *Time- Memory-Data tradeoff* juga tidak dapat diaplikasikan terhadap Rabbit karena jumlah *internal state* yang cukup besar, yakni 513 bit. Para pendesain algoritma ini yakin bahwa untuk melakukan serangan terhadap Rabbit harus dilakukan berdasarkan suatu metode *attack* yang sama sekali baru. Dari hasil pengujian serta analisa, disimpulkan bahwa algoritma Rabbit resistan terhadap metode-metode kriptanalisa yang ada saat ini.

2.2 Network Related

Dalam penelitian ini, penulis telah melakukan studi pustaka yang terkait. Adapun teori-teori yang menjadi dasar serta mendukung penelitian ini sebagai berikut:

Open Systems Interconnection (OSI) Model

Open System Interconnection (OSI) model adalah suatu model standar komunikasi yang dibuat oleh *International Standards Organization* (ISO) dan pertama kali diperkenalkan pada tahun 1970. Tujuan dari OSI *model* adalah untuk menunjukkan pemodelan komunikasi antar sistem yang berbeda tanpa memerlukan adanya perubahan logika pada *hardware* dan *software*nya. OSI *model* bukan merupakan suatu protokol, melainkan hanya merupakan pemodelan untuk memudahkan dalam memahami dan mendesain arsitektur jaringan yang fleksibel. Terdiri dari tujuh *layer*, masing-masing mendefinisikan bagian proses perpindahan informasi yang melalui suatu jaringan (Forouzan, 2010). Ketujuh *layer* tersebut adalah sebagai berikut:

1. *Physical Layer*

   Merupakan *layer* yang berfungsi untuk mengatur proses dalam membawa *bit stream* melalui media fisik. Bergantung pada spesifikasi mekanik dan elektronik pada *interface* dan media transmisinya. Selain itu juga fokus pada representasi bit, *data rate*, sinkronisasi bit, konfigurasi *line*, topologi fisik, dan mode transmisi.

2. *Data Link Layer*

Merupakan *layer* yang berfungsi untuk mengubah *physical layer* menjadi suatu *link*. Menjamin tidak adanya *error* ketika akan naik ke *layer* diatasnya yaitu *network layer*. Selain itu juga bertanggungjawab terhadap *framing, physical addressing, flow control, error control,* dan *access control*.

3. *Network Layer*

   Merupakan *layer* yang bertanggungjawab terhadap pengiriman paket dari sumber ke tujuan. Disamping itu juga bertanggung jawab terhadap *logical addressing* dan *routing*.

4. *Transport Layer*

   Merupakan *layer* yang bertanggungjawab terhadap proses-proses pengiriman pesan. Proses tersebut adalah berupa program aplikasi yang dijalankan oleh *host*. *Layer* ini juga memiliki kemampuan untuk mengenali hubungan antar paket yang dikirimkan. Selain itu juga bertanggungjawab pada *service-point addressing, segmentation and reassembly, connection control, flow control,* dan *error control*.

5. *Session Layer*

   Merupakan dialog controller pada suatu jaringan. Membangun, mengelola, dan melakukan sinkronisasi antar sistem yang sedang berkomunikasi.

6. *Presentation Layer*

Merupakan *layer* yang berfokus pada sintaks dan penerjemahan informasi yang dipertukarkan antara dua sistem. Lebih spesifiknya yaitu *translation*, *encryption*, dan *compression*.

7. *Application Layer*

   Merupakan layer yang memberikan otoritas untuk dapat mengakses sumber daya pada jaringan. Menyediakan user interface dan mendukung adanya layanan email, remote file access, shared database management, dan layanan lainnya.

Gambar 2. 2. OSI *Model* (Forouzan, 2010)

Prototipe VPN *gateway* yang dibuat pada penelitian ini menerapkan teknologi SSL VPN yang bekerja pada *layer* 4 OSI *model.*

2.2.1 Virtual Private Network (VPN)

Virtual Private Network atau VPN adalah suatu jaringan privat yang menggunakan infrastruktur telekomunikasi publik seperti jaringan internet untuk saling bertukar informasi. VPN banyak digunakan oleh suatu organisasi atau perusahaan yang membutuhkan akses eksternal menuju ke jaringan internalnya secara aman. Layanan keamanan informasi yang paling mendasar untuk

diimplementasikan pada VPN adalah enkripsi, otentikasi, dan integritas data. Sehingga seluruh data yang ditransmisikan melalui VPN biasanya terenkripsi untuk mengantisipasi adanya lawan dengan akses ke internet dapat melakukan *eavesdropping* terhadap data yang dipertukarkan melalui jaringan publik. Tanpa adanya otentikasi, pihak-pihak atau entitas yang tidak sah juga dapat bertindak seperti pegawai perusahaan kemudian masuk ke jaringan internal. Integritas data diperlukan karena paket yang dipertukarkan melalui jaringan publik berpotensi besar mengalami perubahan atau modifikasi oleh pihak yang tidak memiliki otoritas.

Beberapa protokol atau teknologi VPN yang populer digunakan, yaitu (HKSAR, 2008):

1. Pada *data link layer* atau *layer* 2 OSI *model* terdapat *Point-to- Point Tunneling Protocol* (PPTP), *Layer 2 Tunneling Protocol* (L2TP), dan *Layer 2 Forwarding* (L2F).
2. Pada *network layer* atau *layer* 3 OSI *model* terdapat IP *Security* (IPsec).
3. Pada *transport layer* atau *layer* 4 OSI *model* terdapat *Secure Sockets Layer* (SSL) atau *Transport Layer Security* (TLS).

Terdapat tiga model arsitektur VPN yang sering diimplementasikan, yaitu (Frankel, et al., 2005):

1. *Gateway-to-gateway*

   Model arsitektur ini melindungi komunikasi dua jaringan yang spesifik, seperti jaringan utama di kantor pusat perusahaan dengan jaringan di kantor cabangnya, atau antara dua jaringan perusahaan yang sudah menjalin kerjasama. Model ini juga biasa disebut dengan *site-to-site*.

2. *Host-to-gateway*

   Model arsitektur ini melindungi komunikasi antara satu *host* atau lebih dengan sebuah jaringan spesifik perusahaan. Biasa digunakan untuk memberikan otoritas kepada *host* yang berada di jaringan publik, seperti pegawai yang sedang dalam perjalanan untuk bisa mendapatkan akses ke layanan yang ada di jaringan internal perusahaan (*email* atau *web server*). Model ini juga disebut dengan *remote access*.

3. *Host-to-host*

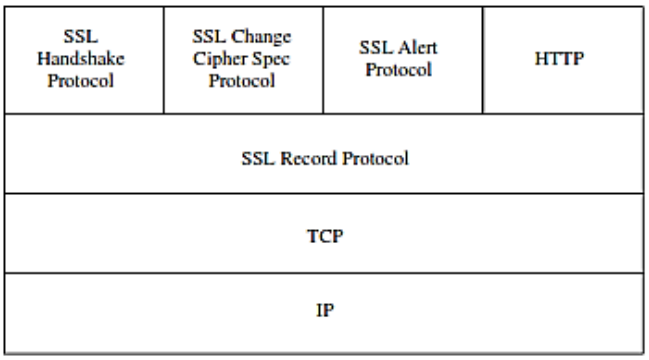
   Model arsitektur ini melindungi komunikasi antara dua komputer yang spesifik. Biasa digunakan pada sebuah jaringan yang jumlah penggunanya relatif sedikit.

Penelitian ini mengimplementasikan model arsitektur VPN *host-to-gateway*. Prototipe VPN *gateway* yang dibangun memiliki kemampuan untuk mengamankan seorang pegawai atau *user* yang

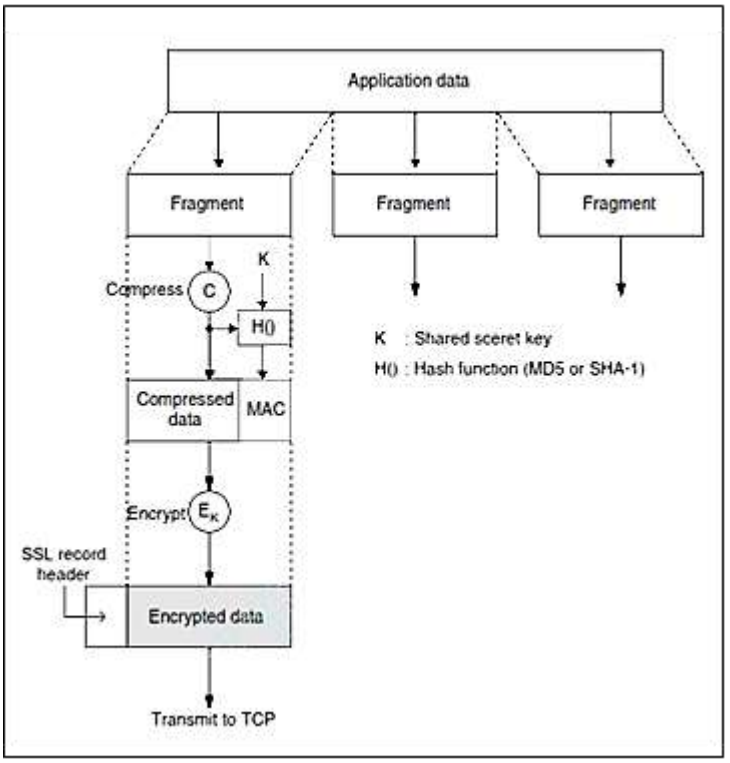
akan melakukan *remote access* ke layanan yang ada di jaringan internal perusahaannya.

2.2.2 Secure Sockets Layer (SSL)

Secure Socket Layer version 3 (SSLv3) diperkenalkan oleh *Netscape Communications Corporation* pada tahun 1995. SSLv3 didesain dengan menerima kritik, saran, dan masukan dari berbagai industri dan dipublikasikan sebagai sebuah dokumen *Internet-Draft*. *Internet Engineering Task Force* (IETF) kemudian membentuk *Transport Layer Security* (TLS) *Working Group* dengan tujuan untuk mengembangkan versi pertama dari TLS sebagai standar protokol keamanan komunikasi melalui internet. Versi pertama dari TLS ini tidak berbeda jauh dengan SSLv3. Protokol TLSv1 menyediakan fitur kerahasiaan komunikasi dan integritas data antara dua entitas yang berkomunikasi melalui internet. Protokol SSL (atau TLS) terdiri atas dua *layer*, yaitu: SSL (atau TLS) *Record Layer* dan SSL (atau TLS) *Handshake Layer* (Rhee, 2003).



Gambar 2. 3. Lapisan Protokol SSL / TLS (Rhee, 2003)



Gambar 2. 4. Operasi pada SSL *Record Layer* (Rhee, 2003)

Secara keseluruhan, operasi-operasi yang ada di dalam SSL *Record Layer*, yaitu:

1. Fragmentasi

   Yaitu pesan dari *layer* yang lebih tinggi akan difragmentasi atau dibagi menjadi blok-blok sepanjang 2^{14} *bytes* atau kurang atau juga disebut dengan SSL*Plaintext records.*

2. Kompresi dan Dekompresi

   Yaitu seluruh *records* akan dikompresi menggunakan suatu algoritma kompresi yang telah didefinisikan sebelumnya. Algoritma kompresi ini akan mengubah struktur SSL*Plaintext* menjadi struktur SSL*Compressed.* Di dalam SSL *Record Protocol*, kompresi ini sifatnya opsional, namun jika akan

diaplikasikan, kompresi ini dilakukan sebelum memasuki proses enkripsi dan komputasi MAC.

3. M*essage Authentication Code* (MAC)

   Komputasi MAC dilakukan sebelum proses enkripsi. Menggunakan kunci *shared secret*, perhitungan MAC adalah sebagai berikut:

   H_1 = *hash*(MAC-*write-secret* || *pad*-1 || *seq-num* || SSL*Compressed.type* || SSL*Compressed.length* || SSL*Compressed.fragment*)

   H = *hash*(MAC-*write-secrte* || *pad*-2 || H_1)

   Keterangan:

MAC-*write-secret*	:	kunci *shared secret*
Hash (H_1 dan H)	:	algoritma fungsi *hash*; MD5 atau SHA-1.
Pad-1	:	byte 0x36 yang diulang sebanyak 48 kali untuk MD5 dan 40 kali untuk SHA-1.
Pad-1	:	byte 0x5C yang diulang sebanyak 48 kali untuk MD5 dan 40 kali untuk SHA-1.

Seq-num	:	sequence number pesan.
SSLCompressed.type	:	protokol dengan level yang lebih tinggi untuk memproses fragmentasi.
SSLCompressed.length	:	panjang fragmen yang dikompresi.
SSLCompressed.fragment	:	fragmen yang dikompresi (fragmen plaintext jika tidak dikompresi).
||	:	simbol concatenation.

Pesan yang terkompresi dan nilai MACnya kemudian dienkripsi dengan enkripsi simetrik. *Block cipher* yang digunakan sebagai algoritma enkripsi, yaitu: DES(56), 3DES (168), IDEA (128), RC5, dan *Fortezza* (80).

4. Menambahkan SSL *Record Header*

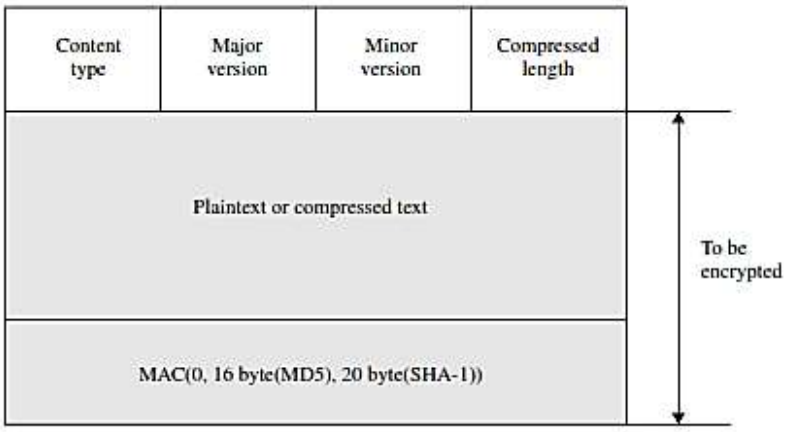
   Merupakan tahap akhir dari SSL *Record Protocol*. Komponen-komponen *header*nya adalah sebagai berikut:

   a. *Content type* (8 bit): merupakan *field* yang mengindikasikan protokol dari *layer* yang lebih tinggi untuk proses fragmentasi.

b. *Major version* (8 bit): merupakan *field* yang mengindikasikan *major version* dari SSL yang digunakan. Sebagai contoh SSLv3, maka nilainya 3.

c. *Minor version* (8 bit): merupakan *field* yang mengindikasikan *minor version* dari SSL yang digunakan. Sebagai contoh SSLv3, maka nilainya 0.

d. *Compressed length* (16 bit): merupakan *field* yang mengindikasikan panjang fragmen *plaintext* atau fragmen *plaintext* yang sudah terkompresi dalam *byte*. Nilai maksimumnya adalah 2^{14} + 2048 *byte*.



Gambar 2. 5. Format SSL *Record Protocol* (Rhee, 2003)

Untuk SSL *Handshake Layer* terdiri dari SSL *Change Cipher Spec Protocol*, *Alert Protocol*, dan *Handshake Protocol*. Penjelasan singkat dari ketiga protokol tersebut adalah sebagai berikut:

1. SSL Change Cipher Spec Protocol

   Merupakan protokol yang paling sederhana diantara tiga protokol tersebut diatas. Protokol ini terdiri dari pesan

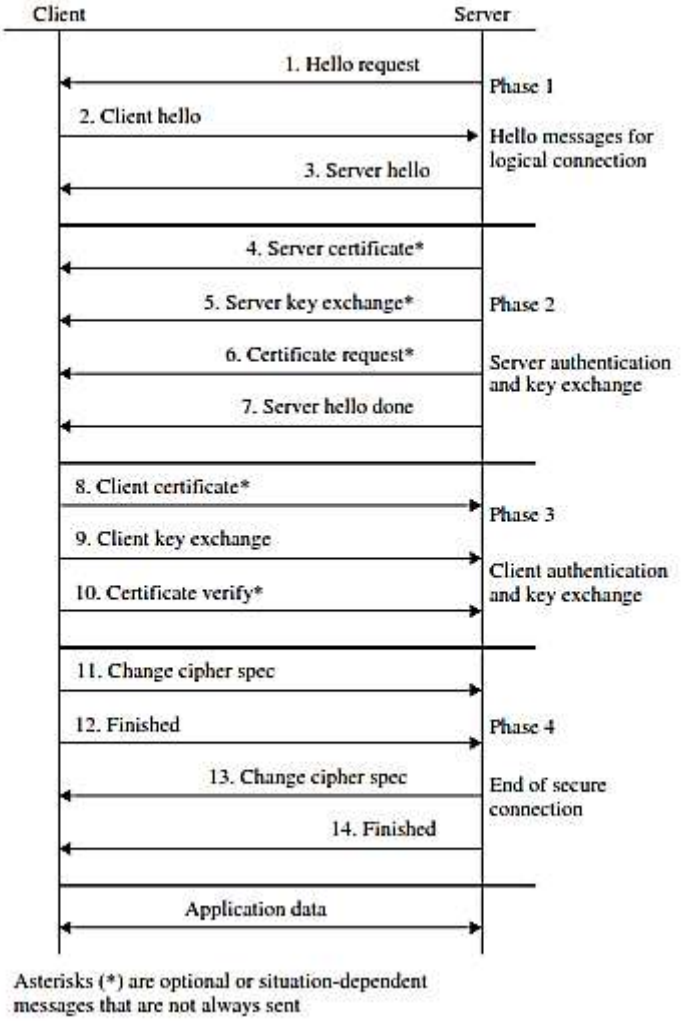
tunggal yang dikompresi dan dienkripsi sesuai dengan *cipher spec* dan kunci yang telah dinegosiasikan. Untuk mengubah *cipher spec*¸ baik *client* dan *server* mengirim pesan berupa *byte* bernilai 1.

2. SSL *Alert Protocol*

   Protokol ini terdiri dari pesan peringatan 2 *bytes*. *Byte* pertama adalah nilai peringatan untuk mengindikasikan tingkat keseriusan pesan. Apabila tingkatannya sudah fatal, SSL akan memutuskan koneksi. *Byte* kedua berisi kode untuk mengindikasikan peringatan yang lebih spesifik. Peringatan- peringatan tersebut berupa *unexpected-message*, *bad-record- mac, decompression-failure,* atau *no-certificate.*

3. SSL *Handshake Protocol*

Merupakan bagian yang paling penting dalam SSL. Protokol ini menyediakan tiga layanan untuk koneksi SSL antara *client* dan *server*, yaitu untuk menyetujui versi protokol, mengotentikasi satu sama lain, dan menegosiasikan algoritma enkripsi serta kunci kriptografinya untuk melindungi data yang dikirimkan.



Gambar 2. 6. SSL *Handshake Protocol* (Rhee, 2003)

Adapun fase-fase di dalam SSL *Handshake Protocol* adalah sebagai berikut:

a. Fase 1: Pembangunan Koneksi

Fase ini merupakan fase *initial* yang berfungsi membangun koneksi antara *client* dan *server*. Proses ini diinisialisasi oleh *client* dengan mengirimkan beberapa parameter, diantaranya:

1) *Version*

   Versi SSL yang didukung oleh *client*.

2) *Random*

   Nilai random yang dibangkitkan oleh *client*. Nilai ini akan digunakan pada proses *key exchange* untuk mencegah *replay attack*.

3) *Session ID*

   Variabel *session* dengan nilai tertentu. Nilai bukan 0 menunjukkan bahwa *client* akan meng-*update* parameter suatu koneksi yang telah dibangun atau membuat koneksi baru pada suatu *session* yang dibangun. Nilai 0 menunjukkan bahwa *client* akan membangun koneksi baru pada *session* baru.

4) *Ciphersuites*

   Daftar kombinasi algoritma kriptografi yang didukung oleh *client*. Setiap *ciphersuite* berisi algoritma *key exchange* dan *CipherSpec*.

5) *Compression Method*

Metode kompresi yang didukung oleh *client*.

b. Fase 2: Otentikasi *Server* dan Pertukaran Kunci

Pihak *server* memulai fase ini dengan mengirimkan pesan yang berisi sertifikat digital. Sertifikat digital ini dibutuhkan dalam setiap metode pertukaran kunci, kecuali *anonymous* Diffie-Hellman. Kemudian, pesan *server key exchange* dikirimkan untuk metode pertukaran kunci sebagai berikut:

1) *Anonymous Diffie-Hellman*
2) *Ephemeral Diffie-Hellman*
3) *RSA Key Exchange*

Tahap terakhir dari fase ini adalah pengiriman pesan *server done* yang menunjukkan bahwa pihak *server* telah selesai mengirimkan seluruh pesan pada fase ini.

c. Fase 3: Otentikasi *Client* dan Pertukaran Kunci

Setelah menerima pesan *server done*, pihak *client* akan memverifikasi apakah sertifikat yang dikirimkan oleh *server* sah atau tidak dan apakah parameter-parameter yang telah dikirimkan tersebut juga dapat diterima. Apabila *server* meminta sertifikat pada fase sebelumnya, maka *client* akan mengirimkan sertifikatnya.

Selanjutnya, pihak *client* akan mengirimkan pesan *client key exchange* dengan metode:

1) *Ephemeral* atau *Anonymous Diffie-Hellman*
2) *RSA Key Exchange*
3) *Fixed Diffie-Hellman*

Tahap terakhir dari fase ini adalah pengiriman pesan *certificate verify* yang menandakan verifikasi sertifikat *client*.

d. Fase 4: Akhir Koneksi

Pada fase ini, pesan *change cipher spec* dikirim oleh *client*. Pihak *client* kemudian mengirimkan pesan yang menandakan bahwa proses pembangunan koneksi telah selesai. Pihak *server* selanjutnya juga mengirimkan pesan *change cipher spec* miliknya. Setelah proses ini, proses *handshaking* telah selesai dan antara *server* dan *client* dapat saling bertukar informasi secara aman.

2.2.3 OpenSSL

OpenSSL[4] merupakan implementasi dari protokol *Secure Socket Layer* (SSL) yang dikembangkan secara *open source*. Pada dasarnya, OpenSSL terdiri dari dua jenis *library* (pustaka) pemrograman, yaitu SSL *library* dan *cryptography library* yang

[4] https://www.openssl.org/

dikemas dalam satu *tool*. SSL *library* merupakan *library* yang berisi implementasi beberapa versi protokol SSL yang telah dipublikasi. Sedangkan *cryptography library* merupakan *library* yang berisi berbagai implementasi algoritma kriptografi populer meliputi algoritma kunci simetrik, *public key*, *message digest*, dsb. *Library* ini juga menyediakan *Pseudo Random Number Generator* (PRNG) serta dukungan untuk manipulasi format sertifikat digital.

Library yang dimiliki oleh OpenSSL dapat sangat membantu dalam pengembangan aplikasi kriptografi terutama dalam menyediakan fungsi- fungsi kriptografi. Komponen-komponen yang ada pada OpenSSL adalah sebagai berikut:

1. **`libssl.a`**

   Merupakan implementasi protokol SSLv2, SSLv3, TLSv1, TLSv1.2 dan beberapa kode program pendukung lainnya.

2. **`libcrypto.a`**

   Merupakan implementasi algoritma kriptografi standar dan X.509 yang diperlukan dalam protokol SSL/TLS, diantaranya:

   a. *Ciphers,*
   b. *Digests,*
   c. *Public Key,*
   d. X.509 *Certificates,*
   e. *Systems,*

f. *Data Structures*.

3. `openssl`

Merupakan aplikasi *console* berbasis *command line interface* (CLI) yang dapat digunakan untuk:

a. Pembuatan parameter kunci RSA, *Diffie-Hellman* dan DSA.
b. Pembuatan sertifikat digital dengan format X.509, *Certificate Signing Request* (CSR), dan *Certificate Revocation List* (CRL).
c. Penghitungan *message digest*.
d. Enkripsi dan dekripsi data menggunakan algoritma standar yang terdapat pada OpenSSL.
e. Pengujian protokol SSL/TLS antara *client* dan *server*.
f. Penanganan *e-mail* yang dienkripsi dan ditandatangani menggunakan S/MIME.

OpenSSL bersifat *open source*, sehingga *source code* dari *library* ini tersedia bebas dan dapat dimodifikasi atau dikembangkan sesuai kebutuhan untuk keperluan penelitian ini.

2.2.4 OpenVPN

OpenVPN merupakan salah satu perangkat lunak VPN yang berbasis protokol SSL atau biasa disebut dengan SSL VPN. Protokol ini cukup banyak digunakan untuk mengamankan transaksi data melalui internet. Kuat dan cukup mudah untuk

dipelajari oleh pengguna baru, selain itu juga lebih sederhana untuk diimplementasikan dan dimanajemen oleh seorang *administrator*. Beberapa kelebihan penggunaan OpenVPN dibandingkan dengan perangkat lunak VPN yang lain, yaitu: (OpenVPN, 2002-2016)

1. Mendukung penggunaan *tunnel* VPN di berbagai macam sistem operasi seperti Linux, Debian, Microsoft Windows, BSD, Mac OS X dan SUN Solaris.
2. Mendukung *port Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP).
3. Lebih mudah dalam instalasi dan konfigurasi. Pada tahap instalasi tidak membutuhkan kompilasi kernel atau *patch*.
4. OpenVPN dibangun dengan tujuan portabilitas dan berjalan pada level *user-space*, bukan pada level kernel.
5. *Library* OpenSSL digunakan untuk enkripsi, otentikasi, dan sertifikasi OpenVPN *tunnel*. Sehingga *cipher,* ukuran kunci dan nilai HMAC yang didukung oleh *library* ini dapat digunakan.
6. Dapat membangun *secure tunnel* menggunakan kunci statik, kunci *pre-shared* atau menggunakan pertukaran kunci dinamis berbasis TLS.
7. Tidak bermasalah dengan perangkat NAT dalam membangun VPN *client.*
8. Mendukung alamat IP dinamis.
9. Pendokumentasiannya baik.

10. Kompatibel dengan SSL/TLS, sertifikat RSA, X.509 PKI, dan driver jaringan virtual TUN/TAP untuk mendukung fungsionalitas IP *tunneling*.

OpenVPN yang di*compile* dengan *library* OpenSSL versi 1.0.2h mampu menyediakan TLS *ciphersuites* kurang lebih sebanyak 31 *ciphersuites* sesuai rincian di dalam tabel berikut:

Tabel 2. 2. Rincian TLS *Ciphersuites* OpenVPN Versi Asli (OpenVPN, 2002-2016)

TLS *Ciphersuites*	Otentikasi	Pertukaran Kunci	Enkripsi	*Message Digest*
TLS_RSA_WITH_AES256_SHA	RSA	RSA	AES256	SHA
TLS_DHE_DSS_WITH_AES256_SHA	DSS	DHE	AES256	SHA
TLS_DHE_RSA_WITH_AES256_SHA	RSA	DHE	AES256	SHA
TLS_RSA_WITH_CAMELLIA256_SHA	RSA	RSA	CAMELLIA256	SHA
TLS_DHE_DSS_WITH_CAMELLIA256_S	DSS	DHE	CAMELLIA256	SHA
TLS_DHE_RSA_WITH_CAMELLIA256_S	RSA	DHE	CAMELLIA256	SHA
TLS_RSA_WITH_3DES_SHA	RSA	RSA	3DES	SHA
TLS_RSA_WITH_3DES_MD5	RSA	RSA	3DES	MD5
TLS_DHE_DSS_WITH_3DES_SHA	DSS	DHE	3DES	SHA
TLS_DHE_RSA_WITH_3DES_SHA	RSA	DHE	3DES	SHA
TLS_RSA_WITH_AES128_SHA	RSA	RSA	AES128	SHA
TLS_DHE_DSS_WITH_AES128_SHA	DSS	DHE	AES128	SHA
TLS_DHE_RSA_WITH_AES128_SHA	RSA	DHE	AES128	SHA
TLS_RSA_WITH_SEED_SHA	RSA	RSA	SEED	SHA
TLS_DHE_DSS_WITH_SEED_SHA	DSS	DHE	SEED	SHA
TLS_DHE_RSA_WITH_SEED_SHA	RSA	DHE	SEED	SHA
TLS_RSA_WITH_CAMELLIA128_SHA	RSA	RSA	CAMELLIA128	SHA
TLS_DHE_DSS_WITH_CAMELLIA128_S	DSS	DHE	CAMELLIA128	SHA
TLS_DHE_RSA_WITH_CAMELLIA128_S	RSA	DHE	CAMELLIA128	SHA
TLS_RSA_WITH_IDEA_SHA	RSA	RSA	IDEA	SHA
TLS_RSA_WITH_IDEA_MD5	RSA	RSA	IDEA	MD5
TLS_RSA_WITH_DES_SHA	RSA	RSA	DES	SHA
TLS_RSA_WITH_DES_MD5	RSA	RSA	DES	MD5
TLS_DHE_DSS_WITH_DES_SHA	DSS	DHE	DES	SHA
TLS_DHE_RSA_WITH_DES_SHA	RSA	DHE	DES	SHA

TLS_RSA_WITH_RC2_SHA	RSA	RSA	RC2	SHA
TLS_DHE_DSS_WITH_RC2_SHA	DSS	DHE	RC2	SHA
TLS_DHE_RSA_WITH_RC2_SHA	RSA	DHE	RC2	SHA
TLS_RSA_WITH_RC4_SHA	RSA	RSA	RC4	SHA
TLS_DHE_DSS_WITH_RC4_SHA	DSS	DHE	RC4	SHA
TLS_DHE_RSA_WITH_RC4_SHA	RSA	DHE	RC4	SHA

Pada penelitian ini akan dilakukan modifikasi pada OpenVPN dengan menambahkan algoritma *stream cipher* Rabbit sebagai salah satu alternatif pilihan TLS *ciphersuites*nya. Selain itu *ciphersuites* RC2 dan *ciphersuites* RC4 juga akan dihilangkan karena algoritma enkripsi yang digunakan sudah *obsolete*. Modifikasi OpenVPN akan menghasilkan TLS *ciphersuites* yang kurang lebih sama seperti dirincikan dalam Tabel 2.1, namun dengan tambahan *ciphersuites* Rabbit, tidak terdapat *ciphersuites* RC2.

2.3 Hardware Related

2.3.1 Single Board Computer (SBC) Raspberry PI

Raspberry Pi merupakan sebuah *Single Board Computer* (SBC) dengan ukuran sebesar kartu kredit. Dikembangkan oleh *Raspberry Pi Foundation* di *United Kingdom* (UK) dengan tujuan untuk mempromosikan pembelajaran komputer di sekolah. Terdapat tiga perusahaan yang memproduksi dan mendistribusikan *Raspberry Pi* yaitu *Newark Element* 14 (*Premier Farnell*), RS *Components* dan *Egoman*. *Raspberry Pi* sudah berkembang menjadi empat model, yaitu model A, model A+, model B, dan model B+. Secara umum, spesifikasi *hardware* utama yang digunakan pada keempat model

tersebut sama. SBC yang akan digunakan pada penelitian ini adalah *Raspberry Pi 3 Model B+* dengan spesifikasi sebagai berikut:

1. *System on Chip* (SoC): *Broadcom* BCM2835
    a. *Central Processing Unit* (CPU): ARM1176JZF-S *core* 700MHz.
    b. *Graphical Processing Unit* (GPU): *Broadcom VideoCore IV@* 250 MHz.
    c. Memori (SDRAM): 512 MB (GPU *shared*).
2. *General Purpose Input Output* (GPIO): 40 GPIO *pins*.
3. *Universal Serial Bus:* 4 USB *ports.*
4. *Onboard Storage: MicroSD card socket.*
5. *Onboard Network:* 10/100 MB/s *Ethernet.*
6. Dimensi: 85,6 mm x 56 mm.
7. *Power rating:* 700 mA (3,5 W).
8. *Power source*: *5 V.*

Raspberry Pi model A menggunakan spesifikasi *hardware* yang relatif sama dengan *Raspberry Pi* model B. Perbedaannya adalah pada model A masih menggunakan SDRAM: 256 MB dan tidak memiliki *port Ethernet* atau *interface* untuk *onboard networking*. Berikut ini adalah tampilan SBC *Raspberry Pi 3 Model B+* yang digunakan:

Gambar 2. 7. *Raspberry Pi 3 Model B+*

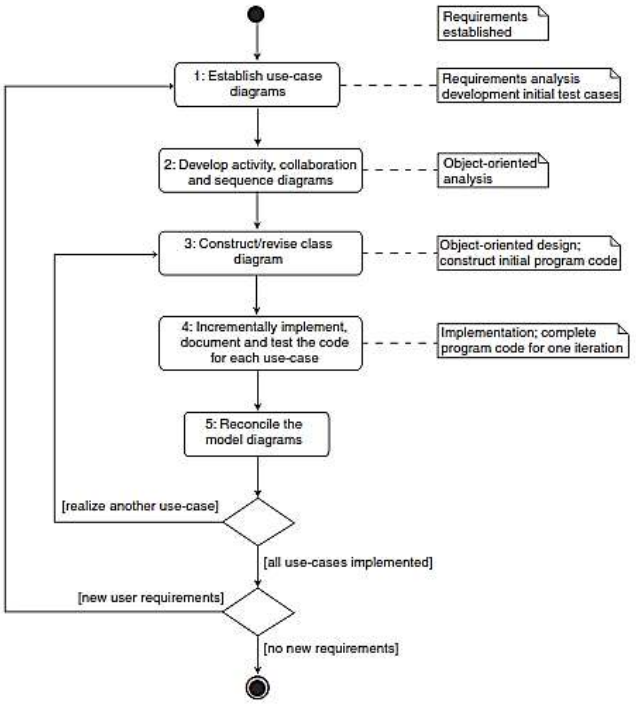
2.4 Unified Modeling Language (UML)

UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan, membuat analisis, membuat desain, dan menggambarkan rancangan suatu sistem yang akan dibangun. UML versi 2.0 memiliki 14 macam diagram yang dibagi menjadi dua grup utama, yaitu *structure diagrams* yang merepresentasikan data dan hubungan statis dalam sistem dan *behavior diagrams* yang menggambarkan hubungan dinamis antar objek yang terlibat dalam sistem. Berikut ini adalah rincian dari kedua grup utama diagram UML tersebut:

Tabel 2. 3. Rincian Diagram UML

Structure Diagrams	*Behavior Diagrams*
a. *Class Diagram*	a. *Activity Diagram*
b. *Object Diagram*	b. *Sequnce Diagram*
c. *Package Diagram*	c. *Communication Diagram*
d. *Deployment Diagram*	d. *Interaction Diagram*
e. *Component Diagram*	e. *Timing Diagram*
f. *Composite Structure Diagram*	f. *Behavior State Machine*
	g. *Protocol State Machine*
	h. *Use Case Diagram*

UML bukan merupakan suatu metode pengembangan sistem, melainkan suatu diagram atau notasi yang digunakan untuk pemodelan rancangan sistem. UML tidak memiliki proses tertentu mengenai proses yang harus diikuti layaknya sebuah metode. Pembuat UML telah mengusulkan sebuah metode pengembangan *Rational Unified Process* (RUP) yang menggunakan keseluruhan diagram UML untuk pemodelan rancangan sistem. Namun, pada beberapa kasus metode ini dianggap terlalu kompleks. Oleh karena itu, dalam (Barclay & Savage, 2004) dijelaskan sebuah metode pengembangan sistem yang lebih sederhana dengan menggunakan sebagian diagram UML untuk pemodelan rancangan sistem. Metode ini disebut dengan *lightweight process*.



Gambar 2. 8. *Lightweight Process* (Barclay & Savage, 2004)

Sesuai dengan Gambar 2.11 di atas, tahap pertama dari *lightweight process* adalah melakukan analisis kebutuhan (*requirement)* sistem yang akan dibangun. Kemampuan atau fitur-fitur sistem yang diinginkan oleh *user* harus didefiniskan terlebih dahulu. Setelah mengetahui kebutuhan dalam sistem yang akan dibangun, kemudian dikembangkan menjadi *use case* yang ditampilkan dengan menggunakan *use case diagram*. Tahap kedua adalah mendetailkan *use case* yang telah dibuat dengan menyusun satu atau lebih diagram analisis. Diagram ini digunakan untuk memastikan setiap objek dan interaksinya telah memenuhi kebutuhan yang dijelaskan pada *use case*. Diagram analisis di dalam UML terdiri dari *activity diagram, sequence diagram,* dan *collaboration diagram*. Setelah penyusunan diagram analisis selesai, tahap ketiga adalah pembuatan *class diagram*. Tahap keempat adalah proses implementasi, pembuatan kode program, dan pengujian terhadap sistem yang dibangun. Tahap kelima adalah menyesuaikan diagram pemodelan apabila terdapat penambahan *use case* atau penambahan *user requirement*.

2.5 Tinjauan Pustaka

Dalam penelitian ini, penulis melakukan studi literatur dari beberapa jurnal dan paper international. Berikut adalah hasil *review* dari tiga literatur:

2.5.1 Literatur I

SOHO Remote Access VPN. Easy As Pie, Raspberry Pi

1. **Masalah**

Bagaiamana cara yang aman dalam melakukan control atau remote jaringan yang ada dirumah menggunakan jaringan public ketika sedang berada ditempat umum.

2. **Metode :**

   Membuat remote access VPN yang akan diberi nama SOHO (*Small Office dan Home Office*) menggunakan Raspberry Pi

3. **Hasil :**

   Hal-hal privasi adalah hal yang paling diutamakan dalam *security* Professionals. Ini pun memaksa kita untuk lebih berhati-hati dalam penggunaan internet secara gratis.pembuatan Remote access VPN yang akan diberi nama yaitu, SOHO. Secara umum SOHO diterapkan dengan baik dan bagus. Pembuatan SOHO menggunakan Raspberry Pi yang sudah dikonfigurasikan dengan baik dengan penambahan *software OpenSource*, yaitu OpenVPN dan OpenSSL.

2.5.2. Literatur II

Using The Raspberry Pi To Establish A Virtual Private Network (VPN) Connection To A Home Network

1. **Masalah :**

   Membutuhkan jalur koneksi yang aman untuk mengakses jaringan internal sementara mengakses sistem tersebut menggunakan jaringan publik

2. **Metode :**

Membuat jaringan VPN menggunakan Raspberry Pi dan OpenVPN sebagai gateway jaringan yang aman dalam melakukan control atau remote terhadap sistem atau jaringan internal (jaringan yang ada pada rumah)

3. **Hasil :**

   Menjadi modul prototipe yang dapat terkoneksi, koneksi dilakukan pada restoran burger king kemudian menggunakan jaringan internet public dan terkoneksi oleh VPN yang telah dibuat sebagai gateway yang aman untuk melakukan control dan remote jaringan internal (jaringan yang ada pada rumah).

2.5.3 Literatur III

Limitations And Differencesof Using IPSec, TLS/SSL or SSH As VPN-Solution

1. **Masalah :**

   Dari tiga metode tersebut, manakah yang paling menguntungkan dalam membangun sebuah VPN, menguntungkan dalam hal ini yaitu, kekuatan dari keamanannya dan harga pada saat implementasinya

2. **Metode :**

   Membandingkan IPsec,TLS/SSL dan SSH dalam membuat VPN

3. **Hasil :**

   Pada paper ini, hasil yang didapatkan yaitu ketiga metode tersebut sangat baik dan memiliki keterbatasan dan

kelebihan pada masing-masing kebutuhan. Dalam hal ini, memilih VPN yang tepat sebagai solusi untuk pengaturan yang berbeda, maupun perbedaan lingkungan sangatlah sulit. Tidak semua VPN ini dapat masuk kedalam jaringan atau konfigurasi host tertentu.

Dari ketiga literatur tersebut, penulis mendapatkan suatu gagasan dan ide untuk melakukan penelitian terkait pembuatan suatu prototipe perangkat VPN gateway untuk digunakan tramsmisi data yang aman menggunakan Rashpbbery Pi dengan mamanfaatkan OpenVPN dan OpenSSL yang dimodifikasi dengan tambahan algoritma stream cipher Rabbit sebagai alternatif *ciphersuite* enkripsi yang dapat dikatakan lebih cepat dari algoritma AES, 3DES, CAMELIA-256 dan SEED.