Keamanan SSL dalam Serangan Internet

Deasy Ramadiyan Sari

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

Email: if13008@students.if.itb.ac.id

Abstrak

SSL merupakan protokol yang digunakan untuk browsing web secara aman. Banyak fitur yang disediakan pada SSL merupakan bentuk dari keamanan dalam *browsing* Internet. Paper ini menjelaskan mengenai keamanan yang disediakan oleh SSL untuk menciptakan web yang aman pada saat melakukan *browsing.* Penjelasan yang dilakukan meliputi fitur-fitur keamanan yang disertakan dalam setiap komponen pesan yang ada pada saat negosiasi pelayanan keamanan yang dibentuk antara *client* dan *server*. Selain itu dijelaskan pula mengenai isu–isu serangan yang memungkinkan ada pada Internet yang mampu diatasi oleh SSL, peluang penyerang melakukan serangan terhadap suatu web melalui SSL, serta sub protokol yang ada pada SSL untuk menjaga keamanan web pada saat *browsing*.

Kata kunci :*SSL, keamanan, serangan Internet*

1. Pendahuluan

1.1 Sejarah

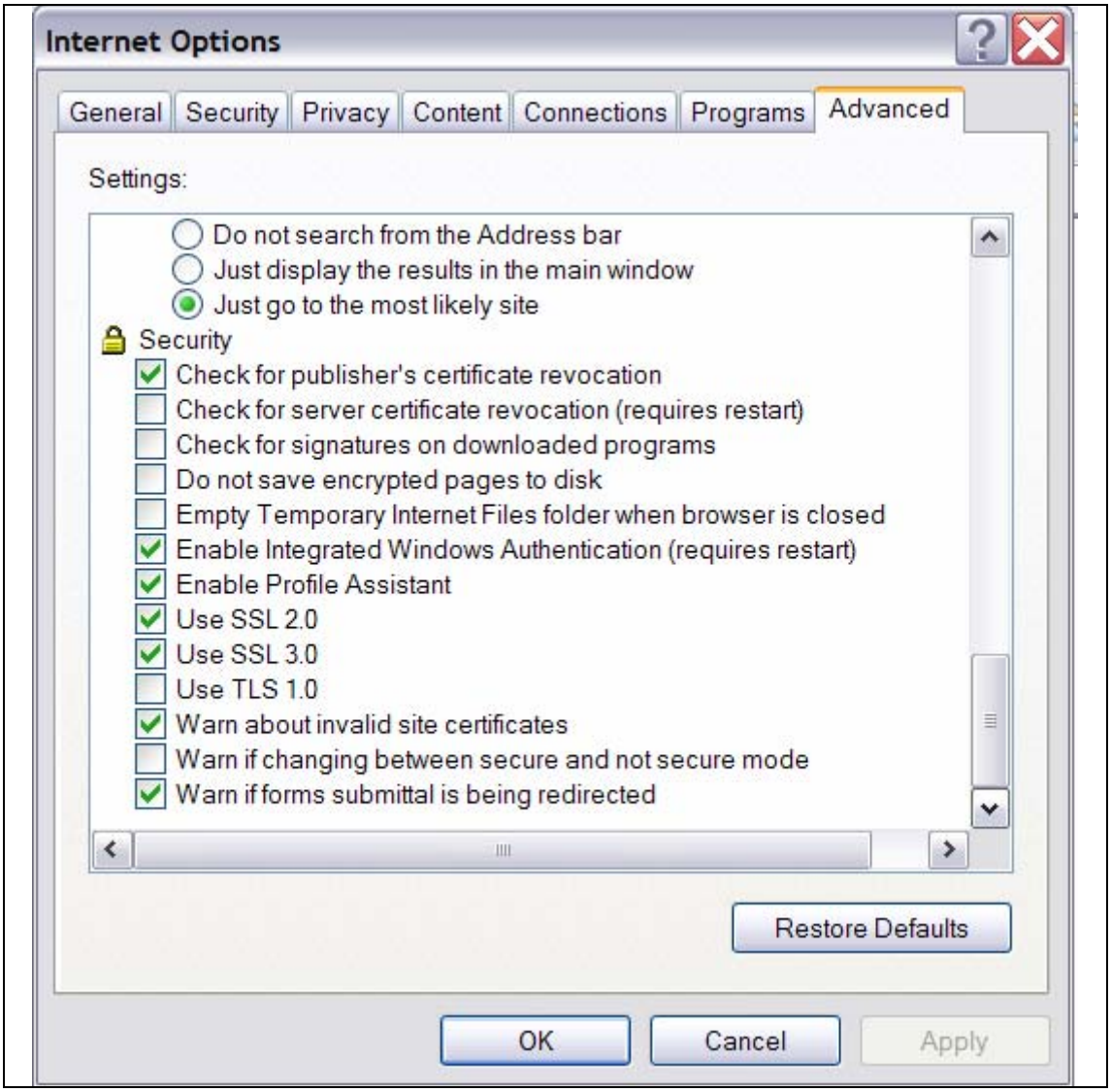
Secure Socket Layer (SSL) adalah protokol yang digunakan untuk browsing web secara aman. SSL bertindak sebagai protokol yang mengamankan komunikasi antara *client* dan *server*. Protokol ini memfasilitasi penggunaan enkripsi untuk data yang rahasia dan membantu menjamin integritas informasi yang dipertukarkan antara website dan web browser.

SSL dikembangkan oleh Netscape Communations pada tahun 1994. Ada beberapa versi SSL, versi 2 dan versi 3, tetapi versi 3 paling banyak digunakan saat ini.

Untuk memastikan apakah Internet Explorer sudah siap menjalankan protokol SSL, klik dari IE :

Tools → Internet Option → Advanced

Lalu cari pilihan Security, kemudian periksa apakah SSL versi 2.0 atau SSL versi 3.0 yang telah diberikan tanda √ (lihat Gambar 1).



Gambar 1 Penandaan SSL Aktif

2. Operasi SSL

2.1 Peran SSL

SSL mempunyai dua buah peran yang berbeda untuk di gunakan dalam komunikasi. Satu sistem selalu menjadi *client*, sementara system yang lain akan terus menjadi *server*. Perbedaan dari dua peran ini sangat penting, karena kelakuan dari setiap peran tersebut juga sangat berbeda. Client merupakan sistem yang menginisiasikan komunikasi yang aman, sementara *server* hanya merespon request dari *client* tersebut.

Untuk SSL sendiri, perbedaan yang paling penting dari *client* dan *server* adalah aksi yang mereka lakukan ketika negosiasi mengenai parameter keamanan. Ketika inisiasi komunikasi dilakukan oleh *client*, *client* mempunyai tanggung jawab untuk mengajukan sekumpulan pilihan SSL yang akan digunakan dalam pertukaran. *Server* hanya memilih dari pilihan yang disediakan oleh *client*, lalu memilih apa yang akan digunakan dalam kedua sistem ini.

2.2 Pesan SSL

Ketika SSL *client* dan *server* berkomunikasi, mereka melakukan komunikasi tersebut dengan bertukaran pesan SSL. Tabel di bawah ini menunjukkan pesan SSL pada setiap level pada protocol:

Tabel 2-1 Pesan SSL

Pesan	Deskripsi
Alert	Menginformasikan pihak lain akan kegagalan komunikasi atau pelanggaran keamanan
Application Data	Informasi yang dipertukarkan oleh kedua belah pihak, yang terenkripsi, terotentikasi, dan terverifikasi oleh SSL
Certificate	Pesan yang membawa sertifikat kunci publik dari pengirim
CertificateRequest	Permintaan oleh *server* agar *client* menyediakan sertifikat kunci publiknya
CertificateVerify	Pesan dari *client* yang memverifikasi bahwa ia mengetahui kunci privat bersesuaian dengan sertifika kunci publik
ChangeCipherSpec	Indikasi untuk mulai menggunakan pelayanan keamanan
ClientHello	Pesan dari *client* yang menandakan keperluan pelayanan keamanan dan kemampuan untuk mendukungnya
ClientKeyExchange	Pesan dari *client* yang membawa kunci kriptografi untuk berkomunikasi
Finished	Indikasi bahwa seluruh negosiasi sudah terlaksana dan komunikasi yang aman sudah terbentuk
HelloRequest	Permintaan oleh *server* bahwa, *client* memulai proces neogosiasi SSL
ServerHello	Pesan dari *server* yang menandakan pelayanan keamanan yang mana yang akan digunakan untuk komunikasi
ServerHelloDone	Indikasi dari *server* bahwa *server* sudah menyelesaikan seluruh permintaan *client* untuk membentuk komunikasi
ServerKeyExchange	Pesan dari *server* yang membawa kunci kriptografi untuk berkomunikasi

2.3 Menciptakan Komunikasi yang Terenkripsi

2.3.1 ClientHello

Pesan *ClientHello* memulai komunikasi SSL antara dua buah pihak. *Client* menggunakan pesan ini untuk menanyakan kepada *server* untuk memulai negosiasi pelayanan keamanan dengan menggunakan SSL. Tabel dibawah ini menjelaskan komponen-komponen pada *ClientHello* :

Tabel 2-2 Komponen Pesan *ClientHello*

Komponen	Kegunaan
Version	Mengidentifikasi versi terbaru dari protokol SSL yang dapat didukung oleh *client*
RandomNumber	32 byte nomer acak yang digunakan untuk perhitungan kriptografi
SessionID	Mengidentifikasi *session* spesifik SSL
CipherSuites	List dari parameter kriptografi yang dapat didukung oleh *client*
CompressionMethods	Mengidentifikasi metoda data kompresi yang didukung oleh *client*

2.3.2 ServerHello

Ketika *server* menerima pesan *ClientHello*, maka *server* akan membalas dengan *ServerHello*. Jika pada ClienHello, *Client* memberikan saran, maka pada *ServerHello* keputusan akhir dibuat oleh *server*. Berikut adalah komponen dari pesan *ServerHello* :

Tabel 2-3 Komponen Pesan *ServerHello*

Komponen	Kegunaan
Version	Mengidentifikasi versi terbaru dari protokol SSL yang akan digunakan pada komunikasi
RandomNumber	32 byte nomer acak yang digunakan untuk perhitungan kriptografi
SessionID	Mengidentifikasi *session* spesifik SSL
CipherSuites	List dari parameter

	kriptografi yang akan digunakan pada komunikasi
CompressionMethods	Mengidentifikasi metoda data kompresi akan digunakan pada komunikasi

2.3.3 ServerKeyExchange

Pesan ini mengandung *ChiperSuite* field pada *ServerHello*. Ketika *ChiperSuite* field mengindikasikan algoritma kriptografi dan ukuran kunci, pesan ini mengandung informasi kunci publiknya. *ServerKeyExchange* dikirim tanpa ada enkripsi, sehingga hanya informasi kunci publik yang dapat secara aman dikirimkan didalamnya.

2.3.4 ServerHelloDone

Pesan ini menyampaikan kepada *Client* bahwa *server* sudah menyelesaikan pesan inisial negosiasi. Pesan ini sendiri tidak mengandung informasi yang lain, tapi merupakan pesan yang penting bagi *Client* karena, jika *Client* telah menerima pesan ini maka *Client* dapat meneruskan ke fase berikutnya dalam membentuk komunikasi yang aman

2.3.5 ClientKeyExchange

Ketika *server* sudah menyelesaikan bagin inisiasi dari negosiasi SSL, *Client* akan merespon dengan pesan *ClientKeyExchange*. Seperti fungsi *ServerKeyExchange* pada *server* yang membawa informasi *key* untuk *server*, pesan *ClientKeyExchange* memberitahukan *server* kunci informasi yang dimiliki oleh *Client*. Dalam kasus ini, kunci informasi adalah untuk algoritma enkripsi simetris yang akan digunakan oleh kedua belash pihak dalam *session*.

Enkripsi melindungi kunci inforamasi ketika berada dalam jaringan, *server* tidak akan mungkin bisa untuk mendekripsikan pesan tersebut. Operasi ini merupakan perlindungan penting jika ada penyerang yang menahan pesan dari *server* yang sah dan berpura=pura menjadi *server* tersebut dengan meneruskan pesan ke *Client* yang tidak menyadari apapun.

2.3.6 ChangeCipherSpec

Setelah *Client* mengirimkan kunci inforamsi pada pesan *ClientKeyExchange*, persiapan negosiasi SSL sudah terpenuhi. Pada titik ini, kedua belah pihak sudah siap untuk mengguanakn pelayanan keamanan yang sudah dinegosiasikan tersebut. Protokol SSL mendefinisikan sebuah pesan *ChangeCipherSpec* yang mengindikasikan bahwa pelayanan keamanan sudah dapat dilaksanakan. Pesan *ChangeCipherSpec* disediakan sebagai penanda pada sebuah sistem untuk mulai menggunakan keamanan informasinya.

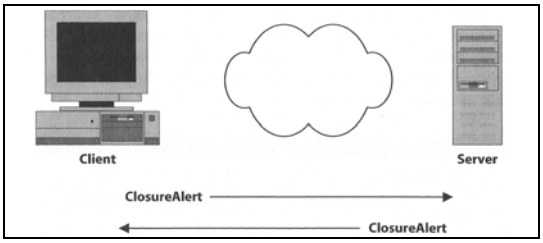
2.3.7 Finished

Segera setelah mengakhiri pesan *ChangeCipherSpec*, setiap sistem juga mengirimkan pesan *Finished*. Pesan *Finished* membolehkan kedua belah pihak untuk memverifiaksi bahwa negosiasi telah berhasil dan keamanan belum dikompromikan. Ada dua aspek keamanan yang dikontribusikan dari pesan *Finished* ini, yaitu:

1. Pesan *Finished* merupakan sebuah subjek yang diperlukan pada saat negosiasi cipher.
   Yang artinya enkripsi dan otentikasi bergantung padanya. Jika pihak penerima tidak berhasil mendekrip dan memverifikasi pesan, sudah tentu ada kejanggalan pada negosiasi keamanan.
2. Content dari pesan *Finished* juga mengandung kemampuan untuk melindungi keamanan pada saat negosiasi SSL. Setiap pesan *Finished* mengandung kriptografi hash dari informasi penting mengenai negosiasi yang baru saja selesai dilakukan. Ini melindungi dari penyerang yang menyerang dengan memasukkan pesan fiktif atau memindahkan pesan yang sah dari komuniakasi. Jikaseorang penyrang melakukannya, maka perhitungan kriptografi hash pada *Client* dan *server* akan berbeda maka akan mudah terdeteksi.

2.4 Mengakhiri Komunikasi yang Aman

SSL mempunyai prosedur aman untuk mengakhiri komunikasi aman yang melibatkan dua buah pihak. Gambar 2 dibawah ini menunjukkan bagaimana caranya.



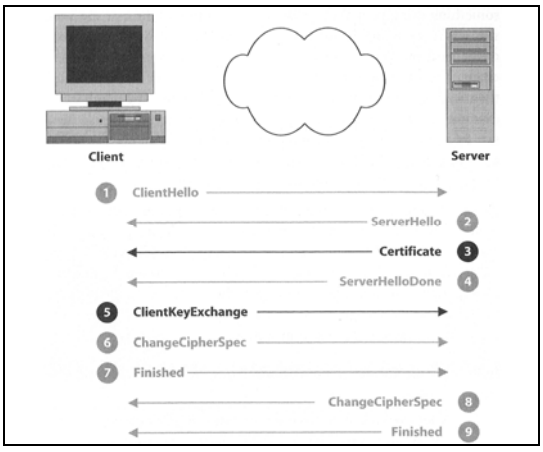
Gambar 2 Mengakhiri Komunikasi yang Aman

Kedua buah sistem bersama-sama mengirimkan *ClosureAlert* kepada satu sama lain. Penutupan secara eksplisit dari sebuah *session* melindungi dari *truncation attack*, yang merupakan serangan dimana penyerang mempunyai kemampuan untuk berkompromi dengan keamanan untuk secara prematur menyudahi komunikasi.

2.5 Otentitkasi Identitas Server

Alasan penggunaan enkripsi adalah untuk menjaga kerahasiaan informasi dari pihak ketiga. Namun, jika pihak ketiga berhasil menyamar sebagai pihak yang menerima informasi, maka enkripsi tidak lagi dapat diharapkan. Data mungkin saja terenkripsi, namun penyerang akan tetap saja mendapatkan seluruh data untuk mendekrispsinya.

Unutuk mencegah serangan ini, SSL menyediakan mekanisme yang membolehkan setiap pihak untuk saling mengotentikasi pihak lain. Dengan mekanisme ini, setiap pihak akan yakin bahwa pihak yang satu lagi adalah pihak yang sebenarnya dan bukan pihak penyamar. Untuk mengotentikasi identitas *server*, dapat dilihat seperti Gambar 3 dibawah ini.



Gambar 3 Otentikasi Identitas *Server*

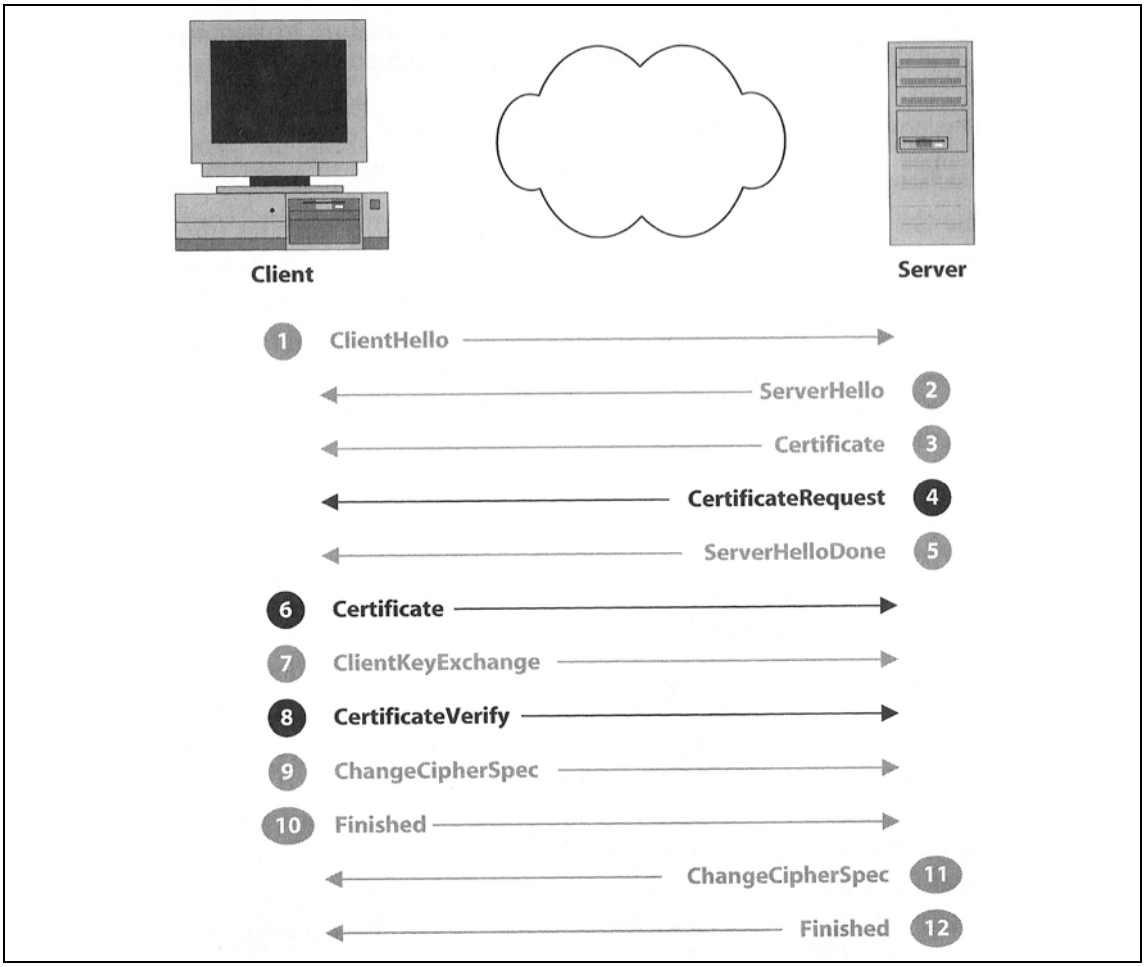
Tabel dibawah ini menunjukkan alur bagaimana sistem dapat mengotentikasi *server*

Tabel 2-4 Alur Otentikasi *Server* oleh Sistem

	Aksi
1	*Client* mengirim pesan ClienHello menawarkan pilihan SSL
2	*Server* merespon dengan mengirimkan pesan *ServerHello* memilih pilihan SSL
3	*Server* mengirimkan sertifikat kunci publiknya dalam pesan *Certificate*
4	*Server* mengakhiri bagian negosiasinya dengan mengirim pesan *ServerHelloDone*
5	*Client* mengirimkan kunci informasi Session dalam pesan ClinetKeyExchange
6	*Client* mengirimkan pesan *ChangeCipherSpec* untuk mengaktifkan pilihan negosiasi untuk pesan kedepannya yang akan dikirim
7	*Client* mengirim pesan *Finished* untuk membiarkan *server* memeriksa pilihan aktivas yang terbaru
8	*Server* mengirim pesan *ChangeCipherSpec* untuk mengaktivasi pilihan negosiasi pesan yang akan dikirim nantinya
9	*Server* mengirim pesan *Finished* untuk membiarkan *Client* memeriksa pilihan aktivas yang terbaru

2.6 Otentikasi Identitas Client

Karena SSL mempunyai mekanisme untuk mengotentikasi identitas *server*, sudah selayaknya kita mengharapkan mekanisme untuk mengotentikasi identitas *Client*. Mekanismenya hapir sama dengan otentikasi identitas *server*. Mekanismenya dapat dilihat pada Gambar 4 dibawah ini



Gambar 4 Otentikasi Identitas *Client*

2.7 Mendapatkan Session Sebelumnya

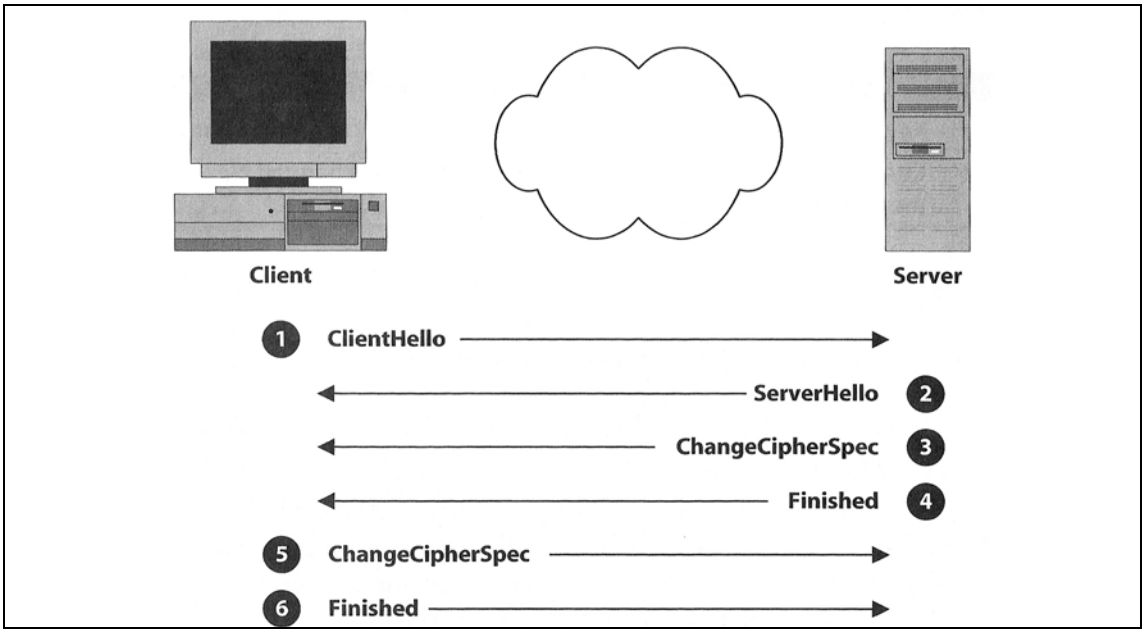
Membentuk *session* SSL mungkin kompleks, memerlukan perhitungan kriptografi yang baik dan beberapa pesan protokol. Untuk meminimalkan biaya dari kalkulasi dan pesan. SSL membuat suatu mekanisme dimana dua buah pihak dapat menggunakan kembali SSL sebelumnya yang sudah dinegosiasikan. Dengan metode ini, pihak tersebut tidak perlu lagi untuk mengulang negosiasi keriptografi atau kalkulasi otentikasi, kedua pihak hanya perlu melanjutkan apa yang mereka tinggalkan sebelumnya.

Table berikut menunjukkan langkah-langkah untuk mendapatkan kembali *session* yang sebelumnya.

Tabel 2-5 Langkah Mendapat Session Kembali

	Aksi
1	*Client* mengirim pesan *ClientHello* yang mengandung *SessionID* yang pernah terbentuk sebelumnya
2	*Server* merspon dengan *serverHello* menyetujui *SessionID* tersebut
3	*Server* mengirim pesan *ChangeCipherSpec* untuk membuat pilihan akif kembali keamanan *session* untuk pesan yang akan dikirim
4	*Server* mengirim pesan *Finsihed* agar *Client* dapat memeriksa pilihan yang baru teraktivasi kembali tersebut
5	*Client* mengirim pesan *ChangeCipherSpec* untuk mengaktivkan kembali pilihan negosiasi untuk pesan yang akan dikirim nantinya
6	*Client* mengirm pesan *Finished* untuk membolehkan *server* memeriksa pilihan yang baru diaktivkan kembali tersebut

Berikut adalah gambar dari langkah-langkah tersebut diatas



Gambar 5 Mendapatkan Session Sebelum

Kunci dari mendapatkan *session* sebelumnya adalah pesan *ClientHello*. *Client* mengajukan untuk menggunakan *session* sebalumnya dengan menyertakan *SessionID* di dalam *ClientHello.* Meskipun, mendapatkan kembali *session* yang pernah terbentuk merupakan kesepakatan yang sangat bagus,dari segi kenyamanan dan efisiensi dari sistem yang menggunakannya, sistem tersebut harus lebih berhati-hati dalam penanganannya. Ketika sebuah kunci dibentuk, enkripsi pasti akan menjadi lebih tidak aman, semakin banyak informasi yang dilindungi, semakin banyak pula waktu yang digunakan, sehingga, penyerang mempunyai waktu yang lebih lama juga untuk menganalisis kelemahan.

3. Format Pesan

3.1 Kebutuhan Transport

SSL merupakan protokol yang bergantung pada lower level protokol untuk memindahkan pesan antar *peers.* Protokol SSL membutuhkan lower layer yang reliable, yang dapat memastikan bahwa transmisi dari pesan SSL akan sukses tanpa ada error dan berjalan dengan urutan yang semestinya. Oleh karena itu, SSL bergantung pada *Transmission Control Protocol* (TCP) untuk memenuhi kebutuhannya.

3.1.1 TCP/IP

SSL beroperasi antara protokol komunikasi TCP/IP (*Transmission Control Protocol/Internet Protocol*) dan aplikasi (lihat Gambar 6).

Application (HTTP, FTP, Telnet)
Security (SSL)
Transport (TCP)
Network (IP)
Data link (PPP)
Phyiscal (modem, ADSL, cable TV)

Gambar 6 Protokol Komunikas SSL

SSL seolah-olah berlaku sebagai lapisan (layer) baru antara lapisasan transpor (TCP) dan lapisan aplikasi. TCP/IP adalah standard protokol yang digunakan untuk menghubungkan komputer dan jaringan dengan jaringan dari jaringan yang lebih besar, yaitu Internet.

3.1.2 Cara kerja TCP/IP (tanpa SSL)

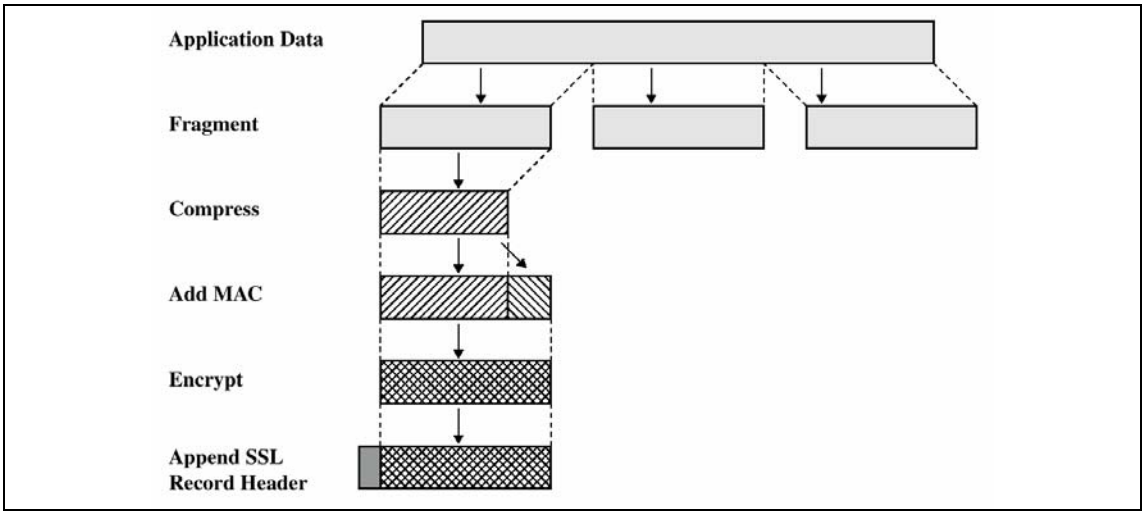
Kebanyakan transmisi pesan di Internet dikirim sebagai kumpulan potongan pesan yang disebut paket. IP bertanggung jawab untuk merutekan paket (lintasan yang dilalui oleh paket).

Pada sisi penerima, TCP memastikan bahwa suatu paket sudah sampai, menyusunnya sesuai nomor urut, dan menentukan apakah paket tiba tanpa mengalami perubahan. Jika paket mengalami perubahan atau ada data yang hilang, TCP meminta pengiriman ulang. TCP/IP tidak memiliki pengamanan komunikasi yang bagus. TCP/IP tidak dapat mengetahui jika pesan diubah oleh pihak ketiga (*man-in-the-middle attack*).

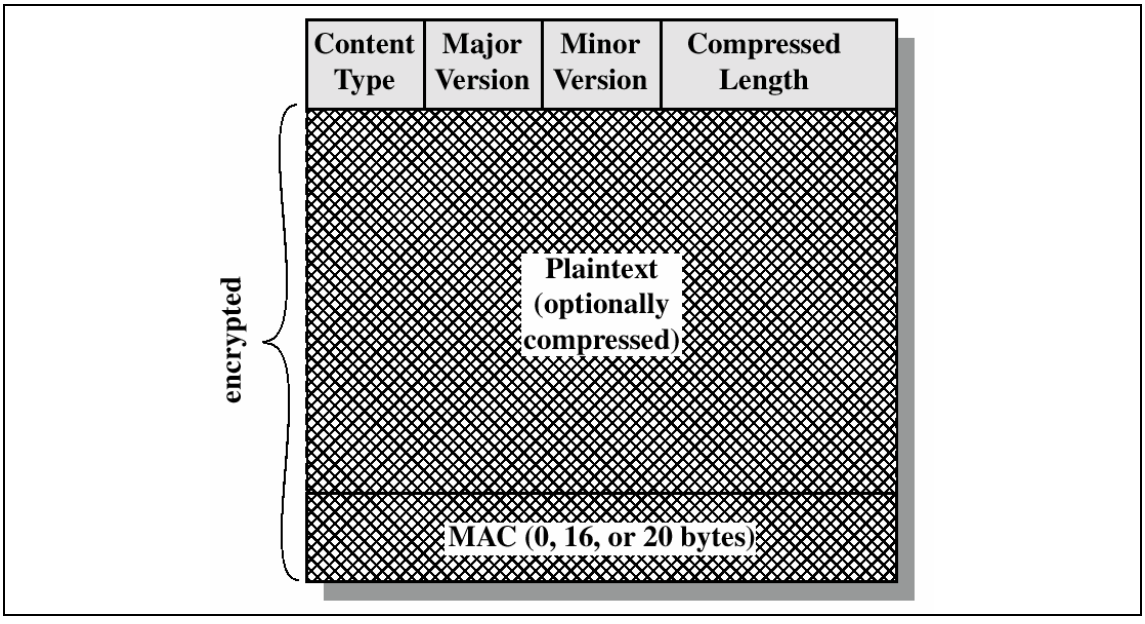
SSL membangun hubungan (*connection*) yang aman antara dua socket, sehingga pengiriman pesan antara dua entitas dapat dijamin keamanannya.

Perlu dicatat bahwa SSL adalah protokol *Client-server*, yang dalam hal ini web browser adalah *Client* dan website adalah *server*. *Client* yang memulai komunikasi, sedangkan *server* memberi respon terhadap permintaan *Client*.

3.2 Sub Protokol Record



Gambar 7 Sub Protokol SSL Record



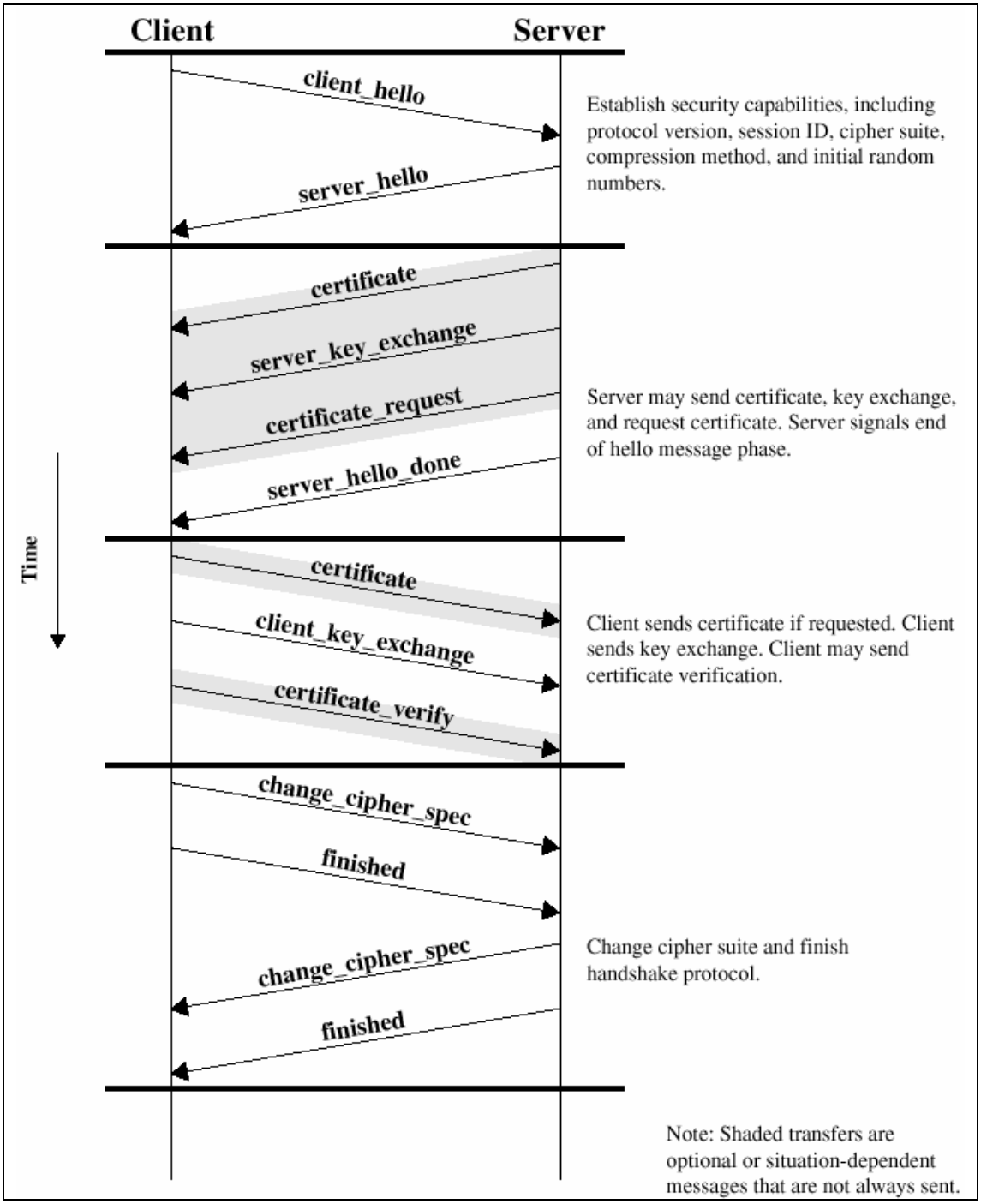
Gambar 8 Format Record

Di tempat penerima, sub-protokol *SSL Record* melakukan proses berkebalikan: mendekripsi data yang diterima, mengotentikasinya (dengan *MAC*), mendekompresinya, lalu merakitnya. Protokol SSL membuat kominkasi menjadi lebih lambat. Piranti keras, seperti kartu *peripheral component interconnect* (*PCI*)

dapat dipasang ke dalam *web server* untuk memproses transaksi *SSL* lebih cepat sehingga mengurangi waktu pemrosesan.

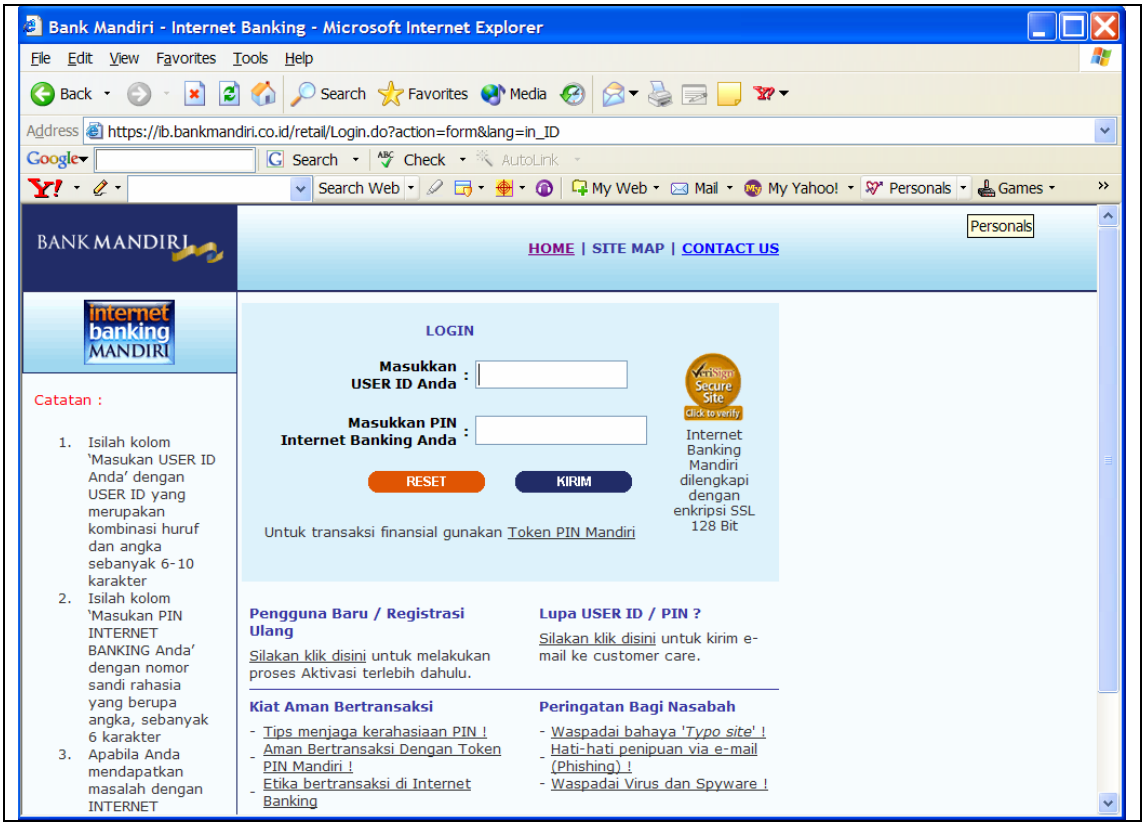
3.3 Protokol Handshake

SSL handshaking, yaitu sub-protokol untuk membangun koneksi (kanal) yang aman untuk berkomunikasi. Protokol ini merupakan bagian yang paling kompleks dari SSL. Melalui protocol ini, *server* dan *Client* dapat saling melakukan otentikasi satu sama lain. Menegosiasikan enkripsi, algoritma MAC dan kunci kriptografi. Digunakan sebelum aplikasi lain dikirim.



Gambar 9 Sub Protokol Handshake

Sampai di sini, proses pembentukan kanal yang aman sudah selesai. Bila sub-protokol ini sudah terbentuk, maka http:// pada URL berubah menjadi https:// (http secure)



Gambar 10 Hasil dari HTTPS

4. SSL Security Check List

Spesialis dari bidang keamanan telah banyak mempelajari hubungan antara SSL dengan aspek lain dari sistem yang mengimplementasikan SSL. Pada kenyataannya, walaupun tidak ada kecacatan dari segi keamanan yang diketahui pada protocol SSL, kelemahan lain dalam sistem yang menggunakan SSL, banyak ditemukan.

4.1 Isu Otentikasi

Otentikasi sepertinya berada pada bagian terbelakang dari enkripsi dalam diskusi keamanan, terutama dalam pemberitaan, dimana laporan dari pemecahan (cracking) algoritma kriptografi menerima ulasan utama. Otentikasi lebih penting daripada keamanan. Tidak ada sejumlah enkripsi yang dapat mencegah pihak yang tidak seharusnya mengirim kepada penyerang yang tidak terotentikasi sebuah kunci *session*. Meskipun menarik untuk melihatnya kembali, menentukan alamat isu otentikasi adalah penting dalam keamanan komunikasi.

Ada beberapa isu dari otentikasi SSL diselesaikan dengan sertifikat x.509. Walaubagaimanapun, beberapa isu otentikasi dikhusukan dengan SSL yang terpisah dari sertifikat kunci publik. Diantaranya adalah:

4.1.1 Otentikasi sertifikat

Otentikasi sertifikat atau yang dikenal dengan CA (*Certificate* Authority) ditandai dengan sertifikat x.509, dan semua penerapan SSL harus menetapkan apakah akan mempercayai CA sebagai peer dari komunikasi. Secara umum, penerapan membandingkan peer yang dimiliki oleh CA dengan internal list dari wewenang yang diketahui oleh penerapan sebagai yang 'dapat dipercaya'. Dengan pendekatan ini, sangat penting bagi penerapannya menggunakan kunci publik dari simpanan internal untuk memverifikasi sertifikat, daripada menggunakan kunci umum yang disediakan oleh sertifikat CA pada peer. Penyerang dapat membuat sertikat CA yang palsu yang sangat serupa dengan sertifikat CA yang sebenarnya dalam segala hal kecuali kunci umum, dengan menukarkan kunci umum disesuaikan dengan kunci pribadi yang dimiliki oleh penyerang. Hanya dengan mengembalikan kunci umum CA dari penyimapanan internal dapat diterapkan untuk mencegah serangan seperti ini.

Jika penerapannya menetapkan untuk menyimpan list internal dari CA yang dipercaya, maka harus sangat hat-hati memikirkan caranya, jika tidak, penerapan tersebut akan memperbolehkan pengguna untuk mengupdate list tersebut.

4.1.2 Tandatangan Sertifikat

Hal ini sangat jelas, tapi hal ini sangat mudah ditinjau; sebuah penerapan SSL harus memvalidasi semua sertifikat yang diterimanya dengan memverifikasi tandatangan CA dengan yang dimiliki olehnya.

4.1.3 Waktu Valid Sertifikat

Semua penerapan SSL harus juga memerikas waktu valid dari semua sertifikat. Perioda validitas melibatkan waktu ”not before” dan ”not after”, keduanya harus di verifikasi.

4.1.4 Penarikan Kembali Status Sertifikat

Penerapan SSL yang beroperasi pada lingkungan yang mendukung penarikan sertifikat harus memeriksa status penarikan dari setiap sertifikat sebelum menerimanya. Sayangnya, tidak semua lingkungan mendukung penarikan kembali status dari sertifikat secara efektif. Dalam hal seperti ini, penerapan SSL mungkin akan menyediakan alternatif lain bagi pengguna, mungkin dengan membolehkan pengiriman list dari sertifikat yang sudah di tarik kembali secara manual.

4.1.5 Subjek Sertifikat

Implementasi tidak hanya memastikan bahwa sertifikat valid, namun juga memastikan bahwa sertifikat tersebut mensertifikasi pihak yang tepat. Penyerang dapat memperoleh sertifikat dari pihak pengeluar sertifikat yang berwenang, sehingga pemeriksaan apakah sertifikat yang dimiliki oleh user adalah sertifikat yang benar, sangat diperlukan.
Cara implementasi memverifikasi sertifikat untuk subjek yang tepat bergantung pada kebijakan dari pihak yang mengeluarkan sertifikat. Contoh: VeriSign Class 3 menempatkan nama host dari website yang tersertifikasi pada field commonName dari subjek sertifikat. Browser seperti Netscape Navigator atau Internet Explorer akan memeriksa field ini terhadap nama host yang dimasukkan user melalui URL.

4.1.6 Diffie-Hellman Trapdoors

Ketika implementasi SSL menggunakan metode pertukaran *key* Diffie-Hellman, *server* akan menspesifikasikan sekumpulan parameter Diffie-Hellman. *Client* yang mensupport metode ini harus memeriksa parameter yang diterima dari *server*. *Client* harus memastikan bahwa *server* telah memilih nilai yang akan mendukung keamanan yang memadai.

4.1.7 Algoritma Rollback

Sebuah pesan *ServerKeyExchange* memungkinkan *server* SSL untuk mengirimkan informasi kunci publik kepada *Client*. *Client* perlu melakukan enkripsi terhadap rahasia premaster untuk *server*. Informasi kunci publik tersebut ditandatangani oleh *server* menggunakan kunci private yang berkorespondensi dengan kunci publik yang ada pada pesan sertifikat *server*. Algoritma kunci publik yang digunakan oleh *Client* tidak terspesifikasi secara eksplisit pada pesan *ServerKeyExchange*. Oleh karena itu informasi tesebut tidak ditandatangani oleh *server*. Hal ini dapat membuat protokol SSL rentan terhadap algoritma rollback attack.

Dalam algoritma rollback attack, penyerang memaksakan kedua belah pihak untuk memiliki pendapat yang berbeda dalam menentukan algoritma kunci publik yang digunakan untuk menandatangani rahasia premaster. Contohnya, *Client* mengira menggunakan algoritma RSA, sementara *server* mengira Diffie-Hellman. David Wagner dan Bruce Schneier menunjukkan bagaimana skenario ini dapat menghancurkan proteksi kriptografi. Jika penyerangan ini terjadi maka penyerang dapat membaca semua informasi yang mungkin atau membuat data palsu dengan mengatasnamakan salah satu pihak.
Agar terlindung dari algoritma rollback attack, implementasi *Client* SSL harus memverifikasi panjang dan jumlah parameter dari pesan *ServerKeyExchange*. Contohnya, RSA memiliki dua parameter sementara Diffie-Hellman menggunakan tiga parameter. Jika ada pesan yang diterima maka jumlah parameter harus diperiksa. Jika jumlah parameter tidak sama maka *Client* harus menolak *session* dan membangkitkan peringatan.

4.1.8 Dropped ChangeCipherSpec Messages

Protokol SSl tdak mengikutsertakan pesan ChangeChiperSpec pada kode autentikasi handshake yang dibawah oleh pesan finished. Pesan ChangeChiperSpec dihilangkan karena SSL tidak menganggap bahwa pesan ChangeChiperSpec adalah pesan protokol handshake. Penghilangan ini membuat implementasi SSL rentan terhadap serangan tertentu ketika implementasi menggunakan *session* authentication-only (tanpa enkripsi).
Untuk mengambil keuntungan terhadap kerentanan ini, penyerang akan menghapus pesan ChangeChiperSpec dari stream komunikasi. Kedua belah pihak (*server* dan *Client*) akan menerima pesan finished yang valid dan mulai melakukan pertukaran data tanpa pernah mengaktifkan chiper suite yang pernah dinegosiasikan (serangan ini tidak dapat dilakukan jika *session* menggunakan enkripsi. Pada kasus tersebut, pihak yang mengirim pesan finished akan mengenkripsi pesan tersebut, dan pihak yang menerima pesan finished, dengan melihat bahwa pesan ChangeChiperSpec tidak hilang, berharap bahwa pesan tersebut tidak terenkripsi).

Untungnya, penolakan terhadap serangan ini cukup mudah. Implementasi SSL seharusnya tidak akan menerima pesan finished yang tidak disertai dengan pesan ChangeChiperSpec.

4.2 Isu Enkripsi

SSL sangat efektif dalam menjaga kerahasiaan dari informasi. Hanya beberapa hal kecil saja yang dapat dipertimbangkan pada isu enkripsi ini, diantaranya adalah:

4.2.1 Ukuran *key* dari enkripsi

Kekuatan dari enkripsi yang ditawarkan oleh SSL merupakan isu yang penting. Kekuatan itu bergantung secara langsung dari ukuran kunci yang digunakan dari algoritma kriptografi simetrik, seperti RC4 dan DES. Secara teori, pengembang dapat menciptakan penerapan dari SSL dengan menggunakan panjang kunci yang cukup besar, dan implementasi yang seperti itu sudah pasti akan sulit dipecahkan.

4.2.2 Analisis Lalu Lintas

Penyerang dapat mempelajari banyak hal mengenai target yang akan diserang hanya dengan melakukan observasi mengenai lalu lintas dari dan menuju target tersebut, meskipun penyerang tidak dapat melakukan dekripsi informasinya. Analisis lalu lintas merupakan sebuah bentuk serangan yang sulit untuk dicegah apalagi di lingkungan yang terbuka seperti Internet.

Walau bagaimanapun, dalam setiap lingkungan, protokol SSL sendiri memperkenalkan analisis lalu lintas tambahan yang mudah diserang. Ketika SSL menggunakan stream cipher untuk melakukan enkripsi, ukuran dari hasil pesan yang di enkripsi tersebut dapat menunjukkan ukuran dari pesan yang belum di enkripsikan; penyerang hanya perlu mengurangi dengan ukuran kode pesan otentikasi. Namun demikian, kelemahan ini tidak muncul ketika enkripsi dilakukan dengan menggunakan block cipher, karena padding yang diterapkan pada block cipher telah secara efektif dapat menyembunyikan ukuran plaintext yang sebenarrnya. Implementasi SSL hanya memilih untuk mendukung enkripsi block cipher agar dapar mencegah serangan analisis lalu lintas.

4.2.3 Serangan Bleinchenbacher

Pada tahun 1998, Daniel Bleichenbacher, seorang peneliti dari Laboratorium Lucent Bell, melaporkan mengenai sebuah serangan terhadap protokol keamanan yang menggunakan enkripsi RSA, termasuk protol SSL. Serangan tersebut mengambil keuntungan dari cara algoritma enkripsi RSA menyandikan data sebelum melakukan enkripsi terhadapnya. Data sandi selalu dimulai dengan dua byte 00 dan 02.

Ada beberapa langkah yang dapat dilakukan dalam implementasi SSL untuk mengurangi dampak dari serangan ini. Diantaranya adalah dengan memeriksa dengan sangat teliti plaintext hasil dari dekripsi sebelum menerimanya sebagai hasil dekripsi yang valid.

4.3 Isu Umum

Masih banyak isu-isu lain yang tidak dapat dikategorikan sebagai isu otentikasi atau isu enkripsi. Isu-isu tersebut dapat dianggap sebagai isu umum,diantaranya adalah:

4.3.1 Ukuran Kunci RSA

Sebagian besar dari penerapan SSL adalah dengan menggunakan algoritma enkripsi RSA untuk digital signature dan enkripsi kunci public. Kekuatan dari algoritma RSA ini bergantung dengan ukuran kunci public dari RSA. Semakin panjang kunci, semakin aman implementasinya.

4.3.2 Versi Serangan Rollback

SSL versi 3.0 memperkanalkan banyak peningkatan daripada versi 2.0, termasuk peningkatan dalam hal keamanan protokol. Spesifikasi SSL menetapkan pendekatan yang sangat spesifik untuk melindungi dari serangan yang memaksa versi rollback. Meskipun demikian, ada satu area yang tidak ditetapkan dalam spesifikasi, yaitu pembukaan kembali *session* yang sebelumnya. Secara sepintas, penerapan SSL membolehkan sebuah *session* yang sebelumnya di bentuk dengan menggunakan versi 3.0 untuk dilanjutkan dengan menggunakan versi 2.0. Hati-hati dengan penerapannya, jangan membolehkan perilaku ini. Jika sebuah *session* terbentuk dengan menggunakan versi 3.0, maka penerapannya harus memastikan bahwa semua usaha untuk melanjutkan *session* harus tetap menggunakan versi 3.0.

4.3.3 Pengkahiran yang Belum Pada Waktunya

Ancaman dari serangan pemotongan berdasarkan pada pengakhiran yang belum pada waktunya dari sebuah *session* SSL merupakan salah satu isu keamanan umum. Jika seorang penyerang dapat menghapus pesan pada protokol ketika sedang terkirim, penyerang itu dapat membuat skenario dimana satu atau dua pihak yang saling berkomunikasi hanya menerima sebagian informasi. Jika informasi yang hilang sangat penting dalam komunikasi, penyerang dapat membahayakan keamanan secara keseluruhan dari pertukaran. Protokol SSL menyediakan pesan Closure*Alert* untuk melindungi dari serang yang bertipe seperti ini. Sayangnya, tidak semua lingkungan dapat bergantung pada Closure*Alert*. Pengguna web browsing contohnya, hanya dengan mematikan komputer mereka setelah menyelesaikan transaksi, sebelum komputer tersebut sempat mengirimkan pesan Closure*Alert*. Perlindungan yang lebih baik jika aplikasi yang menggunakan keamanan SSL lebih teliti dalam menangani kemungkinan terjadinya pengakhiran yang belum pada waktunya.

4.3.4 Nilai ID dari Session

Spesifikasi SSL memberikan *server* fleksibilitas secara lengkap dalam memilih nilai ID dari *session*. Dalam membuat pemilihan, *server* harus dengan hati-hati menerapkan dengan tidak mengikutsertakan informasi yang penting. Nilai ID dari *session* ditransfer dalam pesan *ClientHello* dan *ServerHello* sebelum enkripsi di aktifkan. Nilainya akan sangat terbuka untuk dilihat oleh penyerang.

4.3.5 Generasi Nomer Acak

Nomer acak merupakan operasi yang cukup penting dalam SSL. Nomer acak dipertukarkan pada pesan *ClientHello* dan *ServerHello* sangat menunjukkan kunci dari enkripsi pada *session*. Nomer acak, bagaimanapun juga, memberikan tantangan menarik bagi sistem komputer, perangkat lunak tidak dapat melakukan apapun secara acak. Penerapan dari perangkat lunak hanya bergantung ke algoritma yang dikenal sebagai pseudorandom number generators. Algoritma ini mensimulasikan pengacakan secara sebenarnya dengan perhitungan matematik.

Ada dua buah masalah yang ada pada pseudorandom number generators yang harus dipikirkan SSL dan penerapan dari keamanan yang lain. Masalah yang pertama adalah keefektifan dari algoritmanya sendiri. Kebanyakan dari library perangkat lunak membangkitkan pesudorandum number dengan menggunakan algoritma linear congruential generator. Meskipun algoritma seperti diatas dapat menjadi pseudorandom number generators yang efektif, algoritma tersebut juga bisa menjadi kurang efektif. Ditambah lagi, banyak pengembang yang mencari pembuktian pada algoritma dasar dengan cara yang diperoleh, pada kenyataanya, malah menbawa malapetaka.

Masalah yang lebih serius dengan linear congruential generators adalah mereka sequential , dan dapat diprediksikan secara lengkap. Jika parameter dari algoritma diketahui dan satu dari nilai yang spesifik, sangat mudah untuk memprediksikan semua nilai dimasa yang akan datang yang akan dibangkitkan oleh algoritma. Nomer acak yang dapat diprediksi merupakan masalah yang serius bagi sebuah protokol keamanan., seperti

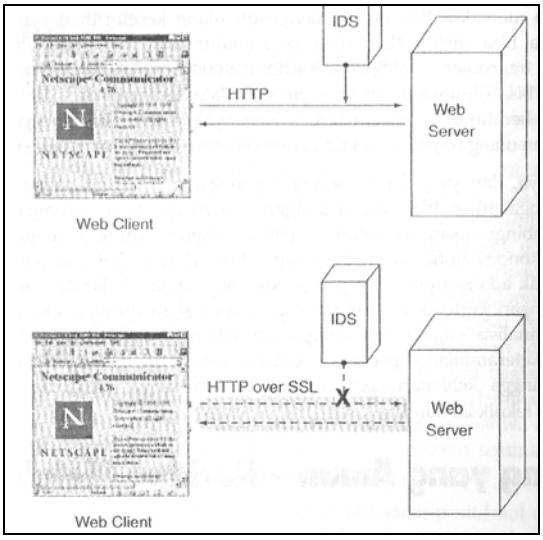
mereka mengizinkan penyerang untuk merencanakan dan menyiapkan diri meraka dengan baik untuk menyerang, menunggu mungkin sebuah nilai yang membahayakan untuk mucul. Penerapan SSL bagaimanapun harus sangat berhati-hati. Apalagi dengan penggunaan library pseudorandom number generators. Standard algoritma kriptografi termasuk enkripsi dan algoritma hash dapat dimodifikasi untuk menyediakan nomer acak yang efektif.

4.3.6 Benih Nomer Acak

Tanpa menghiraukan penerapan algoritma yang digunakan untuk pembangkitan nomer acak, penerapan pada umumnya harus menyediakan algoritma tersebut dengan inisiasi titik mulai atau benih. Dalam aplikasi selain daripada keamanan, kebutuhan utama dari titik mulai ini adalah ia akan berbeda setiap kali pembangkitan dilakukan. Dalam aplikasi keamanan, titik mulai yang dimaksud tidak saja harus acak, tapi juga tidak dapat diprediksi.

5. Hacking yang Aman Melalui SSL

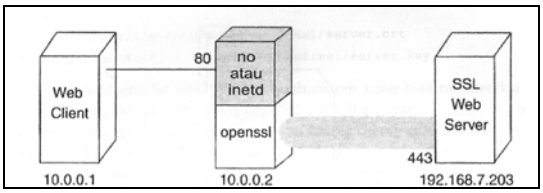
Dalam hal pendeteksi gangguan pada IDS (*Intrusion Detection Sytem*), yang merupakan system pendeteksi gangguan, SSL merupakan halangan terbesar. Kebanyakan IDS berpedoman pada paket *sniffing* jaringan untuk mengumpulkan aktivitas data. Jika itu sendiri terenkripsi, maka tidak bisa dianalisis untuk bisa menentukan apakah aktivitas itu mencurigakan atau tidak.. Semua IDS yang berdasar pada paket *sniffing* pasti akan mengabaikan serangan yang lewat pada SSL. Gambar 11 menggambarkan bagaimana IDS tidak efektif melawan serangan melalui jalur SSL.



Gambar 11 IDS Dikalahkan oleh SSL

5.1 Melancarkan Serangan Melalui Jalur SSL

Menggunakan browser untuk melancarkan serangan HTTP melalui SSL adalah mudah. Satu-satunya yang harus dilakukan oleh penyerang adalah menggunakan URL dengan awalan HTTPS bukan HTTP. Browserlah yang akan menangani *session* negosiasi SSL dan enkripsinya. Tetapi, jika penyerang ingin menjalankan script atau tools untuk mengirim serangan melalui HTTP, namun tidak memiliki fungsi SSL padanya, maka ia akan menggunakan teknik yang disebut *tunneling*. Tunneling SSL memerlukan program penerus port yang terhubung pada port 80 untuk request HTTP standar dan meneruskan ke host tertentu pada koneksi SSL terenkripsi. Dengan cara ini, serangan yang dilancarkan terhadap SSL tunnel secara otomatis terenkripsi dan diteruskan ke sistem target.



Gambar 12 Tunnel SSL, Menggunakan Inetd dan OpenSSL

Mengkonstruksi sebuah *tunnel* SSL dengan OpenSSL adalah sangat mudah, khususnya pada sistem UNIX yang menggunakan inetd.

5.2 Mendeteksi Gangguan Melalui SSL

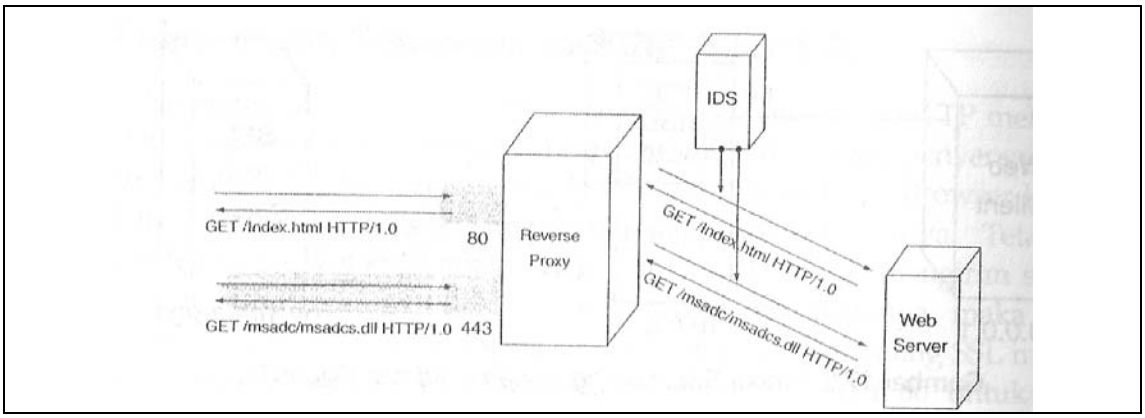
Untuk mendeteksi gangguan melalui SSL, menggunakan *reverse* HTTP merupakan sebuah solusi terbaik. Gambar dibawah ini menunjukkan bagaimana *reverse* HTTP dapat dipakai untuk menangkap lalu lintas SSL sebelum ia mencapai ke *server* dan melewatkan lalu intas HTTP teks biasa ke web*server*. IDS diletakkan antara reverse HTTP proxy dan web *server*. Dengan cara ini, IDS dapat mengambil signature dalam lalu lintas HTTP.

Satu-satunya kekurangan dari sistem ini adalah bahwa sumber alamat IP dari sestem penyerang diganti dengan alamat IP sistem *reverse HTTP proxy*. Untuk melacak sumber alamat IP, penyerang harus menghubungkan log *reverse HTTP proxy* dengan catatan waktu pada log IDS.

5.3 Mendeteksi Lalu Lintas SSL

Jika *session key* dapat dipulihkan, aliran data dapat didekripsi jika bisa ditangkap. Satu-satunya cara untuk memulihkan *session key* SSL adalah sewaktu ia dalam pertukaran antara *server* dan browser. Tetapi pertukaran ini dilakukan dengan enkripsi kunci publik, Dengan demikian untuk mendekripsi kunci *session*, penyerang perlu akses ke sertifikat SSL *server* dan *Client*, bila salah satunya digunakan.

Dengan mengetahui kunci privat *server* SSL, penyerang bisa mendekripsikan data dari koleksi SSL jika pendeteksi atau *sniffer* SSL dimulai sebelum koneksi SSL dibuat. Sniffer menjadi ssldump, yang merupakan program *tcpdump SSL-enabled* sudah tersedia. Untuk rincian bagaimana *ssldumb* berkerja dan dipakai, dapat dilihat pada Gambar 13 dibawah ini.



Gambar 13 Reverse HTTP Proxy dan IDS

5.4 Dekripsi SSL

Dalam hubungannya dengan pendeteksian gangguan pada lalu lintas Web, SSL menjadi rintangan yang terbesar. Jika menggunakan IDS, yang diharapkan menjadi penengah yang menangkap lalu lintas jaringan sebelum mencapai titik akhir dan menganalisis apakah ada *signature* yang berisi serangan, dan SSL yang didesain dengan tujuan untuk menemukan apakah ada penangkapan lalu lintas data yang tidak efektif, maka mendesain IDS yang bisa mengolah SSL merupakan latihan yang bsia mengalahkan tujuan utama dari SSL itu sendiri.

Tetapi, seperti yang telah disebutkan sebelumnya, menerapkan IDS dengan setifikat dan kunci privat SSL dari sisi *server* untuk menjalankan dekripsi SSL atau untuk menerapkan *reverse* HTTP proxy yang bisa mendekripsikan lalu lintas SSL kemudian melewatkannya ke Web *server backend.* Pada situasi selanjutnya, IDS bisa diposisikan diantara *reverse* HPPT proxy dan web *server beckend.*

6. Daftar Pustaka

[01] Stuart McClure, Web Hacking Serangan dan Pertahanannya, Penerbit ANDI Yogyakarta

[02] Munir Rinaldi, Diktat Kuliah IF5054 Kriptografi, Program Studi Teknik Informatika

[03] Stephen Thomas, SSL and TSL Essential Securing the Web, Wiley Computing Publishing

[04] Bruce Schneier. *Applied Cryptography Second Edition.* John Wiley and Sons, Inc., 1996