PROTOCOL HTTP DAN HANDSHAKING CLIENT-SERVER UNTUK BERKOMUNIKASI VIA HTTPS

Ferdian Pramudya P (32582)
Agung kaharesa W (32649)

Jurusan Teknik Elektro FT UGM
Yogyakarta

I. PENDAHULUAN

Proses negosiasi otomatis yang secara dinamis menentukan parameter dalam pembentukan kanal komunikasi antara dua entitas normal sebelum komunikasi melalui channel dimulai, biasa disebut dengan istilah **handshaking**. Biasanya proses ini terjadi bila komputer ingin untuk berkomunikasi dengan perangkat luar pada network untuk membuat aturan untuk dapat berkomunikasi dengan baik.

Biasanya untuk mentransfer dokumen dalam *World Wide Web* (WWW) kita menggunakan HTTP (HyperText Transfer Protocol). Protokol ini adalah protokol ringan, tidak berstatus dan generik yang dapat dipergunakan dalam berbagai macam tipe dokumen. Namun dalam protokol ini segi keamanan data yang dikirim belum diperhatikan, data yang dikirimkan tidak dienkripsi, sehingga data yang dikirim rawan, dapat dibaca/ dirusak oleh orang yang tidak diizinkan. Jika data itu tidak penting tidak menjadi masalah, tetapi apabila data itu sangat privasi dan penting seperti pada saat kita ingin melakukan transaksi online , Bank ingin mengirimkan informasi keuangan rahasia nasabahnya tentunya segi keamanannya sangat rentan dan harus diperhitungkan..

Oleh karena itu, kita pergunakan protokol yang dapat mendukung segi keamanan yaitu, **HTTPS** (HTTP melalui SSL or HTTP Secure), merupakan protokol HTTP yang menggunakan Secure Socket Layer **(SSL)** atau Transport Layer Security (TLS) sebagai sublayer dibawah HTTP aplikasi layer yang biasa. HTTP di enkripisi dan deskripsi dari halaman yang diminta pengguna serta halaman yang dikembalikan oleh web server. HTTPS digunakan untuk melindungi dari orang mengakses tanpa izin dan dari serangan *man-in-the-middle.* HTTPS dikembangkan oleh Netscape. Dengan HTTPS kita dapat melakukan proteksi data yaitu hanya penerima saja yang dapat membaca data, Kenyamanan (data privacy), memungkinkan identifikasi server ataupun client, otentikasi server dan klien, dan integritas data.

Sedangkan **SSL** (Secure Socket Layer) adalah arguably internet yang paling banyak digunakan untuk enkripsi. Ditambah lagi, SSL sigunakan tidak hanya keamanan koneksi web, tetapi untuk berbagai aplikasi yang memerlukan enkripsi jaringan end-to-end.

II. PEMBAHASAN

A. *Handshaking*

Handshaking adalah proses negosiasi otomatis yang secara dinamis menentukan parameter dalam pembentukan kanal komunikasi antara dua entitas normal sebelum komunikasi melalui kanal dimulai. Ia mengikuti pembentukan fisik saluran precedes normal dan mentransfer informasi. Contohnya : ketika sebuah komputer berkomunikasi dengan perangkat lain seperti modem atau printer yang perlu melakukan handshake untuk membuat sambungan.

Proses negosiasi SSL atau "handshake," melibatkan pertukaran *cryptographic keys*, certificate,dan informasi lain , random data digunakan untuk membuat enkripsi satu waktu, dan valuenya digunakan untuk mengidentifikasi SSL yang dibuat dari handshake. Handshake memiliki tiga tujuan:

• Untuk membolehkan client dan server setuju mengenai algoritma yang akan mereka gunakan.

• Untuk melibatkan kumpulan dari *crypto keys* untuk digunakan oleh algoritma tersebut.

• Untuk mengautentikasi klien.

Catatan penting bahwa SSL Handshake memerlukan perhitungan yang sangat kompleks dan perlu komputer dengan processor yang tangguh.Pada akhir *cryptographic key* dibuat dan dipertukarkan antara client dan server, enkripsi berikutnya dibuat cukup mudah sejauh prosesor dari komputer terfokus, namun hal itu tetap menjadikan perlunya performa tinggi dari server. Terutama ketika handshake dengan jumlah besar terjadi dalam waktu bersamaan. Bagaimanapun juga, pekerjaan ini dapat dilakukan oleh processor khusus/spesial yang didesain khusus untuk memproses perhitungan matematis yang melibatkan handshake.

B. *HTTP Protocol*

HTTP adalah sebuah protokol meminta/menjawab antara client dan server. Sebuh client HTTP seperti *web browser*, biasanya memulai permintaan dengan membuat hubungan TCP/IP ke port tertentu di tuan rumah yang jauh (biasanya port 80). Sebuah server HTTP yang mendengarkan di port tersebut menunggu client mengirim kode permintaan (request), seperti "GET / HTTP/1.1" (yang akan meminta halaman yang sudah ditentukan), diikuti dengan pesan MIME yang memiliki beberapa informasi kode kepala yang menjelaskan aspek dari permintaan tersebut, diikut dengan badan dari data tertentu. Beberapa kepala (header) juga bebas ditulis atau tidak, sementara lainnya (seperti tuan rumah) diperlukan oleh protokol HTTP/1.1. Begitu menerima kode permintaan (dan pesan, bila ada), *server* mengirim kembali kode jawaban, seperti "200 OK", dan sebuah pesan yang diminta, atau sebuah pesan error atau pesan lainnya.

Pengembangan HTTP dikoordinasi oleh Konsorsium World Wide Web (W3C) dan grup bekerja Internet Engineering Task Force (IETF), bekerja dalam publikasi satu seri RFC, yang paling terkenal RFC 2616, yang menjelaskan HTTP/1.1, versi HTTP yang digunakan umum sekarang ini.

C. HTTPS, TLS, and SSL

Kini telah terdapat cara untuk menangani masalah keamanan web. Yaitu dengan HTTPS,. **https** adalah versi aman dari HTTP, protokol komunikasi dari World Wide Web. Ditemukan oleh Netscape Communications Corporation untuk menyediakan autentikasi dan komunikasi tersandi dan penggunaan dalam komersi elektris.

Pendekatan HTTPS sangatlah simpel, Client membuat koneksi ke server, melakukan negosiasi koneksi SSL, kemudian mengirim HTTP tersebut melalui aplikasi SSL. Deskripsi ini menjadikannya terlihat mudah.

Selain menggunakan komunikasi plain text, HTTPS menyandikan data sesi menggunakan protokol SSL (Secure Socket layer) atau protokol TLS (Transport Layer Security). Kedua protokol tersebut memberikan perlindungan yang memadai dari serangan eavesdroppers, dan man in the middle attacks. Pada umumnya port HTTPS adalah 443.

Terdapat perbedaan *port* yang spesifik, HTTPS menggunakan *port* 443 sedangkan HTTP menggunakan *port* 80 dalam berinteraksi dengan *layer* yang di bawahnya , TCP/IP/

HTTPS dan SSL mendukung penggunaan dari X.509 sertifikat digital dari server, sehingga jika diperlukan , pengguna dapat mengotentikasi pengirimnya. Kecuali perbedaan *port* yang spesifik, HTTPS menggunakan *port* 443 sedangkan HTTP menggunakan *port* 80 dalam berinteraksi dengan *layer* yang di bawahnya , TCP/IP/

Tingkat keamanan tergantung pada ketepatan dalam mengimplementasikan pada browser web dan perangkat lunak server dan didukung oleh algorithma penyandian yang aktual.Oleh karena itu, pada halaman web digunakan HTTPS, dan URL yang digunakan dimulai dengan ‘https://’ bukan dengan ‘http://’

Efektifitas dari HTTPS dapat dibatasi oleh kurangnya implementasi dari browser atau perangkat lunak server atau kurangnya dukungan dari beberapa algoritma. Selanjutnya, walapun HTTPS dapat mengamankan perjalanan data antara server dan klien, setelah data didekripsi tujuannya, itu hanya aman sebagai *host* komputer

Kesalahpahaman yang sering terjadi pada pengguna kartu kredit di web ialah dengan menganggap HTTPS “sepenuhnya” melindungi transaksi mereka. Sedangkan pada kenyataannya, HTTPS hanya melakukan enkripsi informasi dari kartu mereka antara browser mereka dengan web server yang menerima informasi. Pada web server, informasi kartu mereke secara tipikal tersimpan di database server (terkadang tidak langsung dikirimkan ke pemroses kartu kredit), dan server database inilah yang paling sering menjadi sasaran penyerangan oleh pihak-pihak yang tidak berkepentingan

Mengapa HTTPS :

1. Melindungi data dari akses yang tidak diijinkan, hanya penerima yang diijinkan untuk membaca data
2. Menjaga kerahasiaan data (data privasi).
3. Integritas data
4. Klien dan server autentikasi
5. Memastikan bahwa tidak ada yang bisa merusak data yang ditransmisikan.

SSL

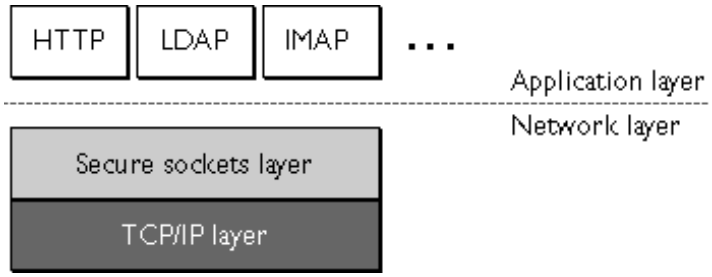
Secure Sockets Layer (SSL) merupakan sistem yang digunakan untuk mengenkripsi pengiriman informasi pada internet, sehingga data dapat dikirim dengan aman. Protokol SSL mengatur keamanan dan integritas menggunakan enkripsi, autentikasi, dan kode autentikasi pesan.

Tanpa SSL :
Layer 7 Application
Layer 6 Presentation
Layer 5 Session
Layer 4 TCP, UDP
Layer 3 Network Layer (IP)
Layer 2 Data Link Layer (Ethernet, Token Ring, ATM, etc.)
Layer 1 Physical Layer (twisted pair, coax, fiber, wireless)

Dengan SSL

Layer 7 Application
Layer 6 Presentation
Layer 5 Session
Layer 4+ SSL
Layer 4 TCP, UDP
Layer 3 Network Layer (IP)
Layer 2 Data Link Layer (Ethernet, Token Ring, ATM, etc.)
Layer 1 Physical Layer (twisted pair, coax, fiber, wireless)

SSL protocol menyedian privasi komunikasi di internet. SSL tidak mendukung file-encryption, access-control, atau proteksi virus, jadi SSL tidak dapat membantu mengatur data sensitif setelah dan sebelum pengiriman yang aman.



Gambar 1. SSL berjalan di atas TCP / IP dan di bawah tingkat tinggi protokol aplikasi. [9]

Protokol SSL berjalan di atas TCP/IP, di bawah protokol tingkat tinggi seperti HTTP/IMAP. Berikut merupakan kemampuan SSL untuk berkomunikasi melalui internet dan TCP/IP :

- SSL server authentication : mengizinkan pengguna untuk mengkonfirmasi identitas server. SSL memungkinkan perangkat lunak klien menggunakan teknik standar dari kriptografi *public key* untuk memeriksa sertifikat sever dan public ID itu sah. Konfirmasi ini penting misalnya ketika , akan mengirimkan nomor kartu kredit melalui jaringan dan ingin memeriksa identitas server penerima.
- SSL client authentication, mengizinkan server untuk mengkonfirmasi identitas klien. Menggunakan teknik yg sama dengan SSL server authentication. SSL memungkinkan perangkat lunak server untuk memriksa sertifikat klien dan public ID dari *certificat authority* (CA). Konfirmasi ini penting, misalnya ketika Bank mengirimkan informasi keuangan rahasia nasabahnya dan ingin memeriksa identitas penerima.
- Encrypted SSL connection, semua informasi yang dikirim antatra klien-server untuk dienkripsi dan dekripsi, sehingga data yang dikirim memiliki tingkat kerahasiaan yang tinggi. Kerasahasiaan sangat penting bagi kedua pihak untuk setiap transaksi. Setiap data yang dikirim melalui SSL dilindung oleh mekanisme yang menjaga data dari kerusakan.

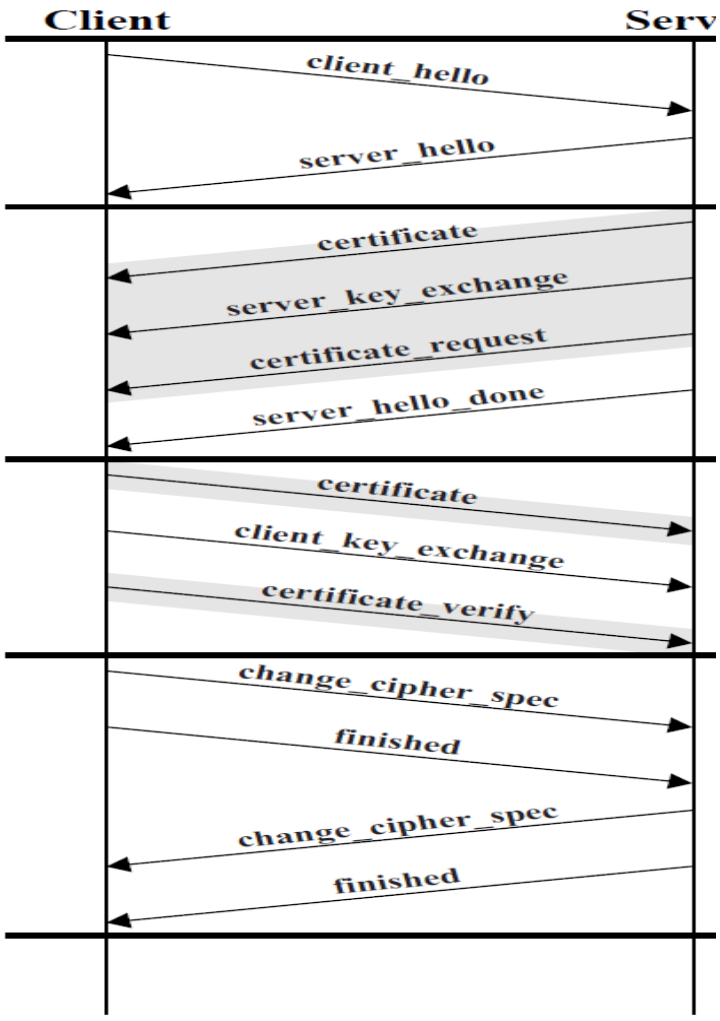
Protokol SSL terdiri dari dua sub-protokol: SSL *record protocol* dan SSL *handshake protocol.* SSL *record protocol* mendefinisikan format yang digunakan untuk mentransmisikan data. Sedangkan SSL *handshake protocol melibatkan* SSL *record protocol* untuk menukarkan serangkaian pesan antara SSL *enabled server* dan *SSL enable client* ketika keduanya pertama kali melakukan koneksi SSL. Pertukaran pesan tersebut digunakan untuk memfasilitasi tindakan sebagai berikut :

- Autentikasi dari server ke klien
- Mengizinkan klien dan server untuk memilih algoritma kriptografi atau sandi, yang mendukung komunikasi keduanya.
- Autentikasi dari klien ke server.
- Menggunakan teknik enkripsi public key untuk membuka data yang dienkripsi
- Membuat enkripsi koneksi SSL

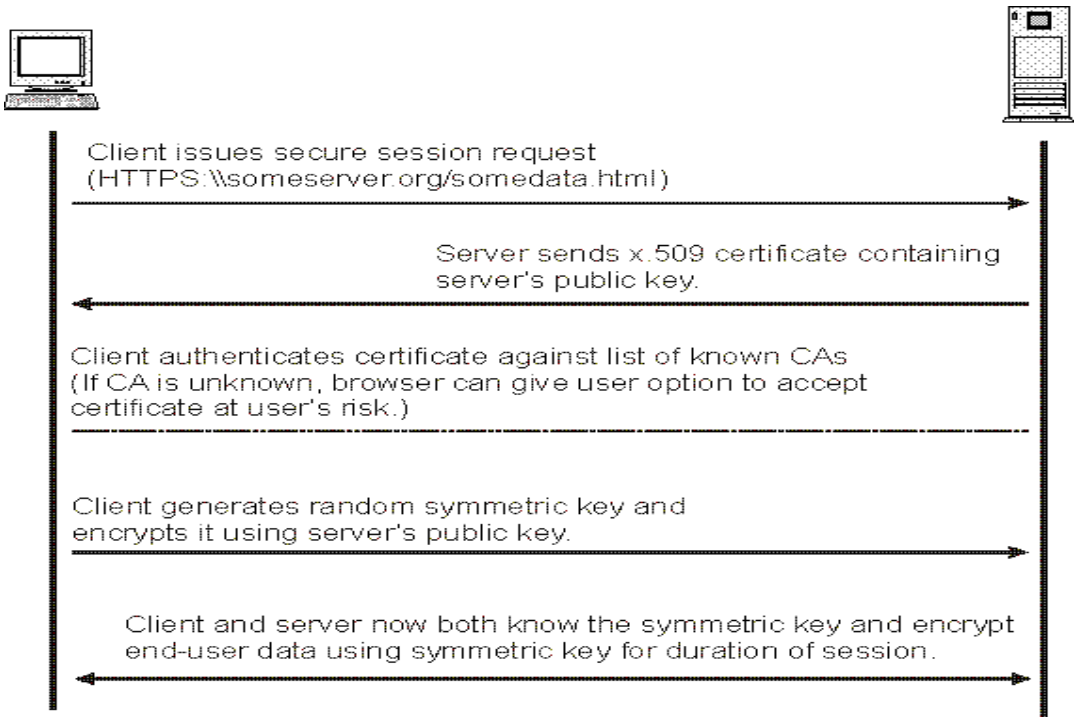
D. HTTP Handshake via SSL

HTTP berbasis koneksi SSL selalu dimulai klien dengan menggunakan URL (https :// daripada http://). Pada awal sesi SSL, SSL melakukan *handshake. Handshake* ini menghasilkan kriptorafi pada sesi tersebut. Cara *Handshake* akan dijelaskan pada diagram berikut :

er



Gambar 2 Handshaking https client-server (1). [8]



Gambar 3 Handshaking https client-server (2).[5]

1. Klien mengirimkan pesan “hello” yang berisi daftar kemampuan kriptografi (diurutkan dalam preferensi order klen), seperti SSL, deretan sandi tersebut didukung oleh klien dan metode data kompresi juga diukung oleh klien. Pesan tersebut terdiri dari 28-byte angka acak.
2. Server merespon pesan “hello” yang berisi metode kriptografi (deretan sandi) dan metode kompresi data yang dipilih oleh server, *session ID*, dan angka acak yang lain.
   Catatan : Klien dan server harus mendukung sedikitnya satu deretan sandi, server umumnya memilih deretan sandi yang paling kuat.

3. Server mengirimkan sertifikat digital (dalam contoh ini, server menggunakan X.509V3 dengan SSL). Jika server menggunakan SSL V3 dan jika aplikasi server (contoh, web server) memerlukan sertifikat digital untuk autentikasi klien, server mengitrimkan “*digital certificate request*”

4. Server mengirim pesan “hello done: dan menunggu respon dari klien

5. Ketika mwnwrima pesan “hello done” dari server, klien(*web browser*) memverifikasi keabsahan dari dsertifikat digital server, dan memerika “hello” server paramater cocok/dapat diterima. Jika server meminta sertifikat digital klien, klien mengirimnya,atau jika sertifikat yang cocok tidak tersedia, klien

mengirim sinyal “no digital certificate”. Sinyal ini hanya sebagai peringatan, tapi merupakan sesi aplikasi dari server dapat gagal jika klien autentikasi diperintahkan/bersifat perintah.

6. Klien mengirimkan pesan “client key exchange” yang berisi *premaster secret, 46 byte* nomor acak digunakan pada pembangkitan kunci enkripsi simetris dan kunci *message authentication code* (MAC), terenkripsi dengan *public key* pada server. Jika klien mengirimkan sertifikat digital ke server, klien mengirimkan pesan tanda tangan “digital certificate verify” dengan *private key* dari klien. Dengan verifikasi tanda tangan dari pesan, server dapat secar jelas melakukan verifikasi kepemilikan dari sertifikat digital klien.

   Catatan : Proses tambahan untuk melakukan verifikasi sertifikat digital tidak diperlukan. Jika server tidak mempunyai *private key* yang dimiliki sertifikat digital. Ia tidak dapat melakukan dekripsi *premaster secret* dan membuat kunci yang benar untuk algoritma enkripsi simetris dan handshake gagal.

7. Klien menggunakan serangkain operawi kriptografi untuk mengubah *premaster secret* ke *master secret*, dari semua kuci yang dibutuhkan untuk enkripsi dan memperoleh autentikasi pesan. Selanjutnya klien mengirimkan pesan “change cipher spec” untuk membuat server beralih ke deretan sandi(cipher suite) yang baru. Pesan berikutnya dikirim oleh klien (pesan “finished”) yang merupakan pesan pertama yang dienkripsi dengan metode *chper* dan *keys* (kunci).
8. Server merespon dengan “change cipher spec” dan pesan “finished”
9. SSL handshake selesai dan enkripsi aplikasi data dapat dikirim.

SSL handshake memberi dampak pada penggunaan aplikasi Web dan desain; satu harus diingat di mana ia berada dan bagaimana biasanya digunakan. Foremost, SSL (sebagaimana
ditetapkan dalam namanya) adalah socket-layer antarmuka. Di susunan network stack model, ini berarti bahwa SSL di bawah aplikasi / presentation layer, dan diatas transportation layer. Bertentangan dengan apa yang banyak orang pikir, SSL tidak bersatu dengan HTTP. Program level aplikasi - bukan hanya HTTP - dapat berjalan dalam suatu koneksi SSL ! 1 Namun demikian, HTTP adalah protokol yang berjalan di dalam SSL.

SSL protokol yang menyediakan privasi komunikasi melalui Internet. SSL tidak mendukung file-enkripsi, akses-kontrol, atau perlindungan virus, sehingga SSL tidak dapat membantu mengelola data sensitif sebelum dan setelah pentransportan. Protokol khusus untuk
membolehkan klien dan server untuk berkomunikasi tanpa peniruan, pengaksesan yang tidak diinginkan, sabotase, dan pemalsuan. Seperti halnya enkripsi sistem, ada satu aspek yang penting pengelolaan melekat dalam penggunaan SSL.

IETF yang telah mencakup SSL sebagai bagian utama dari Transaction Layer

Security (TLS) standard. Walaupun TLS adalah berdasarkan SSL, TLS memperluas SSL di beberapa cara, dan dua protokol tidak interoperate secara langsung. Perbedaan yang jelas antara SSL dan TLS adalah TLS yang mendukung jenis kunci-negosiasi selain SSL publik-key handshake. Semuanya, TLS mengembalikan beberapa crypto-pilihan fleksibilitas (dan kompleksitas) yang telah menjadi usang sejak SSL's awal.

- Mengapa ia digunakan?

Dalam setiap sesi SSL, server (menanggapi) sisi koneksi HARUS mengotentikasi sendiri ke sisi client (requesting). Ancamannya adalah bahwa klien mungkin meminta layanan tertentu tetapi membalas permintaan yang mungkin datang dan mungkin dipantau oleh eavesdropper. Untuk mengcounter kemungkinan dari balasan server palsu, server membuktikan identitasnya kepada server. Pada SSL C3.0 server mungkin perlu juga bagi client untuk meng authenticate dirinya sendiri ke server, hasilnya biasa dikenal dengan "mutual authentication". SSL mendukung otentikasi dengan kriptografi kunci publik, dan melindungi kerahasiaan, dan integritas pesan dengan *simetric key cryptography*.

- Apa yang perlu dikhawatirkan?

Satu dari desain fitur penting HTTP adalah statelessness. server untuk stateless protokol tidak mempertimbangkan klien, jadi server dapat direplikasi untuk load manajemen, dan dapat gagal dan restart, semua tanpa koordinasi dengan sisi klien. Dalam memilih protokol stateless, desain dari HTTP menghindari banyak kompleksitas dari jaringan berbasis lingkungan transaksi. Tetapi, SSL tidak stateless, dan ini menyulitkan performa kinerja SSL.

Dalam stateless protocol, hanya server yang stateless. Masing-masing client mengingat apa yang server perlukan untuk mengetahui mengenai client's session. Efek samping yang paling signifikan dari HTTP statelessness timbul karena sebagian besar load-halaman memerlukan koneksi overhead substansial. Misalnya, untuk membaca sebuah halaman dengan empat gambar, HTTP/1.0 memerlukan lima TCP connections terpisah, satu untuk setiap HTTP GET request With SSL, ini tampaknya menyiratkan biaya SSL handshake untuk setiap satu koneksi GET. Untungnya, untuk kebanyakan aplikasi Anda tidak perlu mengulangi handshake yang berulang-ulang.

Tabel 1 step by step handshaking.[3]

Step	Client/Server	Purpose
I	Establish connection	If a resumed session, GOTO step III; Else, establish protocol version, session ID, cipher suite, compression method, and client/server random values
II	Exchange certificates or keys	Authentication and Signing, i.e., X.509 certificate and/or key exchange and validation; Uses expensive public key encryption & signature
III	(Re)Establish session and connection state	Save session and connection state information for resuming future connections (more) quickly; Uses inexpensive hashing operations.
IV	Handshake Completion	Confirm client/server key agreement before application data is transferred

Secara default, handshake dilakukan setiap kali klien membuat koneksi TCP ke server. Sebuah download memerlukan banyak koneksi, karena sebagian besar dokumen HTML berisi gabungan dari beberapa objek, termasuk HTML teks, JPEG / JPG / GIF / BMP gambar, QT / MOV film, MPEG / MPG video, audio RAM, dokumen PDF, dan lain-lain Untuk menghindari hal-hal yang overhead ini, sebagian besar server mendukung gagasan sesi yang kembali. yaitu, walaupun HTTP adalah protokol stateless, SSL bukan. SSL memaintenance dua jenus state yang berbeda: state, dan state per koneksi. Setiap Klien dan server memiliki beberapa sesi dan setiap sesi memiliki beberapa koneksi. Setiap koneksi memiliki keys yang unique, apakah sesi ini baru atau resumed one, namun mahalnya operasi public key terjadi sekali tiap sesi.

- Kapankah Handshake Terjadi?

SSL handshake terjadi setiap kali klien memulai sesi dengan server baru atau bila salah satu pihak memulai sesi karena alasan apapun, termasuk tanpa alasan baik, ingin membuat Session ID dan Spesifikasi Cipher baru. SSL handshake yang biasanya tidak dilakukan pada setiap GET obyek yang demikian, misalnya, dengan halaman empat gambar dan dua video kemungkinan akan terjadi satu handshake dan tujuh file opens. Sejak symmetrickey enkripsi sangatlebih cepat. Dari public-key operation handshake, multi-layar melindungi transaksi harus tinggal dalam satu sesi SSL, bahkan jika ini berarti bahwa beberapa halaman perlu dienkripsi. Hampir semua aplikasi SSL harus menggunakan non-standar resumable SSL sesi.

III. KESIMPULAN

1. HTTP merupakan protokol yang dipergunakan untuk mentransfer dokumen dalam *World Wide Web*(WWW), termasuk protokol ringan, tidak berstatus dan generik yang dapat dipergunakan pada berbagai macam dokumen dan terdapat pada port 80
2. Handshaking adalah proses negosiasi otomatis yang secara dinamis menentukan parameter dalam pembentukan kanal komunikasi antara dua entitas normal sebelum komunikasi melalui kanal dimulai
3. **HTTPS** (HTTP melalui SSL or HTTP Secure), merupakan protokol HTTP yang menggunakan Secure Socket Layer **(SSL)** atau Transport Layer Security (TLS).
4. SSL handshake terjadi setiap kali klien memulai sesi dengan server baru atau bila salah satu pihak memulai sesi karena alasan apapun.
5. SSL dibuat untuk memberikan keamanan data dan kemudahan untuk autentikasi dari server ke klien dan menyediakan enkripsi yang efisien bila digunakan dengan beberapa protokol seperti HTTP. Beberapa fleksibilitas dari SSL tidak dibutuhkan, karena SSL mendukung banyak turunan sandi yang sekarang tidak relevan digunakan, Bagian yang berharga dari fleksibilitas SSL adalah pilihan autentikasi klien dan *multiple connection resumable session.* Selain fleksibel, SSL juga terdapat di mana-mana dan mudah dipelajari, serta bagus untuk kebutuhan keamanan dari jaringan.

IV. REFERENSI

[1] http://www.rtfm.com/sslbook/chap9-sample.pdf
[2[http://www.ketufile.com/What_Is_SSL.pdf
[3]http:// Brad C. Johnson, Jonathan G. Gossels, & Donald T. Davis," The SSL Handshake",http://www.systemexperts.com/assets/tutors/The%20SSL%20Handshake.pdf
[4]http://cisco.com
[5]http://www.ourshop.org/resources/ssl_step2.html
[6]http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp
[7]http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame2.doc_5.1/ss7aumst18.htm?topic=/com.ibm.mq.csqzas.doc/sy10660_.htm
[8] http://www.cs.bgu.ac.il/~beimel/Courses/crypto/SSL-4.pdf
[9] WikströmEdvard," Secure Communication: Is it possible with SSL and or SSH? ",http://www.ida.liu.se/~TDDC03/oldprojects/2004/final-projects/prj022.pdf