VPN
Secure Socket Layer
Transport Layer Security

Mureed HUSSAIN, Ahmed MEHAOUA, Dominique SERET

Technologies VPN

Communication layers	Security protocols

Application layer	ssh, S/MIME, PGP
Transport layer	SSL, TLS, WTLS
Network layer	IPsec
Data Link layer	PPTP, L2TP
Physical layer	Scrambling, Hopping, Quantum Communications

MPLS

Agenda

- Introduction
  - Motivation, evolution, standardization
  - Applications
- SSL Protocol
  - SSL phases and services
  - Sessions and connections
  - SSL protocols and layers
    - SSL Handshake protocol
    - SSL Record protocol / layer
- SSL solutions and products
- Conclusion

Sécurisation des échanges

- Pour sécuriser les échanges ayant lieu sur le réseau Internet, il existe plusieurs approches :
  - niveau applicatif (PGP)
  - niveau réseau (protocole IPsec)
  - niveau physique (boîtiers chiffrant).
- TLS/SSL vise à sécuriser les échanges au niveau de la couche Transport.
- Application typique : sécurisation du Web

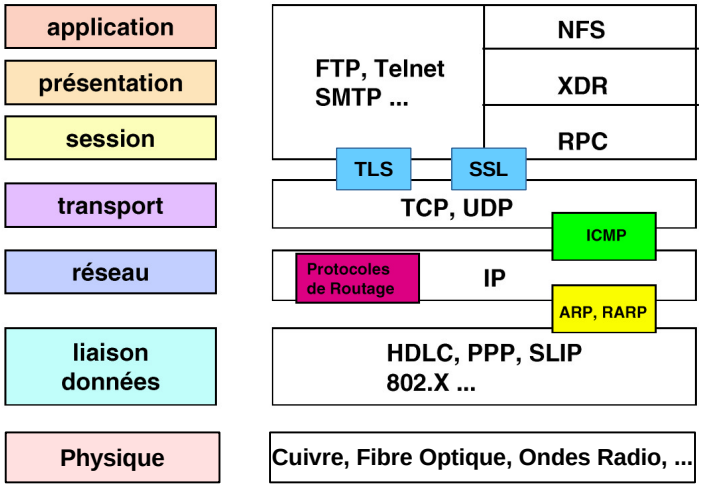
Transport Layer Security

- Advantages
  - Does not require enhancement to each application
  - NAT friendly
  - Firewall Friendly
- Disadvantages
  - Embedded in the application stack (some mis-implementation)
  - Protocol specific --> need to duplicated for each transport protocol
  - Need to maintain context for connection (not currently implemented for UDP)
  - Doesn't protect IP adresses & headers

Security-Sensitive Web Applications

- Online banking
- Online purchases, auctions, payments
- Restricted website access
- Software download
- Web-based Email
- Requirements
  - Authentication: Of server, of client, or (usually) of both
  - Integrity: Of requests, of responses, etc.
  - Confidentiality: Of data transfers
  - Availability: No Denial of Service
- Some minor applications : SSL VPN (end-to-end)
- Main tool: SSL / TLS protocol

IPsec et l'architecture TCP/IP

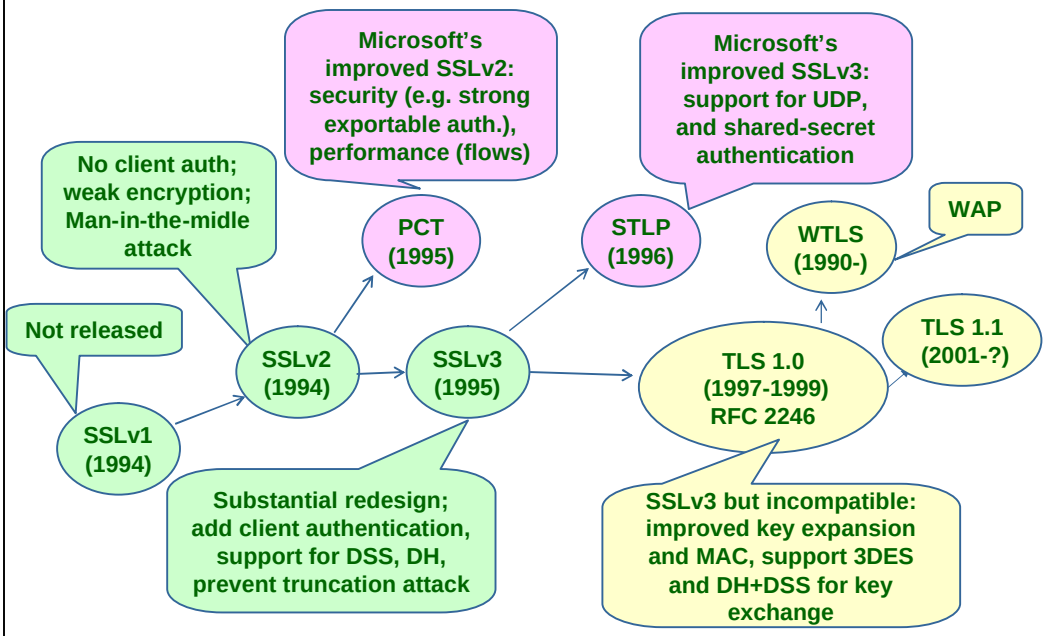


Transport Layer Security Protocols

- Connectionless and connection-oriented transport layer service:
  - Security Protocol 4 (SP4) – NSA, NIST
  - Transport Layer Security (TLSP) – ISO
- Connection-oriented transport layer service:
  - Encrypted Session Manager (ESM) – AT&T Bell Labs.
  - Secure Socket Layer (SSL) – Netscape Communications
  - Transport Layer Security (TLS) – IETF TLS WG

Most popular transport layer security protocols

SSL/TLS Evolution



DSS : Digital Signature Standard (Public-key)
DH : Diffie-Hellman

SSL-TLS

- SSL : Secure Socket Layer (version 3) from Netscape
- TLS is defined in Internet Engineering Task Force (**IETF**) RFC Document 2246, see e.g. at www.ietf.org
- Intermediate security layer between the transport layer and the application layer
- Based on connection-oriented and reliable service (e.g., TCP)
- Able to provide security services for any TCP-based application protocol, e.g., HTTP,FTP, TELNET, POP3, etc.
- Application independent

SSL Services

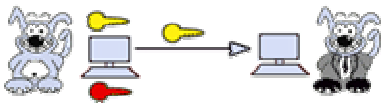
- Server authentication (mandatory)
- Client authentication (optional – if required by the server, and if the client has a certificate)
- Secure connection:
  - Authentication and Integrity: Of messages
  - Confidentiality: Message encryption is optional
  - Reliability: Prevent message re-ordering, truncating, **etc.**
- Efficiency:
  - Allows resuming old SSL sessions in new connections
- Secure negotiation of the cipher-suite

SSL/TLS principles

- Client server
- Provide 4 security services :
  - Authentication of server
  - Confidentiality of exchanged data
  - Integrity of exchanged data
  - Optional : authentication of client (if client has a certificate)
- Combining various security mechanisms :
  - Asymmetric Ciphering : authentication (RSA)
  - Certificate : to validate public key of the server
  - Symmetric Ciphering : Confidentiality of data transmission
  - Hach function : integrity of data

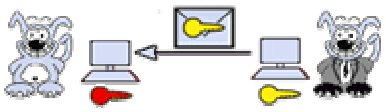
Principe de fonctionnement de SSL: Exemple du paiement en ligne

Sécurisation des transactions par SSL



1 Le client, connecté au site marchand sécurisé par SSL, clique sur un lien hypertexte déclenchant une requête de formulaire sécurisé et la création d'une clé privée que le client va conserver, et d'une clé publique qui sera expédiée au serveur marchand

2 Le serveur marchand crée une clef de session en cryptant un message aléatoire à l'aide de la clef publique, puis l'envoie au client



3 A réception, le client crypte la clef de session à l'aide de la clef privée, puis l'envoie au serveur marchand, qui va la décrypter à l'aide de la clé publique afin de vérifier l'authenticité du message, donc de l'acheteur

4 Le reste des transactions peut alors se faire à l'aide de la clef de session, connue des deux côtés et inconnue des autres entités du réseau

SSL/TLS in a Nutshell

- SSL & TLS provide a `secure TCP tunnel from client to server`:
  - Message Confidentiality
  - Message and connection integrity
  - Authentication of server, optionally also of client
- Implemented in almost all web clients, servers
- Many implementations, libraries, e.g. Open-SSL
- SSL: Secure Socket Layer
  - Version 3 designed by Netscape Corp.
  - Original goal and main use: secure credit card number
  - SSL (& TLS) operate on top of `standard` Sockets API
- TLS: Transport Layer Security
  - Version 1.0 – RFC 2246
  - IETF standard version of SSL
  - We usually say just SSL but refer to both

Chiffrement asymétrique

PB : Tout repose sur la confiance dans la provenance de la clef publique ?

• Si celui qui forge une signature a forgé la clef publique de sa victime ?

• Autrement dit si celui qui souhaite écouter les messages de votre correspondant vous a remis une fausse clef publique pour cette personne ?

Certificat X509 : La solution au problème

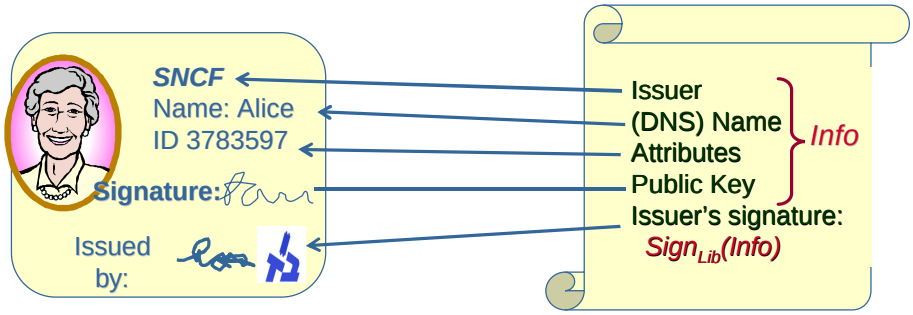
- Solution : une autorité de certification (CA pour Certification Authority) est chargée de signer les clefs publiques : elle chiffre (avec sa clef privée) une empreinte de :
  - L'identité de son titulaire, personne, serveur ou application (*Distinguished Name of Subject*)
  - La clef publique
  - Les informations relatives à l'usage de cette clef, (période de validité, type des opérations possibles, etc).
- L'ensemble est appelé certificat X509. Les certificats X509 font l'objet d'une norme : ITU-T X509 international standard V3 1996, RFC2459

Certificat X509
- Autorité de certification -

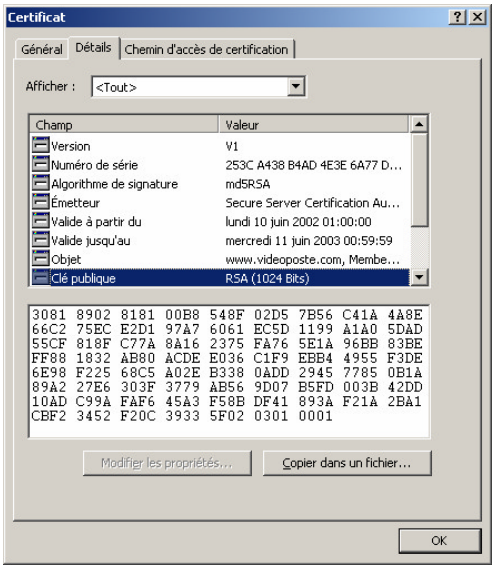
- La CA (Certification Authority) est une organisation qui délivre des certificats à une population.
- Il existe des autorités privées (intranet d'une entreprise), organisationnelles (CRU, CNRS), corporative (notaires), commerciales (Thawte, Verisign, …), très commerciales (microsoft), institutionnelles, etc
- On s'assure de la provenance d'une clef publique en vérifiant la signature qui y a été apposée à l'aide de la clef publique de l'autorité de certification (CA).
- Plus besoin de faire directement confiance à toutes les clefs publiques en circulation mais seulement à celles des autorités de certification.

Public Key Certificates

- Similar to passport or driver's license
- Binds a public key to a name (Alice) and/or other attributes of keyholder, e.g. DNS name for web site
- Signed by a trusted party (Issuer / Certification Authority) (SNCF)
- Allows relying party (Bob, client) to validate name, attributes of key owner (Alice, web site)



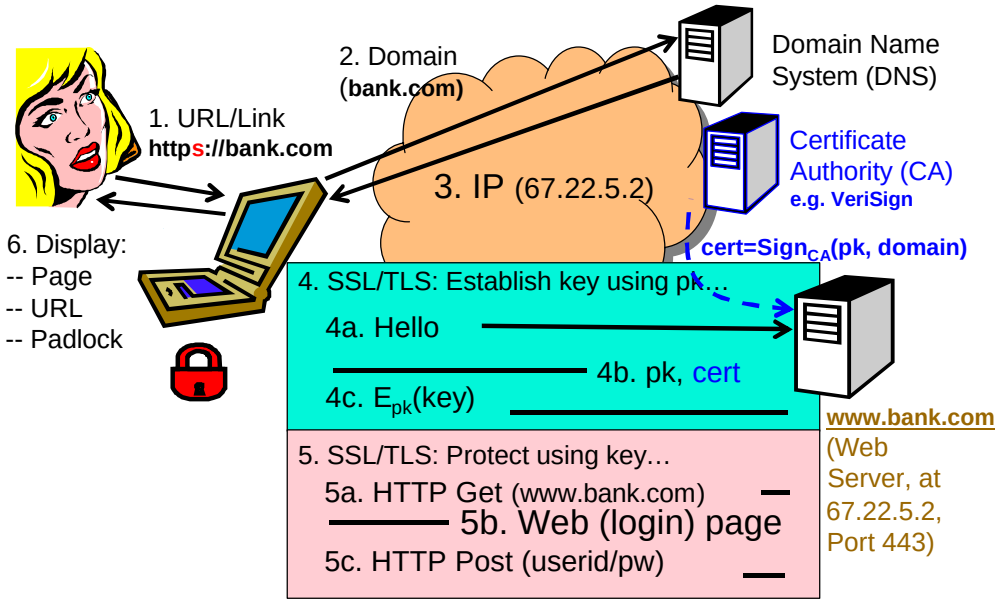
Exemple de Certificat X509



Certificat X509
- Les usages -

- Messagerie S/MIME : signature (certificat de l'émetteur) et/ou chiffrement (certificat du destinataire)
- SSL ou TLS : en particulier HTTPS pour chiffrer les sessions du client et authentifier le serveur. Plus rarement authentifier le client.
- SSL → POPS, IMAPS, LDAPS, SMTP/TLS, …
- VPN avec IPsec

Web Security with SSL/TLS (simplified)



Fonction de hachage cryptographique

- Une fonction de hachage compresse les données fournies en une chaîne de taille constante. Pour une fonction de hachage cryptographique, il est impossible de trouver :
  - Deux entrées qui ont le même résultat (résistant à la collision),
  - Une entrée différente pour une entrée donnée avec le même résultat.

- Ces fonctions sont utilisés pour assurer :
  - l'authentification des systèmes et/ou des messages échangés,
  - L'intégrité des messages échangés
- Les fonctions les plus connues sont : MD5, SHA1, Kerberos, …
- Les applications courantes sont :
  - Dans S/Key (Linux) pour fournir des mots de passe utilisable une seule fois.
  - Dans IPSec et ISAKMPD pour authentifier l'origine des données et assurer l'intégrité du paquet.
  - Dans Linux pour les mots de passe MD5 (non activé par défaut), voir passwd.conf.
  - Dans SSL pour la signature numérique des messages.

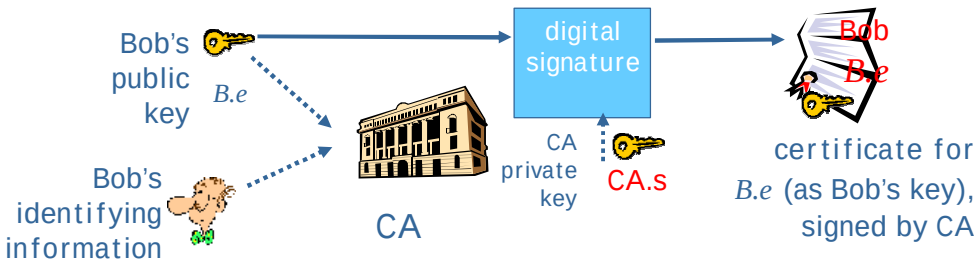
Transformation cryptographique

- Les transformations cryptographiques sont utilisées pour chiffrer et déchiffrer les données. Elles sont normalement utilisées avec une clé de chiffrement qui peut être privé (transformation symétrique) ou une paire de clé publique+privée (transformation asymétrique).

- Ces transformations sont utilisés pour assurer :
  - La confidentialité des messages échangés
- Les transformations les plus connues sont : DES, 3DES, AES, Blowfish, RSA …
- Les applications courantes sont :
  - Dans la libc (Linux) pour créer des mots de passe Blowfish.
  - Dans IPsec pour fournir la confidentialité au niveau de la couche réseau.
  - Dans ISKMPD pour protéger les échanges lorsque des clés IPsec sont négociées.
  - Dans AFS pour protéger les messages transitant sur le réseau, fournissant la confidentialité lors de l'accès au système de fichiers distant.
  - Dans SSL pour permettre aux applications de communiquer à l'aide du protocole cryptographique SSL (qui est un standard de fait).

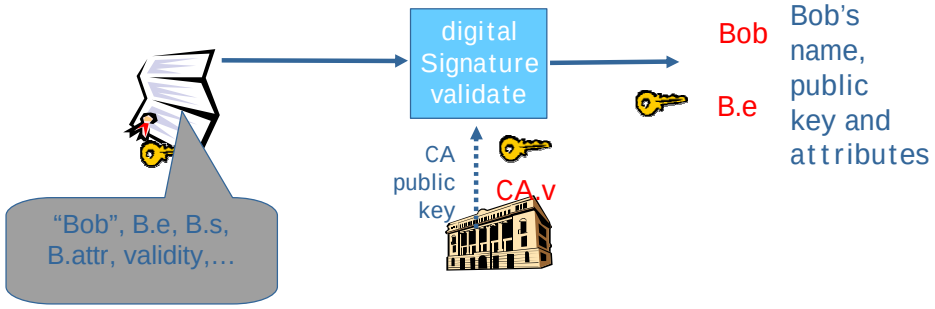
Certification Authorities

- Certification authority (CA): binds public key (e.g. $B.v$, $B.e$) to identifier (e.g. $B.name = \text{`}Bob\text{`}$).
- Bob (person, server) registers $B.v$, $B.e$ with CA.
  - Bob convinces the CA his name is Bob, sends $B.v$, $B.e$.
  - CA creates certificate binding "Bob" to $B.v$, $B.e$.
  - Certificate is digitally signed by CA – CA says "$B.e$ is Bob's public encryption key"



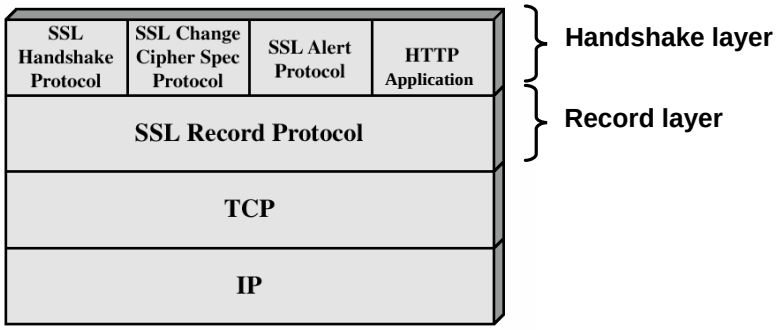
Using Public Key Certificates

- When Alice wants Bob's public key (to encrypt message to Bob or validate Bob's signature):
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA.v (public validation key) to Bob's certificate, get Bob's identity, public keys, …

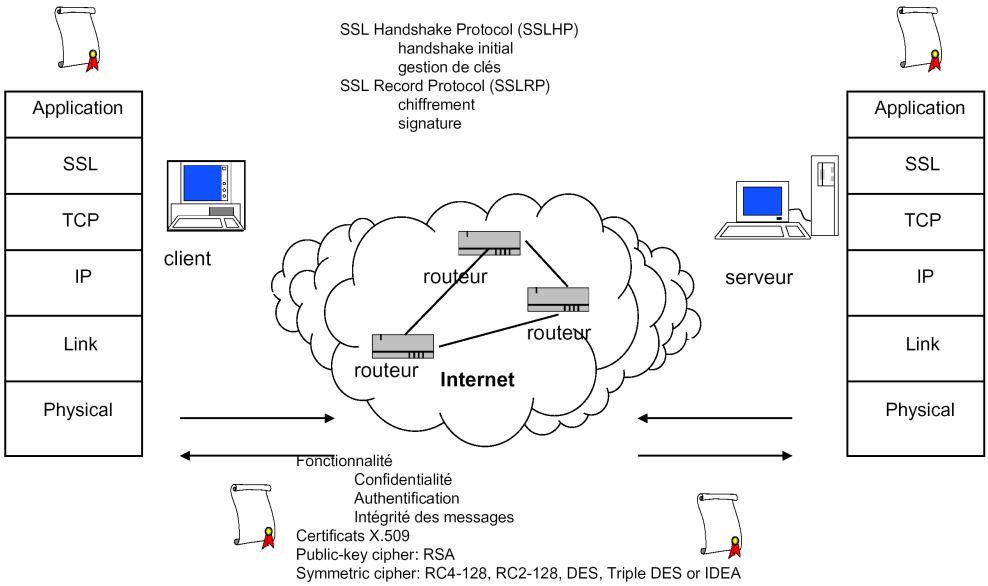


SSL Architecture

- SSL is built in two layers:
  - **SSL Handshake Layer** – used for managing SSL exchanges (cipher suite negotiation, session key generation, etc.)
  - **SSL Record Layer** – used to secure communication between client and server with the established session keys



SSL : Secure Socket Layer



SSL Main Protocols

- SSL Record Protocol
  - Layered on top of a connection-oriented and reliable transport layer service
  - Provides message origin authentication, data confidentiality, and data integrity
- Handshake Protocol
  - Used to mutually authenticate client and server and exchange session key

SSL Sub-Protocols

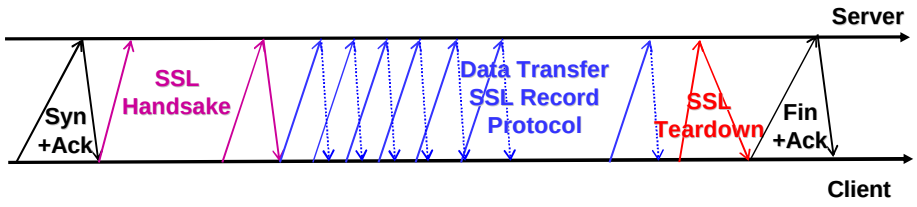
- Layered on top of the SSL Record Protocol
- Provides support for SSL session and connection establishment

- Alert Protocol
  - Used to transmit alerts via SSL Record Protocol
  - Alert message: (alert level, alert description)
- ChangeCipherSpec Protocol
  - Used to change cipher specifications
  - Can be changed at the end of the handshake or later
- Application Protocol
  - Used to directly pass application data to the SSL Record Protocol

SSL Operation Phases

Client uses SSL API to open connection

- TCP Connection Phase
- Handshake Phase (SSL Handshake Protocol)
  - Negotiate (agree on) algorithms, methods
  - Authenticate server and optionally client
  - Establish keys
  - Establish connection (keys and optionally Initialization Vector)
- Data transfer Phase (SSL Record Protocol)
- SSL Secure Teardown Phase



SSL State Information

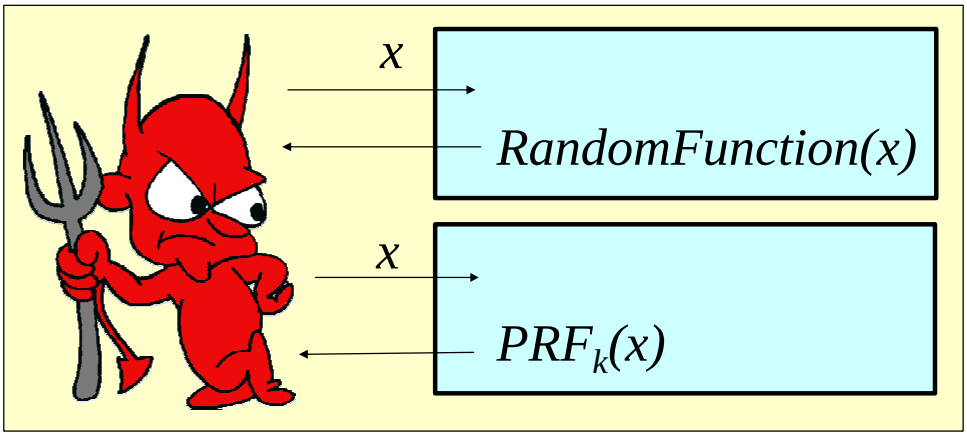
- SSL session is stateful : SSL protocol must initialize and maintain session state information on either side of the session
- SSL run over TCP (not suitable for UDP)
- SSL session can be used for several connections : Connection state information

SSL Connection State Variables

- Master Secret (shared key)
  - Unique to each connection
- Server and client sequence numbers
- *Server_random*, *client_random*: 32 bytes
  - Unique to each connection, selected by server and client
- Cryptographic keys, Initialization Vectors (IV)
  - Derived from Master Secret using a Pseudo-Random Function (PRF)
  - What's a PRF and how we use it?

A Pseudo-Random Function

- An efficient function using secret key
  - TLS's PRF is based on MD5 and SHA-1 (later…)
- That cannot be distinguished from random



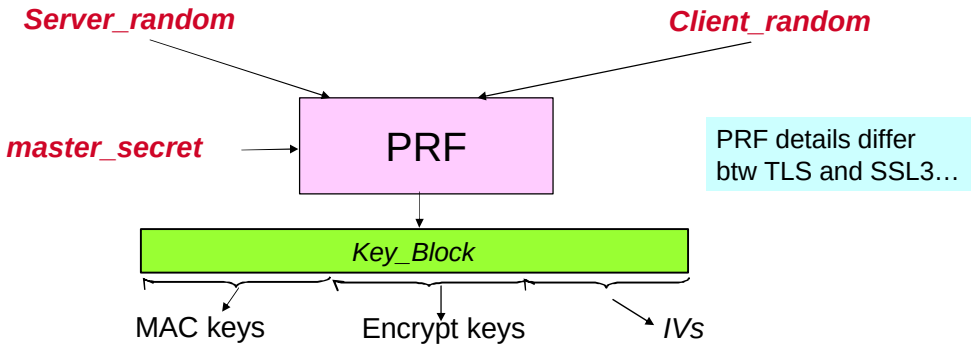
SSL Key Derivation

- The master key is used to derive the following six keys and values:
  - Client MAC key
  - Server MAC key
  - Client encryption key
  - Server encryption key
  - Client Init Vector (for CBC encryption)
  - Server Init Vector (for CBC encryption)
- Separate client and server keys are used
  - So successful attack against server does not compromise client, and vice versa

Deriving Connection Keys, IVs

Key_Block = PRF $_{master\_secret}$ *(“key expansion”|| Server_random _ Client_random)*

Split *Key_Block* to *ClientMACKey, serverMACKey, ClientEncryptKey,…*(using fixed order)



SSL Session State Information Elements

- Session ID: chosen by the server to identify an active or resumable session state
- Peer certificate: certificate for peer entity (X.509 v. 3)
- Compression method: algorithm to compress data before encryption
- Cipher spec: specification of data encryption and Message Authentication Code (MAC) algorithms
- Master secret: 48-byte secret shared between client and server
- Is resumable: flag that indicates whether the session can be used to initiate new connections

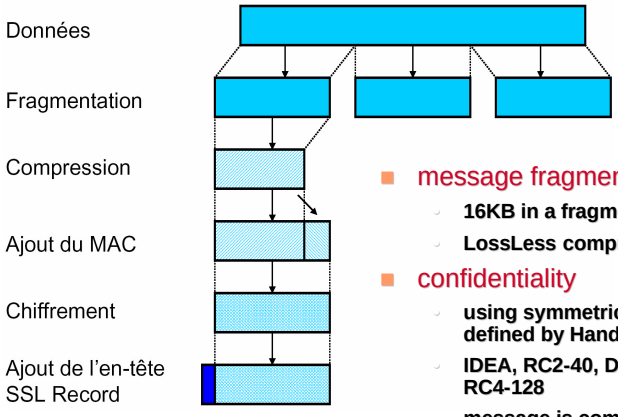
SSL Session State Information Elements (suite)

- Server and client random: byte sequences that are chosen by server and client for each connection
- Server write MAC secret: secret used for MAC on data written by server
- Client write MAC secret: secret used for MAC on data written by client
- Server write key: key used for data encryption by server and decryption by client
- Client write key: key used for encryption by client and decryption by server
- Initialization vector: for CBC block ciphers
- Sequence number: for both transmitted and received messages, maintained by each party

SSL Record Protocol

- Ce protocole fournit 2 services à une connexion SSL :
  - Confidentialité : définit une clé secrète pour le chiffrement
  - Intégrité du message : définit une clé secrète pour le calcul de l'empreinte
- SSL Record Protocol : opérations
  - Fragmentation :
    - Le message est fragmenté en blocs de taille maximum 2^{14} octets
  - Compression :
    - Cette opération est prévue dans les spécifications mais non implémentée
  - Calcul du MAC :
    - Utilise la clé secrète
    - Utilise l'algorithme SHA-1 ou MD5
  - Chiffrement :
    - Le message + MAC sont chiffrés avec un chiffrement symétrique
  - Ajout de l'en-tête :
    - 5 octets, composée de longueur du message, version, etc.

SSL Record Protocol (suite)



- message fragmentation and compression
  - **16KB in a fragment**
  - **LossLess compression (1 KB is OK)**
- confidentiality
  - **using symmetric encryption with a shared secret key defined by Handshake Protocol**
  - **IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128**
  - **message is compressed before encryption (optional)**
- message integrity
  - **using a MAC with shared secret key**
  - **Using SHA-1 or MD5**
- Non replay
  - **Use counter (64 bits) to prevent replay in SSL session**

SSL Alert Protocol

- SSL Alert Protocol signals state changes and indicates errors
- SSL Alert is invoked by:
  - Handshake protocol – in case of problem
  - Record protocol – e.g. if MAC is not valid
  - Application – to close connection (close_notify)
  - Connections should be closed with close_notify to allow detection of truncation attacks (dropping last messages)
  - **Notice:** close_notify is normal, not a failure alert!
- The alerts are carried in an “Alert Record”

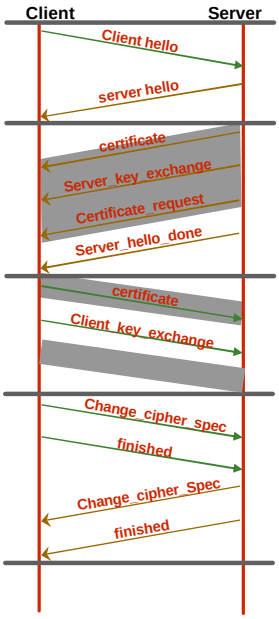
SSL Alert Protocol (2)

- Use two-byte message to convey SSL-related alerts to peer entity
  - First byte is severity level
    - warning(1) or fatal(2)
  - Second byte is specific alert
    - Always fatal: unexpected_message, bad_record_mac, decompression_failure, handshake_failure, illegal_parameter
    - Other alerts: close_notify, no_certificate, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown
- Compressed and encrypted like all SSL data

SSL Handshake Protocol

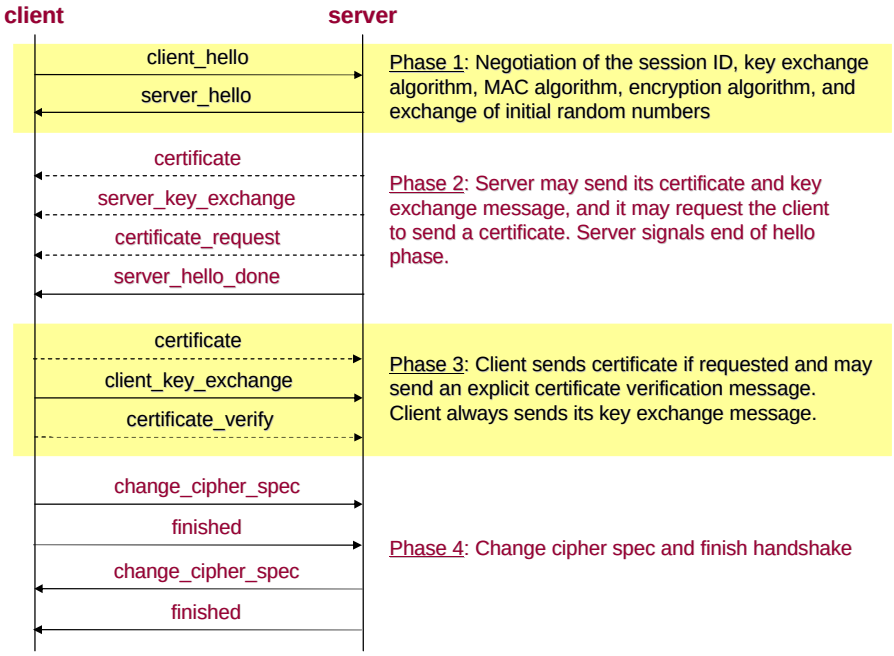
- Allow server and client to agree on cipher suite (algorithms and options):
  - authenticate server (madatory)
  - Authenticate client (option)
  - negotiate encryption algorithms (symmetric or asymmetric)
  - negotiate Signature & MAC algorithms
  - negotiate cryptographic keys to be used
  - Send certificate(s)
  - SSL Message compression
- Comprise a series of messages in 4 phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

SSL Handshake Protocol Phases

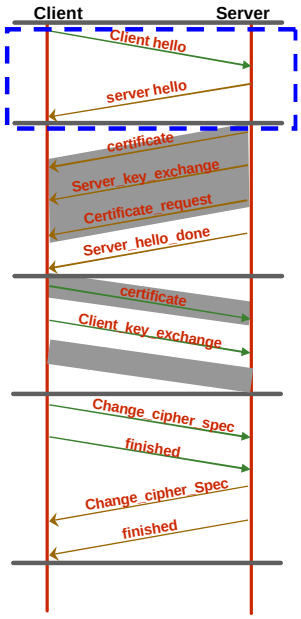


- Phase 1 – Establish parameters
- Phase 2 – Server authentication (optional: server key-exchange)
- Phase 3 – Client key-exchange (optional: client authentication)
- Phase 4 – Finish: validation and begin using exchanged keys

SSL Handshake Protocol Phases

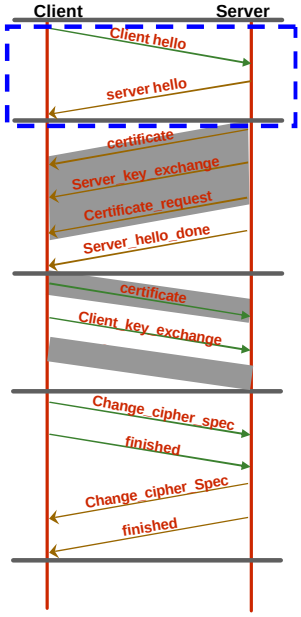


Phase 1 – Establish Parameters



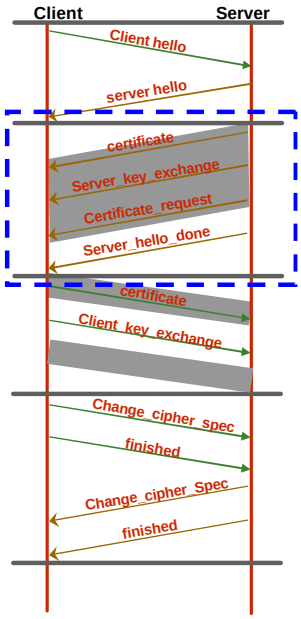
- `Client_hello` message
  - Version (highest available)
  - A timestamp and random string called `clientHello.random`
  - Session identifier (used for existing connections)
  - Cipher-suite – a list of supported cryptographic algorithms
  - Compression method
  - `Client_random, server_random`
  - From TLS 1.1: Extensions
- `server_hello` : same fields

Phase 1 – Establish Parameters (suite)



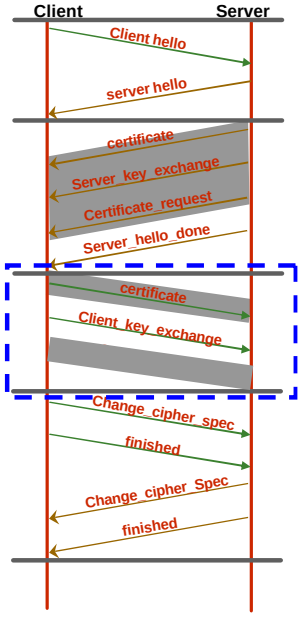
- Cipher-suite elements:
  - Key-exchange method (e.g. RSA)
  - Encryption algorithm (e.g. DES or RC4)
  - MAC (message authentication code) algorithm
- Server chooses one cipher-suite from those sent by client

Phase 2 – Server Authentication



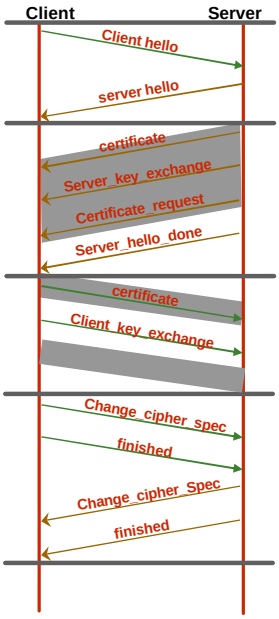
- Phase 2 – Server Authentication and (optional) Server Key-Exchange
- Server sends its **`certificate`**
  - It sends one or a chain of X.509 certificates
- Optional **`Server_key_exchange`** message
  - Used in Diffie-Hellman key exchange
  - Not used in RSA key exchange
- Optional **`certificate_request`** message: for client authentication
- **`server_done`**

Phase 3 – Client Key-Exchange



- Phase 3 – Client Key-Exchange and (optional) Client Authentication
- The client verifies the server's certificate and sends its side of the key exchange
  - In Diffie-Hellman: the D-H key share
  - In RSA: encryption of random string
- If client authentication used (rarely): client sends certificate (but most clients don't have certificates)

Phase 4 – Finish



- Client and server send **`change_cipher_spec`** messages
  - Results in use of new cipher-suite (as negotiated in the hello phase)
- Client and server send **`finished`** messages
  - Messages are HMAC on all the handshake messages using **`master_secret`** as the key
  - The MAC is computed twice – once with MD5 and once with SHA1

Summary: Trust & Security with SSL

- Confidentiality & authenticity of messages

} Good in SSL/TLS

- Server (site) authentication:
  - Customer needs to identify site (bank, etc.)
- Client authentication:
  - Bank needs to identify account holder
  - Company needs to identify employee
  - Content provider needs to identify subscriber

} Done, but…

- Non-repudiation:
  - Proof of making/receiving order/transaction
  - Prevent/resolve dispute, identify corruption
- Denial of service

} Not in SSL

SSL/TLS : Applications and ports

Protocol	Normal Port	Secured Protocol	Encrypted Port	RFC
HTTP	80	**HTTPS**	443	2818
NTP	119		563	
LDAP	389	**SLDAP**	636	2830
FTP-DATA	20	**SFTP**	989	draft
FTP-control	21	**SFTP**	990	draft
TELNET	23	**SSH**	992	4251
IMAP	143	**SIMAP**	993	2595
IRC	194		994	2813
POP3	110	**SPOP3**	995	2595
SMTP	25	**SMTP-TLS**	465 (25)	2487

SSL-based current solutions

- OpenSSL : www.openssl.org
  - a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols
- Win32 SSL API
- JavaSSL : www.bpsinfo.com/javassl/
  - A Java package which uses JNI to provide an SSLSocket class
- Apache-SSL : http://www.apache-ssl.org/
- OpenSSH : www.openssh.com/
  A port of the OpenSSL-based SSH package
- Mocana Security Suite : www.mocana.com
  - Software suite including (SSL, SSH, IPsec, IKE, Radius, …)
- sNFS : www.quick.com.au/ftp/pub/sjg/help/sNFS.html
  - An SSL-based NFS variant
- OpenVPN : www.openvpn.org

SMTP over TLS : Solutions

Serveurs SMTP avec TTLS	Contrôle du relais par certificats client	Authentification des serveurs par certificat	Remarques
Exim 3.20	OUI	OUI	
Infinite Interchange V3.51	?	?	
Inframail Advantage Server	NON	OUI	
Innosoft PMDF/TSL 5.2-31	OUI ?	OUI	
Merak	NON	NON	
Microsoft Exchange 2000	?	OUI	
Microsoft Exchange 5.5 SP3	?	NON	
Netscape Messaging Server 4.15	OUI	?	
Postfix	OUI	OUI	Gratuit
Qmail	OUI	OUI	Gratuit
Sendmail 8.11.0 et +	OUI	OUI	Gratuit
Software.com Intermail	?	?	
Stalker Communigate Pro 3.2 et +	NON	NON	
ZMailer 2.99.51 et +	NON	NON	Gratuit

SSL Applications

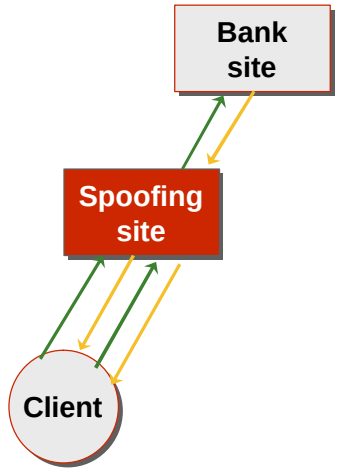
- SSH : Secure Shell
  - RFC
- SFTP : Secure FTP (990:cmd; 989:data)
- HTTPS : HTTP Secured (443) : different from S-HTTP
  - RFC 2818
- SMTP-TLS
  - RFC 2487

Conclusion

- SSL / TLS is the most widely deployed security protocol, standard
  - Easy to implement, deploy and use; widely available
  - Flexible, supports many scenarios and policies
  - Mature cryptographic design
- But SSL is not always the best tool…
  - Use IP-Sec e.g. for anti-clogging, broader protection, multicast
  - Use application security, e.g. s/mime, for non-repudiation, store-and-forward communication (not online)
- Beware of spoofing
  - Many browsers allow hard-to-detect spoofing
  - Many users will not detect simple spoofing (similar URL)

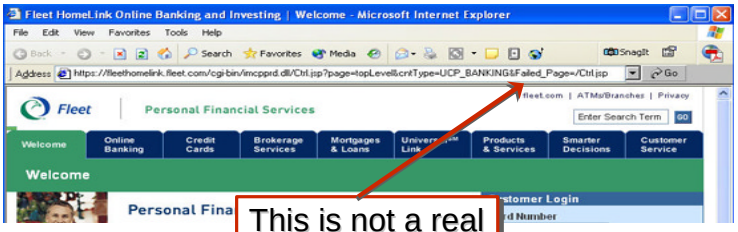
Web Spoofing Attack

- **Web spoofing attack:**
  - Copy & modify target website
  - User visits the spoofing site
  - User exposes password, credit card numbers, etc.
- User is not aware
  - Spoofing site can forward information to the target, to avoid detection
  - Detect incorrect location (URL)?
  - Most users do not notice
  - Or – spoof the location bar too…



Spoofed Location Bar (1)

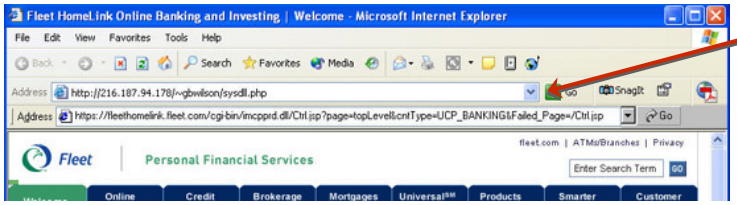
- Spoofed web sites – avoid detection via the location indicator (URL)
  - Using a spoofed location bar:



This is not a real address bar, but a picture. The address bar is not displayed

Spoofed Location Bar (2)

- Spoofed web sites – avoid detection via the location indicator (URL)
  - Using a spoofed location bar:



This is the real address bar