



Security protocols: SSL/TLS

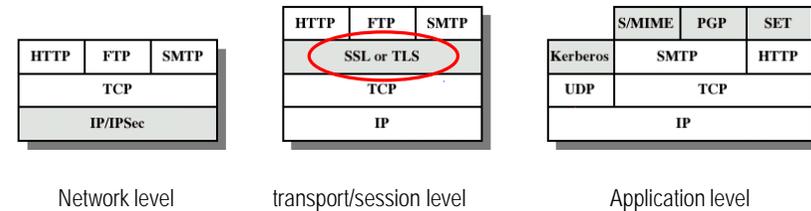
Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Sicurezza nelle reti di telecomunicazioni, a.a. 2006/2007

<http://www.tlc.unipr.it/veltri>

Location of security facilities in the TCP/IP stack



SSL/TLS

- Le applicazioni più diffuse del protocollo sono:
 - **commercio elettronico (pagamenti on-line)**
 - **transazioni finanziarie**
 - **invio di dati personali**
- Si differenzia da IPsec per:
 - **livello ISO/OSI**
 - **orientato ad applicazioni client-server**
 - **possibilità di configurazione/protezione**

Secure Socket Layer (SSL)

- Protocollo per applicazioni client-server che garantisce la privacy delle comunicazioni su Internet (su rete IP)
- E' in grado di prevenire le intrusioni, le manomissioni e le falsificazioni dei messaggi
- Protocollo aperto e non proprietario proposto da Netscape Communication Corporation
- Utilizza TCP come protocollo di trasporto affidabile end-to-end
- Individuato da URL del tipo `https://...` (port 443)
- Supporta qualsiasi applicazione (HTTP, FTP, ...)
- Utilizza *server certificates*
- Supporta anche *client certificates* (optional)

Transport Layer Security (TLS)

- Transport Layer Security (TLS)
- IETF standard RFC 2246
- TLS v1.0 == SSL v3.1
 - **SSL was maintained by Netscape**
 - **TLS is an IETF standard**
- minor differences
 - **in record format version number**
 - **uses HMAC for MAC**
 - **a pseudo-random function expands secrets**
 - **has additional alert codes**
 - **some changes in supported ciphers**
 - **changes in certificate negotiations**
 - **changes in use of padding**

5

SSL/TLS security goals

- **Riservatezza:**
Dopo un handshake iniziale viene definita una chiave segreta (scambiata cifrandola con la chiave pubblica del server)
Per crittografare i dati è usata crittografia simmetrica (DES, RC4)
- **Autenticazione:**
l'identità nelle connessioni può essere autenticata usando la crittografia asimmetrica o a chiave pubblica (RSA, DSS) (certificato X.509v3).
 - **É prevista la certificazione sia del server che del client (opzionale).**
- **Integrità:**
il livello di trasporto include un check dell'integrità del messaggio basato su un apposito MAC (Message Authentication Code) che utilizza funzioni hash (SHA, MD5)

6

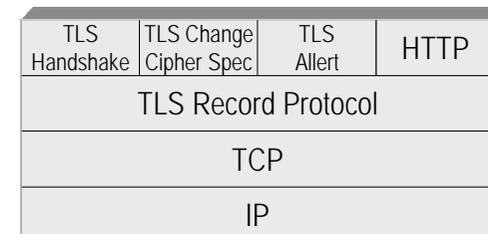
SSL/TLS phases

- Handshake
 - **Establish connection**
 - **Agree on encryption algorithm**
 - **Exchange key**
 - **Authentication**
 - Server only or both client and server
 - Authentication with certificates
- Securing messages
 - **Sending the actual encrypted messages**
 - **Integrity checks with MACs**

7

SSL/TLS protocol stack

- Il protocollo è composto da due strati:
 - **a livello più basso c'è il protocollo SSL Record che è usato per l'incapsulamento dei vari protocolli di livello superiore**
 - **sul protocollo SSL Record si interfaccia l'SSL Handshake Protocol che permette al server ed al client di autenticarsi a vicenda e di negoziare un algoritmo di crittografia e le relative chiavi**



8

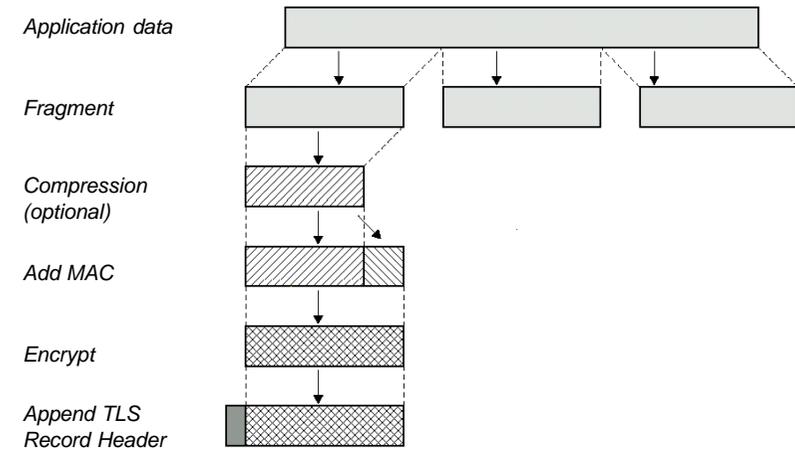
TLS Record Protocol

- Consente di incapsulare dati e informazioni di autenticazione
- I messaggi SSL/TLS sono accorpati in record lunghi fino a 32.767 byte
- Permette di garantire:
 - **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
 - **message integrity**
 - using a MAC with shared secret key
 - similar to HMAC but with different padding
- Each message is appended with a MAC before it is encrypted
- The key for encryption, the key for the MAC and the Initialization Vector (if used) is extracted from the key exchange messages (Handshake Protocol)

9

TLS Record Protocol

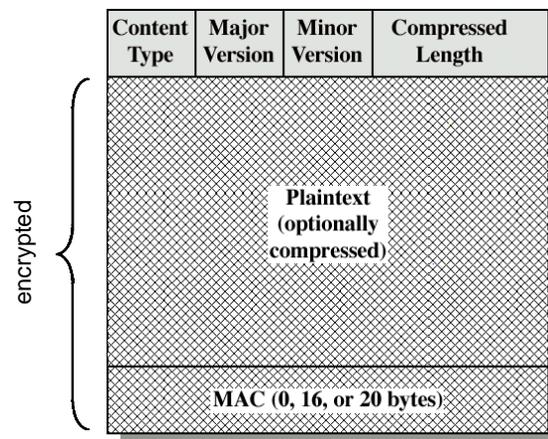
- TLS-RP operation:



10

TLS Record Protocol

- TLS-RP format



11

TLS Change Cipher Spec Protocol

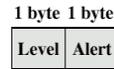
- One of 3 SSL/TLS specific protocols which use the TLS Record protocol
- A single 1 byte message (value=1) 1
- Causes pending state to become current
- Hence updating the cipher suite in use

12

TLS Alert Protocol

- Conveys TLS-related alerts to peer entity

- A single 2 bytes message



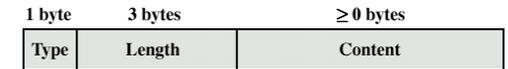
- Specific alert
 - fatal (first byte=2)
 - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
 - warning (first byte=1)
 - close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown

- Compressed & encrypted like all TLS data

13

TLS Handshake Protocol

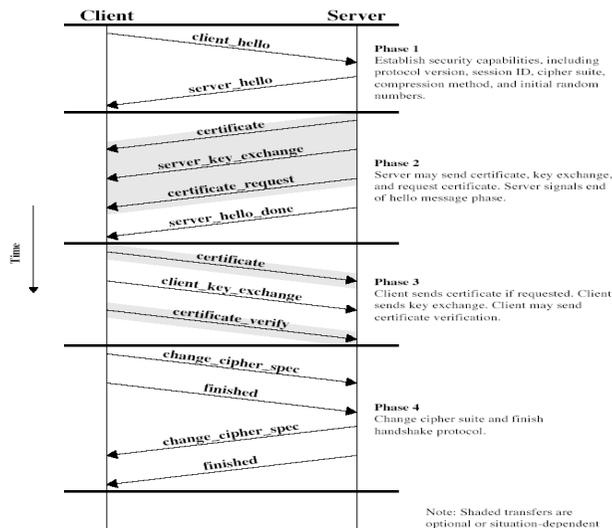
- Allows the server and client to
 - authenticate each other
 - negotiate encryption & MAC algorithms before the application protocol transmits or receives its first byte of data
 - negotiate cryptographic keys to be used
- Comprises a series of messages in phases
 - Establish Security Capabilities
 - Server Authentication and Key Exchange
 - Client Authentication and Key Exchange
 - Finish



- Uses asymmetric cryptography (e.g. RSA) for peer's identity authentication
 - This authentication can be made optional, but is generally required for at least one of the peers

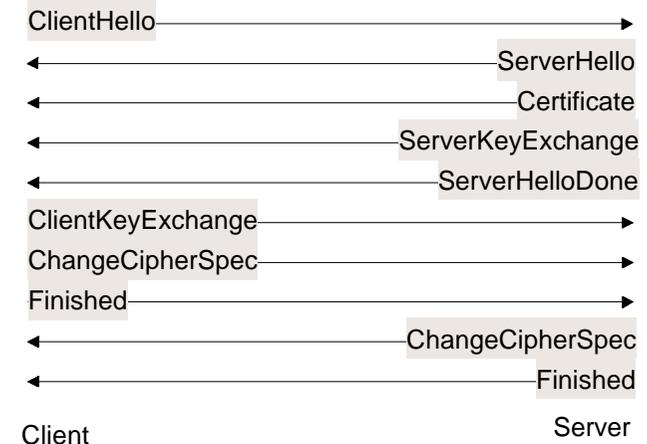
14

TLS Handshake Protocol



15

TLS Handshake



16

ClientHello

- The client initiates the communication by sending the ClientHello message
- The message contains
 - **version number**
 - **optional session ID**
 - used to resume a previous session
 - **list of cipher suites supported**
 - The cipher suite includes key exchange algorithm, symmetric algorithm (including chaining mode) and MAC algorithm

17

ServerHello

- In response to the ClientHello message, the server sends a ServerHello message
- In this message, the server finally decides which cipher suite to use
- The ServerHello message contains
 - **version number**
 - **optional session ID**
 - included if the server allows the client to resume a previous session
 - **the cipher suite to be used, picked from the list of proposals given by the client**

18

Certificate

- The Certificate message contains the server certificate, including the chain leading up to the CA root certificate
 - **Optional according to the TLS specifications, but most (all?) implementations require a server certificate**
 - **If no certificate is sent, the ServerKeyExchange is required**

19

ServerKeyExchange

- The ServerKeyExchange message is used for the key exchange
 - **Includes the server part of the key exchange**
 - **Exact meaning depends on the cipher suite chosen**
 - For RSA, the server's public key is sent
 - For Diffie-Hellman, the modulus p , the generator g and $x = g^a$ is sent
 - **Necessary if no public key is sent in the certificate**
 - If the information in the certificate can be used for signing, the key information is signed

20

ServerHelloDone

- The ServerHelloDone marks the end of the server's part in the handshake. It does not contain any other information

21

ClientKeyExchange

- The ClientKeyExchange message contains the client part in the key agreement
- The exact format depends on the exchange algorithm agreed on previously
 - **For Diffie-Hellman, the message contains $y = g^b$, the client's part in the agreement. From this the symmetric key is extracted**

22

ChangeCipherSpec and Finished

- Indicates that from this point, communication is encrypted
- The Finished message, itself encrypted, marks the end of the handshake

23

ChangeCipherSpec and Finished

- The server's ChangeCipherSpec and Finished messages play the same role as the client's message

24

Next phase, secure messaging

- After the handshake is complete, the client and the server start exchanging encrypted messages

25

Generating secretes

- Key exchange methods establish a "Pre Master Secret" (PMSK) between the client and server
- For all key exchange methods, the same algorithm is used to convert the PMSK into the "Master Secret" (MSK) [48B]
 - the PMSK should be deleted from memory once the MSK has been computed

MSK = PRF(PMSK, "master secret", ClientHello.random + ServerHello.random)

with

PRF(secret,label,seed) = PRF(S1||S2,label,seed) =

= P_MD5(S1,label+seed-i) ⊕ P_SHA-1(S2,label+seed-i)

and

seed-i = HMAC_hash(secret, seed-(i-1))

- note: S1 and S2 are the two halves of the MSK

26

Generating secretes (cont.)

- MSK is used to generate secret material such as
 - client write MAC secret,
 - a server write MAC secret,
 - a client write key,
 - a server write key,
 - a client write IV, and
 - a server write IV

- The master secret is hashed into a sequence of secure bytes

key_block = PRF(MSK, "key expansion", server_random+client_random)

- assigned to the MAC secrets, keys, IVs...

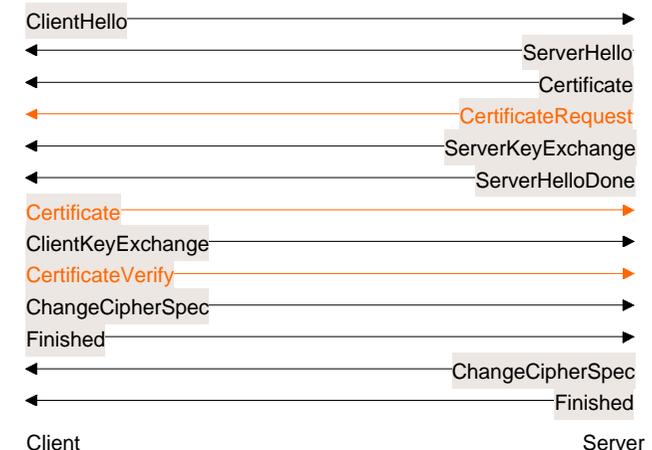
- Final keys:

final_c_w_key = PRF(c_w_key, "client write key", c_rand+s_rand)

final_s_w_key = PRF(s_w_key, "server write key", c_rand+s_rand)

27

TLS Handshake with client authentication



28

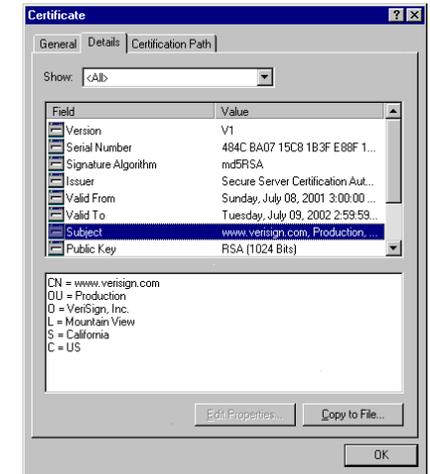
Verifying the certificate

- All certificates in TLS are in the X.509 format
- To verify that a certificate is valid, the verifier must
 - Check that the CA signature is valid
 - Check that the owner of the certificate knows the private key
 - Check that the identifying information is what it should be
- The protocol specifies how to perform the first two parts, but the last part is up to the implementation

29

Certificate contents

- This picture shows how Internet Explorer shows the contents of a certificate.
- Note that the CN field contains the host name of the server.



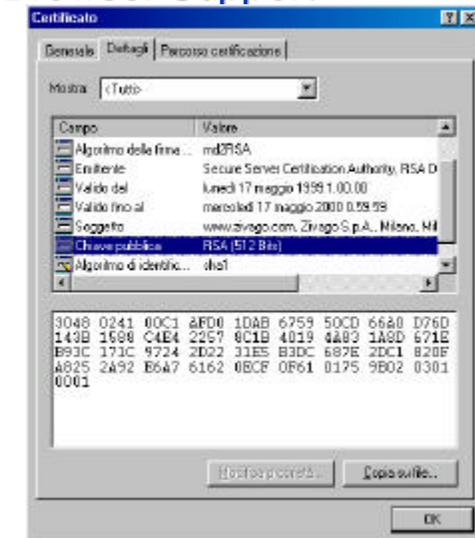
30

Web Browser Support

- Some Root/CA certificates pre-installed
 - IE and Navigator have different lists
 - CA's public key is used to verify the signatures on issued certificates
 - Browsers can accept unverifiable certificates
- Users can install additional certificates
 - Additional Root/CA certificates
 - Security vulnerability
 - Client certificates

31

Web Browser Support



32