UNIVERSITA' DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Security protocols: TLS

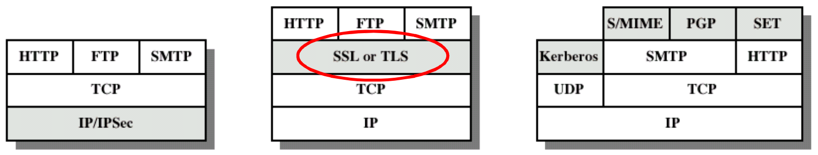
Luca Veltri

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

Corso di Sicurezza nelle reti, a.a. 2009/2010

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

Location of security facilities in the TCP/IP stack



HTTP | FTP | SMTP
TCP
IP/IPSec

HTTP | FTP | SMTP
SSL or TLS
TCP
IP

S/MIME | PGP | SET
Kerberos | SMTP | HTTP
UDP | TCP
IP

Network level

transport/session level

Application level

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 5246 TLSv1.2 (2008)
- 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**

TLS security goals

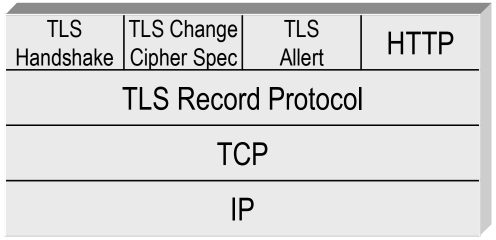
- Confidenzialità:
  Dopo un handshake iniziale viene definita una chiave segreta (scambiata cifrandola con la chiave pubblica del server)
  Per crittografare i dati è usata crittografia simmetrica (DES, RC4)
- Autenticazione:
  l'identità nelle connessioni può essere autenticata usando la crittografia 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)

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 enchrypted messages**
  - **Integrity checks with MACs**

TLS protocol stack

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



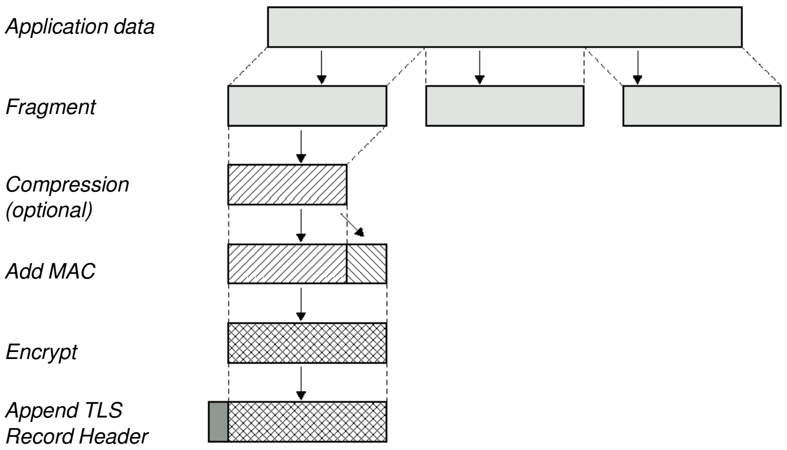
TLS Handshake	TLS Change Cipher Spec	TLS Allert	HTTP
TLS Record Protocol			
TCP			
IP			

TLS Record Protocol

- Consente di incapsulare dati e informazioni di autenticazione
- I messaggi 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, AES
    - message is compressed before encryption
  - **message integrity**
    - using a MAC with shared secret key (by default it uses HMAC)
- 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)

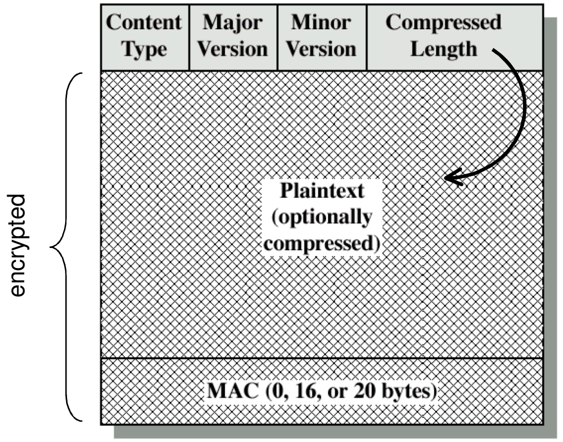
TLS Record Protocol

- TLS-RP operation:



TLS Record Protocol

- TLS-RP format



TLS Change Cipher Spec Protocol

- E' uno dei 3 possibili 3 sottoprotocolli di TLS incapsulati dentro TLS Record protocol

- Consiste in un messaggio di un solo byte (value=1)

1 byte
1

- Trasforma il "pending state" appena negoziato in stato corrente
  - **permette di aggiornare la suite di cifratura da usarsi per i dati che seguono**

TLS Alert Protocol

- Permette di scambiare massaggi di "alert" tra gli estremi del TLS

- Usa un messaggio di 2 byte

1 byte	1 byte
Level	Alert

- 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

- Viene compresso/criptato come ogni altro messaggio TLS

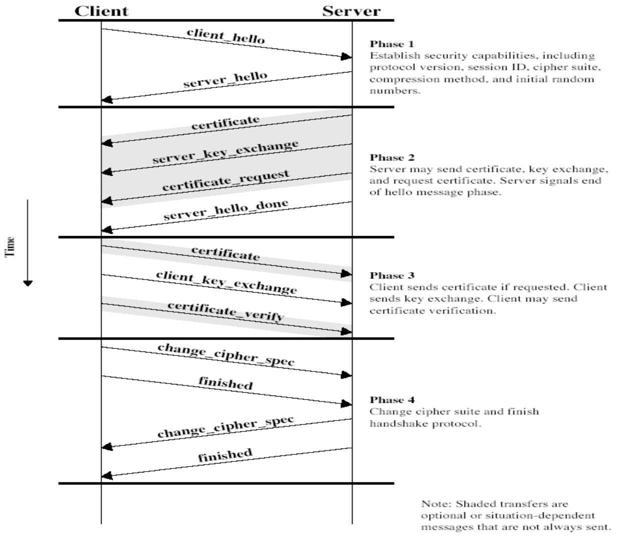
TLS Handshake Protocol

- Permette al TLS client e server di:
  - **autenticarsi tra loro**
  - **negoziare gli algoritmi di cifratura e di autenticazione (MAC) prima di incominciare a far transitare i dati**
  - **negoziare le chiavi di cifratura opportune**
- Utilizza i seguenti tipi di messaggi:
  - **Establish Security Capabilities**
  - **Server Authentication and Key Exchange**
  - **Client Authentication and Key Exchange**
  - **Finish**

1 byte	3 bytes	≥ 0 bytes
Type	Length	Content

- Utilizza crittografia asimmetrica (e.g. RSA) per l'autenticazione delle parti
  - **è opzionale ma normalmente si applica almeno tra client (autenticatore) e server (autenticato)**

TLS Handshake Protocol



Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

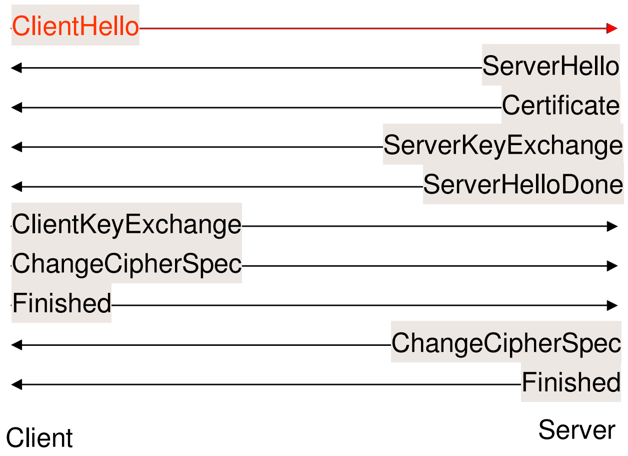
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

TLS Handshake Protocol (cont.)



TLS Handshake Protocol (cont.)

- ClientHello
  - **the communication by sending the ClientHello 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

TLS Handshake Protocol (cont.)

- ServerHello
  - **sent in response to the ClientHello message**
  - **with this message, the server finally decides which cipher suite to use**
  - **contains**
    - version number
    - optional session ID (for resuming a previous session)
    - the cipher suite to be used, picked from the list of proposals given by the client
- Certificate
  - **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

TLS Handshake Protocol (cont.)

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

TLS Handshake Protocol (cont.)

- ClientKeyExchange
  - **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
- ChangeCipherSpec
  - **indicates that from this point, communication is encrypted**
- Finished
  - **is encrypted**
  - **marks the end of the handshake**

TLS Handshake Protocol (cont.)

- ChangeCipherSpec and Finished
  - **play the same role as the client's message**

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

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

Generating secretes (cont.)

- MSK is used to generates secrete 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)

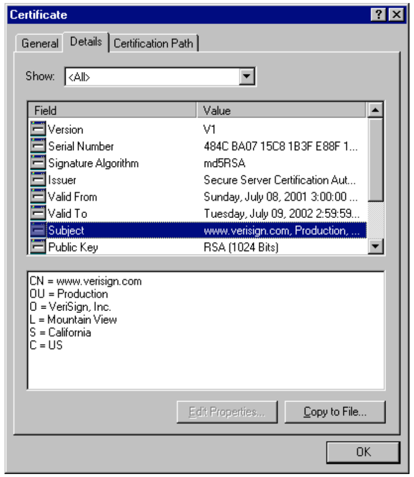
final_s_w_key = PRF(s_w_key,"server write key",c_rand+s_rand)

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

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.



Show: <All>

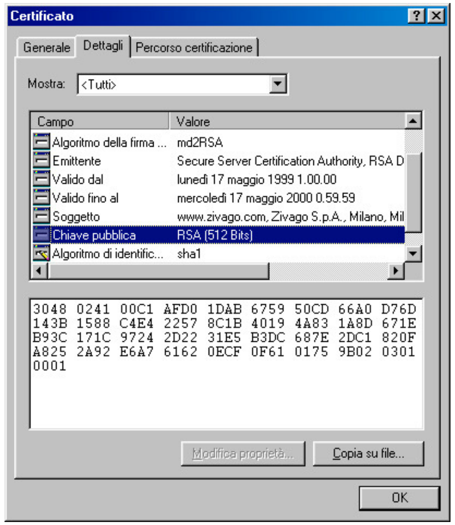
Field	Value
Version	V1
Serial Number	484C BA07 15C8 1B3F E88F 1...
Signature Algorithm	md5RSA
Issuer	Secure Server Certification Aut...
Valid From	Sunday, July 08, 2001 3:00:00 ...
Valid To	Tuesday, July 09, 2002 2:59:59...
Subject	www.verisign.com, Production, ...
Public Key	RSA (1024 Bits)

CN = www.verisign.com
OU = Production
O = VeriSign, Inc.
L = Mountain View
S = California
C = US

Web Browser Support

- Some Root/CA certificates pre-installed
  - **Firefox Opera and IE have different lists**
  - **CA’s public key is used to verify the signatures on issued certificates**
  - **Browsers can accept unverifiable certificates or alert the user**
- Users can install additional certificates
  - **Additional Root/CA certificates**
    - Security vulnerability
  - **Client certificates**

Web Browser Support



Mostra: <Tutti>

Campo	Valore
Algoritmo della firma ...	md2RSA
Emittente	Secure Server Certification Authority, RSA D
Valido dal	lunedì 17 maggio 1999 1.00.00
Valido fino al	mercoledì 17 maggio 2000 0.59.59
Soggetto	www.zivago.com, Zivago S.p.A., Milano, Mil
Chiave pubblica	RSA (512 Bits)
Algoritmo di identific...	sha1

```
3048 0241 00C1 AFD0 1DAB 6759 50CD 66A0 D76D
143B 1588 C4E4 2257 8C1B 4019 4A83 1A8D 671E
B93C 171C 9724 2D22 31E5 B3DC 687E 2DC1 820F
A825 2A92 E6A7 6162 0ECF 0F61 0175 9B02 0301
0001
```