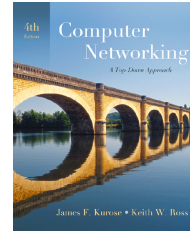


Gestion et sécurité des réseaux informatiques

Guy Leduc

Chapter 4: Securing TCP connections



Computer Networking: A Top Down Approach, 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.
(section 8.6)

Network Security - PRIVATE Communication in a PUBLIC World
C. Kaufman, R. Pearlman, M. Speciner
Pearson Education, 2002.
(chapter 19)

Computer Networks, 4th edition
Andrew S. Tanenbaum
Pearson Education, 2003.
(section 8.9.3)

4: Securing TCP connections 4-1

Chapter 4: Securing TCP connections

Chapter goals:

- security in practice:
 - Security in the transport layer (versus other layers)
 - SSL / TLS

4: Securing TCP connections 4-2

Chapter Roadmap

- Security in the transport layer
- SSL - The big picture
- SSL - A more complete picture

4: Securing TCP connections 4-3

Relative Location of Security Facilities in the TCP/IP Stack

HTTP FTP SMTP
SSL / TLS
TCP
IP

Security at transport level

HTTP FTP SMTP
TCP / UDP
IP / IPsec

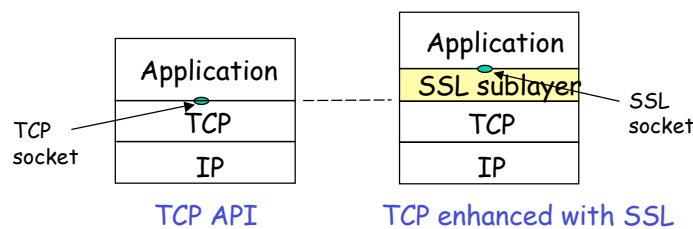
Security at network level

- Both are general-purpose (i.e. application independent) solutions
- But
 - SSL is specific to TCP
 - Does not work with UDP, contrary to IPsec
 - But makes SSL simpler (no worry about loss and retransmission of data)
 - SSL only protects the TCP payload
 - Traffic analysis is thus possible

4: Securing TCP connections 4-4

Secure sockets layer (SSL)

- ❑ provides transport layer security to any TCP-based application using SSL services
 - e.g., between Web browsers, servers for e-commerce (https)
- ❑ security services:
 - server authentication, data encryption, integrity-protection, client authentication (optional)



© From Computer Networking, by Kurose&Ross

4: Securing TCP connections 4-5

Secure Socket Layer - SSL

- ❑ SSLv2 was originated by Netscape in 1995
- ❑ Later came SSLv3
- ❑ IETF then introduced the TLS (Transport Layer Security) standard
 - Kind of SSLv3.1
 - Only minor changes, but not interoperable
 - Session key made stronger (harder to cryptanalyse)
- ❑ SSL is designed to make use of TCP to provide a reliable end-to-end secure service

4: Securing TCP connections 4-6

Chapter Roadmap

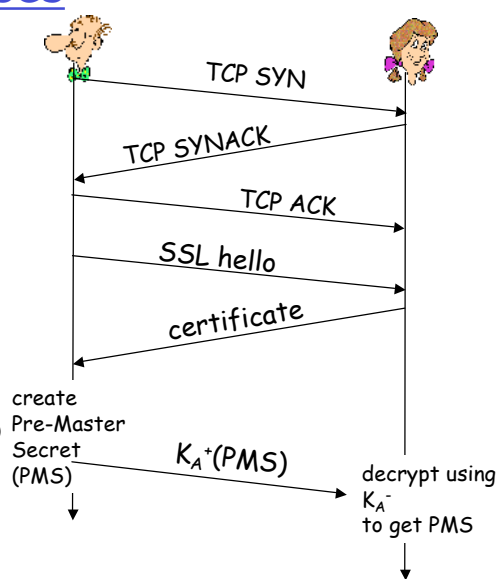
- Security in the transport layer
- **SSL - The big picture**
- SSL - A more complete picture

4: Securing TCP connections 4-7

SSL: three phases

1. Handshake:

- Bob establishes TCP connection to Alice
- authenticates Alice via CA signed certificate
- creates, encrypts (using Alice's public key), sends pre-master secret key to Alice
 - nonce exchange not shown



© From Computer Networking, by Kurose&Ross

4: Securing TCP connections 4-8

SSL: three phases

2. Key Derivation:

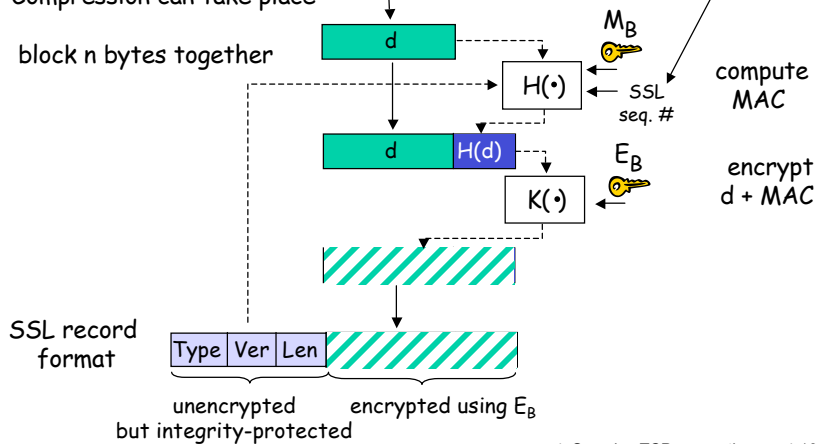
- Alice, Bob use shared pre-master secret (PMS) to generate 4 keys:
 - E_B : Bob→Alice data encryption key
 - E_A : Alice→Bob data encryption key
 - M_B : Bob→Alice MAC key
 - M_A : Alice→Bob MAC key
- encryption and MAC algorithms negotiable between Bob, Alice
- considered safer to use to use different cryptographic keys, in particular for encryption and integrity checking

SSL: three phases

3. Data transfer

TCP byte stream.
Compression can take place

block n bytes together



Chapter Roadmap

- ❑ Security in the transport layer
- ❑ SSL - The big picture
- ❑ **SSL - A more complete picture**

4: Securing TCP connections 4-11

Secure Socket Layer - SSL

- ❑ SSL is composed of 2 sublayers
 - The lower layer is the SSL Record Protocol:
 - Provides Integrity and Confidentiality
 - The main protocol of the upper layer is the SSL Handshake Protocol, that we will study in more detail

SSL Handshake Protocol	SSL Change Cipher Spec Protocol	SSL Alert Protocol	Application
SSL Record Protocol			
TCP			
IP			

4: Securing TCP connections 4-12

SSL Handshake Protocol

- ❑ The most complex part of SSL
- ❑ This protocol allows
 - the client to authenticate the server (the reverse authentication is also possible)
 - the server and the client to negotiate security parameters:
 - an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record
- ❑ This mechanism is called session creation
 - a session defines the set of cryptographic security parameters to be used
 - multiple secure TCP connections between a client and a server can share the same session
 - less computation cost
- ❑ This handshake protocol is used before any application data is transmitted

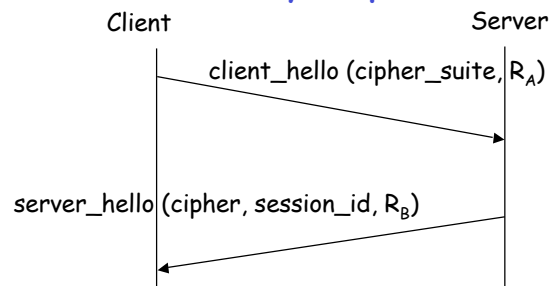
4: Securing TCP connections 4-13

The four phases of the SSL Handshake Protocol

- ❑ 1. Establish Security Capabilities
 - Protocol version, session ID, cipher suite (cryptographic algorithms) and initial random numbers
- ❑ 2. Server Authentication (and key exchange)
- ❑ 3. (Client Authentication and) key exchange
- ❑ 4. Finish

4: Securing TCP connections 4-14

Phase 1 of SSL Handshake: Establish Security Capabilities



- **Client_hello** contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference
- **Server_hello** is the selection by the server
 - Assign a session_id
 - Select the CipherSpec
- The client_hello can contain a session_id
 - To resume a previous session
- Both messages have nonces (used later to generate master secret)

4: Securing TCP connections 4-15

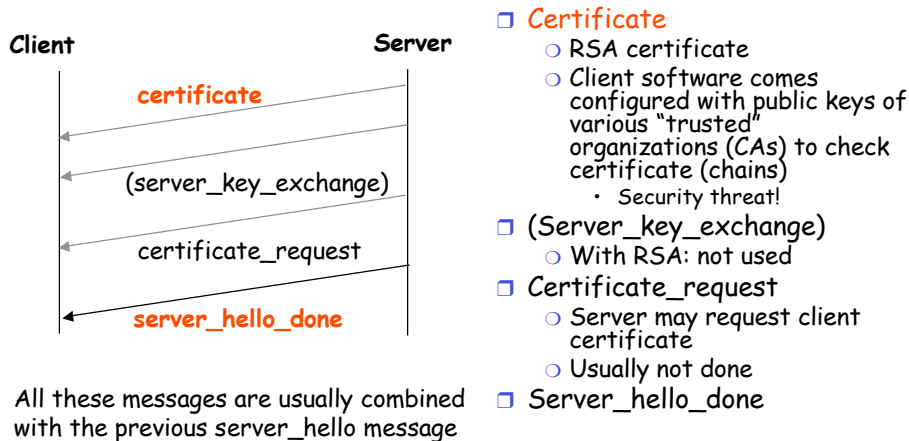
CipherSpecs

- The CipherSpec contains fields like:
 - Cipher Algorithm (DES, 3DES, ...)
 - MAC Algorithm (MD5, SHA-1)
 - ...

4: Securing TCP connections 4-16

Phase 2 of SSL Handshake: Server Authentication (and Key Exchange)

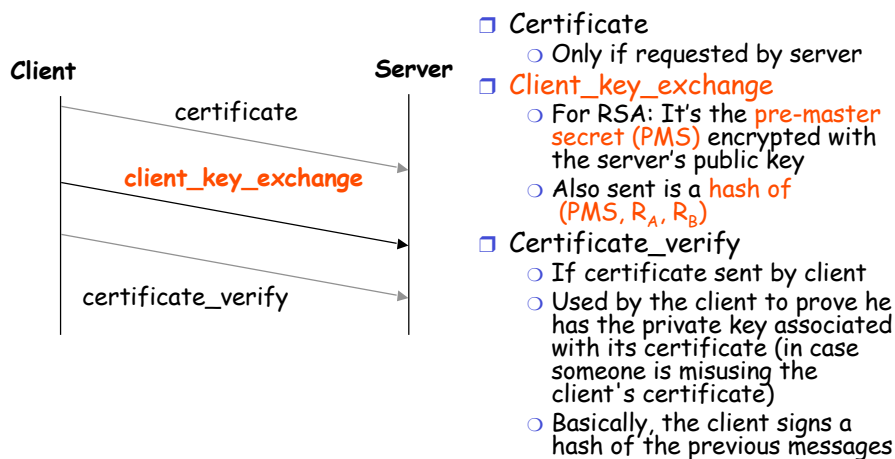
We first consider the most classical key exchange protocol: **RSA**



- **Certificate**
 - RSA certificate
 - Client software comes configured with public keys of various "trusted" organizations (CAs) to check certificate (chains)
 - Security threat!
- **(Server_key_exchange)**
 - With RSA: not used
- **Certificate_request**
 - Server may request client certificate
 - Usually not done
- **Server_hello_done**

4: Securing TCP connections 4-17

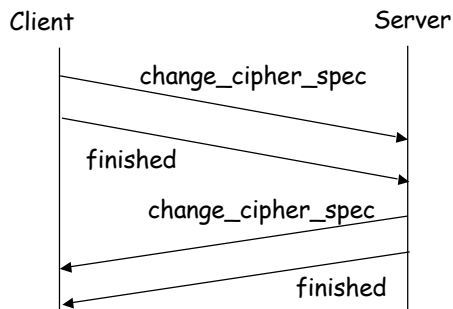
Phase 3 of SSL Handshake: (Client Authentication and) Key Exchange



- **Certificate**
 - Only if requested by server
- **Client_key_exchange**
 - For RSA: It's the **pre-master secret (PMS)** encrypted with the server's public key
 - Also sent is a **hash of (PMS, R_A, R_B)**
- **Certificate_verify**
 - If certificate sent by client
 - Used by the client to prove he has the private key associated with its certificate (in case someone is misusing the client's certificate)
 - Basically, the client signs a hash of the previous messages

4: Securing TCP connections 4-18

Phase 4 of SSL: Handshake Finish



- **Change_cipher_spec**
 - Its purpose is to cause the pending state to be copied into the current state
 - From now on, all records are encrypted and integrity-protected
 - Is part of the Change Cipher Spec protocol
- **Finished**
 - Verifies that the key exchange and authentication processes were successful
 - It is the concatenation of 2 hash values calculated from the previous messages

4: Securing TCP connections 4-19

Role of the Finish phase

- Counter the **downgrade attack**:
 - An attacker could have removed the cipher suites with strong encryption from the client_hello message, causing the entities to agree upon a weaker cipher
- Counter the **truncation attack**:
 - An attacker could close the underlying connection (by sending a TCP close message) which, in SSLv2, would have terminated the SSL session abnormally
- Note: The truncation attack can also occur later during the data transfer:
 - The solution is to indicate in the type field of the SSL record whether it is the last record
 - In normal operations, the TCP connection cannot be closed before these type fields have been exchanged over the SSL connection

4: Securing TCP connections 4-20

Main Key exchange methods based on RSA

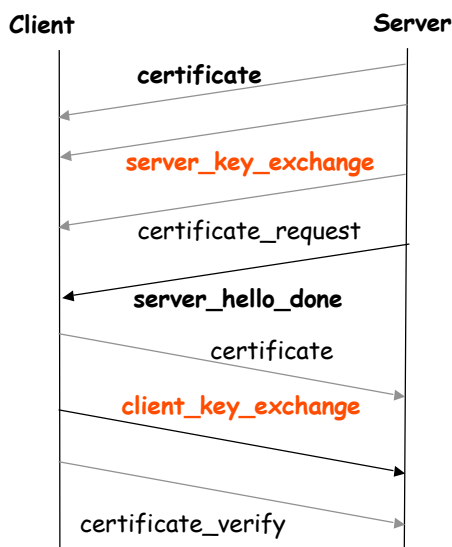
- RSA
 - The classical method shown in previous slides
 - Client sends a PMS encrypted with the server's certified RSA public key
 - Server needs a certified **encryption** public key

- RSA with signature-only key
 - Encryption with a RSA key longer than 512 bits was not exportable
 - Server first generates a temporary pair of RSA (short) keys and sends the public one to the client, signed by its RSA (long-term) key
 - Client sends a PMS **encrypted with the server's temporary RSA public key**

 - Note: this scheme designed for exportability actually enhances security because it allows (weak-key) perfect **forward secrecy**:
 - Breaking or stealing the temporary private key does not allow Trudy to decrypt previous SSL connections

4: Securing TCP connections 4-21

Phases 2 and 3: RSA with "signature-only"



- **Server_key_exchange**
 - With RSA "signature-only": it is the **temporary public key, signed by the server's long-term key**
- **Client_key_exchange**
 - It's the PMS encrypted with the server's **temporary public key**
 - + **hash of (PMS, R_A, R_B)**

4: Securing TCP connections 4-22

Other key exchange methods (DH)

- **Anonymous Diffie-Hellman (DH)**
 - Public DH parameters are sent in `server_key_exchange` and `client_key_exchange` messages
 - The pre-master secret is the shared key computed with DH
 - No need to send it
 - No protection against man-in-the-middle attack, as DH parameters are not authenticated
- **Fixed Diffie-Hellman**
 - The DH public (key) parameters are fixed and signed by a CA
 - Resists to man-in-the-middle attack, but allows an attacker to use brute force on long-standing DH public-key parameters
- **Ephemeral Diffie-Hellman**
 - Ephemeral DH public-key parameters are exchanged
 - They can vary from session to session. More robust to brute force.
 - They are signed using the sender's private RSA key
 - Resists to man-in-the-middle thanks to this authentication of public-key parameters
 - Sender should have a secret RSA key to sign
 - So, client needs a certified RSA key too!

4: Securing TCP connections 4-23

Master secret and keys

- The previous phases have generated a **pre-master secret**
 - Key chosen and sent encrypted to the server (with RSA)
 - Or, the DH secret key
- The **master secret** is generated from
 - The pre-master secret
 - The two nonces (R_A and R_B) exchanged in the `client_hello` and `server_hello` messages
- **Six keys** are derived from this master secret:
 - Secret key used with MAC (for data sent by server)
 - Secret key used with MAC (for data sent by client)
 - Secret key and IV used for encryption (by server)
 - Secret key and IV used for encryption (by client)

4: Securing TCP connections 4-24

Server Gated Cryptography (SGC)

- The US allows an exported client to use strong crypto when talking to some servers doing financial transactions
 - Those servers have an SGC certificate signed by Verisign (trusted by the Government, which turns out to be the real matter!)
 - This is wired in the implementation
 - Other trust authorities can be modified by the user
- Start with weak (exportable) cryptography, and then upgrade to strong cryptography if the server has an SGC certificate
 - The SGC certificate is discovered in phase 2 only
 - The client continues with a second handshake protected by the first master secret
 - Use Change_Cipher_Spec to switch to strong crypto
 - This does not require the server to run any special SGC code

SSL Alert Protocol

- This protocol is used to report errors
 - Examples
 - Unexpected message
 - Bad record MAC
 - Decompression failure
 - Handshake failure (i.e. security parameters negotiation failed)
 - Illegal parameters
- It is also used for other purposes
 - Examples
 - To notify closure of the TCP connection
 - To notify the absence of certificate (when requested)
 - To notify that a bad or unknown certificate was received
 - To notify that a certificate is revoked or has expired

Pros and Cons of Transport Layer Security

□ Pros

- Transport Layer Security is transparent to applications
- Server is authenticated
- Application layer headers are hidden
- OK for direct client to server communication
- More fine-grained than IPSec (see later) because it works at the transport connection level

□ Cons

- TCP/IP headers are in clear
- Only applicable to secure TCP-based applications (not UDP)
- Not enough to secure applications using intermediate servers and a chain of TCP connections (e.g. email)
- Nonrepudiation not provided by SSL
- Client authentication, if needed, must be implemented above SSL (e.g. username and password sent over the SSL connection)