

# Internet Security

## CSS441: Security and Cryptography

Sirindhorn International Institute of Technology  
Thammasat University

Prepared by Steven Gordon on 20 December 2015  
`css441y15s2l11, Steve/Courses/2015/s2/css441/lectures/internet-security.tex, r4295`

# Contents

## Web Security Issues

## TLS/SSL

## HTTPS

## Secure Shell

# Web Security Issues

- ▶ Original Internet protocols do not have built-in security (IP, TCP, HTTP, ...)
- ▶ Many threats arise for web and other Internet applications
- ▶ Issues at: client, server and traffic between client and server
- ▶ Cover: SSL/TLS, SSH, IPsec

# Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
<b>Integrity</b>	<ul style="list-style-type: none"> <li>•Modification of user data</li> <li>•Trojan horse browser</li> <li>•Modification of memory</li> <li>•Modification of message traffic in transit</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Compromise of machine</li> <li>•Vulnerabilty to all other threats</li> </ul>	Cryptographic checksums
<b>Confidentiality</b>	<ul style="list-style-type: none"> <li>•Eavesdropping on the net</li> <li>•Theft of info from server</li> <li>•Theft of data from client</li> <li>•Info about network configuration</li> <li>•Info about which client talks to server</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Loss of privacy</li> </ul>	Encryption, Web proxies
<b>Denial of Service</b>	<ul style="list-style-type: none"> <li>•Killing of user threads</li> <li>•Flooding machine with bogus requests</li> <li>•Filling up disk or memory</li> <li>•Isolating machine by DNS attacks</li> </ul>	<ul style="list-style-type: none"> <li>•Disruptive</li> <li>•Annoying</li> <li>•Prevent user from getting work done</li> </ul>	Difficult to prevent
<b>Authentication</b>	<ul style="list-style-type: none"> <li>•Impersonation of legitimate users</li> <li>•Data forgery</li> </ul>	<ul style="list-style-type: none"> <li>•Misrepresentation of user</li> <li>•Belief that false information is valid</li> </ul>	Cryptographic techniques

# Security Options in TCP/IP

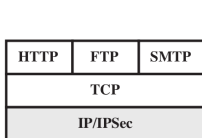
Internet Security

Web Security

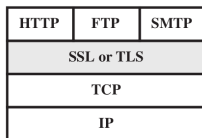
TLS/SSL

HTTPS

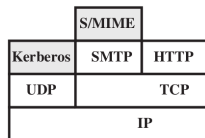
SSH



(a) Network Level



(b) Transport Level



(c) Application Level

- ▶ **IPsec**: Security for IP datagrams; general solution for all Internet traffic; implemented in OS
- ▶ **SSL/TLS**: Security for TCP segments; general solution for all TCP-based applications; implemented in libraries/applications (e.g. OpenSSL)
- ▶ **Application-specific**: Security for application messages; specific to each applications; implemented in single application

# Contents

Web Security Issues

TLS/SSL

HTTPS

Secure Shell

# SSL and TLS

- ▶ Secure Sockets Layer (SSL) originated in Netscape web browser
- ▶ Transport Layer Security (TLS) standardised by IETF
- ▶ SSLv3 and TLS are almost the same
- ▶ SSL provides security services to application layer protocols using TCP
- ▶ SSL architecture consists of multiple protocols

# SSL Architecture

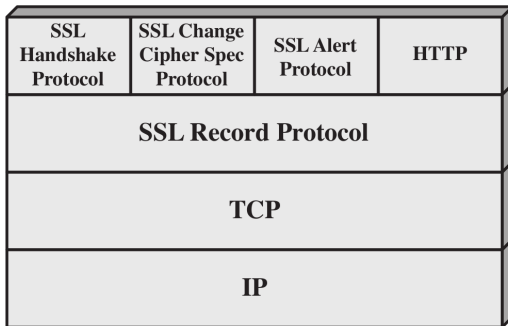
Internet Security

Web Security

TLS/SSL

HTTPS

SSH



**Record:** provides confidentiality and message integrity

**Handshake:** authenticate entities, negotiate parameter values

**Change Cipher:** change cipher for use in connection

**Alert:** alert peer entity of status/warning/error



## Connections and Sessions

- ▶ SSL connection corresponds with TCP connection
  - ▶ Client and server may have multiple connections
- ▶ SSL session is association between client and server
  - ▶ Session created with Handshake protocol
  - ▶ Multiple connections can be associated with one session
  - ▶ Security parameters for session can be shared for connections
- ▶ State information is stored after Handshake protocol
  - ▶ Session: ID, certificate, compression, cipher spec, master secret, . . .
  - ▶ Connection: random values, encrypt keys, MAC secrets, IV, sequence numbers, . . .

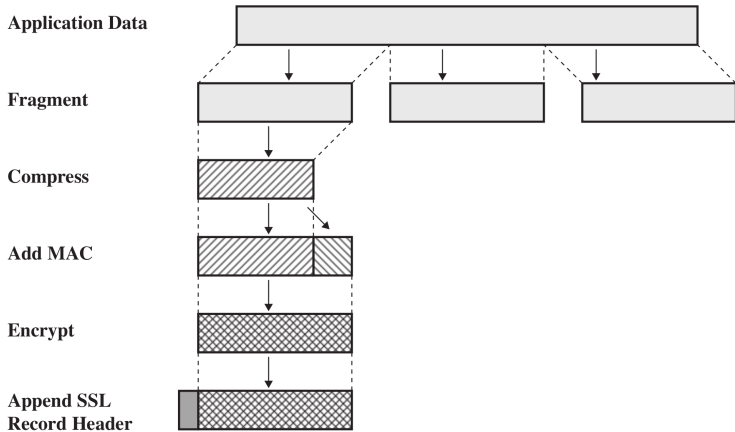
# SSL Record Protocol Operation

Web Security

TLS/SSL

HTTPS

SSH



# SSL Record Protocol

- ▶ Fragmentation: maximum fragment size is 16384 Bytes
- ▶ Compression: lossless; algorithm chosen in Handshake
- ▶ MAC: HMAC applied on compressed data; MAC secret key for connection used; MAC appended to compressed fragment
- ▶ Encrypt: applied to compressed fragment and MAC; algorithm chosen in Handshake
- ▶ SSL record header:
  - ▶ Content type: higher layer protocol (change cipher spec, alert, handshake, application)
  - ▶ Version
  - ▶ Compressed length in bytes

# SSL Record Format

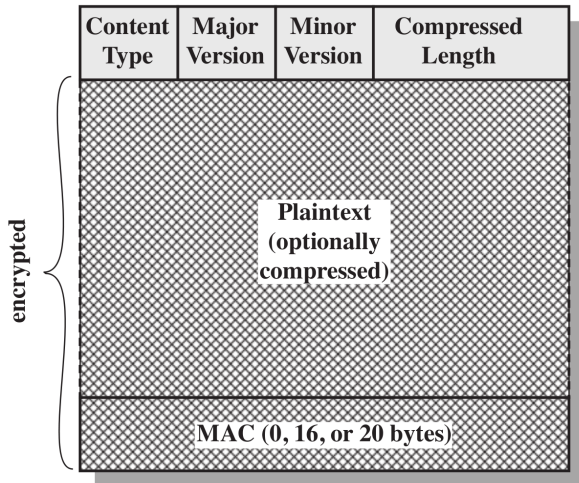
Internet Security

Web Security

TLS/SSL

HTTPS

SSH



# SSL Record Protocol Payload

Internet Security

Web Security

TLS/SSL

HTTPS

SSH

1 byte



(a) Change Cipher Spec Protocol

1 byte

3 bytes

 $\geq 0$  bytes

(c) Handshake Protocol

1 byte 1 byte



(b) Alert Protocol

 $\geq 1$  byte

(d) Other Upper-Layer Protocol (e.g., HTTP)

# SSL Handshake Protocol

- ▶ Allow client and server to authenticate each other
- ▶ Negotiate encryption and MAC algorithms, exchange keys
  - ▶ Key Exchange: RSA, Diffie-Hellman
  - ▶ MAC: HMAC using SHA or MD5
  - ▶ Encryption: RC4, RC2, DES, 3DES, IDEA, AES
- ▶ Multiple phases:
  1. Establish security capabilities: client proposes algorithms, server selects one
  2. Server authentication and key exchange
  3. Client authentication and key exchange
  4. Finish setting up connection

# SSL Handshake Protocol Messages

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

# SSL Handshake Protocol Operation

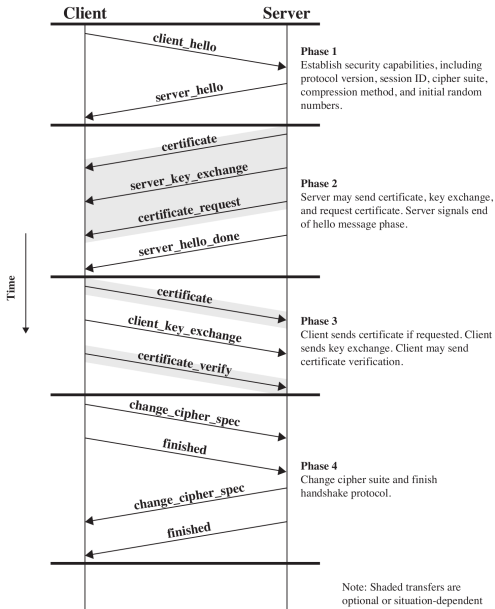
Internet Security

Web Security

TLS/SSL

HTTPS

SSH





# Contents

Web Security Issues

TLS/SSL

**HTTPS**

Secure Shell

# HTTPS

- ▶ HTTPS: HTTP over SSL (or TLS)
- ▶ URL uses https://
- ▶ Web server listens on port 443
- ▶ Encrypt: URL of requested document, contents of document, contents of browser forms, cookies, contents of HTTP header
- ▶ Server is authenticated using certificate (using SSL)
- ▶ Client is authenticated using password (using HTTP)

# Contents

Web Security Issues

TLS/SSL

HTTPS

Secure Shell

## Secure Shell

- ▶ TELNET provides a remote login facility; insecure
- ▶ Secure Shell (SSH) designed for secure remote login
- ▶ SSH also supports secure file transfer and tunnelling
- ▶ SSHv2 developed by IETF
- ▶ SSH architecture consists of 3 protocols

# SSH Protocol Stack

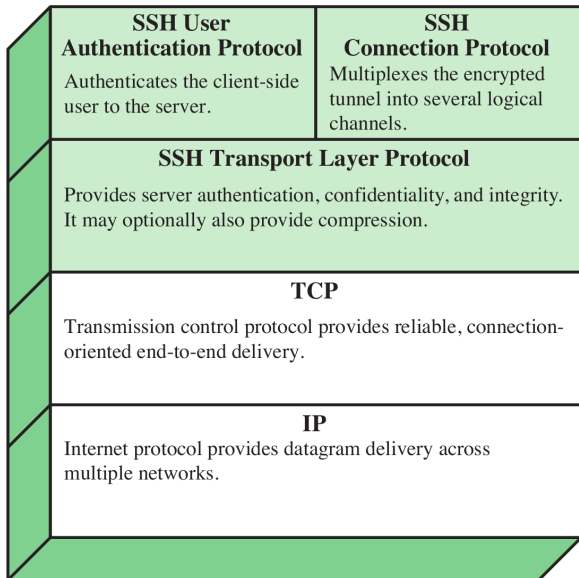
Internet Security

Web Security

TLS/SSL

HTTPS

SSH



# SSH Authentication

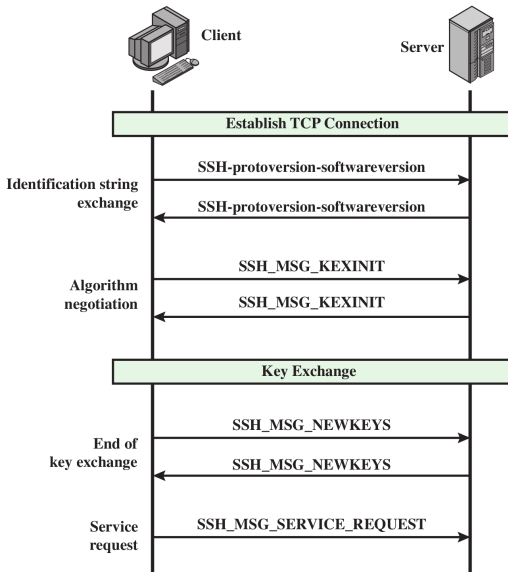
## Server Authentication

- ▶ Server has public/private key pair
- ▶ Assume client knows server's public key
- ▶ During key exchange, server signs message with public key

## Client Authentication

- ▶ Key-based: client has public/private key pair; server knows client public key
- ▶ Password-based: client sends password (encrypted); server knows password

# SSH Transport Layer Packet Exchange



# SSH Transport Layer Protocol

- ▶ Identification string exchange: each entity identifies protocol and software version
- ▶ Algorithm negotiation: client and server send list of supported algorithms, in order of preference; first common algorithm chosen
- ▶ Key exchange: Diffie-Hellman
- ▶ End of key exchange: new keys generated from shared secret, e.g.

$$K_{c2s} = \text{Hash}(K || H || C' || \text{session\_id})$$

where

$$H = \text{Hash}(ID_C || ID_S || M_C || M_S || PU_S || Y_A || Y_B || K)$$

- ▶ Service request for User Authentication or Connection Protocol



# SSH Algorithms

Cipher	
3des-cbc*	Three-key 3DES in CBC mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Twofish in CBC mode with a 256-bit key
twofish192-cbc	Twofish with a 192-bit key
twofish128-cbc	Twofish with a 128-bit key
aes256-cbc	AES in CBC mode with a 256-bit key
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC

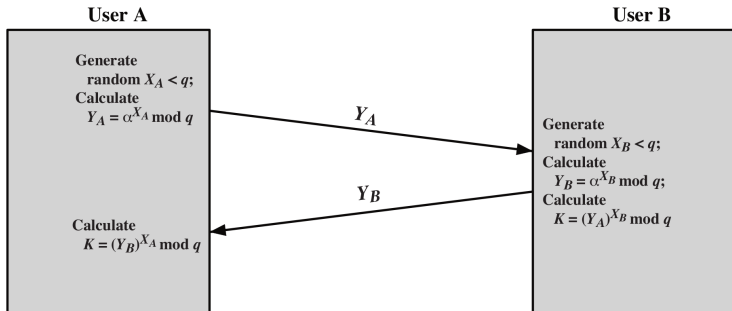
MAC algorithm	
hmac-sha1*	HMAC-SHA1; digest length = key length = 20
hmac-sha1-96**	First 96 bits of HMAC-SHA1; digest length = 12; key length = 20
hmac-md5	HMAC-SHA1; digest length = key length = 16
hmac-md5-96	First 96 bits of HMAC-SHA1; digest length = 12; key length = 16

Compression algorithm	
none*	No compression
zlib	Defined in RFC 1950 and RFC 1951

\* = Required

\*\* = Recommended

# Key Exchange with Diffie-Hellman



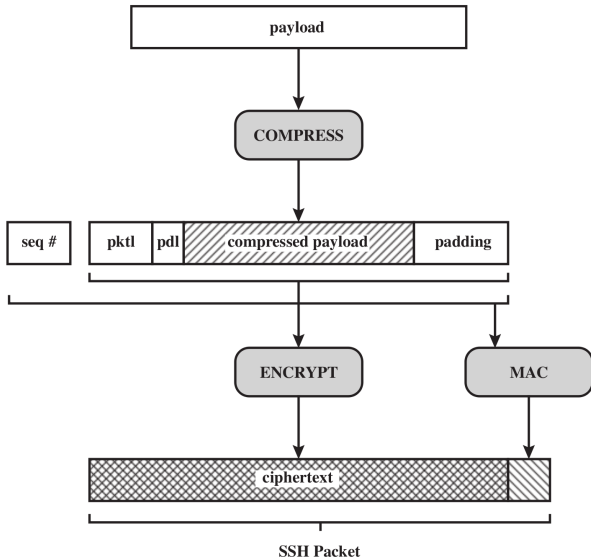
## SSH Key Exchange with Diffie-Hellman

- ▶ SSH notation:  $q = P$ ,  $\alpha = G$ ,  $Y_A = e$ ,  $Y_B = f$
- ▶ ID string for client and server:  $ID_C$ ,  $ID_S$ ;  
SSH\_MSG\_KEXINIT message from client and server:  
 $M_C$ ,  $M_S$
- ▶ Server key pair:  $(PU_S, PR_S)$ ; assume client knows/trusts  $PU_S$
- ▶ Client and server have agreed upon hash and encryption algorithms

# SSH Key Exchange with Diffie-Hellman

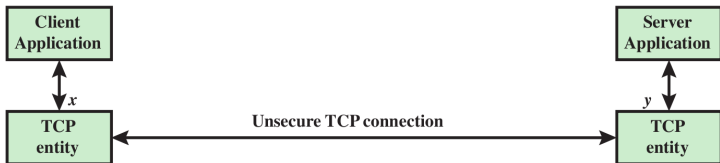
(see Wireshark capture)

# SSH Transport Layer Packet Formation



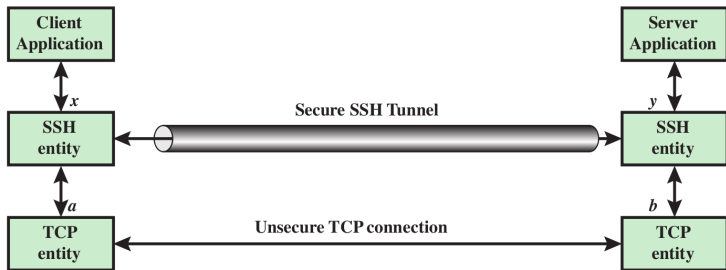
pktl = packet length  
pdl = padding length

# TCP Connection



$a$  and  $b$  are application port numbers

# SSH Tunnel over TCP Connection



$x$  and  $y$  are application port numbers,  $a$  and  $b$  are port numbers used by SSH

# SSH Tunnels

- ▶ Allow normal (unsecured) applications to securely transfer data
- ▶ Bypass firewalls by using different ports
- ▶ Local forwarding: traffic to local port is sent via SSH client to remote port
- ▶ Remote forwarding: traffic to remote port is sent via SSH server to local port