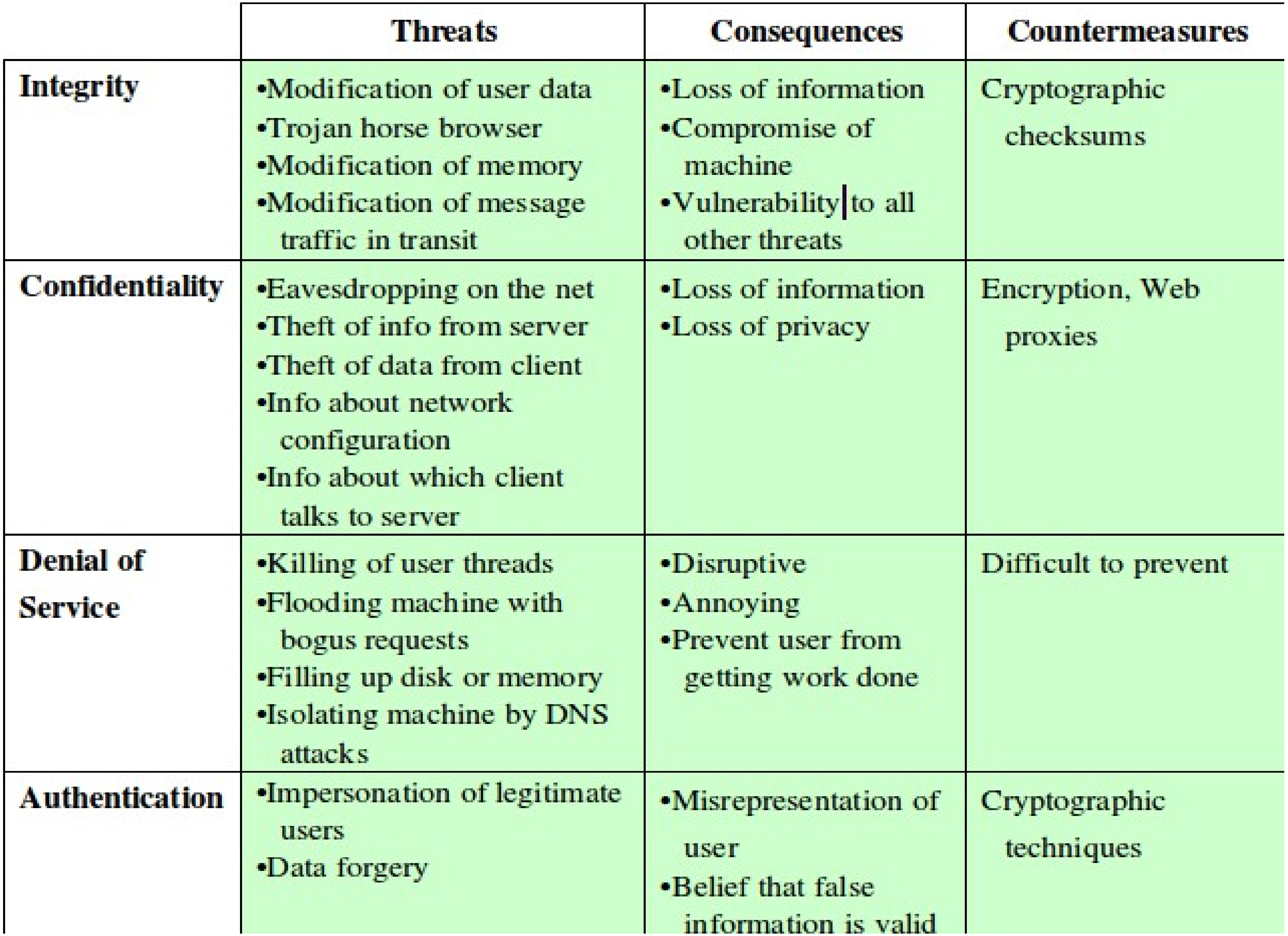
WEB SECURITY
Section 16.1, 16.2, 16.3 and 16.4

Web Security Considerations

- The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
- The following characteristics of Web usage suggest the need for tailored security tools:
  - Web servers are relatively easy to configure and manage
  - Web content is increasingly easy to develop
  - The underlying software is extraordinarily complex
  - May hide many potential security flaws
  - A Web server can be exploited as a launching pad into the corporation’s or agency’s entire computer complex
  - Casual and untrained (in security matters) users are common clients for Web-based services
  - Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

Table 6.1 A Comparison of Threats on the Web



	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerability to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

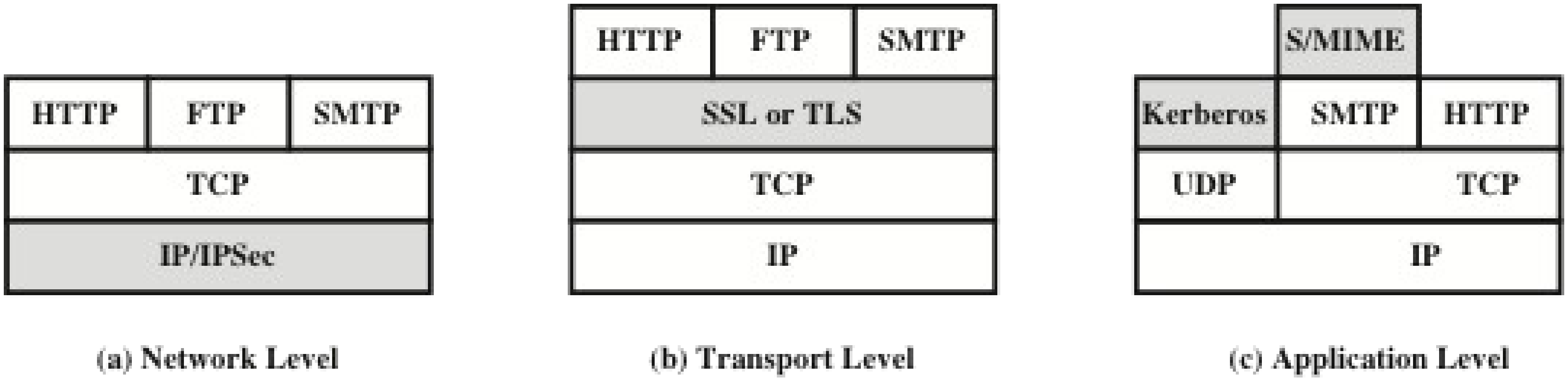


Figure 6.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

SSL , TLS http://en.wikipedia.org/wiki/Transport_Layer_Security

- **Transport Layer Security (TLS)** and its predecessor, **Secure Sockets Layer (SSL)**, are cryptographic protocols which are designed to provide communication security over the Internet.
- They use X.509 certificates and hence asymmetric cryptography to assure the counter party with whom they are communicating, and to exchange a symmetric key.
- This session key is then used to encrypt data flowing between the parties. This allows for data/message confidentiality, and message authentication codes for message integrity and as a by-product, message authentication.
- Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging, and voice-over IP (VoIP).
- An important property in this context is forward secrecy, so the short term session key cannot be derived from the long term asymmetric secret key.

SSL, TLS

- Applications use TLS or SSL to establish secure connections between two communicating parties. The primary goal of both protocols is to provide privacy and data integrity. Other goals are as follows:

- Enabling interoperability between applications
- Providing an extensible framework that can readily incorporate new public key and bulk encryption methods
- Ensuring relative computational efficiency

- Both TLS and SSL comprise two layers: a Record Protocol and a Handshake Protocol.

- Although the two protocols are similar, the differences are sufficiently significant that SSL 3.0 and the various versions of TLS do not interoperate.
- **Recommended Video: Using SSL/TLS** http://www.youtube.com/watch?v=RPvqpjLqcbQ

- One of the most widely used security services
- A general purpose service implemented as a set of protocols that rely on TCP
  - Could be provided as part of the underlying protocol suite and therefore be transparent to applications
  - Can be embedded in specific packages

SSL and TLS

- SSL was developed by Netscape and is now used by most browsers including Microsoft
- TLS working group was formed within IETF
- First version of TLS can be viewed as an SSLv3.1 and became Internet Standard
- Uses TCP to provide relaible end to end service

SSL Architecture

- SSL is not a single protocol, but two layers of protocols:
- SSL Record Protocol provides basic security services to higher layer protocols, especially to HTTP
- 3 higher layer protocols are part of SSL:
  - Handshake protocol
  - Change Cipher Spec protocol
  - Alert protocol

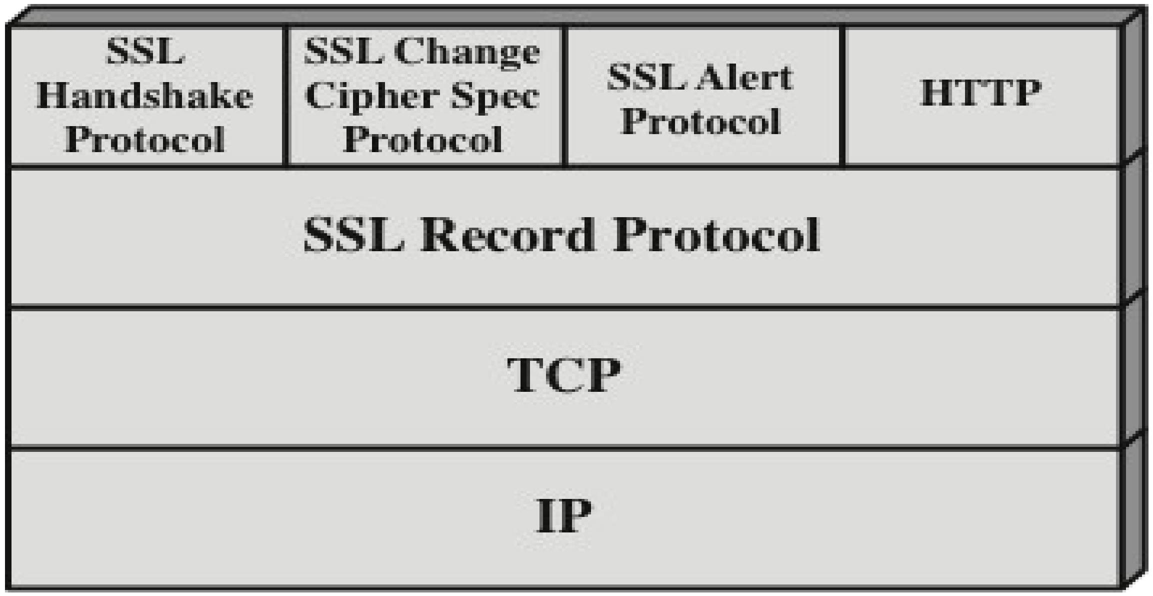


Figure 6.2 SSL Protocol Stack

SSL Architecture

- **SSL connection**
  - connections are transient, peer-to-peer
  - every connection associated with one SSL session
- **SSL session**
  - an association between client and server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections

SSL Session State

- A Session state is defined by:
  - Session identifier-id for session state
  - Peer Certificate – X509.v3 certificate
  - Compression Method – used before encryption
  - Cipher Space- data encryption algorithm
  - Master Secret- shared between client and server
  - Is resumable- can initiate new connection

SSL Connection State

- A connection state is defined by:
- Server and Client random
- Server write MAC secret
- Client write Mac secret
- Server write key
- Client write key
- Initialization vectors
- Sequence numbers

A connection state is defined by the following parameters.

• **Server and client random**: Byte sequences that are chosen by the server and client for each connection.

• **Server write MAC secret**: The secret key used in MAC operations on data
sent by the server.

• **Client write MAC secret**: The secret key used in MAC operations on data sent by the client.

A connection state is defined by the following parameters:

• **Server write key:** The secret encryption key for data encrypted by the server and decrypted by the client.

• **Client write key:** The symmetric encryption key for data encrypted by the client and decrypted by the server.

• **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.

• **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero.

SSL Record Protocol

SSL Record Protocol defines two services for SSL connections:

- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.

- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC), which is similar to HMAC

SSL Record Protocol

- SSL Record Protocol provides two services for SSL connection:
- **confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - AES, 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

SSL Record Protocol Operation

- Takes and application message to be transmitted, fragments it into blocks, compresses the data, applies a MAC, encrypts, adds a header and transmits the unit in a TCP segment
- Received data are decrypted, verified, decompressed, and reassembled and then delivered to high-level user.

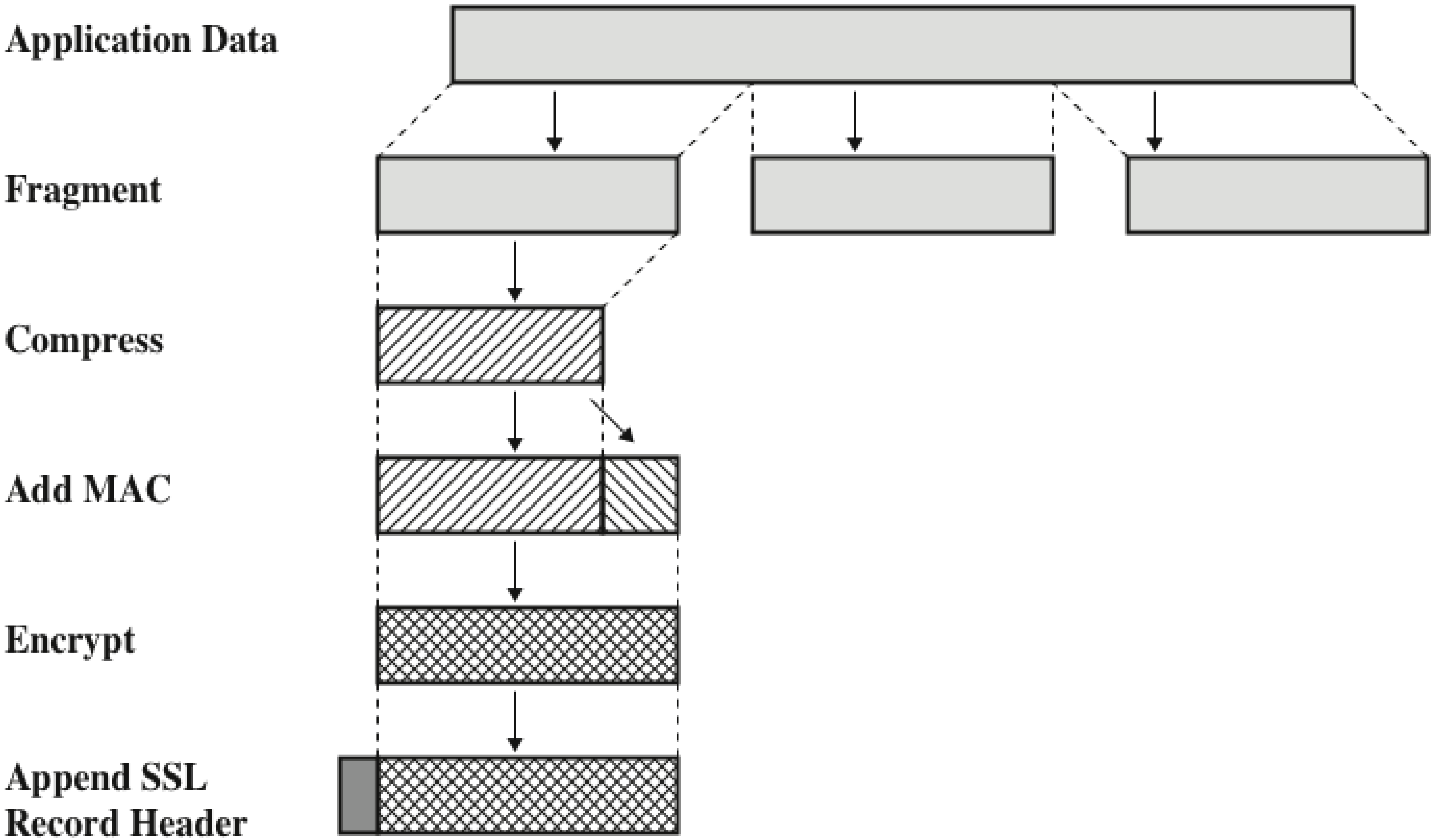
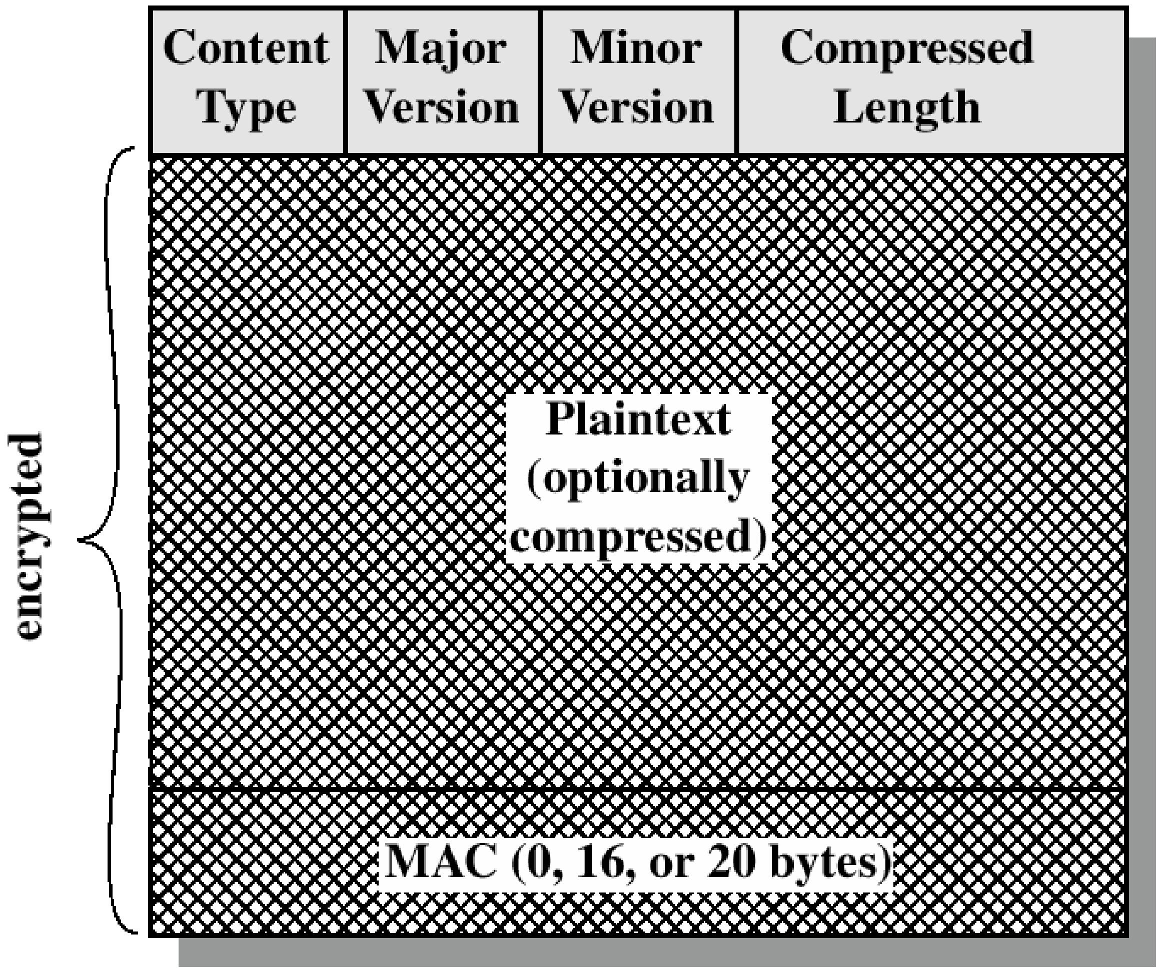


Figure 17.3 SSL Record Protocol Operation

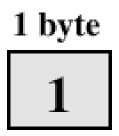
SSL Record Protocol Operation

- Fragmentation- message broken into 2^{14} byte blocks
- Compression – lossless and optional
- Compute a message authention code (MAC) over the compressed data, using a shared secret key
- Compresses message and MAC are encrypted using symmetric encryption
- A header is prepared containing content type, versions and compressed length

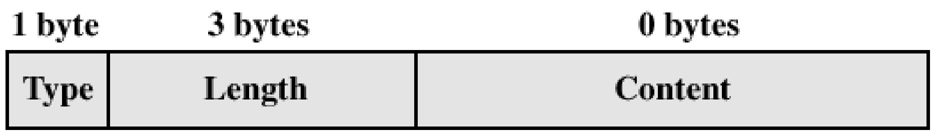
SSL Record Format



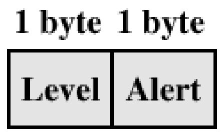
SSL Record Protocol Payload



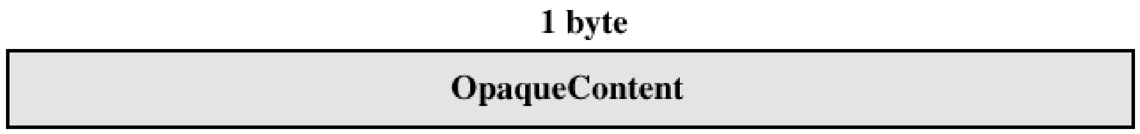
(a) Change Cipher Spec Protocol



(c) Handshake Protocol



(b) Alert Protocol



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

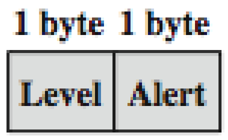
SSL Change Cipher Spec Protocol

- One of 3 SSL specific protocols which use the SSL Record protocol-simplest
- A single message, single byte = 1
- Purpose causes pending state to become current
- Updates the cipher suite in use

(a) Change Cipher Spec Protocol

SSL Alert Protocol

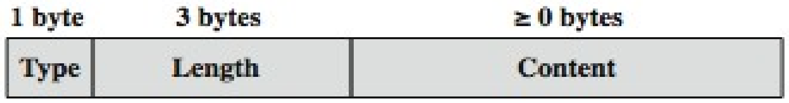
- Conveys SSL-related alerts to peer entity
- First byte - severity
  - warning or fatal (terminates)
- Second byte - specific alert
  - fatal: unexpected message, bad record MAC, decompression failure, handshake failure, illegal parameter
  - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- compressed and encrypted like all SSL data



(b) Alert Protocol

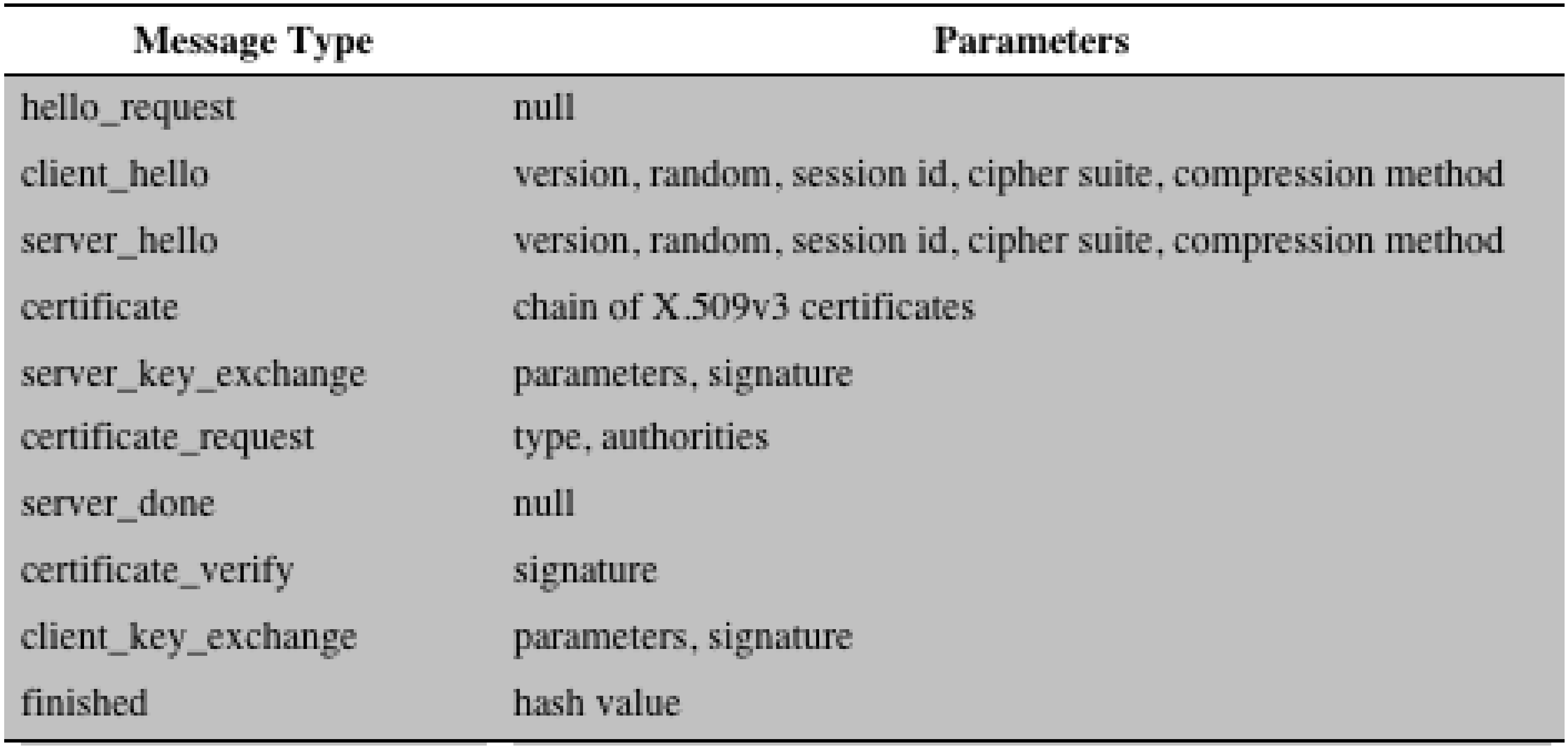
Handshake Protocol

- Most complex part of SSL. Used before application data are transmitted.
- Allows the server and client:
  - to authenticate each other.
  - negotiate encryption, MAC algorithm and cryptographic keys.



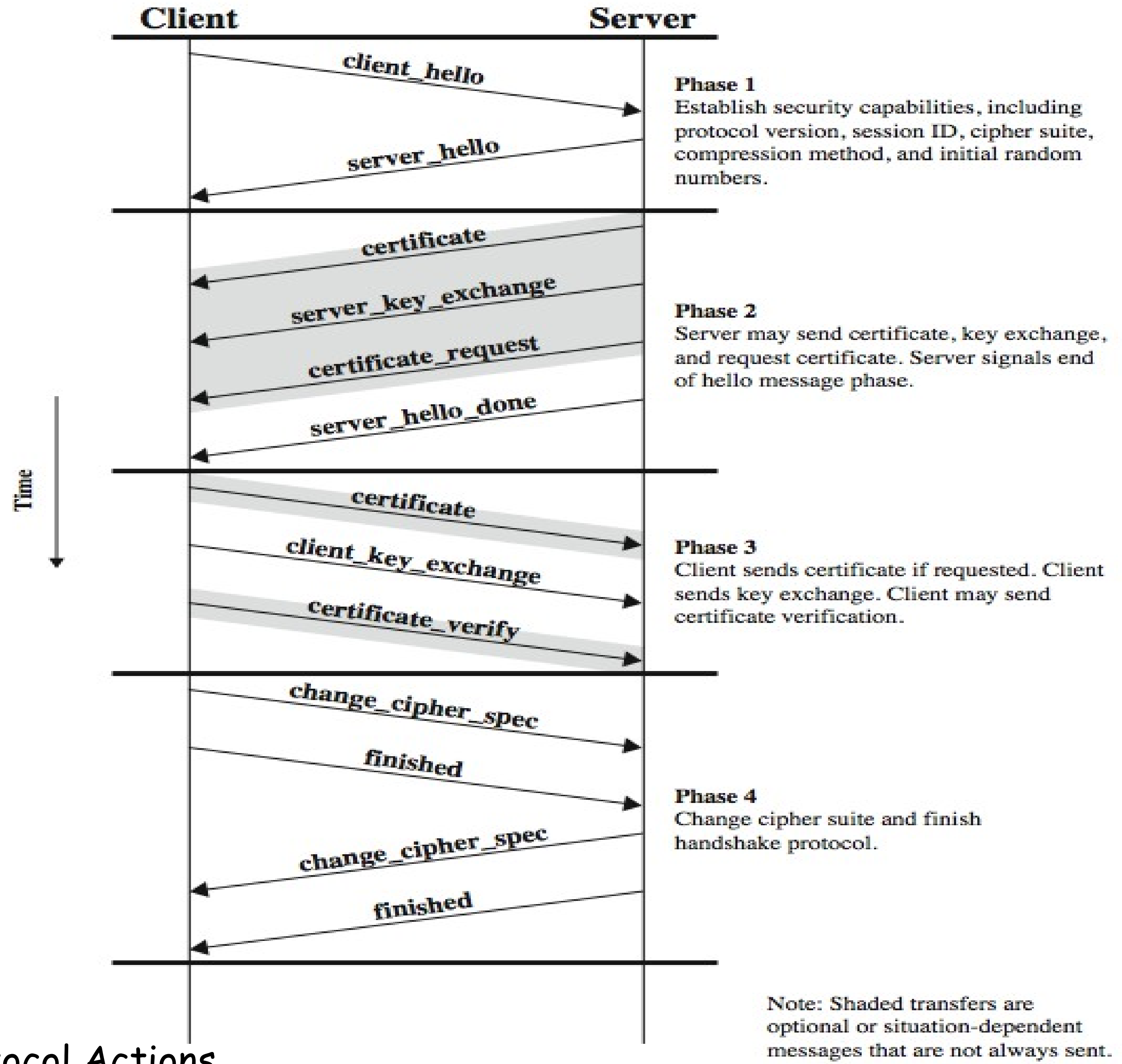
(c) Handshake Protocol

- Comprises a series of messages in phases
  1. Establish Security Capabilities (RSA, DH…)
  2. Server Authentication and Key Exchange
  3. Client Authentication and Key Exchange
  4. Finish



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

Table SSL Handshake Protocol Message Types


Client
Server
client_hello
server_hello
certificate
server_key_exchange
certificate_request
server_hello_done
certificate
client_key_exchange
certificate_verify
change_cipher_spec
finished
change_cipher_spec
finished
Time
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.

Handshake Protocol Actions

Cryptographic Computations

- Two further items are of interest:
  - The creation of a shared master secret by means of the key exchange
    - The shared master secret is a one-time 48-byte value generated for this session by means of secure key exchange
  - The generation of cryptographic parameters from the master secret
    - CipherSpecs require a 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 which are generated from the master secret in that order
      - These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters

Transport Layer Security (TLS)

- An IETF standardization initiative whose goal is to produce an Internet standard version of SSL
- Is defined as a Proposed Internet Standard in RFC 5246
  - RFC 5246 is very similar to SSLv3
  - Differences include:
    - Version number
    - Message Authentication Code
    - Pseudorandom function
    - Alert keys
    - Cipher suites
    - Client certificate types
    - Certificate_verify and Finished Messages
    - Cryptographic computations
    - Padding

Transport Layer Security

- IETF standard RFC 2246 similar to SSLv3
- Minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
    - based on HMAC using SHA-1 or MD5
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate types & negotiations
  - changes in crypto computations & padding

Let us Recall HMAC

http://en.wikipedia.org/wiki/HMAC

Then HMAC(K,m) is mathematically defined by

$$\mathrm{HMAC}(K,m) = H((K \oplus opad) \parallel H((K \oplus ipad) \parallel m))$$

- $H(\cdot)$ be a cryptographic hash function
- K be a secret key padded to the right with extra zeros to the block size of the hash function
- m be the message to be authenticated
- $\parallel$ denote concatenation and $\oplus$ denote exclusive or (XOR)
- opad be the outer padding (0x5c5c5c…5c5c, one-block-long hexadecimal constant)
- ipad be the inner padding (0x363636…3636, one-block-long hexadecimal constant)

HMAC

- specified as Internet standard RFC2104
- uses hash function on the message:

$$\mathrm{HMAC}_K(M) = \mathrm{Hash}[(K^+ \text{ XOR } opad) \;||\; \mathrm{Hash}[(K^+ \text{ XOR } ipad) \;||\; M)]\;]$$

  - where K^+ is the key padded out to size
  - `opad`, `ipad` are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

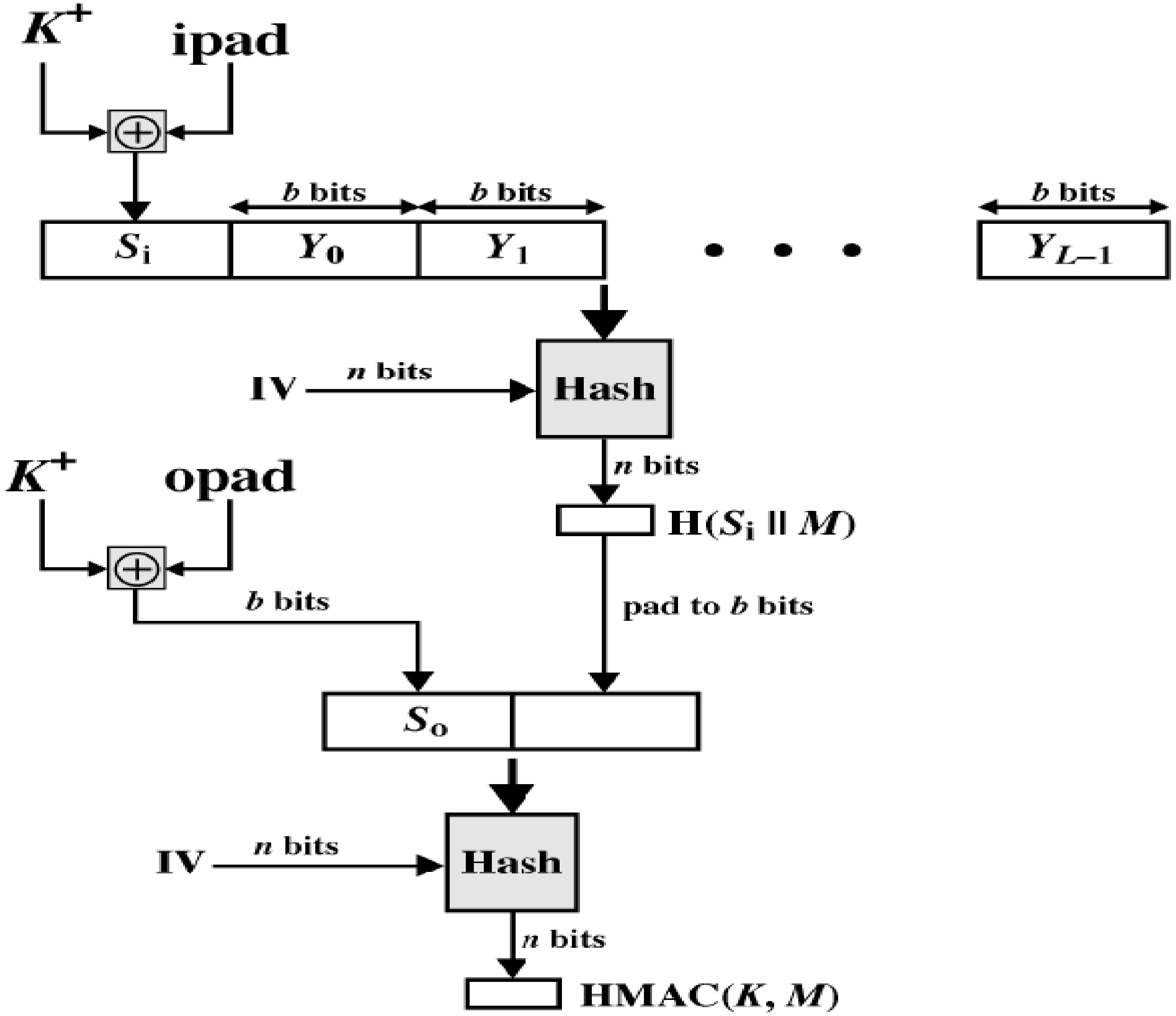


Figure 3.6 HMAC Structure

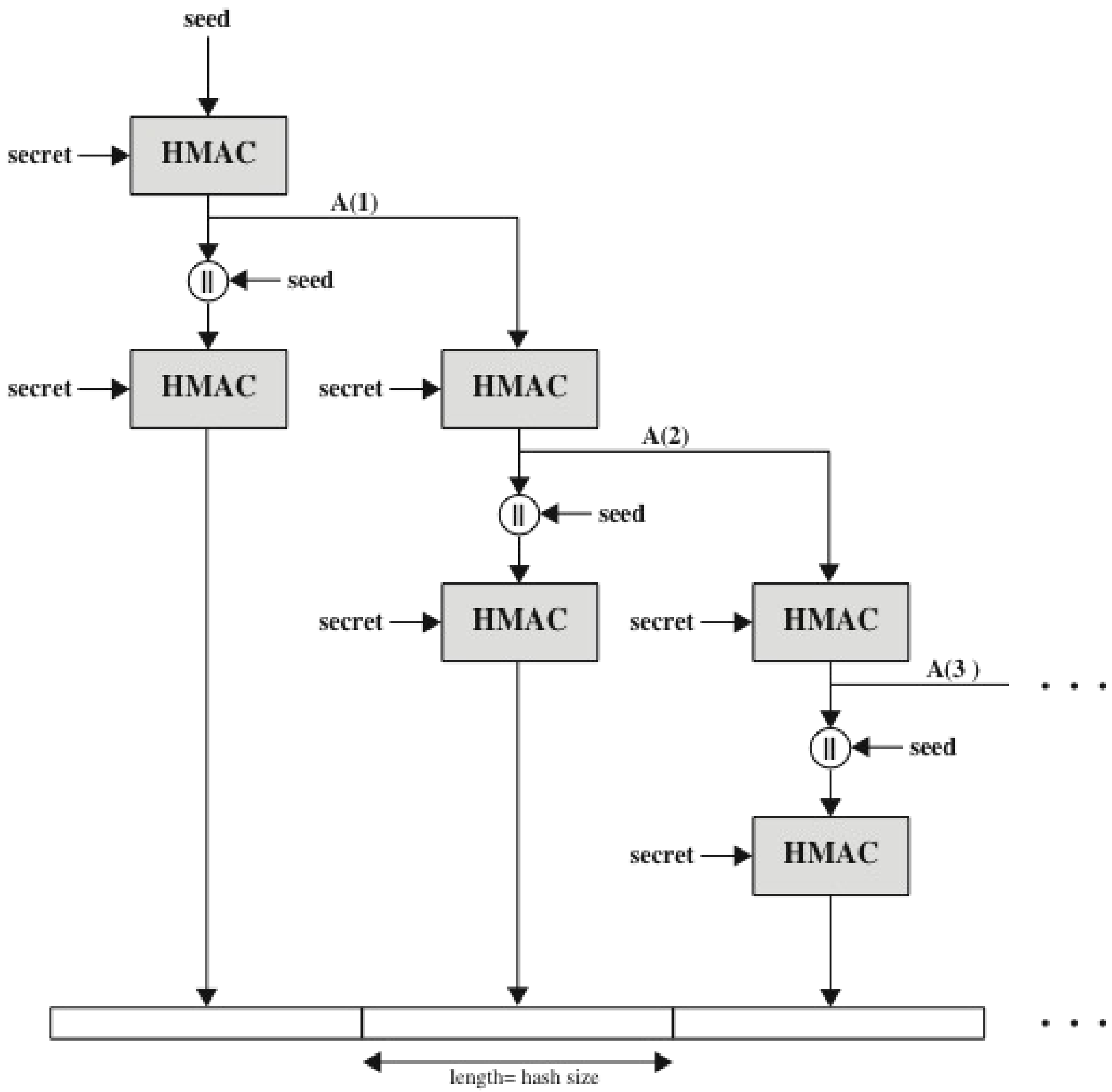


Figure 17.7 TLS Function P_hash (secret, seed)

HTTPS (HTTP over SSL)

- Refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- The HTTPS capability is built into all modern Web browsers
- A user of a Web browser will see URL addresses that begin with https:// rather than http://
- If HTTPS is specified, port 443 is used, which invokes SSL
- Documented in RFC 2818, *HTTP Over TLS*
  - There is no fundamental change in using HTTP over either SSL or TLS and both implementations are referred to as HTTPS
- When HTTPS is used, the following elements of the communication are encrypted:
  - URL of the requested document
  - Contents of the document
  - Contents of browser forms
  - Cookies sent from browser to server and from server to browser
  - Contents of HTTP header

Connection Initiation

For HTTPS, the agent acting as the HTTP client also acts as the TLS client. The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake.

When the TLS handshake has finished, the client may then initiate the first HTTP request.

All HTTP data is to be sent as TLS application data. Normal HTTP behavior, including retained connections, should be followed.

There are three levels of awareness of a connection in HTTPS. At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer. Typically, the next lowest layer is TCP, but it also may be TLS/SSL.

At the level of TLS, a session is established between a TLS client and a TLS server. This session can support one or more connections at any time. As we have seen, a TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side.

Connection Closure

- An HTTP client or server can indicate the closing of a connection by including the line `Connection: close` in an HTTP record
- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection
- TLS implementations must initiate an exchange of closure alerts before closing a connection
  - A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”
- An unannounced TCP closure could be evidence of some sort of attack so the HTTPS client should issue some sort of security warning when this occurs

Security of SSL and TLS

Reference: http://en.wikipedia.org/wiki/Transport_Layer_Security#Security

- SSL 2.0 is flawed in a variety of ways

- Identical cryptographic keys are used for message authentication and encryption.
- SSL 2.0 has a weak MAC construction that uses the MD5 hash function with a secret prefix, making it vulnerable to length extension attacks.
- SSL 2.0 does not have any protection for the handshake, meaning a man-in-the-middle downgrade attack can go undetected.
- SSL 2.0 uses the TCP connection close to indicate the end of data. This means that truncation attacks are possible: the attacker simply forges a TCP FIN, leaving the recipient unaware of an illegitimate end of data message (SSL 3.0 fixes this problem by having an explicit closure alert).
- SSL 2.0 assumes a single service and a fixed domain certificate, which clashes with the standard feature of virtual hosting in Web servers. This means that most websites are practically impaired from using SSL.

- SSL 2.0 is disabled by default, beginning with Internet Explorer 7, Mozilla Firefox 2 Opera 9.5 and Safari

Security of SSL and TLS

- SSL 3.0 improved upon SSL 2.0 by adding SHA-1 based ciphers and support for certificate authentication.

- From a security standpoint, SSL 3.0 should be considered less desirable than TLS 1.0.
- The SSL 3.0 cipher suites have a weaker key derivation process; half of the master key that is established is fully dependent on the MD5 hash function, which is not resistant to collisions and is, therefore, not considered secure.

  Under TLS 1.0, the master key that is established depends on both MD5 and SHA-1 so its derivation process is not currently considered weak.

- It is for this reason that SSL 3.0 implementations cannot be validated under FIPS 140-2

Security of TLS

TLS has a variety of security measures:

- Protection against a downgrade of the protocol to a previous (less secure) version or a weaker cipher suite.

- Numbering subsequent Application records with a sequence number and using this sequence number in the message authentication codes (MACs).

- Using a message digest enhanced with a key (so only a key-holder can check the MAC). The HMAC construction used by most TLS cipher suites is specified in RFC 2104 (SSL 3.0 used a different hash-based MAC).

- The message that ends the handshake ("Finished") sends a hash of all the exchanged handshake messages seen by both parties.

- The pseudorandom function splits the input data in half and processes each one with a different hashing algorithm (MD5 and SHA-1), then XORs them together to create the MAC. This provides protection even if one of these algorithms is found to be vulnerable.

Attacks against TLS/SSL

- **Renegotiation attack**
  - it allows an attacker who can hijack an https connection to splice their own requests into the beginning of the conversation the client has with the web server.
  - The attacker can't actually decrypt the client-server communication, so it is different from a typical man-in-the-middle attack. A short-term fix is for web servers to stop allowing renegotiation, which typically will not require other changes unless client certificate authentication is used.
  - To fix the vulnerability, a renegotiation indication extension was proposed for TLS.

Attacks against TLS/SSL

- Version rollback attacks
- BEAST attack (Browser Exploit Against SSL/TLS) using a Java applet to violate same origin policy constraints, for a long-known cipher block chaining (CBC) vulnerability in TLS 1.0

  Practical exploits had not been previously demonstrated for this vulnerability, which was originally in 2002.

  The vulnerability of the attack had been fixed with TLS 1.1 in 2006
- **RC4 attacks**

  In spite of existing attacks on RC4 that break it, the cipher suites based on RC4 in SSL and TLS were considered secure because of how the cipher was used in these protocols. In 2011 RC4 suite was actually recommended as a work around for the BEAST attack. But, newly statistical biases in RC4 key table were discovered to recover parts of plaintext with large number of TLS encryptions.

Attacks against TLS/SSL

- **Truncation attack** Published in July 2013, the attack causes web services such as Gmail and Hotmail to display a page that informs the user that they have successfully signed-out, while ensuring that the user's browser maintains authorization with the service, allowing an attacker with subsequent access to the browser to access and take over control of the user's logged-in account.
- Excellent presentation titled **“Cryptographic Analysis of TLS”** given by Kenny Paterson for TLS Security (in FOSAD 2013):

  http://www.sti.uniurb.it/events/fosad13/slides/paterson-fosad13.pdf

  See Slide 167 for some discussion of the use of TLS

  (“Most TLS implementations now patched against BEAST; Many TLS implementations patched against Lucky 13 ; No simple TLS patch for RC4 attack; Disable TLS compression to prevent CRIME;

  We need TLS 1.2!”)

Attacks against TLS/SSL

- **Apple's SSL/TLS bug (22 Feb 2014)**

  https://www.imperialviolet.org/2014/02/22/applebug.html

  http://www.imore.com/understanding-apples-ssl-tls-bug

  “ The result of this code is that an attacker on the same network as you could perform a man-in-the-middle attack where they fake a certificate keychain to a secure site, like your bank. You can't trust any secured connections in affected version of iOS and OS X. Everybody should update their iOS devices and Apple TVs if they haven't already.”

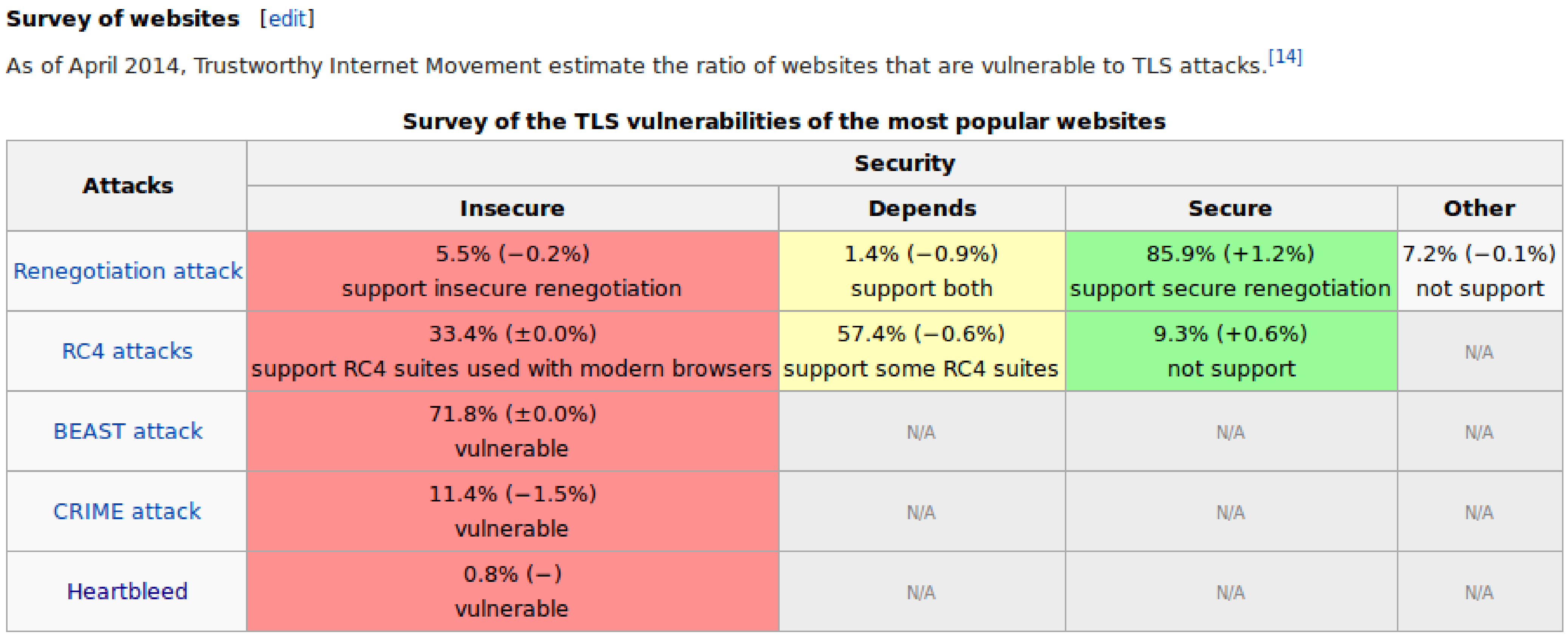
Attacks against TLS/SSL

- **Heartbleed Bug**

  The Heartbleed bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).
- The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

Attacks against TLS/SSL

http://en.wikipedia.org/wiki/Transport_Layer_Security#Security



Survey of websites [edit]

As of April 2014, Trustworthy Internet Movement estimate the ratio of websites that are vulnerable to TLS attacks.[14]

Survey of the TLS vulnerabilities of the most popular websites

Attacks	Security			
	Insecure	Depends	Secure	Other
Renegotiation attack	5.5% (−0.2%) support insecure renegotiation	1.4% (−0.9%) support both	85.9% (+1.2%) support secure renegotiation	7.2% (−0.1%) not support
RC4 attacks	33.4% (±0.0%) support RC4 suites used with modern browsers	57.4% (−0.6%) support some RC4 suites	9.3% (+0.6%) not support	N/A
BEAST attack	71.8% (±0.0%) vulnerable	N/A	N/A	N/A
CRIME attack	11.4% (−1.5%) vulnerable	N/A	N/A	N/A
Heartbleed	0.8% (−) vulnerable	N/A	N/A	N/A

Security of Javascript

- “ JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the web. Browser authors contain this risk using two restrictions.
- First, scripts run in a sandbox in which they can only perform web-related actions, not general-purpose programming tasks like creating files. Second, scripts are constrained by the same origin policy: scripts from one web site do not have access to information such as usernames, passwords, or cookies sent to another site. Most JavaScript-related security bugs are breaches of either the same origin policy or the sandbox.
- There are subsets of general JavaScript — ADsafe, Secure ECMA Script (SES) — that provide greater level of security, especially on code created by third parties ... “

  http://en.wikipedia.org/wiki/JavaScript#Security
- For the detailed discussion on Javascript security, see another presentation titled “Information flow control for the web” by Prof. Frank PIESSENS in FOSAD 2013: http://www.sti.uniurb.it/events/fosad13/slides/piessens-fosad13.pdf
- Recommended Video: JavaScript Security http://www.youtube.com/watch?v=hzAf5PY2Ws0

Example of Cryptographic Misuse

- “An Empirical Study of Cryptographic Misuse in Android Applications”

  www.cs.ucsb.edu/~chris/research/doc/ccs13_cryptolint.pdf

  – “ ..We develop program analysis techniques to automatically check programs on the Google Play marketplace, and find that 10,327 out of 11,748 applications that use cryptographic APIs 88% overall – make at least one mistake..”

  – **Some rules violated**:

    - Rule 1: Do not use ECB mode for encryption
    - Rule 2: Do not use a non-random IV for CBC encryp-tion
    - Rule 3: Do not use constant encryption keys