CSC 580
Cryptography and Computer Security

Putting the Pieces Together: Protocols
SSL/TLS and SSH

Chapter 17

April 17, 2018

First: Poll for April 19 Topic

Thursday, April 19 will be “Student’s Choice” Topic

Your interest, but not optional - yes, it will be on the exam

Possible Topics

- Authenticated data structures and the Bitcoin ledger
- Tor and anonymous communication
- Hardware security support: TPMs, secure boot, enclaves, ...
- Crypto gets weird: Zero-knowledge proofs, oblivious transfer, …
- Physics gets weird: Quantum computing and cryptography

Protocols

A ***protocol*** is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more computers. Networks have to follow these rules to successfully transmit data.

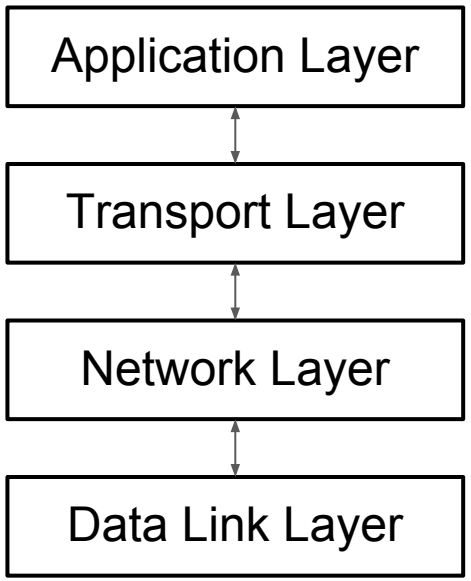
-- Techopedia

Protocols for secure communication use cryptographic operations that you learned about in this class to support higher-level security and communication objectives.

Basic Network Layers

Simplified network model (OSI model has 7 layers).

Each layer interacts with the one below it which has is less capable (less abstraction) than the one above.



"I want to retrieve the web page at http://www.google.com/"

"I want to connect to 74.125.136.105 and create a channel to send and receive bytes to a specific application/endpoint."

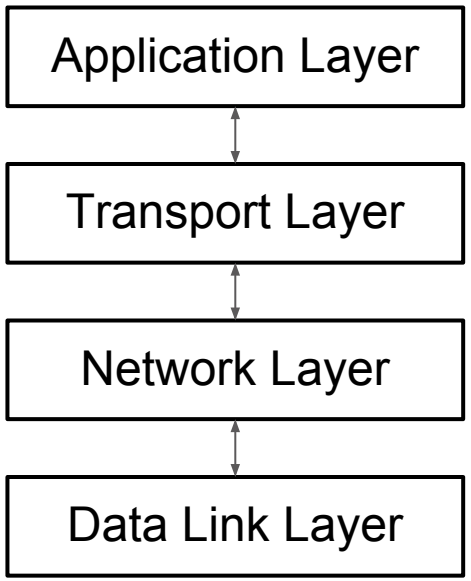
"I want to send this small packet of bytes to 74.125.136.105", which is somewhere else in the world...

"I want to send this small packet of bytes to this other computer that I am directly connected to."

Basic Network Layers

Simplified network model (OSI model has 7 layers).

Each layer interacts with the one below it which has is less capable (less abstraction) than the one above.



Examples: HTTP, SMTP, FTP, DNS, IMAP, SSH, ...

Examples: TCP, UDP, ICMP

Example (the only one that matters): IP

Examples: Ethernet (wired or wireless), FDDI, ATM, PPP, ...

Locations for security services

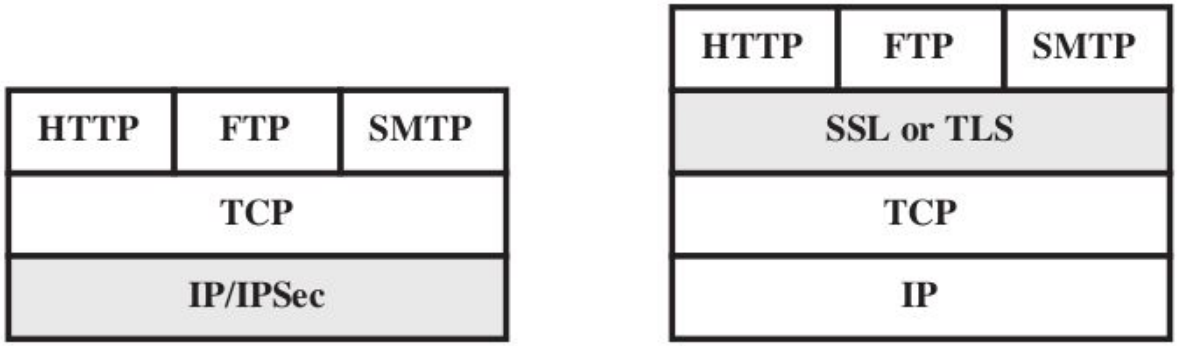


Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

Advantage of (a): Can protect all traffic (TCP, UDP, ICMP, …)

- Particularly good for VPNs

Advantage of (b): Understands “connections”

- Particularly good for protecting connections to specific applications

Locations for security services

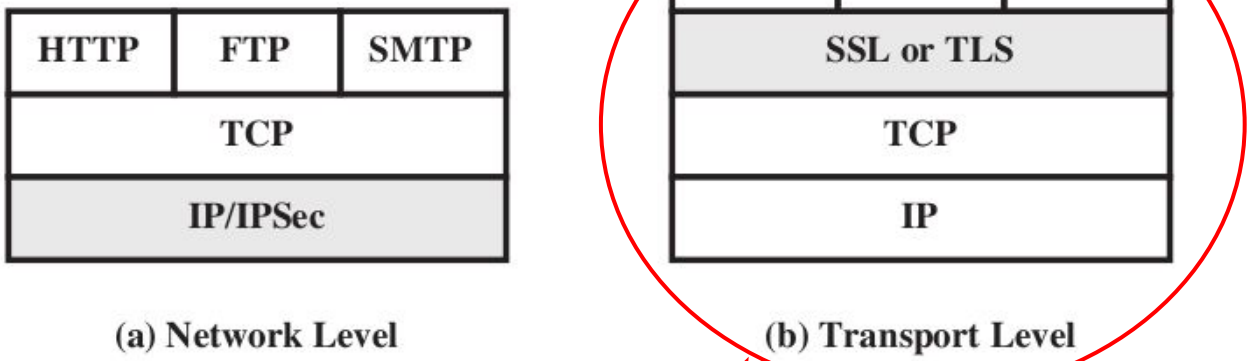


Figure 17.1 Relative L… Facilities in the TCP/I…

We will focus on connection-oriented communication, so focus on this model...

Advantage of (a): Can protect all traffic (TCP, UDP, ICMP, …)

- Particularly good for VPNs

Advantage of (b): Understands “connections”

- Particularly good for protecting connections to specific applications

SSL/TLS - History and Background

Generally associated with HTTPS, but protects many applications!

- Examples: IMAPS, POP3, LDAPS, SMTPS, …

SSL - “Secure Sockets Layer”

- Name: Traditional network programming API based on “sockets”
- Invented by Netscape to enable secure web browsing/e-commerce
  - Fundamental to Netscape’s business model
  - First release version was “Version 2.0” - released in 1995
  - Quickly followed by security-fixes in version 3.0 (1996)

TLS - “Transport Layer Security”:

- TLS 1.0 is SSL 3.1 (released 1999)
- Name change: Partly to avoid proprietary claims from Netscape
  - Also better reflects what it does (network layer rather than programming model)
- Latest standard version: TLS 1.2 (2008) [version 1.3 in draft form now]
  - Backward-compatible “protocol downgrade” has caused multiple vulnerabilities

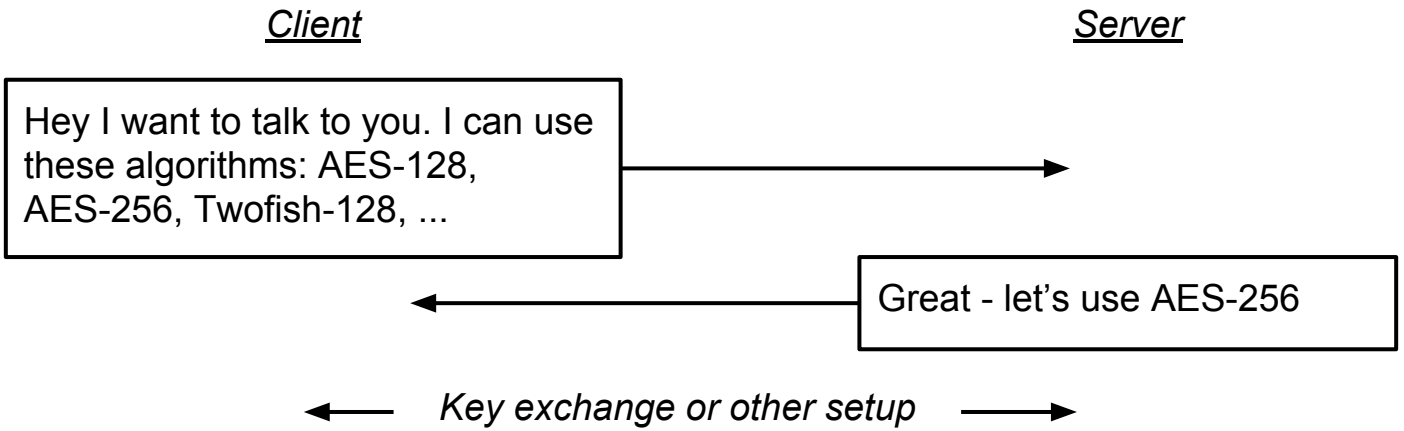
General Protocol Design

Handshakes and algorithm negotiation

Design Goals: Protocols should not hard-code specific algorithms, parameters, or key sizes. Need to be able to update dynamically!

But: Different implementations, different versions, different configs, must interoperate!

Solution: All protocols start with a “handshake phase” - idea:



Note: Real negotiation more complex: symmetric cipher, key exchange, integrity protections, ...

SSL and TLS Handshake

More detail about handshake:

Negotiates a “cipher suite”

- Combination of symmetric cipher, key exchange algorithm, signature scheme (site integrity), and MAC (indiv message integrity)

Notes:

- Possible for both sides to use certs!
- Change_cipher_spec is basically saying “go secure!”
- Note - initial handshake and cert messages are before “go secure”

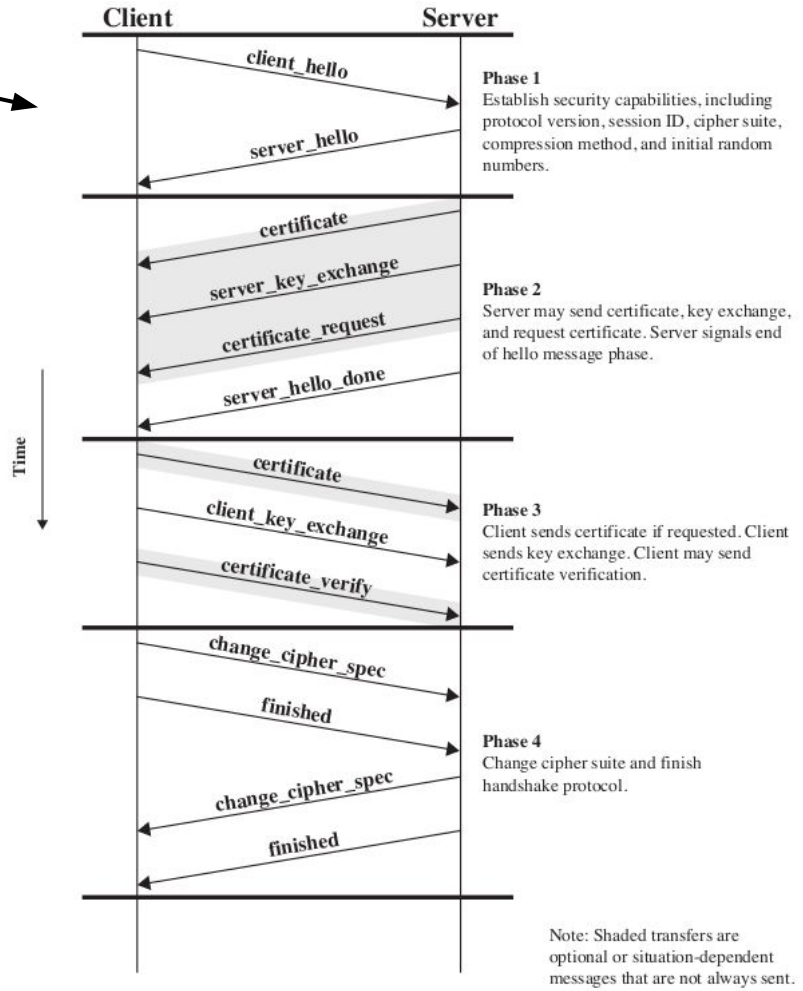
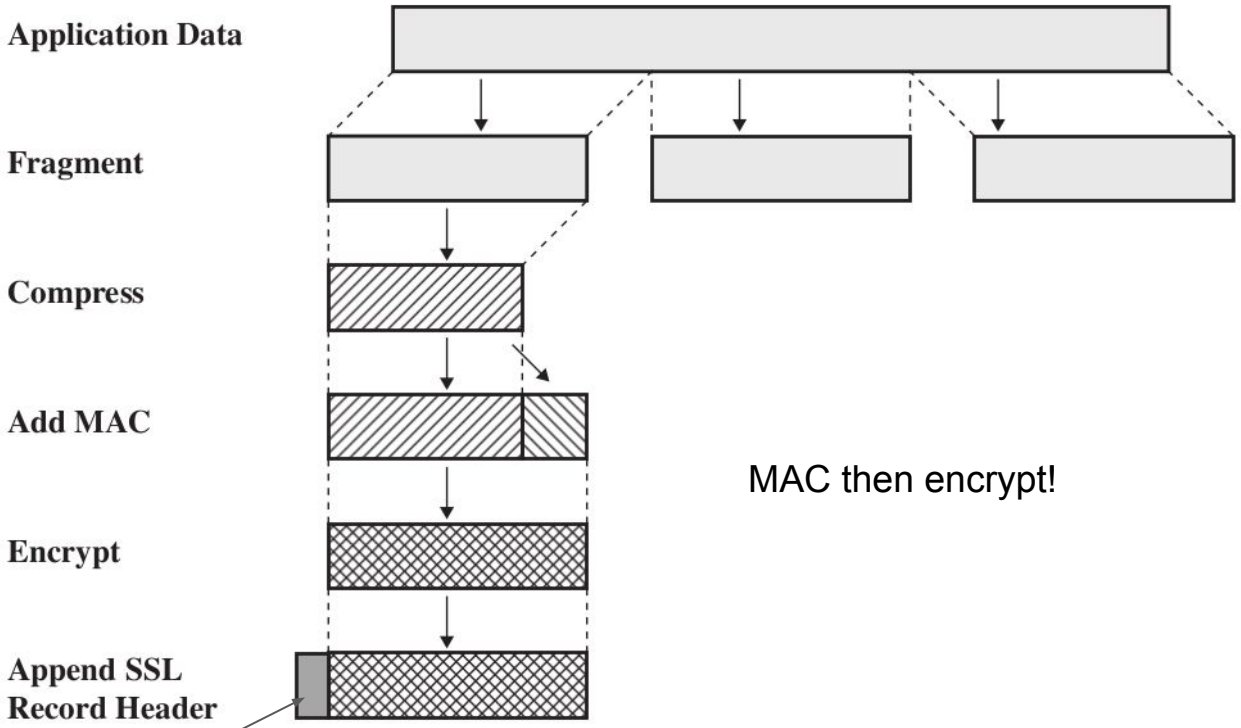


Figure 17.6 Handshake Protocol Action

TLS Record Layer

Every TLS packet is encapsulated in a “TLS Record”:



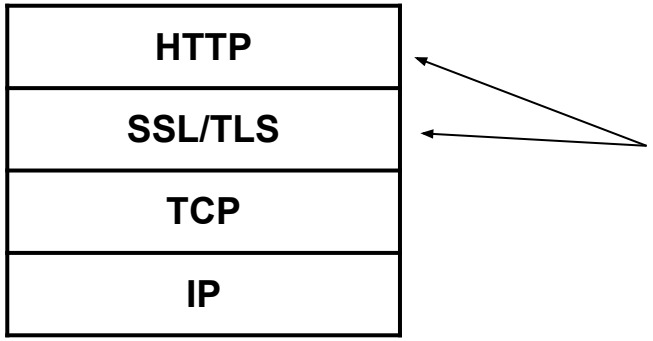
MAC then encrypt!

Figure 17.3 SSL Record Protocol Operation

Content type, version, length

SSL for the World Wide Web: HTTPS

HTTPS is not a different protocol - it's HTTP sitting on SSL/TLS:



For the most part these are independent and unaware of each other - SSL just provides a "secure pipe" for communication.

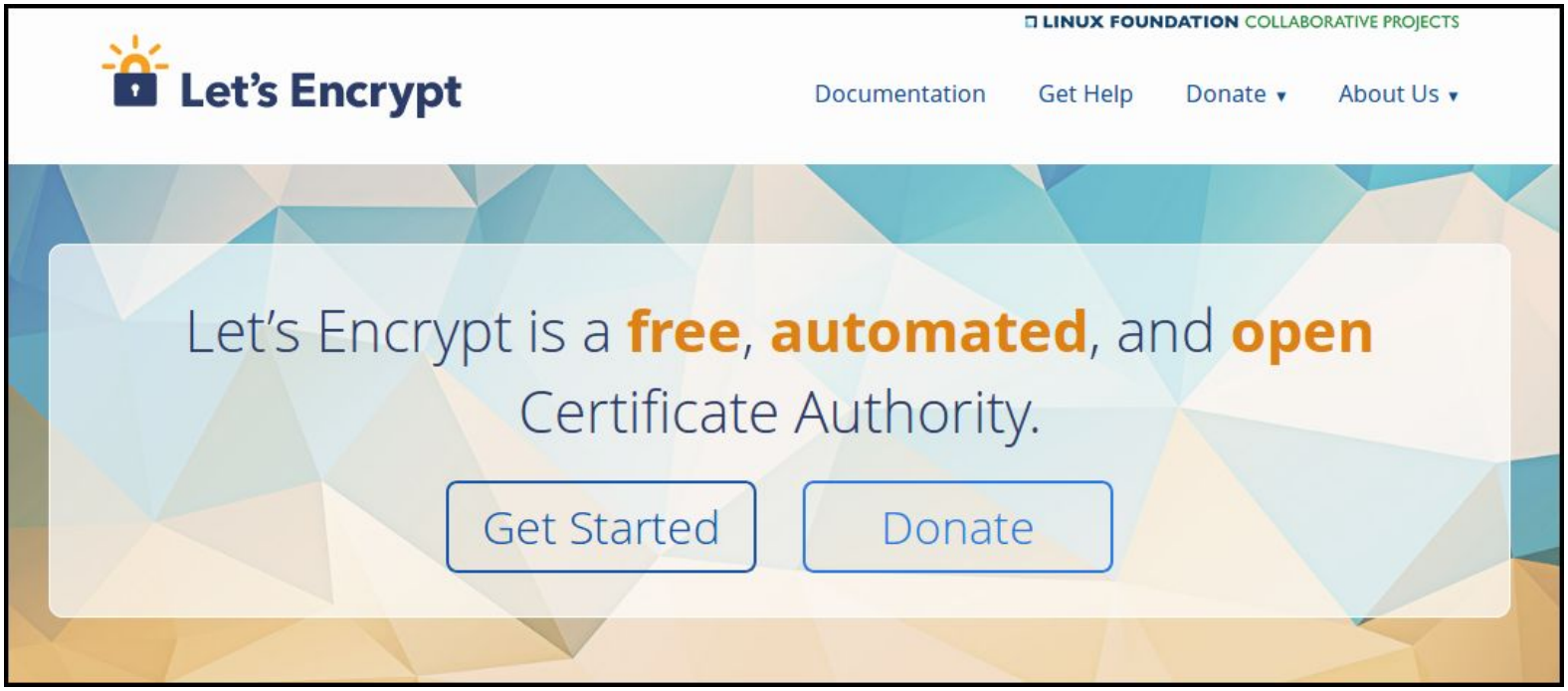
Some exceptions…

- Strict Transport Security: Server can include a Strict-Transport-Security line in the HTTP header to tell the browser that only HTTPS connections should be made
  - Browser should automatically convert http links to https
  - Refuse to connect if not secure (no downgrades, strict cert checks, etc.)
- Cookies: Secure flag
  - Normally cookies sent to any host in a given domain
  - Cookies with the secure flag will only be sent over https connections

Certificates for Web Sites

In the past: Buy a certificate - good for 1-3 years - could be expensive!

The new kid on the block – `letsencrypt.org`:



Goal: Promotes "encryption everywhere" - that's good!

Bad: Not as carefully vetted as commercial certs, and expires often (every 3 mos)

SSH - Purpose

Before 1995:

- Log in to work on a remote machine: `rlogin` or `telnet`
- Transfer files: `ftp`
- Remote command execution: `rsh`

> All used logins/passwords, and none were encrypted!
> Plaintext passwords flying all over the place!
>
> *Note: Kerberos (klogin) was an exception, but not widely used.*

SSH (secure shell) was a reaction to widespread sniffing attacks.

Originally used mostly for logins (`slogin`), but has evolved to provide:

- File transfers (`scp` and `sftp`)
- Remote command execution (`ssh`)
- Port forwarding for encrypting any TCP connection (“poor-man’s VPN”)

Also: Better, non-password-based authentication w/o Kerberos-style infrastructure

SSH - Handshake and Packet/Record

Same concepts as SSL - different details

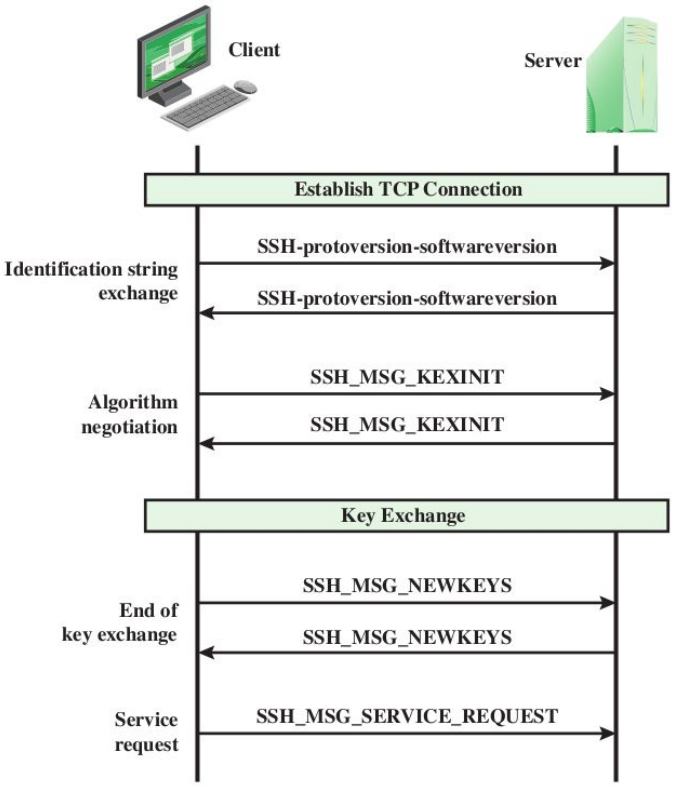
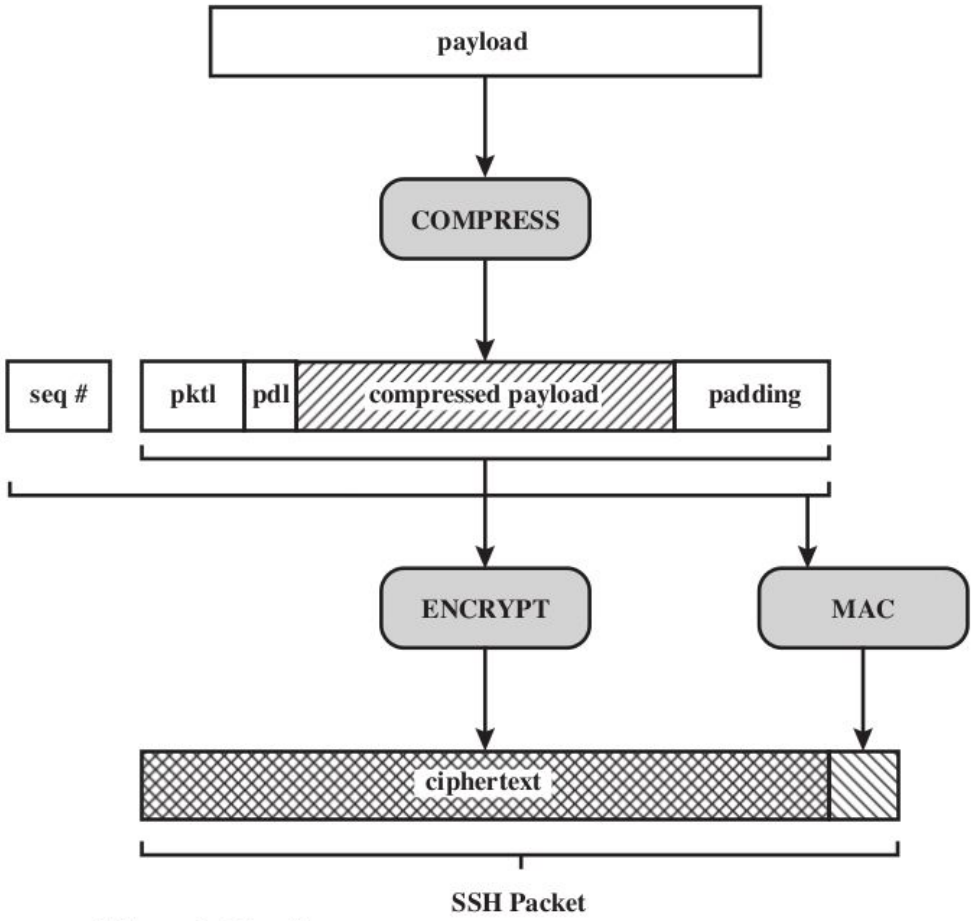


Figure 17.9 SSH Transport Layer Protocol Packet Exchanges



pktl = packet length
pdl = padding length

Figure 17.10 SSH Transport Layer Protocol Packet Formation

Demos!

In the remaining time: Demos looking into protocol packets