Network Application Security

Janaka Alawatugoda

Outline

- Basic Network Concepts
- Basic Web Authentication
- Application Protocols over Security Protocols
    - SSH Protocol
    - SSL/TLS Protocols
- Further Reading Guidance
- Technologies You Can Use

Basic Network Concepts

Basic Network Concepts

- Network: A number of computers and devices, connected by communications channels, and hardware and software to enable data exchange
- The connections can be:
  - Physical
    - Examples: cable, fibre optic
  - Wireless
    - Example: Radio signal, Infrared, Satellite

Basic Network Concepts

- Network types and information security:
  - Need to secure both
    - the devices and
    - the communication channels.
  - Easier to do this for a LAN with physical connections: can provide physical protection at the location to restrict access to authorised users
    - For wireless connections and larger networks, the communication may extend beyond physical boundaries of organization

Internet Communication

Computer networks can be thought of as being made up of several **layers**.

- Each computer connected to the network has these layers.
- For a message to be sent it must pass down all the layers on the sending computer.
- When a message is received it must go up all these layers on the receiving computer.

At each layer there is a **protocol** which defines message format and information

- Protocol: A set of rules governing the exchange of data between two or more entities.

Internet Engineering Task Force (IETF) Internet Protocol Suite

	Layer	Examples
Most defined by the Internet Engineering Task Force (IETF)	**Application**	web (HTTP, HTTPS) email (SMTP, POP3, IMAP) login (SSH, Telnet)
	Transport	connection-oriented (TCP) connectionless (UDP)
	Internet	addressing and routing: • IPv4, IPv6 control (ICMP) security (IPsec)
	Link	packet framing (Ethernet) physical connection • WLAN (WEP, WPA) • ADSL • GSM/3G

Link (a.k.a. network access) layer

The **link** or **network access layer** is the physical layer and is associated with computer hardware.

Computer networks can use a large number of connections and transmission media

- Telephone wires
- Ethernet (twisted pair) cables
- Optic Fibre cables
- Satellite communications
- Mobile phone networks
- Wireless networks
- Bluetooth

At this layer physical addresses identify network nodes

- Ethernet MAC address

Internet (a.k.a. network) layer

The **Internet layer** runs a low level protocol called the Internet Protocol (IP) (plus a few extra helpers, e.g. ICMP).

- IPv4 (1981), IPv6 (1996)

Host addressing and identification:

- Each host has a unique IP address:
  - IPv4, 32 bit, e.g., 131.181.118.220
  - IPv6, 128 bit, e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Packet routing:

- Organizations are assigned a range of IP addresses that they manage and assign to their computers.

Transport Layer

The **transport layer** establishes basic data channels for applications. It uses **ports** to distinguish between different applications on the same host.

TCP (Transmission Control Protocol)

- **connection-oriented** protocol
  - back-and-forth, ongoing connections
- **reliability**
  - large messages split into packets
  - in-order delivery of packets, recombined to large message
  - error checking
  - retransmission of lost packets
  - congestion control

UDP (User Datagram Protocol)

- **connectionless** protocol
  - send a packet, that's it
- **unreliable**
  - simple error checking
  - no retransmission of lost packets
  - used for streaming
    - audio, video, VOIP

Application layer

Application layer protocols are used by applications to provide user services over a network.

Each application protocol has unique message formats that are sent and received to achieve their tasks.

- HTTP (web)
- FTP (file transfer)
- SSH, Telnet (login)
- SMTP, POP3, IMAP (email)
- XMPP (chat)
- BitTorrent (I'm sure you know what this is used for)

Each application protocol requires the lower network layers (TCP, IP, Network Access) to communicate on the network.

Many use an intermediate protocol called SSL/TLS for encryption and authentication.

Client-Server on the Internet

- Each **application server** listens for messages on a particular port number. Common ports:
  - web servers: port 80 (HTTP), 443 (HTTPS)
  - login: port 22 (SSH), 23 (Telnet)
  - file transfer: port 20/21 (FTP), 22 (SFTP/SCP)
  - email servers: port 25 (SMTP), 220/993 (IMAP), 110 (POP)

- Clients identify the machine they want to connect to using an IP address.
- Clients identify the program they want to use using a port number.

requesting a webpage

Application Layer – Web browser

- Constructs the request in a specific format – HTTP request.
- Includes address information of the server (IP address and port number)

Transport Layer

- Breaks HTTP request into TCP packets (each with address info – IP address and port)

Internet Layer

- Routes TCP packets to destination IP address (packet switching)

Link Layer

- Packets are transmitted across wire, wireless, satellite etc depending on how computer is connected

receiving a webpage request

Application Layer – Web server

- Processes the HTTP request, maybe prepares a response

Transport Layer

- Assembles packets of request.
- Determines if there are any errors, and if so requests retransmission.
- Sends complete HTTP request to specified port

Internet Layer

- Collects packets for this IP address

Link Layer

- Receives packets across “wire”

Basic Web Authentication

Basic Web Authentication

- HTTP Basic authentication (BA) implementation is the simplest technique for enforcing access controls to web resources.

- It doesn't require cookies, session identifiers, or login pages.

- It uses standard fields in the HTTP header, obviating the need for handshakes.

- As the BA field has to be sent in the header of each HTTP request, the web browser needs to cache credentials (usernames and passwords) for a reasonable period of time.

Server-side

- When the server wants the user agent to authenticate itself towards the server, it must respond appropriately to unauthenticated requests.

- Unauthenticated requests should return a response whose header contains a HTTP 401 Unauthorized status and a WWW-Authenticate response header as shown below:

```
HTTP/1.1 401 Access Denied
WWW-Authenticate: Basic realm="My Server"
Content-Length: 0
```

- The word Basic in the WWW-Authenticate selects the authentication mechanism that the HTTP client must use to access the resource. The realm string can be set to any value to identify the secure area and may used by HTTP clients to manage passwords.

Client-side

- When the user agent wants to send the server authentication credentials it may use the Authorization field.
- The Authorization field is constructed as follows:
    - The username and password are combined into a string separated by a colon, e.g.: username:password
    - The resulting string is encoded using the RFC2045-MIME variant of Base64, except not limited to 76 char/line.
    - The authorization method and a space i.e. "Basic " is then put before the encoded string.

```
GET /securefiles/ HTTP/1.1
Host: www.httpwatch.com
Authorization: Basic aHR0cHdhdGNoOmY=
```

- The Authorization specifies the authentication mechanism (in this case Basic) followed by the username and password. Although, the string aHR0cHdhdGNoOmY= may look encrypted it is simply a base64 encoded version of <username>:<password>

https://www.httpwatch.com/httpgallery/authentication/

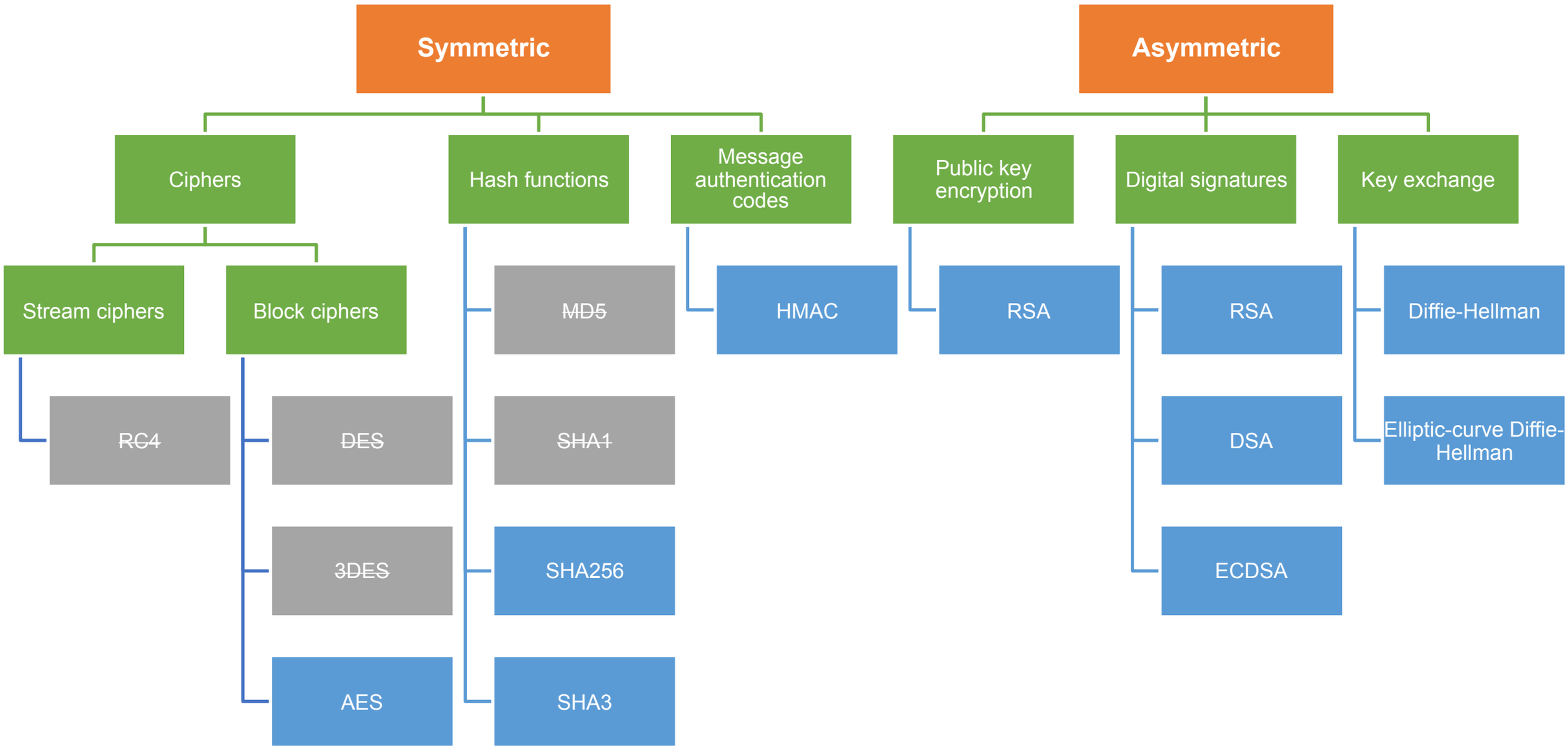
Application Protocols over Security Protocols

Network Security Protocols

- Network-related security protocols in common use include:
  - Secure Shell (SSH):
    Used for remote login, file transfer, and limited VPN service.
  - Transport Layer Security (TLS):
    Used extensively on the web and is often referred to in privacy policies as a means of providing confidential web connections.
  - IP Security (IPsec):
    Provides security services at the IP level and is used to provide Virtual Private Network (VPN) services.
  - WiFi security (WEP, WPA):
    Provides security services at the link layer for wireless communication

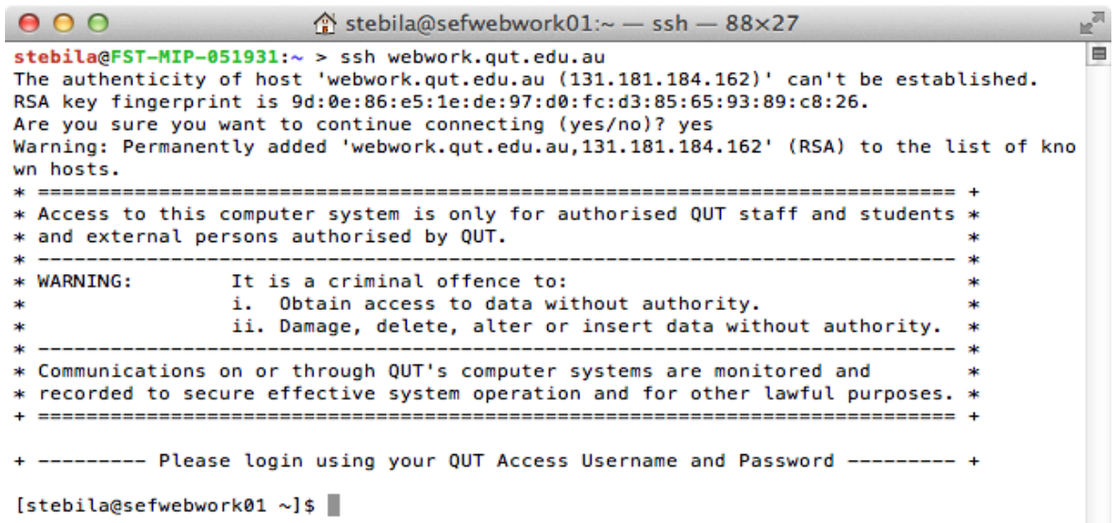
Layer	Examples
Application	web (HTTP, HTTPS) email (SMTP, POP3, IMAP) login (SSH, Telnet)
Transport	connection-oriented (TCP) connectionless (UDP)
Internet	addressing and routing: • IPv4, IPv6 control (ICMP) security (IPsec)
Link	packet framing (Ethernet) physical connection • WLAN (WEP, WPA) • ADSL • GSM/3G

Underlying Cryptographic Building Blocks



Secure Shell Protocol

SSH (Secure Shell) Protocol



- SSH used for secure remote access (like telnet, but secure)
- Provides public key authentication of servers and clients and encrypted communication
- Specified in RFCs by the IETF

Use of SSH

- Primarily used as an application itself (remote login)
- Occasionally used as a “poor man’s VPN”

Layer	Examples
Application	web (HTTP, HTTPS) email (SMTP, POP3, IMAP) login (SSH, Telnet)
Transport	connection-oriented (TCP) connectionless (UDP)
Internet	addressing and routing: • IPv4, IPv6 control (ICMP) security (IPsec)
Link	packet framing (Ethernet) physical connection • WLAN (WEP, WPA) • ADSL • GSM/3G

SSH Security Services

- Message Confidentiality.
  - Protects against unauthorised data disclosure.
  - Accomplished by the use of encryption mechanisms.
- Message Integrity.
  - SSH can determine if data has been changed (intentionally or unintentionally) during transit.
  - Integrity of data can be assured by using a message authentication code (MAC).
- Message Replay Protection.
  - The same data is not delivered multiple times.
- Peer Authentication.
  - Server to client authentication based on public keys
  - Client to server authentication based on passwords or public keys
  - Ensures that network traffic is being sent from the expected party.

Client Authentication in SSH

- **SSH** (Secure Shell) is often used for remote command-line access in Unix and Mac OS X.
- It supports public key authentication.
  - A security-conscious SSH installation would support **only** public key authentication and disable password-based authentication.
- Each account can have multiple associated public keys.
  - Multiple users can login to a single account without having to be told the password for that account. Easy to revoke one user's access to that account.
  - One user could have a different key from each local computer (laptop, desktop, …); if one of local computer is lost/compromised, easy to revoke its access.
- Users can associate the same key with multiple accounts.
  - Yields a form of single sign-on.
  - Users can protect their private key using a password.

Secure Sockets Layer/ Transport Layer Security Protocols (SSL/TLS)

SSL/TLS

SSL: Secure Sockets Layer

- Proposed by Netscape
  - SSLv2: 1995
  - SSLv3: 1996

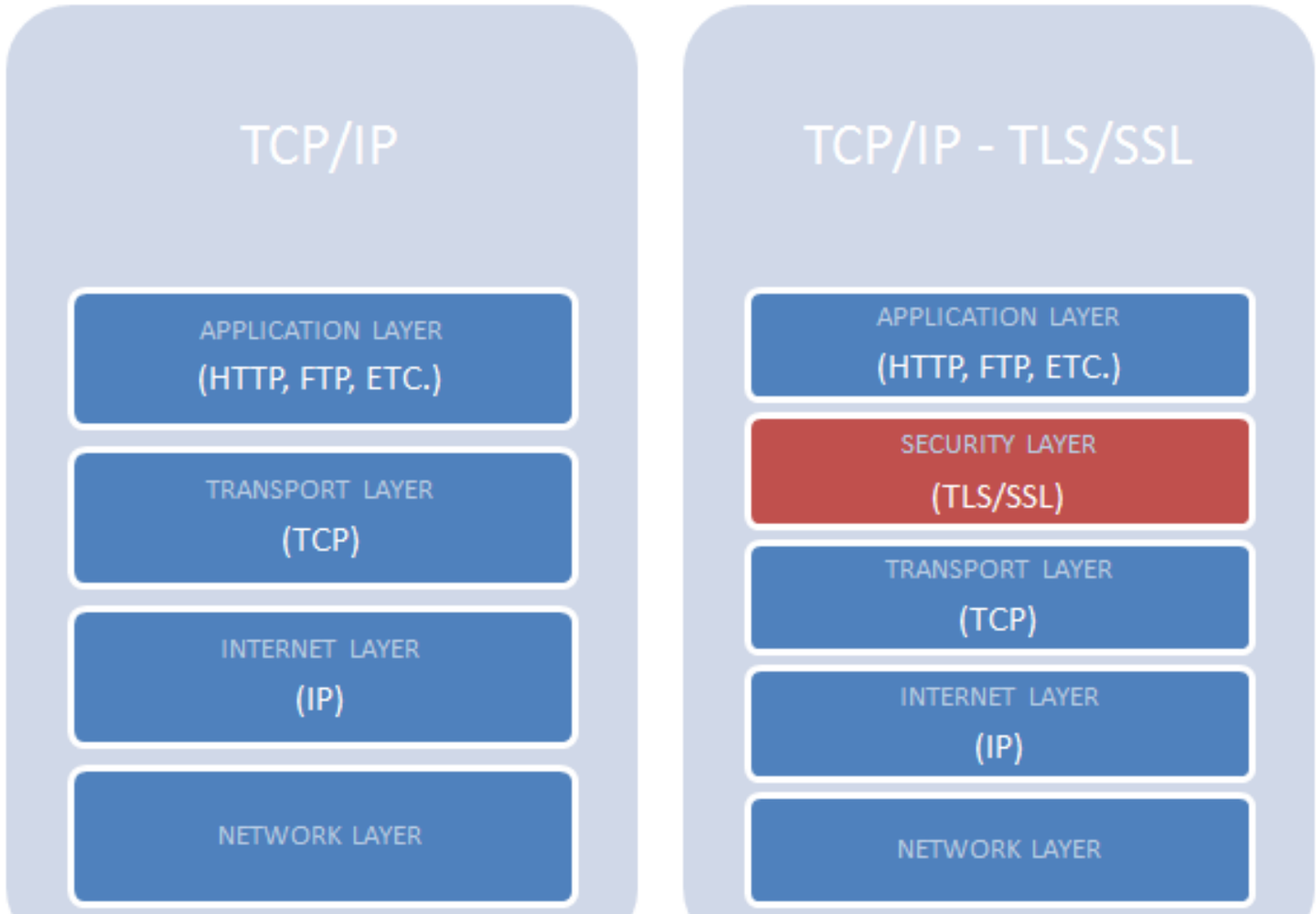
TLS: Transport Layer Security

- IETF Standardization of SSL
  - TLSv1.0 = SSLv3: 1999
  - TLSv1.1: 2006
  - TLSv1.2: 2008
  - TLSv1.3: in development

TLS is the new name for SSL. Namely, SSL protocol got to version 3.0; TLS 1.0 is "SSL 3.1". We sometimes say "SSL/TLS".

HTTPS is HTTP-within-SSL/TLS. SSL (TLS) establishes a secured, bidirectional tunnel for arbitrary binary data between two hosts.

HTTPS: HTTP (Hypertext Transport Protocol) over SSL



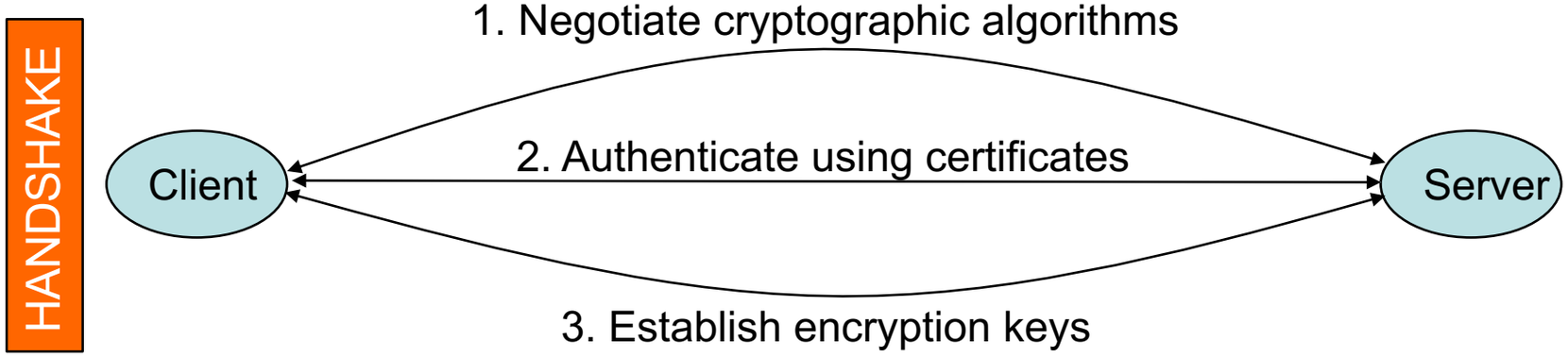
SSL/TLS

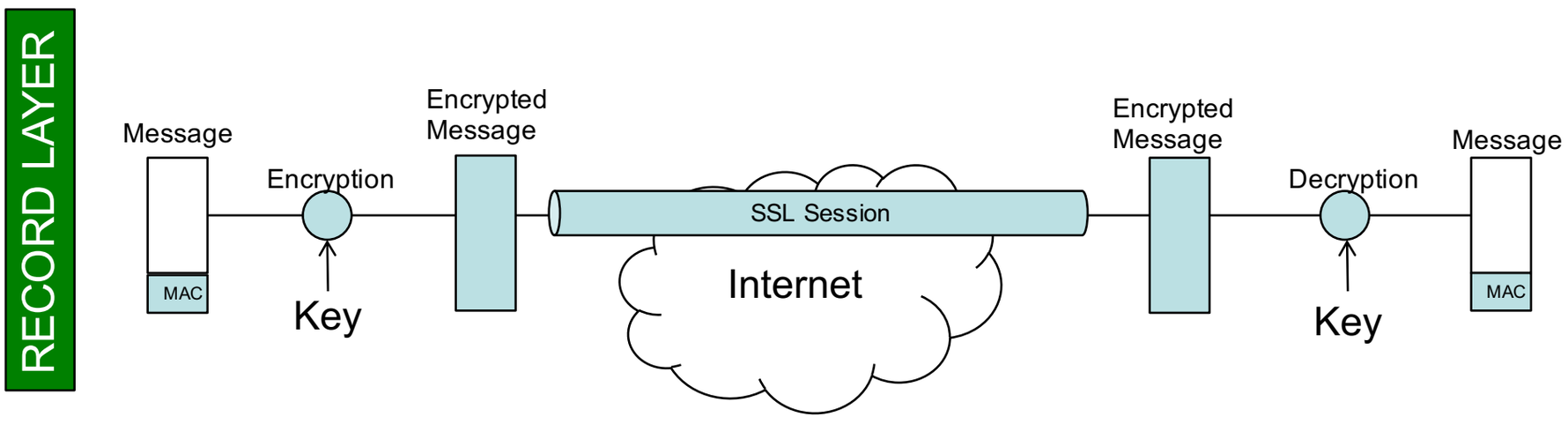
- Transport Layer Security (TLS) is a cryptographic protocol that operates at the transport layer (actually, above the transport layer)
  - Say, on top of Transmission Control Protocol (TCP)
- Uses encryption and PKI to provide protection for network communication protocols operating at higher levels
  - For example, to protect HTTP communications
  - Use X.509 certificates to obtain public keys
  - Can be used to provide confidentiality and authentication

- Encryption and authentication layer added to the protocol stack between TCP and applications.

TLS and HTTP

- TLS can be used to provide protection for HTTP communications:
  - Port 443 is reserved for HTTP over TLS
  - HTTPS is the name of the URL scheme used with this port.
  - http://www.develop.com implies the use of standard HTTP using port 80.
  - https://www.develop.com implies the use of HTTP over TLS using port 443.





Certificates

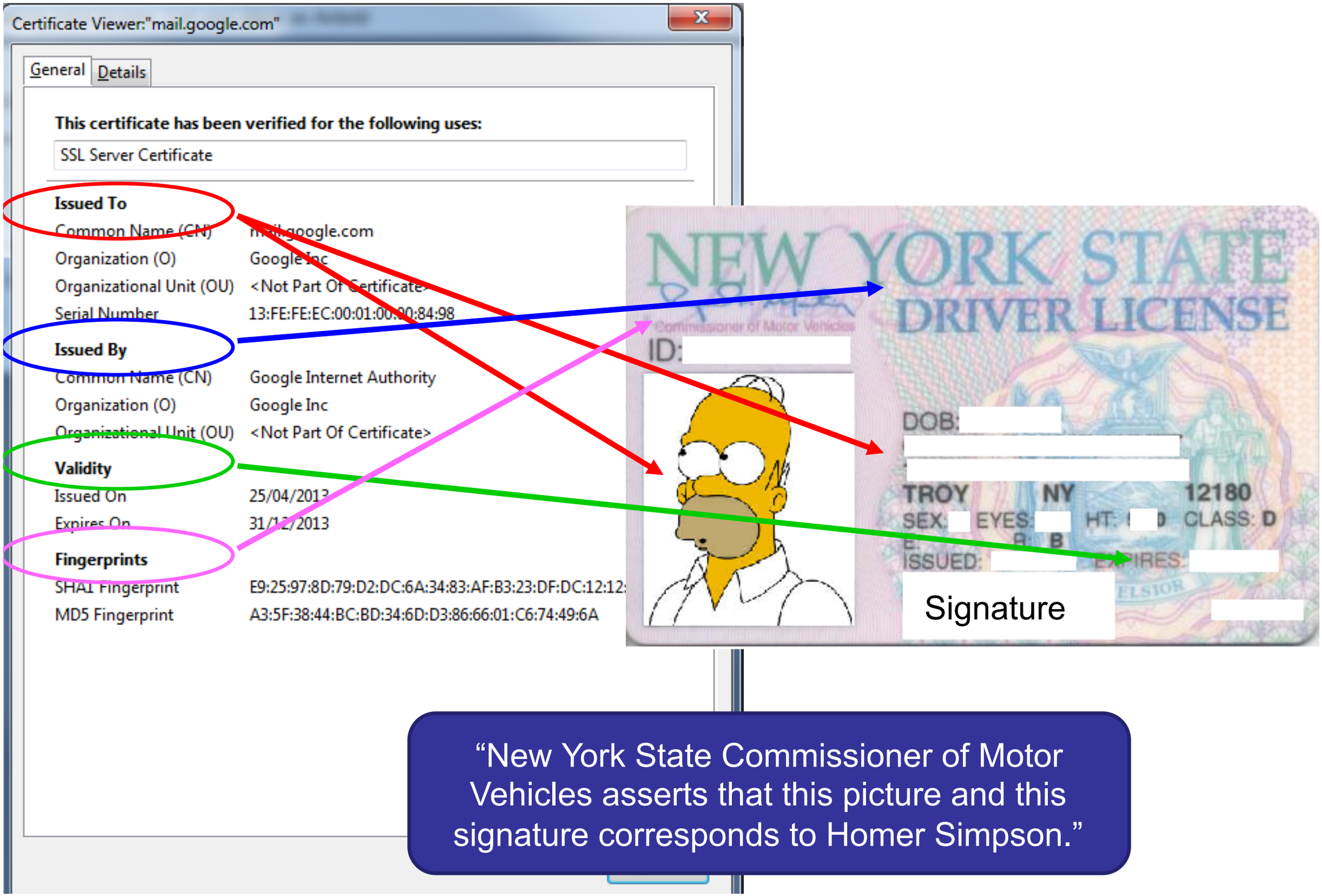
A **certificate** is an assertion by a trusted third party that a particular public key belongs to a particular entity.

The **certificate authority** generates a certificate by

1. Obtaining the user's public key by some trusted mechanism.
2. Verifying that the user really is who she says she is.
3. Signing (using the certificate authority's public key) the user's public key and name.

This allows two parties who have never met to establish trust between them:

- Exchange certificates.
- Do authentication using digital signatures.
- If they each trust the certificate authority that signed the other party's certificate, they can now be certain who the other party is.


Certificate Viewer:"mail.google.com"
General
Details
This certificate has been verified for the following uses:
SSL Server Certificate
Issued To
Common Name (CN)
Organization (O)
Google Inc
Organizational Unit (OU)
<Not Part Of Certificate>
Serial Number
13:FE:FE:EC:00:01:00:00:84:98
Issued By
Common Name (CN)
Google Internet Authority
Organization (O)
Google Inc
Organizational Unit (OU)
<Not Part Of Certificate>
Validity
Issued On
25/04/2013
Expires On
31/12/2013
Fingerprints
SHA1 Fingerprint
E9:25:97:8D:79:D2:DC:6A:34:83:AF:B3:23:DF:DC:12:12:
MD5 Fingerprint
A3:5F:38:44:BC:BD:34:6D:D3:86:66:01:C6:74:49:6A
NEW YORK STATE
DRIVER LICENSE
ID:
DOB:
TROY
NY
12180
SEX:
EYES:
HT:
CLASS: D
ISSUED:
EXPIRES:
Signature
“New York State Commissioner of Motor Vehicles asserts that this picture and this signature corresponds to Homer Simpson.”

TLS: Handshake Protocol

- Provides one service for TLS connections.
  - Authentication (server-to-client):
    - Ensures that the connection really is with the server with the given domain name.

- Also establishes keys that will be used in the record protocol for additional security services.

TLS Handshake Example

Client (web browser)

Server (web server)

1. Hello! Here are the algorithms I know how to use: …

2. Hello! Okay, let's use this "ciphersuite": RSA signatures, ECDH key exchange, AES-CBC encryption, HMAC-SHA256 authentication.

3. Here's my certificate with my RSA public key. Here's a Diffie-Hellman key g^x. And here's a signature on my DH public key to prove it's mine.

4. Check that the certificate is valid. (Signed by a trusted CA, domain name of server matches subject common name in certificate, validity period okay.)

5. Check that signature on DH public key is okay.

Client (web browser)

Server (web server)

6. Here's my DH public key g^y.

7. Compute DH shared secret g^{xy}.

7. Compute DH shared secret g^{xy}.

8. Here's a MAC authentication of all messages we exchanged so far, using the DH shared secret.
This proves no one modified our handshake.

Notice shared secret g^{xy} is independent of long-term keys => forward secrecy

9. Check the MAC.

10. Here's a MAC authentication of all messages we exchanged so far, using the DH shared secret.

11. Check the MAC.

12. Handshake's done!

12. Handshake's done!

TLS Record layer

13. Start sending application data, encrypted using AES-CBC and authenticated using HMAC-SHA256, keys derived from DH shared secret.

TLS: Record Protocol

- Provides two services for TLS connections.
  - Message Confidentiality:
    - Ensure that the message contents cannot be read in transit.
    - The Handshake Protocol is used to establish a symmetric key to be used to encrypt SSL/TLS payloads in the record protocol.
  - Message Integrity:
    - Ensure that the receiver can detect if a message is modified in transmission.
    - The Handshake Protocol establishes a shared secret key used to construct a Message Authentication Code.

TLS Record Protocol Operation

Application Data

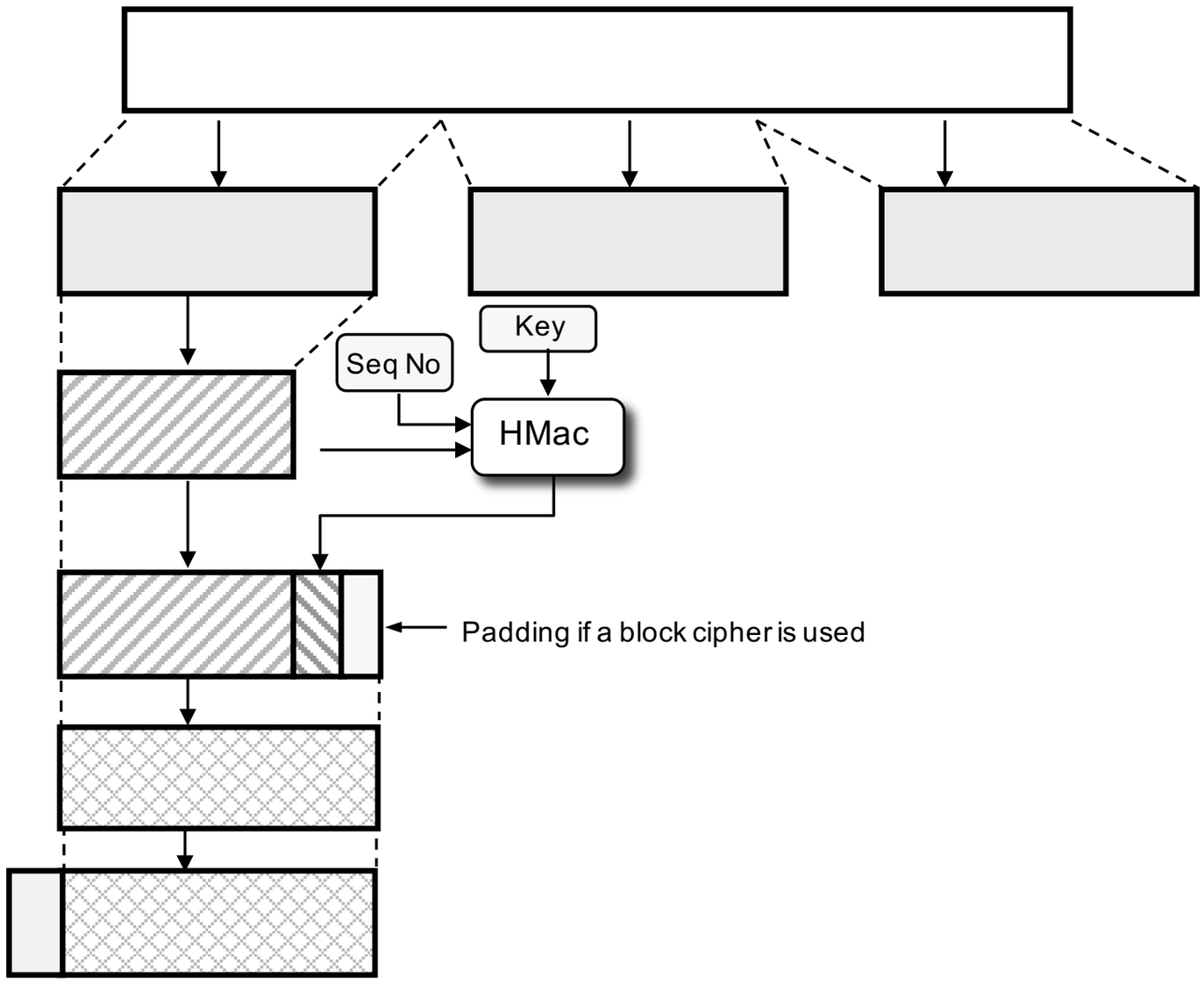
Fragment

Compress

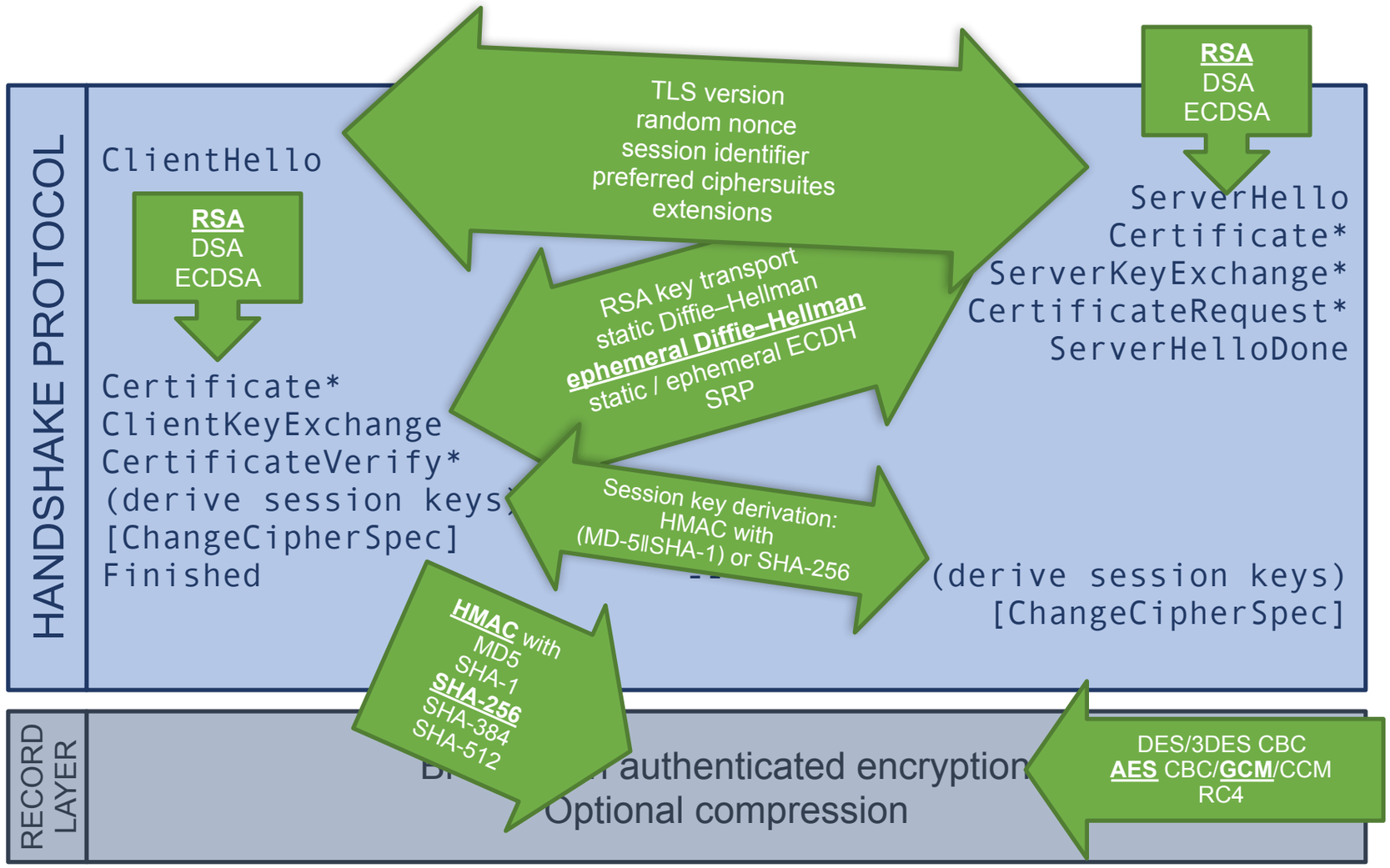
MAC

Encryption

Prepend Header



Structure of TLS



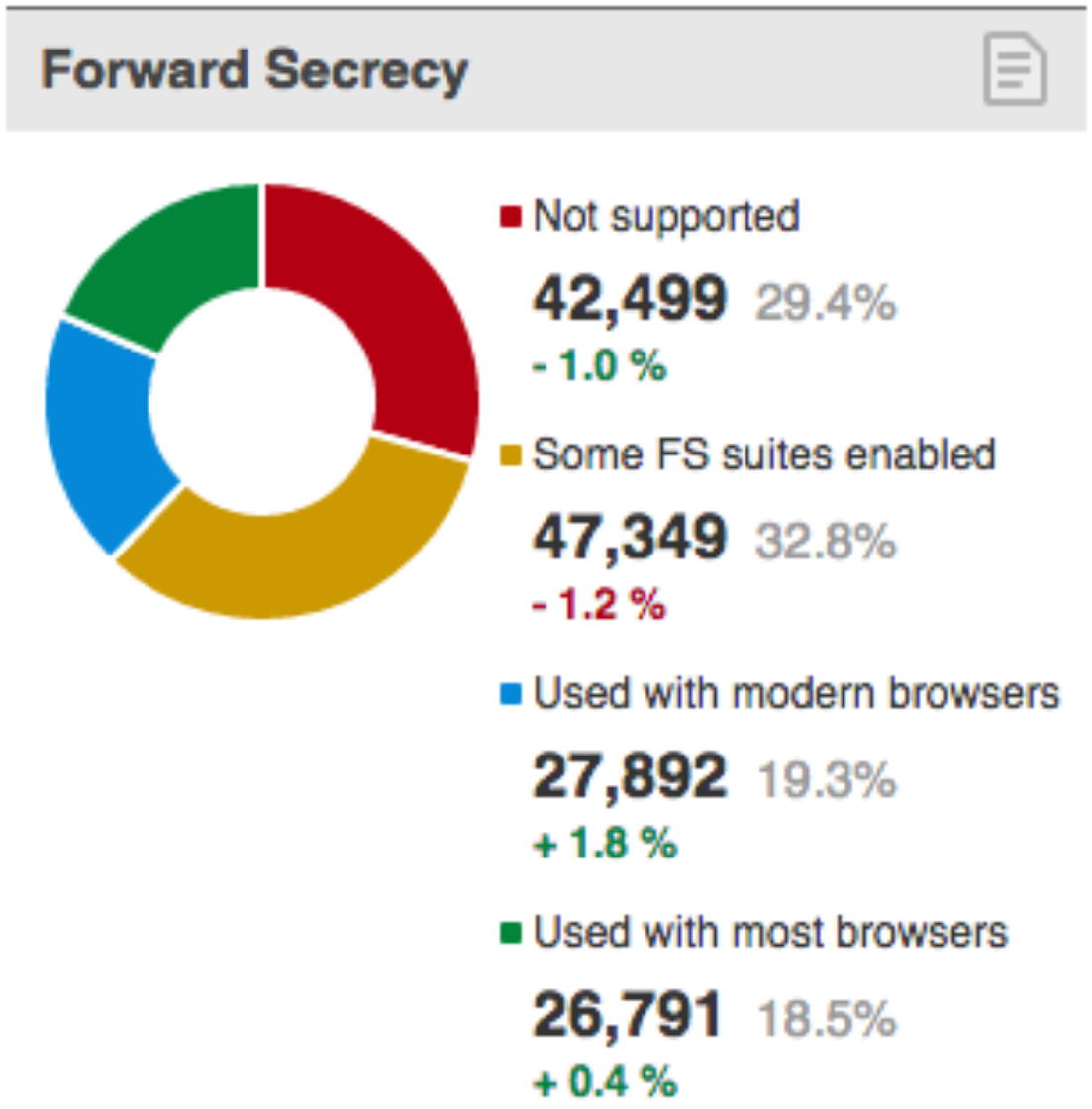
TLS Security Considerations

- Trust and digital certificates:
  - TLS uses public keys – provided in digital certificates
  - Certificates should be verified – requires tracing certificate pathways
  - Web browsers come with pre-configured lists of root certificates but users can add or remove root CAs
- One-way or mutual authentication?
  - Authentication is usually of server to client only, not mutual
  - Users usually do not have client certificates
  - Typically, authentication of users is not performed in handshake
  - Instead, password authentication over server-authenticated HTTPS channel

(Perfect) Forward Secrecy

- An adversary who later learns the server's long-term private key shouldn't be able to read previous transmissions

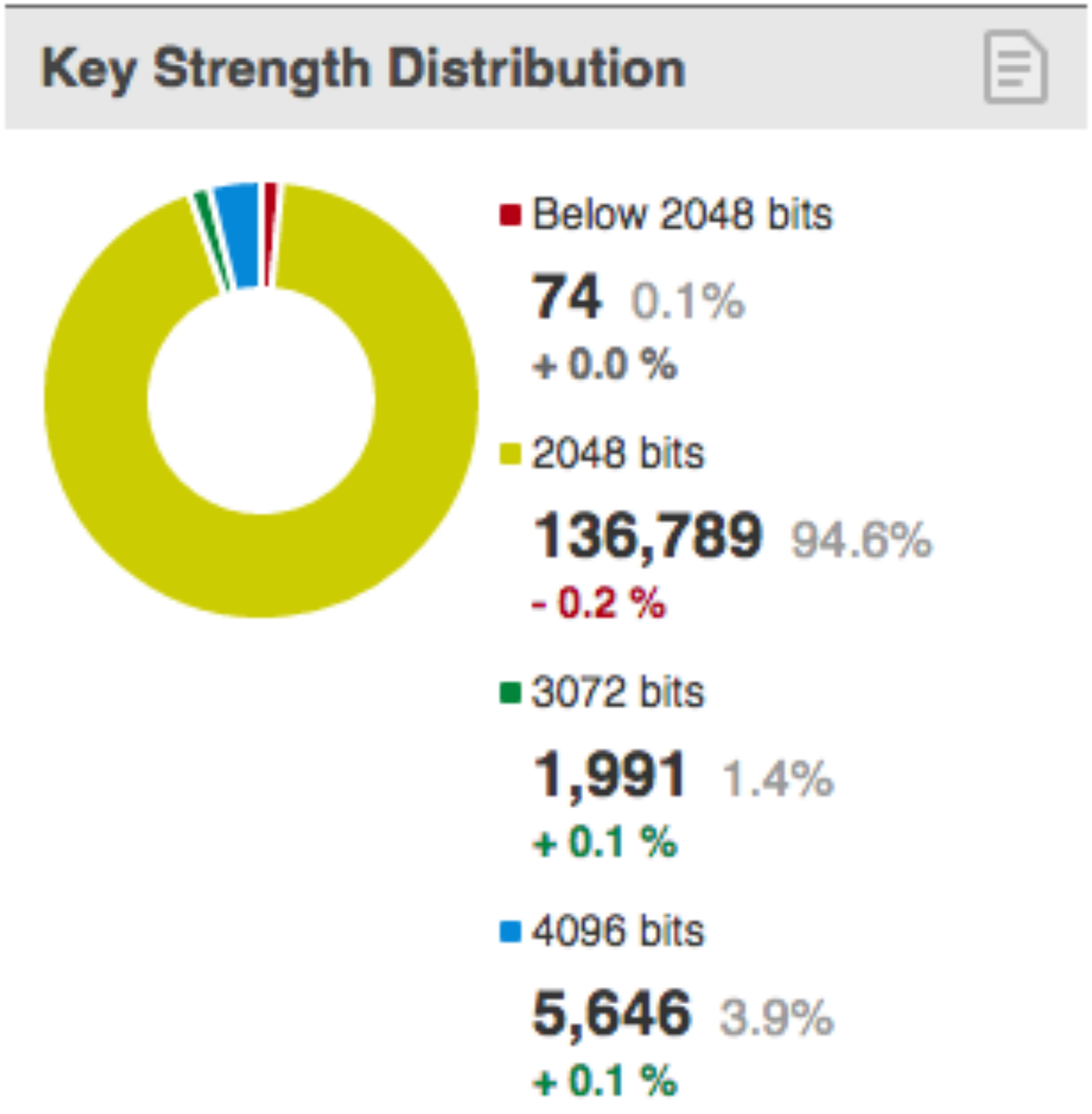
- RSA key transport: no PFS
- signed Diffie–Hellman: PFS



https://www.trustworthyinternet.org/ssl-pulse/ September 3, 2015

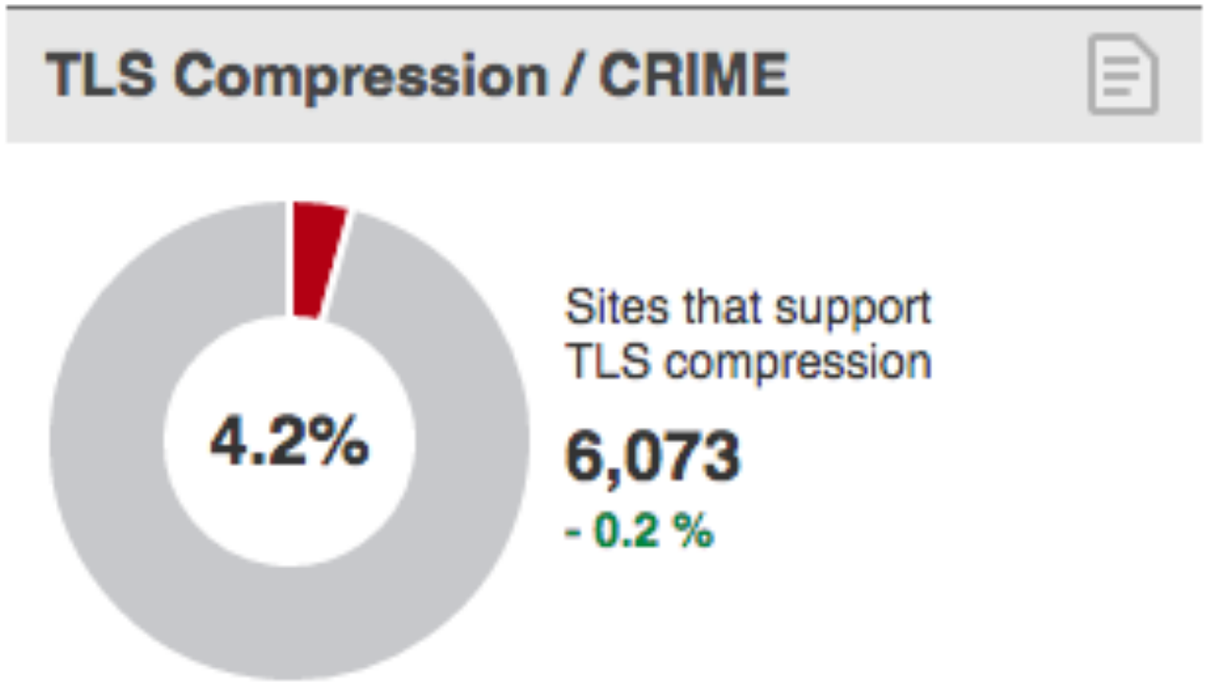
Deprecated Feature: small RSA Keys

- Researchers can break 768-bit RSA keys
- Likely that government agencies can break 1024-bit RSA keys
  - Common up until ~2010
- Servers should use at least 2048-bit RSA keys or 256-bit elliptic curve keys



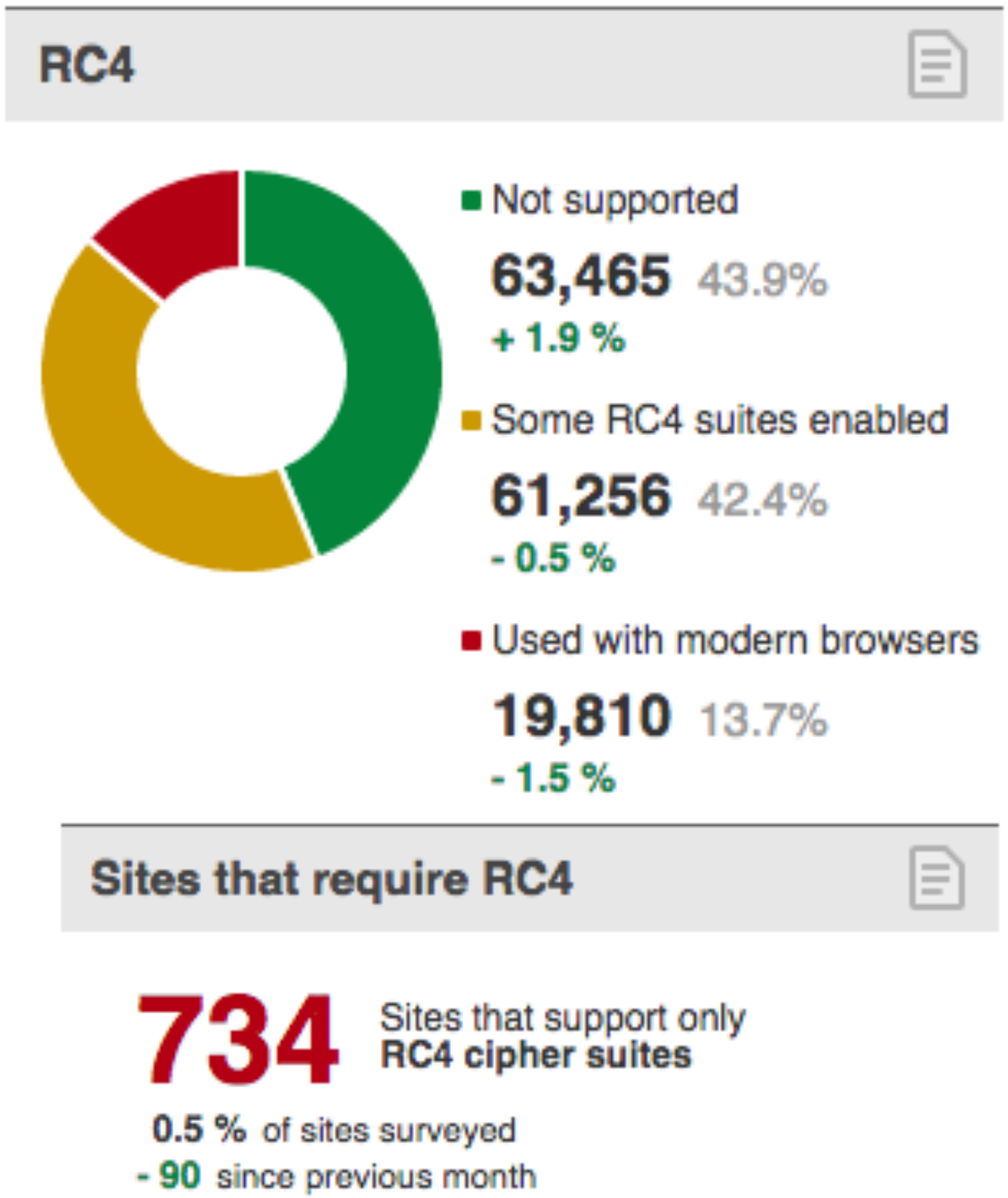
Deprecated Feature: Compression

- TLS supports optional compression
- Message broken up into chunks, each chunk compressed then encrypted
- Size of ciphertext
  => amount of compression
  => leaks plaintext info
- “CRIME” attack, Sept. 2012
- Fix: disable compression



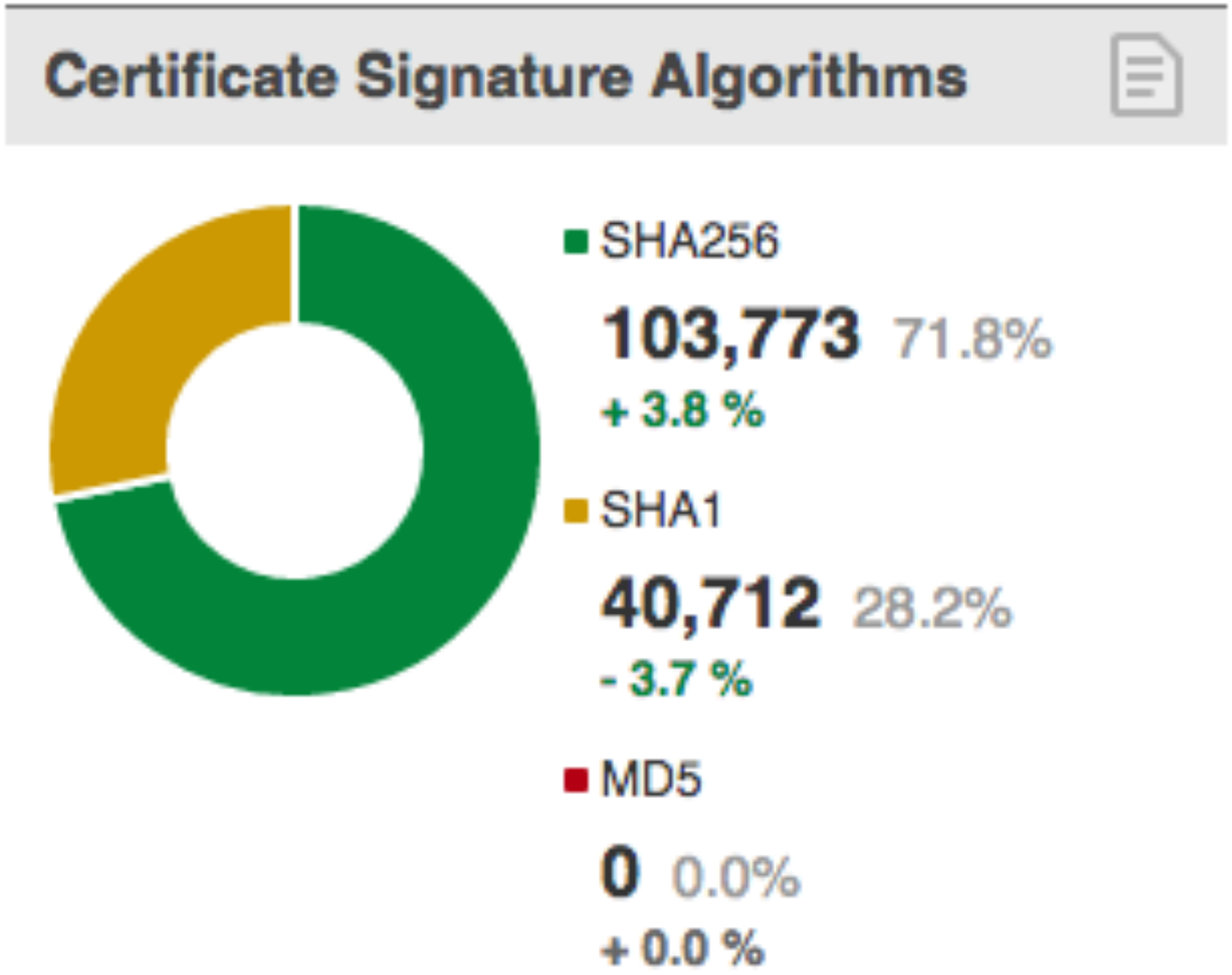
Deprecated Feature: RC4

- RC4 is a stream cipher
  - Keystream output by RC4 should look random
  - But actually there are small (but detectable) biases that make it look non-random
- RC4 should be disabled



Deprecated Feature: SHA-1 Certificates

- SHA-1 hash function is used in certificates
  - Digital signatures are only secure if the hash function is "collision resistant"
- Researchers are getting very close to finding collisions in SHA-1
- Most browsers will reject SHA-1 certs in Jan. 2017 and start warning in Jan. 2016
- Servers should stop using SHA-1 certs by Dec. 2015



TLS version1.3: The Next Generation

- Currently under development at the IETF

- Primary goals:
  - remove ciphersuites without forward secrecy
  - remove obsolete / deprecated algorithms
  - provide low-latency mode with fewer round trips
  - encrypt more of the handshake to improve privacy

- Deployed in 2017+

Cross-Site Request Forgery (CSRF) Attacks

- Cross-site request forgery, also known as one-click attack or session riding.

- CSRF is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

- CSRF attacks specifically target state-changing requests, not theft of data.

- With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing.

Using TLS on a Web Server

1. Need to generate a key pair and get a certificate
2. Need to configure web server to use certificate
3. Need to configure web pages to load all resources over TLS (images, CSS, Javascript, external scripts)

Getting TLS Certificate

1. Generate an RSA public key / private key pair
2. Generate a “certificate signing request” (CSR) that contains your public key and the name of the domain you want the certificate for
3. Send the CSR to a certificate authority (CA), who will perform some validation
   - Domain validation: prove you own the domain name by receiving an email at webmaster@domain.com or putting up a special code at www.domain.com/something
     - Free domain-validated certs available from https://www.startssl.com/ and soon https://letsencrypt.org/
   - Extended validation: prove you are the legal organization corresponding to that domain, and are authorized to act on behalf of that organization
     - EV certs show a green bar in the browser address field
4. CA signs your public key into a certificate

Configuring TLS Server

1. Need to install your certificate, along with the (intermediate) certificates of your CA so that browsers can verify the chain from your certificate to a trusted root
2. Need to say which directories should be protected by TLS
3. Need to say which ciphersuites to prefer, enable forward secrecy, etc.
   - https://wiki.mozilla.org/Security/Server_Side_TLS

Configuring Web Pages to use TLS

- Web pages load resources (images, stylesheets, JS)
  - If the page is loaded over TLS but a Javascript is not, then an attacker can just replace the Javascript and have their own code executed
  - "Mixed content"
- Complicated on big sites with lots of external scripts, ads, trackers, …

Which pages should be protected by TLS?

- Just login page?
- Login page plus all pages after you've logged in?
- All pages?

Google starting to use HTTPS as a ranking signal

- http://googlewebmastercentral.blogspot.com.au/2014/08/https-as-ranking-signal.html

Making Sure HTTPS is Used

- Once you've set up HTTPS on your server, how do you ensure people actually use it?
    - If a user types in an address or follows a link, it might first request the non-HTTP version

As a user:
Install HTTPS Everywhere extension

- For well-known sites, automatically rewrites all HTTP URLs to HTTPS URLs
- https://www.eff.org/https-everywhere

As a web server administrator:
Enable HTTP Strict Transport Security

Sets a flag that tells the browser: next time you come to this web site, automatically use HTTPS

What TLS Certificates Can/Cannot Do?

They do…

- Provide assurance that the server is who it says it is, assuming the certification authority is honest
  - Different levels of assurance depending on different levels of validation
    - Domain validation
    - Extended validation

They don't…

- Prevent web site vulnerabilities
  - E.g., cross-site scripting, SQL injection, …
- Indicate that the website has been validated by a third-party auditor

What Can a Malicious CA do?

- The web server never gives its private key to the CA, so the CA cannot decrypt traffic between the web server and clients
- A malicious CA could impersonate any server by issuing a fraudulent certificate
  - Hard to detect: no way of knowing if a cert was honestly issued or not
  - A new initiative called “certificate transparency” tries to monitor all certificates issued to watch out for fraud

Further Reading Guidance

Further Reading Guidance

- Metadata privacy using TOR  anonymity network
- Email Security and Using PGP (Pretty Good Privacy)

- Instant Messaging Privacy and using OTR (Off-the-record)

Technologies you can use

Technologies you can use

- In your browser: HTTPS Everywhere
- For your network connection: Tor or VPNs
- For email: PGP
- For instant messaging:
  - iMessage
  - better: OTR, TextSecure/Signal, …
- On your web server: TLS

EFF Surveillance Self-Defense https://ssd.eff.org/

Lecture slides are adapted from lecture slides of Dr. Ernest Foo and Dr. Douglas Stebila

https://www.youtube.com/watch?v=5X3reWHabXc

https://www.youtube.com/watch?v=TfO-Mi4uP7c