Defending Computer Networks
Lecture 23: Transport Layer Security

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- Apologies again for last Thursday
- HW 4 due tomorrow
- No class this Thursday (Thanksgiving)
- Regular lecture next Tuesday (12/2)
- Quiz 3, final class that Thursday (12/4)

SECRET MALWARE IN EUROPEAN UNION ATTACK LINKED TO U.S. AND BRITISH INTELLIGENCE

Complex malware known as Regin is the suspected technology behind sophisticated cyberattacks conducted by U.S. and British intelligence agencies on the European Union and a Belgian telecommunications company, according to security industry sources and technical analysis conducted by *The Intercept*.

Regin was found on infected internal computer systems and email servers at Belgacom, a partly state-owned Belgian phone and internet provider, following reports last year that the company was targeted in a top-secret surveillance operation carried out by British spy agency Government Communications Headquarters, industry sources told *The Intercept*.

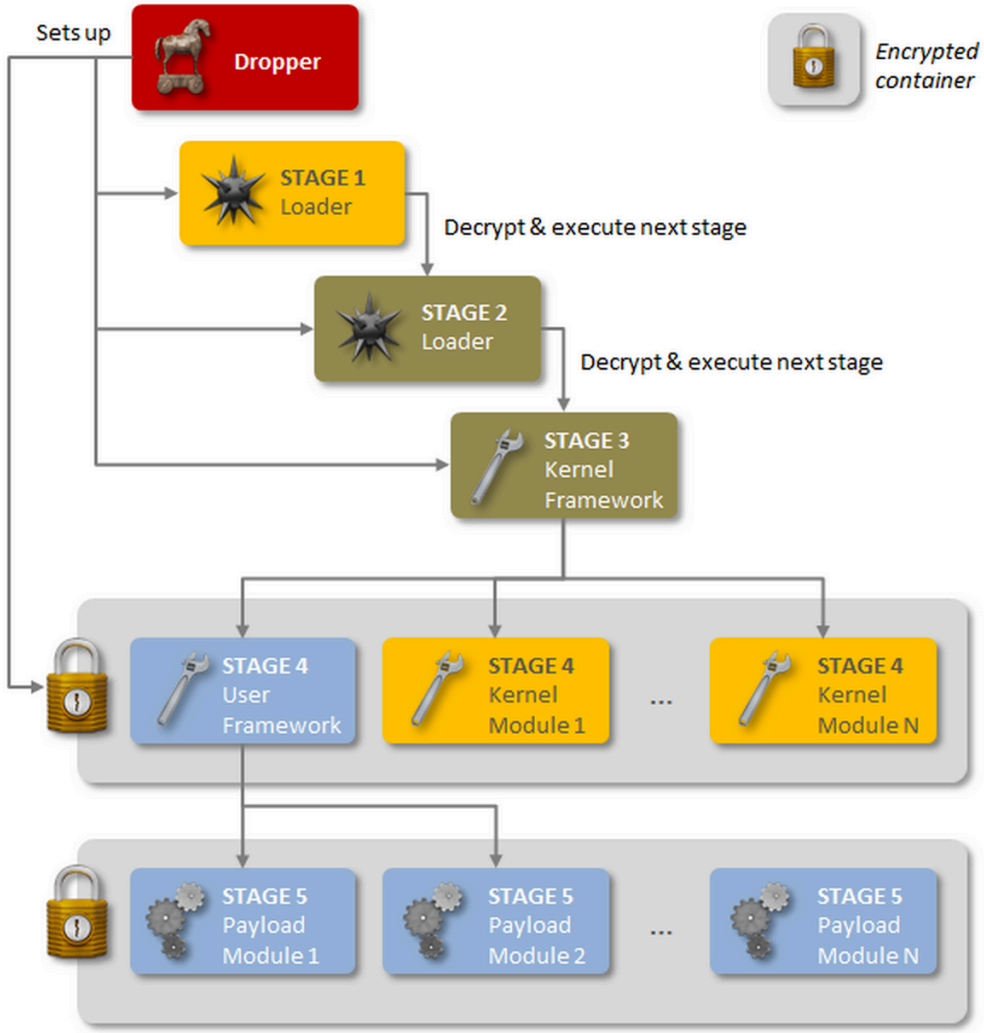
The malware, which steals data from infected systems and disguises itself as legitimate Microsoft software, has also been identified on the same European Union computer systems that were targeted for surveillance by the National Security Agency.

The hacking operations against Belgacom and the European Union were first revealed last year through documents leaked by NSA whistleblower Edward Snowden. The specific malware used in the attacks has never been disclosed, however.

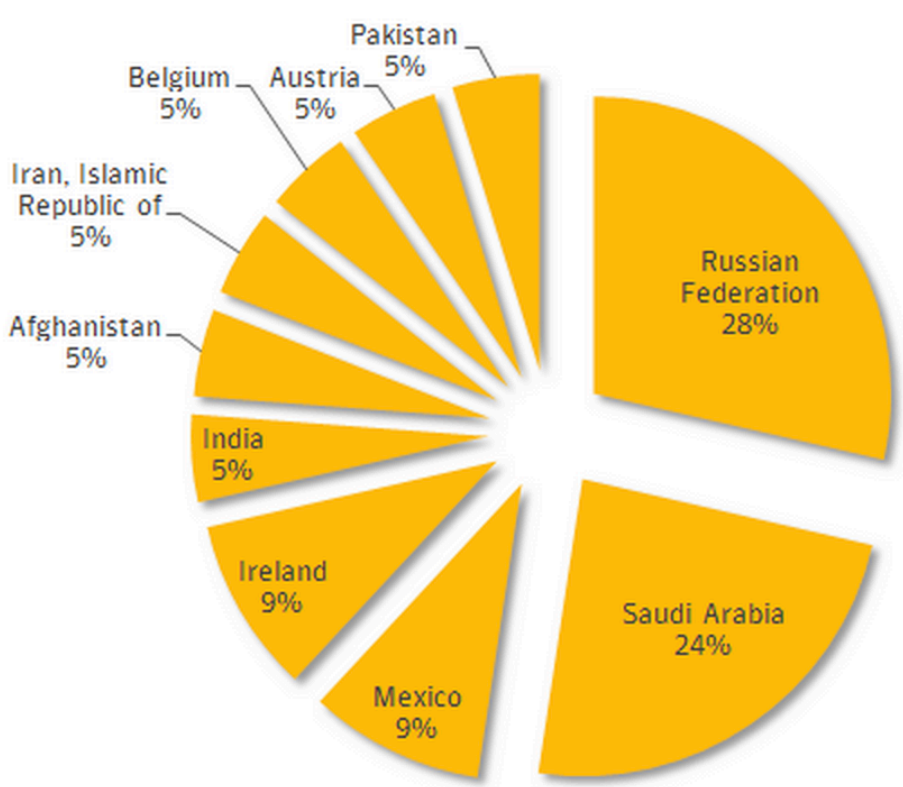
https://firstlook.org/theintercept/2014/11/24/secret-regin-malware-belgacom-nsa-gchq/

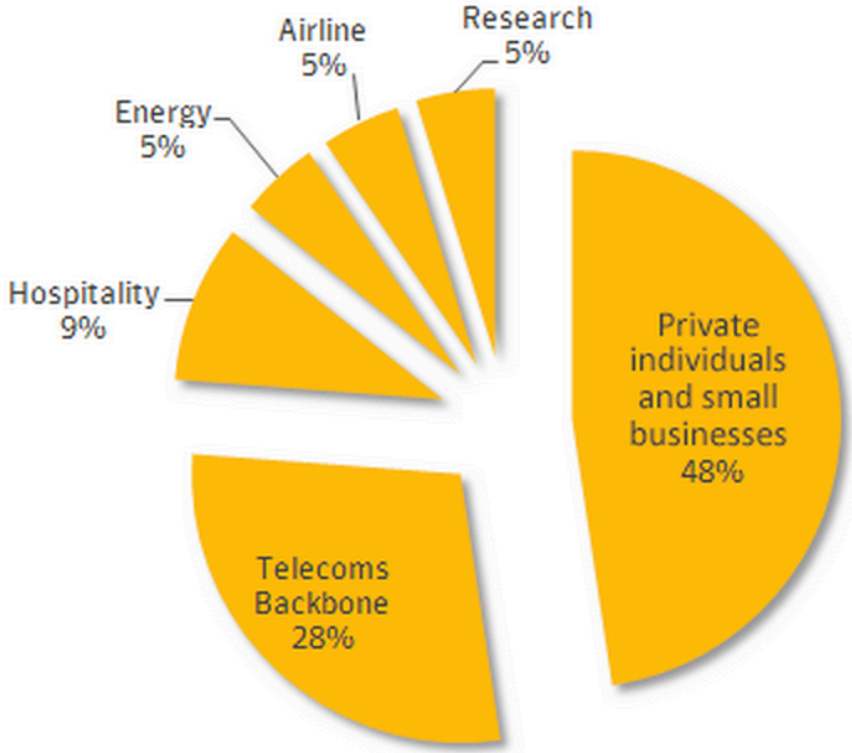
Regin: Top-tier espionage tool enables stealthy surveillance

An advanced spying tool, Regin displays a degree of technical competence rarely seen and has been used in spying operations against governments, infrastructure operators, businesses, researchers, and private individuals.



http://www.symantec.com/connect/blogs/regin-top-tier-espionage-tool-enables-stealthy-surveillance


Pakistan
5%
Belgium
5%
Austria
5%
Iran, Islamic
Republic of
5%
Afghanistan
5%
India
5%
Ireland
9%
Mexico
9%
Saudi Arabia
24%
Russian
Federation
28%


Airline
5%
Research
5%
Energy
5%
Hospitality
9%
Telecoms
Backbone
28%
Private
individuals
and small
businesses
48%

04h	DWORD	File offset
08h	DWORD	Offset to first sector holding the file data
0Ch	BYTE[taglen]	File tag

Table 5. The container's sectors

Offset	Type	Description
00h	DWORD	Next sector offset, or 0
04h	BYTE[sectsize-4]	Data

Regin's C&C operations are extensive. These backchannel operations are bidirectional, which means either the attackers can initiate communications with compromised computers on the border network or the compromised computers can initiate communications with the attacker. Furthermore, compromised computers can serve as a proxy for other infections and command and control can also happen in a peer-to-peer fashion. All communications are strongly encrypted and can happen in a two-stage fashion where the attacker may contact a compromised computer using one channel to instruct it to begin communications on a different channel. Four transport protocols are available for C&C:

- ICMP: Payload information can be encoded and embedded in lieu of legitimate ICMP/ping data. The string 'shit' is scattered in the packet for data validation. In addition, CRC checks use the seed '31337'.
- UDP: Raw UDP payload
- TCP: Raw TCP payload
- HTTP: Payload information can be encoded and embedded within cookie data under the names SESSID, SMSWAP, TW, WINKER, TIMESET, LASTVISIT, AST.NET_SessionId, PHPSESSID, or phpAds_d. This information can be combined with another cookie for validation under the names USERIDTK, UID, GRID, UID=PREF=ID, TM, __utma, LM, TMARK, VERSION, or CURRENT

The C&C operations are undertaken by various modules, including major groups C373h, 19h, 9, as well as Stage 5 payloads, such as C375h and 1Bh.

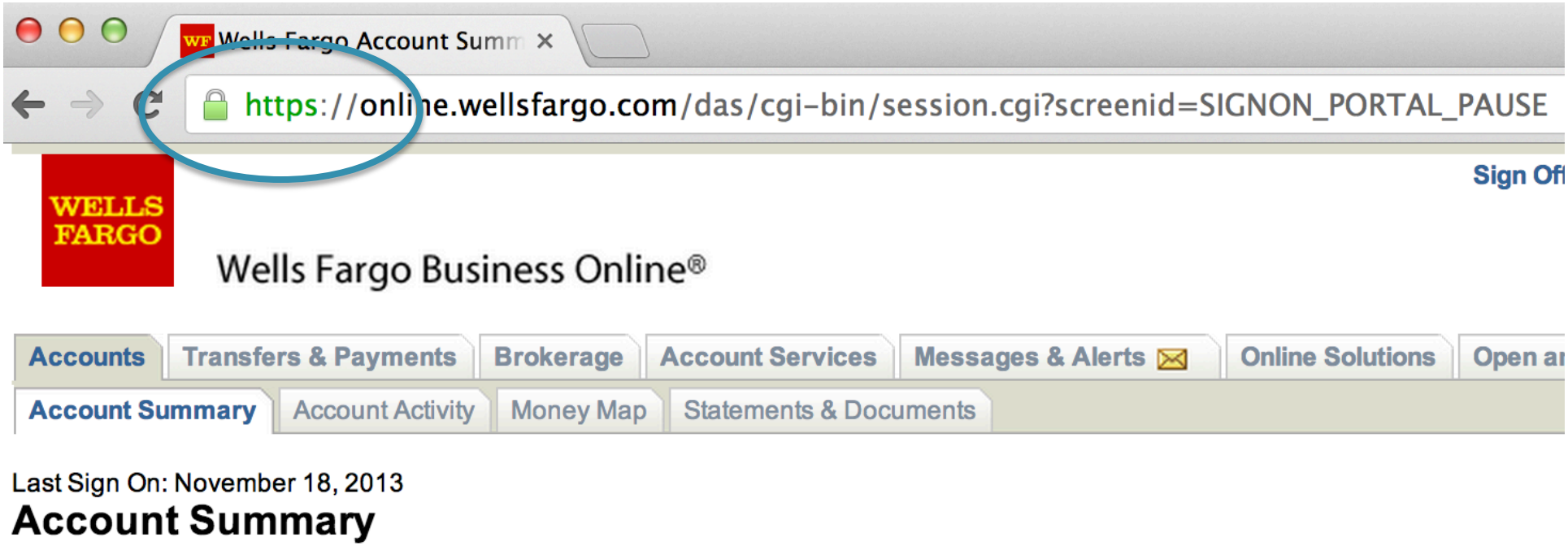
Main Focus of Today

- Start on TLS (formerly known as SSL)
  - And various attendant technologies

Main Goals of TLS/SSL

- Transport Layer Security/Secure Sockets Layer
- Original goal was secure HTTP: HTTPS
- Now heavily used as basis for VPNs
  - Virtual Private Network
  - Way to provide secure network connections to remote users
- Used for email (POP/SMTP/IMAP over SSL)
- Used for SIP (VOIP protocol)

HTTPS


Wells Fargo Account Summ
https://online.wellsfargo.com/das/cgi-bin/session.cgi?screenid=SIGNON_PORTAL_PAUSE
Sign Off
WELLS FARGO
Wells Fargo Business Online®
Accounts
Transfers & Payments
Brokerage
Account Services
Messages & Alerts
Online Solutions
Account Summary
Account Activity
Money Map
Statements & Documents
Last Sign On: November 18, 2013
Account Summary

TLS History

- 1995: SSL 2.0 (Netscape)
- 1996: SSL 3.0 (Netscape, later RFC 6101)
- 1999: TLS 1.0 (RFC 2246)
- 2006: TLS 1.1 (RFC 4346)
- 2008: TLS 1.2 (RFC 5246, 6176)
  - Supported in latest versions of all major browsers
- TLS 1.3 under development in 2014

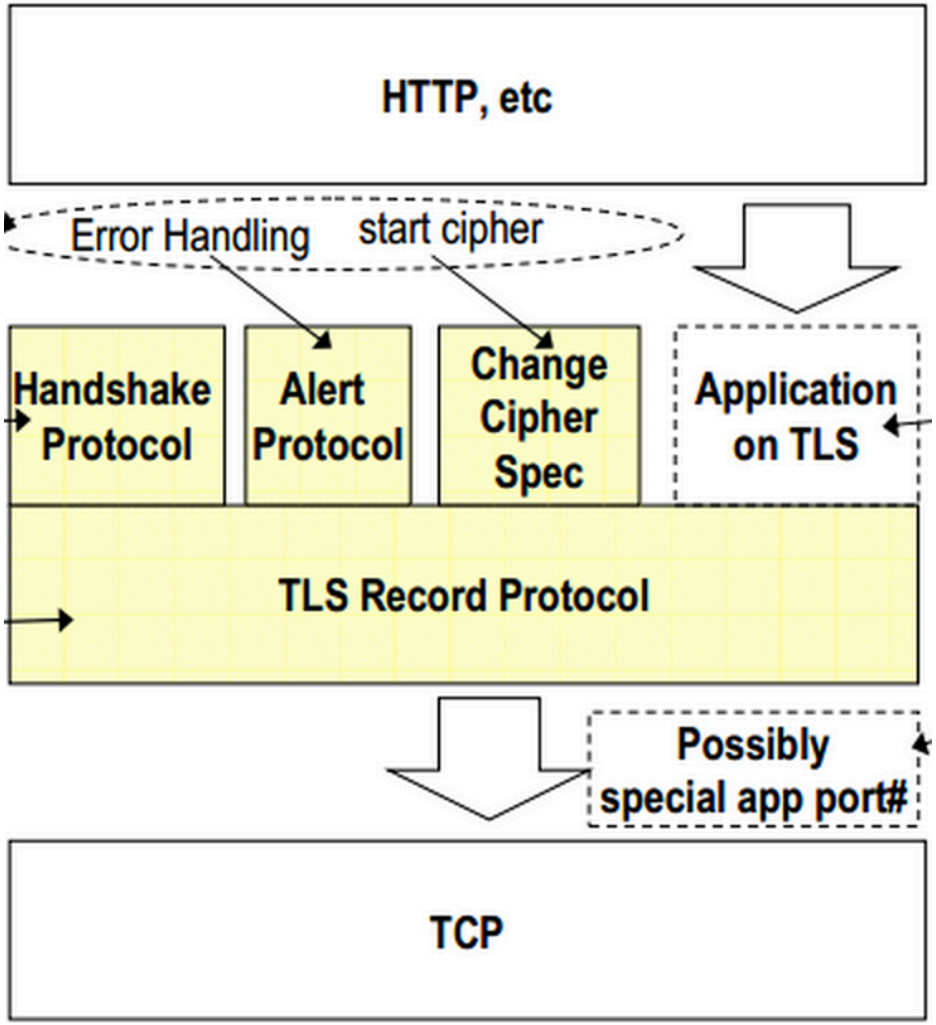
TLS: Step 1

- Establish unencrypted connection
  - Typically over TCP
  - Eg HTTPS over port 443
  - Has also been implemented over UDP
  - And…

Possible Future Implementations

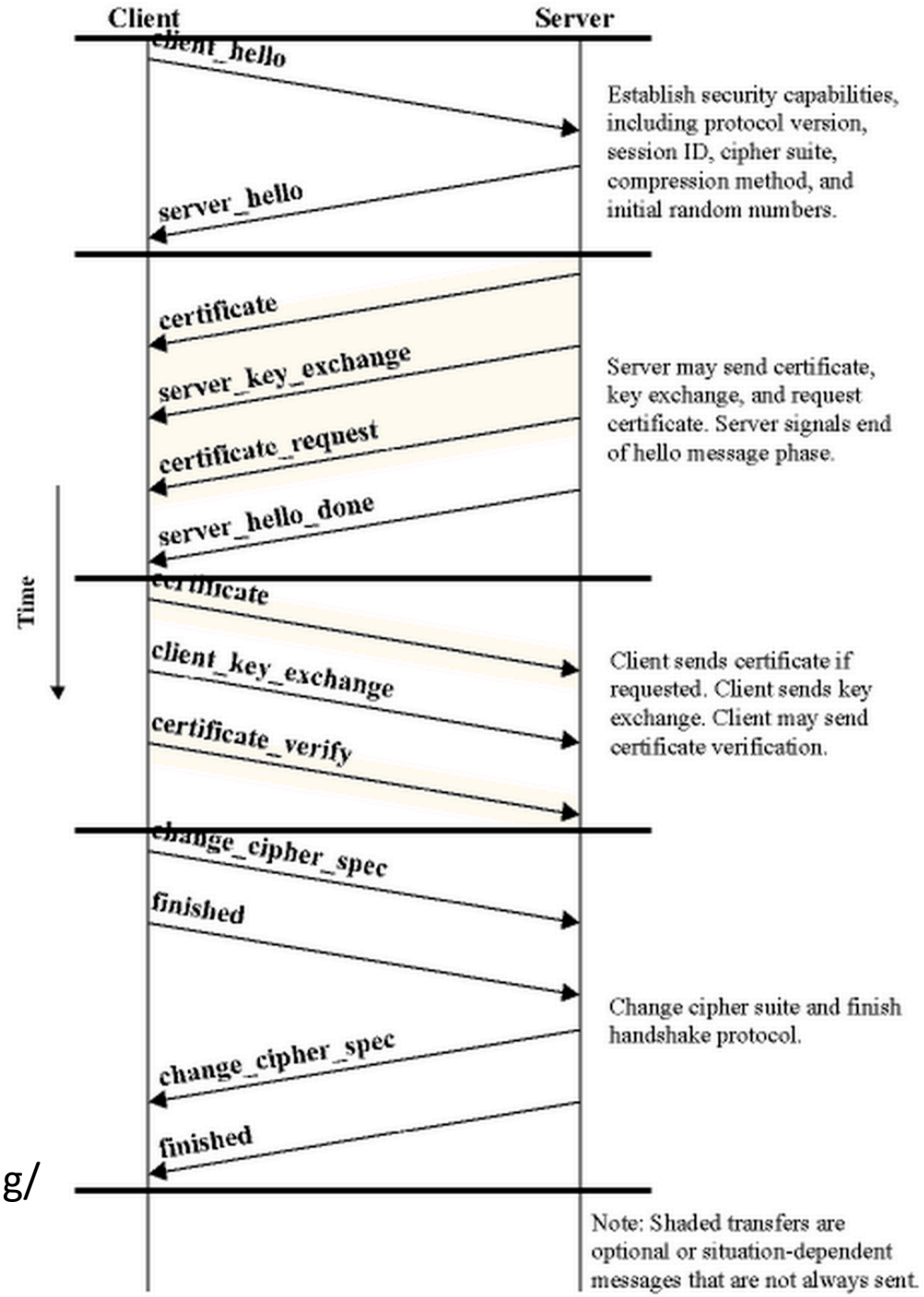
TLS Handshake

- To establish parameters of remaining conversation
- Works over TLS Record layer
- Can also change operation of record layer



http://tech.yanatm.com/?p=338

Handshake Overview



http://www.cs.bham.ac.uk/~mdr/teaching/modules06/netsec/lectures/tls/tls.html

TLS Client Hello

Client Hello. The client initiates a session by sending a Client Hello message to the server. The Client Hello message contains:

- **Version Number.** The client sends the version number corresponding to the highest version it supports. Version 2 is used for SSL 2.0, version 3 for SSL 3.0, and version 3.1 for TLS. Although the IETF RFC for TLS is TLS version 1.0, the protocol uses 3.1 in the version field to indicate that it is a higher level (newer and with more functionality) than SSL 3.0.

- **Randomly Generated Data.** ClientRandom[32], the random value, is a 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number that will ultimately be used with the server random value to generate a master secret from which the encryption keys will be derived.

- **Session Identification (if any).** The sessionID is included to enable the client to resume a previous session. Resuming a previous session can be useful, because creating a new session requires processor-intensive public key operations that can be avoided by resuming an existing session with its established session keys. Previous session information, identified by the sessionID, is stored in the respective client and server session caches.

- **Cipher Suite.** The A list of cipher suites available on the client. An example of a cipher suite is TLS_RSA_WITH_DES_CBC_SHA, where TLS is the protocol version, RSA is the algorithm that will be used for the key exchange, DES_CBC is the encryption algorithm (using a 56-bit key in CBC mode), and SHA is the hash function.

- **Compression Algorithm.** The requested compression algorithm (none currently supported).

http://technet.microsoft.com/en-us/library/cc785811(v=ws.10).aspx

TLS Server Hello

Server Hello. The server responds with a Server Hello message. The Server Hello message includes:

- **Version Number.** The server sends the highest version number supported by both sides. This is the lower of: the highest version number the server supports and the version sent in the Client Hello message.
- **Randomly Generated Data.** ServerRandom[32], the Random Value, is a 4-byte number of the server's date and time plus a 28-byte randomly generated number that will be ultimately used with the client random value to generate a master secret from which the encryption keys will be derived
- **Session Identification (if any).** This can be one of three choices.
    - New session ID – The client did not indicate a session to resume so a new ID is generated. A new session ID is also generated when the client indicates a session to resume but the server can't or won't resume that session. This latter case also results in a new session ID.
    - Resumed Session ID– The id is the same as indicated in the client hello. The client indicated a session ID to resume and the server is willing to resume that session.
    - Null – this is a new session, but the server is not willing to resume it at a later time so no ID is returned.
- **Cipher Suite.** The server will choose the strongest cipher that both the client and server support. If there are no cipher suites that both parties support, the session is ended with a "handshake failure" alert.
- **Compression Algorithm.** Specifies the compression algorithm to use (none currently supported).

TLS Certificate

- Next the server sends its certificate.
- So what is a certificate?
- That requires something of a detour…
  - Cryptographic hash/message digest
  - Digital signature
  - Certificate

Cryptographic Hash

- Take an input plain text *m*
- Outputs a message digest of fixed size *h(m)*
- Such that
  - Any change in input will change output significantly (usually totally)
  - Computationally infeasible, given *h*, to find *m*
  - Computationally infeasible to find m_1 and m_2 such that $h(m_1) = h(m_2)$

Example Cryptographic Hashes

- ~~MD5~~ (long gone bad)
- ~~SH~~A-1 (starting to go bad)
- SHA-256 (currently recommended)
- SHA-3 (on horizon)
- Try it
  - openssl md5 bank.c
  - openssl sha1 bank.c
  - openssl sha256 bank.c
  - openssl no-sha256 or openssl help
  - openssl version

How SHA-1 Works

- Examine the pseudo-code at
- http://en.wikipedia.org/wiki/SHA-1

Digital Signature

- Scheme for guaranteeing that a message is what it appears to be
  - Authentication
    - was not created by an imposter instead of sender
  - Non-repudiation
    - Sender cannot deny having sent it
  - Integrity
    - Message not altered in transit

Digital Signature Implementation

- Sender
  - Hash message with cryptographic hash fn
  - Encrypt hash with *private* key (eg RSA)
  - Attach signature to message
- Receiver
  - decrypt signature with *public* key
  - Recompute hash
  - Make sure signature matches message hash
- Essentially proves that sender was in possession of private key associated with given public key

X.509 Certificates

- Public Key Infrastructure
  - Dates back to 1988
  - RFC 5280 (IETF version) for practical purposes
- Tries to solve the problem
  - How do I find/validate the public key for X?
- Relies on the idea of chain of trust

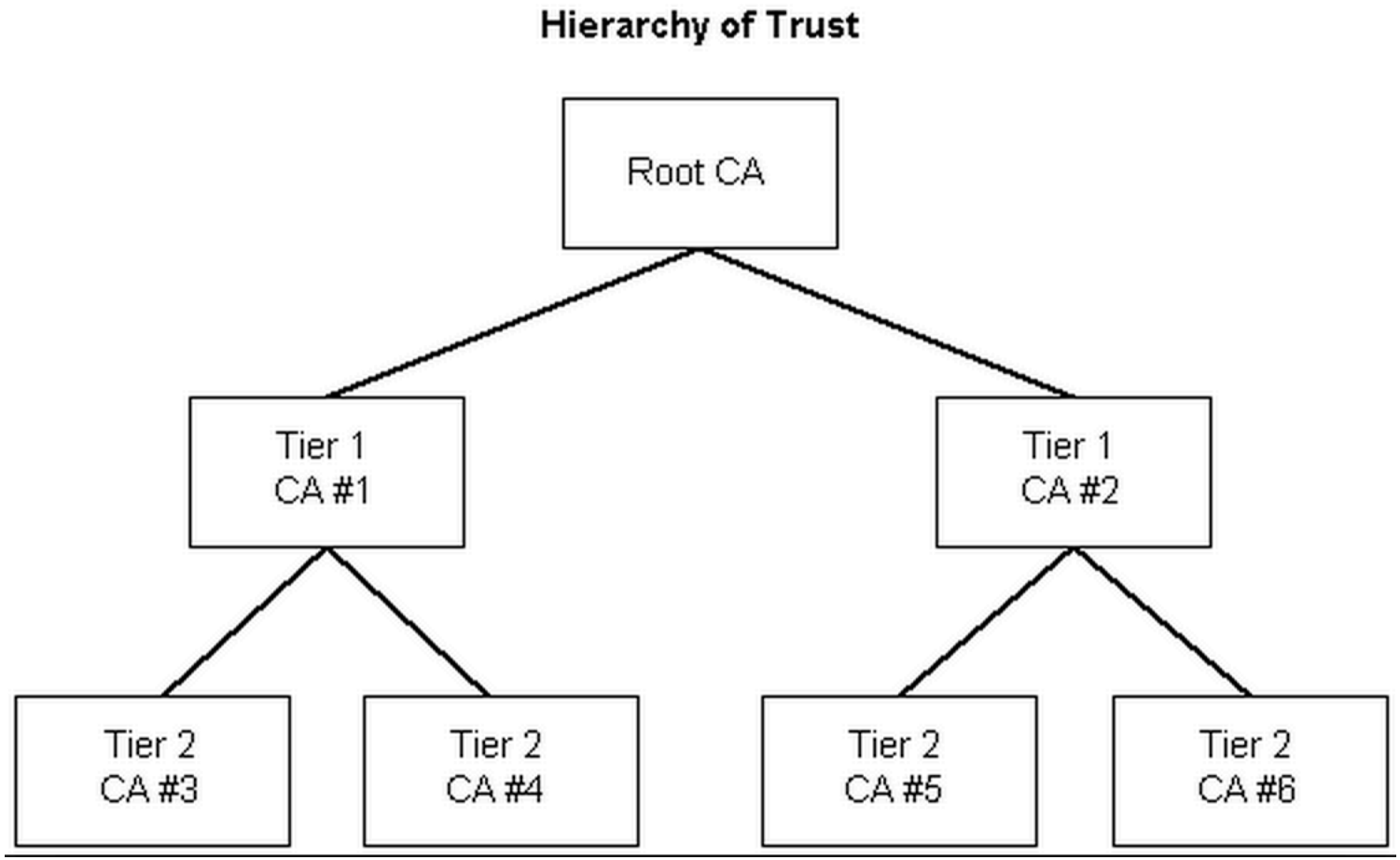
What is in X.509 Cert?

- Certificate version (eg 3)
- Serial Number (eg 480025)
- Algorithm ID (eg ‘sha1WithRSAEncryption’)
- Issuer (CA = Certificate Authority, eg Geotrust)
- Validity Time Interval
  - Cert not to be used outside of this

More in X.509 Cert

- Subject: (eg domain of website)
- Subject Public Key info
  - Algorithm
  - Public Key data
- X509 extensions
- Signature

Certificate Authorities



http://msdn.microsoft.com/en-us/windows/aa382479(v=vs.90)

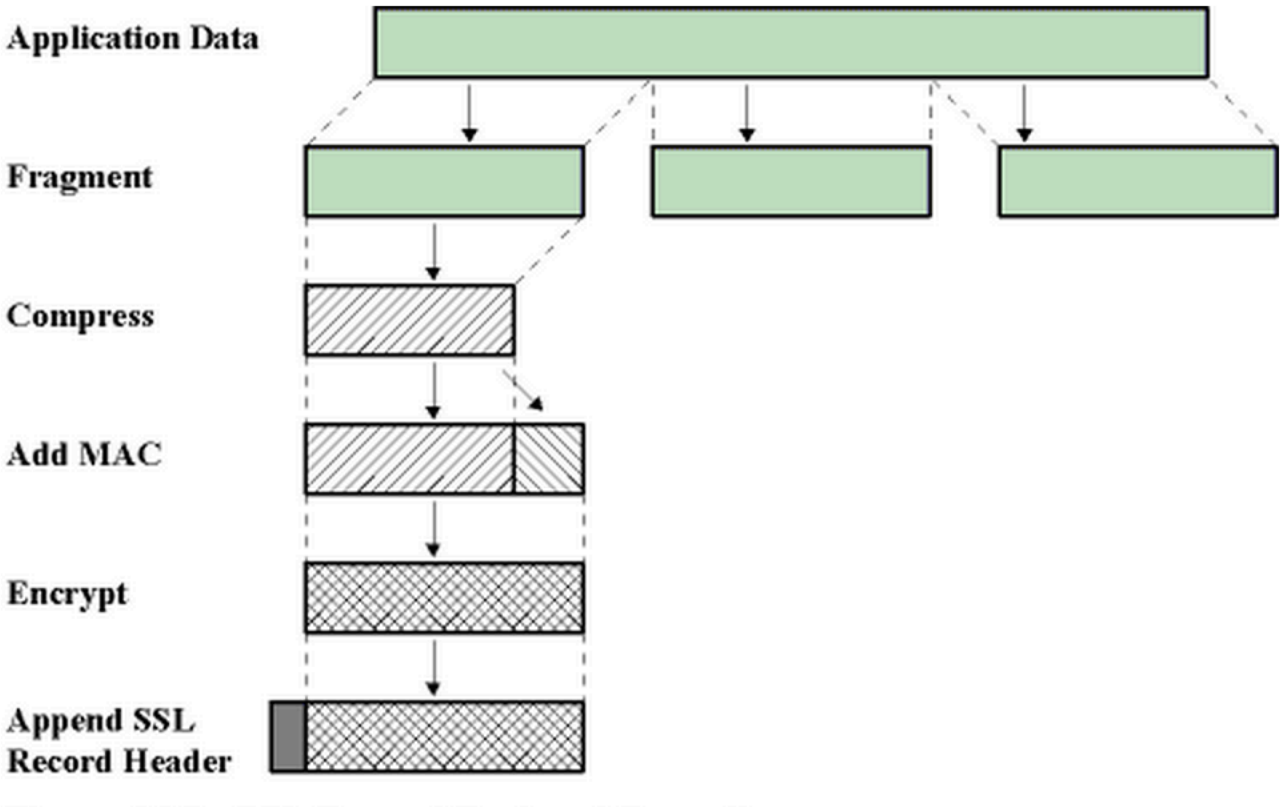
Hundreds of Root CAs

- Eg let's examine the list for Mozilla:
- http://www.mozilla.org/projects/security/certs/included/

Obtaining Certificate to Play

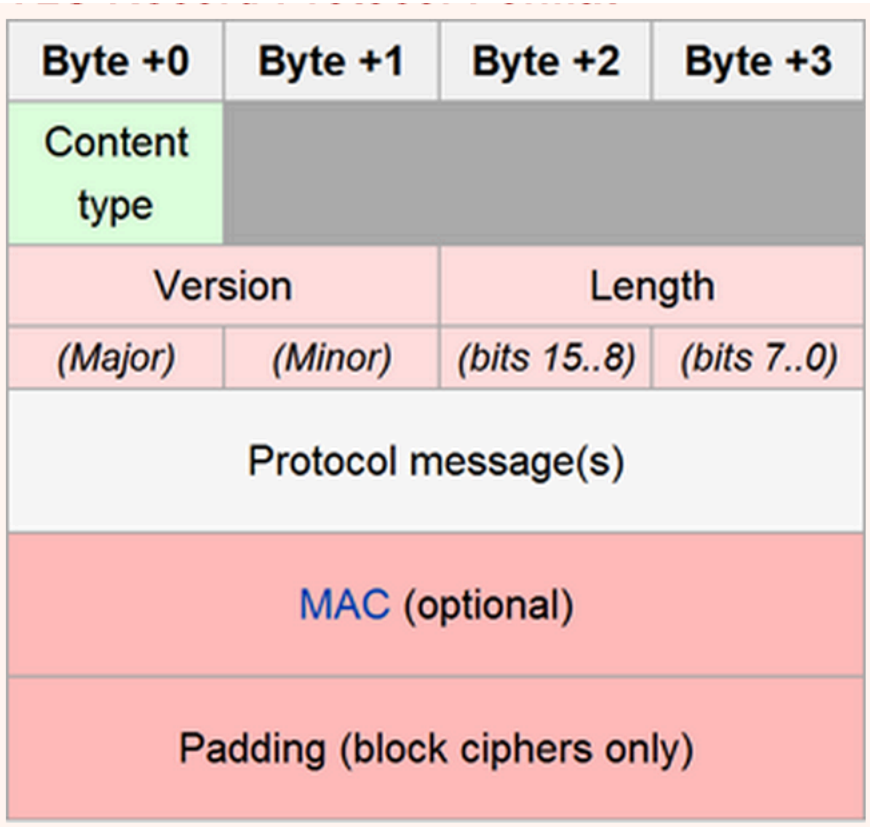
- https://www.networking4all.com/en/support/tools/site+check/
- Eg try with www.nytimes.com
- openssl x509 -in nytimes-cert.txt -text |more

TLS Record Layer



http://www.cs.bham.ac.uk/~mdr/teaching/modules06/netsec/lectures/tls/tls.html

TLS Record Format



Byte +0	Byte +1	Byte +2	Byte +3
Content type			
Version		Length	
(Major)	*(Minor)*	*(bits 15..8)*	*(bits 7..0)*
Protocol message(s)			
MAC (optional)			
Padding (block ciphers only)			

Major Version	Minor Version	Version Type
3	0	SSL 3.0
3	1	TLS 1.0
3	2	TLS 1.1
3	3	TLS 1.2

Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application

http://tech.yanatm.com/?p=338