

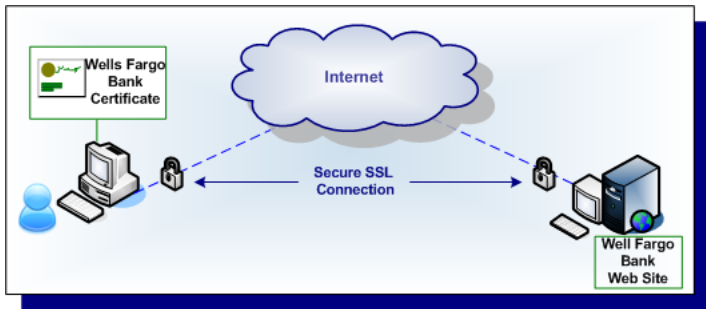
The SSL/TLS protocol

Myrto Arapinis
School of Informatics
University of Edinburgh

October 30, 2017

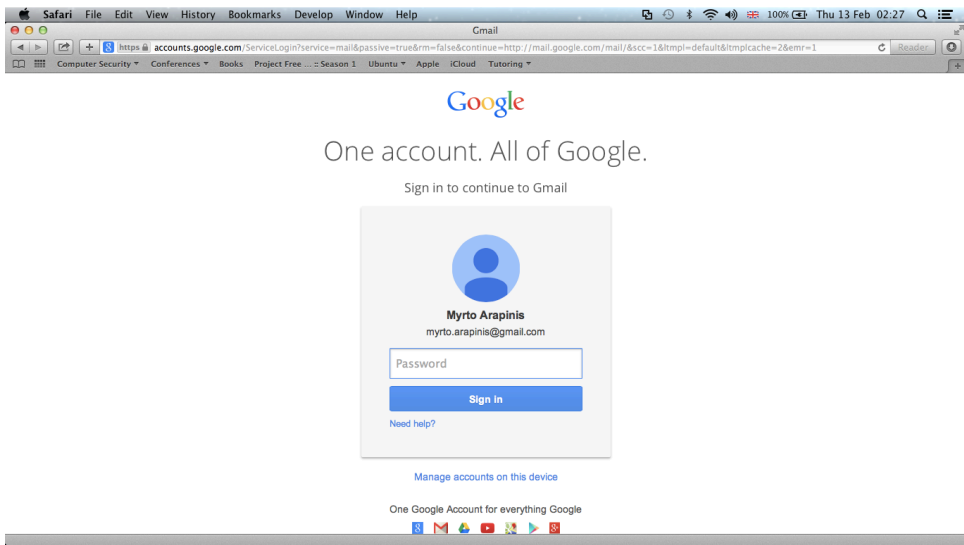
SSL/TLS protocol

Goals: Confidentiality, Integrity, Non repudiation

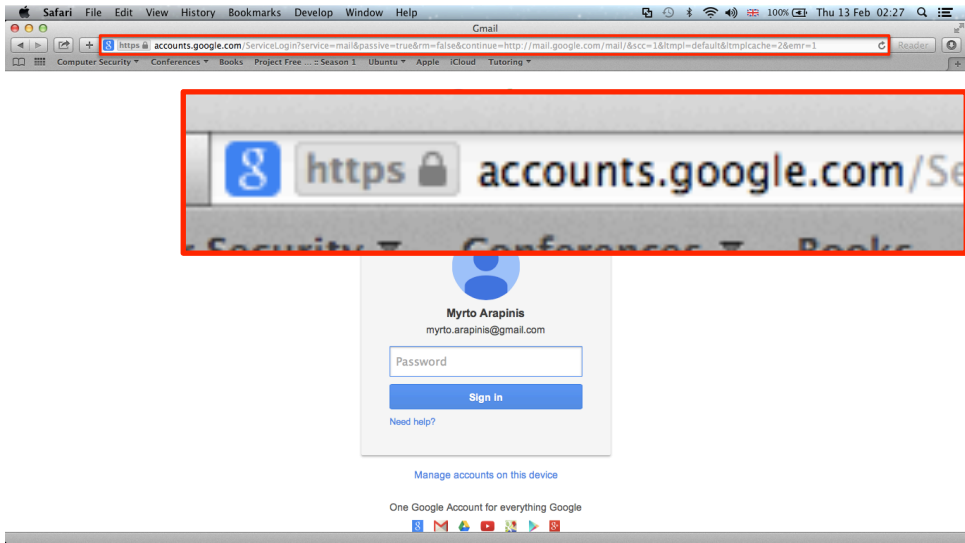


SSL/TLS use X.509 certificates and hence asymmetric cryptography to exchange a symmetric key. This session key is then used to encrypt subsequent communication. This allows for **data/message confidentiality**, and message authentication codes for **message integrity** and thus, **message authentication**.

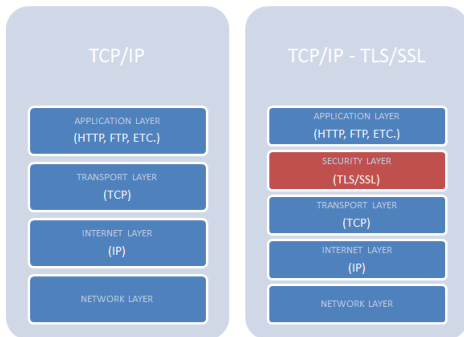
SSL/TLS protocol



SSL/TLS protocol

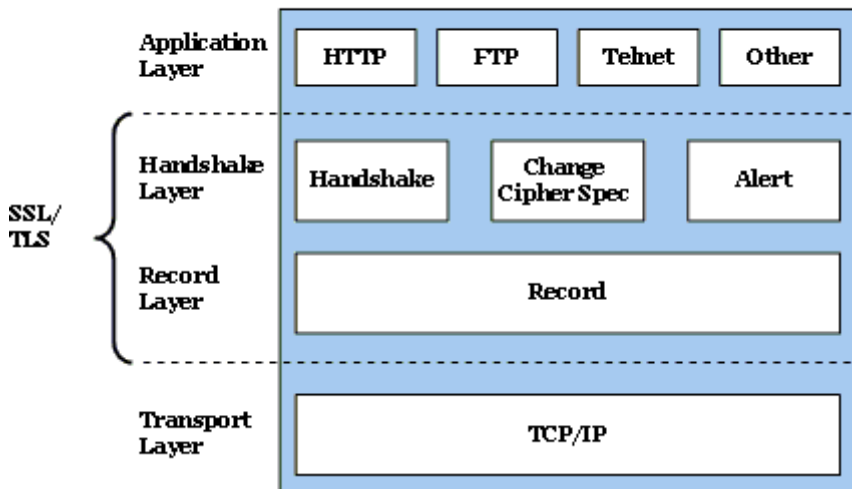


TCP/IP protocol stack



- ▶ TCP/IP provides end-to-end connectivity and is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved
- ▶ The SSL/TLS library operates above the transport layer (uses TCP) but below application protocols

SSL/TLS protocol layers



SSL/TLS handshake protocol



Client Hello

The image shows a Wireshark network packet capture of a TLS Client Hello message. The packet list on the left shows four packets: a SYN, a SYN-ACK, an ACK, and the Client Hello. The Client Hello packet (No. 134) is selected, and its details are expanded on the right. The details pane shows the following structure:

- Cipher Suites Length: 30
 - Cipher Suites (15 suites)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc0ca8)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
 - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
 - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
 - Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
 - Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
 - Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
 - Compression Methods Length: 1
 - Compression Methods (1 method)
 - Extensions Length: 127
 - Extension: server_name
 - Extension: Extended Master Secret
 - Extension: renegotiation_info
 - Extension: elliptic_curves
 - Extension: ec_point_formats
 - Extension: SessionTicket
 - Extension: next_protocol_negotiation
 - Extension: Application Layer Protocol Negotiation

The packet bytes pane at the bottom shows the raw data of the Client Hello packet, starting with the TLS record structure (0030 72 10 bd 9e 00 00 16 03 01 00 ca 01 00 00 c6 03).

Secure Sockets Layer (SSL), 207 bytes

Packets: 475 - Displayed: 475 (100.0%) - Load time: 0:0.11 - Profile: Default

Server Hello

The image shows a Wireshark network packet capture of a TLS handshake. The packet list at the top shows several packets, with packet 135 (6.015031282) selected, which is a TLSv1.2 Client Hello. The packet details pane on the right shows the structure of the TLSv1.2 Record Layer, Handshake Protocol: Server Hello. The packet bytes pane at the bottom shows the raw hex and ASCII data of the packet.

Wireshark Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
132	5.977515105	172.16.76.158	172.217.23.45	TCP	54	35638→443 [ACK] Seq=1 Ack=1 Win=29200 Len=0
133	5.978337850	172.16.76.158	172.217.23.45	TLSv1..	261	Client Hello
134	5.979058758	172.217.23.45	172.16.76.158	TCP	60	443→35638 [ACK] Seq=1 Ack=208 Win=64240 Len=0
135	6.015031282	172.217.23.45	172.16.76.158	TLSv1..	2814	Server Hello
136	6.015054625	172.16.76.158	172.217.23.45	TCP	54	35638→443 [ACK] Seq=208 Ack=2761 Win=33580 Len=0

Packet Details:

- Frame 135: 2814 bytes on wire (22512 bits), 2814 bytes captured (22512 bits) on interface 0
- Ethernet II, Src: Vmware_f0:7d:d2 (00:50:56:f0:7d:d2), Dst: Vmware_9e:08:02 (00:0c:29:9e:08:02)
- Internet Protocol Version 4, Src: 172.217.23.45, Dst: 172.16.76.158
- Transmission Control Protocol, Src Port: 443, Dst Port: 35638, Seq: 1, Ack: 208, Len: 2760
- Secure Sockets Layer
 - TLSv1.2 Record Layer: Handshake Protocol: Server Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 76
 - Handshake Protocol: Server Hello
 - Handshake Type: Server Hello (2)
 - Length: 72
 - Version: TLS 1.2 (0x0303)
 - Random
 - Session ID Length: 0
 - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
 - Compression Method: null (0)
 - Extensions Length: 32
 - Extension: renegotiation_info
 - Extension: server_name
 - Extension: Extended Master Secret
 - Extension: SessionTicket TLS
 - Extension: Application Layer Protocol Negotiation
 - Extension: ec_point_formats

Packet Bytes:

Offset	Hex	ASCII
0030	fa f0 c7 97 00 00 16 03 03 00 4c 02 00 00 48 03L...H.
0040	03 58 11 20 c9 d4 fa 52 3c d5 63 f2 c3 90 11 14	.X. ...R <.c....
0050	55 6e 3b 4c ea 4c 6f 28 71 0e 38 42 1b 8e b6 c7	Un;L.Lo[q.8B....
0060	42 00 c0 2f 00 00 20 ff 81 00 01 00 00 00 00	B../. . q.....
0070	00 17 00 00 00 23 00 00 00 10 00 05 00 03 02 68h
0080	32 00 0b 00 02 01 00 16 03 03 0c 2a 0b 00 0c 26	2.....*...6
0090	00 0c 23 00 04 a5 30 82 04 a1 30 82 03 89 a0 03	..#.0. .0.....

Secure Sockets Layer (ssl), 2760 bytes

Packets: 475 - Displayed: 475 (100.0%) - Load time: 0:0.11 - Profile: Default

Certificate

Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tlsGmail.pcapng

Apply a display filter ... <38/>

No.	Time	Source	Destination	Protocol	Length	Info
134	5.979058758	172.217.23.45	172.16.76.158	TCP	60	443->35638 [ACK] Seq=1 Ack=208 Win=64240 Len=0
135	6.015031282	172.217.23.45	172.16.76.158	TLSv1..	2814	Server Hello
136	6.015054625	172.16.76.158	172.217.23.45	TCP	54	35638->443 [ACK] Seq=208 Ack=2761 Win=33580 Len=0
137	6.015117278	172.217.23.45	172.16.76.158	TLSv1..	841	CertificateServer Key Exchange, Server Hello Done
138	6.015126370	172.16.76.158	172.217.23.45	TCP	54	35638->443 [ACK] Seq=208 Ack=3548 Win=36500 Len=0

serialNumber: 7377627938644829374

- signature (sha256WithRSAEncryption)
- issuer: rdnSequence (0)
- validity
- subject: rdnSequence (0)
- subjectPublicKeyInfo
 - algorithm (rsaEncryption)
 - subjectPublicKey: 3082010a0282010100aa0c2a0f111bb011132301a5fcdff...

modulus: 0x0aa0c2a0f111bb011132301a5fcdff7762a8fc0fd68a...

publicExponent: 65537

extensions: 8 items

- algorithmIdentifier (sha256WithRSAEncryption)
 - Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)

0120 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01 000.....
0130 aa 00 c2 a0 f1 11 bb 01 11 32 30 1a 5f cd fd ff20.....
0140 77 62 a8 fc 0f d6 0a 85 67 fe ef cb f7 93 4e 9ag.....N..
0150 ce 49 bc d5 8c 3b 67 b7 4f c6 a7 ab da f8 e8 04I...;g. 0.....
0160 8a 89 c6 da 99 ec 3d 42 8c 0e c0 86 0c c4 25 e3B.....
0170 ee 93 af 17 0c b3 51 88 54 f8 86 71 bb 73 df b4Q. T...q.S..
0180 cf 0e 3a e1 ab 72 f9 9e 88 78 26 5b a4 f7 d8 0dx&.....
0190 c2 a0 70 79 56 c2 43 07 38 de bd 15 a5 54 be 78pyV.C. 8.....T.x
01a0 c7 af 1a cd 3b 71 07 aa ec 2e f8 18 ee fe 78 16q.....X..
01b0 ab 5b 37 e4 97 c0 42 d6 00 48 3d 15 64 0b 76 7c[7...B...H=d.v]
01c0 b3 8b 09 16 41 3b 4b 0d a5 f7 d8 76 d8 7c e7 4bA;K...v...j.K
01d0 ba 3e a1 a0 a4 1e 32 58 fb c4 9d 30 01 fe 7f 90>...2X...l..0..
01e0 61 2a ec a3 d1 28 7d 57 1c 1f af e1 48 df 65 02a...{W...H.e..
01f0 75 37 2b c3 52 72 7c b6 5e be f5 ed 16 0d bc 7cu7+Rr].
0200 ef 09 2f b5 9e 57 46 e2 f8 2c d3 ed 4a 14 7a 57/...WF...J.zW
0210 04 e4 07 8e a1 b4 10 fe 27 8c ca 7e e3 e1 b6 27[.....
0220 07 b5 44 52 99 0b 0c 92 36 47 22 0c e7 19 bdDR.....6G..
0230 02 03 01 00 01 a3 82 01 67 30 82 01 63 30 1d 06g0...c0..
0240 03 55 1d 25 04 16 30 14 06 08 2b 06 01 05 05 07U.%..0...+.....
0250 03 01 06 08 2b 06 01 05 05 07 03 02 30 35 06 03+.....05..

Frame (841 bytes) Reassembled TCP (3119 bytes)

INTEGER (pkcs1 modulus), 257 bytes

Packets: 475 - Displayed: 475 (100.0%) - Load time: 0:0.11 - Profile: Default

www.gmail.com's certificate



Safari is using an encrypted connection to accounts.google.com.

Encryption with a digital certificate keeps information private as it's sent to or from the https website accounts.google.com.

Choose an account



GeoTrust Global CA



Google Internet Authority G2



accounts.google.com

Common Name Google Internet Authority G2

Serial Number 7377627938644829374

Version 3

Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)

Parameters none

Not Valid Before Thursday, 6 October 2016 13:59:57 British Summer Time

Not Valid After Thursday, 29 December 2016 12:28:00 Greenwich Mean Time

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters none

Public Key

256 bytes : AA 00 C2 A0 F1 11 8B 01 11 32 30 1A 5F CD FD FF 77 62
A8 FC 0F D6 0A 85 67 FE EF CB F7 93 4E 9A CE 49 8C D5 8C 3B 67 B7
4F C6 A7 AB DA F8 E8 04 8A 89 C6 DA 99 EC 3D 42 8C 0E C0 86 0C
C4 25 E3 EE 93 AF 17 0C B3 51 88 54 F8 86 71 8B 73 DF B4 CF 0E 3A
E1 AB 72 F9 9E 88 78 26 5B A4 F7 D8 0D C2 A0 70 79 56 C2 43 07
38 DE BD 15 A5 54 BE 78 C7 AF 1A CD 3B 71 07 AA EC 2E FB 18 EE FE
78 16 AB 58 37 E4 97 C0 42 D6 00 48 3D 15 64 0B 76 7C B3 8B D9
16 41 3B 4B 0D A5 F7 D8 76 D8 7C E7 4B BA 3E A1 A8 A4 1E 32 58 FB
4C 9D 30 01 FE 7F 90 61 2A EC A3 D1 28 7D 57 1C 1F A7 E1 48 DF
65 02 75 37 28 C3 52 72 7C B6 5E BE F5 ED 16 D0 8C 7C EF 09 2F B5
9E 57 46 E2 F8 2C D3 ED 4A 1A 7A 57 04 E4 07 8E A1 84 10 FE 27 8C
CA 7E E3 E1 86 27 D7 B5 44 52 99 08 0B 0C 92 36 47 22 0C E7 19 BD

Exponent 65537

Key Size 2048 bits

Key Usage Encrypt Verify Derive



Hide Certificate

OK

Key exchange

The image shows a Wireshark packet capture of a TLS handshake. The packet list on the left shows four packets: a TCP SYN, a TLSv1.2 CertificateServer Key Exchange, a TCP ACK, and a TLSv1.2 Client Key Exchange. The packet details pane on the right shows the structure of the Client Key Exchange message, including the Handshake Protocol, Handshake Type, and EC Diffie-Hellman Client Params. The packet bytes pane at the bottom shows the raw data of the Client Key Exchange message, including the EC Diffie-Hellman client pubkey.

Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tlsGmail.pcapng

Apply a display filter ... <36 />

No.	Time	Source	Destination	Protocol	Length	Info
136	6.015054625	172.16.76.158	172.217.23.45	TCP	54	35638→443 [ACK] Seq=208 Ack=2761 Win=33580 Len=0
137	6.015117278	172.217.23.45	172.16.76.158	TLSv1..	841	CertificateServer Key Exchange, Server Hello Done
138	6.015126370	172.16.76.158	172.217.23.45	TCP	54	35638→443 [ACK] Seq=208 Ack=3548 Win=36500 Len=0
139	6.017904477	172.16.76.158	172.217.23.45	TLSv1..	180	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
140	6.019370071	172.16.76.158	172.217.23.14	DCSP	491	Request

Secure Sockets Layer

- ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 70
 - ▼ Handshake Protocol: Client Key Exchange
 - Handshake Type: Client Key Exchange (16)
 - Length: 66
 - ▼ EC Diffie-Hellman Client Params
 - Pubkey Length: 65
 - Pubkey: 04580c88228565d38a865aa51d1c08e2d75d731d40c5b5b7...

▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

▼ TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages

0000 00 50 56 f0 7d d2 00 0c 29 9e 08 02 00 00 45 00 .PV...}.....E.
0010 00 a6 c3 9a 40 00 40 06 ba 02 ac 10 4c 9e ac d9 ...@...L...
0020 17 2d 8b 36 01 bb 8f a7 2f 49 2e 21 d9 12 50 18 -.6.../I...P.
0030 8e 94 bd 4d 00 00 16 03 03 00 46 10 00 00 42 41 ...M...F...BA
0040 04 58 0c 08 22 85 65 03 8a 86 5a a5 1d 1c 08 e2 .X...e...Z...
0050 d7 5d 73 1d 40 c5 b5 b7 04 1c cf 0e 2f d8 77 71 .Is@...../mq
0060 de 13 74 92 b7 83 5d 9f 0b 01 2c 00 4b a5 e1 30 .t...]....K..0
0070 5c 9b c7 99 36 2f c3 a1 f4 43 b5 d4 20 1c 5b 336/...C...[3
0080 35 14 03 03 00 01 01 16 03 03 00 20 00 00 00 00 5.....
0090 00 00 00 00 a8 98 79 12 f3 90 21 a7 d6 24 cc ddy...l...\$
00a0 16 37 73 75 62 63 4c 07 10 6c f2 83 c2 34 a3 71 .7subcL...l...4.q
00b0 da 81 62 e8 ..b.

EC Diffie-Hellman client pubkey (ssl.handshake.client_point), 65 bytes

Packets: 475 - Displayed: 475 (100.0%) - Load time: 0:0.11 - Profile: Default

Change cipher spec

The image shows a Wireshark packet capture of a TLS session. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains icons for various actions like opening files, saving, and zooming. The status bar at the bottom indicates 'Record Layer (ssl.record), 6 bytes', 'Packets: 475 - Displayed: 475 (100.0%)', 'Load time: 0:0.11', and 'Profile: Default'.

The packet list on the left shows several packets. Packet 140 is highlighted, showing a 'Change Cipher Spec' message. The packet details pane on the right shows the structure of the message:

- Handshake Protocol: Client Key Exchange
 - Handshake Type: Client Key Exchange (16)
 - Length: 66
 - EC Diffie-Hellman Client Params
 - Pubkey Length: 65
 - Pubkey: 04580c8B228565d38aB65aa51d1c08e2d75d731d48c5b5b7...
- TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
 - Content Type: Change Cipher Spec (20)
 - Version: TLS 1.2 (0x0303)
 - Length: 1
 - Change Cipher Spec Message
- TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages

The packet bytes pane at the bottom shows the raw data of the selected packet, including the 'Change Cipher Spec' message structure.

miTLS, Triple Handshake, SMACK, FREAK, Logjam, and SLOTH							
mitls.org/pages/attacks							
SimSec ▾	CExam ▾	La cryptogra...ts dévoilés	Conferences ▾	ResearchProfiles ▾	Security-Club ▾	Teaching ▾	Tutoring
		miTLS	Publications	Attacks	Code	FlexTLS	People
		Alert	3SHAKE	VHC	SMACK	Logjam	SLOTH
		Protocol	Cryptographic	Implementation	Deployment		

A Zoo of TLS attacks

Attacks on TLS that break the intuitive security property of a virtual recreation of a physically secure channel can be categorized along three dimensions.

1. Protocol logic vs. cryptographic design flaw
2. Specification/Standard vs. Implementation errors
3. TLS vs. Context

Flaws in the protocol logic

Attacks targeting the protocol logic may for instance cause the client and server to negotiate the use of weak algorithms even though they both support strong cryptography.

If the faulty negotiation logic conforms to the specification, then the attack is on the specification itself (as, e.g., partially enabled by the **False Start** modification). If an implementation deviates from the specification to implement a faulty negotiation logic [Dilmece, Langley] it is an attack on the implementation. As many aspects of the standard can be underspecified or ambiguous, it is not always possible to distinguish between these two cases.

Another class of protocol logic flaws are state-machine bugs [Early CCS Attack, SMACK Attack].

The attack can also be either an attack on TLS proper, or on its context, e.g. if the attacker can just change the configuration files to deactivate strong cryptography. As the TLS standard does not describe APIs or configuration file formats, context specific attacks are always implementation specific.

The renegotiation attack [TLS_Reneg_Attack] is a logical attack on the TLS standard, where one peer believes it is running the first handshake on a connection, while the other peer is running a re-handshake. miTLS prevents the renegotiation attack by implementing the renegotiation extension.

More generally, the TLS specification is vague about how applications should handle data coming from consecutive sessions, e.g. whether it is safe to join them and consider them as a single stream, or if the user should be notified of the change of context. The renegotiation extension partially fixes the problem, but it still leaves room for our alert attack, where the attacker can turn any authentic fatal alert into a warning alert, which gets ignored by default.

Much more seriously, resuming the attacker controlled session on a different connection re-enables the renegotiation attack. This attack is known as the triple handshake attack and is an instance of a larger class of attacks resulting from inadequate channel binding in compound authentication protocols. The miTLS security theorem does not promise channel binding across different connections and is thus not violated by the attack. To be secure, applications making use of miTLS have, however, to be carefully designed to make use of the provided security cues. We give a basic HTTPS client, miHTTPS, as an example for such an application. Subsequently the flaw is also being patched at the TLS level using a new extension.

Cryptographic design flaws

Attacks exploiting cryptographic design flaws may simply result from cryptanalytic progress against the cryptographic building blocks of TLS. They can, however, also result from improper non-blackbox use of otherwise secure cryptographic constructions. An example for this is chosen ciphertext chaining (CBC) mode encryption. Early versions of TLS allow using knowledge of the next initialization vector (IV) to set up adaptive plaintext attacks, see, e.g. [OpenSSL archive](#) for a first mention of the 'BEAST attack'.

SSL/TLS renegotiation

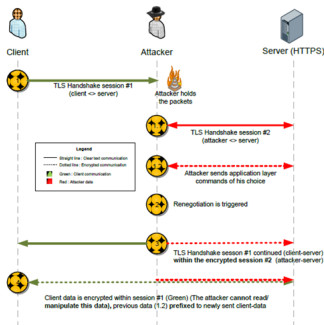
Client and server are allowed to initiate renegotiation of the session encryption in order to:

- ▶ Refresh keys
- ▶ Increase authentication
- ▶ Increase cipher strength
- ▶ ...

Client or server can trigger renegotiation by sending a hello message

SSL/TLS renegotiation weaknesses

- ▶ Renegotiation has priority over application data!
- ▶ Renegotiation can take place in the middle of an application layer transaction!



(Detailed on the board)

Incorrect implicit assumption: the client doesn't change through renegotiation

Marsh Ray's plaintext injection attack on HTTPS

Marsh Ray's plaintext injection attack on HTTPS

Attacker:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1  
X-Ignore-This:(no carriage return)
```

Marsh Ray's plaintext injection attack on HTTPS

Attacker:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1  
X-Ignore-This:(no carriage return)
```

Victim:

```
GET /pizza?toppings=sausage;address=victim_str HTTP/1.1  
Cookie:victim_cookie
```

Marsh Ray's plaintext injection attack on HTTPS

Attacker:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1  
X-Ignore-This:(no carriage return)
```

Victim:

```
GET /pizza?toppings=sausage;address=victim_str HTTP/1.1  
Cookie:victim_cookie
```

Result:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1  
X-Ignore-This:GET /pizza?toppings=sausage;address=victim_str HTTP/1.1  
Cookie:victim_cookie
```

Marsh Ray's plaintext injection attack on HTTPS

Attacker:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:(no carriage return)
```

Victim:

```
GET /pizza?toppings=sausage;address=victim_str HTTP/1.1
Cookie:victim_cookie
```

Result:

```
GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:GET /pizza?toppings=sausage;address=victim_str HTTP/1.1
Cookie:victim_cookie
```

⇒ **Server uses victim's account to send a pizza to attacker!**

Anil Kurmus' plaintext injection attack on HTTPS

Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to `http://twitter.com/statuses/update.xml`, as well as the user name and password

Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to `http://twitter.com/statuses/update.xml`, as well as the user name and password

Attacker:

POST /statuses/update.xml HTTP/1.1

Authorization: Basic username:password

User-Agent: curl/7.19.5

Host: twitter.com

Accept: */*

Content-Length: 140

Content-Type: application/x-www-form-urlencoded
status=

Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to `http://twitter.com/statuses/update.xml`, as well as the user name and password

Attacker:

```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password
User-Agent: curl/7.19.5
Host: twitter.com
Accept: */*
Content-Length: 140
Content-Type: application/x-www-form-urlencoded
status=
```

Victim:

```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password...
```

Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to `http://twitter.com/statuses/update.xml`, as well as the user name and password

Attacker:

```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password
User-Agent: curl/7.19.5
Host: twitter.com
Accept: */*
Content-Length: 140
Content-Type: application/x-www-form-urlencoded
status=
```

Victim:

```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password...
```

⇒ **the attacker gets the user name and password of the victim!**

The SAML Single Sign On (SSO) protocol


SAML SSO protocol

Chrome File Edit View History Bookmarks Window Help Thu 13 Feb 00:50


BBC - Homepage www.bbc.co.uk

BBC Sign in News Sport Weather iPlayer TV Radio More Search

THURSDAY 13 FEBRUARY




Storm updates: Get the latest for where you are



Triple killer's accomplices guilty

BBC NEWS


- » UK parties 'will block money union'
- » Comedian Sid Caesar dies at 91
- » Carney adjusts interest rates policy
- » Ketamine to become Class B drug




Fulham 2-3 Liverpool

BBC SPORT

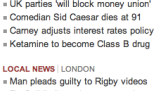
- » Decisive day for Team GB in Sochi
- » Collingwood handed England role
- » Arsenal 0-0 Manchester United
- » Everton fan's 30-year wait goes on



BBC ONE COMEDY
Outnumbered
Episode 3

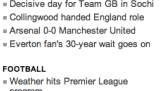


BBC RADIO 2 | FACTUAL
David Attenborough and th...




LOCAL NEWS | LONDON

- » Man pleads guilty to Rigby videos
- » Ex-Guildhall teacher on rape charges
- » Murdered boy's parents collect MBE



FOOTBALL

- » Weather hits Premier League program...
- » Newcastle United 0-4 Tottenham Hots...
- » Hodgson rules out Terry return



Parks and Rec is back
Behold the Pyramid!

MARK KERMODE
Shia Shocker!

BBC now Entertainment News Lifestyle Knowledge Sport

https://sst.bbc.co.uk/tid/signin

SAML SSO protocol

Chrome File Edit View History Bookmarks Window Help

BBC - Sign in

https://ssl.bbc.co.uk/id/signin

BBC Sign in News Sport Weather iPlayer TV More Search

SIGN IN BBC iD

Don't have a BBC ID? Please register.

Email or username

Password

[Forgot your password?](#)

☒ Remember me Untick if you're using a shared computer.

[Sign in](#) [Cancel](#)

Other ways to sign in

You'll be signed in to the BBC for 30 days.

[Facebook](#) [Google](#)

Please only use these if you are 16 or over.
We won't share or post your activity to Facebook or Google

About BBC iD

Simple
Register quickly and easily to comment, add favourites, and more...

Safe
We store your information securely, and we never share it without your permission.

Spam-free
We'll only send you emails if you ask for them.

[BBC iD help](#)

BBC

iWonder

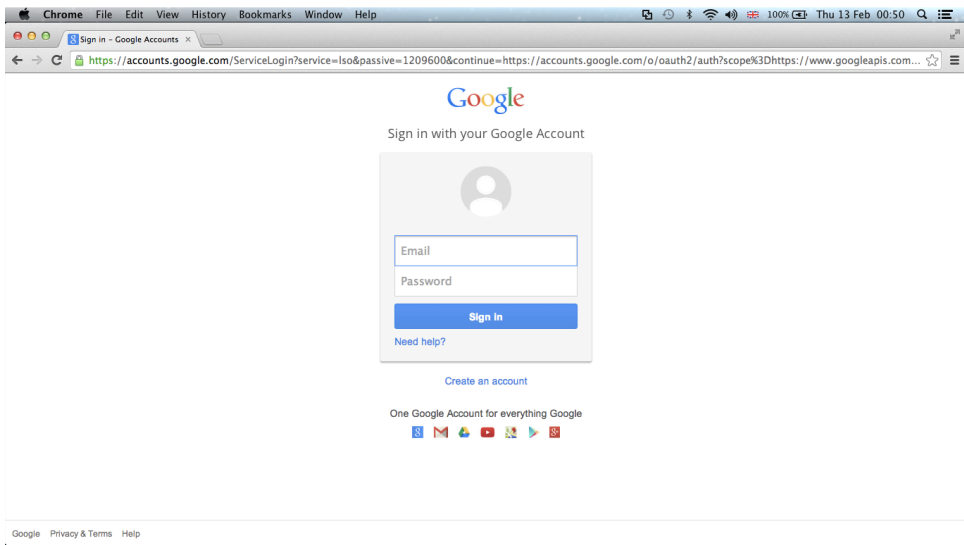
How did Pack Up Your Troubles become the viral hit of WW1?

Gareth Malone explains why the song was a success

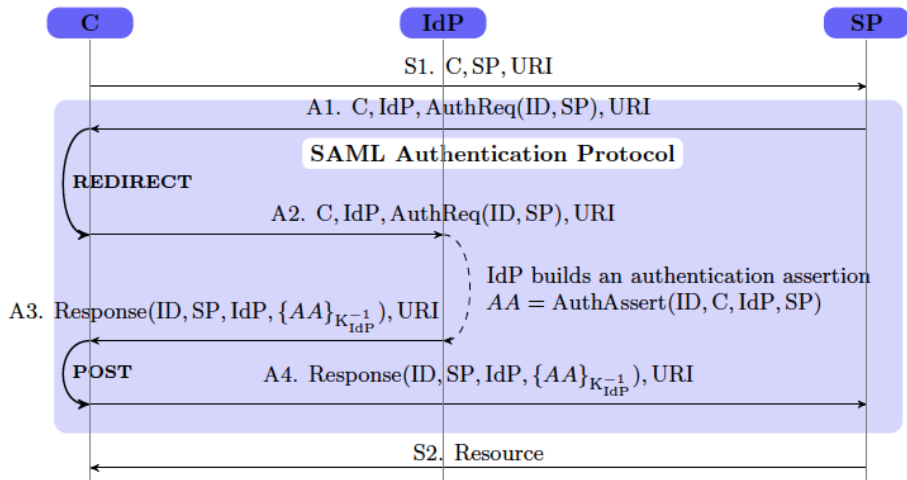
News	Sport	Weather	iPlayer	TV	Radio
pebbles	Comedy	Food	History	Learning	

https://ssl.bbc.co.uk/id/statecookie/google.com

SAML SSO protocol



SAML SSO protocol (OASIS 2005)



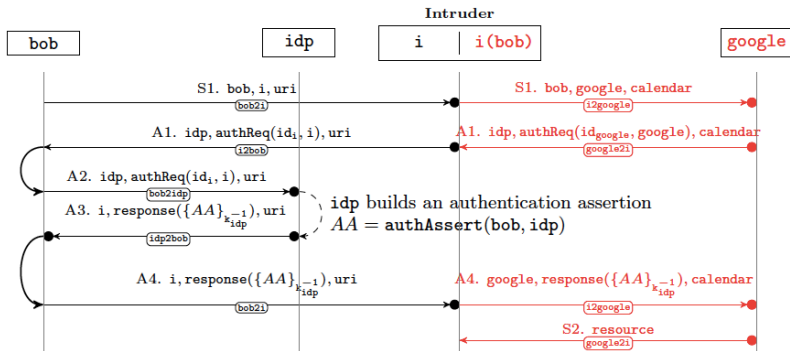
Google's implementation of SSO

Google's SAML-based Single Sign-On for Google Applications deviates from the above protocol for a few, seemingly minor simplifications in the messages exchanged:

- G1. ID and SP are not included in the authentication assertion, *i.e.* $AA = \text{AuthAssert}(C; IdP)$ instead of $\text{AuthAssert}(ID; C; IdP; SP)$;
- G2. ID , SP and IdP are not included in the response, *i.e.* $Resp = \text{Response}(\{AA\}_{K_{IdP}^{-1}})$ instead of $\text{Response}(ID; SP; IdP; \{AA\}_{K_{IdP}^{-1}})$.

Attack Google's SSO implementation

[A. Armando, R. Carbone, L. Compagna, J. Cullar, L. Tobarra, "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps", (FMSE'08)]



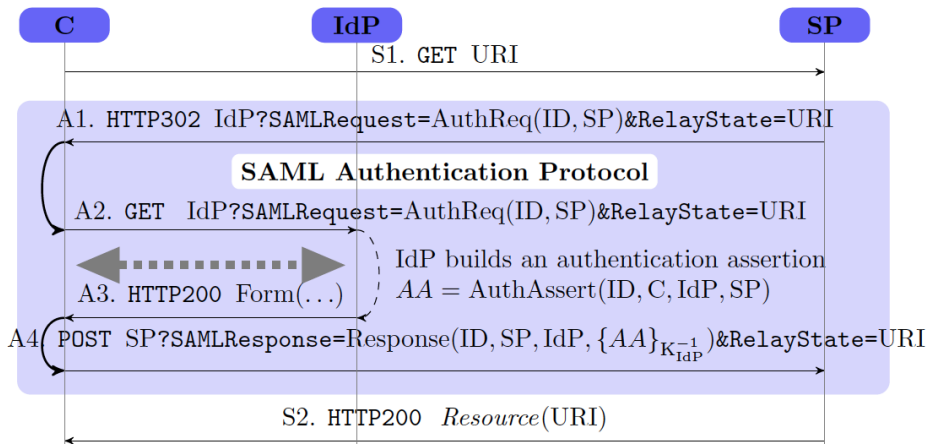
Legend:

$A \xrightarrow{\text{ch}} B$: A sends M on ch confidential to B

$A \xrightarrow{\bullet \text{ch}} B$: A sends M on ch authentic for A

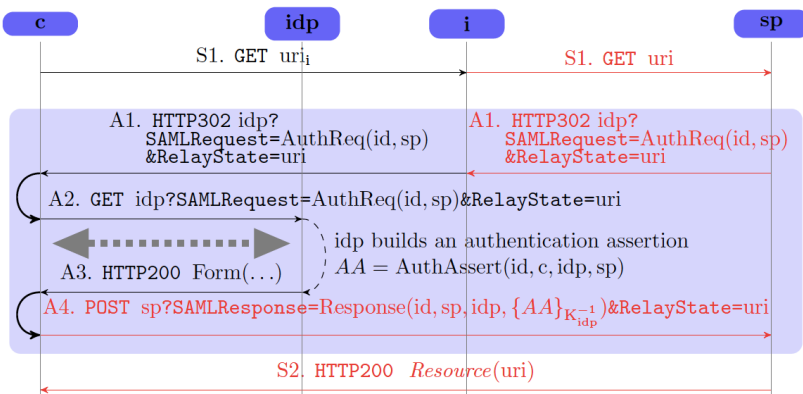
$A \xrightarrow{\bullet \text{ch}} \bullet B$: M is sent on ch authentic for A and confidential to B

SAML SSO protocol (OASIS 2012)



Attack SAML SSO protocol (OASIS 2012)

[A. Armando, R. Carbone, L. Compagna, J. Cullar, G. Pellegrino, A. Sorniotti, "From Multiple Credentials to Browser-Based Single Sign-On: Are We More Secure?", Chapter in Future Challenges in Security and Privacy for Academia and Industry]



⇒ XSS attack on SAML-base SSO for Google Apps