

CSE 127: Introduction to Security

TLS and Secure Channels

George Obaido

UCSD

Spring 2022

Some material from Nadia Heninger, Deian Stefan, Dan Boneh, Stefan Savage

Reminder: Cryptographic primitives

Symmetric crypto

Public-key crypto

Confidentiality

Symmetric encryption
(e.g. AES)

Public-key encryption
(e.g. RSA)

Integrity/Authenticity

MACs

Digital signatures

Reminder: Network Attacker Threat Model

Network Attacker:

- Controls infrastructure: Routers, DNS
- Eavesdrops, injects, drops, or modifies packets

Examples:

- WiFi at internet cafe
- Internet access at hotels

Goal: Establish a secure channel to a host that ensures

- Confidentiality and Integrity of messages
- Authentication of the remote host

Common cryptographic network protocols

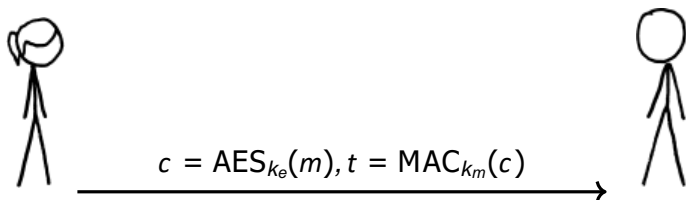
- TLS (Transport Layer Security)
 - Used to provide an encryption wrapper around HTTP to make HTTPS, and for many other application layer protocols.
 - Security goals: Authenticate server, confidentiality and integrity of traffic

- SSH (Secure Shell)
 - Use to access remote machines
 - Security goals: Authenticate server and client, confidentiality and integrity of traffic

- IPsec (Internet Protocol Security)
 - Provides an encrypted, authenticated alternative to IP
 - Commonly used for VPNs (Virtual Private Networks)
 - Security goals: client and server authentication, authenticate headers, optionally encrypt headers, ensure confidentiality and integrity of payloads

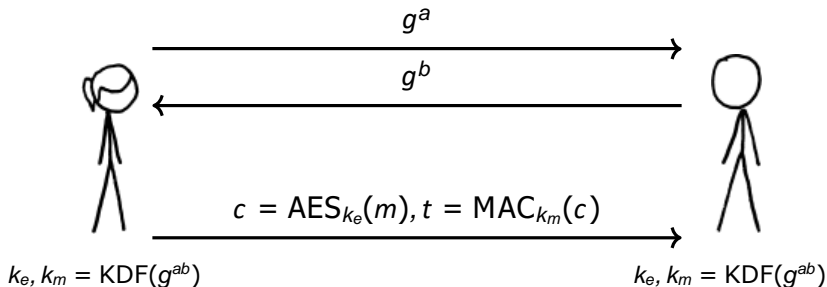
Constructing a secure encrypted channel

- To ensure confidentiality and integrity: Encrypt and MAC data



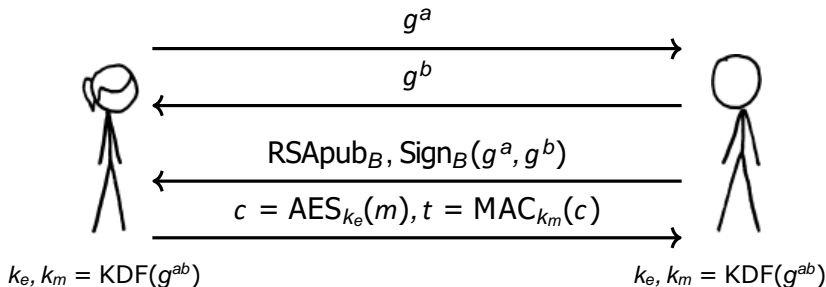
Constructing a secure encrypted channel

- To ensure confidentiality and integrity: Encrypt and MAC data
- To negotiate shared symmetric keys: Diffie-Hellman key exchange. Key Derivation Function (KDF) maps shared secret to symmetric key.



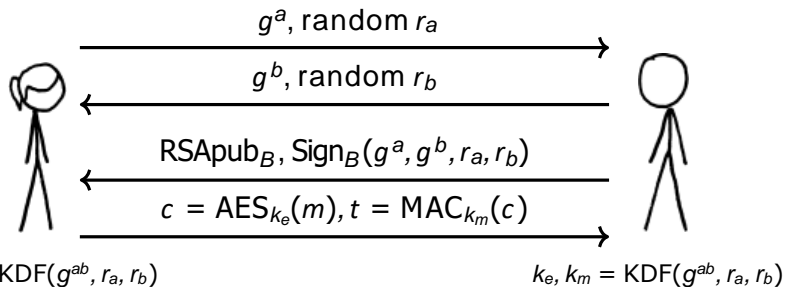
Constructing a secure encrypted channel

- To ensure confidentiality and integrity: Encrypt and MAC data
- To negotiate shared symmetric keys: Diffie-Hellman key exchange. Key Derivation Function (KDF) maps shared secret to symmetric key.
- To ensure authenticity of endpoints: Digital Signatures



Constructing a secure encrypted channel

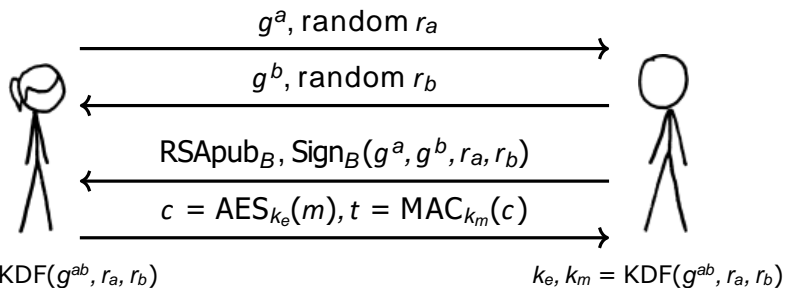
- To ensure confidentiality and integrity: Encrypt and MAC data
- To negotiate shared symmetric keys: Diffie-Hellman key exchange. Key Derivation Function (KDF) maps shared secret to symmetric key.
- To ensure authenticity of endpoints: Digital Signatures
- To ensure an adversary can't reuse a signature later, add some random unique values ("nonces")



This is not exactly what TLS looks like, but it's similar.

Constructing a secure encrypted channel

- To ensure confidentiality and integrity: Encrypt and MAC data
- To negotiate shared symmetric keys: Diffie-Hellman key exchange. Key Derivation Function (KDF) maps shared secret to symmetric key.
- To ensure authenticity of endpoints: Digital Signatures
- To ensure an adversary can't reuse a signature later, add some random unique values ("nonces")



How does Alice know to trust Bob's public signing key?

Public Key Infrastructure: Establishing Trust in Keys

Ways to establish trust in keys:

- Meet in person to exchange keys.
 - Not practical at scale over the internet

Public Key Infrastructure: Establishing Trust in Keys

Ways to establish trust in keys:

- **Fingerprint verification**

- Verify a cryptographic hash of a public key through a separate channel, or “**trust on first use**” (TOFU).
- This is used by SSH for host keys.

```
$ ssh elk.sysnet.ucsd.edu
```

```
The authenticity of host 'elk.sysnet.ucsd.edu (137.110.222.162)' can't be established.
```

```
ED25519 key fingerprint is SHA256:rl/PqZezDo18EbK8U8/HXesu07iCoNUGa+8r3t3qGxw.
```

```
This key is not known by any other names
```

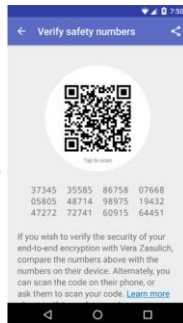
```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Public Key Infrastructure: Establishing Trust in Keys

Ways to establish trust in keys:

- **Fingerprint verification**

- Verify a cryptographic hash of a public key through a separate channel, or “trust on first use” (TOFU).
- This is used by SSH for host keys.
- This is also used by encrypted messaging apps like **Signal**



Public Key Infrastructure: Establishing Trust in Keys

Ways to establish trust in keys:

- **Certificate Authorities**
 - A certificate authority (CA) is a kind of commercial trusted intermediary.
 - Certificate Authorities verify public keys and sign them in exchange for money.
 - If you trust the certificate authority, you transitively trust the keys it signs.
 - This is used for TLS, software signing keys.

Public Key Infrastructure: Establishing Trust in Keys

Ways to establish trust in keys:

- Web of Trust (e.g., PGP)
 - In a web of trust, you establish trust in intermediaries of your choice.
 - You then transitively trust the keys they sign.

```
$ gpg --edit-key dickey@invisible-island.net
gpg (GnuPG) 2.2.29; Copyright (C) 2021 Free Software Foundation, Inc. This is free
software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
pub dsa1024/702353E0F7E48EDB
   created: 2004-01-05  expires: 2022-07-05  usage: SC trust: unknown
   validity: unknown
sub elg1024/0296C3D9E4374AE1
   created: 2004-01-05  expires: never        usage: E
[ unknown] (1). Thomas Dickey <dickey@invisible-island.net> gpg> trust
pub dsa1024/702353E0F7E48EDB
   created: 2004-01-05  expires: 2022-07-05  usage: SC trust: unknown
   validity: unknown
sub elg1024/0296C3D9E4374AE1
   created: 2004-01-05  expires: never        usage: E
[ unknown] (1). Thomas Dickey <dickey@invisible-island.net>
```

Please decide howfar you trust this user to correctly verify other users' keys (by looking at passports, checking fingerprints from different sources, etc.)

- 1 = I don't know or won't say
 - 2 = I do NOT trust
 - 3 = I trust marginally
 - 4 = I trust fully
 - 5 = I trust ultimately
 - m = back to the main menu
- Your decision?

A more modern and practical WoT: Keybase

The screenshot shows a web browser window with the URL <https://keybase.io/deian>. The profile for 'deian' (Deian Stefan, 50) is displayed. It includes a profile picture, a search bar, an 'Install' button, and a 'Chat with deian' button. A note states: 'Your conversation will be end-to-end encrypted.' Below the profile, there are two columns of users: 'Following (13)' and 'Followers (16)'. The 'Following' list includes: marcovassena (Marco Vassena), nadia (Nadia Hennings), senyk (Sany Katsira), hannahedavis (Hannah Davis), and mihirbellare (Mihir Bellare). The 'Followers' list includes: marcovassena (Marco Vassena), dmazieres (David Mazieres), tondivakar (Arjun Tondivakar), balyuli (Biny), and nadia (Nadia Hennings).

The screenshot shows a chat interface for a contact named 'cravens.birthdays'. The chat history includes messages from 'mlennox, suzyq' (7:28 PM) saying 'Your nice', 'cravenc' (7:27 PM), and 'cravena' (7:26 PM) saying 'Yup!'. The current message from 'cravens.birthdays' (Wed) is a photo of a Visa credit card. The card is a Visa Signature card with the number 4544 1500 5155 0150, expiration date 09/08 09/11, and the name 'DADDY BOE'. Below the photo, there are several replies from other users: 'cravenc' (11:52 AM - Yesterday) says 'thank!', 'drcraven' (11:53 AM - Yesterday) says 'Maybe a bag for his head too 😊', 'cravenc' (12:11 PM - Yesterday) says 'HAHA 😂', and 'cravena' (12:11 PM - Yesterday) says '👍👍👍👍👍👍'.

Disclaimer: Available on Keybase.

<https://keybase.io/>

TLS: Transport Layer Security

- TLS provides an encrypted channel for application data.
- Used for HTTPS: HTTP inside of a TLS session
- Used to be called SSL (Secure Sockets Layer) in the 90s.

SSL 1.0 Terribly insecure; never released.

SSL 2.0 Released 1995; terribly insecure,
deprecated in 2011

SSL 3.0 Released 1996; insecure, deprecated in
2015.

TLS 1.0 Released 1999; deprecated in 2020.

TLS 1.1 Released 2006; deprecated in 2020.

TLS 1.2 Released 2008. Ok.

TLS 1.3 Standardized in August 2018 and is being
rolled out now; major change from TLS 1.2.

TLS 1.2 with Diffie-Hellman Key Exchange

Step 1: The client (browser) tells the server what kind of cryptography it supports.



client hello: client random

[list of cipher suites]



Cipher suites: list of options like:

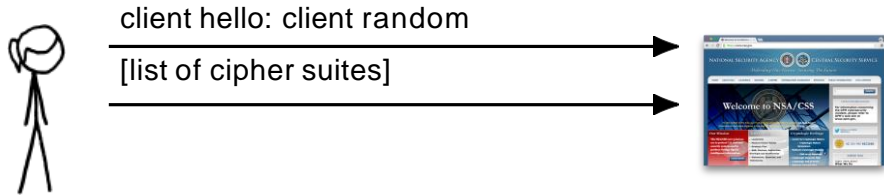
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

This says to use

- elliptic curve Diffie-Hellman for key exchange
- RSA digital signatures
- 128-bit AES for symmetric encryption
- GCM (Galois Counter Mode) AES mode of operation
- SHA-256 for hash function

TLS 1.2 with Diffie-Hellman Key Exchange

Step 1: The client (browser) tells the server what kind of cryptography it supports.



Cipher suites: list of options like:

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Server cipher suite configuration can be confusing and difficult for sysadmins. Many insecure options like

TLS_DHE_RSA_WITH_DES_CBC_SHA

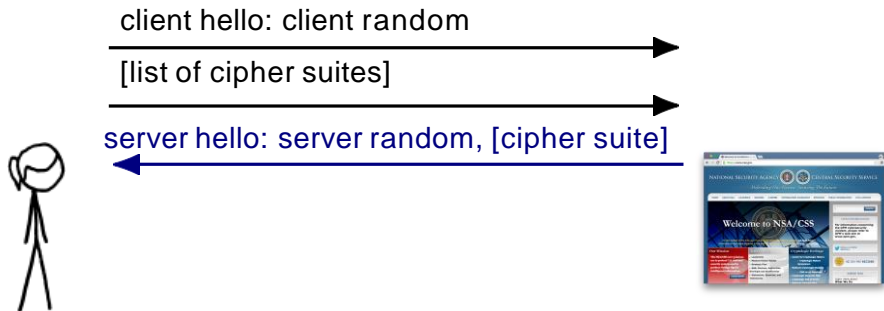
or

TLS_NULL_WITH_NULL_NULL

Subtle protocol errors around cipher suite negotiation.

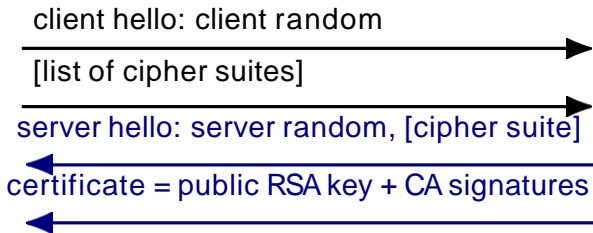
TLS 1.2 with Diffie-Hellman Key Exchange

Step 2: The server tells the client which kind of cryptography it wishes to use.



TLS 1.2 with Diffie-Hellman Key Exchange

Step 3: The server sends over its certificate which contains the server's public key and signatures from a certificate authority.



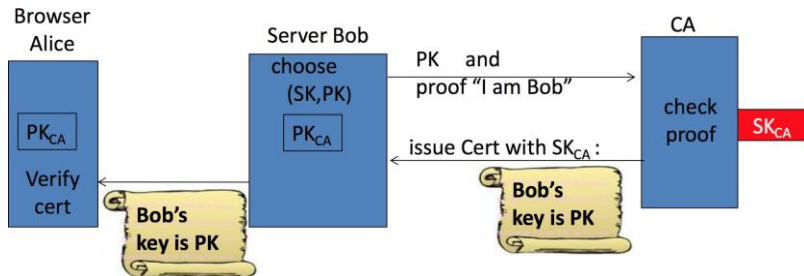
Certificates and Certificate Authorities in TLS

Website public keys are encoded into certificates.

Certificates signed by CAs.

Browsers come with set of trusted CAs.

To verify a certificate, browsers verify chain of digital certificates back to trusted root CA.



Certificates typically valid for 3 months to multiple years.

Sample certificate



mail.google.com

Issued by: Google Internet Authority G3

Expires: Wednesday, June 20, 2018 at 6:25:00 AM Pacific Daylight Time

✔ This certificate is valid

▼ Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com



Issuer Name	
Country	US
Organization	Google Trust Services
Common Name	Google Internet Authority G3
Serial Number	3495829599616174946
Version	3
Signature Algorithm	SHA-256 with RSA Encryption



Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 D5 63 FC 4D F9 4E 91 ...
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 3F FE 04 7B BE B0 32 1D ...



USERTrust RSA Certification Authority

InCommon RSA Server CA

cse.ucsd.edu



cse.ucsd.edu

Issued by: InCommon RSA Server CA

Expires: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time

✔ This certificate is valid

▼ Details

Subject Name

Country US
Postal Code 92093
State/Province CA
Locality La Jolla
Street Address 9500 Gilman Drive
Organization University of California, San Diego
Organizational Unit UCSD
Common Name cse.ucsd.edu

Issuer Name

Country US
State/Province MI
Locality Ann Arbor
Organization Internet2
Organizational Unit InCommon
Common Name InCommon RSA Server CA

Serial Number 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F E8 98

Version 3

Signature Algorithm SHA-256 with RSA Encryption
(1.2.840.113549.1.1.11)

USERTrust RSA Certification Authority

InCommon RSA Server CA

cse.ucsd.edu



cse.ucsd.edu

Issued by: InCommon RSA Server CA

Expires: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time

✔ This certificate is valid

▼ **Details**

Subject Name

Country US

Postal Code 92093

State/Province CA

Locality La Jolla

Street Address 9500 Gilman Drive

Organization University of California, San Diego

Organizational Unit UCSD

Common Name cse.ucsd.edu

Issuer Name

Country US

State/Province MI

Locality Ann Arbor

Organization Internet2

Organizational Unit InCommon

Common Name InCommon RSA Server CA

Serial Number 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F E8 98

Version 3

Signature Algorithm SHA-256 with RSA Encryption
(1.2.840.113549.1.1.11)

Who are we trusting?

USERTrust RSA Certification Authority

InCommon RSA Server CA

cse.ucsd.edu



cse.ucsd.edu

Issued by: InCommon RSA Server CA

Expires: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time

This certificate is valid

▼ **Details**

Subject Name

Country US

Postal Code 92093

State/Province CA

Locality La Jolla

Street Address 9500 Gilman Drive

Organization University of California, San Diego

Organizational Unit UCSD

Common Name cse.ucsd.edu

Issuer Name

Country US

State/Province MI

Locality Ann Arbor

Organization Internet2

Organizational Unit InCommon

Common Name InCommon RSA Server CA

Serial Number 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F E8 98

Version 3

Signature Algorithm SHA-256 with RSA Encryption
(1.2.840.113549.1.1.11)

Who are we trusting?

Who is this cert for?

Key ID 1E 05 A3 77 8F 6C 96 E2 5B 87 4B A6 B4 86 AC
71 00 0C E7 38

Extension Subject Alternative Name (2.5.29.17)

Critical NO

DNS Name cse.ucsd.edu
DNS Name cs.ucsd.edu
DNS Name www-cs.ucsd.edu
DNS Name www-cse.ucsd.edu
DNS Name www.cs.ucsd.edu
DNS Name www.cse.ucsd.edu

Extension Certificate Policies (2.5.29.32)

Critical NO

Policy ID #1 (1.3.6.1.4.1.5923.1.4.3.1.1)

Qualifier ID #1 Certification Practice Statement (1.3.6.1.5.5.7.2.1)

CPS URI https://www.incommon.org/cert/repository/cps_ssl.pdf

Policy ID #2 (2.23.140.1.2.2)

Extension CRL Distribution Points (2.5.29.31)

Critical NO

URI <http://crl.incommon-rsa.org/InCommonRSAserverCA.cr>

Extension Certificate Authority Information Access
(1.3.6.1.5.5.7.1.1)

Critical NO

Method #1 CA Issuers (1.3.6.1.5.5.7.48.2)

URI http://crt.usertrust.com/InCommonRSAserverCA_2.crt

Method #2 Online Certificate Status Protocol
(1.3.6.1.5.5.7.48.1)

URI <http://ocsp.usertrust.com>

Who is this cert for?

Issuer Name
Country US
State/Province MI
Locality Ann Arbor
Organization Internet2
Organizational Unit InCommon
Common Name InCommon RSA Server CA

Serial Number 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F
E8 98

Version 3

Signature Algorithm SHA-256 with RSA Encryption
(1.2.840.113549.1.1.1)

Parameters None

Not Valid Before Thursday, January 4, 2018 at 4:00:00 PM Pacific
Standard Time

Not Valid After Monday, January 4, 2021 at 3:59:59 PM Pacific
Standard Time

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters None
Public Key 256 bytes : FA F9 1A 08 92 86 9C 7B ...
Exponent 65537
Key Size 2,048 bits
Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes : 6F 62 36 46 B7 43 28 04 ...

Extension Key Usage (2.5.29.15)
Critical YES
Usage Digital Signature, Key Encipherment

CSE's pub key info

Key ID 1E 05 A 77 8F 6C 96 E2 5B 87 4B A6 B4 86 AC
71 00 0C E7 38

Extension Subject Alternative Name (2.5.29.17)

Critical NO

DNS Name cse.ucsd.edu

DNS Name cs.ucsd.edu

DNS Name www-cs.ucsd.edu

DNS Name www-cse.ucsd.edu

DNS Name www.cs.ucsd.edu

DNS Name www.cse.ucsd.edu

Extension Certificate Policies (2.5.29.32)

Critical NO

Policy ID #1 (1.3.6.1.4.1.5923.1.4.3.1.1)

Qualifier ID #1 Certification Practice Statement (1.3.6.1.5.5.7.2.1)

CPS URI https://www.incommon.org/cert/repository/cps_ssl.pdf

Policy ID #2 (2.23.140.1.2.2)

Extension CRL Distribution Points (2.5.29.31)

Critical NO

URI <http://crl.incommon-rsa.org/InCommonRSAServerCA.crl>

Extension Certificate Authority Information Access (1.3.6.1.5.5.7.1.1)

Critical NO

Method #1 CA Issuers (1.3.6.1.5.5.7.48.2)

URI http://crt.usertrust.com/InCommonRSAServerCA_2.crt

Method #2 Online Certificate Status Protocol (1.3.6.1.5.5.7.48.1)

URI <http://ocsp.usertrust.com>

Where we should
check for revocation
information

Revocation

- Problem: keys get compromised
 - Attacker with a key can impersonate you and read messages encrypted to you
- Key expiration helps, but not enough
- CA and PGP PKIs support revocation
 - “I, Alice, revoke my public key . . . do not use it.”
 - Signs revocation with her private key
 - Others can verify Alice’s signature, stop using key

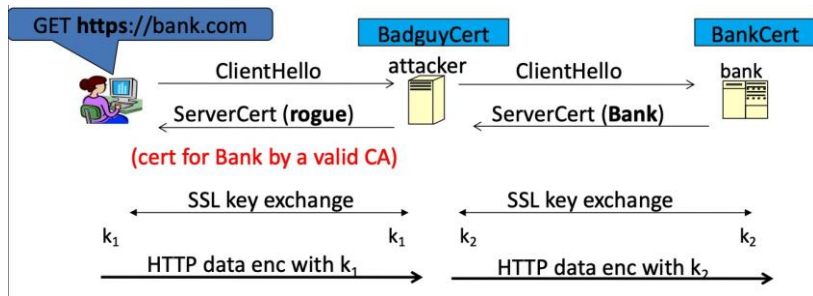
Root CAs on OS X

The screenshot shows the Keychain Access application. The top pane displays the details for the **GlobalSign Root CA** certificate, which is valid and expires on Friday, January 28, 2028 at 4:00:00 AM Pacific Standard Time. The bottom pane shows a list of system root certificates.

Name	Kind	Expires	Keychain
Entrust Root Certification Authority - G2	certificate	Dec 7, 2030 at 9:55:54 AM	System Roots
Entrust.net Certification Authority (2048)	certificate	Dec 24, 2019 at 10:20:51...	System Roots
Entrust.net Certification Authority (2048)	certificate	Jul 24, 2029 at 7:15:12 AM	System Roots
ePKI Root Certification Authority	certificate	Dec 19, 2034 at 6:31:27...	System Roots
Federal Common Policy CA	certificate	Dec 1, 2030 at 8:45:27 AM	System Roots
GeoTrust Global CA	certificate	May 20, 2022 at 9:00:00...	System Roots
GeoTrust Primary Certification Authority	certificate	Jul 16, 2036 at 4:59:59 PM	System Roots
GeoTrust Primary Certification Authority - G2	certificate	Jan 18, 2038 at 3:59:59...	System Roots
GeoTrust Primary Certification Authority - G3	certificate	Dec 1, 2037 at 3:59:59 PM	System Roots
Global Chambersign Root	certificate	Sep 30, 2037 at 9:14:18...	System Roots
Global Chambersign Root - 2008	certificate	Jul 31, 2038 at 6:31:40 AM	System Roots
GlobalSign	certificate	Mar 18, 2029 at 3:00:00...	System Roots
GlobalSign	certificate	Jan 18, 2038 at 7:14:07 PM	System Roots
GlobalSign	certificate	Jan 18, 2038 at 7:14:07 PM	System Roots
GlobalSign	certificate	Dec 15, 2021 at 12:00:00...	System Roots
GlobalSign Root CA	certificate	Jan 28, 2028 at 4:00:00...	System Roots
Go Daddy Class 2 Certification Authority	certificate	Jun 29, 2034 at 10:06:20...	System Roots
Go Daddy Root Certification Authority - G2	certificate	Dec 31, 2037 at 3:59:59...	System Roots
Government Root Certification Authority	certificate	Dec 31, 2037 at 7:59:59...	System Roots
Hellenic Academic and Research Institutions RootCA 2011	certificate	Dec 1, 2031 at 5:49:52 AM	System Roots
Hongkong Post Root CA 1	certificate	May 14, 2023 at 9:52:29...	System Roots
IdenTrust Commercial Root CA 1	certificate	Jan 16, 2034 at 10:12:23...	System Roots
IdenTrust Public Sector Root CA 1	certificate	Jan 16, 2034 at 9:53:32...	System Roots
ISRG Root X1	certificate	Jun 4, 2035 at 4:04:38 AM	System Roots
izenpe.com	certificate	Dec 13, 2037 at 12:27:25...	System Roots
izenpe.com	certificate	Dec 13, 2037 at 12:27:25...	System Roots
KISA RootCA 1	certificate	Aug 24, 2025 at 1:05:46...	System Roots

Which CA can issue a certificate for fbi.gov?
Which CA can issue a certificate for google.com?

Man-in-the-middle attack using rogue cert



Attacker proxies data between user and bank.

Sees all traffic and can modify data at will.

CA Hacks and Vulnerabilities

There is a long history of CAs getting hacked or doing the wrong thing.

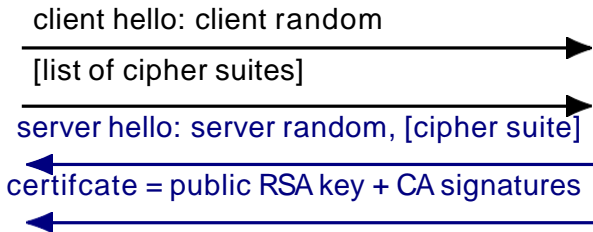
- 2011: Comodo and DigiNotar CAs hacked, used to issue fraudulent certificates for Hotmail, Gmail, Skype, Yahoo Mail, Firefox...
 - Fraudulent certificates later used in man-in-the-middle attack against Iran.
- 2013: TurkTrust issued fraudulent certificate for Gmail.
- 2014: Indian NIC issue certs for Google and Yahoo!
- 2016: WoSign issues cert for GitHub.

Mitigations:

- Certificate pinning.
 - Hard code certificates for some sites in browser.
- Certificate Transparency.
 - Public append-only log of certificate issuances to track fraudulent certs.

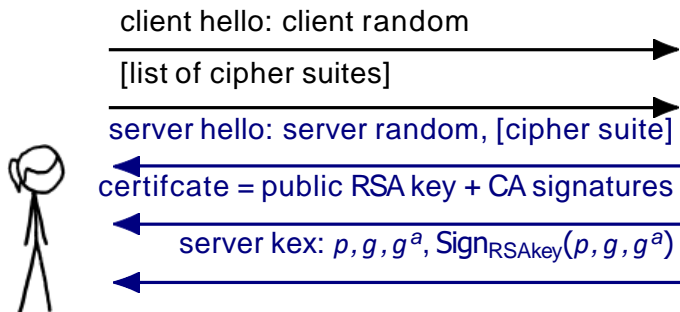
TLS 1.2 with Diffie-Hellman Key Exchange

Step 3: The server sends over its certificate which contains the server's public key and signatures from a certificate authority.



TLS 1.2 with Diffie-Hellman Key Exchange

Step 4: The server initiates a Diffie-Hellman key exchange.

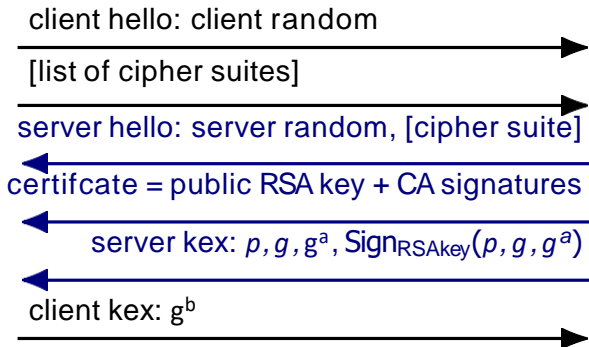


To protect against man-in-the-middle attacks, the server uses its public key to sign the Diffie-Hellman key exchange.

TLS also allows client authentication, but this is rare.

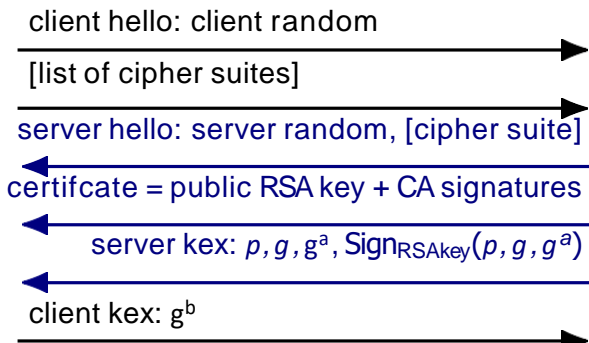
TLS 1.2 with Diffie-Hellman Key Exchange

Step 5: The client responds with its half of the Diffie-Hellman key exchange.



TLS 1.2 with Diffie-Hellman Key Exchange

Step 6: The client and server derive symmetric encryption keys from the shared secret using a key derivation function.

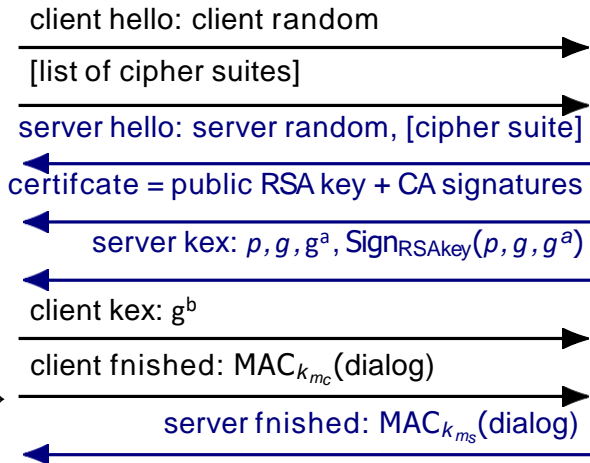


$KDF(g^{ab}, \text{random}) \rightarrow k_{m_c}, k_{m_s}, k_e$

$KDF(g^{ab}, \text{random}) \rightarrow k_{m_c}, k_{m_s}, k_e$

TLS 1.2 with Diffie-Hellman Key Exchange

Step 7: The client and server verify the integrity of the handshake using the MAC keys they have derived.



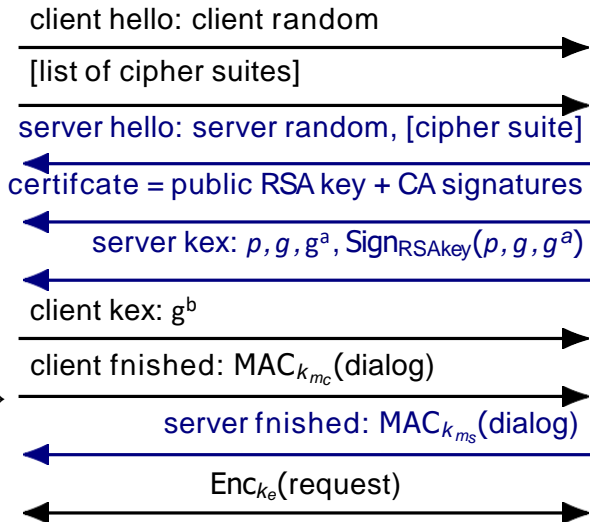
$\text{KDF}(g^{ab},$
random) \rightarrow
 k_{m_c}, k_{m_s}, k_e



$\text{KDF}(g^{ab},$
random) \rightarrow
 k_{m_c}, k_{m_s}, k_e

TLS 1.2 with Diffie-Hellman Key Exchange

Step 8: The client and server can now send encrypted application data (e.g. HTTP) using their secure channel.



$\text{KDF}(g^{ab}, \text{random}) \rightarrow$
 k_{m_c}, k_{m_s}, k_e

$\text{KDF}(g^{ab}, \text{random}) \rightarrow$
 k_{m_c}, k_{m_s}, k_e

TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).

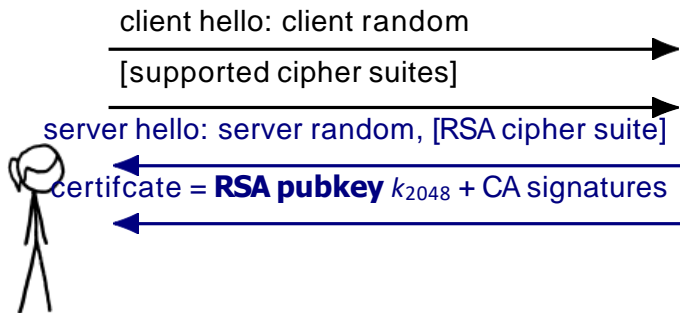
client hello: client random

[supported cipher suites]



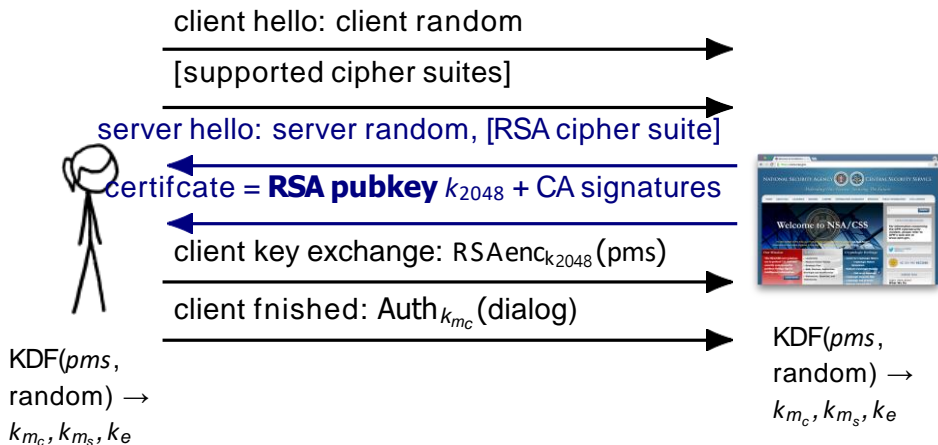
TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).



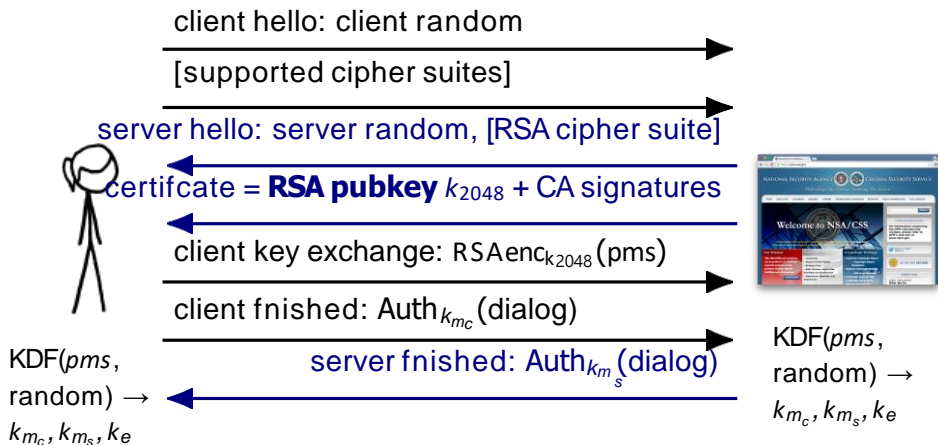
TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).



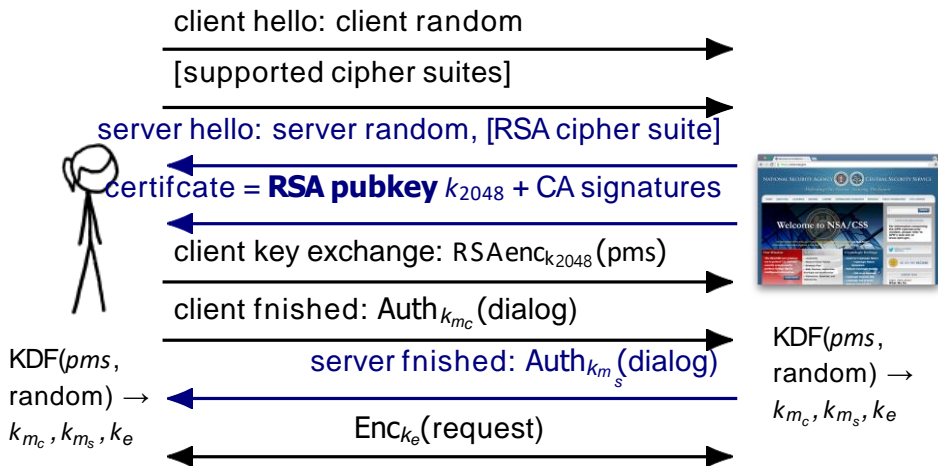
TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).



TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).



How TLS achieves its security goals

- **What happens if a passive eavesdropper watches all the traffic?**

How TLS achieves its security goals

- **What happens if a passive eavesdropper watches all the traffic?**
 - The application-layer traffic is encrypted, and Diffie-Hellman and RSA are secure against a passive eavesdropper so the attacker cannot discover the keys.
 - The eavesdropper can see all the IP and TCP-layer packet headers.
 - The eavesdropper can also see the initial handshake and metadata (which includes the server certificate)

How TLS achieves its security goals

- **What happens if an active attacker tries to man-in-the-middle the connection?**

How TLS achieves its security goals

- **What happens if an active attacker tries to man-in-the-middle the connection?**
 - For Diffie-Hellman, the key exchange is digitally signed by the private key corresponding to the public key in the server's certificate and the attacker doesn't know the server's key, so they cannot forge the signature. The client will not accept the key exchange.
 - For RSA, the attacker does not know the private key corresponding to the public key in the server's certificate, so cannot learn the client's choice of premaster secret to learn the session keys.

How TLS achieves its security goals

- **What happens if a network attacker tries to impersonate the server?**

How TLS achieves its security goals

- **What happens if a network attacker tries to impersonate the server?**
 - For Diffie-Hellman, the attacker does not know the private key corresponding to the public key in the server's certificate, so they cannot generate a valid signature on their Diffie-Hellman key exchange that will be accepted by the client.
 - For RSA the attacker does not know the server's private key so cannot decrypt the client's encrypted premaster secret message.

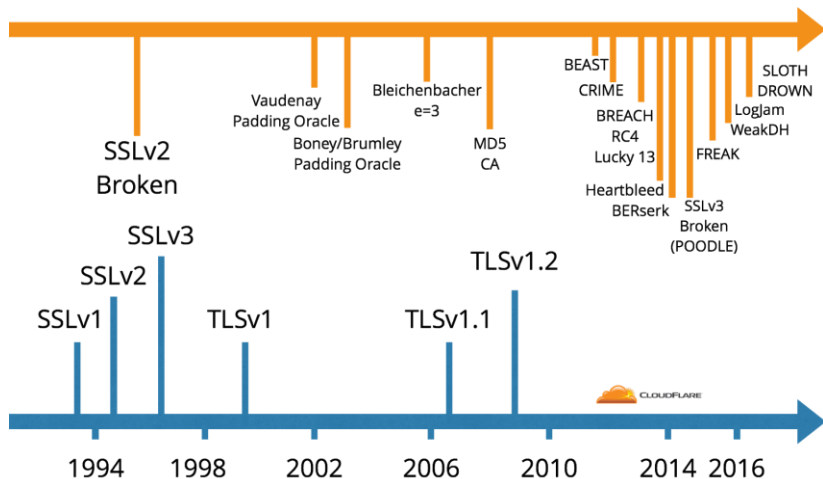
What if a private key gets stolen or compromised?

If an adversary obtains a server certificate private key:

- With Diffie-Hellman key exchange, the adversary can:
 - actively man-in-the-middle a connection.
 - impersonate the server to anyone.
- With RSA key exchange, the adversary can:
 - impersonate the server to anyone.
 - decrypt any traffic from now and any point in the past.

TLS v. 1.2 and below have had a lot of vulnerabilities

- Early versions of SSL developed before cryptographic protocol design was fully understood.
- Later protocol versions retained insecure options for backwards compatibility.



TLS 1.3 is the new standard

Developed over several years as a collaboration between cryptographers from industry and academia.

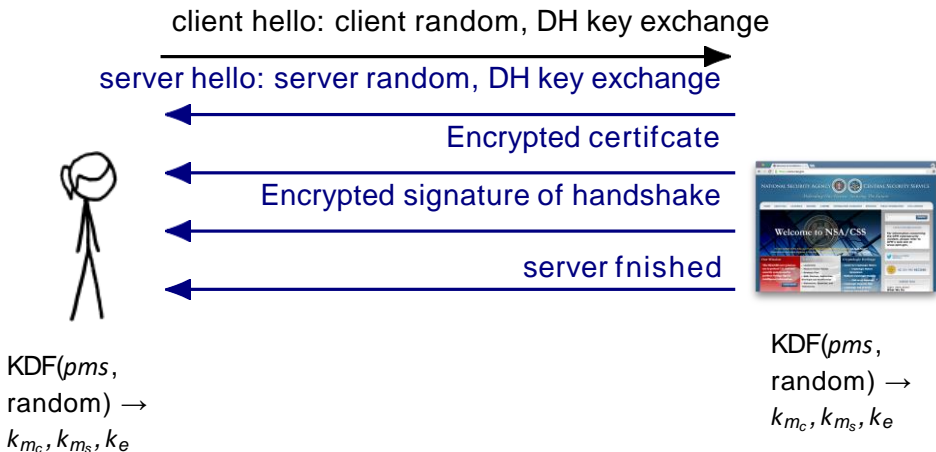
Standardized August 2018 by IETF.

Major differences from TLS 1.2 and below:

- RSA key exchange removed.
 - Protects against passive decryption attacks.
- Only secure Diffie-Hellman parameters allowed.
 - Protects against attacks exploiting bad choices of parameters.
- Handshake encrypted immediately after key exchange.
 - Limits the amount of metadata visible to a passive eavesdropper.
- Protocol downgrade protection.
 - Protects against protocol being downgraded to prior insecure versions.

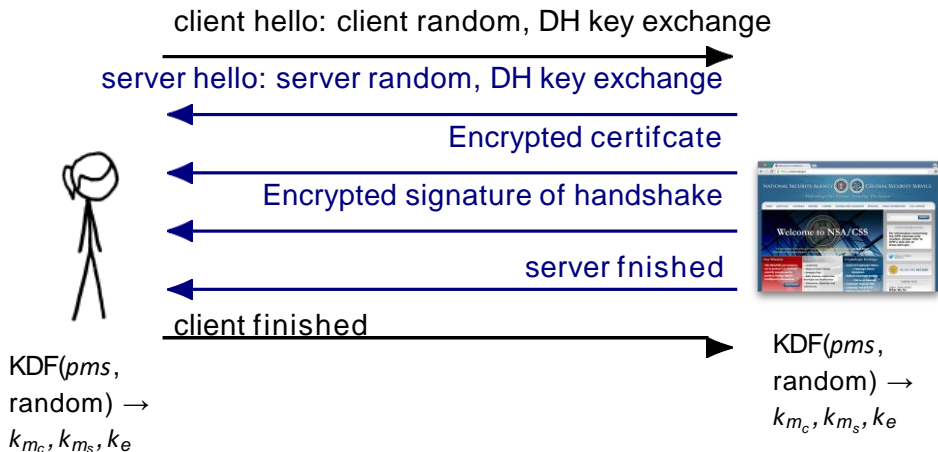
TLS 1.3

TLS 1.3 encrypts the handshake immediately after doing a Diffie-Hellman key exchange.



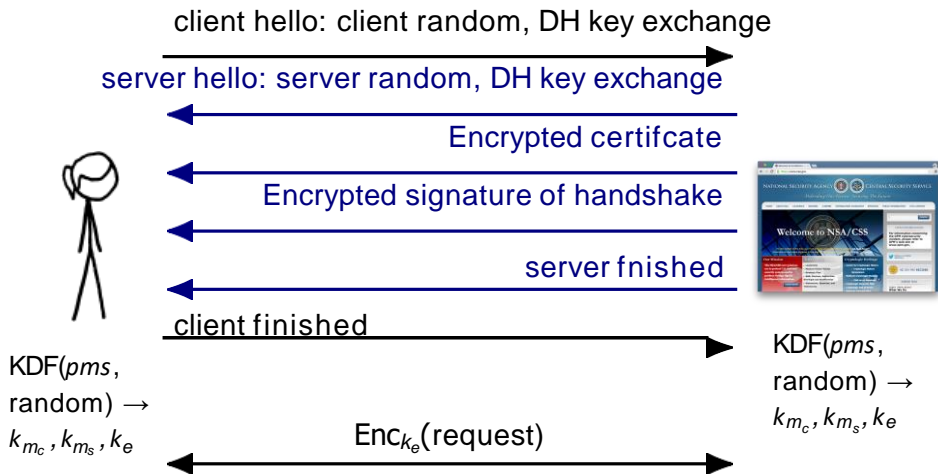
TLS 1.3

TLS 1.3 encrypts the handshake immediately after doing a Diffie-Hellman key exchange.



TLS 1.3

TLS 1.3 encrypts the handshake immediately after doing a Diffie-Hellman key exchange.



TLS 1.3 deployment difficulties

TLS 1.3 deployment is slower than it should be, but now \approx 63% of TLS traffic (f5 labs).

Major reasons:

- HTTPS proxies extremely common in industry.
- Many of them rely on RSA key exchange to make passive decryption and traffic analysis easier.
- Removing RSA key exchange breaks all these boxes.
- Man-in-the-middle hardware is also quite common.
- Bad implementations have hard-coded values like TLS versions and there is no way to update them.

END: Cryptography and TLS

Read more: TLS key theft and other risks in the wild

<https://www.theguardian.com/world/2013/oct/03/lavabit-ladar-levison-fbi-encryption-keys-snowden>

Lavabit employed two stages of encryption for its paid subscribers: storage encryption and transport encryption. Storage encryption protects emails and other data that rests on Lavabit's servers. Theoretically, no person other than the email user could access the data once it was so encrypted. By using storage encryption, Lavabit held a unique market position in the email industry, as many providers do not encrypt stored data.

YOU ARE COMMANDED to appear and testify before the United States district court at the time, date, and place shown below to testify before the court's grand jury. When you arrive, you must remain at the court until the judge or a court officer allows you to leave.

Place: UNITED STATES DISTRICT COURT 401 Courthouse Square Alexandria, Virginia 22314	Date and Time: July 16, 2013 9:30 AM
--	--------------------------------------

You must also bring with you the following documents, electronically stored information, or objects (blank if not applicable):


In addition to your personal appearance, you are directed to bring to the grand jury the public and private encryption keys used by lavabit.com in any SSL (Secure Socket Layer) or TLS (Transport Security Layer) sessions, including HTTPS sessions with clients using the lavabit.com web site and encrypted SMTP communications (or Internet communications using other protocols) with mail servers;

Any other information necessary to accomplish the installation and use of the pen/trap device ordered by Judge Buchanan on June 28, 2013, unobtrusively and with minimum interference to the services that are accorded persons with respect to whom the installation and use is to take place;

If such information is electronically stored or unable to be physically transported to the grand jury, you may provide a copy of the information to the Federal Bureau of Investigation. Provision of this information to the FBI does not excuse your personal appearance.

Date: July 11, 2013

CLERK OF COURT


Signature of the Clerk or Deputy Clerk

UNDER SEAL

UNITED STATES DISTRICT COURT

for the
Eastern District of Virginia

In the Matter of the Search of
(Briefly describe the property to be searched
or identify the person by name and address)
INFORMATION ASSOCIATED WITH
[REDACTED]
THAT IS STORED AT PREMISES
CONTROLLED BY LAVABIT, LLC

)
)
)
)
)
)
)

Case No. 1:13SW522

SEARCH AND SEIZURE WARRANT

To: Any authorized law enforcement officer

An application by a federal law enforcement officer or an attorney for the government requests the search of the following person or property located in the Northern District of Texas

(identify the person or describe the property to be searched and give its location):
See Attachment A

ATTACHMENT B

Particular Things to be Seized

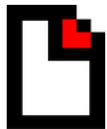
I. Information to be disclosed by Lavabit, LLC (the "Provider")

To the extent that the information described in Attachment A is within the possession, custody, or control of the Provider, including any emails, records, files, logs, or information that has been deleted but is still available to the Provider, the Provider is required to disclose the following information to the government for each account or identifier listed in Attachment A:

- a. All information necessary to decrypt communications sent to or from the Lavabit e-mail account [REDACTED] including encryption keys and SSL keys;
- b. All information necessary to decrypt data stored in or otherwise associated with the Lavabit account [REDACTED]

Despite the unequivocal language of the August 1 Order, Lavabit dallied and did not comply. Just before the 5:00 pm August 2 deadline, for instance, Levison provided the FBI with an 11-page printout containing largely illegible characters in 4-point type, which he represented to be Lavabit's encryption keys. The Government instructed Lavabit to provide the keys in an industry-standard electronic format by the morning of August 5. Lavabit did not respond.

August 2013



Lavabit

My Fellow Users,

I have been forced to make a difficult decision: to become complicit in crimes against the American people or walk away from nearly ten years of hard work by shutting down Lavabit. After significant soul searching, I have decided to suspend operations. I wish that I could legally share with you the events that led to my decision. I cannot. I feel you deserve to know what's going on--the first amendment is supposed to guarantee me the freedom to speak out in situations like this. Unfortunately, Congress has passed laws that say otherwise. As things currently stand, I cannot share my experiences over the last six weeks, even though I have twice made the appropriate requests.

What's going to happen now? We've already started preparing the paperwork needed to continue to fight for the Constitution in the Fourth Circuit Court of Appeals. A favorable decision would allow me resurrect Lavabit as an American company.

This experience has taught me one very important lesson: without congressional action or a strong judicial precedent, I would strongly recommend against anyone trusting their private data to a company with physical ties to the United States.

Sincerely,
Ladar Levison
Owner and Operator, Lavabit LLC



The server's security certificate is revoked!

You attempted to reach **lavabit.com**, but the certificate that the server presented has been revoked by its issuer. This means that the security credentials the server presented absolutely should not be trusted. You may be communicating with an attacker.

[Back to safety](#)

▶ [Help me understand](#)

A CRYPTO NERD'S IMAGINATION:

HIS LAPTOP'S ENCRYPTED.
LET'S BUILD A MILLION-DOLLAR
CLUSTER TO CRACK IT.



WHAT WOULD ACTUALLY HAPPEN:

HIS LAPTOP'S ENCRYPTED.
DRUG HIM AND HIT HIM WITH
THIS \$5 WRENCH UNTIL
HE TELLS US THE PASSWORD.



<https://xkcd.com/538/>

“Actual actual reality: nobody cares about his secrets. Also, I would be hard-pressed to find that wrench for \$5.”

The “crypto wars” and the historical development of TLS.

International Traffic in Arms Regulations

April 1, 1992 version

Category XIII--Auxiliary Military Equipment ...

(b)Information Security Systems and equipment, cryptographic devices, software, and components specifically designed or modified therefore, including:

(1)Cryptographic (including key management) systems, equipment, assemblies, modules, integrated circuits, components or software with the capability of maintaining secrecy or confidentiality of information or information systems, except cryptographic equipment and software as follows:

(i)Restricted to decryption functions specifically designed to allow the execution of copy protected software, provided the decryption functions are not user-accessible.

(ii)Specially designed, developed or modified for use in machines for banking or moneytransactions, and restricted to use only in such transactions. Machines for banking or moneytransactions include automatic teller machines, self-service statement printers, point of sale terminals or equipment for the encryption of interbanking transactions.

...

Timeline of US cryptography export control

- Pre-1994: Encryption software requires individual export license as a munition.
- 1994: US State Department amends ITAR regulations to allow export of approved software to approved countries without individual licenses. 40-bit symmetric cryptography was understood to be approved under this scheme.
- 1995: Netscape develops initial SSL protocol. Includes weakened “export” cipher suites.
- 1996: Bernstein v. United States; California judge rules ITAR regulations are unconstitutional because “code is speech”
- 1996: Cryptography regulation moved to Department of Commerce.
- 1999: TLS 1.0 standardized. Includes weakened “export” cipher suites.
- 2000: Department of Commerce loosens regulations on mass-market and open source software.

Commerce Control List: Category 5 - Info. Security

(May 23, 2019 version)

aAsymmetric algorithm employing a key length in excess of 56-bits; not including parity bits; or

bAn asymmetric algorithm where the security of the algorithm is based on any of the following:

1. Factorization of integers in excess of 512 bits (e.g., RSA);

2. Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits (e.g., Diffie-Hellman over Z/pZ); or

3. Discrete logarithms in a group other than mentioned in 5A002.a.1.b.2 in excess of 112 bits (e.g., Diffie-Hellman over an elliptic curve);

...

a. Designed or modified to perform 'cryptanalytic functions.'

Commerce Control List: Category 5 - Info. Security

(May 23, 2019 version)

cAn “asymmetric algorithm” where the security of the algorithm is based on any of the following:

1.Shortest vector or closest vector

problems associated with lattices (e.g., NewHope, Frodo, NTRUEncrypt, Kyber, Titanium);

2.Finding isogenies between Supersingular elliptic curves (e.g., Supersingular Isogeny Key Encapsulation); or

3.Decoding random codes (e.g., McEliece, Niederreiter).

Technical Note: An algorithm described by Technical Note 2.c. may be referred to as being post-quantum, quantum-safe or quantum-resistant.

US Politicians on Cryptography

“The government must be wary of suffocating [the encryption software] industry with regulation in the new digital age, but we must be able to strike a balance between the legitimate concerns of the law enforcement community and the needs of the marketplace.” — U.S. Vice President Al Gore, September 1997

“Because, if, in fact, you can’t crack that [encryption] at all, government can’t get in, then everybody is walking around with a Swiss bank account in their pocket – right? So there has to be some concession to the need to be able to get into that information somehow.” — President Obama, March 2016

“To think that Apple won’t allow us to get into her cellphone? Who do they think they are?” — US Presidential Candidate Trump, 2016

Deliberately weakened cryptography in TLS

- SSLv2, SSLv3, and TLS 1.0 included options for weakened cryptography to comply with US export control in the 90s.
- Browsers outside the US were supposed to request weakened cryptography, and those in the US were allowed to request normal strength cryptography.
- Browsers were updated long ago to never request these weakened options once US regulations changed.
- Even though the political situation changed, many servers never removed these options.
- 2015–2016: A series of academic, mostly impractical attacks (FREAK, Logjam, DROWN) show that even current browsers at the time could be vulnerable.