Transport Layer Security (TLS)

CSIE 7190 Cryptography and Network Security, Spring 2018

https://ceiba.ntu.edu.tw/1062csie_cns

cns@csie.ntu.edu.tw

Hsu-Chun Hsiao

Housekeeping

5/17: HW2 due

5/27: Reading critique #6 due

Reading critique #6

Write a critique on:

- Jackson and A. Barth, "ForceHTTPS: Protecting High-Security Web Sites from Network Attacks," in *WWW*, 2008.
  * Also check SSL Stripping & HTTP Strict Transport Security (HSTS)

Text only, one page

工商服務時間

演講
Title: Towards mitigating data security risks in JD cloud environments
Date: 2018-05-25 (Fri.) 14:20-15:30
Location: R103, CSIE
Speaker: Dr. Yueh-Hsun Lin, JD Silicon Valley R&D Center, USA
link: https://www.csie.ntu.edu.tw/app/news.php?Sn=13759

演講
Title: Internet-of-Things Attacks in Practice and Reconfigurable Security based on Edge Computing
Date: 2018-05-28 (Mon.), 13:30-16:20
Location: R210, CSIE
Speaker: Dr. RUEI-HAU HSU, Scientist, Data Storage Institute, Agency for Science, Technology and Research (A*STAR), Singapore
Link: https://www.csie.ntu.edu.tw/app/news.php?Sn=13779

AIS3暑期資安課程 (Advanced Information Security Summer School)
Registration deadline: 2018-5-27
Pre-exam: 2018-6-1 to 2018-6-3
Course: 2018-7-30 to 2018-8-5
Link: https://ais3.org/

Agenda

SSL/TLS overview

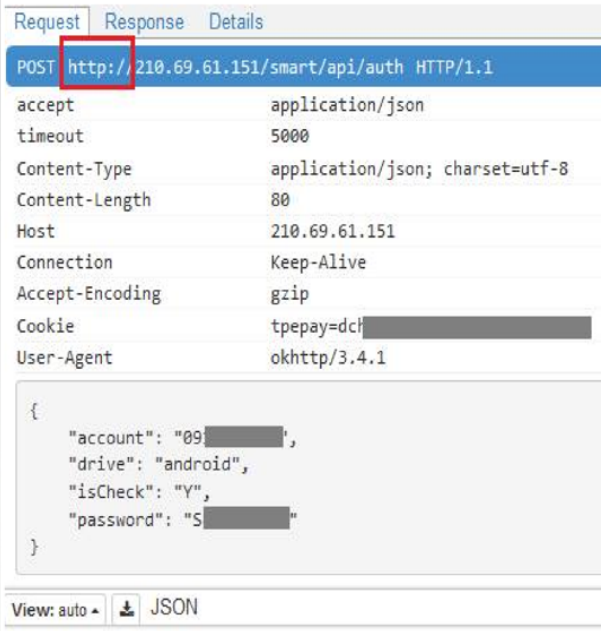
Known issues in SSL/TLS

Attack examples

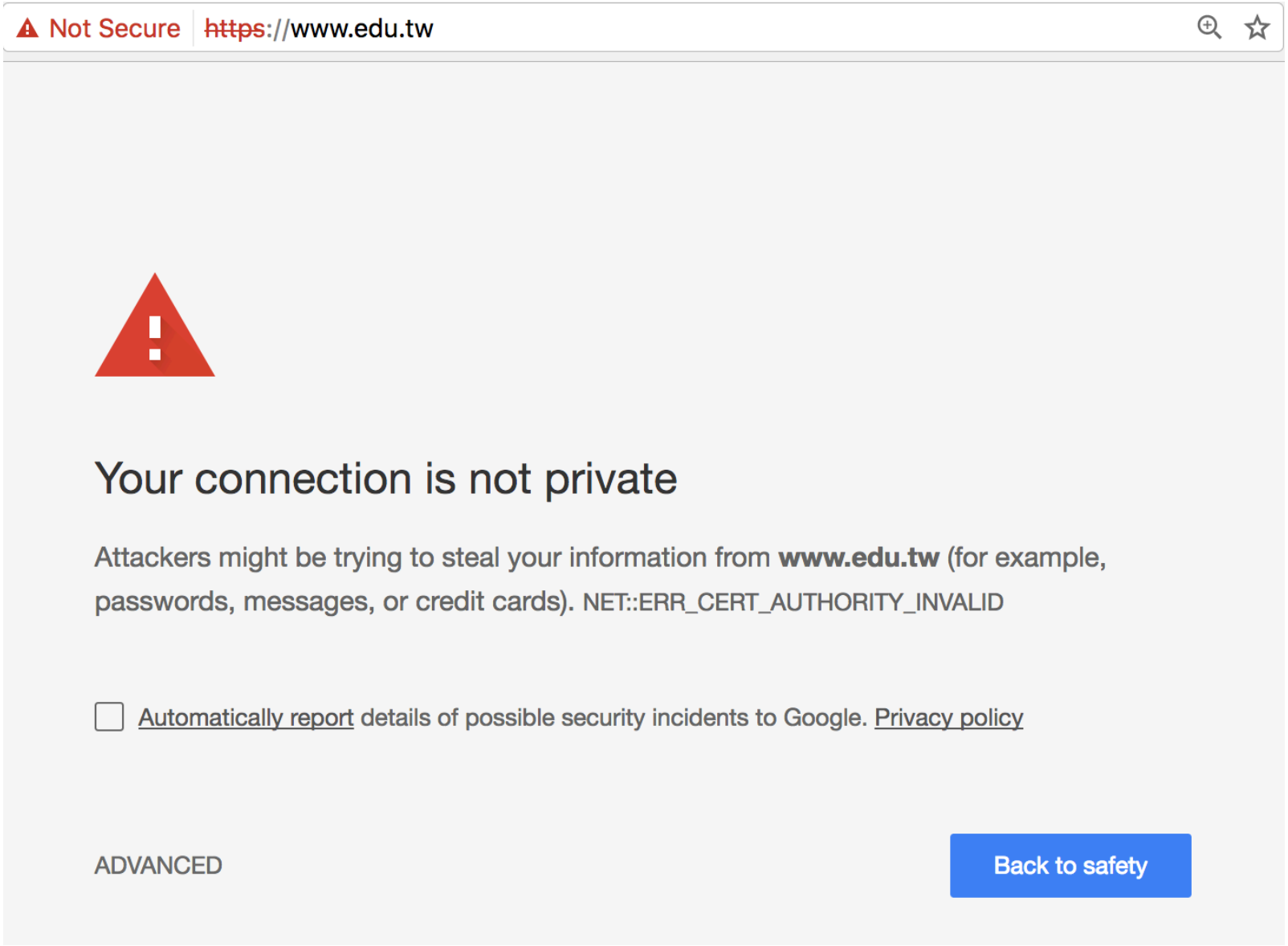
SSL/TLS Overview

這個網站安全嗎？


台北市政府智慧支付平台
方便、快速、智慧
網路市民帳號
網路市民密碼
登入
註冊
臺北市政府
Taipei City Government
ver 0.9.0


Request
Response
Details
POST http://210.69.61.151/smart/api/auth HTTP/1.1
accept application/json
timeout 5000
Content-Type application/json; charset=utf-8
Content-Length 80
Host 210.69.61.151
Connection Keep-Alive
Accept-Encoding gzip
Cookie tpepay=dc
User-Agent okhttp/3.4.1
{
"account": "09
"drive": "android",
"isCheck": "Y",
"password": "S
}
View: auto
JSON

這個網站安全嗎？


Not Secure
https://www.edu.tw
Your connection is not private
Attackers might be trying to steal your information from www.edu.tw (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID
Automatically report details of possible security incidents to Google. Privacy policy
ADVANCED
Back to safety

這個網站安全嗎？


https://my.ntu.edu.tw
請使用計中帳號登入！
SSO1.3
登入
※ 預防帳號遭盜用，請定期修改密碼！
National Taiwan University
myNTU
搜尋
學生專區
個人資訊 >
課務資訊 >
生活資訊 >
助學資訊 >
社團活動資訊 >
畢業生資訊 >
課程學習
選課專區
停修課程網路申請系統
臺大實習計畫
more ...
教職申辦
自然人憑證簽到退
薪資入帳變更申請
新進人員健檢系統
more ...
十大熱門服務 >
成績查詢
到勤差假申請/簽核
活動報名
學生請假
滿意度問卷調查
學士班修課檢視表

SSL Report: my.ntu.edu.tw (140.112.8.80)

Assessed on: Sun Apr 26 05:25:50 PDT 2015 | **HIDDEN** | Clear cache

Scan Another »

Summary

SSL Report: my.ntu.edu.tw (140.112.8.80)

Assessed on: Wed, 04 May 2016 13:02:29 UTC | Hide | Clear cache

Scan Another

Summary

SSL Report: my.ntu.edu.tw (140.112.8.80)

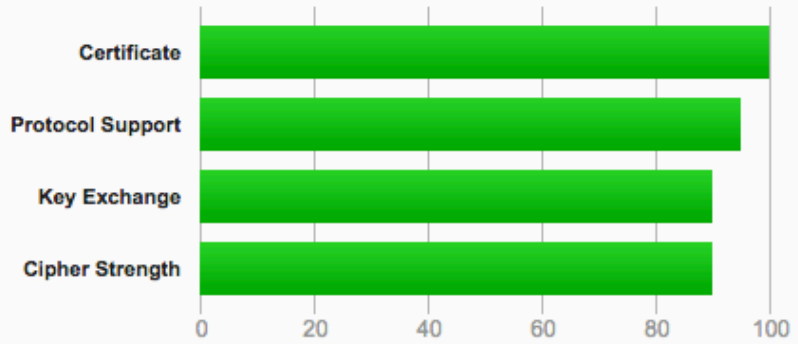
Assessed on: Tue, 08 May 2018 04:55:37 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

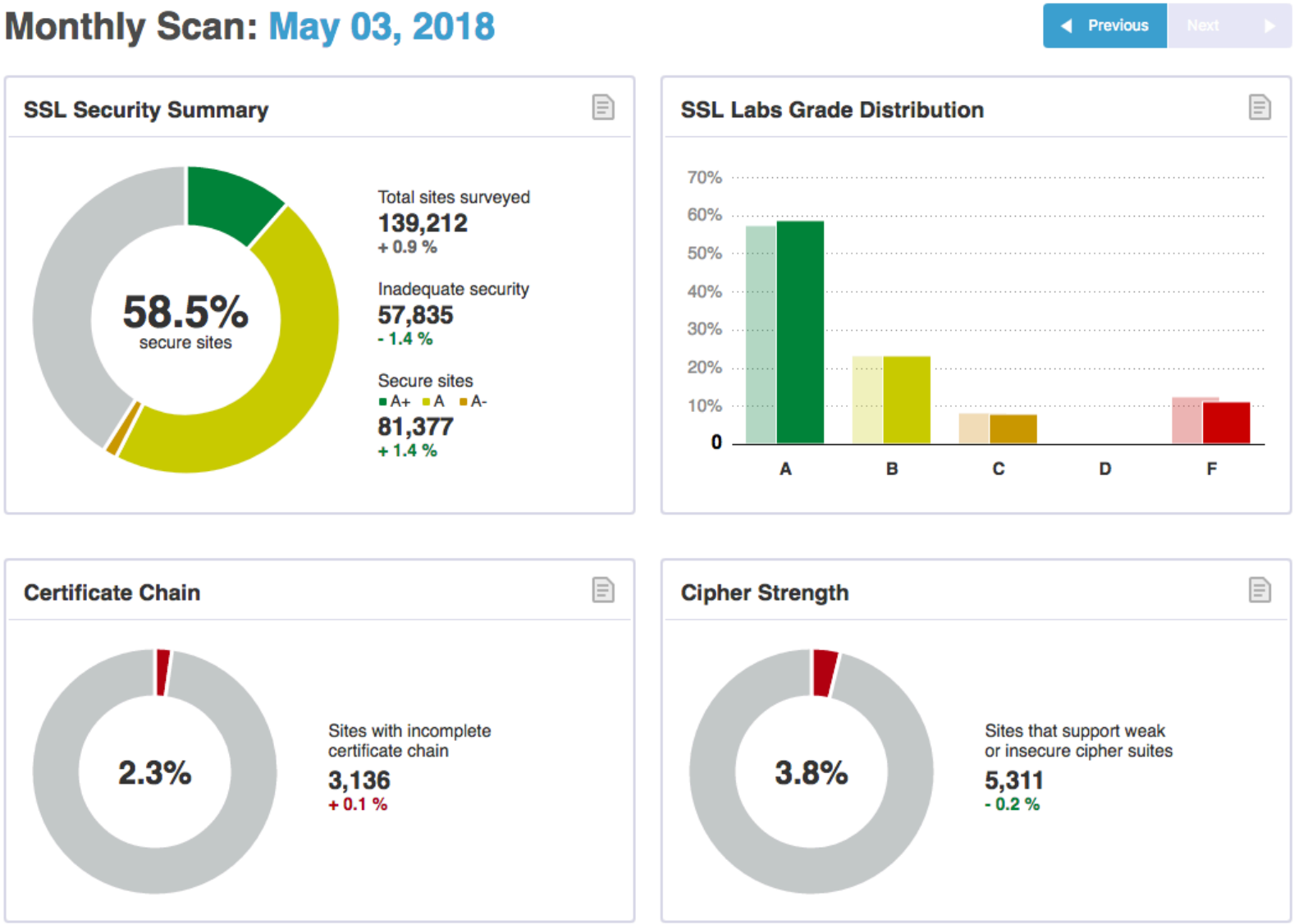
A



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server's certificate is not trusted by Java trust store (see below for details).

https://www.ssllabs.com/ssltest



https://www.ssllabs.com

SSL/TLS overview

Goal: building a secure end-to-end channel

- SSL = Secure Sockets Layer (predecessor)
- TLS = Transport Layer Security (standard)
- HTTPS = HTTP over SSL/TLS

HTTPS is a dominant protocol for securing HTTP communication

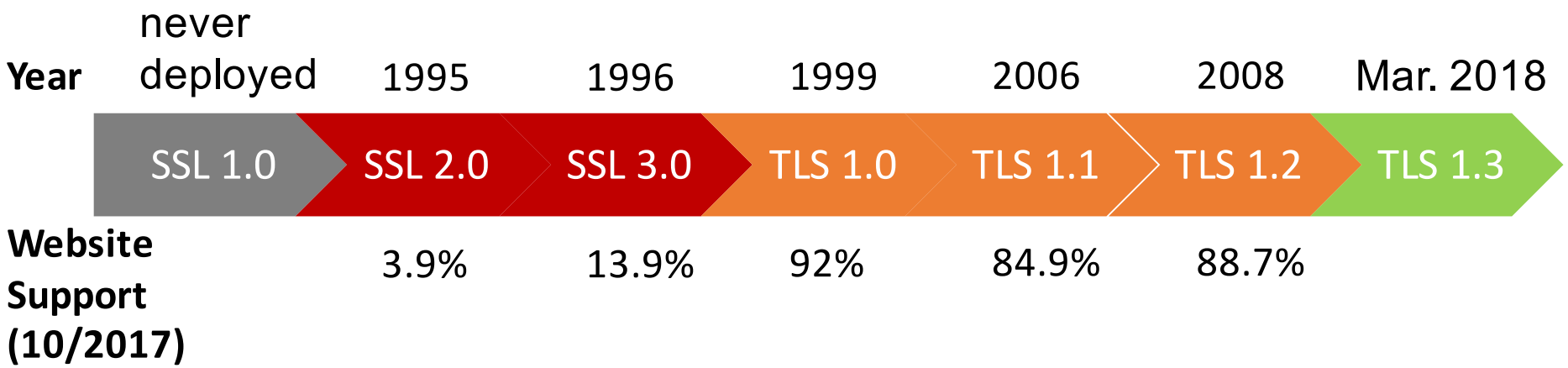
SSL/TLS overview

Security requirements

- Confidentiality
- Integrity
- Authentication (mostly server authentication, client authentication in TLS is rare)

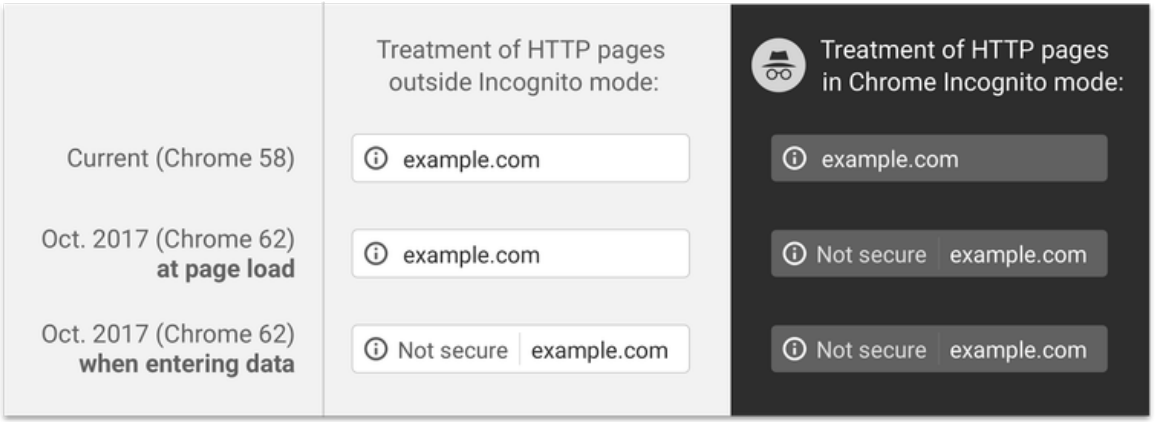
=> prevents a man-in-the-middle attacker from eavesdropping, manipulation, or impersonation

SSL/TLS history

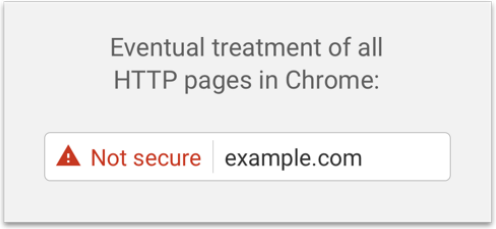


* SSL 2.0 and 3.0 are disabled by default or not supported anymore for Chrome after v40, Firefox after v34, IE 11, Android 5.1, OSX 10.11.
* IE6 only supports SSL 2.0 and 3.0.

Google Chrome and Firefox marked HTTP as a non-secure protocol



"Eventually, we plan to label all HTTP pages as non-secure, and change the HTTP security indicator to the red triangle that we use for broken HTTPS."



White House memorandum

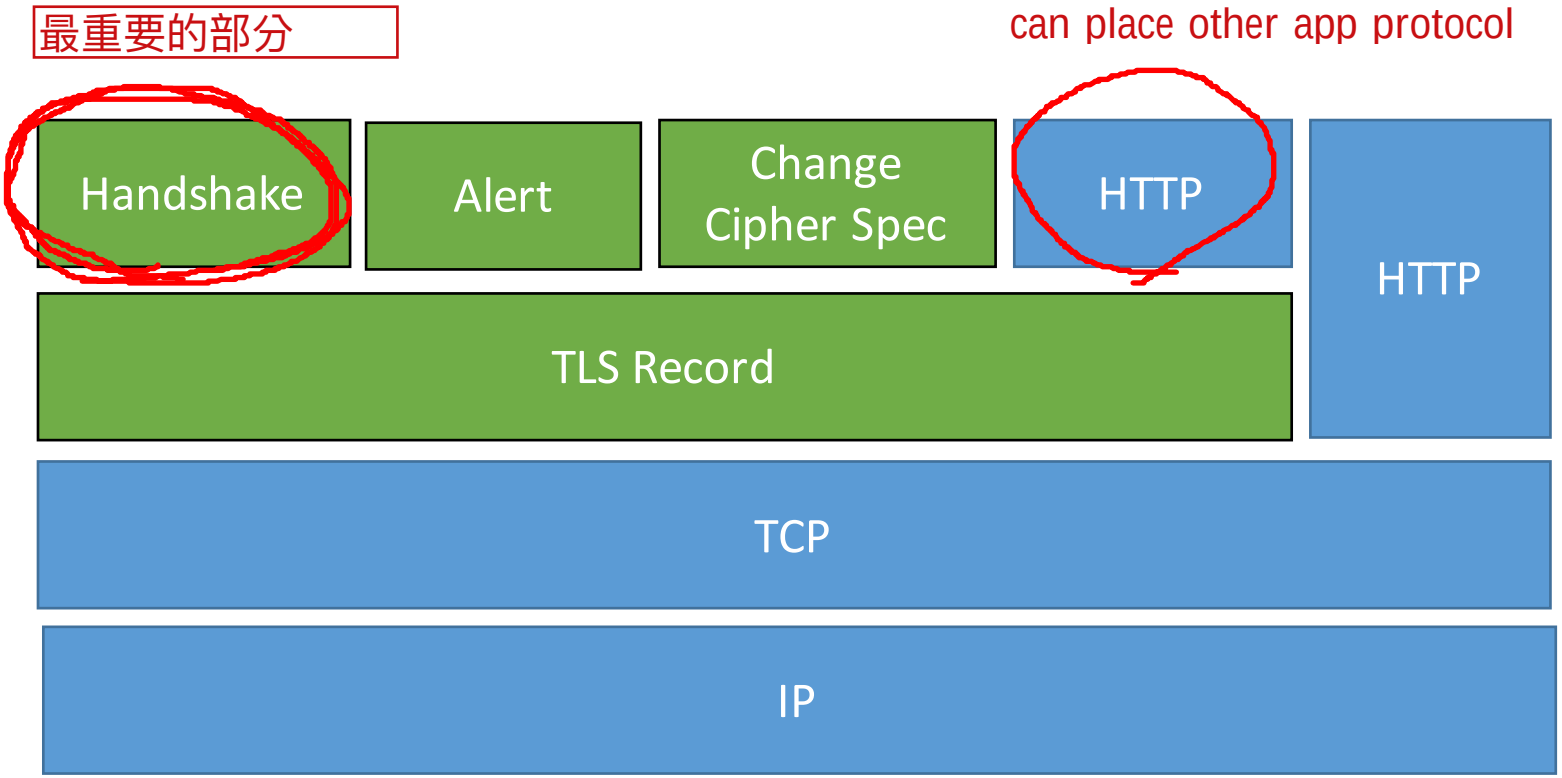
The HTTPS-Only Standard

The American people expect government websites to be secure and their interactions with those websites to be private.

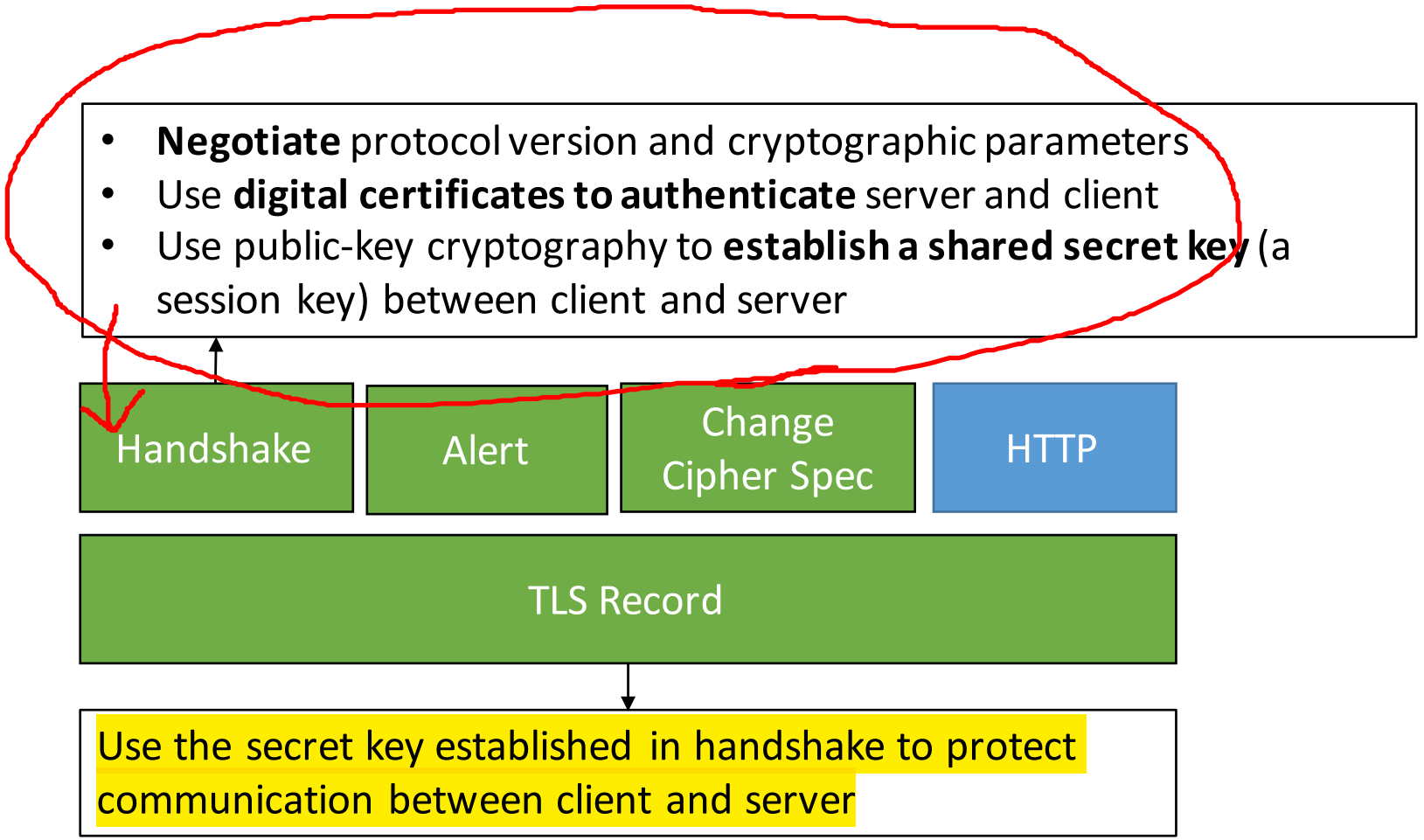
This site contains a web-friendly version of the White House Office of Management and Budget memorandum M-15-13, **"A Policy to Require Secure Connections across Federal Websites and Web Services"**, and provides technical guidance and best practices to assist in its implementation.

Agencies must make all existing websites and services accessible through a secure connection [3] (HTTPS-only, with HSTS) by December 31, 2016.

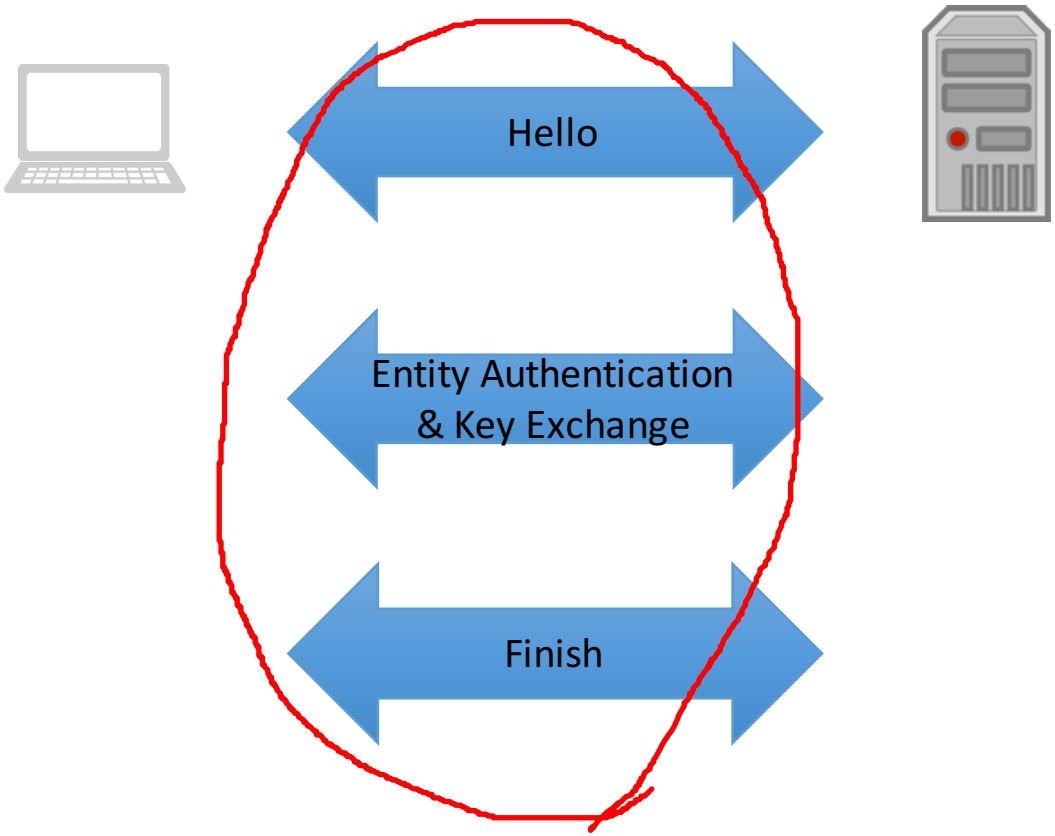
TLS overview



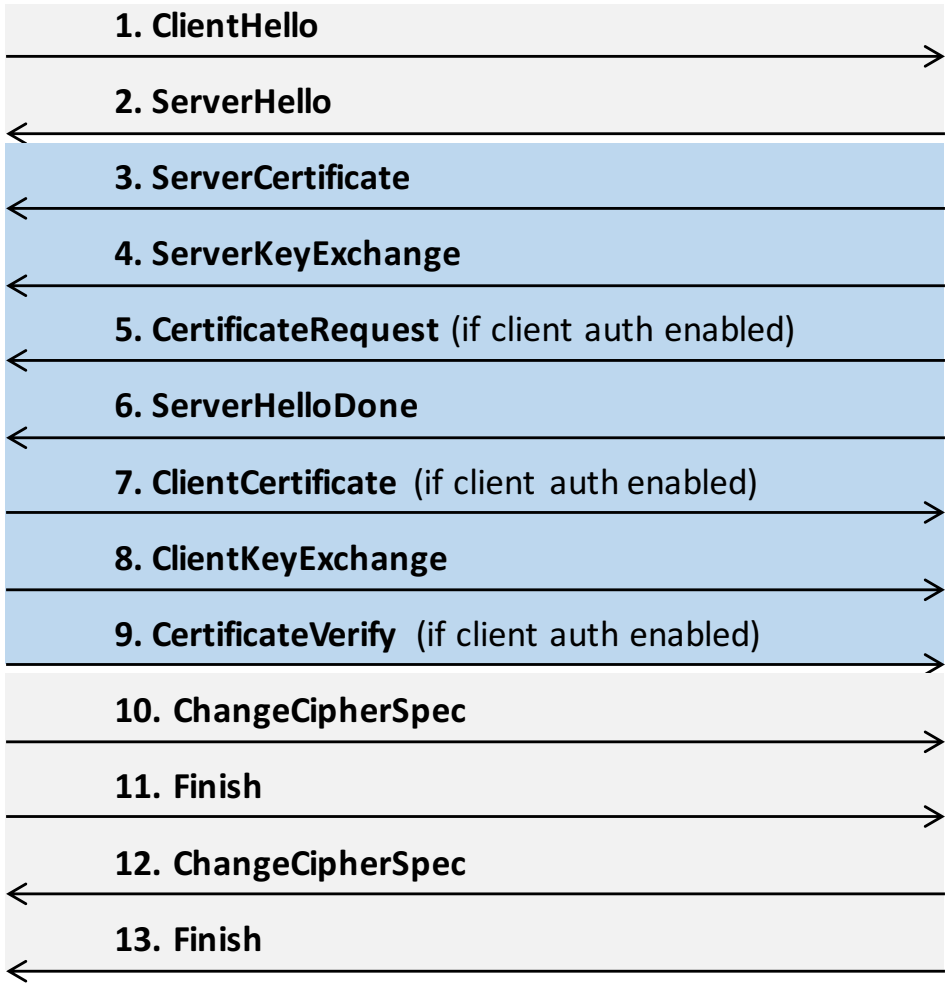
TLS overview

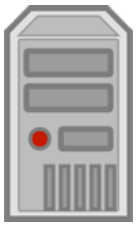


TLS Handshake Protocol (Simplified)



TLS Handshake Protocol





TLS Handshake: Client Hello & Server Hello

Client announces the **highest supported protocol version** and **supported cryptographic algorithms** (a.k.a. **cipher suite**) in decreasing order of preference

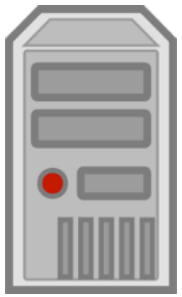
Server responses with strongest protocol version and algorithm supported by both client and server

The exchange is in plaintext

- Need to check integrity at the end

1. ClientHello: I can support these

2. ServerHello: ok let's pick this one



TLS Handshake: Client Hello & Server Hello

pv_{max}: highest supported protocol version

CR: client's random number

pv: selected protocol version

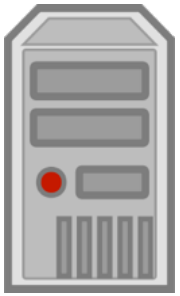
SR: server's random number

sid: session ID

CR, SR用來產生shared secret key

1. ClientHello: pv_{max}, CR, CipherSuiteList

2. ServerHello: pv, SR, sid, CipherSuite



Cipher Suite

Cipher suite = key exchange + signature + encryption + hash algorithm

`TLS_RSA_WITH_AES_256_CBC_SHA`

`TLS_ECDHE_RSA_WITH_RC4_128_SHA`

Cipher Suite

Key exchange algorithms

client自己選secret，用server public key加密後和server shared

TLS_RSA	RSA, encrypt key with receiver's RSA public key
TLS_DH TLS_ECDH	Fixed Diffie-Hellman, public-key certificate contains public DH key
TLS_DHE TLS_ECDHE	Ephemeral Diffie-Hellman, public key is used to sign temporary DH key
TLS_DH_anon TLS_ECDH_anon	Anonymous Diffie-Hellman, DH without authentication

Strongly discouraged, Susceptible to MitM

Recommended: Provide (perfect) forward secrecy

Cipher Suite

Signature algorithms: RSA, DSA, ECDSA

Encryption algorithm: RC4, RC2, DES, 3DES, DES40, IDEA, AES

Message authentication algorithm: HMAC-MD5, HMAC-SHA1, HMAC-SHA256/384

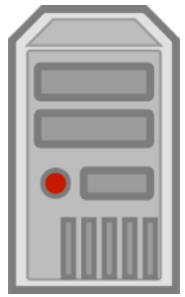
TLS Handshake: Server authentication and key exchange

3. ServerCertificate: digital certificate $cert_s$, pk_s

4. ServerKeyExchange: message depending on the chosen key exchange algorithm

5. CertificateRequest: list of acceptable certificate authorities (if client auth enabled)

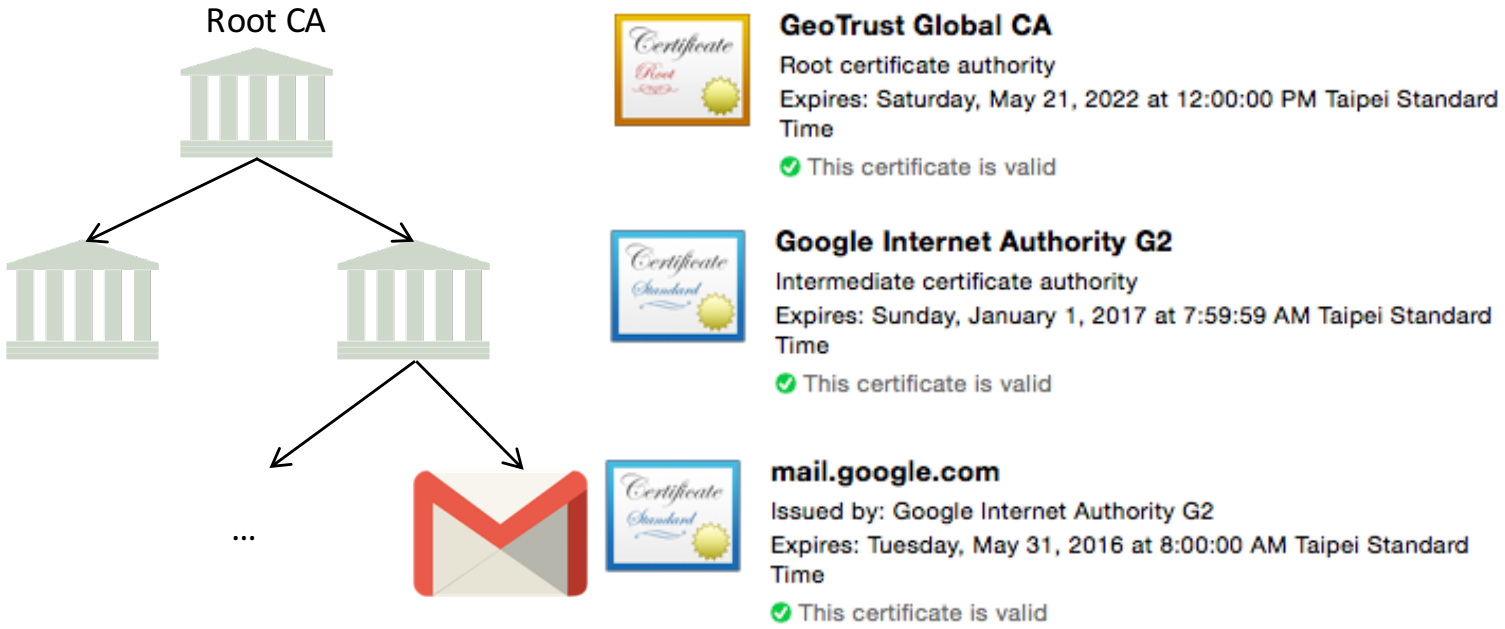
6. ServerHelloDone



Certificate Authorities

X.509 Public Key Infrastructure (PKI)

A hierarchy of Certification Authorities

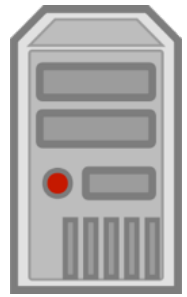


Root certificates

All your trust relationships online are reduced to trusting this list of root certificates

OS X Keychain

TLS Handshake: Client authentication and key exchange



7. ClientCertificate: $cert_c$, pk_c (if client auth enabled)

→

8. ClientKeyExchange

- RSA: client generates 48-byte pre-master secret, encrypts it with server's public key
- Eph or anon DH: client's public DH value
- Fixed DH: none (certificate contained DH key)

Compute or select a pre-master secret (PS)

→

9. CertificateVerify: sign(handshake_log1) (if client auth enabled)

→

Compute or receive the pre-master secret (PS)

Establishing shared keys

Compute master secret (MS) from pre-master secret (PS), client random (CR) and server random (SR)

From the shared master secret, derive

- Client MAC key
- Server MAC key
- Client encryption key
- Server encryption key
- Client IV
- Server IV

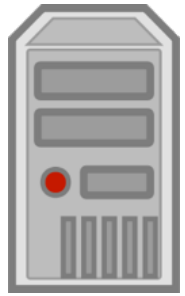
TLS Handshake: Change cipher spec and finish

10. ChangeCipherSpec: alerts server to start using the negotiated ciphers and key

11. Finish: Verify_data(MS, handshake_log2)

12. ChangeCipherSpec

13. Finish: Verify_data(MS, handshake_log3)

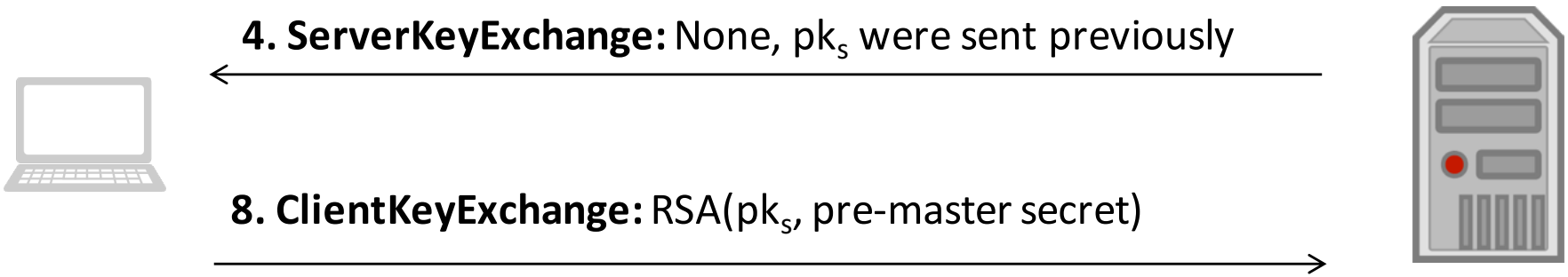


Forward secrecy

A desired property that the short-term session keys cannot be derived from the long-term key (in case it's compromised in the future) Why important?

Does **key exchange using RSA** provide forward secrecy?

- Client encrypts the session key with the server's long-term RSA public key

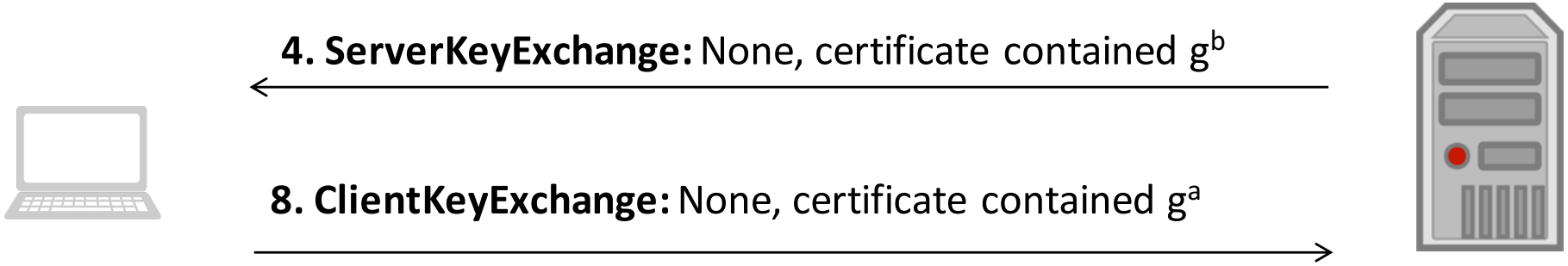


Forward secrecy

A desired property that the short-term session keys cannot be derived from the long-term key (in case it's compromised in the future)

Does key exchange using **Fixed Diffie-Hellman** provide forward secrecy?

- Client and server exchange certificates containing long-term public DH parameters

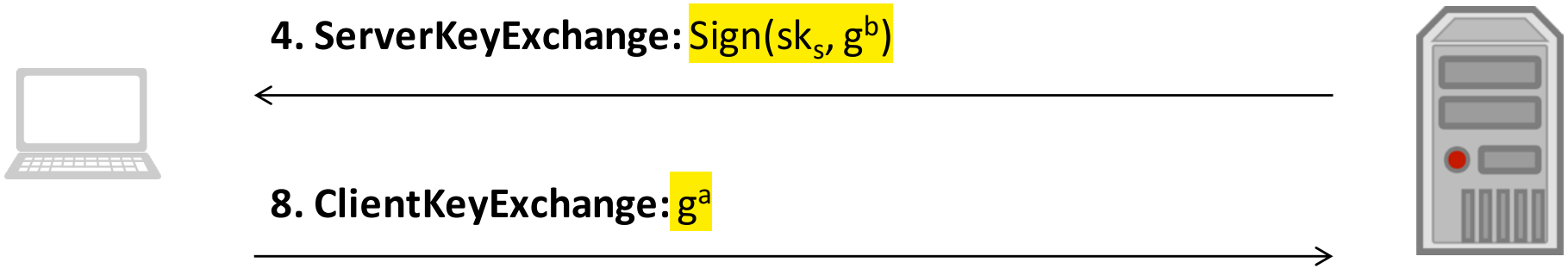


Forward secrecy

A desired property that the short-term session keys cannot be derived from the long-term key (in case it's compromised in the future)

Does key exchange using **Ephemeral Diffie-Hellman** provide forward secrecy?

- Server sends a temporary DH parameter signed by long-term key 用long term secret key來簽新產生的short term key



Known issues in SSL/TLS

Known issues in SSL/TLS

Weakness in crypto primitives

- Weak encryption & signature key lengths (40, 56, 64 bit encryptions subject to brute force attack)
- Weak hash functions (MD5)

Oracle attacks

- Compression
- CBC Initialization (e.g., predictable IVs)
- CBC padding

Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In Proceedings of IEEE Symposium on Security and Privacy, 2013.

Known issues in SSL/TLS

Protocol-level attack

- Ciphersuite downgrade attack
- Version downgrade attack
- Renegotiation attack

Implementation flaws

- Remote timing attacks
- PRNG seeding

Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In Proceedings of IEEE Symposium on Security and Privacy, 2013.

Known issues in SSL/TLS

Trust model issues

- Certification: hostname validation, parsing attacks, EV downgrading
- Anchoring trust: CA compromise, compelled certificates
- Transitivity of trust: basic constraints
- Maintenance of trust: blocking revocation, ownership transfer
- Indication and interpretation of trust: stripping TLS, spoofing browser, ignoring warnings, low-risk warnings

Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. In Proceedings of IEEE Symposium on Security and Privacy, 2013.

Attacks Examples

Example	type
Heartbleed	Implementation flaws
Low-risk warnings mask high-risk warnings	Trust model issues
Incorrect certificate validation	Trust model issues Implementation flaws
SSL stripping attack	Protocol-level attack
FREAK attack	Implementation flaws
Logjam attack	Weak crypto Protocol-level attack
POODLE attack	Oracle attack Protocol-level attack
RC4 NOMORE attack	Weak crypto

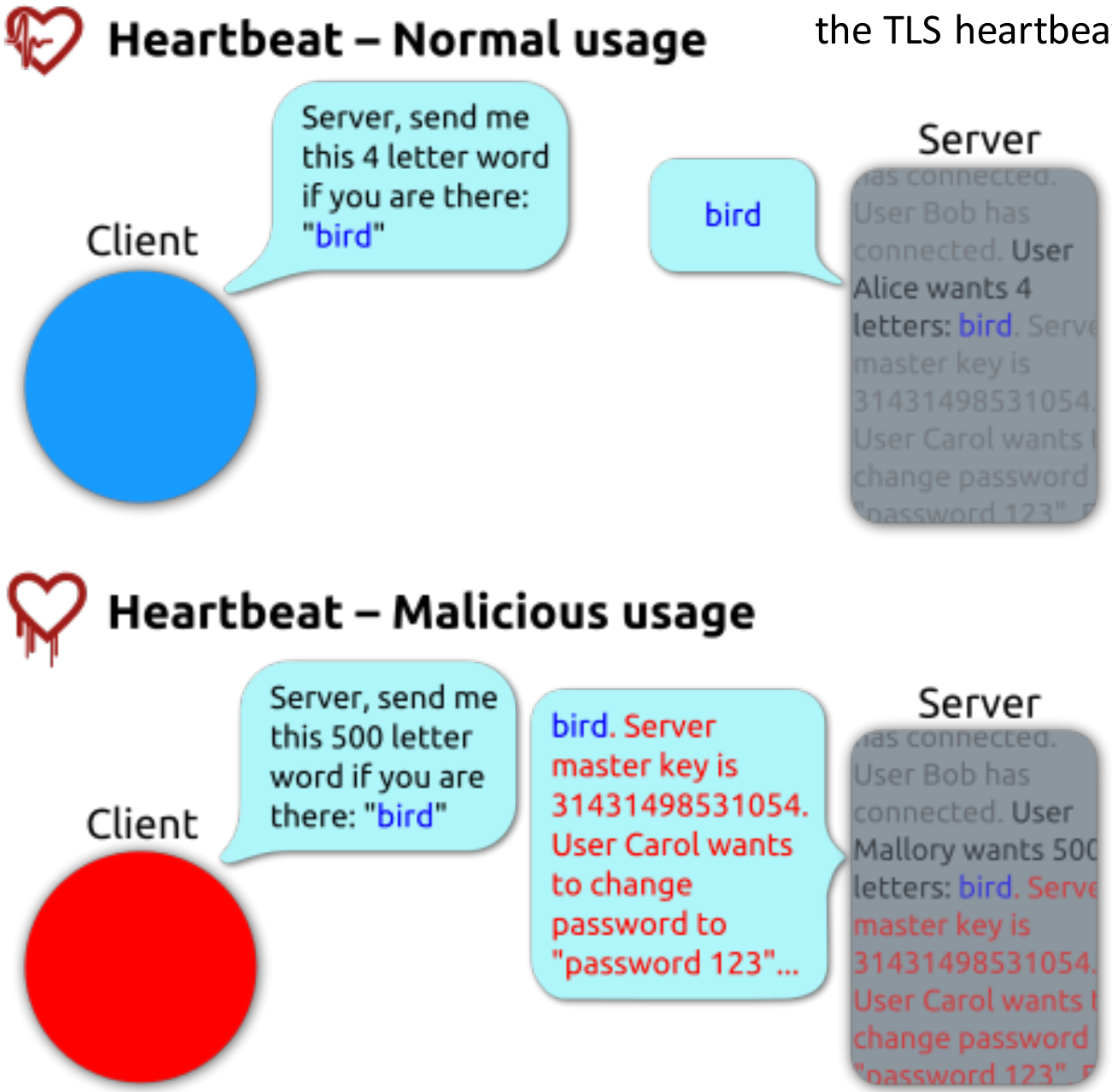
Example of implementation flaws: Heartbleed

A buffer over-read bug in the OpenSSL library

The attacker can access data in the memory, which may contain sensitive information such as passwords and session cookies

About 17% (half a million) servers were vulnerable

in OpenSSL's implementation of the TLS heartbeat extension



Source: Wikipedia

See xkcd as well: http://xkcd.com/1354/

Heartbleed Bug

Root cause

```
memcpy(bp, pl, payload);
```

Before patched

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
```

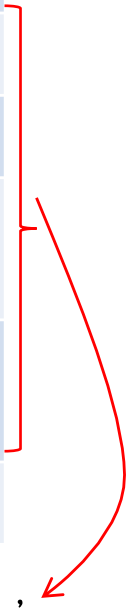
長度

After patched

```
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0;  /* silently discard per RFC 6520 sec. 4 */
pl = p;
```

Examples

Example	type
Heartbleed	Implementation flaws
Low-risk warnings mask high-risk warnings	True model issues
Incorrect certificate validation	Trust model issues Implementation flaws
SSL stripping attack	**Protocol-level attack**
FREAK attack	**Implementation flaws**
Logjam attack	**Weak crypto** **Protocol-level attack**
POODLE attack	**Oracle attack** **Protocol-level attack**
RC4 NOMORE attack	Weak crypto



概念上都是把TLS降級，
讓它更容易被破解

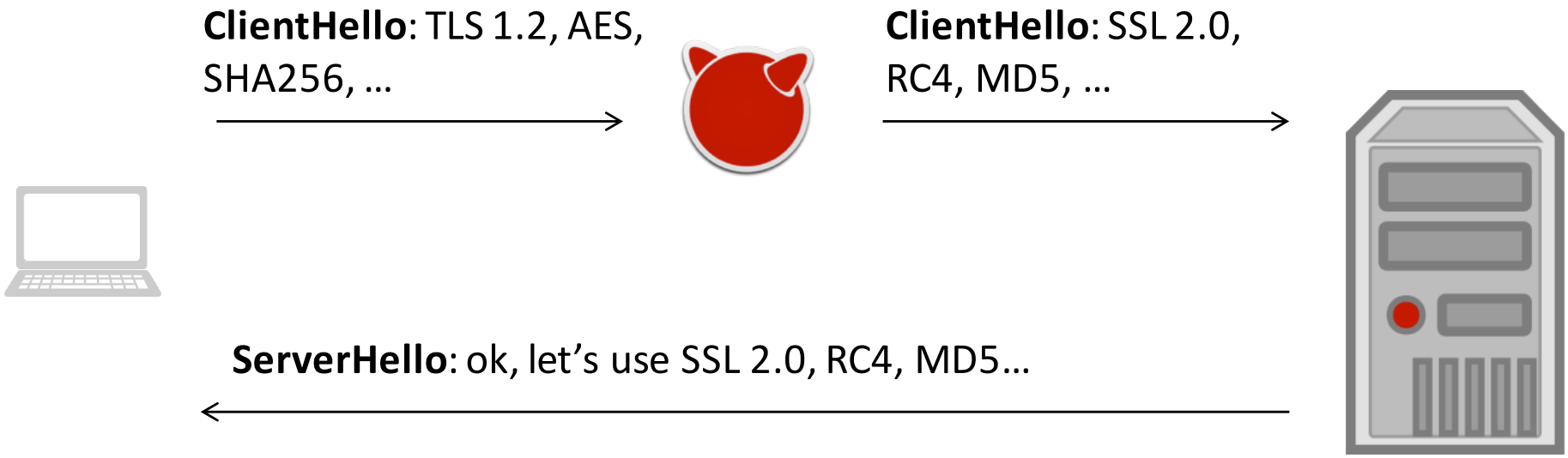
Downgrade Attacks

Rollback attack in SSL 2.0

This is why we need FINISH state

What if Eve manipulate the hello messages?

Protocol version and cipher suite preferences were not authenticated in SSL 2.0



Rollback attack fixed in SSL 3.0

Finished message contains a MAC on all the handshake protocol messages keyed by the master secret

Server can detect tampering at the end of the handshake and terminate the session if necessary

Q: What if an attacker downgrades it to SSL 2.0?

無解。不過現在瀏覽器都已經直接disable 2.0。

SSL Stripping attack

Users are not used to type **`https://url`** directly

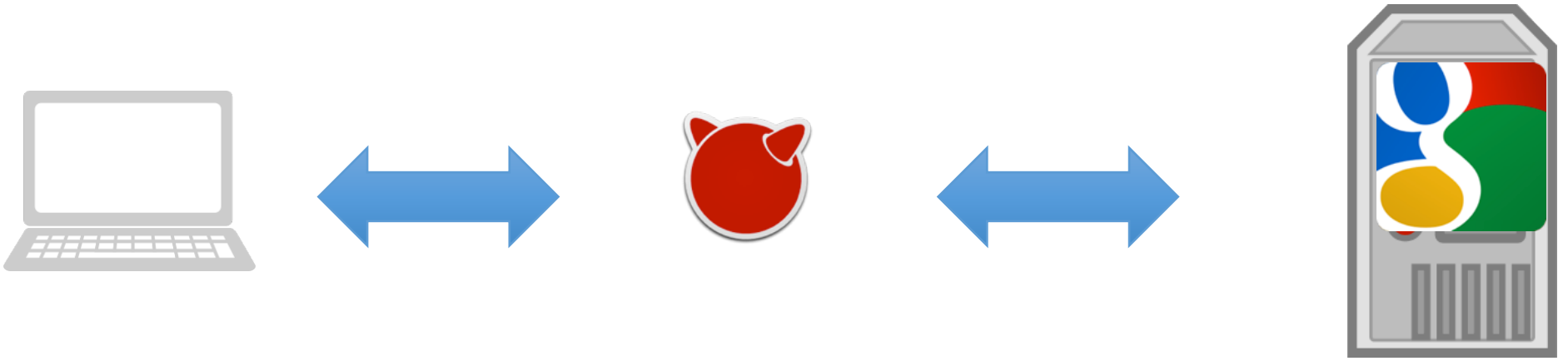
https://github.com/moxie0/sslstrip

SSL Stripping attack

Users are not used to type **`https://url`** directly

A MitM attacker "strips" **`HTTPS`** into **`HTTP`**

`http://www.google.com` `https://www.google.com`



https://github.com/moxie0/sslstrip

SSL Stripping attack

Root cause: HTTPS is not used since beginning!

Countermeasures

- HTTPS Everywhere Browser Plugin
  - Client-side protection
  - EFF's HTTPS Everywhere Browser Plugin automatically replaces HTTP with HTTPS if exists
  - https://www.eff.org/https-everywhere/
- HTTP Strict Transport Security (HSTS)
  - A web security policy that "allows a web server to declare that web browsers should only interact with it using secure HTTPS connections"
  - A better solution when the server can cooperate

HTTP Strict Transport Security (HSTS)

Allows a **domain** (**server-side)** to enforce HTTPS

How can the domain tell the browser that it enables HSTS?

- Option 1: Return HSTS header over an **HTTPS** connection (**Trust-On-First-Use**)
- Option 2: Add itself to the browser **preload** list (can protect the first request too)

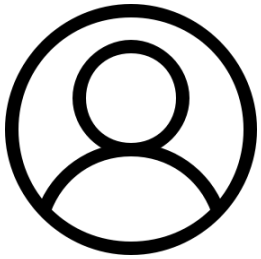
```
2219    { "name": "nbl.org.tw", "include_subdomains": true, "mode": "force-https" },
2220    { "name": "nctx.co.uk", "include_subdomains": true, "mode": "force-https" },
2221    { "name": "nemovement.org", "include_subdomains": true, "mode": "force-https" },
2222    { "name": "newkaliningrad.ru", "include_subdomains": true, "mode": "force-https" },
```

Option 1: Return HSTS header over an **HTTPS** connection (**Trust-On-First-Use**: assume the first request is secured)

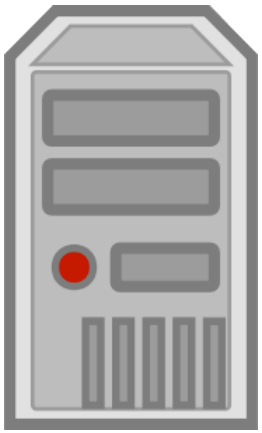

```
$ curl -s -D- https://paypal.com/ | grep Strict
Strict-Transport-Security: max-age=63072000

$ curl -s -D- https://github.com/ | grep Strict
Strict-Transport-Security: max-age=31536000;
includeSubdomains; preload
```

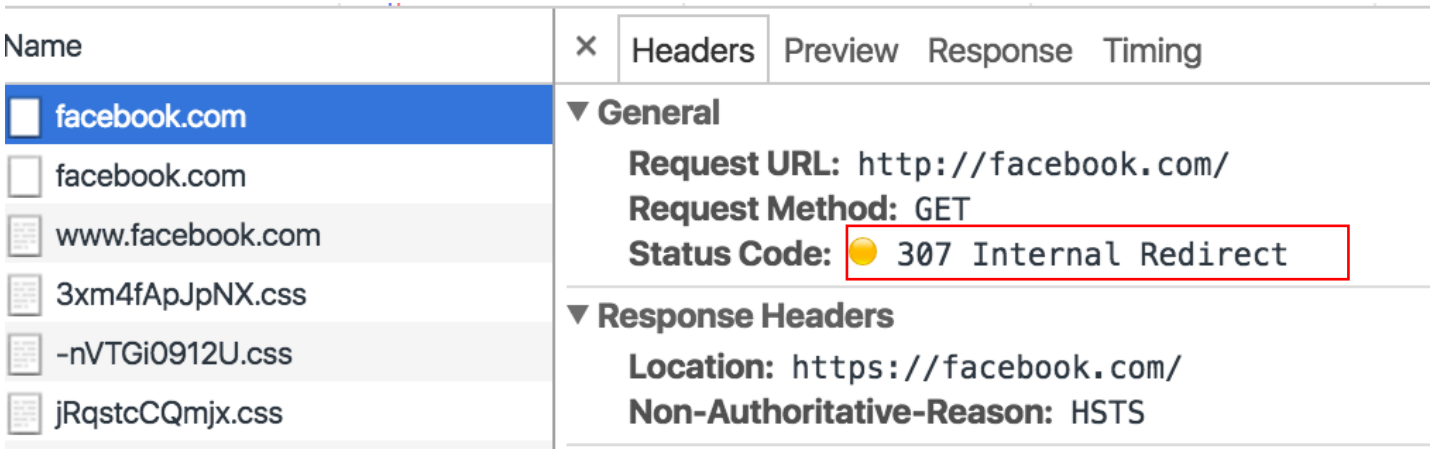
HSTS in action

User type in
`http://facebook.com/`



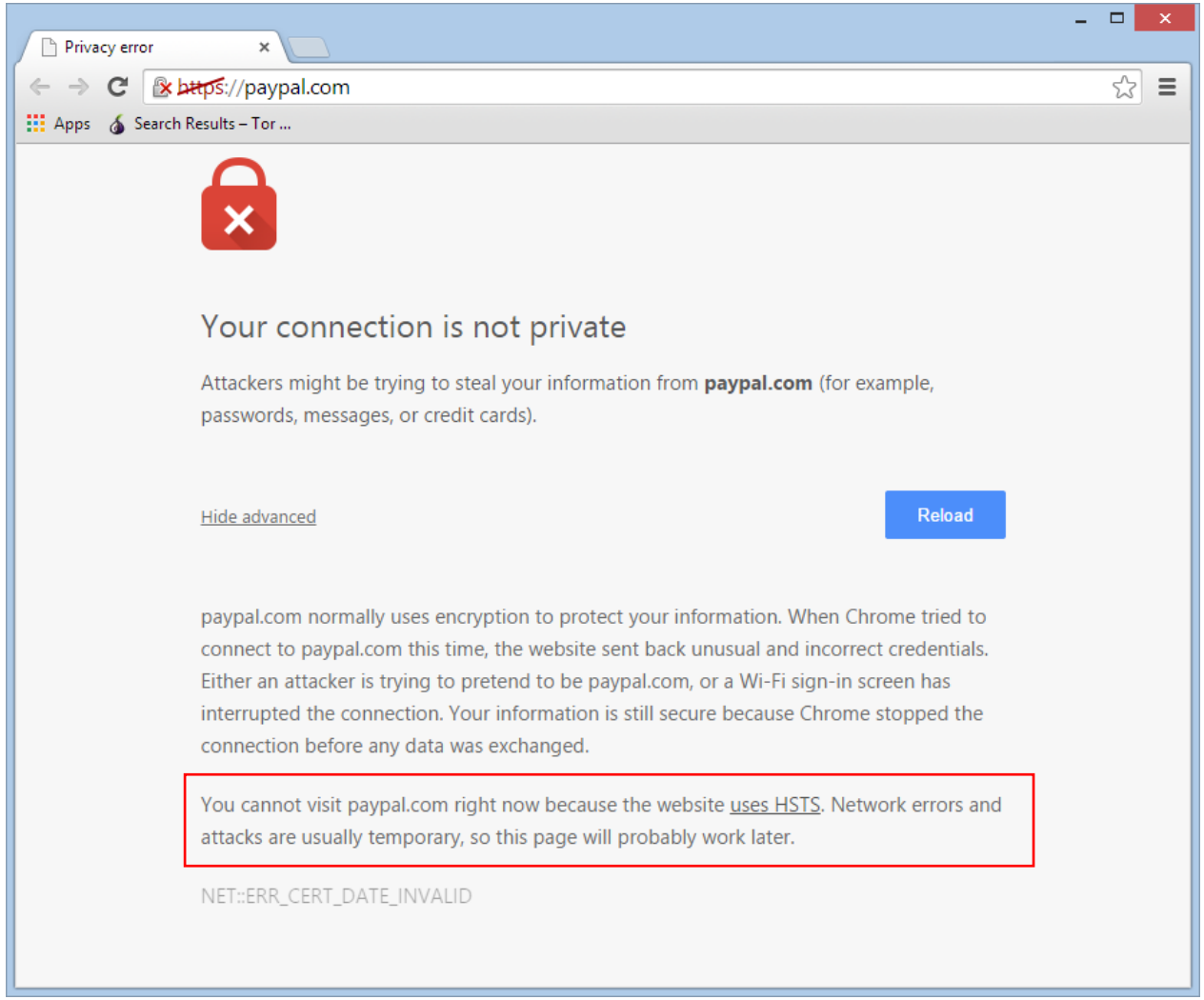
Browser internally redirects to
`https://facebook.com/`





HSTS in action

Prevent users from clicking through certificate warnings



Checking whether HSTS is enabled

https://hstspreload.org

chrome://net-internals/#hsts

Issues with HSTS adoption

HSTS設定較複雑，adoption rate低

	Alexa top 1M		Preloaded domains	
	Domains	%	Domains	%
Attempts to set dynamic HSTS	12,593	—	751	—
Doesn't redirect HTTP→HTTPS	5,554	44.1%	23	3.1%
Sets HSTS header only via HTTP	517	4.1%	3	0.4%
Redirects to HTTP domain	774	6.1%	9	3.1%
HSTS Redirects to non-HSTS	74	0.6%	3	0.4%
Malformed HSTS header	322	2.6%	12	1.6%
`max-age` = 0	665	5.3%	0	0%
0 < `max-age` <= 1 day	2,213	17.6%	5	0.7%
Sets HSTS securely w/o errors	5,099	40.5%	659	87.7%

Kranch, Michael, and Joseph Bonneau. "Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning." NDSS, 2015.

Privacy Concerns: Web Tracking via HSTS

Browsers remember websites' HSTS settings

Privacy concerns

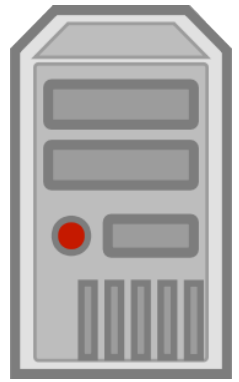
- Leverage the stored HSTS setting to infer websites visited by a client
- Use client's HSTS cache to store "super cookie" for cross-site tracking

https://www.leviathansecurity.com/blog/the-double-edged-sword-of-hsts-persistence-and-privacy
https://arstechnica.com/information-technology/2015/10/unpatched-browser-weaknesses-can-be-exploited-to-track-millions-of-web-users/
https://webkit.org/blog/8146/protecting-against-hsts-abuse/

HSTS Super Cookie

Initial visit:

Server S



request

Assign ID=10011

Client C

Load webpage with tracker script

Tracking service T

https://bit0.example.com
https://bit1.example.com
https://bit4.example.com

HSTS enabled
bit0.example.com
bit1.example.com
bit4.example.com

Responses with HSTS header

HSTS Super Cookie

Server S

Subsequent visits:

request

Ask client to load
http://bit{x}.example.com over HTTP

https://bit0.example.com
https://bit1.example.com
http://bit2.example.com
http://bit3.example.com
https://bit4.example.com

Tracking service T

HSTS enabled
bit0.example.com
bit1.example.com
bit4.example.com

Return different content
for HTTP and HTTPS

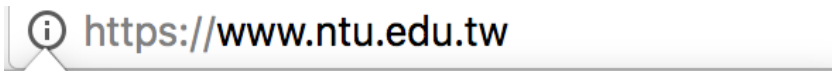
S can identify C based on the loaded content!

Examples

Example	type
Heartbleed	Implementation flaws
Low-risk warnings mask high-risk warnings	**Trust model issues**
Incorrect certificate validation	**Trust model issues** **Implementation flaws**
SSL stripping attack	Protocol-level attack
FREAK attack	Implementation flaws
Logjam attack	Weak crypto Protocol-level attack
POODLE attack	Oracle attack Protocol-level attack
RC4 NOMORE attack	Weak crypto

Browser indicator

No HTTP



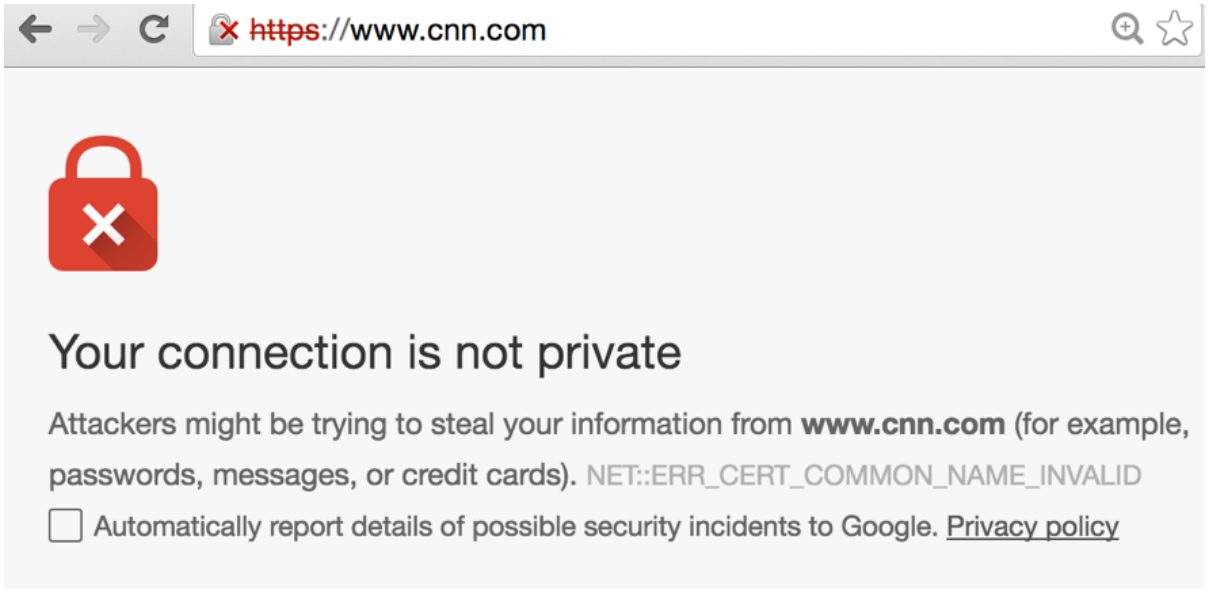
HTTPS but some part is not private (e.g., mixed content)

HTTPS but has certificate errors

Domain-validated (DV) or organization-validated (OV) certificate

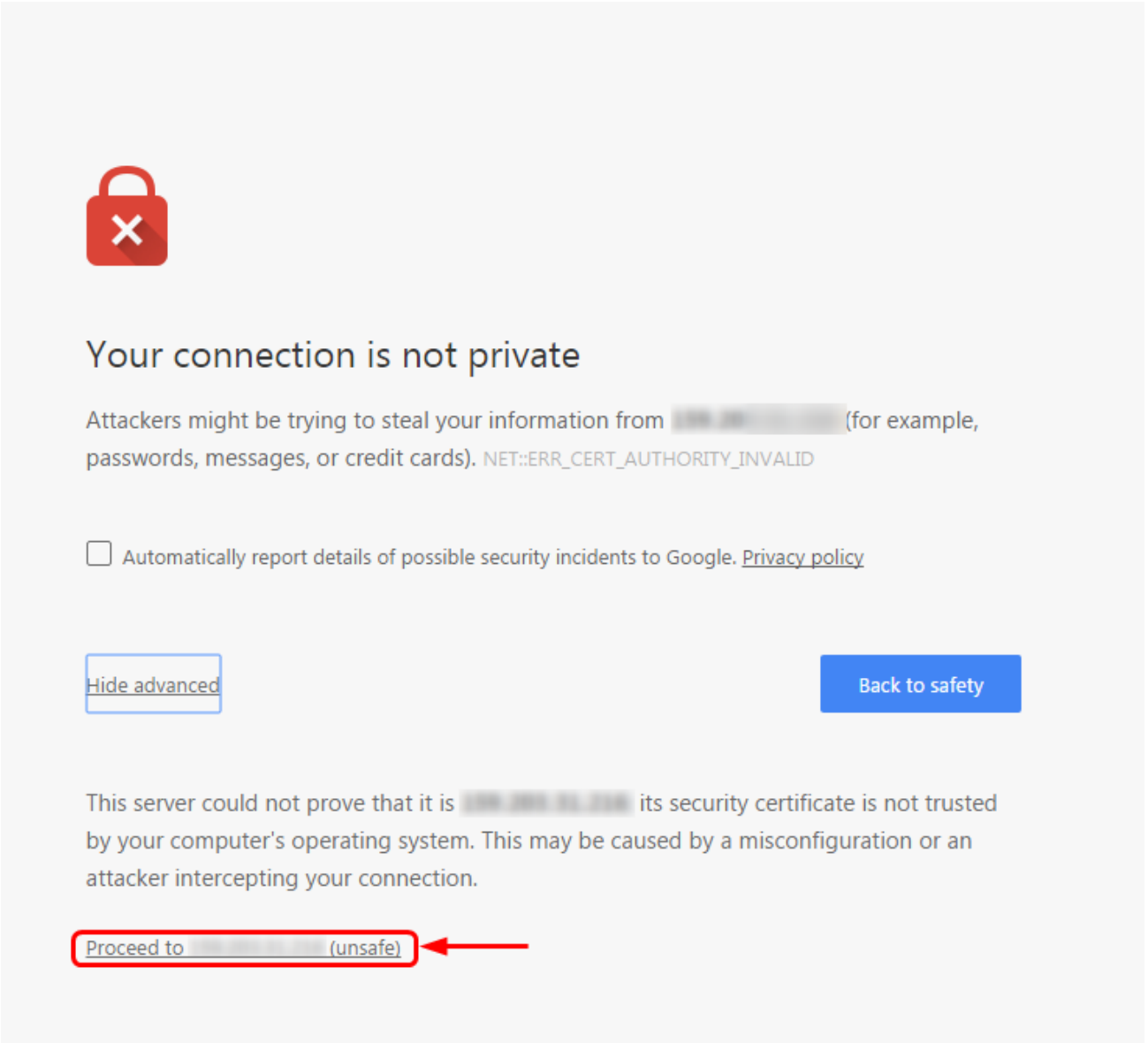
Extended-validated (EV) certificate

Certificate warnings



This server could not prove that it is www.cnn.com; its security certificate is from a.ssl.fastly.net. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed anyway…



Example of trust model issues:
Low-risk warnings mask high-risk warnings

Some browser implementation shows only one error

Users may just click through if the one being shown looks relatively low risk

Safari 7 or Chrome 30 (on Linux): for a self-signed recently-expired certificate, the only error warning is ‘Expired certificate’

Opera: for an invalid certificate containing a 512-bit RSA key, Opera only warns ‘Weak Key’

Brubaker, Chad, et al. "Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations." In Proceedings of IEEE Symposium on Security and Privacy, 2014.

Real-world issues with PKIs/CAs

Too many CAs

Rouge CAs or stolen certificates

Certificate revocation

root CA還是太多了

Too many CAs & root certificates

Pre-loaded into browser and/or OS

~150 root certificates from ~50 organizations

Roots certificates can authorize intermediate CAs

EFF's SSL Observatory reports that Microsoft IE and Mozilla Firefox trust 1482 different CA public keys held by 651 organizations located throughout the world

Any CA can issue an acceptable certificate for any site

Rouge CAs & stolen certificates

(亂發的)

Attacker uses a fake certificate that passes browser check

Who wants to do this?

- Organizations monitoring traffic
- Governments doing censorship
- Cloud providers
- …

公司自己發root certificate，所有對外連線都可以做mitm

How?

- Compromised CAs
- Compelled certificates
- Browser not checking correctly (perhaps on purpose)
- …

Lenovo PCs ship with man-in-the-middle adware that breaks HTTPS connections [Updated]

Superfish may make it trivial for attackers to spoof any HTTPS website.

by **Dan Goodin** - Feb 20, 2015 12:36am CST

Share Tweet 333

How the Nokia Browser Decrypts SSL Traffic: A "Man in the Client"

JANUARY 11, 2013 BY STEVE SCHULTZE

NSA disguised itself as Google to spy, say reports

If a recently leaked document is any indication, the US National Security Agency -- or its UK counterpart -- appears to have put on a Google suit to gather intelligence.

by **Edward Moyer** @edatnews / September 12, 2013 2:19 PM PDT

APR
2
2015

Distrusting New CNNIC Certificates

kwilson

96 responses

Efforts to handle CA trust issues

Certificate Transparency (and its variants)

- Chrome requires this for newly issued EV Certificates in 2015

HTTP Public Key Pinning (HPKP) also trust on first use

- RFC 4769
- Currently, HPKP is supported in Firefox and Chrome

DNS Certificate Authority Authorization (CAA)

- RFC 6844

Online Certificate Status Protocol (OCSP), OCSP stapling

- RFC 6960
- Supported by major browsers
- Chrome disables OCSP by default

布告欄只能貼不能刪，照時間排序

太最近的才發的certificate就有問題。布告欄上也都有歷史紀錄，什麼網站過去用什麼CA簽

Certificate Transparency

"An open framework for monitoring and auditing certificates in nearly real time"

Maintain publicly verifiable and accessible logs that can be queried by users and domain owners

Transparency eases the problem of detecting fraudulent certificates

http://www.certificate-transparency.org/

Certificate Transparency

Chrome starting to require Certificate Transparency for all newly issued, publicly trusted certificates starting in April 2018.

Group by Issuer

Criteria	Identity = 'nslab.csie.ntu.edu.tw'

Certificates

crt.sh ID	Logged At	Not Before	Not After	Issuer Name
334498905	2018-02-18	2018-02-18	2018-05-19	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3
259292251	2017-11-20	2017-11-20	2018-02-18	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3
195274977	2017-08-21	2017-08-21	2017-11-19	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3
142807665	2017-05-23	2017-05-23	2017-08-21	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3
132271966	2017-05-03	2017-05-03	2017-08-01	C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3

https://crt.sh

Background:
如何確保雲端資料的完整性？

Assumption: secure communication

Attacker model: cloud may return wrong or tempered data

How can Alice ensure the integrity of her data?

如果每個檔案都存一個hash，local端需要存很多hash值 -> 用hash tree

D_0

D_1

D_2

Get file, id = 1

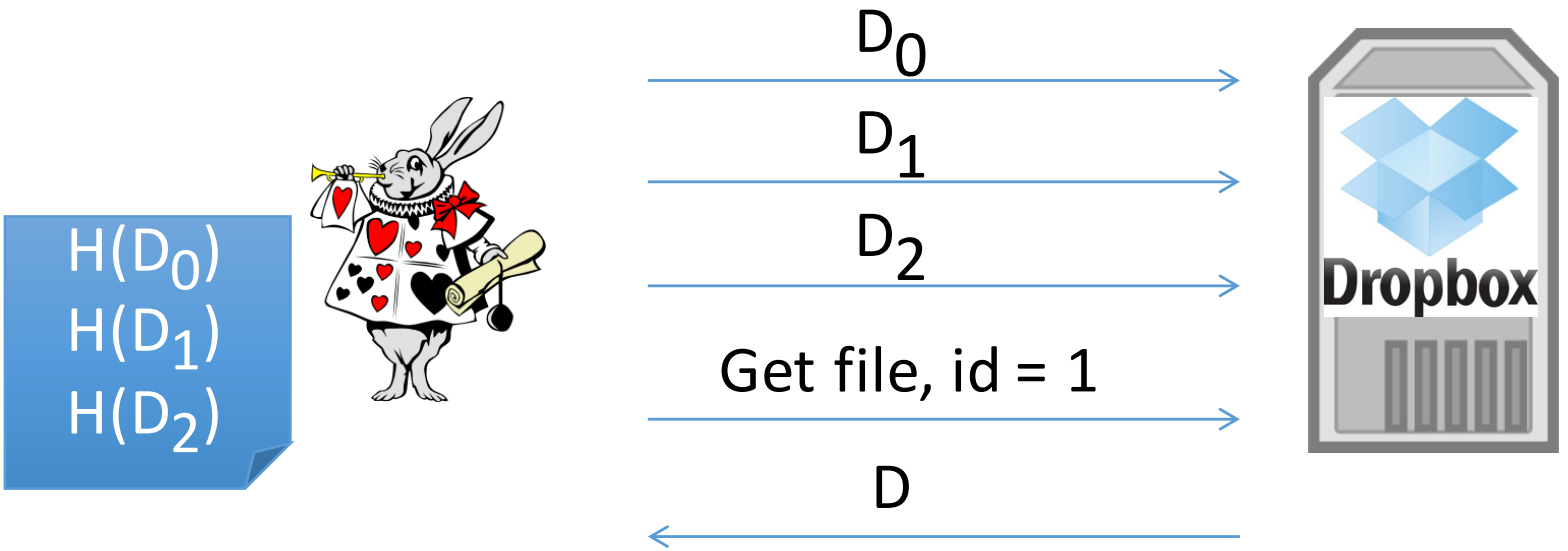
D

Is $D = D_1$?

Background: 如何確保雲端資料的完整性？

Approach #1: keep file hashes

Disadvantages: storage overhead at client, Alice needs to copy this large hash file when using another computer



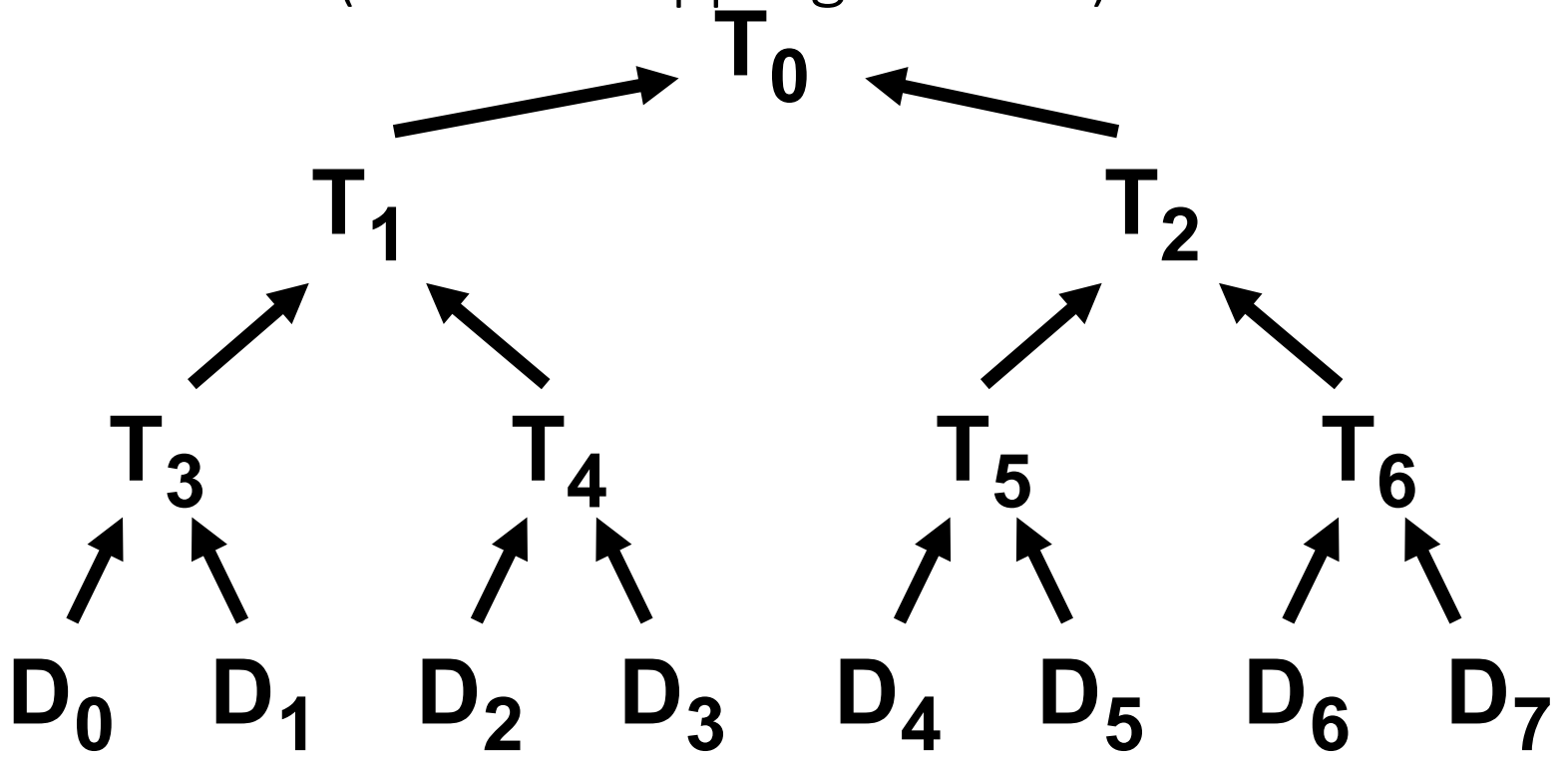
Verify if H(D) = H(D_1)

Background: Merkle Hash Tree (Merkle 1979)

Construct binary tree over data values

Leaf nodes are data values

Parent = Hash(Left-child | | Right-child)



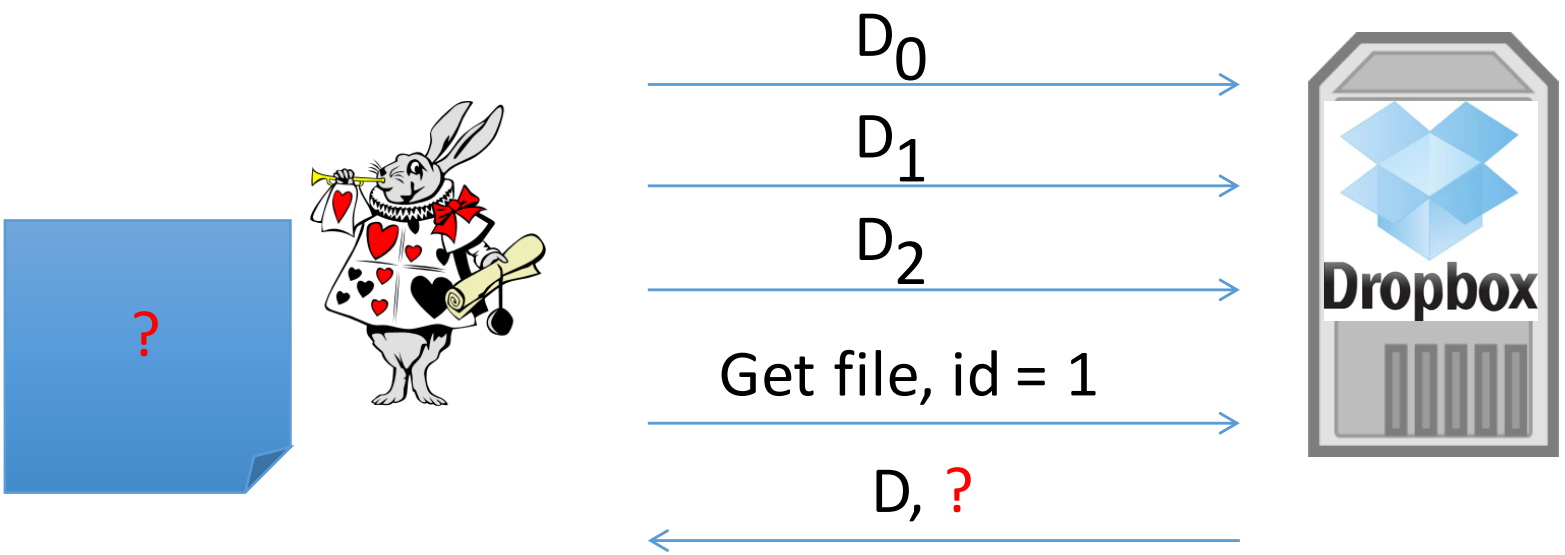
Background: 如何確保雲端資料的完整性？

Approach #2: using a Merkle hash tree

For simplicity, assume Alice never updates her files

What should Alice keep and verify?

What should the cloud return?



Verify if ?

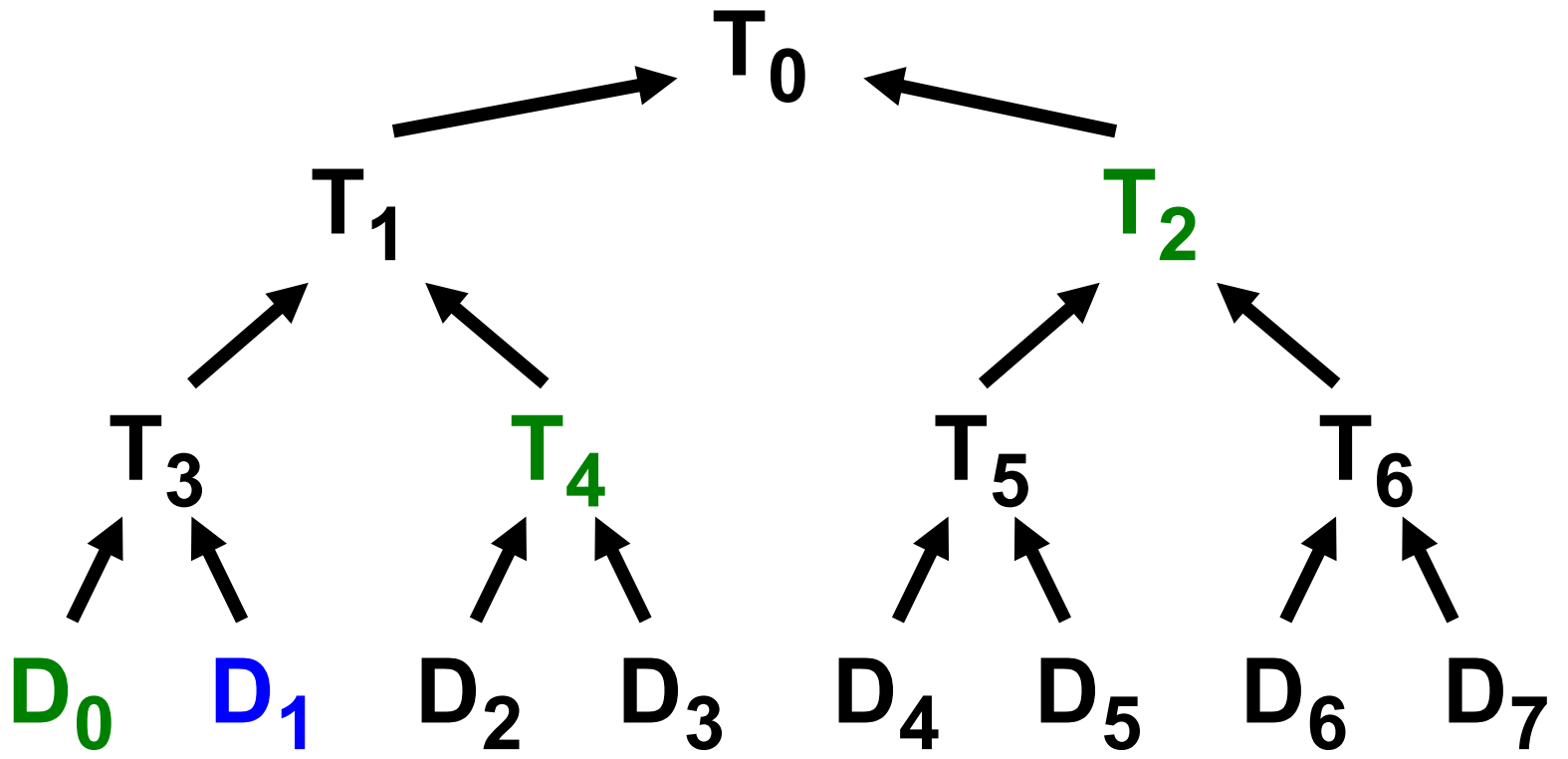
Integrity check using Hash Trees

Alice computes T_0 before uploading D_0 to D_7

Alice requests file id=1 along with a proof

Cloud returns D_1, {D_0, T_4, T_2}

Alice verifies $T_0 = H(H (H(D_0 || D_1) || T_4) || T_2)$



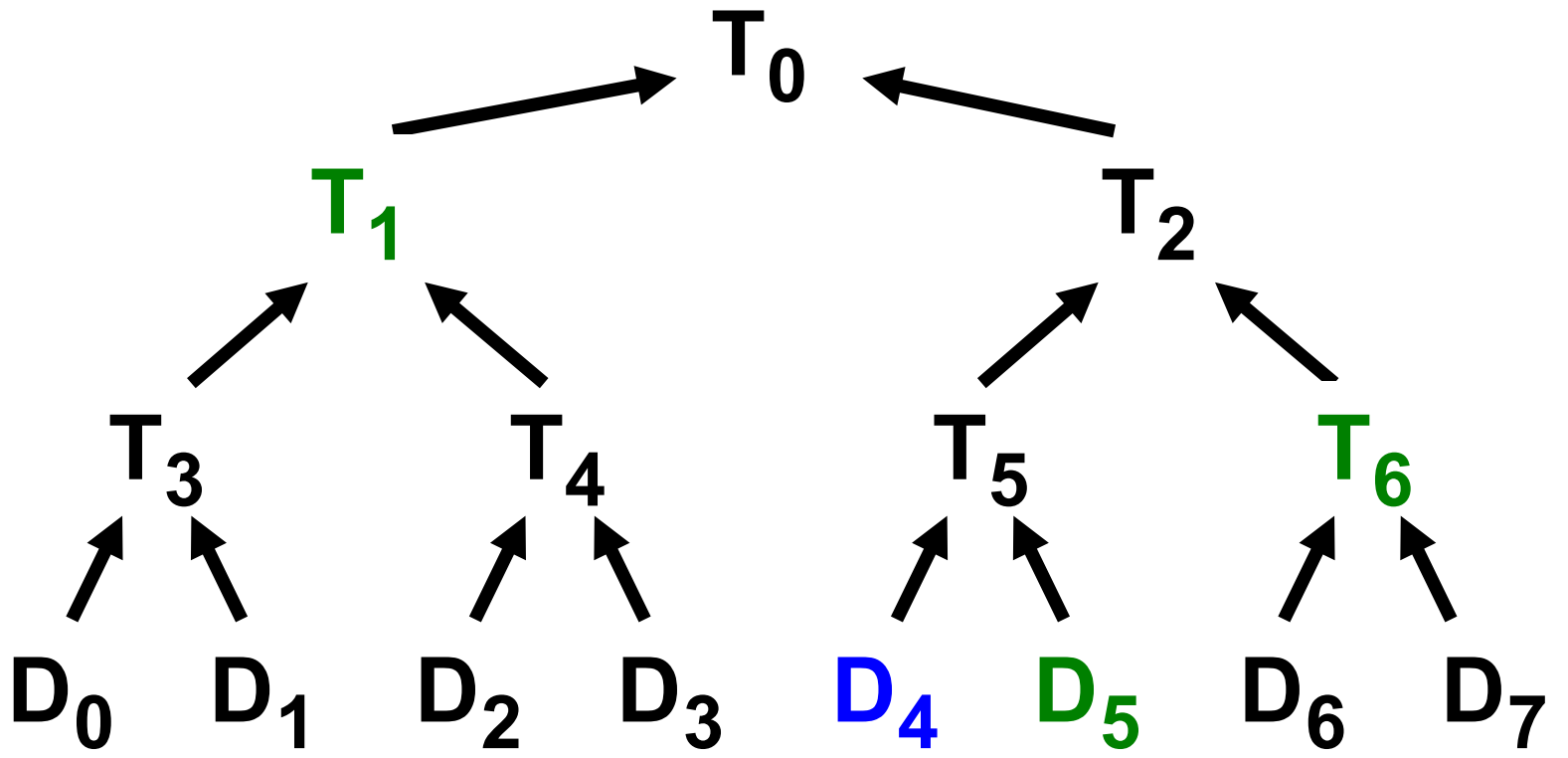
Integrity check using Hash Trees

Alice computes T_0 before uploading D_0 to D_7

Alice requests file id=4 along with a proof

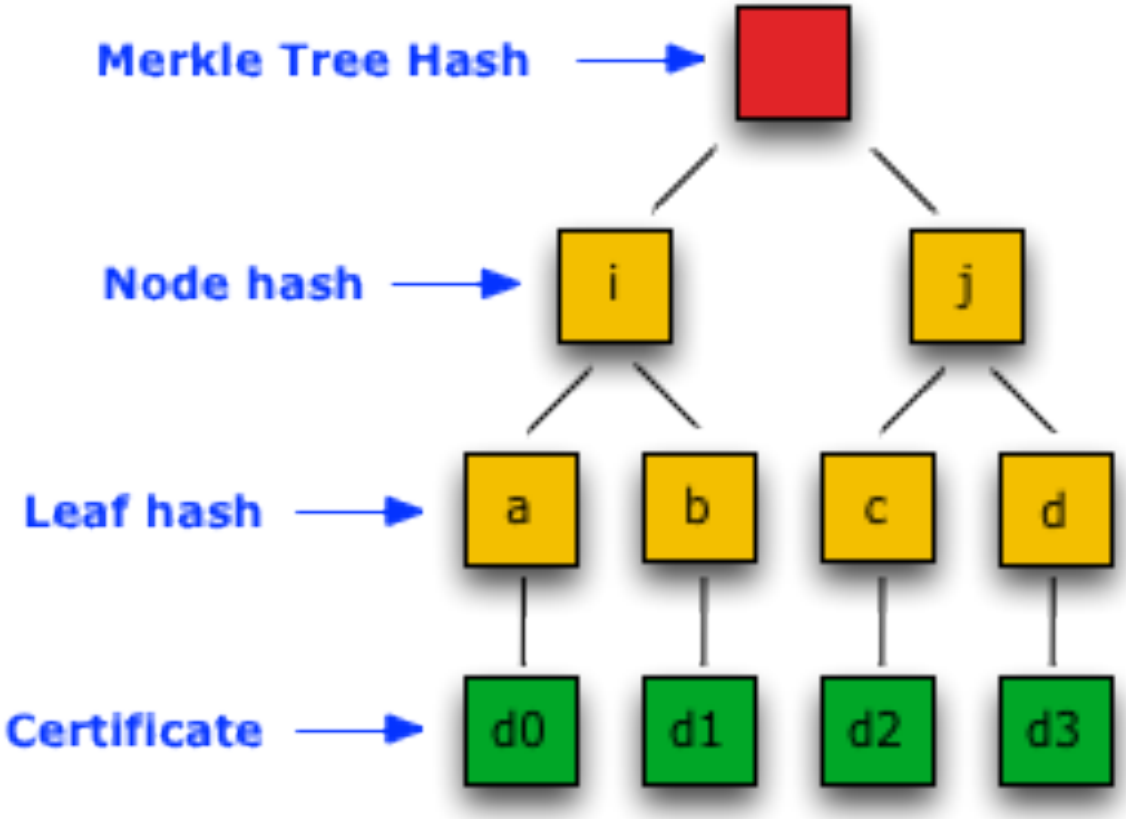
Cloud returns D_4, {?}

Alice verifies T_0 = ?



Certificate Transparency: certificate logs

The log server signs the root of the Merkle tree

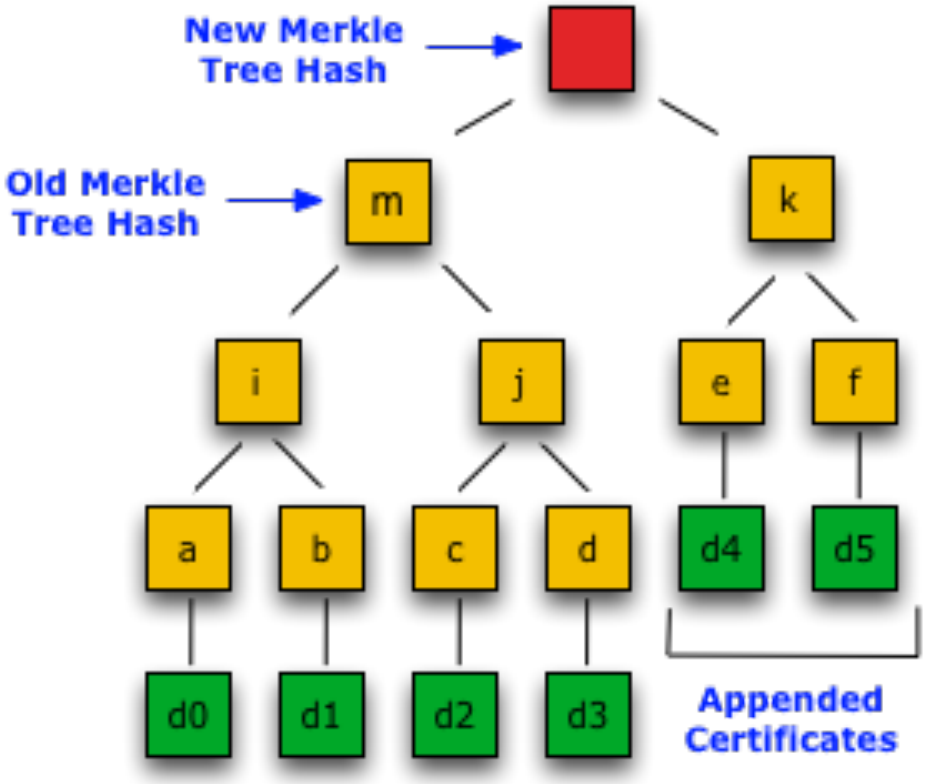


Certificate Transparency: append new certificates

Append-only data structure

New tree root contains the old ones and appended certs

The log server signs the root of the new Merkle tree

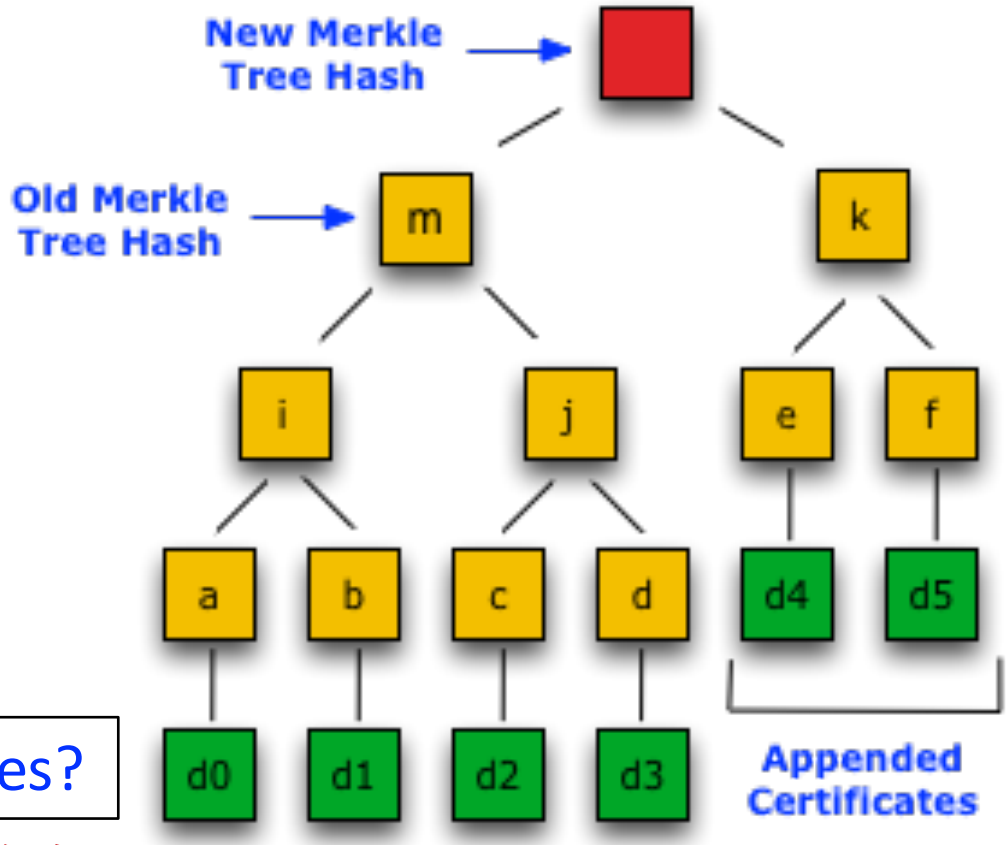


Certificate Transparency: consistency check

整棵樹的結構是公開透明的

Certificates cannot be inserted to an earlier part of the log (i.e., cannot be back-dated)

Certificates in the log cannot be modified



Q: how to detect bogus certificates?

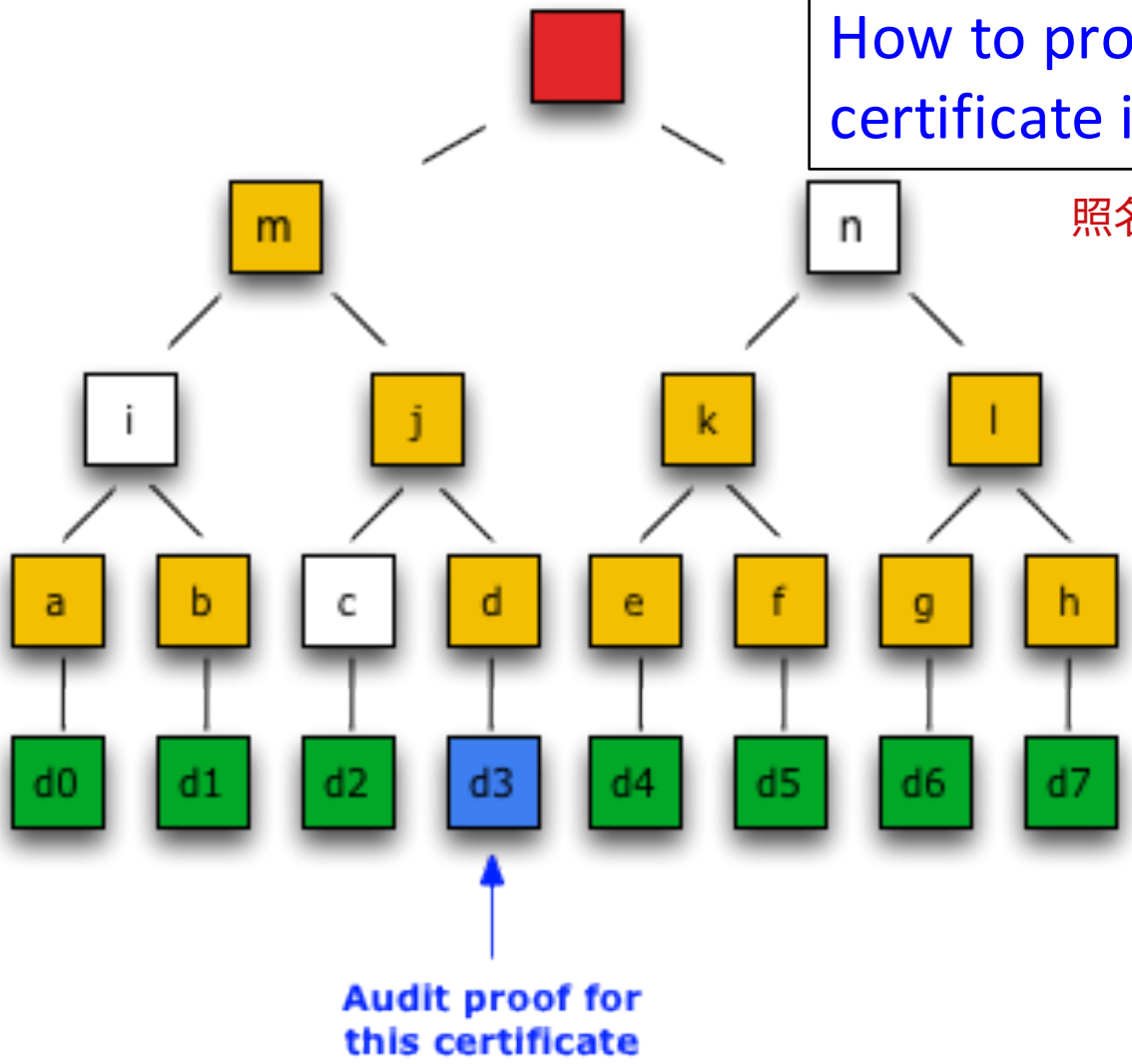
偵測某時間點之前是否已經有certificate了

Certificate Transparency: audit proofs

A particular certificate has been appended to the log

How to prove that a certificate is NOT in the log?

照名字顺序排序



HTTP Public Key Pinning (HPKP)

Allows a domain (server-side) to specify a whitelist of public keys for validating certificate chains

- The certificate chain must contain at least one public key in the whitelist

How can the domain tell the browser that it enables HPKP?

- Very similar to HSTS
- Option 1: Return HPKP header over a HTTPS connection
- Option 2: Add itself to the browser preload list

Option 1: Return HPKP header over an HTTPS connection (Trust-On-First-Use: assume the first request is secured)

```
Public-Key-Pins: max-age=2592000;
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
report-uri="http://example.com/pkp-report"
```

Option 2: Add itself to the browser preload list (can protect the first request too)

```
{
  "name": "google",
  "static_spki_hashes": [
    "GoogleBackup2048",
    "GoogleG2",
    "GeoTrustGlobal"
  ],
  "report_uri": "http://clients3.google.com/cert_upload_json"
},
```

Issues with HPKP adoption

Very low adoption rate: Only 375 sites deploy HPKP as of August 2016 [1]

Why?

- Hard to do it right 直接連不到
- HPKP misconfiguration can be more devastating than HSTS 不安全

In case of privacy key lost or certificate change, sites may be unavailable for a long time

- Can be abused by attackers
- Workaround: HPKP requires at least one backup key

Google plans to deprecate HPKP support in Chrome…

[1] https://scotthelme.co.uk/alexa-top-1-million-crawl-aug-2016/

Certificate revocation

server private key leak
CA find server is malicious

Certificate revocation list (CRL)

- A list of certificates that should be revoked
- Signed by the CA
- Challenge: how to distribute the list to users in an efficient and timely manner?
  - Query for a new CRL whenever receiving a certificate?
  - Periodic update?
  - Using short-lived certificates?

Certificate revocation

Online Certificate Status Protocol (OCSP)

- Bob requests Alice's CA to verify her certificate online
- Challenge:
  - CAs need to be online all the time
  - Increase communication latency

OCSP stapling

- Alice obtains a timestamped OCSP response and present it to Bob
- Challenge:
  - Low implementation rate

Examples

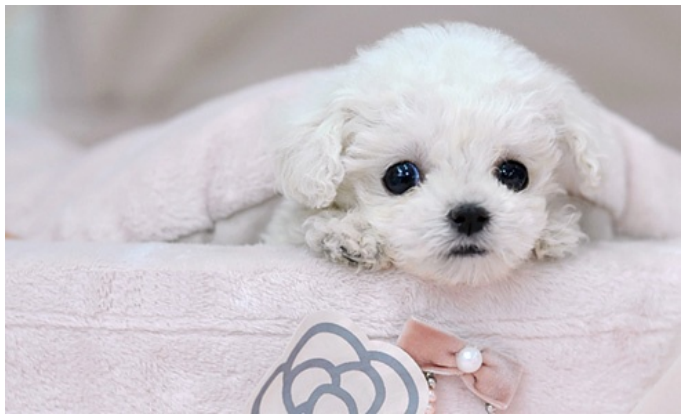
Example	type
Heartbleed	Implementation flaws
Low-risk warnings mask high-risk warnings	Trust model issues
Incorrect certificate validation	Trust model issues Implementation flaws
SSL stripping attack	Protocol-level attack
FREAK attack	Implementation flaws
Logjam attack	Weak crypto Protocol-level attack
POODLE attack	**Oracle attack** **Protocol-level attack**
RC4 NOMORE attack	Weak crypto

POODLE Attack

POODLE = **Padding Oracle** On **Downgraded** Legacy Encryption

Downgrade to SSL 3.0

Can decrypt one byte in 256 requests



Downgrade again?

Didn't we check the handshake in the finished message?

Yes, but it's more complex than that in reality.

- https://tools.ietf.org/html/rfc7507

連線失敗時client常常會自動downgrade

"To work around interoperability problems with legacy servers, many TLS client implementations do not rely on the TLS protocol version negotiation mechanism alone but will intentionally reconnect using a downgraded protocol if initial handshake attempts fail."

CBC Mode Revisited

A ciphertext block depends on all plaintext blocks up to this point

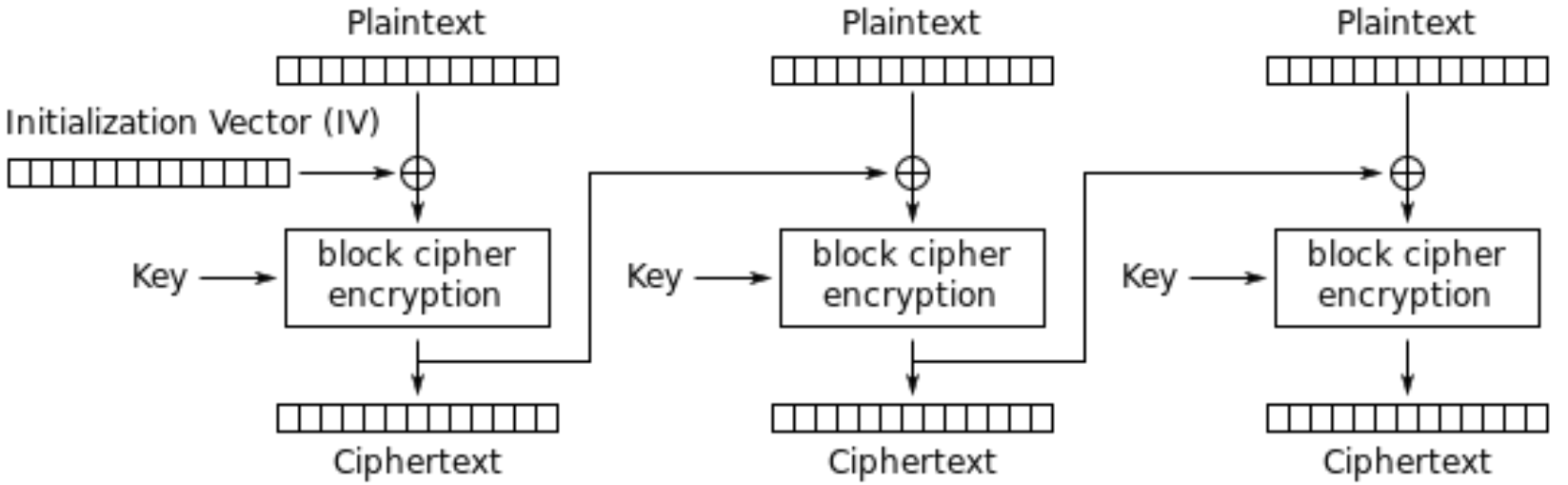
An initialization vector (IV) adds randomness, ensuring unique ciphertext

- IV doesn't need to be secret
- IV should be unpredictable and unique

Same plaintext block -> different ciphertext block

Encryption

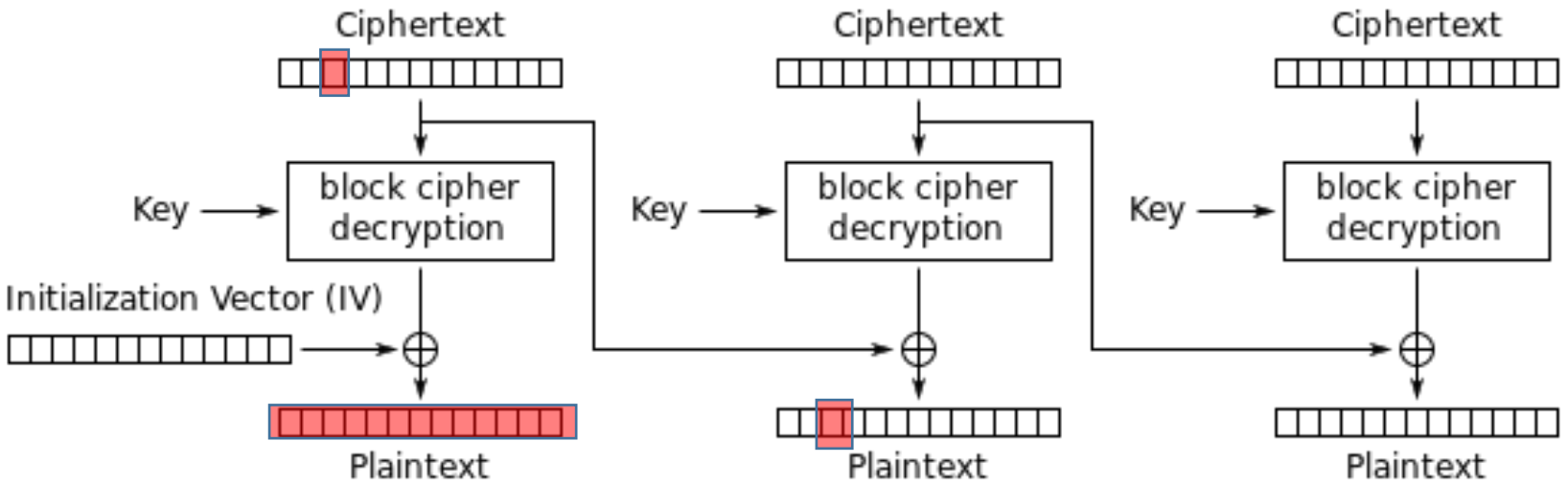
$$C_i = E_K(P_i \oplus C_{i-1})$$
$$C_0 = IV$$



Cipher Block Chaining (CBC) mode encryption

Decryption

$$P_i = D_K(C_i) \oplus C_{i-1}$$
$$C_0 = IV.$$



Cipher Block Chaining (CBC) mode decryption

What happen when plaintext size is not a multiple of n?

Some modes of operation don't require the plaintext size to be divisible by n

- E.g., CTR, OFB

Padding to the block boundary

Padding Oracle Attack

However, the predictable padding format can be exploited.

Suppose we have access to a **padding oracle** that tells us whether the padding format is correct

- How?
- Error messages, timing difference, …
- In SSL 3.0, the server first checks the validity of the padding and returns an error code for invalid padding.

The attacker can use the oracle to verify the guesses of plaintext.

PKCS#7 Padding

Padding to the block boundary

The value of each added byte is the number of bytes that are added.

1-byte padding: 01

2-byte padding: 02 02

3-byte padding: 03 03 03

…

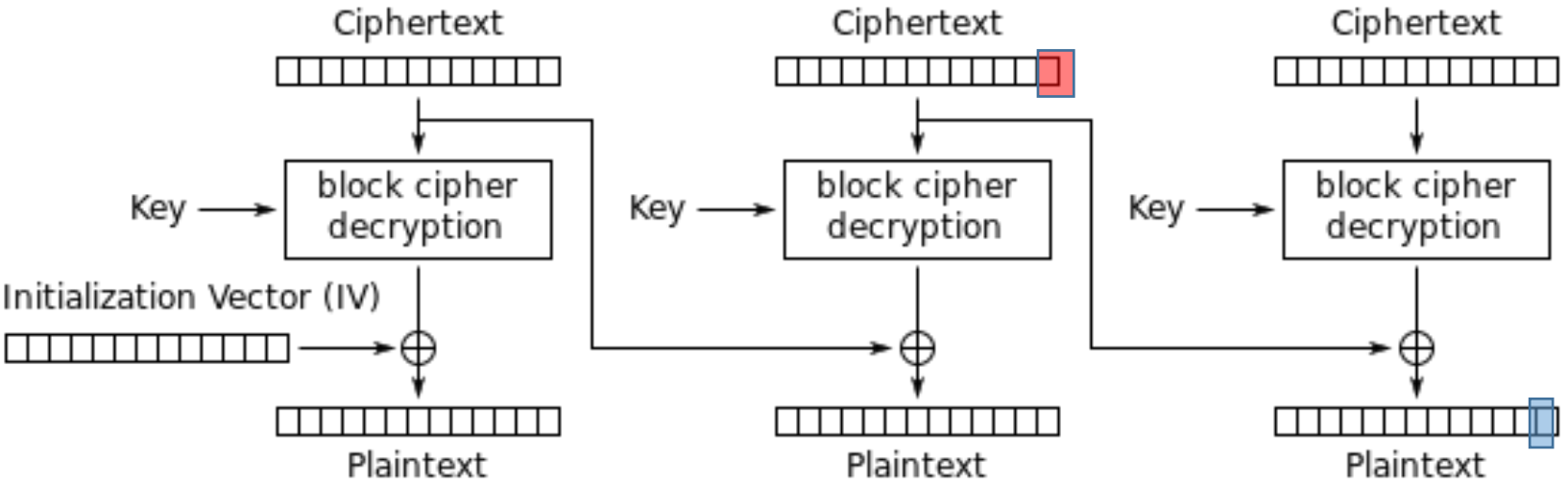
CBC Padding Oracle Attack (against PKCS#7 Padding)

We know C_1, C_2, C_3 and want to get P_3

Guess the last byte of P_3, say z_{-1}

Change the last byte of C_2, say b_{-1}, into $b_{-1} = b_{-1} \oplus z_{-1} \oplus 0x01$

If the guess is correct, then the last byte of P_3 becomes 0x01, thus no padding error



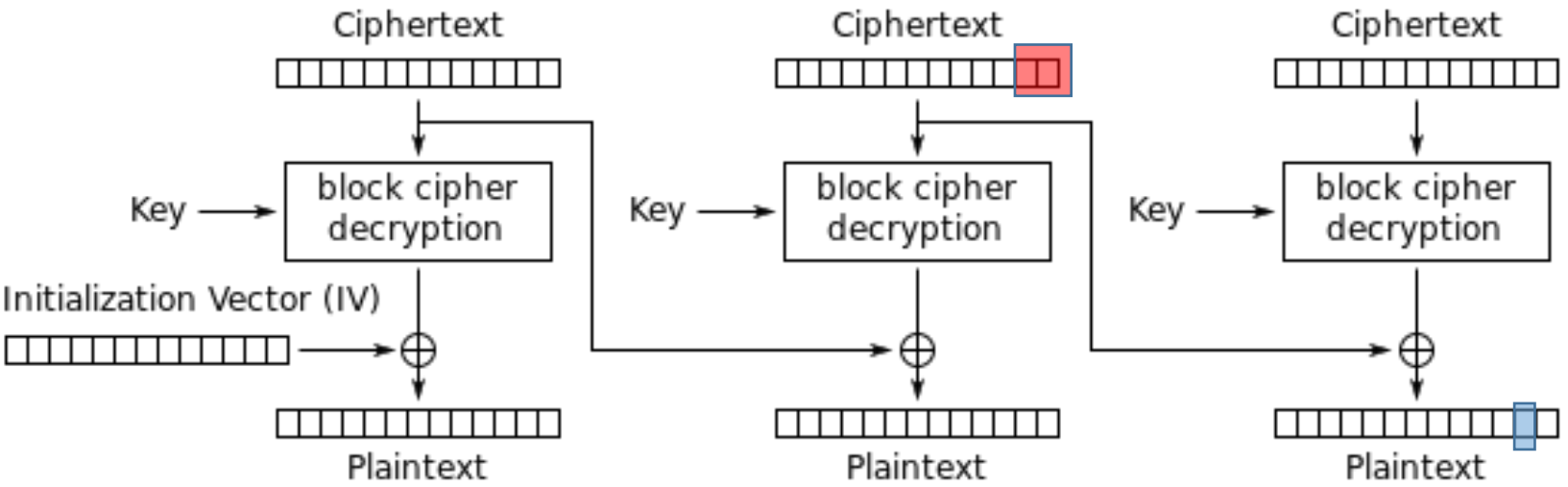
Cipher Block Chaining (CBC) mode decryption

CBC Padding Oracle Attack (against PKCS#7 Padding)

Guess the second to the last byte of P_3, say z_{-2}

Change the last two byte of C_2 into
$b_{-1} = b_{-1} \oplus z_{-1} \oplus 0x02$ and $b_{-2} = b_{-2} \oplus z_{-2} \oplus 0x02$

If the guess is correct, then the last two byte of P_3 becomes 0x02 0x02, thus no padding error



Cipher Block Chaining (CBC) mode decryption

CBC Padding Oracle Attack (against PKCS#7 Padding)

Repeat the process until recovering the full block

O(256*#of bytes)

How about Random Padding (such as in SSL3.0)?

- POODLE attack (Padding Oracle On Downgraded Legacy Encryption) against SSL 3.0
- Homework 2

https://blog.cryptographyengineering.com/2014/10/15/attack-of-week-poodle/

How to prevent padding oracle attacks?

Ensure no information leak about the padding

Verify message authentication code (MAC) or signature before checking padding

MAC-then-encrypt is bad; use encrypt-then-MAC or authenticated encryption

How to prevent POODLE attacks?

Disable SSL 3.0

Use TLS_FALLBACK_SCSV

- Signaling Cipher Suite Value (SCSV)

Conclusion

SSL Report: my.ntu.edu.tw (140.112.8.80)

現在，再看一次?

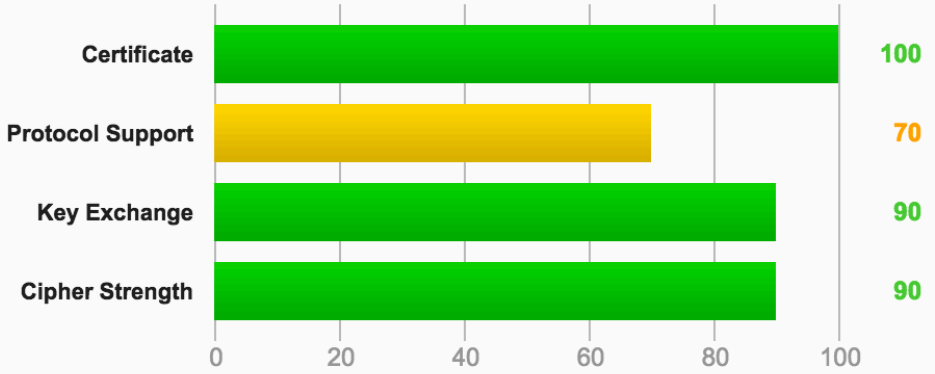
Assessed on: Sun Apr 26 05:25:50 PDT 2015 | **HIDDEN** | Clear cache

Scan Another »

Summary

Overall Rating

B



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server uses SSL 3, which is obsolete and insecure. Grade capped to B. MORE INFO »

Certificate uses a weak signature. When renewing, ensure you upgrade to SHA2. MORE INFO »

The server supports only older protocols, but not the current best TLS 1.2. Grade capped to B.

This server accepts the RC4 cipher, which is weak. Grade capped to B. MORE INFO »

There is no support for secure renegotiation. MORE INFO »

The server does not support Forward Secrecy with the reference browsers. MORE INFO »

https://www.ssllabs.com/ssltest

Using HTTPS is much better than HTTP, then why hasn't it been used everywhere?

Minor problems:

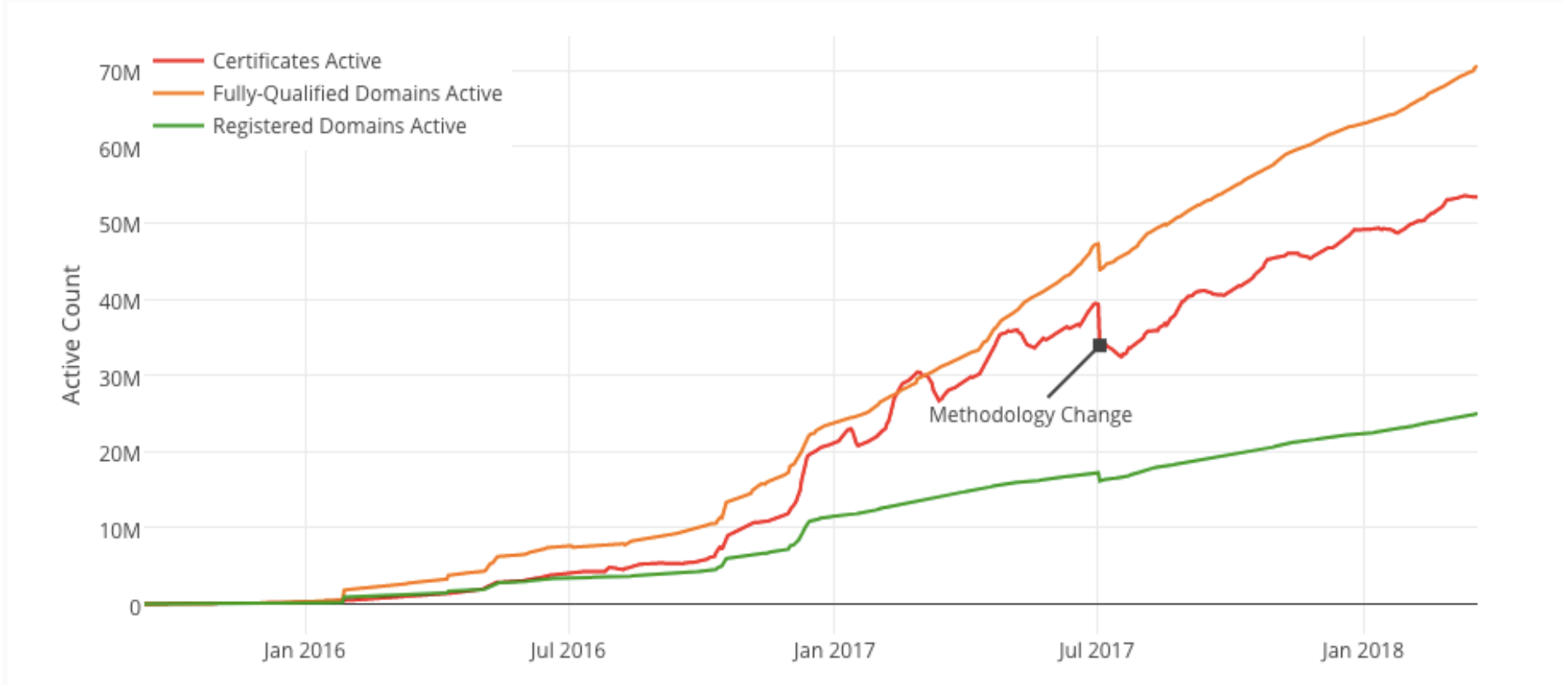
- Heavyweight public-key crypto 現在機器都很強
- Certificate cost 已經有free certificate
- DoS amplification
- Break caching 無法用proxy cache內容，latency增高

Major problem: high latency due to extra round trips

- TLS 1.3 will add support for 1-RTT handshakes, which might alleviate this problem

Let’s Encrypt

More than 50 Million Websites Install Free Certificate (and Counting…)



TLS 1.3

Simplicity, ”fewer, better choices"

No weak crypto

- No RC4, MD5, 3DES

Fewer choices

- No AES-CBC
- No arbitrary DH groups, curves

1-RTT, 0-RTT

https://tools.ietf.org/html/draft-ietf-tls-tls13-28

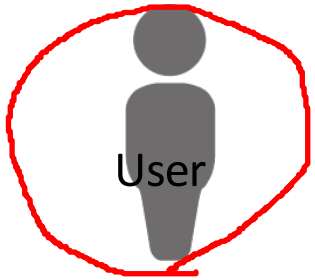
Trust assumptions for TLS

- No CA root key compromised
- No employee of any CA can issue a bogus certificate
- All CAs thoroughly verify owner of domain to issue certificate to

Crypto, protocol, implementation

- All crypto algorithms are "secure", SHA-1, MD5 vulnerabilities?
- Protocol and implementation are correct

- All root certificates in browser are correct
  - No bogus certificate was added by adversary
  - Browser was securely downloaded
  - No virus has altered root certificates
- Browser does not contain any remotely exploitable vulnerabilities
- Malicious site cannot overwrite lock and https with bogus images
- No unicode / homograph vulnerability



- User checks for https and for closed lock
- User cannot be tricked into bogus URL, e.g., http://www.cnn.com@1234567
- User does not accept bogus certificate
- User does not ignore warning if certificate is from different site

Problems that SSL / TLS does not take care of

TCP-level denial of service

TCP/IP的header都不是encrypted的

- SYN flooding
- RST injection (but does protect against data injection!)

SQL injection / XSS / server-side coding/logic flaws

Vulnerabilities introduced by server inconsistencies

Browser coding/logic flaws

User flaws

- Weak passwords
- Phishing

Issues of trust …

- User has to make correct trust decisions