Public-key Infrastructure

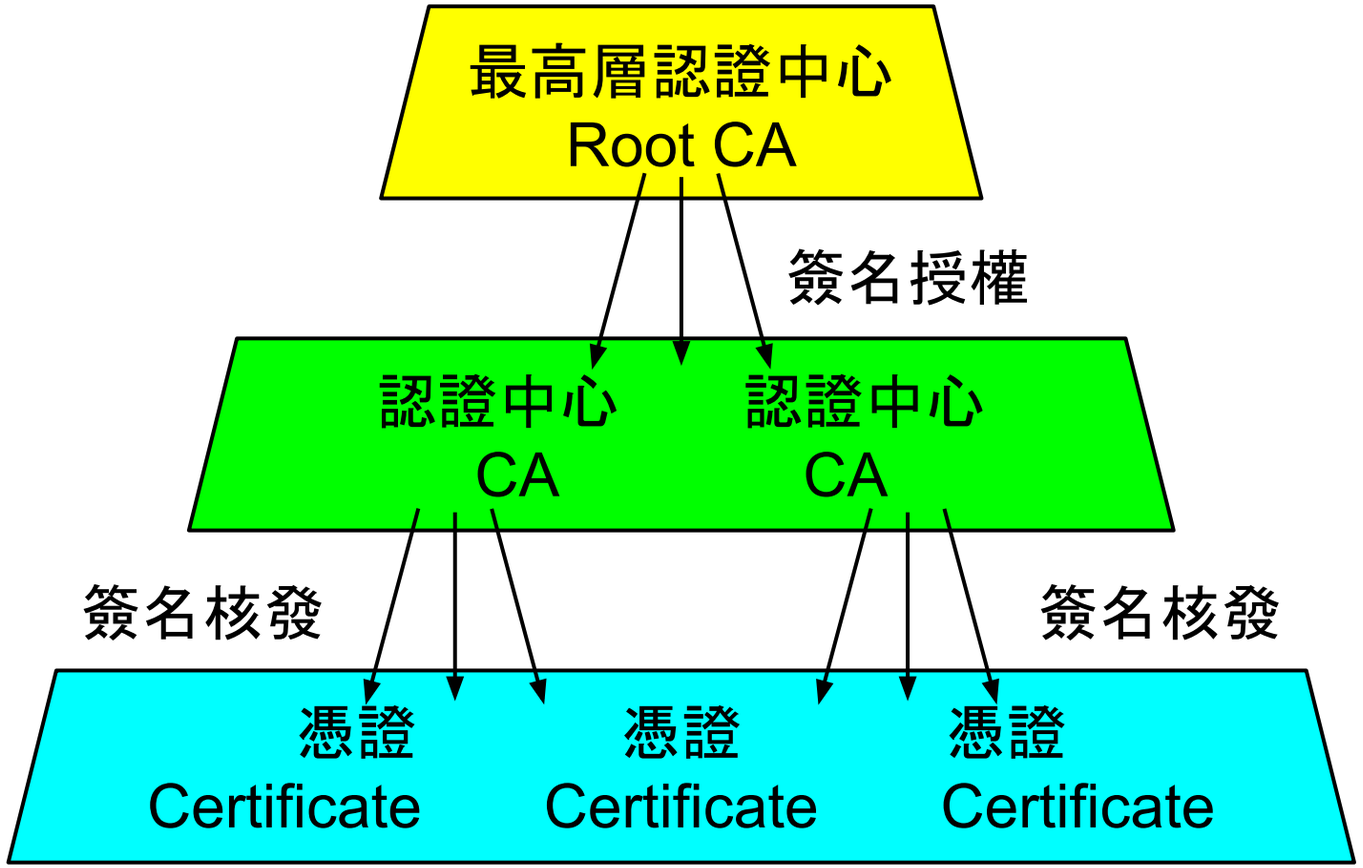
bogay (2025, CC BY-SA)
tsaimh (2022-2024, CC BY-SA)
wangth (2017-2021, CC BY-SA)
? (1996-2016)

陽明交大資工系資訊中心
Information Technology Center of Department of Computer Science, NYCU

Public-key Infrastructure

- A set of hardware, software, people, policies, and procedures
- To create, manage, distribute, use, store, and revoke digital certificates
- Encryption, authentication, signature
- Bootstrapping secure communication protocols

CA: Certificate Authority (1)



CA: Certificate Authority (2)

- Certificate
  - Contains data of the owner, such as Company Name, Server Name, Name, Email, Address,…
  - Public key of the owner.
  - Followed by some digital signatures.
    - Sign for the certificate.
  - In X.509
    - A certificate is signed by a CA.
    - To verify the correctness of the certificate, check the signature of CA.

CA: Certificate Authority (3)

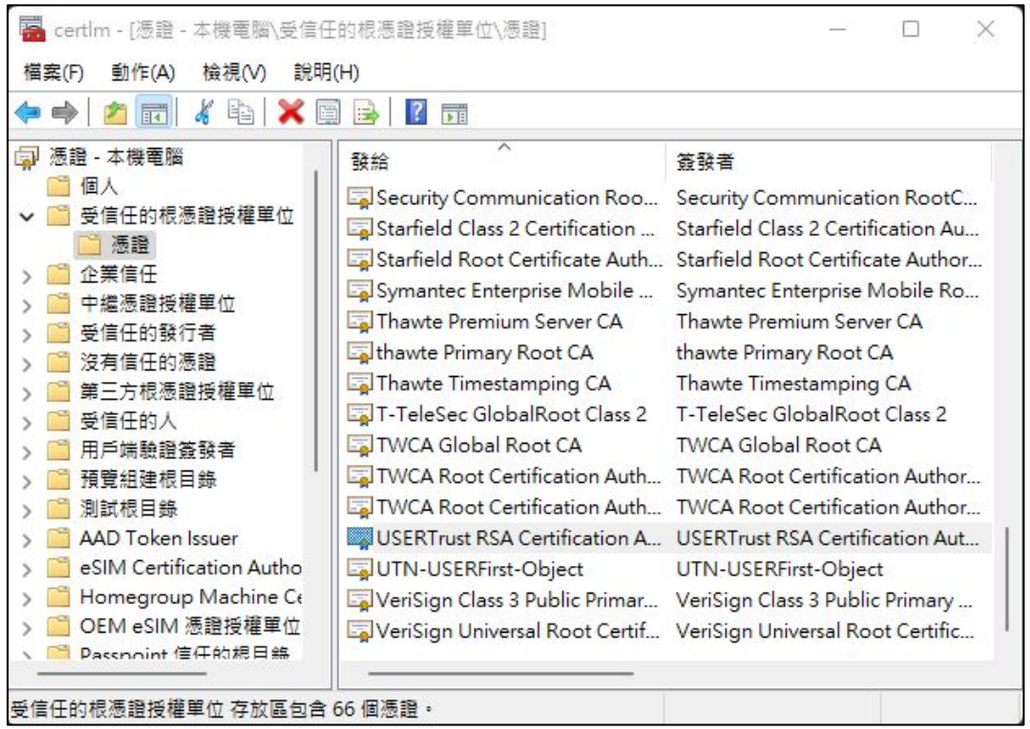
- Certificate Authority (CA)
  - “憑證授權” in Windows CHT version.
  - In X.509, it is itself a certificate.
    - The data of CA.
    - To sign certificates for others.
  - Each CA contains a signature of Root CA.
  - To verify a valid certificate
    - Check the signature of Root CA in the certificate of CA.
    - Check the signature of CA in this certificate.
- Reference: http://www.imacat.idv.tw/tech/sslcerts.html

What is a CA ? (1)

- *Certificate Authority* (認證中心)
- Trusted server which signs certificates
- One private key and relative public key
- Tree structure of X.509
  - *Root CA*

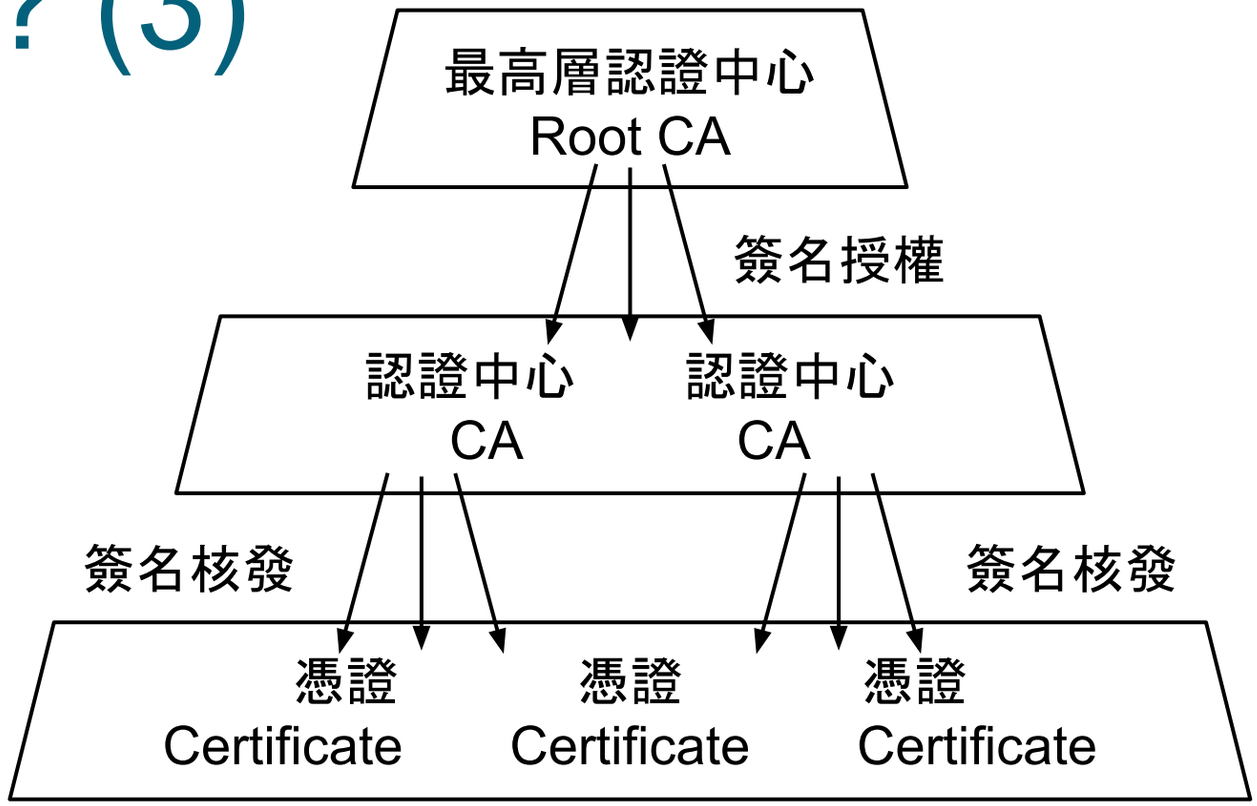
What is a CA ? (2)

- Root CA (最高層認證中心)
  - In Windows:「受信任的根憑證授權單位」
  - Root CA do not sign the certificates for users
    - Authorize CA to sign the certificates for users, instead.
  - Root CA signs for itself
  - To trust Root CA
    - Install the certificate of Root CA via secure channel.
  - security/ca_root_nss
    - Root certificate bundle from the Mozilla Project



What is a CA ? (3)

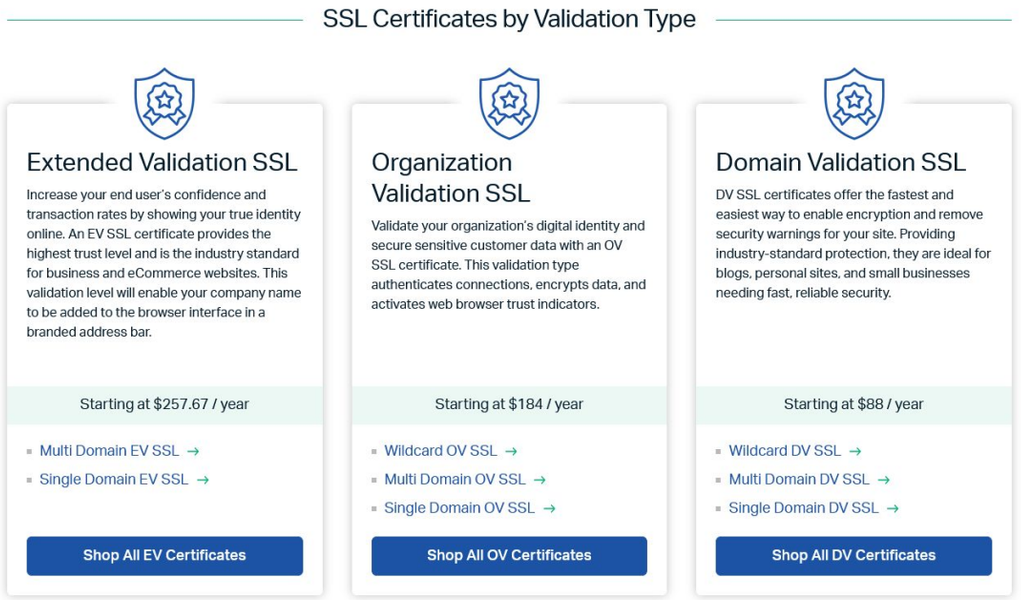
- Tree structure of CA

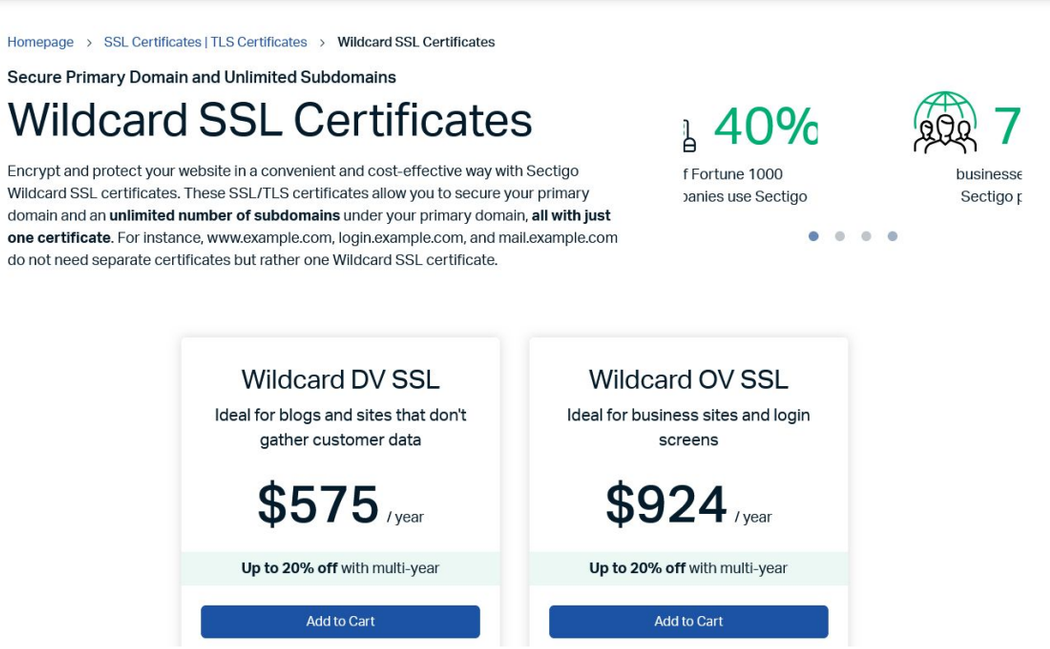


- Cost of certificate
  - PublicCA : NT $3,780 / per year / per host
  - Myself : NT $0
  - Let's Encrypt : NT $0
    - https://letsencrypt.org

What is a CA ? (4)

- Validation
  - DV (Domain Validation)
  - OV (Organization Validation)
  - EV (Extended Validation)





[Purchase Wildcard SSL Certificates | Sectigo® Official]

Certificate Validity

- Certificate max validity
  - ~ March 15, 2026: 398 days.
  - ~ March 15, 2026: 200 days.
  - ~ March 15, 2027: 100 days.
  - ~ March 15, 2029: 47 days.

[Voting Period Begins: SC-081v3: Introduce Schedule of Reducing Validity and Data Reuse Periods]

Voting Period Begins: SC-081v3: Introduce Schedule of Reducing Validity and Data Reuse Periods 瀏覽次數：53,608 次

Clint Wilson
收件者：server...@groups.cabforum.org

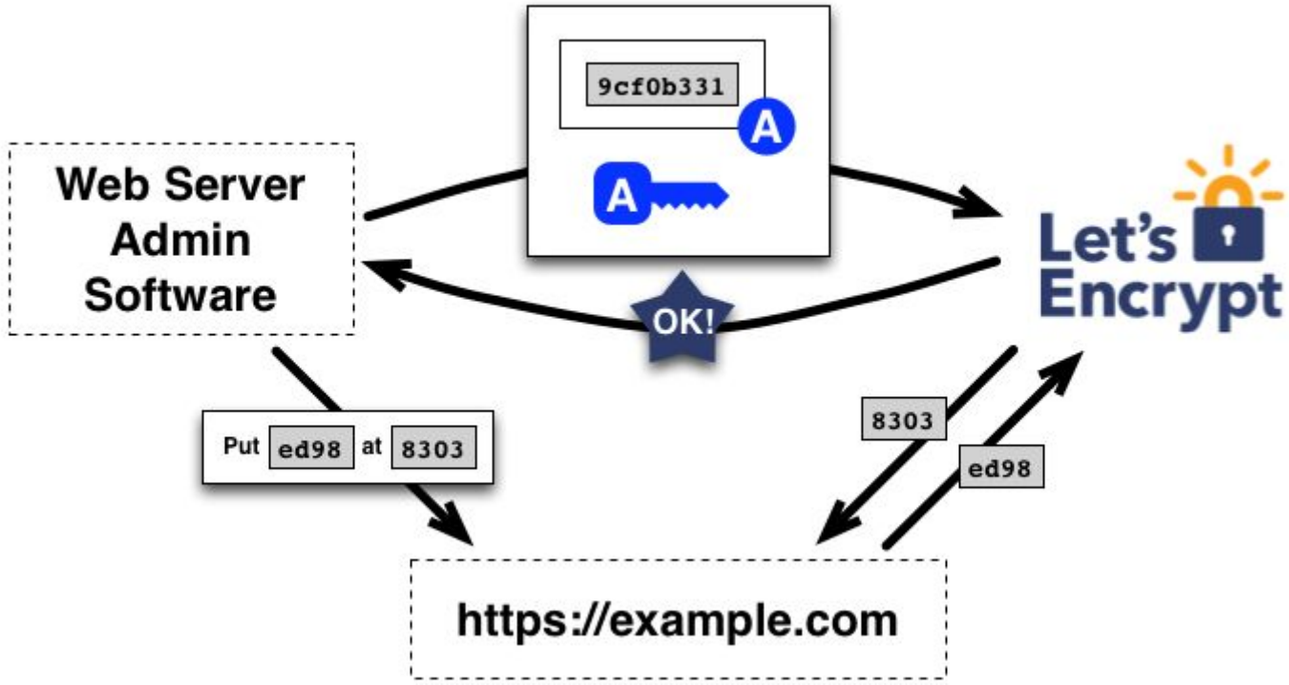
Ballot SC-081v3

Overview

- Expand Section 4.2.1 to detail the allowed data reuse periods for validation data (both for domains/IPs and for everything else in Section 3.2)
  - Eventual reduction of non-SAN validation data reuse from 825 to 398 days
  - Eventual reduction of SAN validation data reuse from 398 days to 10 days
- Expand Section 6.3.2 to detail a schedule for reducing Public TLS certificate maximum validity periods in coming years
  - Eventual reduction of maximum validity period from 398 days to 47 days
- These reductions are proposed to occur starting in March 2026 and concluding in March 2029

ACME

- RFC 8555 - Automatic Certificate Management Environment (ACME)
- Challenges
  - http-01, dns-01, tls-alpn-01 (RFC8737)

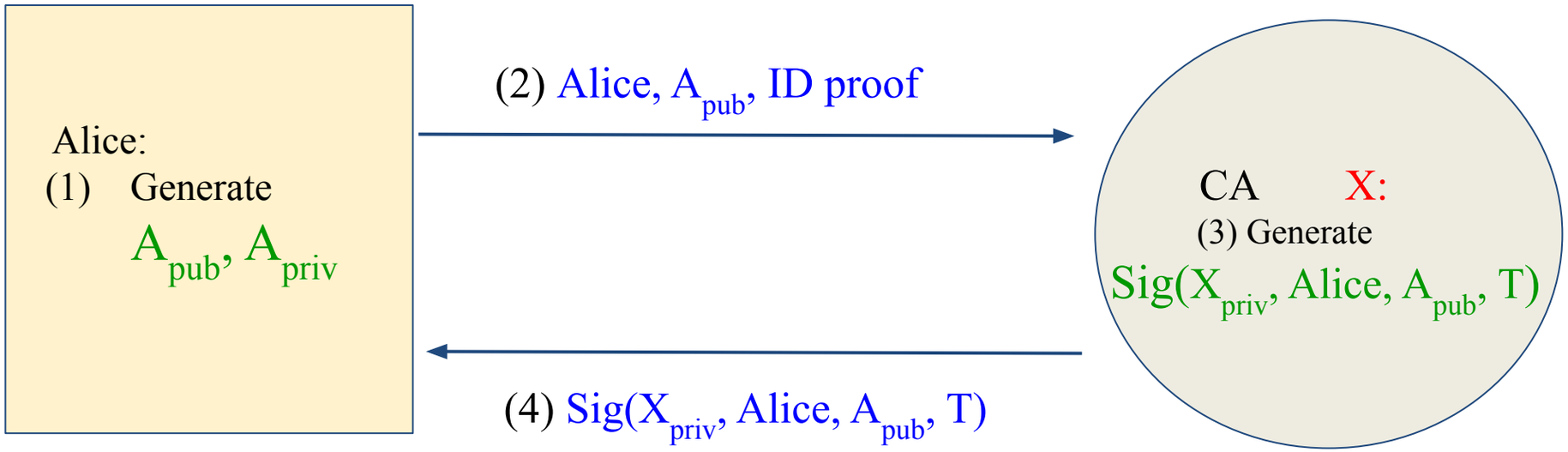


Let's Encrypt 的運作原理

Certificate (1)

- Digital Certificate, Public-key Certificate, Network Identity
- A certificate is issued by a CA *X*
- A certificate of a user A consists:
  - The name of the issuer CA *X*
  - His/her public key A_{pub}
  - The signature $Sig(X_{priv}, A, A_{pub})$ by the CA *X*
  - The expiration date
  - Applications
    - Encryption / Signature
- RFC 5280

Certificate (2)



$Cert_{A,X}$=[Alice, A_{pub}, $Sig(X_{priv}, Alice, A_{pub}, T)$]

Note: CA does not know A_{priv}

Certificate (3)

- Guarantee of CA and certificate
  - Guarantee the public key is of someone
  - Someone is not guaranteed to be safe
- Security of transmitting DATA
  - Transmit session key first
    - Public-key cryptosystem
  - Transmit DATA by session key
    - Symmetric-key cryptosystem

SSL & TLS

陽明交大資工系資訊中心

Information Technology Center of Department of Computer Science, NYCU

SSL/TLS

- SSL/TLS
  - Provide communication security over the Internet
    - Prevent eavesdropping and tampering
  - Encrypt segments over Transport Layer

※ SSL: Secure Sockets Layer

※ TLS: Transport Layer Security

History – (1)

- SSL - developed by Netscape
  - SSL 1.0: never publicly released
  - SSL 2.0: released in 1995
    - A number of security flaws
  - SSL 3.0: released in 1996
    - A complete redesign
    - Newer versions of SSL/TLS are based on SSL 3.0
  - SSL 2.0 was prohibited in 2011 by RFC 6176, and SSL 3.0 followed in June 2015 by RFC 7568

History – (2)

- TLS - IETF RFC
  - TLS 1.0 (SSL 3.1): RFC 2246 in 1999.
    - Backward compatible to SSL 3.0
    - CBC vulnerability discovered in 2002
  - TLS 1.1 (SSL 3.2): RFC 4346 in 2006
    - Prevent CBC attacks
  - TLS 1.2 (SSL 3.3): RFC 5246 in 2008
    - Enhance security strength
    - Introduce new cryptographic algorithms
  - TLS 1.3: RFC 8446 in 2018

SSL/TLS Negotiation

- (C) Request a secure connection, and present a list of supported ciphers and hash functions
- (S) Select common cipher and hash function, and send back with server's digital certificate
- (C) Confirm the validity of the certificate
- (C) Encrypt a random number with server's public key, and send it to server
- (C/S) Generate session key(s) from the random number

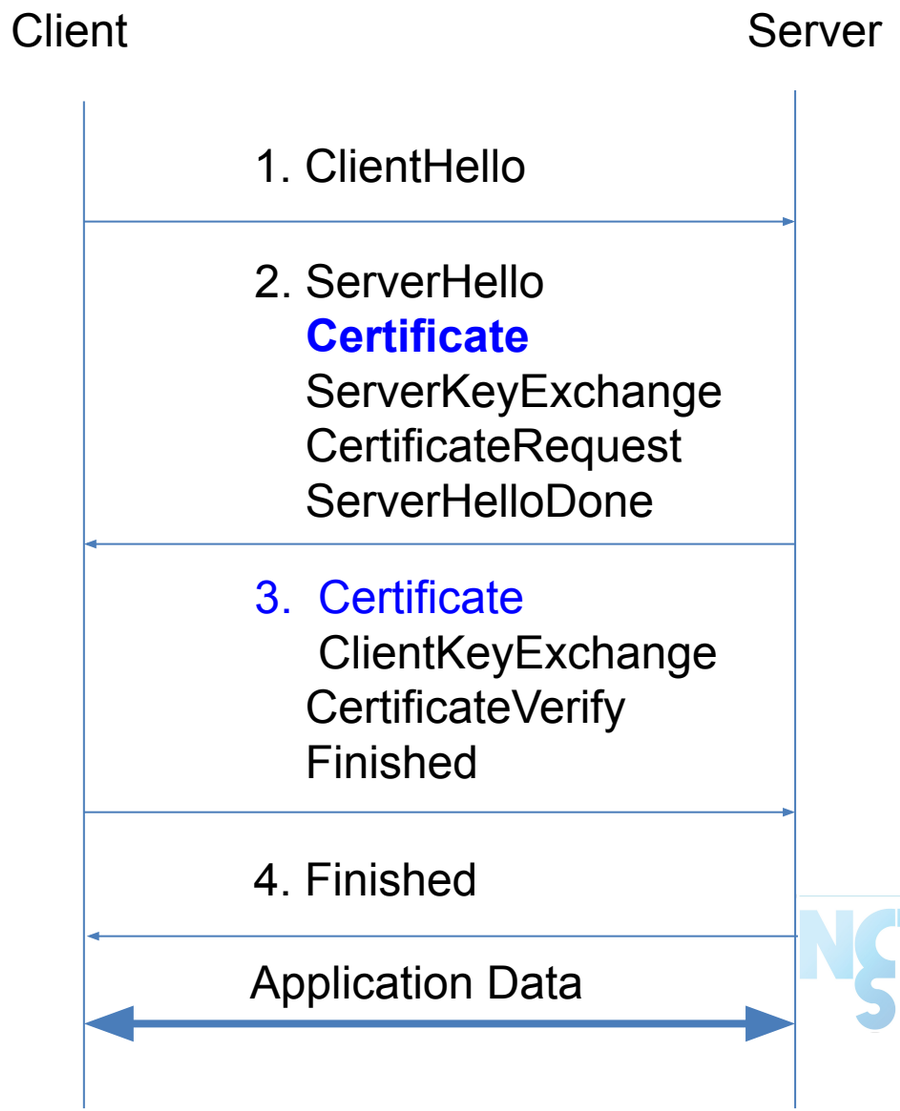
C: Client / S: Server

SSL/TLS Applications

- Implemented on top of Transport Layer protocols
  - TCP
  - UDP (DTLS)
- Protect application-specific protocols
  - HTTP, FTP, SMTP, NNTP, …
  - VPN (OpenVPN), SIP, VoIP
- Activate SSL/TLS connection
  - Use a different port number (https/443, smtps/465)
  - Use a protocol specific mechanism (STARTTLS)

SSL/TLS Problems for Virtual Hosts

- At step 2, the server has to select and send the certificate to the client immediately after the ClientHello.
- At this moment, it doesn't have any information about the requested host name (which will only follow later in the Host: HTTP header, after completion of the full handshake).



Support for Named-based Virtual Servers

- Two solutions:
  - Wildcard certificate (all virtual servers belong to the same domain)
  - Add all virtual host names in subjectAltName
    - Disadvantages:

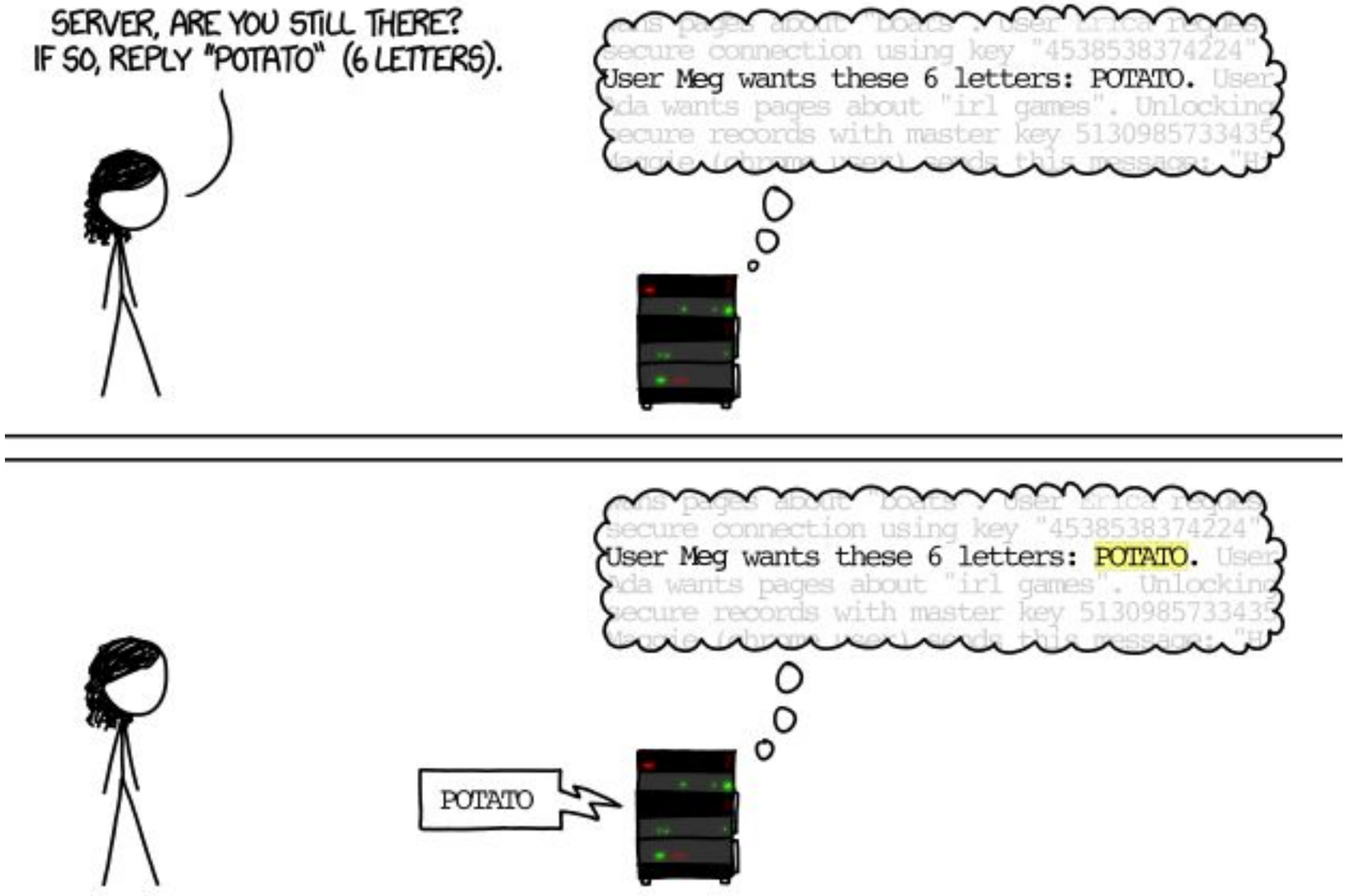
    Certificate needs reissuing whenever adding a new virtual server
- Server Name Indication (SNI) extension (RFC 4366)
  - Allows the client to include the FQDN of the host the client wants to connect to in the ClientHello message.
  - http://wiki.apache.org/httpd/NameBasedSSLVHostsWithSNI
  - The client browser must also support SNI
  - https://www.digicert.com/ssl-support/apache-multiple-ssl-certificates-using-sni.htm

OpenSSL

Heartbleed bug

- CVE-2014-0160
- http://heartbleed.com/
- https://www.openssl.org/news/secadv/20140407.txt
- Test https://filippo.io/heartbleed/

Heartbleed illustrated (1)



https://teskalabs.com/blog/heartbleed

Heartbleed illustrated (2)

https://teskalabs.com/blog/heartbleed

Heartbleed illustrated (3)

https://teskalabs.com/blog/heartbleed

Security Advisories

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
- https://www.freebsd.org/security/advisories/FreeBSD-SA-14:06.openssl.asc
- https://security-tracker.debian.org/tracker/CVE-2014-0160

Example: Apache SSL settings

Example: Apache SSL settings - Flow

- Flow
  - Generate RootCA
    - Generate private key of RootCA
    - Sign the certificate itself.
  - Generate certificate of Web Server
    - Generate private key of Web Server
    - Fill the Request of certificate
    - Sign the certificate using RootCA
  - Modify apache configuration and restart apache

Example: Apache SSL settings - Generate private key of RootCA

- openssl genrsa -aes256 -out rootca-key-file num
  - % openssl genrsa -aes256 ~~-rand /etc/ssl/RootCA/private/.rnd~~ \
    -out /etc/ssl/RootCA/private/rootca.key.pem 2048
    - Note: phrase are asked (something like password)
    - openssl-genrsa(1)
    - "Random State Options" in openssl(1)
- `openssl req -newkey rsa:2048 -noenc \`
  `-keyout rootca.key -x509 \`
  `-days 365 \`
  `-out rootca.pem`

Example: Apache SSL settings - Sign the certificate itself

- ```
  openssl req \
    -newkey rsa:2048 -noenc -keyout rootca.key \
    -x509 -days 365 -out rootca.pem
  ```
  ```

Example: Apache SSL settings - Generate private key of Web Server

- `openssl req \`
  `-newkey rsa:2048 -noenc -keyout nasa.key \`
  `-out nasa.csr`

Example: Apache SSL settings - Fill the Request of Certificate

- `openssl req \`
  `-newkey rsa:2048 -noenc -keyout nasa.key \`
  `-out nasa.csr`
  - Certificate Signing Request (CSR)

```
Enter pass phrase for rootca-key-file:

Country Name (2 letter code) [AU]:TW
State or Province Name (full name) [Some-State]:Taiwan
Locality Name (eg, city) []:HsinChu
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NCTU
Organizational Unit Name (eg, section) []:CS
Common Name (eg, YOUR name) []:nasa.cs.nctu.edu.tw
Email Address []:tsaimh@cs.nctu.edu.tw

A challenge password []: (No need, Enter please)
An optional company name []: (Enter please)
```

Example: Apache SSL settings - Sign the certificate using RootCA

- Transmit host-req-file to Root CA, and do following steps in RootCA
  - ```
    openssl x509 -req -in nasa.csr \
        -extensions usr_cert \
        -CA rootca.pem \
        -CAkey rootca.key \
        -CAserial rootca.srl \
        -out nasa.pem
    ```
- Transmit host-crt-file back to Web Server
    ```

Example: Apache SSL settings - Certificate Authority

- Include etc/apache22/extra/httpd-ssl.conf

```
##
## SSL Virtual Host Context
##
<VirtualHost _default_:443>
#   General setup for the virtual host
DocumentRoot /home/wwwadm/data
<Directory "/home/wwwadm/data">
    Options Indexes FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
</Directory>
ServerName nasa.cs.nctu.edu.tw:443
ServerAdmin tsaimh@nasa.cs.nctu.edu.tw
ErrorLog /var/log/httpd/nasa.cs-error.log
CustomLog /var/log/httpd/nasa.cs-access.log common
Q
SSLEngine on
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:!SSLv2:+EXP:+eNULL
SSLCertificateFile /etc/ssl/nasa/nasa.crt.pem
SSLCertificateKeyFile /etc/ssl/nasa/private/nasa.key.pem
```

Example: Caddy SSL settings

- Update caddyfile

  tls (Caddyfile directive) — Caddy Documentation

```
example.com {
    tls /etc/ssl/nasa/nasa.crt.pem /etc/ssl/nasa/private/nasa.key.pem
    # ...existing configuration
}
```

View the content of Certificate - (1)

- % vim host-crt-file

```
-----BEGIN CERTIFICATE-----
MIIE0DCCA7igAwIBAgIJAL5UBzbv+hl1MA0GCSqGSIb3DQEBBQUAMIGgMQswCQYD
VQQGEwJUVzEPMA0GA1UECBMGVGFpd2FuMRAwDgYDVQQHEwdIc2luQ2h1MQ0wCwYD
VQQKEwROQ1RVMQswCQYDVQQLEwJBTTEiMCAGA1UEAxMZZXZpbGJpZzUubWF0aC5u
.....
9xMw8qMBHnxUVHOUVbECAwEAAaOCAQkwggEFMB0GA1UdDgQWBBR958Azmc9N7gbm
kFLgfOpw+9RW9TCB1QYDVR0jBIHNMIHKgBR958Azmc9N7gbmkFLgfOpw+9RW9aGB
pqSBozCBoDELMAkGA1UEBhMCVFcxDzANBgNVBAgTBlRhaXdhbjEQMA4GA1UEBxMH
SHNpbkNodTENMAsGA1UEChMETkNUVTELMAkGA1UECxMCQU0xIjAgBgNVBAMTGWV2
aWxiaWc1Lm1hdGgubmN0dS5lZHUudHcxLjAsBgkqhkiG9w0BCQEWH3JhbmR5QGV2
aWxiaWc1Lm1hdGgubmN0dS5lZHUudHeCCQC+VAc27/oZdTAMBgNVHRMEBTADAQH/
MA0GCSqGSIb3DQEBBQUAA4IBAQCLkNba9LSpCTOh7Ws3h18WSKQXVxnLHxWUepC8
ZG3Q/dT++L54EiyBLmXwnr67gfUPhN1Qb/v1ixThlNBIjIrOZvEiyqjrmrQBABpt
x0+APW8TAdYfslQvGfhDptNeKWoYc7fxlxw3TXwQf2JhL+a10m2ZeEMSg1iuIyqg
+Dq3jGCb3R66NoKo/ToO5J6CAnkG7spYiDNukkvoEPNKaqXMC3K6pOzBDQwWBpH7
pCE9dEqbmHvUb+hwvI2OTJAKcM0G1wBmFF7au1G9e6O9hj34voppLdfVz5+mu5ai
ELgGQXpVrFPSzZG0PyAr5rxtOI8E7yl7jl2pu7yXk9jgsiWl
-----END CERTIFICATE-----
```

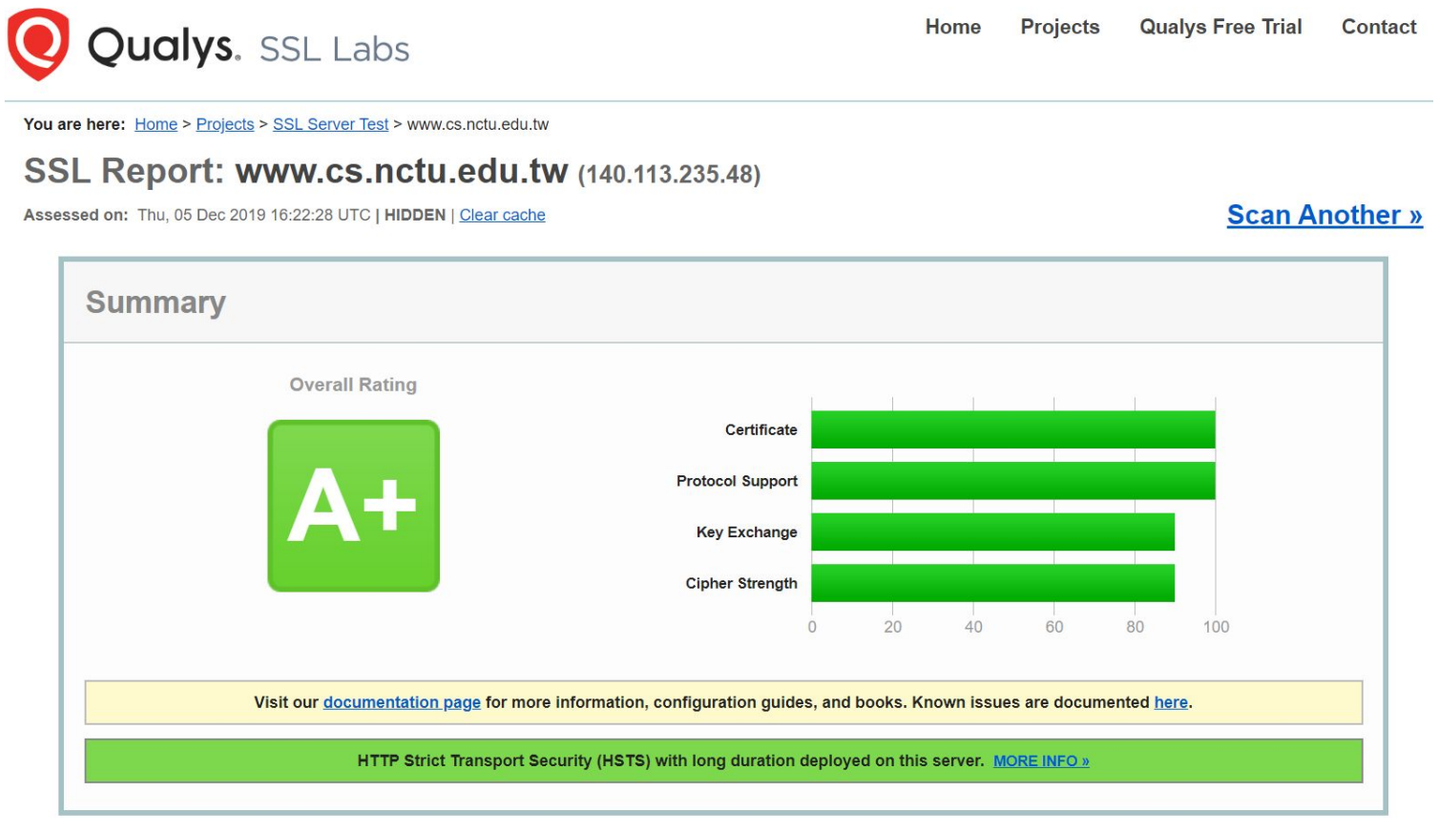
View the content of Certificate - (2)

- % openssl x509 -text -in host-crt-file

```
Certificate:
    Data:
        ......
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=TW, ST=Taiwan, L=HsinChu, O=NCTU, OU=CS, CN=../emailAddress=..
        Validity ...
        Subject: C=TW, ST=Taiwan, L=HsinChu, O=NCTU, OU=CS, CN=../emailAddress=.
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    ......
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            ......
    Signature Algorithm: sha1WithRSAEncryption
        8b:90:d6:da:f4:b4:a9:09:33:a1:ed:6b:37:87:5f:16:48:a4:
        ......
        e0:b2:25:a5
-----BEGIN CERTIFICATE-----
MIIE0DCCA7igAwIBAgIJAL5UBzbv+hl1MA0GCSqGSIb3DQEBBQUAMIGgMQswCQYD
......
ELgGQXpVrFPSzZG0PyAr5rxtOI8E7yl7jl2pu7yXk9jgsiWl
-----END CERTIFICATE-----
```

SSL Server Test

- https://www.ssllabs.com/ssltest/
- https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices



Appendix: PGP

PGP

- Pretty Good Privacy
- Public key system
  - Encryption
  - Signature
- security/gnupg (GNU Privacy Guard)
- Will talk more in Network Administration
- Reference:
  - http://security.nknu.edu.tw/textbook/chap15.pdf
  - http://blog.theerrorlog.com/using-gpg-zh.html