

Public-key Infrastructure

wangth (2017-2020, CC BY-SA)
? (1996-2016)

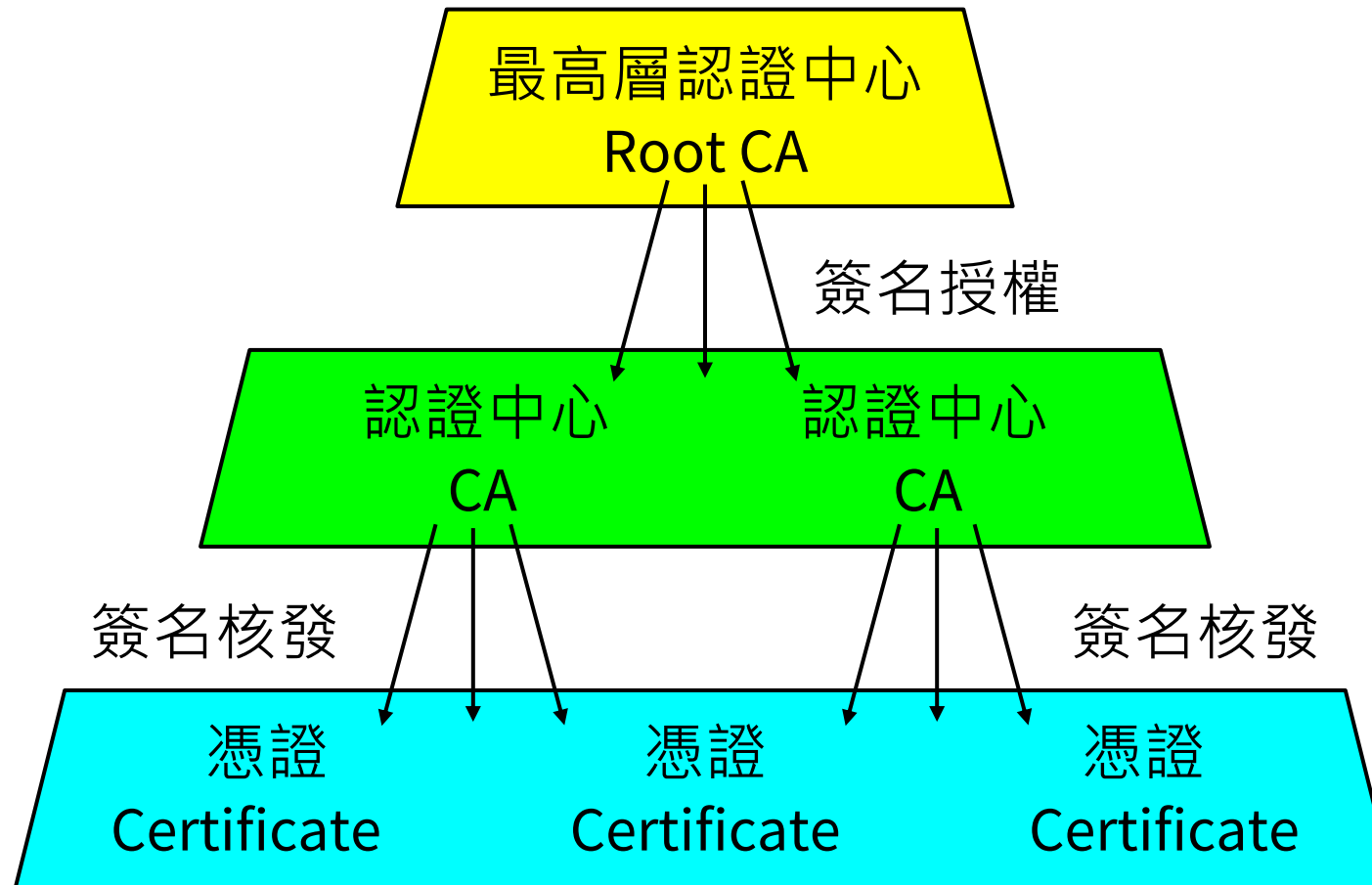
交大資工系資訊中心

Computer Center of Department of Computer Science, NCTU

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)

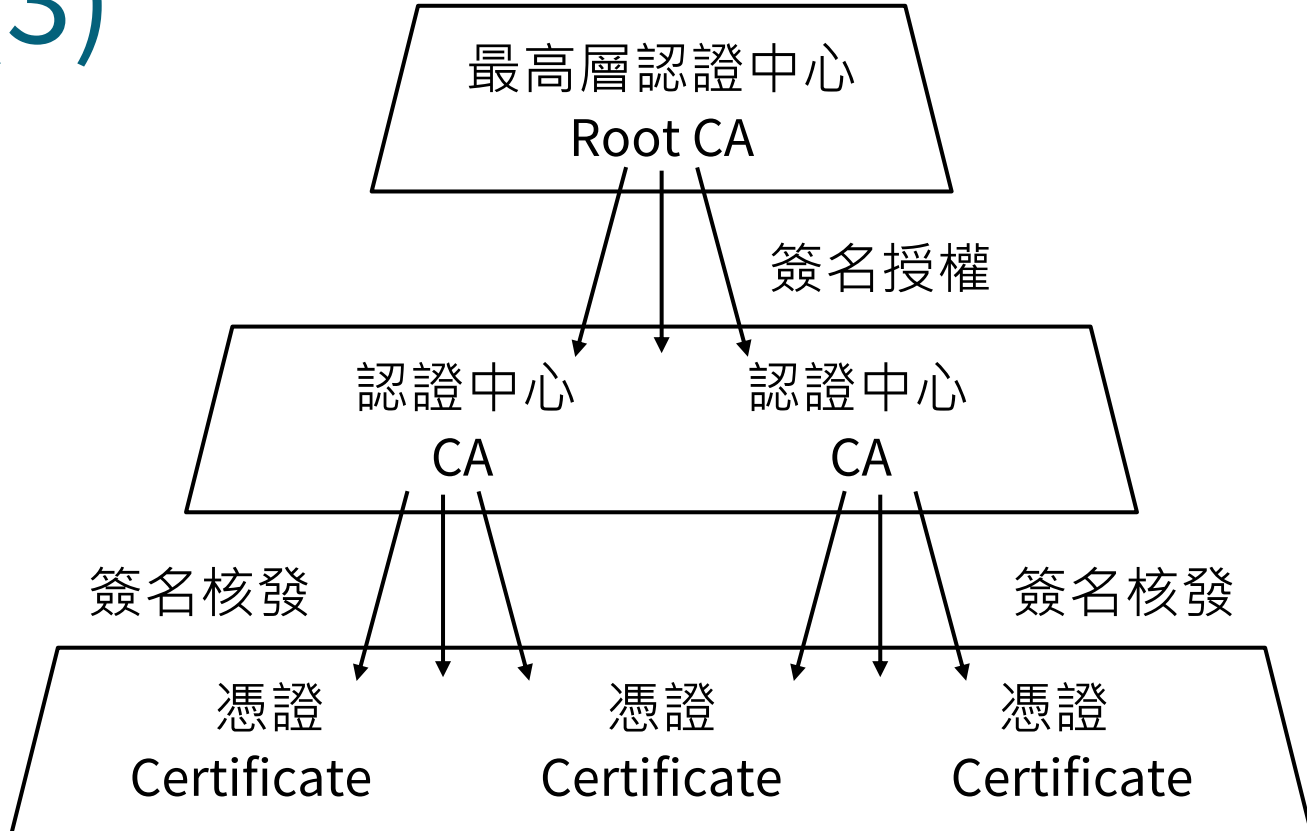
- *Certificate Authority* (認證中心)
- Trusted server which signs certificates
- One **private key** and relative **public key**
- Tree structure of **X.509**
 - *Root CA*

What is a CA ? (2)

- **Root CA** (最高層認證中心)
 - In Microsoft: 「**根目錄授權憑證**」
 - Root CA do not sign the certificates for users
 - Authorize CA to sign the certificates for users, instead.
 - Root CA signs for itself
 - It is in the sky
 - 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



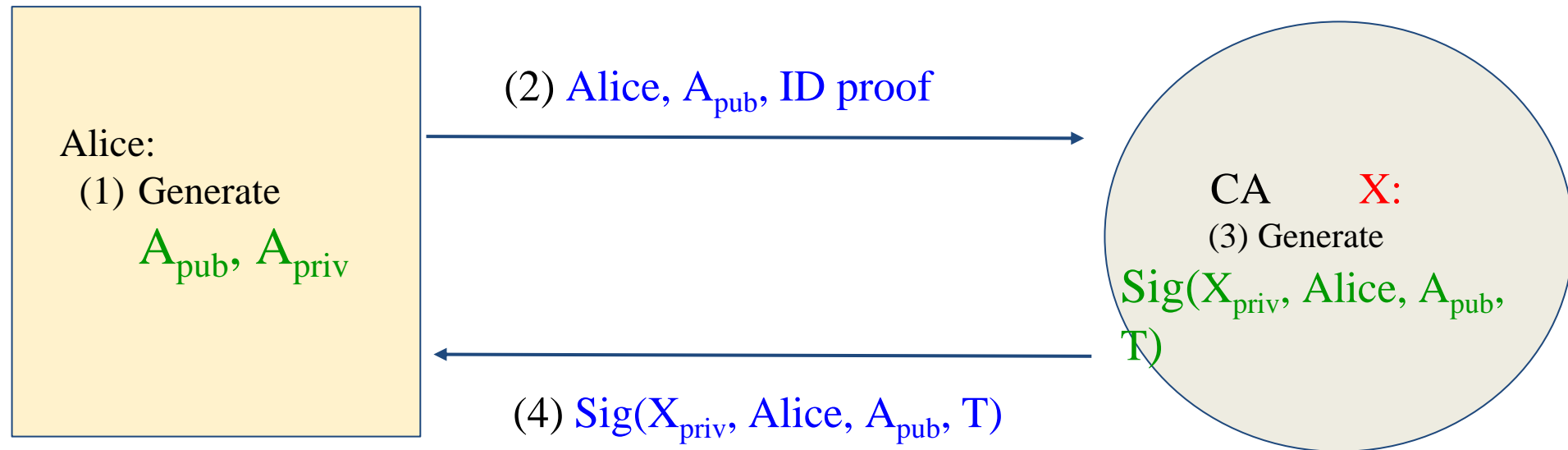
- Cost of certificate

- PublicCA : NT \$9,600 / per year / per host
- Myself : NT \$0
- Let's Encrypt : NT \$0
 - <https://letsencrypt.org>

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 $\text{Sig}(X_{\text{priv}}, A, A_{\text{pub}})$ by the CA X
 - The expiration date
 - Applications
 - Encryption / Signature

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

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/433, smtps/465)
 - Use a protocol specific mechanism (STARTTLS)

Support for Named-based Virtual Servers

- All virtual servers belong to the same domain
 - Wildcard certificate
 - Add all virtual host names in subjectAltName
 - Disadvantages
 - Certificate needs reissuing whenever adding a new virtual server
 - Cannot support named-based virtual hosts for web service
- Server Name Indication (SNI)
 - RFC 4366
 - <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

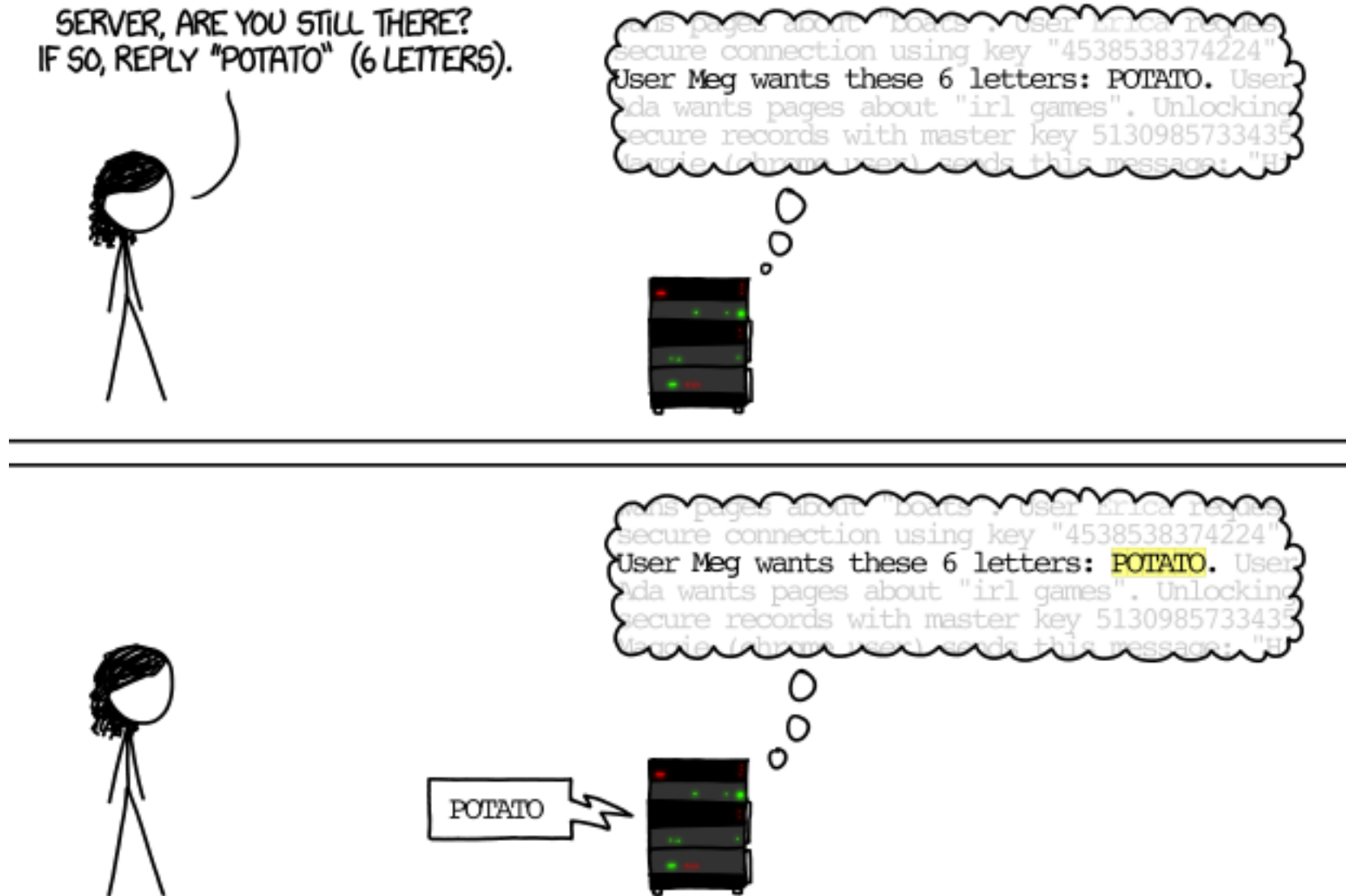
OpenSSL

- <https://www.openssl.org/>
- In system
 - /usr/src/crypto/openssl
- In ports
 - security/openssl
- SSL library selection (in make.conf)
 - WITH_ options is deprecated
 - WITH_OPENSSL_BASE, WITH_OPENSSL_PORT
 - Base OpenSSL and Ports' OpenSSL, LibreSSL or their -devel versions
 - Possible values: base, openssl, openssl-devel, libressl, libressl-devel
 - **DEFAULT_VERSIONS+=ssl=base** https://wiki.freebsd.org/DEFAULT_VERSIONS

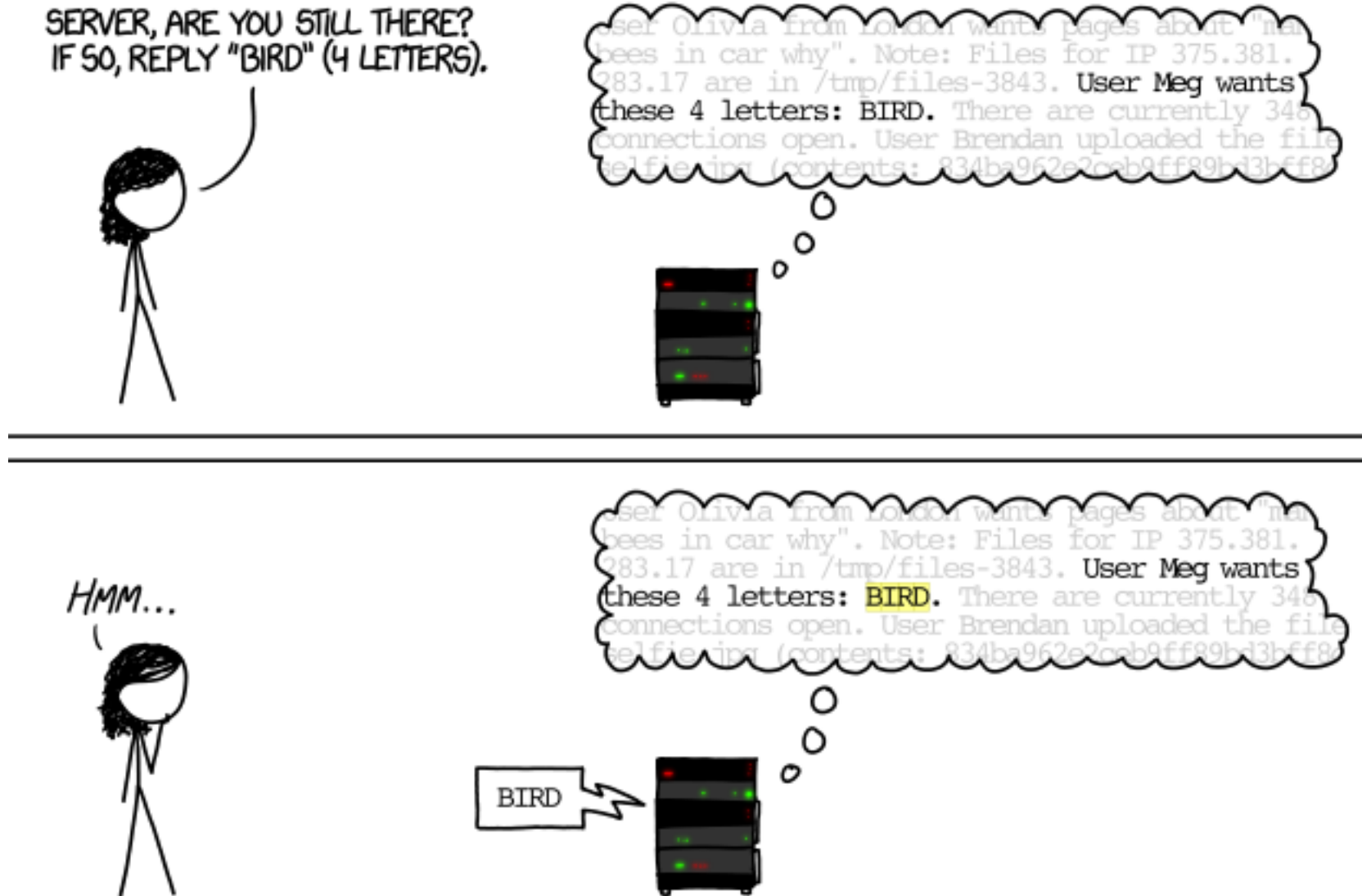
Heartbleed bug

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

Heartbleed illustrated (1)



Heartbleed illustrated (2)



Heartbleed illustrated (3)

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "HAT" (500 LETTERS).

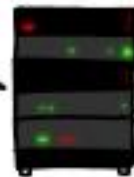


A connection. Jake requested pictures of deer.
User Meg wants these 500 letters: HAT. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User



HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User Isabel requests pages

A connection. Jake requested pictures of deer.
User Meg wants these 500 letters: **HAT**. Lucas
requests the "missed connections" page. Eve
(administrator) wants to set server's master
key to "14835038534". Isabel wants pages about
snakes but not too long". User Karen wants to
change account password to "CoHoBaSt". User



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 random seed
 - Generate RootCA
 - Generate private key of RootCA
 - Fill the Request of Certificate.
 - 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 restart apache

Example: Apache SSL settings -

Generate random seed

- `openssl rand -out rnd-file num`
 - `% openssl rand -out /etc/ssl/RootCA/private/.rnd 1024`
- `chmod go-rwx rnd-file`
 - `% chmod go-rwx /etc/ssl/RootCA/private/.rnd`

Example: Apache SSL settings - Generate private key of RootCA

- `openssl genrsa -aes256 -rand rnd-file -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)`
- `chmod go-rwx rootca-key-file`
 - `% chmod go-rwx /etc/ssl/RootCA/private/rootca.key.pem`

Example: Apache SSL settings - Fill the Request of Certificate

- `openssl req -new -key rootca-key-file -out rootca-req-file`
 - `% openssl req -new -key /etc/ssl/RootCA/private/rootca.key.pem \`
`-out /etc/ssl/RootCA/private/rootca.req.pem`
- `chmod go-rwx rootca-req-file`
 - `% chmod go-rwx /etc/ssl/RootCA/private/rootca.req.pem`

```
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 []:liuyh@cs.nctu.edu.tw
```

```
A challenge password []: (No need · Enter please)
```

```
An optional company name []: (Enter please)
```

Example: Apache SSL settings - Sign the certificate itself (1)

- `openssl x509 -req -days num -sha1 -extfile path_of_openssl.cnf -
extensions v3_ca -signkey
rootca-key-file -in rootca-req-file -out rootca-crt-file`
`% openssl x509 -req -days 5109 -sha1 -extfile /etc/ssl/openssl.cnf
-extensions v3_ca -signkey /etc/ssl/RootCA/private/rootca.key.pem
-in /etc/ssl/RootCA/private/rootca.req.pem -out
/etc/ssl/RootCA/private/rootca.crt.pem`

Example: Apache SSL settings - Sign the certificate itself (2)

- `rm -f rootca-req-file`
 - `% rm -f /etc/ssl/RootCA/private/rootca.req.pem`
- `chmod go-rwx rootca-crt-file`
 - `% chmod go-rwx /etc/ssl/RootCA/private/rootca.crt.pem`

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

- `openssl genrsa -out host-key-file num`
 - `% openssl genrsa -out /etc/ssl/nasa/private/nasa.key.pem 2048`
- `chmod go-rwx host-key-file`
 - `% chmod go-rwx /etc/ssl/nasa/private/nasa.key.pem`

Example: Apache SSL settings - Fill the Request of Certificate

- `openssl req -new -key host-key-file -out host-req-file`
 - `% openssl req -new -key /etc/ssl/nasa/private/nasa.key.pem -out /etc/ssl/nasa/private/nasa.req.pem`
- `chmod go-rwx host-req-file`
 - `% chmod go-rwx /etc/ssl/nasa/private/nasa.req.pem`

Example: Apache SSL settings - Sign the certificate using RootCA (1)

- Transmit host-req-file to Root CA, and do following steps in RootCA
 - `openssl x509 -req -days num -sha1 -extfile path_of_openssl.cnf -extensions v3_ca -CA rootca-crt-file -CAkey rootca-key-file -CAserial rootca-srl-file -CAcreateserial -in host-req-file -out host-crt-file`

Example: Apache SSL settings -

Sign the certificate using RootCA (2)

- Transmit host-req-file to Root CA, and do following steps in RootCA
 - `% openssl x509 -req -days 365 -sha1 -extfile /etc/ssl/openssl.cnf -extensions v3_ca -CA /etc/ssl/RootCA/private/rootca.crt.pem -CAkey /etc/ssl/RootCA/private/rootca.key.pem -CAserial /etc/ssl/RootCA/private/rootca.srl -CAcreateserial -in /etc/ssl/nasa/private/nasa.req.pem -out /etc/ssl/nasa/private/nasa.crt.pem`
 - `rm -f host-req-file` (in both RootCA and Web Server)
 - `% rm -f /etc/ssl/nasa/private/nasa.req.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 liuyh@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
```

View the content of Certificate - (1)

- % vim host-crt-file

```
-----BEGIN CERTIFICATE-----  
MIIE0DCCA7igAwIBAgIJAL5UBzbv+h11MA0GCSqGSIb3DQEEBBQUAMIGgMQswCQYD  
VQQGEwJUVzEPMA0GA1UECBMGVGFpd2FuMRAwDgYDVQQHEwdIc2luQ2h1MQ0wCwYD  
VQQKEwROQ1RVMQswCQYDVQQLLEwJBTTEiMCAGA1UEAxMZZXZpbGJpZzUubWF0aC5u  
.....  
9xMw8qMBHnxUVHOUVbECAwEAAaOCAQkwggEFMB0GA1UdDgQWBBR958Azmc9N7gbm  
kFLgfOpw+9RW9TCB1QYDVR0jBIHNMIHKgBR958Azmc9N7gbmkFLgfOpw+9RW9aGB  
pqSBozCBODELMAkGA1UEBhMCVFcxZzANBgNVBAgTB1RhaXdhbGJlEQMA4GA1UEBxMH  
SHNpbkNodTENMASGA1UEChMETkNUVTELMakGA1UECjxMCQU0xIjAgBgNVBAMTGWV2  
aWxiaWc1Lm1hdGgubmN0dS5lZHUudHcxLjAsBgkqhkiG9w0BCQEWH3JhbmR5QGVS  
aWxiaWc1Lm1hdGgubmN0dS5lZHUudHeCCQC+VAc27/oZdTAMBgNVHRMEBTADAQH/  
MA0GCSqGSIb3DQEEBBQUAA4IBAQClnNba9LSpCTOh7Ws3h18WSKQXVxnLHxWUepC8  
ZG3Q/dT++L54EiyBLmXwnr67gfUPhN1Qb/v1ixTh1NBIjIrOZvEiyqjrmrQBABpt  
x0+APW8TAdYfslQvGfhDptNeKWoYc7fx1xw3TXwQf2JhL+a10m2ZeEMsg1iuIyqg  
+Dq3jGcb3R66NoKo/To05J6CAnkG7spYiDNukkvoEPNkaqXMC3K6pOzBDQwWBpH7  
pCE9dEqbmHvUb+hwvI2OTJAKcM0G1wBmFF7au1G9e609hj34voppLdfVz5+mu5ai  
ELgGQXpVrFPSzZG0PyAr5rxtOI8E7y17j12pu7yXk9jgsiWl  
-----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+h11MA0GCSqGSIb3DQEBBQUAMIGgMQswCQYD
.....
ELgGQXpVrFPSzZG0PyAr5rxtoI8E7yl7jl2pu7yXk9jgsiWl
-----END CERTIFICATE-----
```

SSL Server Test

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

The screenshot shows the Qualys SSL Labs report for the domain **www.cs.nctu.edu.tw**. The overall rating is **A+**. The report includes a navigation menu with links for Home, Projects, Qualys Free Trial, and Contact. The breadcrumb trail indicates the current page is under SSL Server Test. The assessment was performed on Thursday, 05 Dec 2019 at 16:22:28 UTC, and the results are hidden. A 'Scan Another' link is available.

Summary

Overall Rating: **A+**

Category	Score
Certificate	100
Protocol Support	100
Key Exchange	90
Cipher Strength	90

Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. [MORE INFO »](#)

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>