Certificates and SSL/TLS: A Look Inside

Presenter: Charles Mills
Charles Mills Consulting, LLC
charlesm@mcn.org

Abstract

Starting with z/OS V2R3, IBM is no longer shipping "standard" certificate-authority (CA) certificates with RACF, putting more responsibility on you to understand and manage certificates on your own. You may have attended other sessions that have shown how to install a certificate in RACF, ACF2 or TSS. This session will give you an understanding of how the certificate process actually works under the covers. It is equally relevant to RACF, ACF2 and TSS systems.

The session will start with a quick review of the underlying technologies and their limitations in the absence of certificates: secret key, public key, Base64, and digital signatures and hashes; and go on to cover in detail the protocol flows with server certificates, intermediate certificates, CA certificates and revocation lists. Finally the session will introduce you briefly (with resources for further learning) to advanced topics such as Alternative Names, client certificates, code signing, and more.

About the Presenter

Charles has been writing mainframe software for longer than he cares to admit. He developed security software for eight years at CorreLog, where he authored the zDefender and SyslogDefender products which were acquired by BMC. He is currently the Chief Development Officer for Cloud Compiling and also does freelance project-oriented development.

He has a PhD in certificate technology from the University of Hard Knocks.

The University of Certificate Hard Knocks

Windows product

- My introduction to the nitty-gritty of certificates
- Implements both ends of TLS protocol
- Built using OpenSSL
- Open Source – "lightly" documented – forces one to learn as one goes
- Result can be "the most dangerous code in the world" https://bit.ly/2Djr76W

z/OS product

- Built using IBM z/OS SystemSSL ("GSK")
- Designed to force you not to make mistakes
- Highly recommended

Why Certificates?

Automate security for remote connections

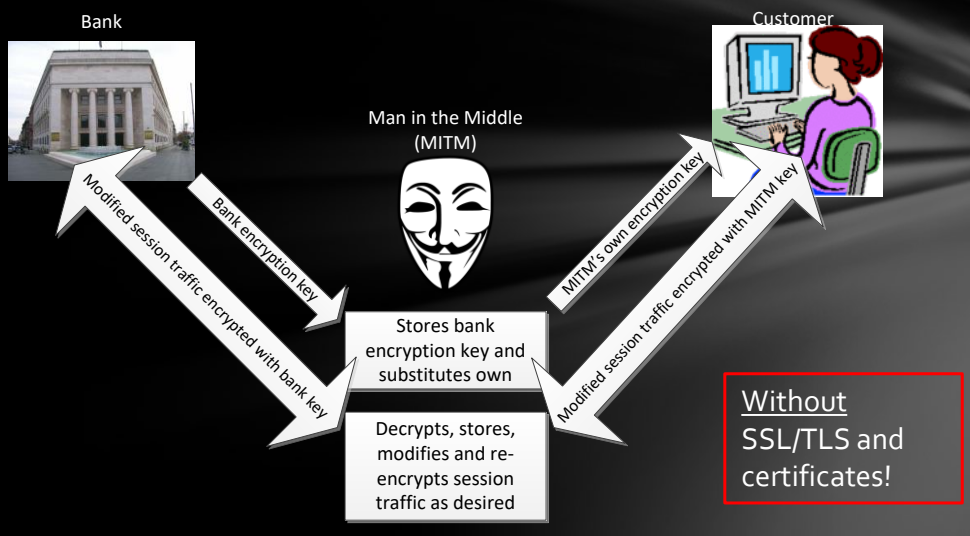
- Authentication: is this site really who it says it is?
- Encryption for data traffic
- For Web, FTP, TN3270 and potentially any similar connection

Authenticate users: is she really who she says she is?

Authenticate e-mail: is this e-mail really from the supposed sender, and how do I know it has not been altered?

Guarantee that software has not been tampered with since it left the publisher

Man in the Middle Attack



Brief history of SSL and TLS ("SSL/TLS")

1994: Netscape, first company to commercialize the Internet, perceives browser communication not secure enough for e-commerce

1994: Develops Secure Sockets Layer (SSL) version 1 (never released)

1995 and 1996: SSL versions 2 and 3 (both now deprecated)

1998: Netscape crashes and burns in Microsoft browser wars

1999: SSL v3 becomes IETF Transport Layer Security (TLS) v1.0

TLS now at Versions 1.2 and 1.3. TLS 1.3 was defined in RFC 8446 in August 2018.

Certificates are a fundamental component of SSL/TLS

http://bit.ly/2DCRGiY
http://bit.ly/2DkGmus

Agenda

100 MPH review of underlying technologies

- With links for additional reference

Details of the certificate protocol flow

100 MPH introduction to some advanced features

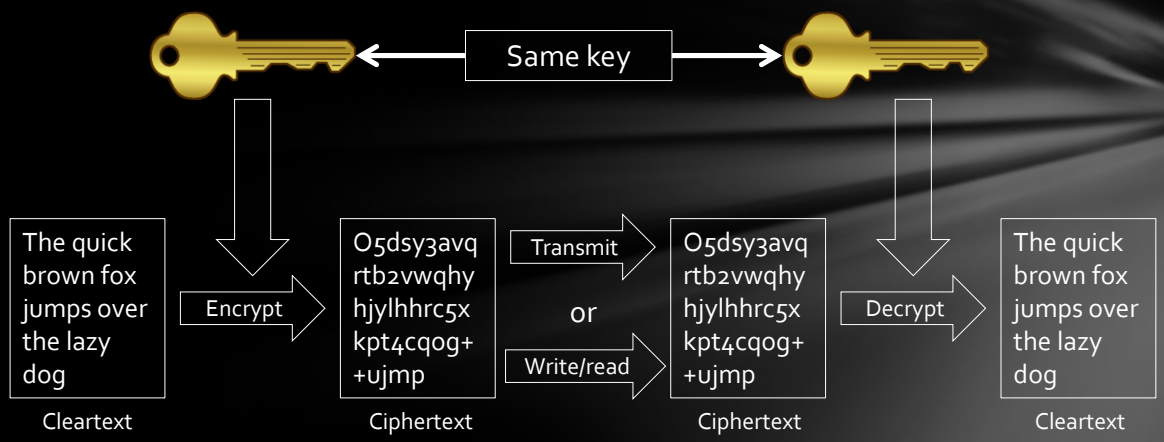
- With links for additional reference

AlphaCoders.com

Reference links for more information

100 MPH Review of Underlying Technologies

Secret (Symmetric) Key Encryption



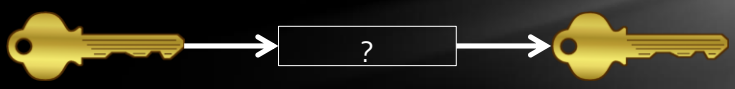
http://bit.ly/2ilor54

Secret Key Encryption

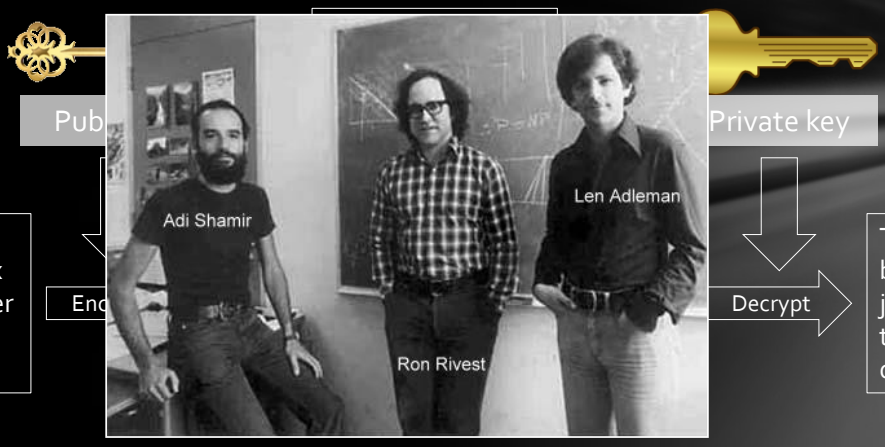
What's the big problem?

Key Management

- How do we get that secret key from one end to the other?
- How do we keep track of thousands of individual secret keys?
  - Chase Bank has 4 million customers

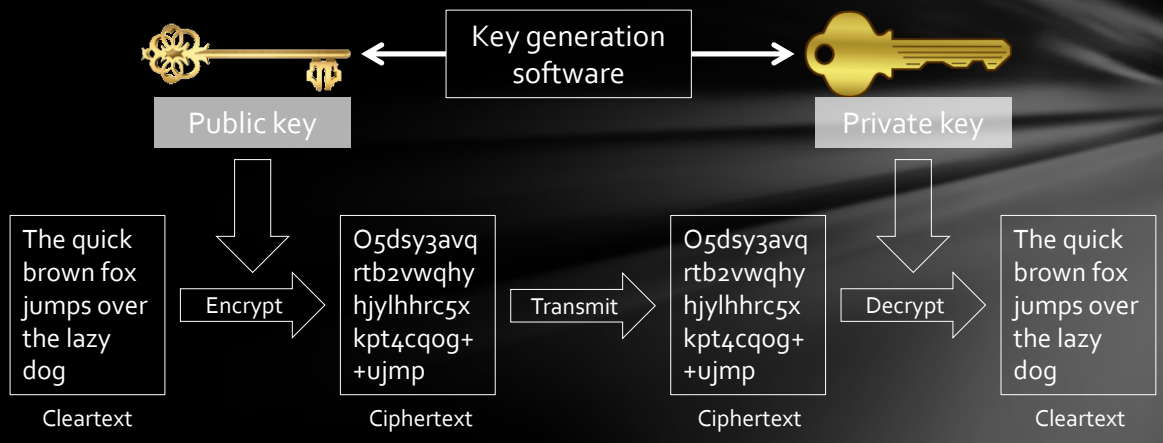


Public Key (Asymmetric) Encryption



http://bit.ly/2i05WvQ

Public Key (Asymmetric) Encryption



http://bit.ly/2io5WvQ

My Public Key

```
Public-Key: (2048 bit)
Modulus:
00:ad:3d:3a:cf:fd:39:8f:b0:d9:6d:8e:27:ad:37:
c7:74:a2:b3:7a:05:b0:de:f9:06:96:f7:c6:a1:16:
d5:2b:39:28:30:d2:63:3c:96:f5:3e:d1:9f:9b:9a:
1f:3e:29:71:be:7d:6b:c3:a3:90:de:ce:41:b0:e8:
5d:fe:ce:05:0d:d5:55:7f:fa:58:df:3b:5b:25:98:
8e:cb:c2:d1:6e:0d:be:44:88:87:9f:b1:a0:cf:de:
ae:7d:e3:fd:d1:81:64:2b:48:f1:7a:83:d7:e9:66:
9f:32:3a:9a:26:d5:41:50:3e:8a:a4:9c:18:9a:c1:
21:ea:9b:b5:23:b1:57:27:55:e0:85:a0:d6:0e:c4:
3b:ea:8e:03:b7:4e:28:e0:c8:57:de:db:fe:a4:dc:
32:11:09:aa:d8:6d:04:e0:f6:d5:e2:08:c4:87:30:
29:3a:bd:0f:2b:45:7d:b8:6e:8c:71:22:ff:8c:3c:
68:7d:64:87:f7:87:a5:66:2c:d2:71:e2:97:84:48:
26:82:58:e4:0b:d6:59:e3:57:0a:07:24:77:e3:39:
3a:07:04:f6:ac:23:e1:33:28:ba:f3:5b:7c:df:91:
27:a4:79:a1:e5:6c:e9:c7:23:74:81:a7:cc:7f:75:
c4:9e:d4:7e:27:af:23:9f:32:87:2e:f1:87:e7:38:
0f:31
Exponent: 65537 (0x10001)
```

But absolutely imperative to keep that private key private!

Public Key Encryption

What are the big problems?

Keys are HUGE! As you saw on the previous slide!

Need a program to generate a pair of keys

Key management (are you noticing a theme?)

Unidirectional

Very slow

Don't say Mills said Public Key was no good – we will see how certificates solve all of these problems in a moment

Digital Hash

Function that takes a possibly very long "message" and returns a relatively short fixed-length binary value

Easy to calculate

Deterministic: same message yields same hash

Very highly unlikely two slightly different messages have same hash

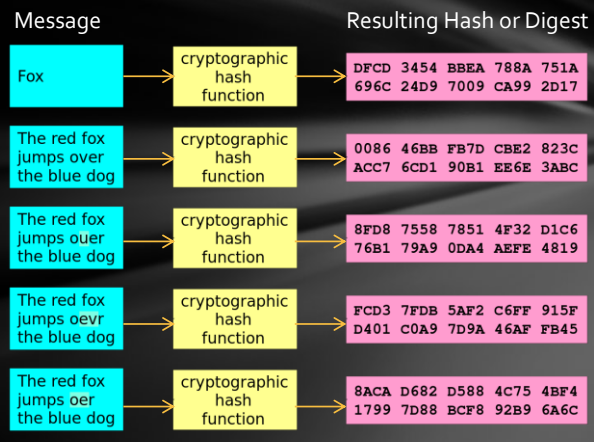
Almost impossible to construct a message with a predictable hash

Same hash = same message

Also called message digest

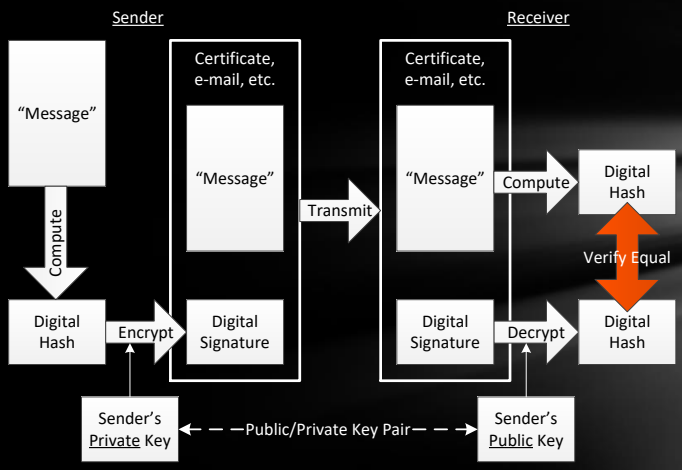
Examples: SHA-2 (MD5, SHA-1)

http://bit.ly/2mNgIor



Source: Wikipedia
By User:Jorge Stolfi based on Image:Hash_function.svg by Helix84 - Original work for Wikipedia, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5290240

Digital Signatures



Public key in reverse: encrypt hash with private key, decrypt with public key

Verifies message not tampered with: any change to message would change the hash

Authenticates: only owner of private key could have signed (encrypted)

Also non-repudiation

http://bit.ly/2rTHa54

Digital Signatures and Trust

You receive a "message" (could be anything) with an attached digital hash encrypted with some private key. You also compute your own digital hash for the message.

If you can decrypt the attached digital hash with my public key (well known) and it is the same as the digital hash you computed then the message is from me and is unaltered. If you trust me then you can trust the message.

By extension, if the message contains a public key, then you can trust any message signed with that public/private key pair.

This – the "chain of trust" – is the essence of digital signatures, certificates and trust.

Certificate Authority

Company or group within a company that issues certificates

Uses self-created "root" certificate to sign them

May be well-known CA

- Comodo (40%+ market share)
- IdenTrust (owned by 8 large banks)
- Symantec (acquired Verisign*; CA business sold to DigiCert)
- GoDaddy
- DigiCert
- Entrust*
- Let's Encrypt – Free! Highly automated. Transparent (public log). Ninety days only!

Or department or individual

Well-known CA required for public-use SSL/TLS

*CA Certificates that formerly shipped with RACF. Also Thawte, acquired by Verisign (acquired by Symantec, acquired by DigiCert).

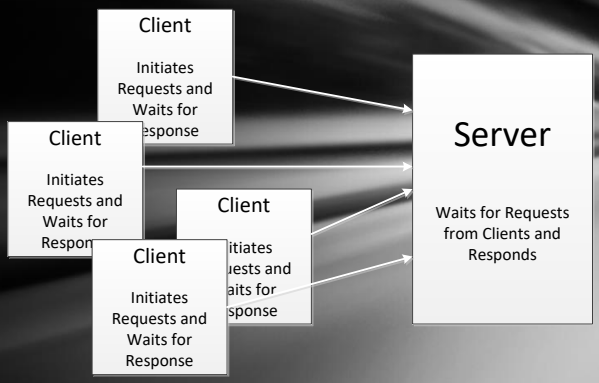
http://bit.ly/20KougM

Client and Server

Nothing to do with color or size of boxes

Often software, not hardware

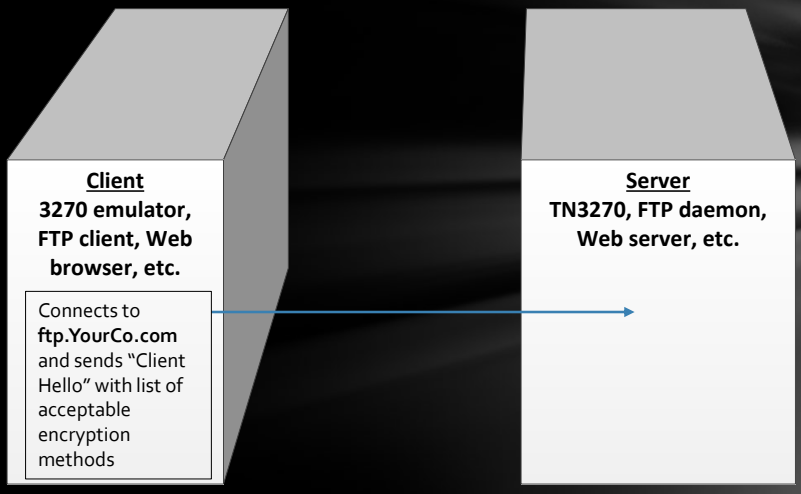
The predominant architecture for complex applications (Web browser, FTP, e-mail, 3270 emulation)



http://bit.ly/2DjfNWk

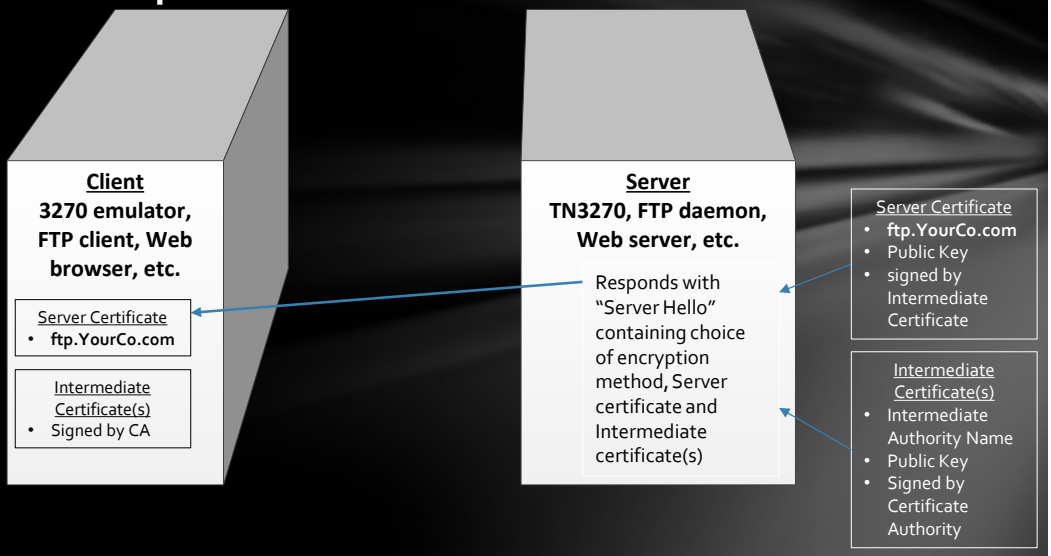
TLS certificate protocol flow

Client initiates connection to server



https://ibm.co/2rfON5g

Server responds with certificates



What's in a Certificate?

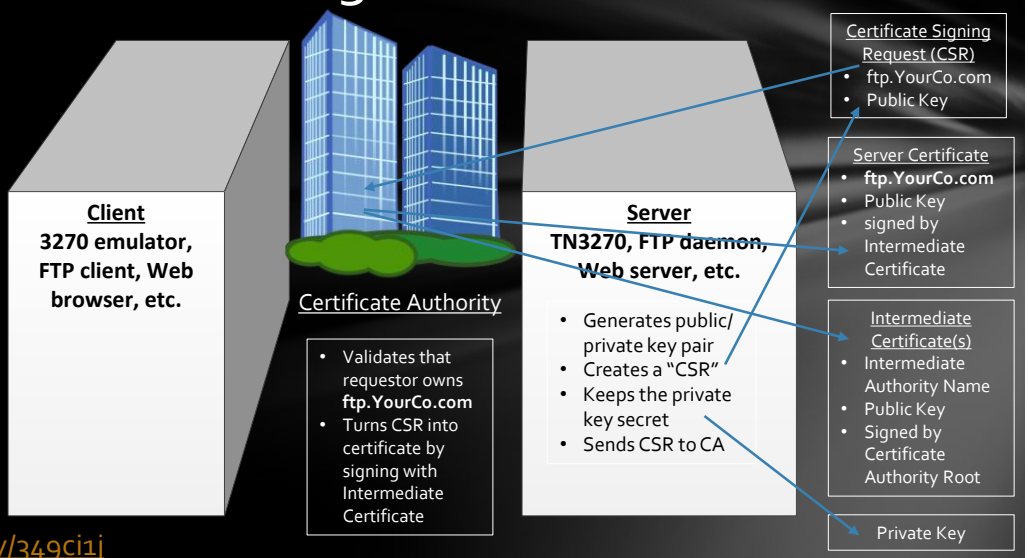
Subject Name	The URL of the server for which it was issued
Issuer	The name of the certificate that signed this one
Serial Number	MUST be unique within CA
Effective Dates	Start and end date and time
Public Key	Half of this certificate's key pair
Digital Signature	Attests to the authenticity of this certificate

What's Never in a Certificate?

Private Key	Sometimes *packaged with* the certificate but never *part of* the certificate

https://bit.ly/3m8orXH

How did the server get that certificate?



https://bit.ly/349ci1j

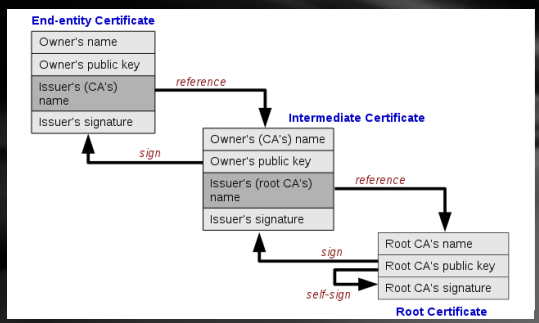
Why Intermediate Certificates?

The compromise of a CA root key would render root and all certificates issued by CA untrustworthy – a disaster!

Certificate Authorities store their root keys off-line to help prevent compromise

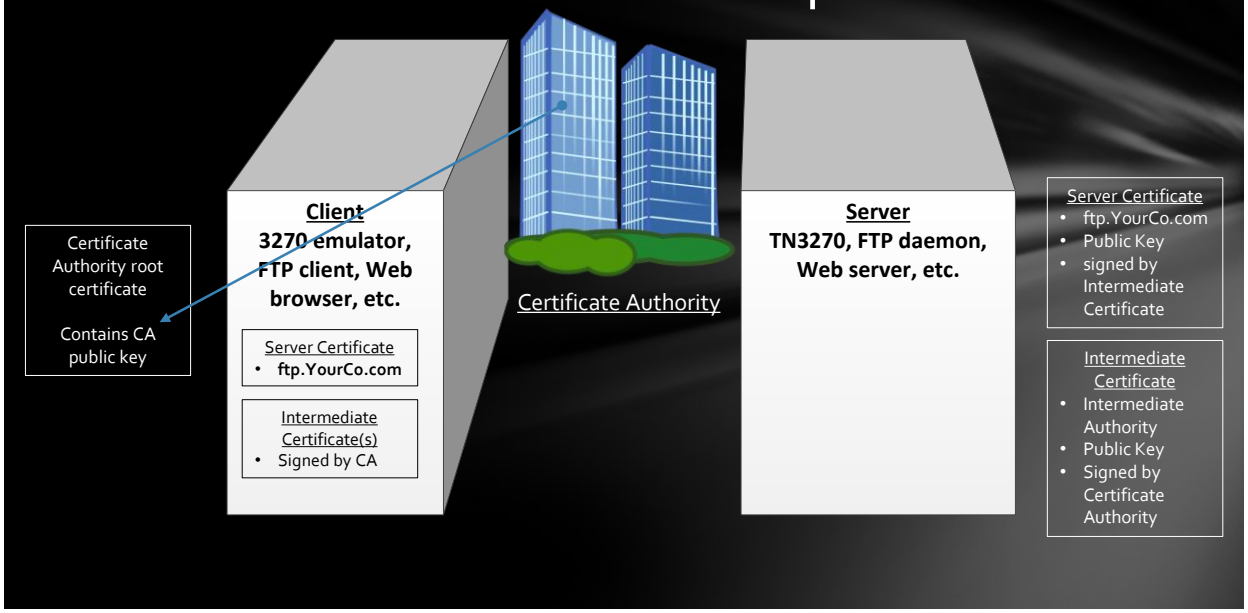
They use medium-term intermediate certificates – signed by their root certificate – to issue end-user certificates

Intermediate certificates are signed by the root certificate: "Chain of trust"

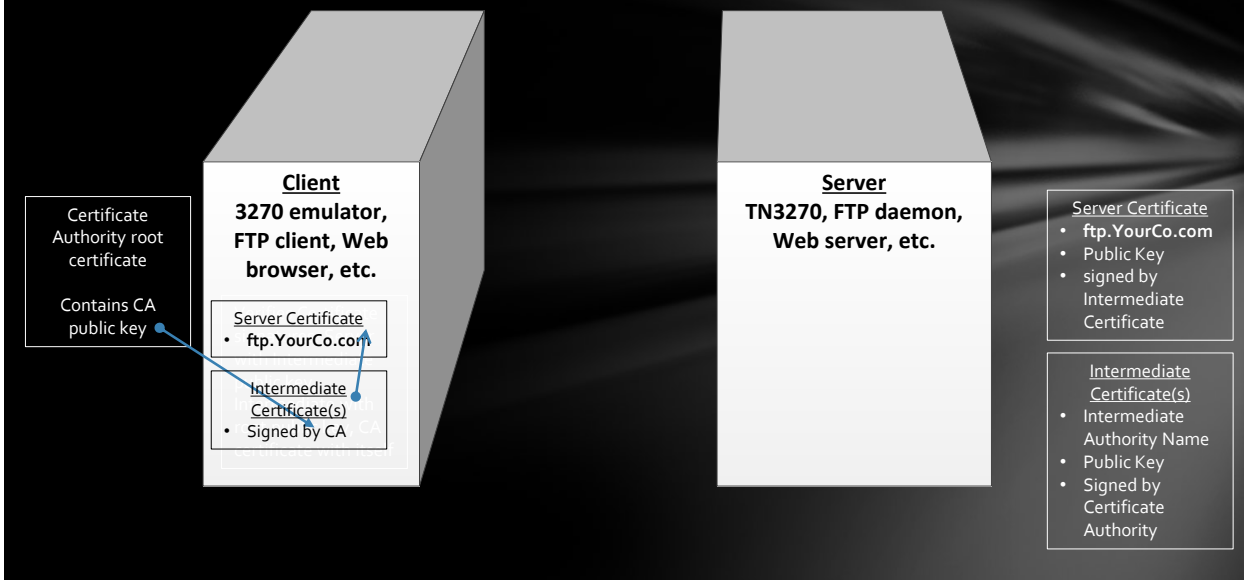


Source: Wikipedia
By Yanpas - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=46369922

Client has CA root certificate pre-installed



Client validates certificate signatures



Client validates certificate subject name

Client
3270 emulator, FTP client, Web browser, etc.

Verifies subject name matches intended URL

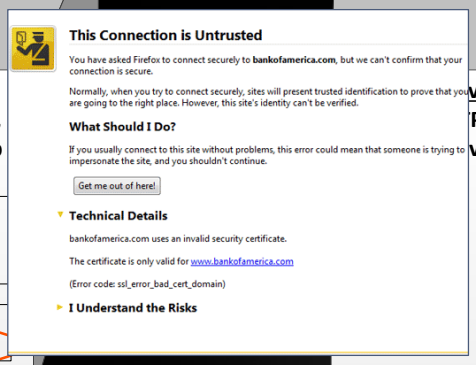
Server Certificate
- **ftp.YourCo.com**

Server
TN3270, FTP daemon, Web server, etc.

Server Certificate
- **ftp.YourCo.com**
- Public Key
- signed by Intermediate Authority

Intermediate Certificate(s)
- Intermediate Authority Name
- Public Key
- Signed by Certificate Authority

Client validates certificate subject name



Client
3270 emulator, FTP client, Web browser, etc.

Verifies subject name matches intended URL

Server Certificate
- **ftp.YourCo.com**

ver
P daemon,
ver, etc.

Server Certificate
- **ftp.YourCo.com**
- Public Key
- signed by Intermediate Authority

Intermediate Certificate(s)
- Intermediate Authority Name
- Public Key
- Signed by Certificate Authority

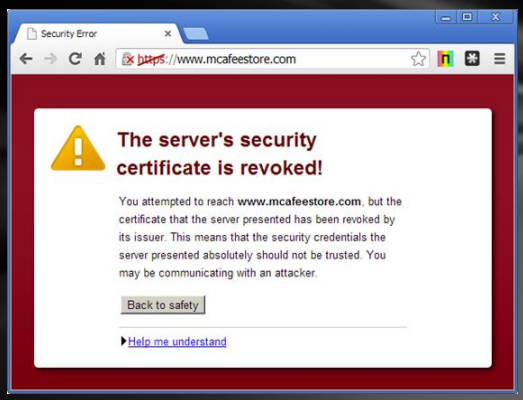
Certificate Revocation

Why would a certificate be revoked?

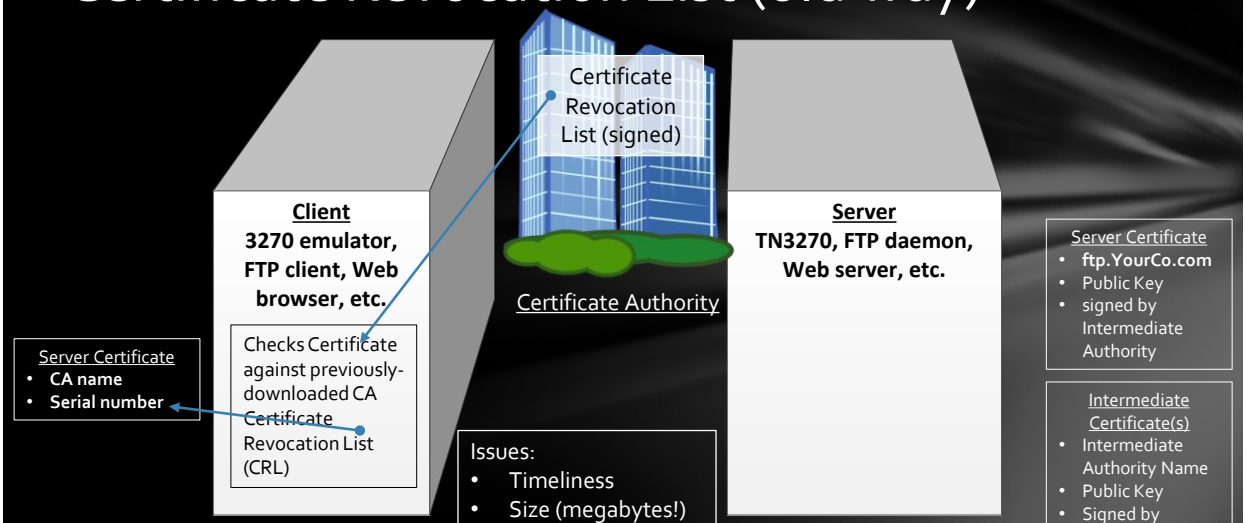
- Issued in error
- Key compromised
- CA root key compromised (disaster!)

Clients should check for server certificate revocation

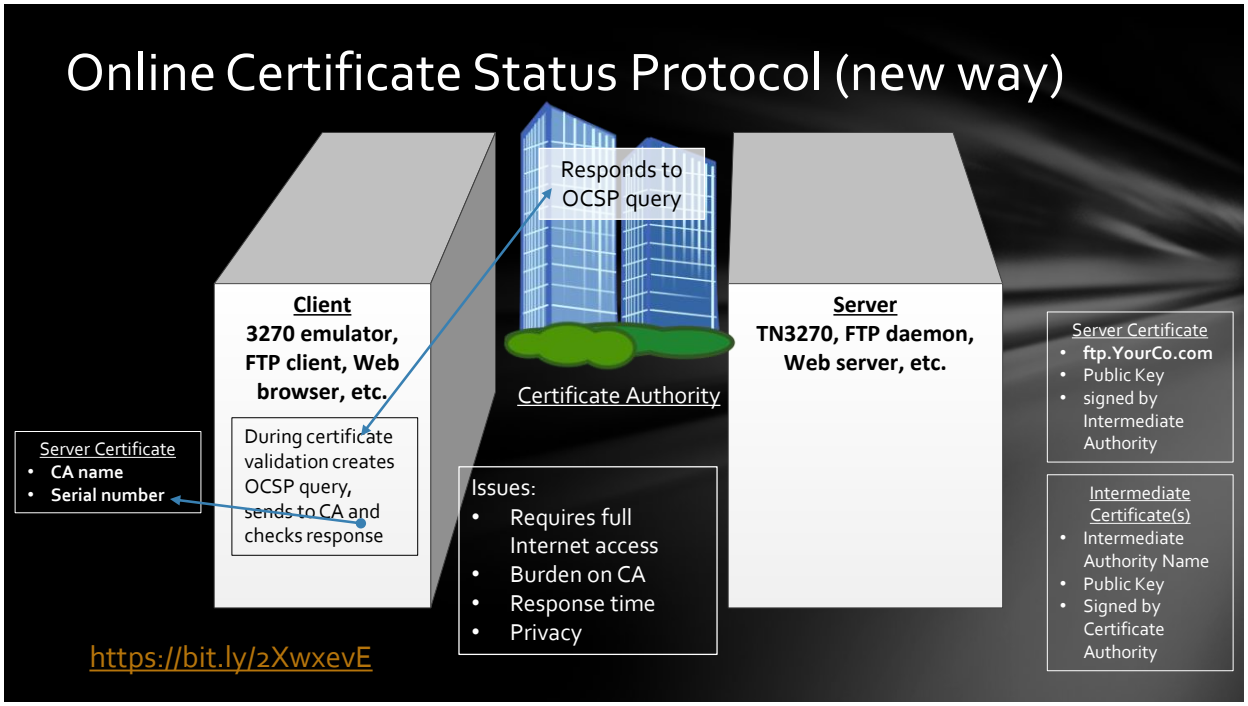
- Some clients do not, and some users ignore the error – bad idea!

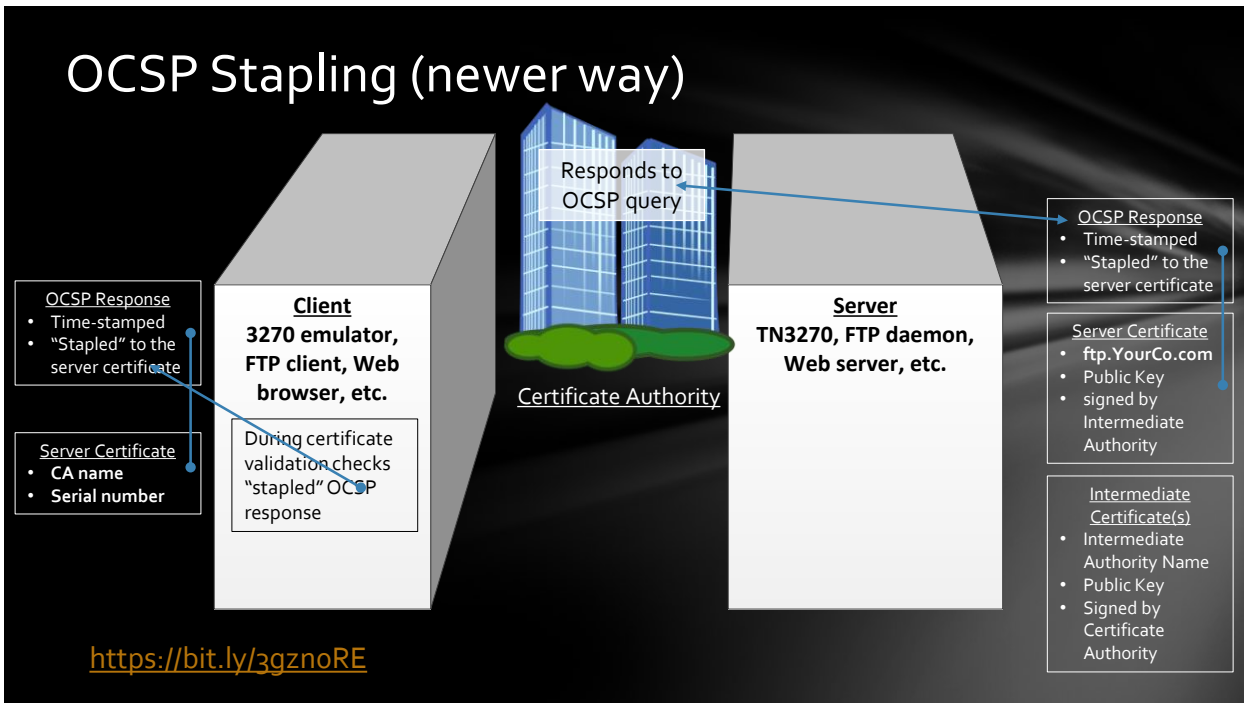


Certificate Revocation List (old way)

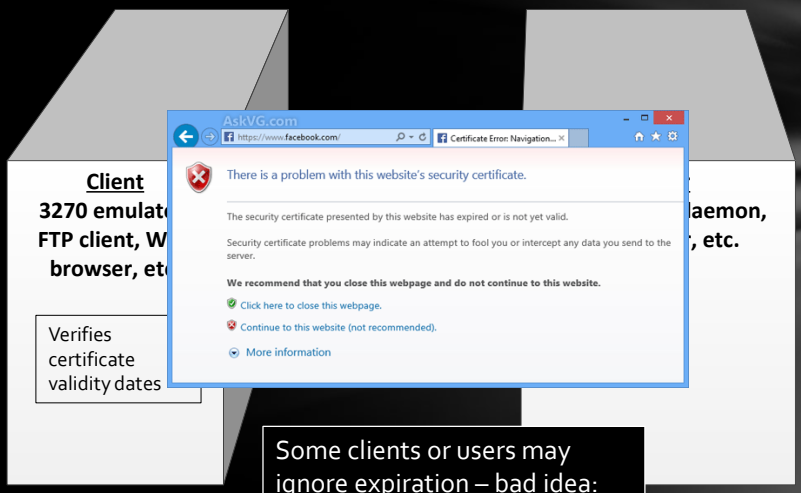


https://bit.ly/31h30MM

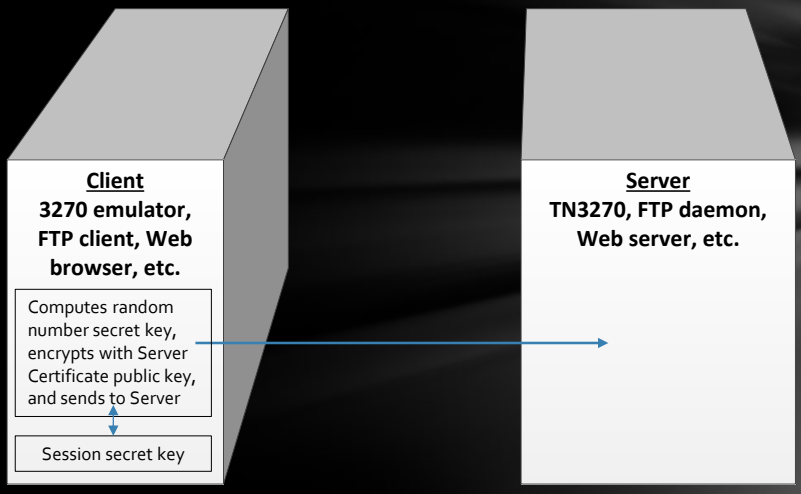

Online Certificate Status Protocol (new way)
Responds to OCSP query
Client
3270 emulator, FTP client, Web browser, etc.
Certificate Authority
Server
TN3270, FTP daemon, Web server, etc.
Server Certificate
• ftp.YourCo.com
• Public Key
• signed by Intermediate Authority
Server Certificate
• CA name
• Serial number
During certificate validation creates OCSP query, sends to CA and checks response
Issues:
• Requires full Internet access
• Burden on CA
• Response time
• Privacy
Intermediate Certificate(s)
• Intermediate Authority Name
• Public Key
• Signed by Certificate Authority
https://bit.ly/2XwxevE


OCSP Stapling (newer way)
Responds to OCSP query
Client
3270 emulator, FTP client, Web browser, etc.
Certificate Authority
Server
TN3270, FTP daemon, Web server, etc.
OCSP Response
• Time-stamped
• "Stapled" to the server certificate
Server Certificate
• ftp.YourCo.com
• Public Key
• signed by Intermediate Authority
OCSP Response
• Time-stamped
• "Stapled" to the server certificate
Server Certificate
• CA name
• Serial number
During certificate validation checks "stapled" OCSP response
Intermediate Certificate(s)
• Intermediate Authority Name
• Public Key
• Signed by Certificate Authority
https://bit.ly/3gznoRE

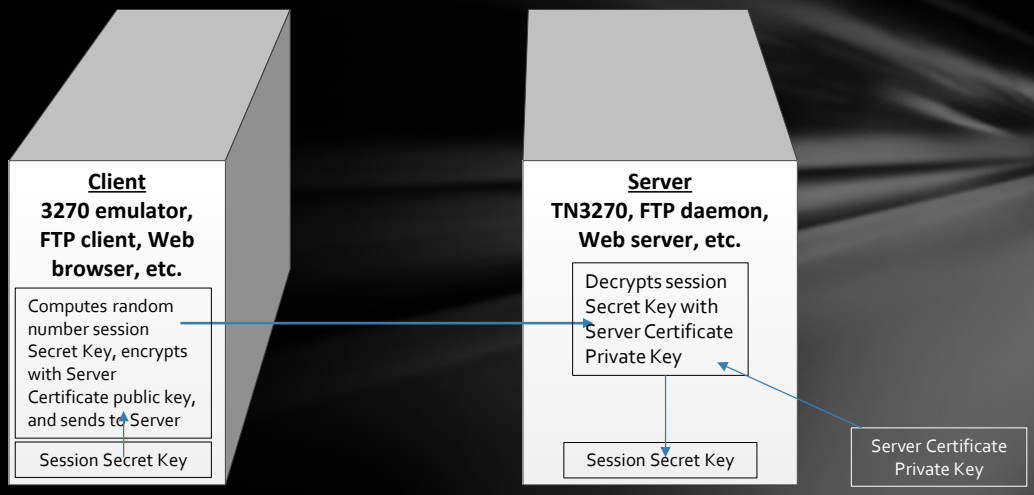
Client checks certificate validity dates



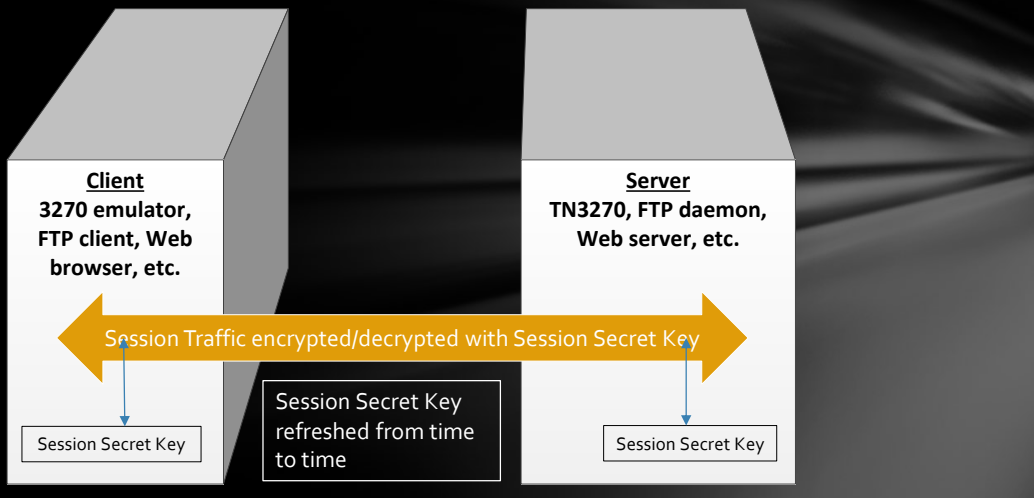
Client creates and sends session secret key



Server decrypts session secret key

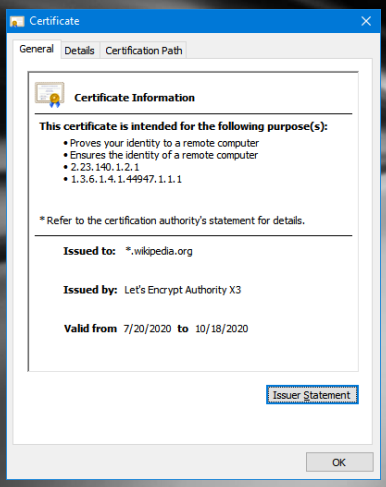


Data traffic at last!

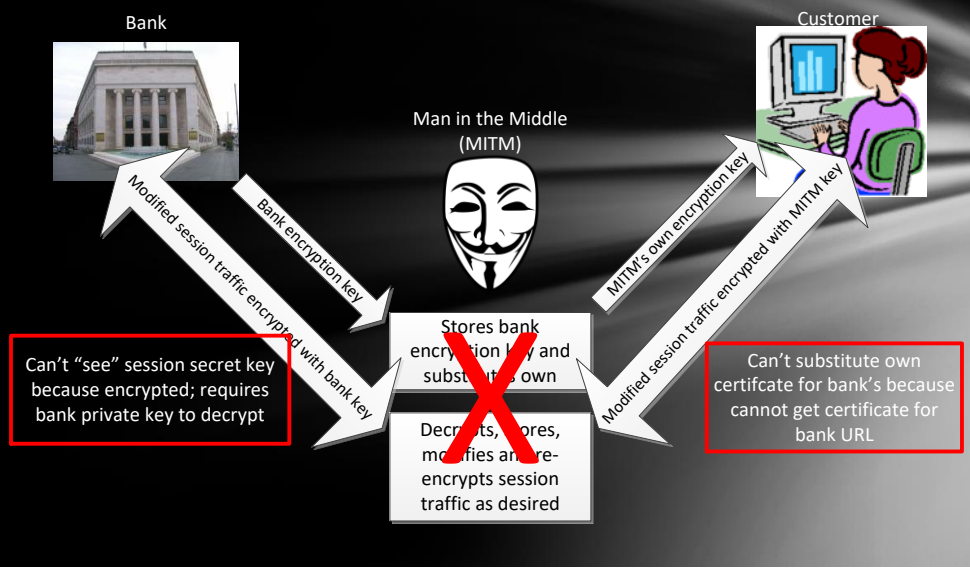


Certificates Solve the Crucial Problems

- Authentication
- Encryption
- Secure key delivery
- Automation of key delivery
- Bi-directionality
- Speed
- Man in the Middle attack



How certificates prevent man in the middle



Certificate Issues

Complexity

Certificate management

- Especially expiration

Key management

- Keeping private key private
- But not losing them!

CA Root Certificates and Trust

Certificate Authority Issues

- Sloppiness, fraud?
- Corrupt government pressures CA to facilitate Man-in-the-Middle?
- Dutch CA DigiNotar hacked; fraudulent Google.com certificate used for Man-in-the-Middle interception of Iranian citizens https://bit.ly/3hN1b2n
- Name validation by CA
  - Requirement for CA to validate URLs at odds with modern certificate volumes
  - In March of 2017, Google announced Chrome would stop honoring Symantec certificates for among other things sloppiness in validating certificate names https://bit.ly/2QGs1xd
    - Death penalty! Symantec sold CA business to DigiCert
- Any CA can issue a certificate for any site!

Questions?

100 MPH Overview of Some Advanced Features

Self-signed certificates

Misunderstood concept

Self-signing is not inherently bad – all CA root certificates are self-signed

Means the certificate signs itself, not that the company that issued the certificate is its own CA

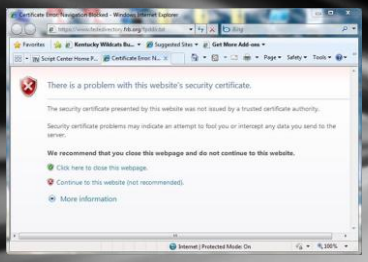
Generally frowned upon for end-point certificates

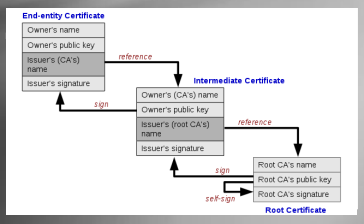
Provide encryption

Provide authentication only if pre-installed on client

Nothing wrong with your company being its own CA

- Saves money, time and trouble
- Works only for internal clients – external users do not have CA root certificate
- Possibly more secure to control it all yourself





Source: Wikipedia
By Yanpas - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=46369922

Alternative Names

Certificates support multiple "subject alternative names" (SANs) in addition to the main "common name"

Thus one certificate could be valid for YourCo.com, MyCo.com and HerCo.com

Using an Alternative Name for the server URL is now preferred to Common Name (RFC 2818)

Sometimes called a Multi-Domain or SAN Certificate

http://bit.ly/2B8AL4Z

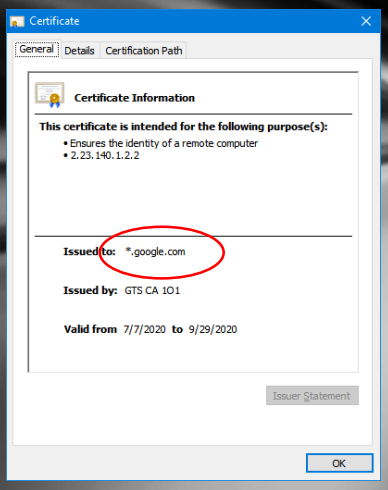
Subject Name Wildcards

Certificates support wildcard subject names (Common or Alternative)

Asterisk may be last or only character of leftmost sub-domain name: `*.YourCo.Com` or `w*.YourCo.Com`

- Or last dotted address octet: `192.168.17.*` (infrequent)

One certificate for www.YourCo.com, ftp.YourCo.com and mail.YourCo.com



http://bit.ly/2Djbw6g

Client Certificates

Server certificate authenticates server identity and provides for encryption

Client certificate authenticates client identity only

- Does not provide for or configure encryption
- Must be CA-signed or else pre-installed on server

An alternative to passwords

Good choice if relatively small number of clients, over which you have control

- Good for branch offices, not for customers

Server makes protocol request for certificate from client, so configuration is a server option

- FTP Example (server-side):
  `SECURE_LOGIN VERIFY_USER`

Validation protocol similar to server certificate

Code signing with certificates

Verifies that software is authentic

Does not prove that code is good, merely authentic!

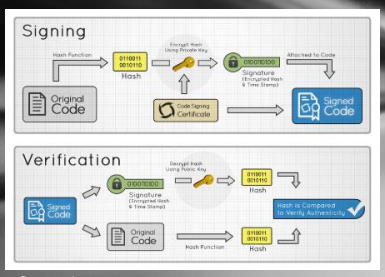
Verifies that software has not been altered/tampered with

Requires special code-signing software

May be CA-signed or software-vendor signed

Time-stamping

- Allows for fact that certificate may expire after software is published but before it is installed



Source itcs.com

Constraints and Key Usage

Basic constraint
- CA key or not

Key usage
- Signatures
- Encipherment
- Etc.

Extended key usage
- Server
- Client
- Code signing
- Email
- Etc.

http://bit.ly/2B7lZvy

```
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
X509v3 Extended Key Usage:
        TLS Web Server Authentication,
        TLS Web Client Authentication
```

Summary

Why certificates?

100 MPH review of underlying technologies
- With links for additional reference

Details of the certificate protocol flow

100 MPH introduction to some advanced features
- With links for additional reference

More questions?
charlesm@mcn.org


Thank you