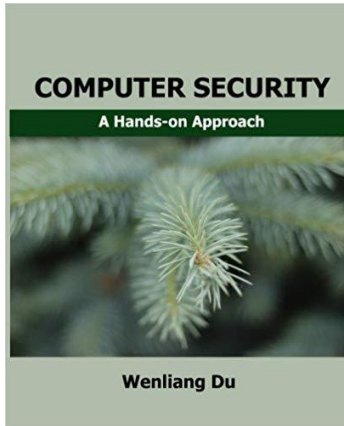


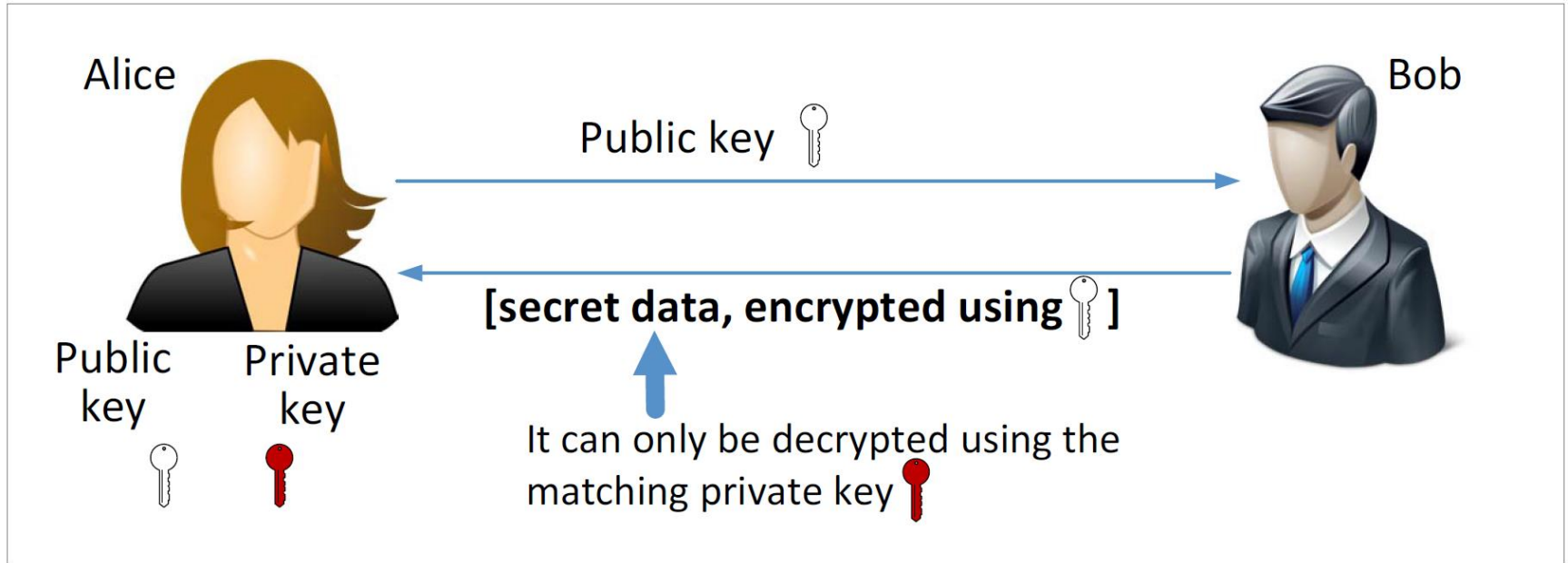
Public Key Infrastructure



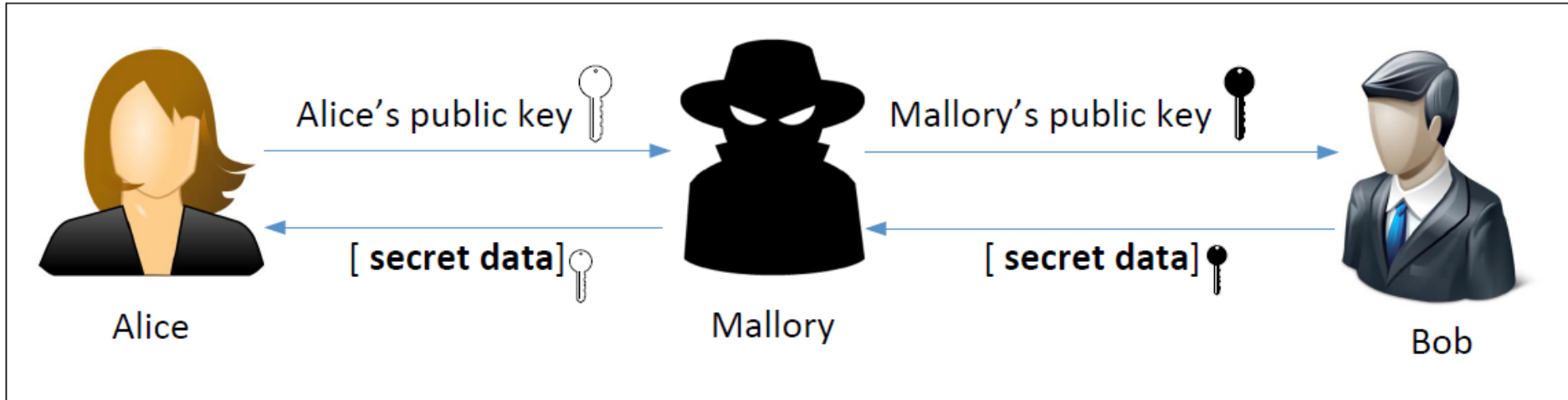
Based on slides accompanying the book
Computer Security: A hands-on approach
by Wenliang Du (2017)

Public Key Cryptography

- The **public** key and the **private** key in the same key pair are *inverse*.



Man-in-the-Middle (MITM) Attack against sharing of public keys



Q: Whose DNS might be polluted?

What Is the Fundamental Problem?

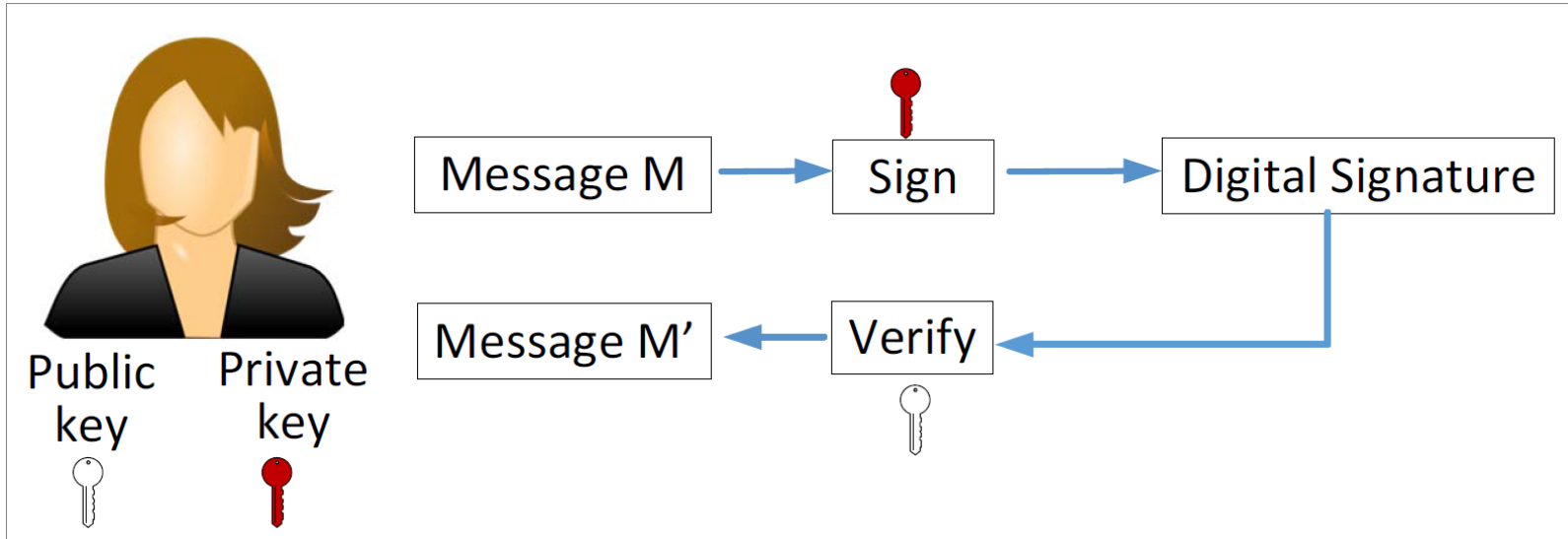
Fundamental Problem: Bob has no way to tell whether the public key he has received belongs to Alice or not.

Q: Which type of security service is needed here?

Solution:

- Find a trusted party to verify the identity (**of whom?**)
- Bind an identity to a public key in a certificate
- The certificate cannot be forged or tampered with (using digital signature)

Digital Signature



- If the signature is not tampered with, M' will be the same as M
- Only Alice can sign (she has the private key)
- Everybody can verify (public key is known publically)

Defeating MITM Attacks using Digital Signature

- Alice needs to go to a **trusted party** to get a certificate.
- After verifying Alice's identity, the trusted party issues a certificate with Alice's name and her public key.
- Alice sends the entire certificate to Bob.
- Bob verifies the certificate using the trusted party's public key.
- Bob now knows the **true owner** of a public key.

Q: What is the *foundation* of such a scheme?

Public Key Infrastructure

- **Certificate Authority (CA):** a **trusted party**, responsible for verifying the identity of a user/entity, and then bind the verified identity to a public key.
- **Digital Certificates:** A document certifying that the public key included inside does belong to the identity described in the document.
 - X.509 standard

Digital Certificate

- Let's get paypal's certificates

```
$ openssl s_client -showcerts -connect www.paypal.com:443 </dev/null
```

```
-----BEGIN CERTIFICATE-----  
MIIHWTCCBkGgAwIBAgIQLNQVEFQ30N5KOSAFavbCfzANBgkqhkiG9w0BAQsFADB3  
MQswCQYDVQQGEwJVUzEdMBsGA1UEChMUU3ltYW50ZWMgQ29ycG9yYXRpb24xHzAd  
... (omitted) ...  
GN/QMQ3a55rjwNQN3s2WWuHGPaE/jMG17iiL20/hUdIvLE9+wA+fWrey5//74x1  
NeQitYiySDIepHGnng==  
-----END CERTIFICATE-----
```

- Save the above data to paypal.pem, and use the following command decode it (see next slide)

```
$ openssl x509 -in paypal.pem -text -noout
```


Example of X.509 Certificate (1st Part)

```
Certificate:
Data:
  Serial Number:
    2c:d1:95:10:54:37:d0:de:4a:39:20:05:6a:f6:c2:7f
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network,
    CN=Symantec Class 3 EV SSL CA - G3
  Validity
    Not Before: Feb  2 00:00:00 2016 GMT
    Not After  : Oct 30 23:59:59 2017 GMT
  Subject: 1.3.6.1.4.1.311.60.2.1.3=US/
    1.3.6.1.4.1.311.60.2.1.2=Delaware/
    businessCategory=Private Organization/
    serialNumber=3014267, C=US/
    postalCode=95131-2021, ST=California,
    L=San Jose/street=2211 N 1st St,
    O=PayPal, Inc., OU=CDN Support, CN=www.paypal.com
```

The CA's identity
(Symantec)

The owner of
the certificate
(paypal)

Example of X.509 Certificate (2nd Part)

Public key

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

Modulus:

00:da:43:c8:b3:a6:33:5d:83:c0:63:14:47:fd:6b:22:bd:

bf:4e:a7:43:11:55:eb:20:8b:e4:61:13:ee:de:fe:c6:e2:

... (omitted) ...

7a:15:00:c5:01:69:b5:10:16:a5:85:f8:fd:07:84:9a:c9:

Exponent: 65537 (0x10001)

CA's signature

Signature Algorithm: sha256WithRSAEncryption

4b:a9:64:20:cc:77:0b:30:ab:69:50:d3:7f:de:dc:7c:e2:fb:93:84:fd:

78:a7:06:e8:14:03:99:c0:e4:4a:ef:c3:5d:15:2a:81:a1:b9:ff:dc:3a:

... (omitted) ...

fb:00:3e:7d:6a:de:cb:9f:ff:ef:8c:65:35:e4:22:b5:88:b2:48:32:1e:

An example X.509 intermediate CA certificate

Certificates

Intended purpose: <All>

Personal Other People Intermediate Certification Authorities Trusted Root Certification

Issued To	Issued By	Expirati...	Friendly Name
Microsoft Windo...	Microsoft Root Aut...	12/31/2...	<None>
RapidSSL RSA C...	DigiCert Global Ro...	11/6/20...	<None>
Root Agency	Root Agency	12/31/2...	<None>
Symantec Class ...	VeriSign Class 3 P...	10/30/2...	<None>
Thawte RSA CA...	DigiCert Global Ro...	11/6/20...	<None>
Trustwave Orga...	SecureTrust CA	9/29/20...	<None>
TWCA Global Ro...	TWCA Root Certifi...	10/28/2...	<None>
USERTrust RSA ...	AddTrust External ...	5/30/20...	<None>
www.verisign.c...	Class 3 Public Pri...	10/24/2...	<None>

Import... Export... Remove Advanced

Certificate intended purposes
<All> View

Close

Certificate

General Details Certification Path

Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- Protects e-mail messages
- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: Symantec Class 3 Secure Server CA - G4

Issued by: VeriSign Class 3 Public Primary Certification Authority - G5

Valid from 10/30/2013 **to** 10/30/2023

Issuer Statement

OK

Certificate

General Details Certification Path

Show <All>

eld	Value
Version	V3
Serial number	51 3f b9 74 38 70 b7 34 40 41 8d 30 9...
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	VeriSign Class 3 Public Primary Certifi...
Valid from	Wednesday, October 30, 2013 6:00:0...
Valid to	Monday, October 30, 2023 5:59:59 PM

V3

Edit Properties... Copy to File...

OK

Certificate

General Details Certification Path

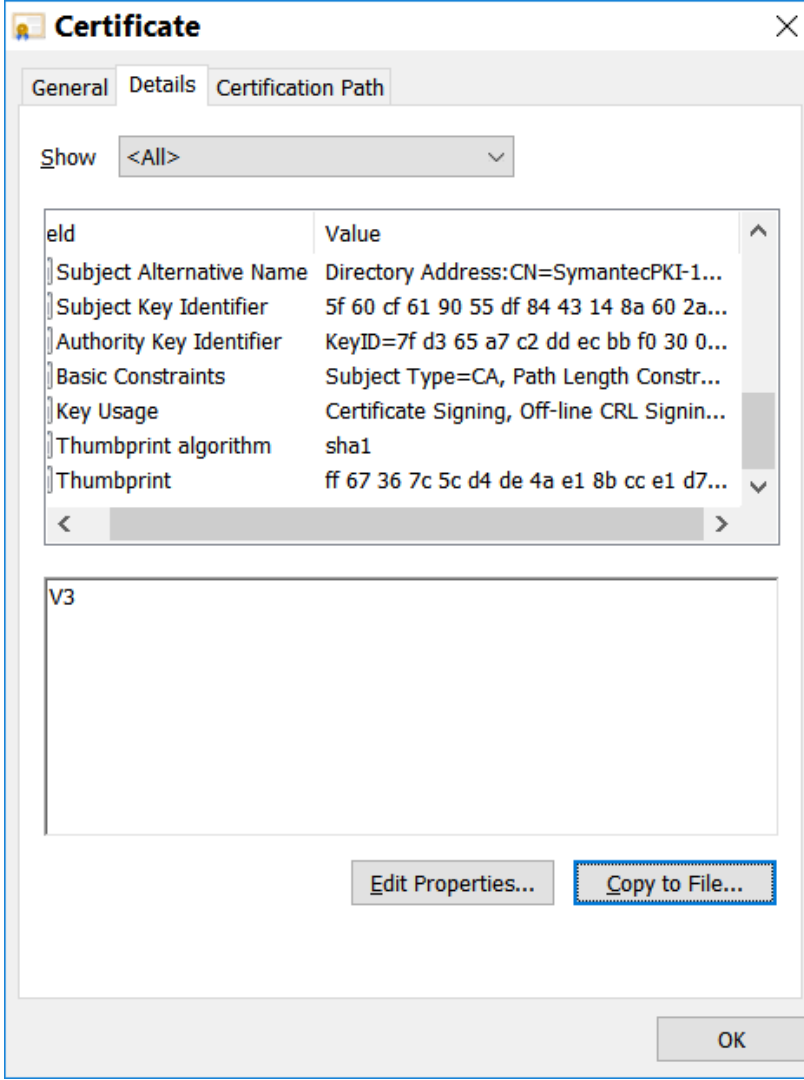
Show <All>

eld	Value
Valid to	Monday, October 30, 2023 5:59:59 PM
Subject	Symantec Class 3 Secure Server CA - ...
Public key	RSA (2048 Bits)
Public key parameters	05 00
CRL Distribution Points	[1]CRL Distribution Point: Distribution ...
Authority Information A...	[1]Authority Info Access: Access Meth...
Certificate Policies	[1]Certificate Policy:Policy Identifier=2...

V3

Edit Properties... Copy to File...

OK



The Core Functionalities of CA

- **Verify the subject**
 - Ensure that the person applying for the certificate either owns or represents the identity in the subject field.
- **Signing digital certificates**
 - CA generates a digital signature for the certificate using its private key.
 - Once the signature is applied, the certificate cannot be modified.
 - Signatures can be verified by anyone with the CA's public key. (**Q:** So, what is the *prerequisite* of certificate verification?)

Being a Certificate Authority

- Let's go through the process
 - How a CA issues certificates
 - How to get a certificate from a CA
 - How to set up a web server using a certificate

CA Setup

- Our CA will be called ModelCA
- We need to set up the following for ModelCA:
 - Generate public/private key pair
 - Create a X.509 certificate (who is going to sign it?)
 - We assume ModelCA is a **root CA**, so it is going to sign the certificate itself, i.e., ModelCA's certificate is **self-signed**.
- The following command generates a self-signed X.509 certificate

```
$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650  
-keyout modelCA_key.pem -out modelCA_cert.pem
```


Discussion Question

- **Question:** If the ModelCA's certificate is self-signed, how do we verify it?
- **Answer:** There is no way to verify it. We just make sure that the certificate is obtained in a trusted way
 - Come with the operating system (if we trust OS, we trust the cert.)
 - Come with the software (if we trust the software, we trust the cert.)
 - Manually added (if we trust our own decision, we trust the cert.)
 - **Sent to us by somebody whom we don't trust** (don't trust the cert.)

Get a Certificate from CA: Step 1

- Step 1: Generate a public/private key pair

```
$ openssl genrsa -aes128 -out bank_key.pem 2048
```

RSA key size



Encrypt the output file using AES (128-bit)



Contains both private and public keys

Get a Certificate from CA: Step 2

- Step 2: Generate a **certificate signing request (CSR)**; identity information needs to be provided

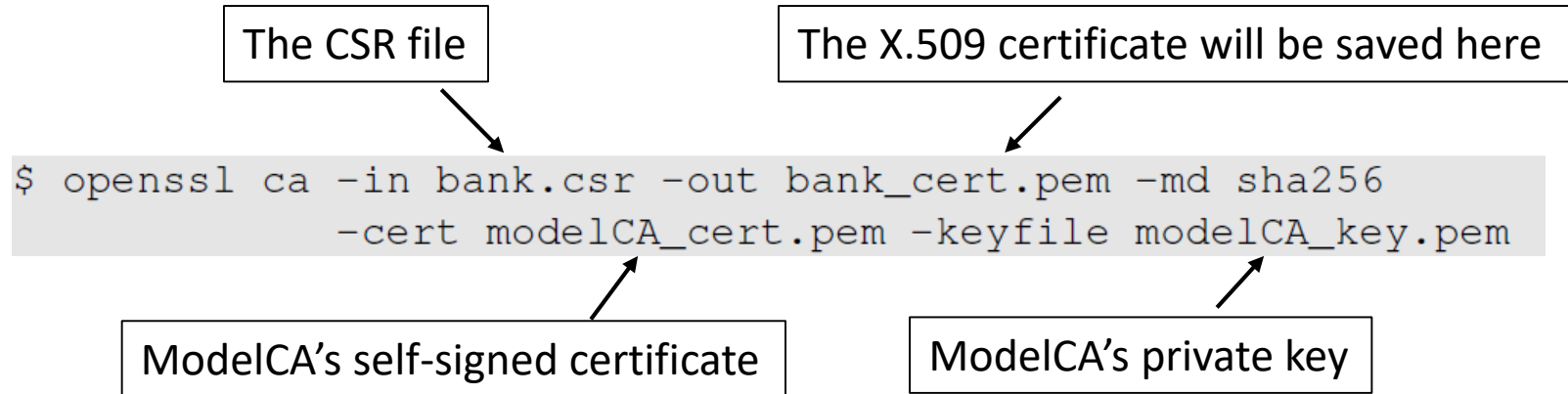
```
$ openssl req -new -key bank_key.pem -out bank.csr -sha256
```

```
$ openssl req -in bank.csr -text -noout
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: C=US, ST=New York, L=Syracuse, O=Example Inc,
            CN=example.com/emailAddress=email@example.com
```

CA will verify this subject information

CA: Issuing X.509 Certificate

- We (the bank) need to send the CSR file to ModelCA.
- ModelCA will verify that we are the actual owner of (or can represent) the identity specified in the CSR file.
- If the verification is successful, ModelCA issues a certificate.



Deploying Public Key Certificate in Web Server

- We will first use openssl's built-in server to set up an **HTTPS** web server

```
$ cp bank.key bank.pem
$ cat bank.crt >> bank.pem
$ openssl s_server -cert bank.pem -accept 4433 -www
```

- Access the server using Firefox (<https://example.com:4433>), we get the following error message. Why?

```
example.com:4433 uses an invalid security certificate.
```

```
The certificate is not trusted because no issuer chain was provided.
The certificate is only valid for example.com
```

```
(Error code: sec_error_unknown_issuer)
```

Answer to the Question in the Previous Slide

- Firefox needs to use ModelCA's public key to verify the certificate
- Firefox does not have ModelCA's public key certificate
- We can manually add ModelCA's certificate to Firefox

`Goto Edit -> Preference -> Advanced -> View Certificates`

`Import ModelCA_cert.pem`

Apache Setup for HTTPS

- We add the following VirtualHost entry to the Apache configuration file:

```
<VirtualHost *:443>
  ServerName example.com
  DocumentRoot /var/www/Example
  DirectoryIndex index.html

  SSLEngine On
  SSLCertificateFile      /etc/apache2/ssl/bank_cert.pem ①
  SSLCertificateKeyFile  /etc/apache2/ssl/bank_key.pem ②
</VirtualHost>
```

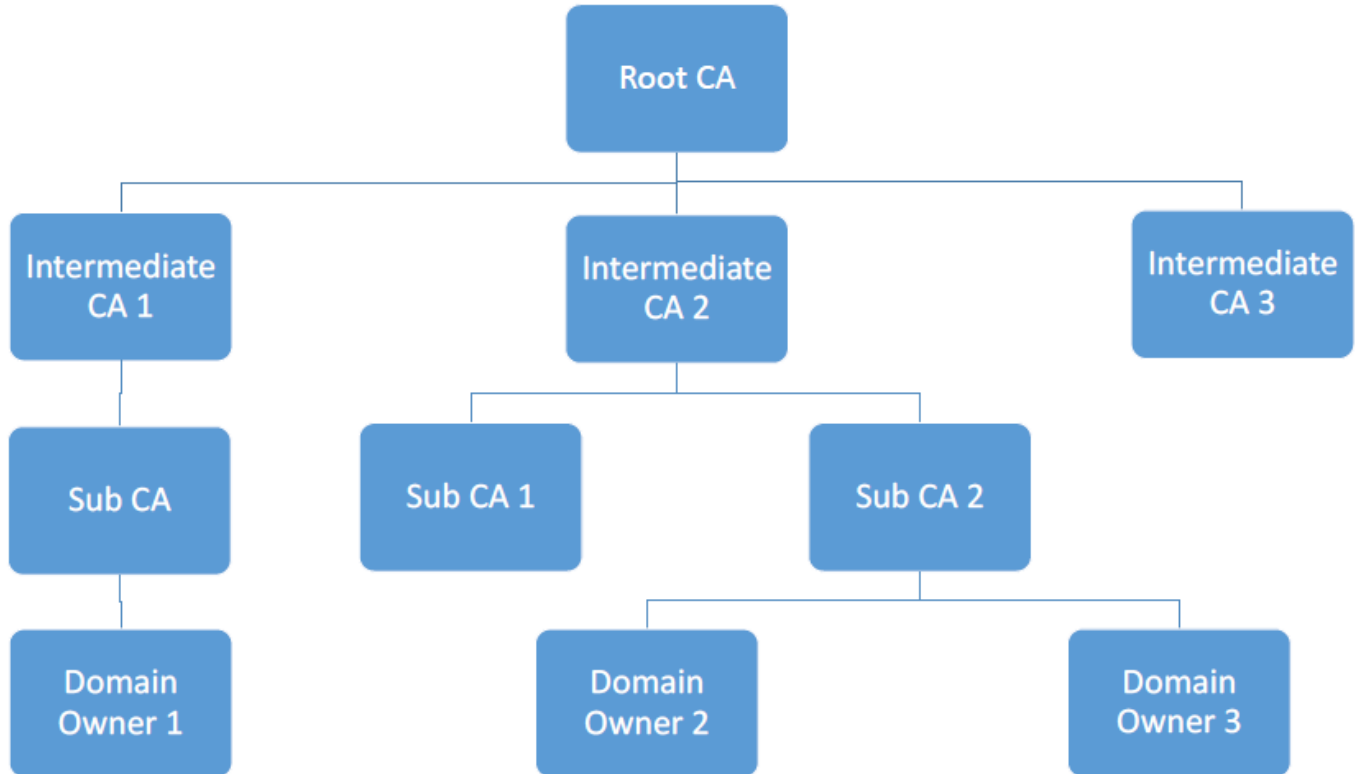
The server's
certificate

The server's
private key

Note: Apache configuration file is located at
`/etc/apache2/sites-available/default`

Root and Intermediate Certificate Authorities

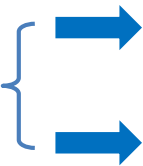
There are many CAs in the real world, and they are organized in a hierarchical structure.



Root CAs and Self-Signed Certificate

- A root CA's public key is also stored in an X.509 certificate. It is self-signed.
- **Self-signed:** the entries for the issuer and the subject are identical.

Same



```
Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network,  
        OU=(c) 2006 VeriSign, Inc. - For authorized use only,  
        CN=VeriSign Class 3 Public Primary Certification Authority - G5  
Subject: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network,  
        OU=(c) 2006 VeriSign, Inc. - For authorized use only,  
        CN=VeriSign Class 3 Public Primary Certification Authority - G5
```

Q: How can they be trusted?

- Public keys of root CAs are pre-installed in the OS, browsers and other software

Intermediate CAs and Chain of Trust

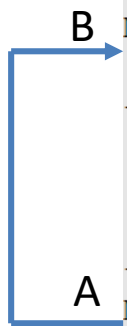
```
$ openssl s_client -showcerts -connect www.paypal.com:443

Certificate chain
 0 s: ... /CN=www.paypal.com
  i: ... /CN=Symantec Class 3 EV SSL CA - G3
-----BEGIN CERTIFICATE-----
MIIHWTCCBkGgAwIBAgIQLNQVEFQ30N5KOSAFavbCfzANBgkqhkiG9w0BAQsFADB3
...
-----END CERTIFICATE-----
 1 s: ... /CN=Symantec Class 3 EV SSL CA - G3
  i: ... /CN=VeriSign Class 3 Public Primary Certification
      Authority - G5
-----BEGIN CERTIFICATE-----
MIIFKzCCBBOgAwIBAgIQfuFKb2/v8tN/P61lTTratDANBgkqhkiG9w0BAQsFADCB
...
-----END CERTIFICATE-----
```

Paypal's certificate

Intermediate CA's certificate

A is used to verify B



Something else is needed to verify A (certificate from another intermediate CA or root CA)

Manually Verifying a Certificate Chain

- Paypal.pem: Save Paypal's certificate to a file
- Symantec-g3.pem: Save certificate from "Symantec Class 3 EV SSL CA – G3"
- VeriSign-G5.pem: Save the VeriSign-G5's certificate from the browser

Root CA's certificate



```
$ openssl verify -verbose -CAfile VeriSign-G5.pem  
-untrusted Symantec-G3.pem Paypal.pem  
Paypal.pem: OK
```

Chain of certificates




Creating Certificates for Intermediate CA

- When generating a certificate for an intermediate CA, we need to do something special:

```
$ openssl ca -in modelIntCA.csr -out modelIntCA_cert.pem -md sha256  
             -cert modelCA_cert.pem -keyfile modelCA_key.pem  
             -extensions v3_ca
```

- The extension field of the certificate will look as follows:

```
X509v3 extensions:  
  X509v3 Basic Constraints:  
    CA:TRUE
```



TRUE means the certificate can be used to verify other certificates, i.e., the owner is a CA. For non-CA certificates, this field is FALSE.

Apache Setup

- A server has a responsibility to send out all the intermediate CA's certificates needed for verifying its own certificate.
- In Apache, all certificates including those from Intermediate CAs are put inside the certificate file listed in the directive.

```
<VirtualHost *:443>
  ServerName example.com
  DocumentRoot /var/www/Example
  DirectoryIndex index.html

  SSLEngine On
  SSLCertificateFile      /etc/apache2/ssl/bank_cert2.pem
  SSLCertificateKeyFile  /etc/apache2/ssl/bank_key.pem
  SSLCertificateChainFile /etc/apache2/ssl/modelIntCA_cert.pem
</VirtualHost>
```

Trusted CAs in the Real World

- Not all of the trusted CAs are present in all browsers.
- According to W3Techs in April 2017, Comodo takes most of the market share followed by IdenTrust, Symantec Group, GoDaddy Group, GlobalSign and DigiCert.
- The list of trusted CAs supported by browser can be found:
 - **For the Chrome browser:**
 - Settings -> Show advanced settings -> Manage Certificates
 - **For the Firefox browser:**
 - Edit -> Preferences -> Advanced -> Certificates -> View Certificates -> Certificate Manager -> Authorities

How PKI Defeats the MITM Attack

- Assume that Alice wants to visit `https://example.com`
- When the server sends its public key to Alice, an attacker intercepts the communication. The attacker can do the following things:
 - (a) Attacker forwards the authentic certificate from `example.com`
 - (b) Attacker creates a fake certificate
 - (c) Attacker sends his/her own certificate to Alice

(a) Attacker Forwards the Authentic Certificate

- The **goal** of the attack? To impersonate the server in order to do MITM attack between Alice and the server.
- Attacker (Mike) forwards the authentic certificate
- Alice sends to the server a **secret**, encrypted using the public key.
- The **secret** is used for establishing an encrypted channel between Alice and server
- Mike doesn't know the corresponding private key, so he cannot find the **secret**.
- Mike can't do much to the communication, except for DoS.
- **MITM attack fails.**

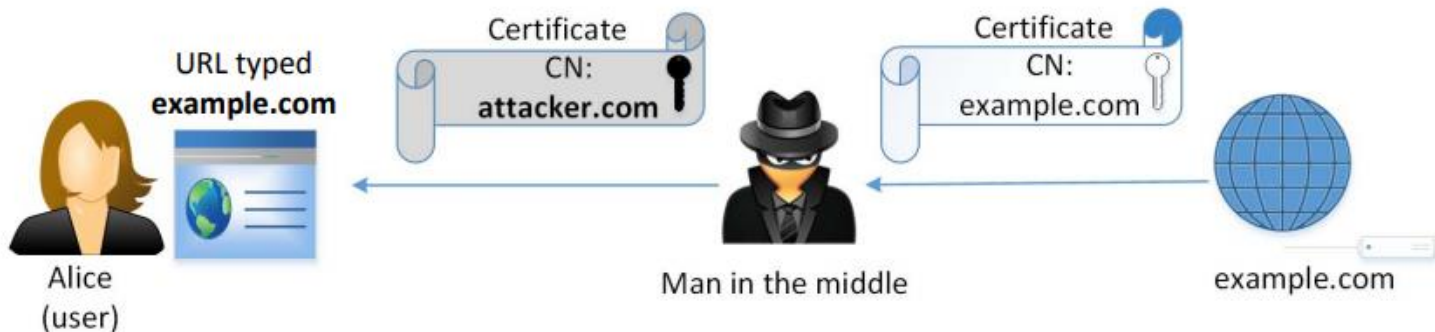
(b) Attacker Creates a Fake Certificate

- Attacker (Mike) creates fraudulent certificate for the example.com domain.
- Mike replaces the server's public key with his own public key.
- Trusted CAs will not sign Mike's certificate request as he does not own example.com.
- Mike can sign the fraudulent certificate by himself and create a self-signed certificate.
- Alice's browser will not find any trusted certificate to verify the received certificate and will give the following warning:

```
example.com uses an invalid security certificate.  
The certificate is not trusted because it is self-signed.
```

- **MITM attack fails** if the user decide to terminate the connection

(c) Attacker Sends His/Her Own Certificate



- Attacker's certificate is valid.
- Browser checks if the identity specified in the subject field of the certificate matches Alice's intent.
 - There is a mismatch: `attacker.com` \neq `example.com`
- Browser terminates handshake protocol: **MITM fails**

Emulating an MITM Attack

- DNS Attack is a typical approach to achieve MITM
 - We emulate an DNS attack by manually changing the /etc/hosts file on the user's machine to map example.com to the IP address of the attacker's machine.
- On attacker's machine we host a website for example.com.
 - We use the attacker's X.509 certificate to set up the server
 - The Common name field of the certificate contains **attacker32.com**
- When we visit example.com, we get an error message:

```
example.com uses an invalid security certificate.  
The certificate is only valid for attacker32.com  
(Error code: ssl_error_bad_cert_domain)
```

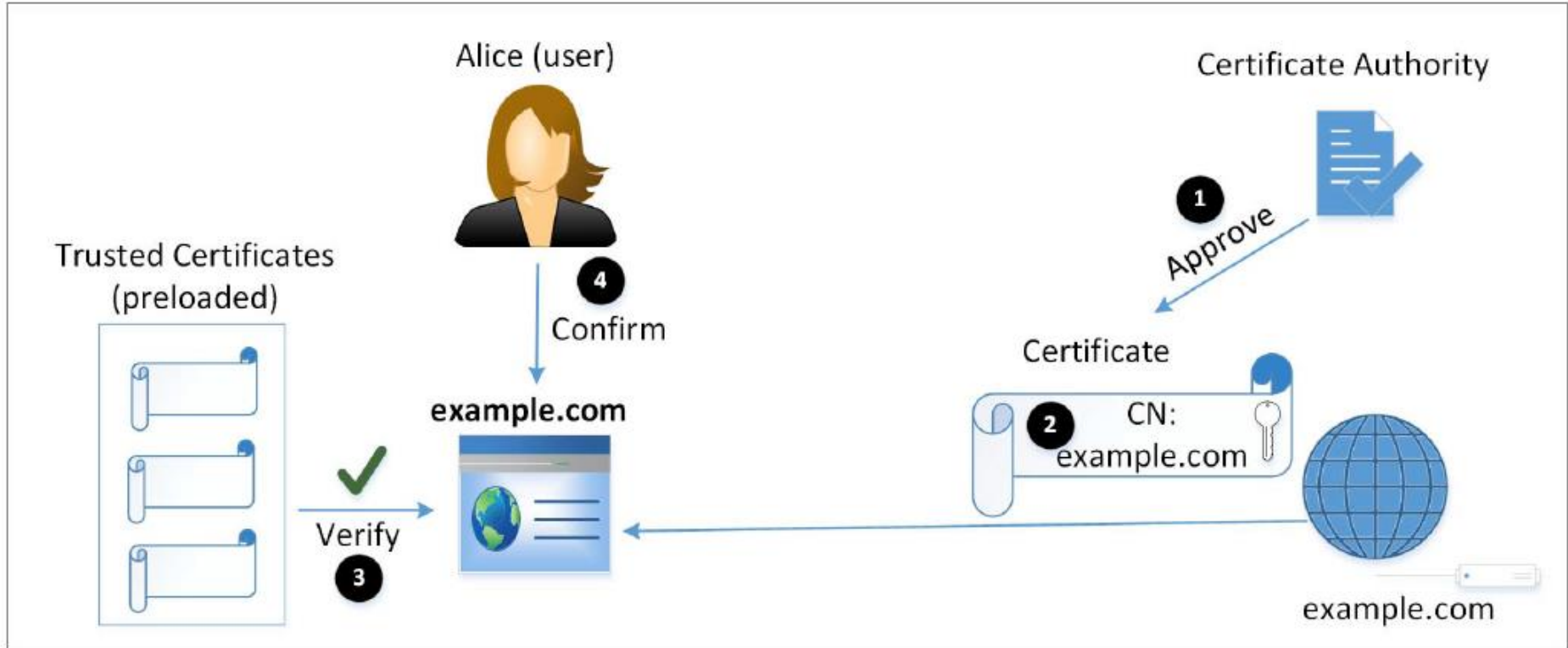
The Importance of Verifying Common Name

- During TLS/SSL handshake browsers conduct two important validations
 - 1) Checks whether the received certificate is valid or not.
 - 2) Verifies whether **the subject** (Common Names) in the certificate is the same as **the hostname of the server**.
- Not verifying the common name is a common mistake in software

The Man-In-The-Middle Proxy

- Proxy creates a self-signed CA certificate, which is installed on the user's browser
- The routing on the user machine is configured; all outgoing HTTPS traffic is directed towards the proxy machine
- When user tries to visit an HTTPS site:
 - Proxy intercepts communication
 - Creates a fake certificate
 - Browser already has the proxy's certificate in its trusted list to be able to verify all the fake certificates
 - Proxy becomes MITM

Attacks Surfaces on PKI



Attack on CA's Verification Process

- CA's job has two parts:
 - Verify the relationship between certificate applicant and the subject information inside the certificate
 - Put a digital signature on the certificate
- **Case study: Comodo Breach [March 2011]**
 - Popular root CA.
 - **The approval process in Southern Europe was compromised.**
 - Nine certificates were issued to seven domains and hence the attacker could provide false attestation.
 - One of the affected domain (a key domain for the Firefox browser):
`addons.mozilla.org`

Attack on CA's Signing Process

- If the CA's private key is compromised, attackers can sign a certificate with any arbitrary data in the subject field.
- **Case Study: the DigiNotar Breach [June-July 2011]**
 - A top commercial CA
 - Attacker got DigiNotar's private key
 - 531 rogue certificates were issued.
 - Traffic intended for Google subdomains was intercepted: MITM attack.
- How CAs Protect Their Private Key
 - Hardware Security Model (HSM)

Attacks on Algorithms

- Digital Certificates depend on two types of algorithms
 - one-way hash function and digital signature
- **Case Study: the Collision-Resistant Property of One-Way Hash**
 - At CRYPTO2004, Xiaoyun Wang demonstrated **collision attack against MD5**.
 - In February 2017, Google Research announced **SHattered attack against SHA-1**
 - Attack broke the collision-resistant property of SHA-1
 - Two different PDF files with the same SHA-1 has was created.
- **Countermeasures:** USE stronger algorithm, e.g. SHA256.

Attacks on User Confirmation

- After verifying the certificate from the server, client software is sure that the certificate is valid and authentic
- In addition, the software needs to confirm that the server is what the user intends to interact with.
- Confirmation involves two pieces of information
 - Information provided or approved by user
 - The common name field inside the server's certificate
 - Some software does not compare these two pieces of information: **security flaw**

Attacks on Confirmation: Case Study

Phishing Attack on Common Name with Unicode

- Zheng found out several browsers do not display the domain name correctly if name contains Unicode.
- xn-80ak6aa92e.com is encoded using Cyrillic characters. But domain name displayed by browser looks like apple.com
- Attack:
 - Get a certificate for xn-80ak6aa92e.com
 - Get user to visit xn-80ak6aa92e.com, so the common name is matched
 - User's browser shows that the website is apple.com. **User can be fooled.**
- Had the browser told the user that the actual domain is not the real apple.com, the user would stop.

Types of Digital Certificate

- From cheap/simple to more expensive/extensive validation processes:
 - 1) Domain Validated Certificates (DV)
 - 2) Organizational Validated Certificates (OV)
 - 3) Extended Validated Certificates (EV)

Domain Validated Certificates (DV)

- Most popular type of certificate.
- The CA verifies the domain records to check if the domain belongs to applicant.
- Domain Control Validation (DCV) is performed on domain name in the certificate request.
- DCV uses information in the WHOIS database
- DCV is conducted via
 - Email
 - HTTP
 - DNS

Organizational Validated Certificates (OV)

- Not very popular type of certificate.
- CAs verify the following before issuing OV certificates:
 - Domain control validation.
 - Applicant's identity and address.
 - Applicant's link to organization.
 - Organization's address.
 - Organization's WHOIS record.
 - Callback on organization's verified telephone number.

Extended Validated Certificates (EV)

- CAs issuing EV certificates require documents that are legally signed from registration authorities.
- EV CA validate the following information:
 - Domain control validation.
 - Verify the identity, authority, signature and link of the individual.
 - Verify the organization's physical address and telephone number.
 - Verify the operational existence.
 - Verify the legal and proper standings of the organization.
- EV certificate, hence, costs higher but is trustworthy.


How Browsers Display Certificate Types

Chrome browser

Cannot be verified

 Not secure | ~~https://~~test-sspev.verisign.com:2443/test-SSPEV-revoked-verisign.html

DV/OV Certificate

 Secure | https://www.microsoft.com/en-us/

EV Certificate

 PayPal, Inc. [US] | https://www.paypal.com/us/home

Firefox browser

Cannot be verified

 https://test-sspev.verisign.com:2443/test-SSPEV-revoked-verisign.html

DV/OV Certificate

 https://www.microsoft.com/en-us/

EV Certificate

 PayPal, Inc. (US) | https://www.paypal.com/us/home

Summary

- MITM attacks on public key cryptography
- Public-Key Infrastructure
- X.509 digital certificate
- Certificate Authority and how CA signs certificate
- How PKI defeats MITM attacks
- Attacks on PKI
- Different types of digital certificates