

# CSE 127 Computer Security

Stefan Savage, Fall 2025, Lecture 14

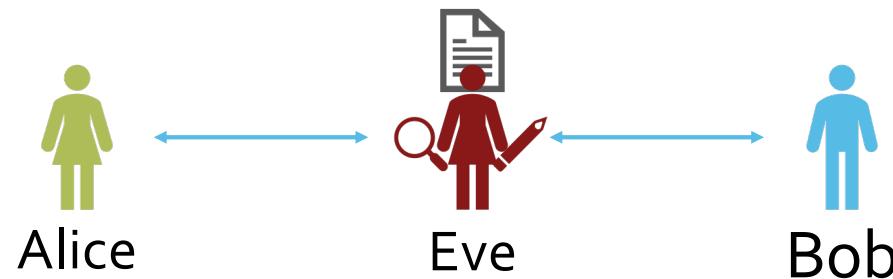
---

Cryptography II: PKI

## Using Cryptography (review)

---

Alice wants to send (a plaintext)  $m$  to Bob, via a channel that is controlled by Eve



# Cryptographic Primitives (review)

---

## Confidentiality

- Symmetric Encryption  
 $c = E_k(m), m = D_k(c)$
- Asymmetric Encryption  
 $c = E_K(m), m = D_k(c)$
- Combining Asymmetric with Symmetric  
 $k' \leftarrow r, E_K(k') || E_k(m)$
- You can rely on plaintext remaining secret.  
Ciphertext reveals nothing about plaintext contents
- You cannot rely on plaintext remaining unmodified.

## Integrity and Authenticity

- Symmetric MAC  
 $a = MAC_k(m)$
- Asymmetric Signature  
 $s = S_k(H(m))$   
 $V_K(s, H(m))$ : returns true or false
- You can rely that whoever generated the tag (MAC or signature) had the secret key.
- You cannot rely on tag not leaking information about the message.

# Using Cryptography

---

Assume we encrypt and sign a message from Alice to Bob

Assume decryption is successful and the signature verifies

What can Alice and Bob assume?

Bob knows that

- Alice knows the plaintext
- Alice signed the plaintext at some point in the past

Alice knows that

- Only Bob can extract the plaintext from the encrypted channel
- Bob can prove that Alice signed the plaintext
- True?

# Using Cryptography

---

Assume we encrypt and sign a message from Alice to Bob

Assume decryption is successful and the signature verifies

What can Alice and Bob assume?

Bob knows that

- Alice knows the plaintext (and anyone else she shared it with or copied it from)
- Alice (or someone with her private key) signed the plaintext at some point in the past

Alice knows that

- Only Bob (or someone with his private key) can extract the plaintext from the encrypted channel
- Bob (or anyone else) can prove that Alice (or someone with her private key) signed the plaintext

Remember, keys represent parties... they are not the parties themselves

# Quick Aside: Digital Signatures

---

## What Does Signing Mean?

- Signing is a mechanical operation that has *no meaning* in itself.

## What cryptography promises:

- Only someone who knows the private key can create a signature that verifies using the corresponding public key

## Meaning of a digital signature is a matter of convention

- Code signing: signer attests software is authorized to be installed
- Email signing: signer attests she wrote message
- Certificate signing: (coming up next!)

Both signer and verifier need to **agree** on meaning and trust that the meaning is enforced locally

# Using Cryptography

---

Things Alice does not know:

- Whether Bob received the message
- When Bob received the message
- How many times Bob received the message
- Whether Bob keeps the message secret

Things Bob does not know:

- Did Alice address this message to Bob
- Who sent this copy of the message
- When the message was sent
- Who else knows the plaintext

# Using Cryptography

---

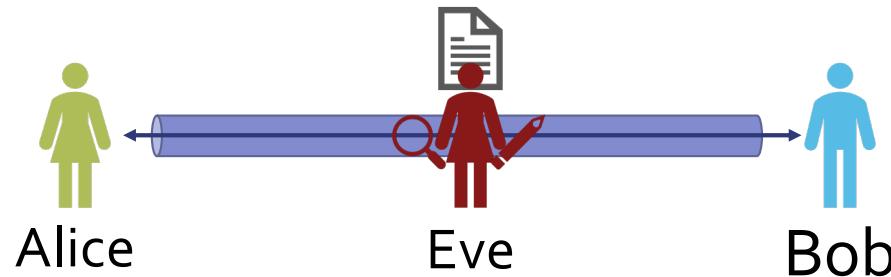
Alice wants to send (a plaintext)  $m$  to Bob, via a channel that is controlled by Eve.

Alice and Bob know each other's public keys. (assume pk crypto)

Goal: Alice and Bob establish a secure "pipe" (e.g., like https or ssh)

- Sign and encrypt all content (or encrypt and MAC for symmetry encryption)

If successful, Eve cannot see plaintext contents inside the pipe, or modify them without detection.

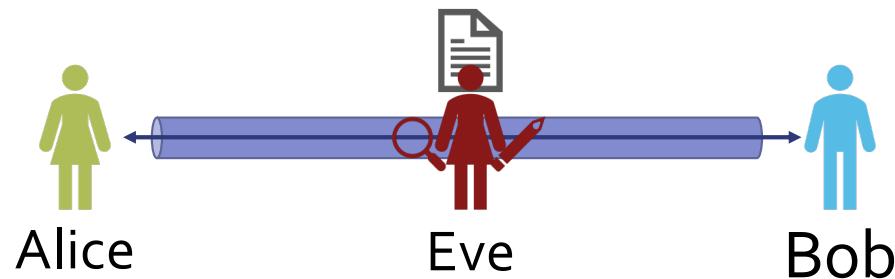


# Using Cryptography

---

Alice and Bob got secrecy + integrity + authenticity and everyone lived happily ever after, right?

Let's try to understand exactly how we might achieve this, and the problems along the way



# Public Key Infrastructure (PKI)

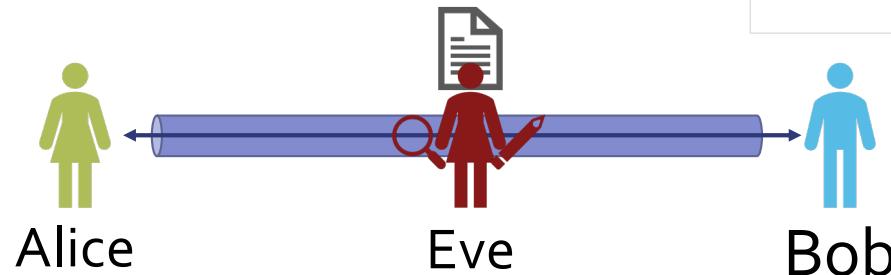
---

# Using Cryptography

Alice wants to send (a plaintext)  $m$  to Bob, via a channel that is controlled by Eve.

Alice and Bob know each other's public keys.

Alice and Bob establish a secure "pipe".



## Asymmetric Cryptography

- Public directory contains everyone's public key
- To encrypt to a person, get their public key from directory
- No need for shared secrets!



## Getting Public Keys

---

Alice and Bob need a way to get each other's public key.

Alice can send an unencrypted message to Bob:

- “Hey, send me your public key. Here’s mine.”

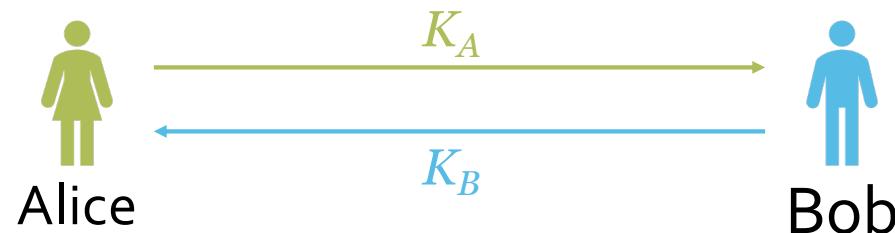
Bob sends Alice his public key.

They communicate securely ever after?

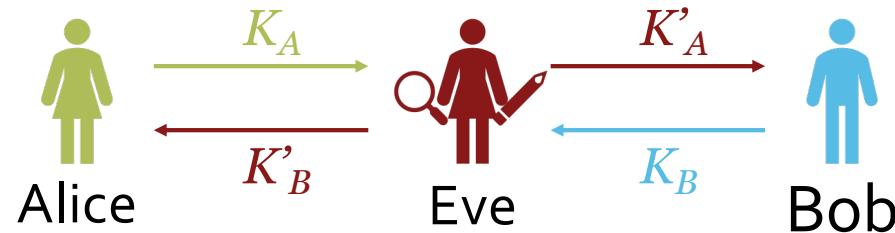
# Getting Public Keys

---

What they want to happen



What might happen instead

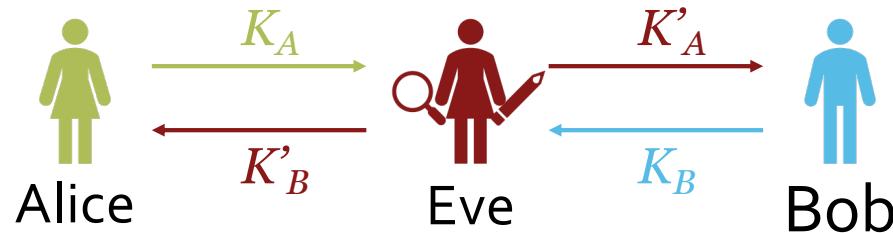


## Getting Public Keys

---

If Eve has person-in-the-middle capability, she can impersonate Alice to Bob and Bob to Alice.

- Eve becomes invisible gateway between them.
- Alice and Bob have no idea Eve is there.



# Getting Public Keys

---

Alice and Bob need a way to know that each has the **real** public key of the other.

Ideal solution: Alice and Bob meet in person and exchange public keys

Roughly equivalent: Alice and Bob meet in person and exchange public key fingerprints

- Key fingerprint: cryptographic hash of public key
- Public key itself can be sent in the open
- (aside: this is what Signal does)

  
Alice



  
Bob

# Getting Public Keys

---

Problem with ideal:

- We are back to pair-wise key establishment
- Alice and Bob need to meet
- Impractical to meet and verify key of everyone you talk to

Many security problems can be solved with a ***trusted third party***

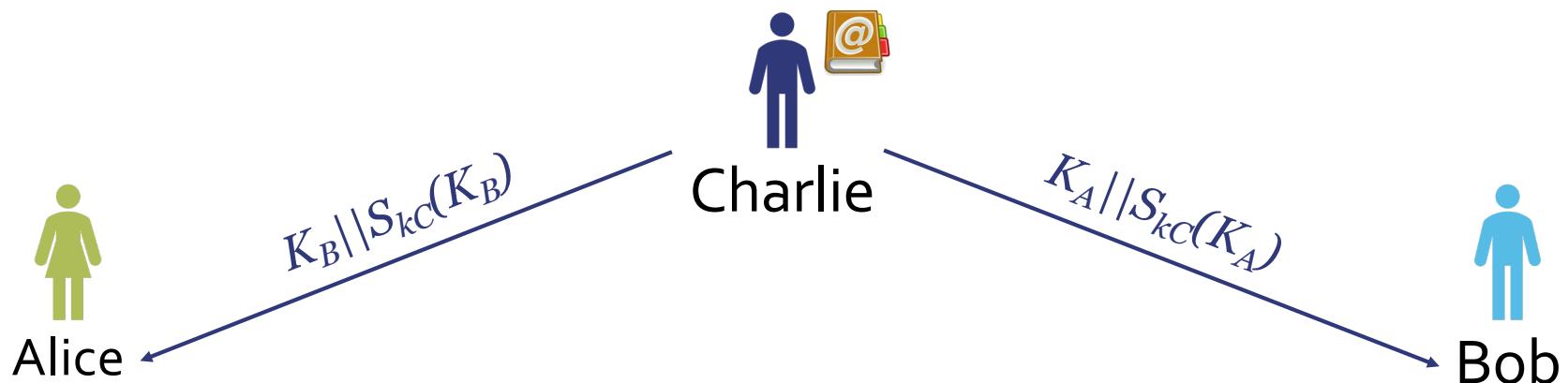


# Getting Public Keys

---

## Using a trusted intermediary

- Alice and Bob have already exchanged keys with Charlie
- Charlie sends signed message with Alice's key to Bob
- Charlie sends signed message with Bob's key to Alice
- Alice and Bob trust Charlie to send the real public keys
- Alice and Bob now have each other's public key



# Getting Public Keys

---

But every transaction is centralized through Charlie!  
We can do better...

Charlie creates a ***certificate*** that *attests*:

- "I, Charlie, verified that Alice's key is ..."

Charlie signs the certificate with his private key and **gives it** to Alice

- Alice now has certificate **attesting** to her public key

Alice sends Bob Charlie's certificate

Bob verifies the signature on certificate

Bob trusts Charlie, accepts public key from Alice

## Who is Charlie?

---

Two common models:

- PGP: Charlie is any other person you trust.
- Almost everywhere else: Charlie is a *Certificate Authority*.

# PGP Web of Trust

---

Pretty Good Privacy (PGP) is an application (and associated protocols) used for signing, encrypting, and decrypting texts, e-mails, files, directories, etc.

PGP allows one user to attest to the trustworthiness of another user's public key — ***key signing***

- PGP does not use the term “certificate”, but that's because its old...
- Public key has set of attestation signatures (certificates)

A user can indicate how much she trusts another user's signature on a key



## PGP Web of Trust

---

Alice's signature on Bob's PGP key means Alice claims that this is really Bob's key (and *ideally* has verified this)

- Email address and name associated with key are really his

Other people who trust Alice can use her signature on Bob's key to be sure it is Bob's key

How to decide?

# Certificate Authorities

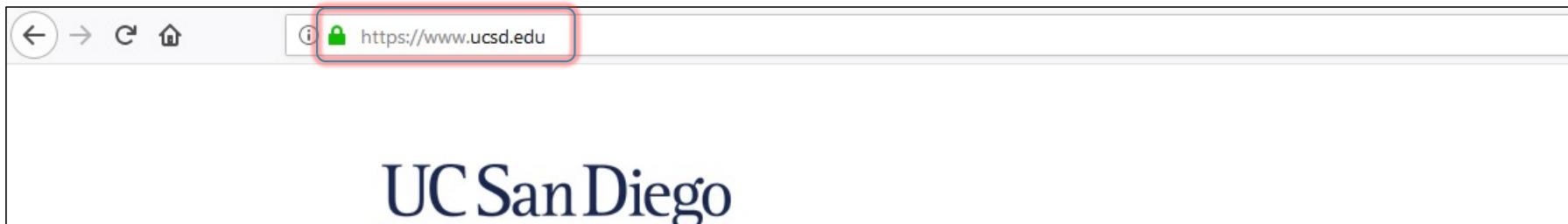
---

An alternative to PGP-like web of trust is to rely on centralized **Certificate Authorities**: trusted signers of public keys.

CA model used to sign certificates used on Web.

Your browser has a set of public keys of trusted CAs.

- Who makes this list?
- How many CAs are on the list?
- Who are these CAs?



## Certificate Authorities

File	Home	Insert	Page Layout	Formulas	Data	Review	View	Help	Tell me what you want to do		
A1	Owner										
1	Owner	Certificate Issuer Organization	Common Name or Certificate Name	Certificate Serial Number	Valid From [GMT]	Valid To [GMT]	Public Key Algorithm	Signature Hash Algorithm	Trust Bits	EV Policy OID(s)	Geographic Focus
2	AC Camerfirma, S.A.	AC Camerfirma S.A. CAF 82743287	Chambers of Commerce Root	003a0da427eab1ea	2003 Sep 30	2019 Sep 30	RSA 4096 bits	sha1WithRSAEncryption	Email;WebTrust;Email	1.3.6.1.4.1.17326.10.14.2.1.2	Spain
3	AC Camerfirma, S.A.	AC Camerfirma S.A.	Chambers of Commerce Root - 2008	003a0da427eab1ea	2008 Aug 01	2018 Jul 31	RSA 4096 bits	sha1WithRSAEncryption	Email;WebTrust;Email	1.3.6.1.4.1.17326.10.14.2.1.2	Spain
4	AC Camerfirma, S.A.	AC Camerfirma S.A. CAF 82743287	Global ChamberSign Root - 2008	006cc0d3e05d762c3	2008 Aug 01	2018 Jul 31	RSA 4096 bits	sha1WithRSAEncryption	Email;WebTrust;Email	1.3.6.1.4.1.17326.10.8.12.1.2	Spain
5	Actalis	Actalis S.p.A./03538052067	Actalis Authentication Root CA	5705115742d4e3c1	2010 Sept 22	2010 Sept 22	RSA 4096 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.159.1.17.1	Italy
6	Amazon Trust Services	Amazon	Amazon Root CA 1	f8dca39e20f783a69635bc3	2015 May 26	2018 Jun 17	RSA 4096 bits	sha256WithRSAEncryption	WebTrust;Email	2.23.140.1	USA, Global
7	Amazon Trust Services	Amazon	Amazon Root CA 2	5869fa0fa5e567fb83	2015 May 26	2018 Jun 17	RSA 4096 bits	sha384WithRSAEncryption	WebTrust;Email	2.23.140.1	USA, Global
8	Amazon Trust Services	Amazon	Amazon Root CA 3	73666397e009d9e87e0630324a	2015 May 26	2018 Jun 26	EC secp256r1	ecdsaWithSHA256	WebTrust;Email	2.23.140.1	USA, Global
9	Amazon Trust Services	Amazon	Amazon Root CA 4	b104c2945e717b2c2d1a1e01	2015 May 26	2018 Jun 26	EC secp384r1	ecdsaWithSHA384	WebTrust;Email	2.23.140.1	USA, Global
10	Amazon Trust Services	Starfield Technologies, Inc.	Starfield Services Root Certificate Authority®	0	2009 Sept 07	2017 Oct 31	RSA 2048 bits	sha256WithRSAEncryption	WebTrust	2.23.140.1	USA, Global
11	Asseco Data Systems S.A.	Unitech S.p.z.o.	Certum CA	0	1992002 Sept 11	2027 Jun 11	RSA 2048 bits	sha1WithRSAEncryption	Email	Not EV	Poland
12	Asseco Data Systems S.A.	Unitech Technologies S.A.	Certum Network CA	0	2004 Sept 11	2024 Sept 11	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	1.2.16.1.11527.2.5.1.1	Poland
13	Asseco Data Systems S.A.	Unitech Technologies S.A.	Certum Trusted Network CA 2	904a7f50295f23237a5e120d9	2011 Oct 06	2046 Oct 06	RSA 4096 bits	sha12WithRSAEncryption	WebTrust;Email	1.2.16.1.11527.2.5.1.1	Poland
14	Atos	Atos	Atos TrustRoot 2011	53c3cd26253c332	2011 July 07	2030 Dec 31	RSA 2048 bits	sha256WithRSAEncryption	WebTrust;Email	Not EV	Germany, Europe
15	Autoridad de Certificación Firmaprofesional	Firmaprofesional	Autoridad de Certificación Firmaprofesional	53e3c3beefb2485f	2009 May 20	2010 Dec 31	RSA 4096 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.11177.10.13.10	Spain
16	Buypass	Buypass AS-981363327	Buypass Class 2 Root CA	2	2010 Oct 26	2040 Oct 26	RSA 4096 bits	sha256WithRSAEncryption	WebTrust	Not EV	Norway, Europe
17	Buypass	Buypass AS-981363327	Buypass Class 3 Root CA	2	2010 Oct 26	2040 Oct 26	RSA 4096 bits	sha256WithRSAEncryption	WebTrust	2.16.578.1.26.1.3.3	Norway, Europe
18	CerticáImara	Sociedad Cameral de Certificación A.R.A.-2 CerticáImara S.A.	e52937b01e5357f989bc0e80	2009 Nov 27	2030 Apr 02	RSA 4096 bits	sha1WithRSAEncryption	Email	Not EV	Colombia and Andean Region	
19	Certinomis / Docopact	Certinomis	Certinomis - Root CA	1	2013 Oct 21	2033 Oct 21	RSA 4096 bits	sha256WithRSAEncryption	WebTrust	Not EV	France
20	certSIGN	certSIGN	certSIGN Root CA	2.000651e11	2009 Jun 04	2013 Jul 04	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	Romania
21	Certum	Certum	Certum Internal Certification Authority	0	19940511 Sept 11	2024 Sept 11	184048 bits	sha1WithRSAEncryption	WebTrust;Email	1.2.16.1.11527.2.5.1.1	China
22	ChungHwa Telecom	ChungHwa Telecom Co., Ltd.	ChungHwa Telecom Co., Ltd. - PKI Root	fe56457caf897005e4064061	2009 Dec 20	2030 Dec 20	RSA 4096 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	Taiwan
23	Comodo CA	Comodo CA Limited	AAA Certificate Services	1	2008 Jan 01	2028 Dec 31	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	USA, UK, Global
24	Comodo CA	Comodo CA Limited	AddTrust Class 1 Root	1	2009 May 30	2020 May 30	RSA 2048 bits	sha1WithRSAEncryption	Email	Not EV	USA, UK, Global
25	Comodo CA	Comodo CA Limited	AddTrust External CA Root	1	2009 May 30	2020 May 30	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
26	Comodo CA	Comodo CA Limited	COMODO ECC Certification Authority	8a826500b02ee203a50243d	2006 Dec 01	2029 Dec 31	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
27	Comodo CA	Comodo CA Limited	COMODO ECC Certification Authority	faa0020700544019b963992a	2009 Mar 06	2018 Jan 18	EC secp384r1	ecdsaWithSHA384	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
28	Comodo CA	Comodo CA Limited	COMODO RSA Certification Authority	9ac6d305f4ed8350389d	2010 Jan 19	2018 Jan 18	RSA 4096 bits	sha384WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
29	Comodo CA	Comodo CA Limited	USERTrust ECC Certification Authority	9e55494e5a27155d6e02d989c0e	2010 Feb 01	2018 Jan 18	EC secp384r1	ecdsaWithSHA384	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
30	Comodo CA	Comodo CA Limited	USERTrust Network	d193a03a04527a02d989c0e	2010 Feb 01	2018 Jan 18	RSA 4096 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.1649.1.2.1.5.1	USA, UK, Global
31	Comodo CA	Comodo CA Limited	UTM-UserTrust Client Authentication and Echosign	04040044d11c303525679899	2019 Jun 09	2019 Jun 09	RSA 2048 bits	sha1WithRSAEncryption	Email	Not EV	USA, UK, Global
32	ComSign	ComSign	ComSign	68314553ce7a53d43747744	2009 Mar 24	2020 Mar 19	RSA 2048 bits	sha1WithRSAEncryption	Email	Not EV	Israel
33	Concordi Administradora	Agencia Catalana de Certificado (NIE-ACC)	SecureSign Root CA11	jebd2121e4a8924ed401	2001 Jan 07	2031 Jan 07	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	Catalunya in Spain
34	Cybertrust Japan	Japan Certification Services, Inc.	D-TRUST Root CA 3	0	2001 Apr 08	2028 Apr 08	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	Japan
35	D-TRUST	D-TRUST GmbH	D-TRUST Root Class 3 A 2009	0	01/03/2013	2028 Sep 20	RSA 2048 bits	sha256WithRSAEncryption	Email	Not EV	Germany, Europe, Global
36	D-TRUST	D-TRUST GmbH	D-TRUST Root Class 3 A 2009	0983f3	2009 Nov 05	2029 Nov 05	RSA 2048 bits	sha256WithRSAEncryption	WebTrust	Not EV	Germany, Europe, Global
37	D-TRUST	D-TRUST GmbH	D-TRUST Root Class 3 A 2009 EV	0983f4	2009 Nov 05	2029 Nov 05	RSA 2048 bits	sha256WithRSAEncryption	WebTrust	1.3.6.1.4.1.4788.2.20.2.1	Germany, Europe, Global
38	Deutscher Sparkassen Ver	Deutscher Sparkassen Ver	Deutscher Sparkassen Ver Root CA	1b23405b64d42d52d60e1e	2010 Oct 22	2028 Oct 21	RSA 2048 bits	sha256WithRSAEncryption	Email	Not EV	Germany
39	Deutscher Sparkassen Ver	Deutscher Sparkassen Ver	TrustCenter Class 3 II	a4700010002e5d505d401	2009 Jan 12	2028 Jan 12	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	Germany
40	Deutche Post	Deutche Post	Deutsche Post	0f0eefc7a0491d401	2009 Jan 01	2028 Dec 31	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	France, Europe
41	DigiCert	Baltimore	Baltimore CyberTrust Root	0	02/09/2000	2020 May 12	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	France, Europe
42	DigiCert	Baltimore	Baltimore CyberTrust Root	0	02/09/2000	2020 May 12	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	France, Europe
43	DigiCert	Cybertrust, Inc	Cybertrust Global Root	0	2002 Dec 15	2021 Dec 15	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	1.3.6.1.4.1.6334.1.100.1	USA, Global
44	DigiCert	DigiCert Inc	DigiCert Assured ID Root CA	i0e57b646f47e50f1cbf0309	2013 Nov 10	2013 Nov 10	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	USA, Global
45	DigiCert	DigiCert Inc	DigiCert Assured ID Root G2	l3cd39367e6723bfa94d	2013 Aug 01	2018 Jan 15	RSA 2048 bits	sha256WithRSAEncryption	WebTrust;Email	Not EV	USA, Global
46	DigiCert	DigiCert Inc	DigiCert Assured ID Root G3	fa1af0d0b5494fa5d24a0e	2013 Aug 01	2018 Jan 15	EC secp384r1	ecdsaWithSHA384	WebTrust;Email	Not EV	USA, Global
47	DigiCert	DigiCert Inc	DigiCert Global Root CA	05696426b1a715a5991c0	2009 Nov 10	2013 Nov 10	RSA 2048 bits	sha1WithRSAEncryption	WebTrust;Email	Not EV	USA, Global
48	DigiCert	DigiCert Inc	DigiCert Global Root G2	eda11a9eb2b284a5109f04	2013 Aug 01	2018 Jan 15	RSA 2048 bits	sha256WithRSAEncryption	WebTrust;Email	Not EV	USA, Global
49	DigiCert	DigiCert Inc	DigiCert Global Root G3	0b0ff4a3d153e13af5f5a837	2013 Aug 01	2018 Jan 15	EC secp384r1	ecdsaWithSHA384	WebTrust;Email	Not EV	USA, Global

Count: 153

# Certificate Authorities (2021)

---

## Mozilla

- ~143 root certificates
- [https://wiki.mozilla.org/CA/Included\\_Certificates](https://wiki.mozilla.org/CA/Included_Certificates)

## iOS

- ~203 root certificates
- <https://support.apple.com/en-us/HT212140>

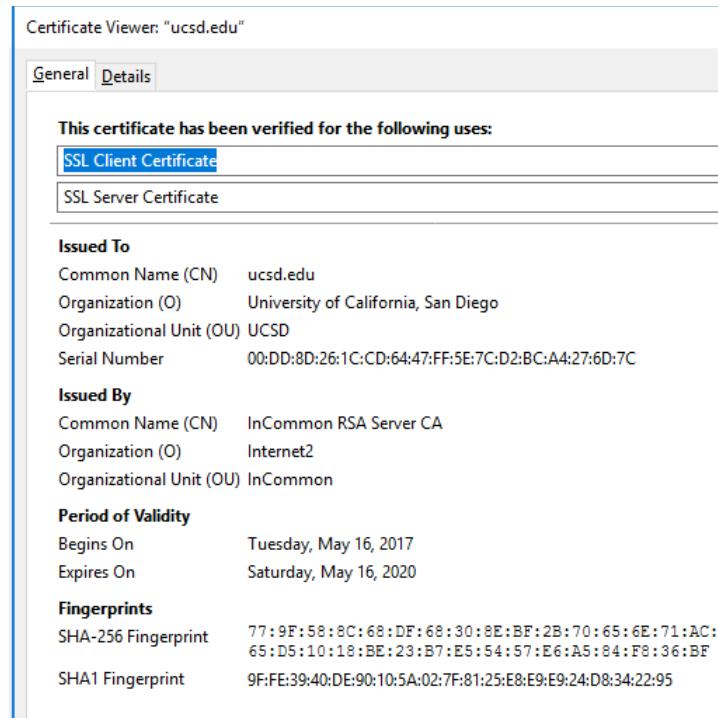
## Microsoft

- ~417 root certificates
- <http://aka.ms/RootCert>

# Certificate Authorities

## Certificate semantics:

- Subject (name, domain)
- Issuing CA
- Validity period
- Limitations on use  
(e.g. can it be used to sign other certificates)



# Using Certificates: Transport Layer Security (TLS)

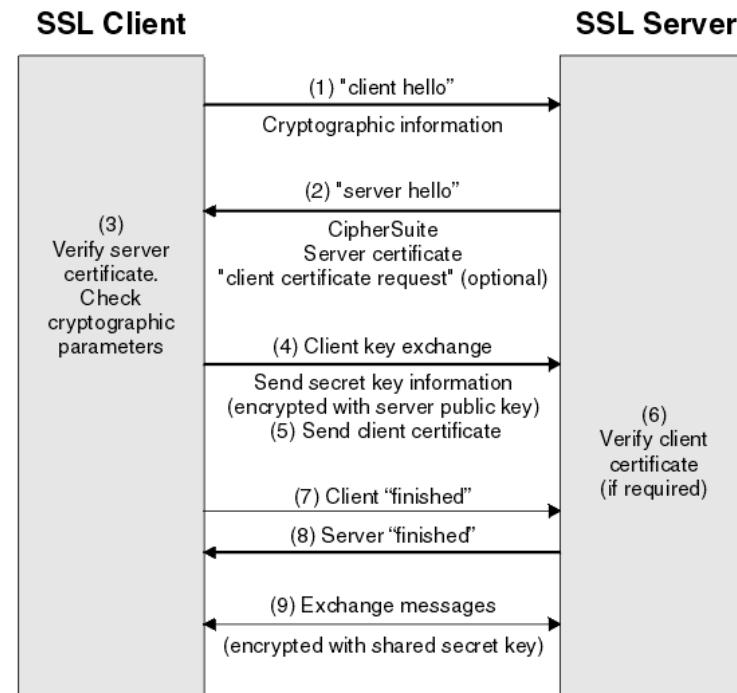
This is what makes https:// work

*“When secured by TLS, connections between a client (e.g., a web browser) and a server (e.g., wikipedia.org) have one or more of the following properties:*

- The connection is private (or secure) because **symmetric cryptography** is used to **encrypt** the data transmitted...
- The identity of the communicating parties can be **authenticated using public-key cryptography**...
- The connection ensures integrity because each message transmitted includes a message integrity check using a **message authentication code** to prevent undetected loss or alteration of the data during transmission.”

Details of protocol are complex, but the basic idea isn't:

- Browser gets and verifies server's certificate, and extracts PK
- Use PK to encrypt **random** symmetric session key
- Use session key to encrypt session and to key HMAC for integrity



## A quick step back...

---

Using TLS (i.e., https) what security is being claimed?

- Authenticity
  - That the server is who they say they are (e.g., [www.amazon.com](http://www.amazon.com))
  - That the client is who they say they are?
- Confidentiality
  - That no one but the client and server can read the messages sent between them
- Integrity
  - That no one can alter messages between client and server without being detected

What are we depending on?

- Crypto works
- The attestations of the CAs are reliable and their services are secure

# Certificate Authorities

---

Which CA can issue a certificate for mycompany.com?

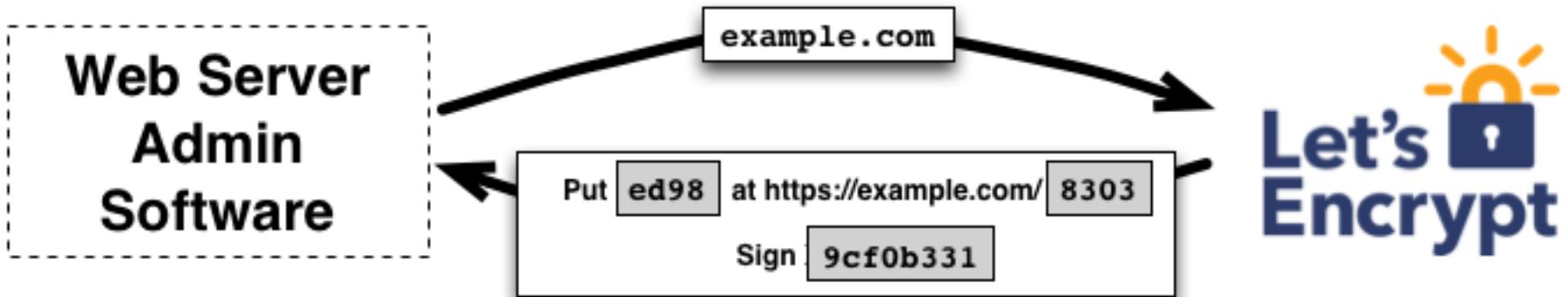
For fbi.gov?

How do site owners prove they are authorized to get a certificate for example.com?

- Traditionally
  - Provide tangible evidence of **ownership** (e.g., corp documents via FAX) and pay a fee
- Today, a new model dominates
  - Provide evidence of **control** over site domain name

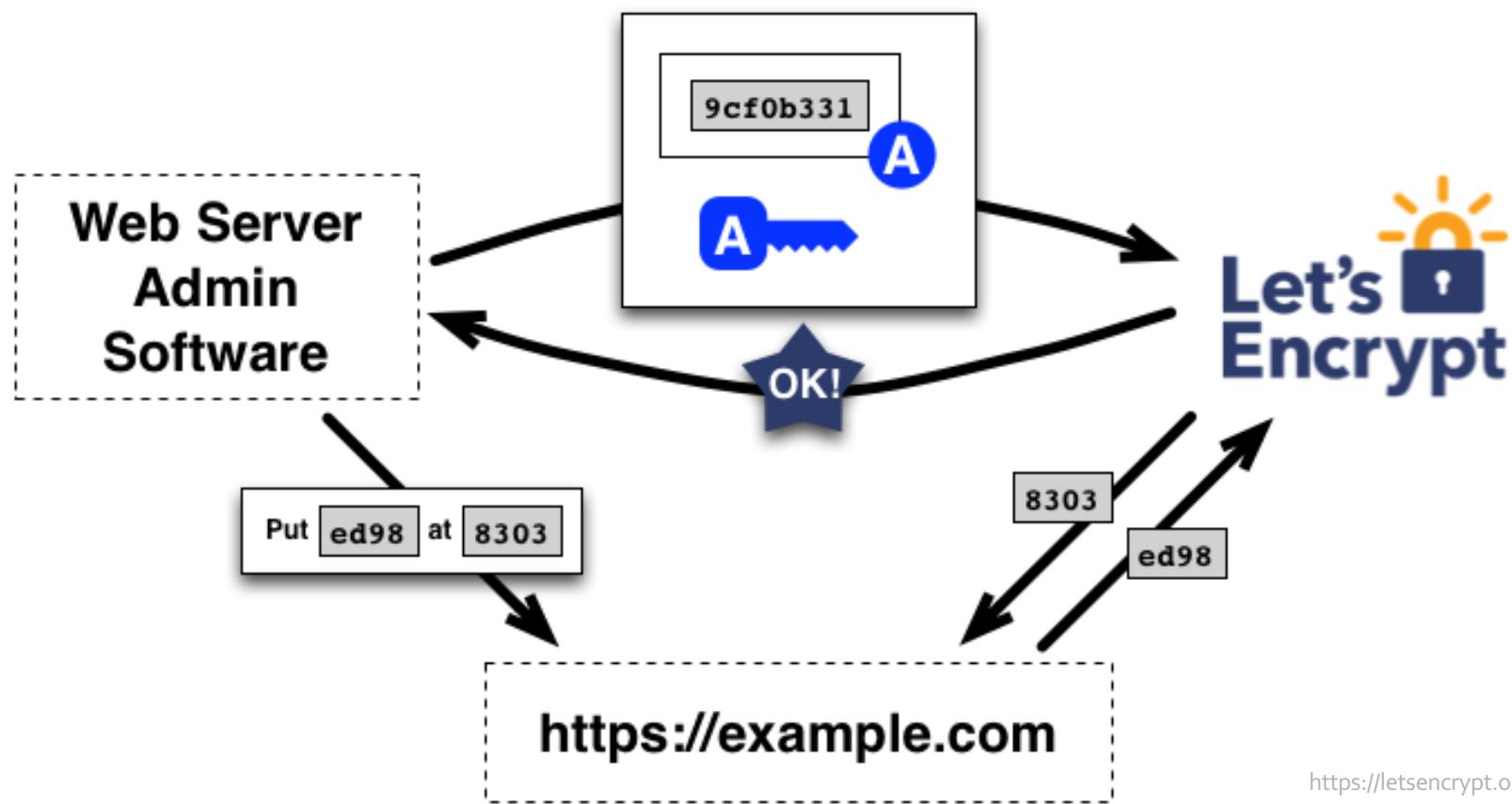
# Let's Encrypt

---



<https://letsencrypt.org/how-it-works/>

# Let's Encrypt



<https://letsencrypt.org/how-it-works/>

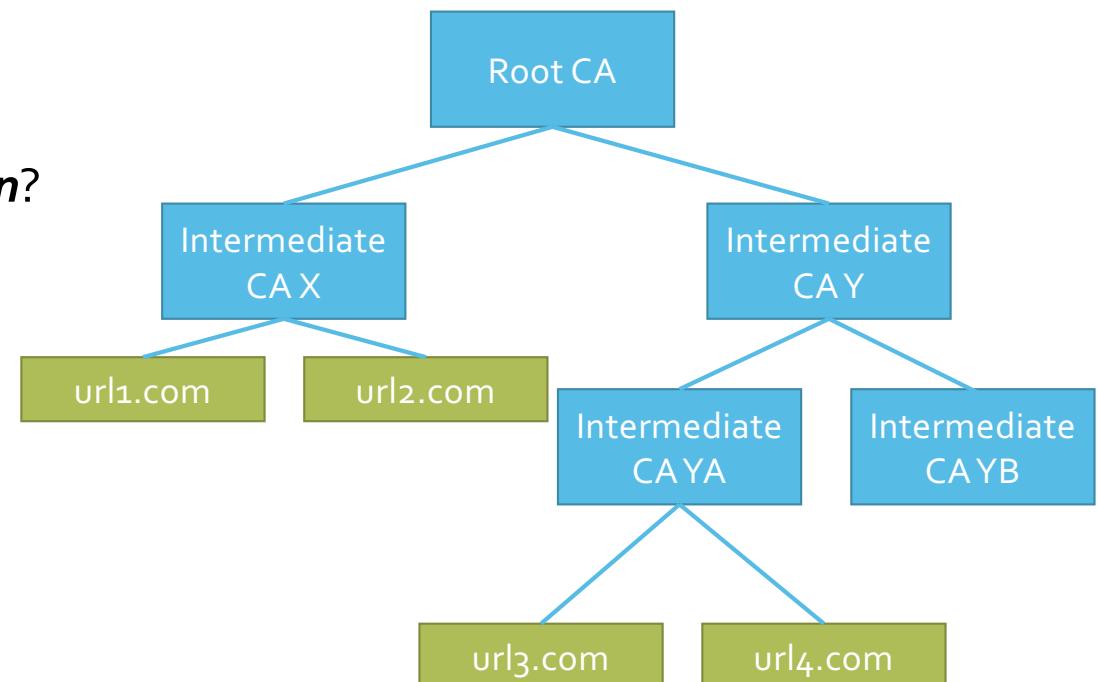
# Certificate Authorities

---

What if we take a  
***Trusted Third Party***  
and combine it with  
***another Layer of Indirection?***

## ***Certificate Hierarchy***

***Root CA*** signs keys for  
***Intermediate CAs***, which in  
turn sign keys for users (or  
other intermediate CAs)



# Certificate Authorities

Certificate hierarchy for ucsd.edu



Certificate Viewer: "ucsd.edu"

General Details

**Certificate Hierarchy**

- USERTrust RSA Certification Authority
- InCommon RSA Server CA
- ucsd.edu

**Certificate Fields**

- ucsd.edu
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
      - Not Before

## Aside: other uses of certificates

---

Certificates also used in code signing

- <https://source.android.com/security/apksigning/>
- <https://docs.microsoft.com/en-us/windows-hardware/drivers/install/driver-signing>
- <https://developer.apple.com/support/code-signing/>

Who is the CA?

What is the meaning of the signature?

- Alice released this app?
- Alice authorizes this app to run?
- Alice authorizes this app to access privileged resources?

# Certificate Revocation

---

What happens if someone steals your private key?

- They can impersonate you and read messages encrypted to you

Certificate expiration helps with this but not enough

- "Window of vulnerability"

CA and PGP PKIs support revocation

- Owner says: "I, Alice, revoke my public key ... do not use it."
- Signs revocation with her private key
- Others can verify Alice's signature, stop using key

## Certificate Revocation

---

How does Bob know if Alice's key has been revoked?

Bob asks Alice: "Has your key been revoked?"

Alice sends signed message: "No."

Does not work: if Alice's key has been compromised, then Eve could have forged the message "No."

Availability of trusted **revocation list** critical

# Certificate Revocation

---

In PGP model, only Alice can revoke her own key

- If Alice loses her private key, she can't revoke
  - Do not lose private PGP key
- Option: generate **revocation transaction** with key, store in secure place

In CA model, Alice asks CA to revoke certificate

- Alice does not need private key to do this, can authenticate herself through other means (e.g. login to CA service)

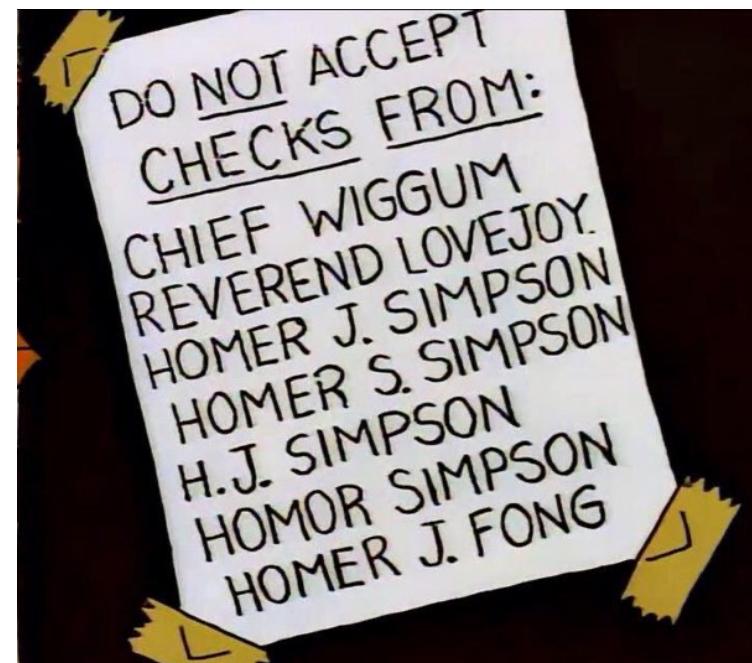
# Certificate Revocation

---

Two Mechanisms: CRL and OCSP

## ***Certificate Revocation List (CRL):***

- Certificate says where to get CRL
- Clients periodically download updated CRLs
- What if CRL server is down?



# Certificate Revocation

---

Two Mechanisms defined in standards: CRL and OCSP

## ***Online Certificate Status Protocol (OCSP):***

- Query CA about status of cert before trusting it
- “You said I can trust this key, but are you **still** sure?”

### OCSP Stapling

- Server includes recent OCSP status (signed by CA) along with certificate

### Aside: Certificate Pinning

- Remember which certificate was used for a particular domain and raise an alert if a different one is used later
- Fragile – doesn’t let host roll out new cert before old one expires

In practice: browsers used to check CRLs/OSCP, but most don’t now by default

- Why not? Performance and they break in some contexts (e.g., captive portals)  
Visit <https://revoked-isrgrootx1.letsencrypt.org/> with your browser to see
- Instead: ad-hoc solutions (e.g., Google harvests important revoked certs and pushes them to clients, CRLSets) in between standardization

## Some additional complexity: CDNs

HTTPS is secured by TLS

Page Info - https://www.fbi.gov/

General Media Feeds Permissions Security

Website Identity

Website: www.fbi.gov  
Owner: This website does not supply ownership information.  
Verified by: COMODO CA Limited  
Expires on: Monday, May 28, 2018

View Certificate

Privacy & History

Have I visited this website prior to today?  
Is this website storing information (cookies) on my computer?  
Have I saved any passwords for this website?

View Cookies View Saved Passwords

Technical Details

Connection Encrypted (TLS\_AES\_128\_GCM\_SHA256, 128 bit keys, TLS 1.3)

The page you are viewing was encrypted before being transmitted over the Internet.  
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Help

https://www.fbi.gov

Secure Connection

FBI

# Some additional complexity: CDNs

HTTPS is secured by TLS



Certificate Viewer: "ssl538122.cloudflaressl.com"

General Details

This certificate has been verified for the following uses:

SSL Client Certificate  
SSL Server Certificate

**Issued To**

Common Name (CN) **ssl538122.cloudflaressl.com** <Not Part Of Certificate>  
Organization (O) **<Not Part Of Certificate>**  
Organizational Unit (OU) Domain Control Validated  
Serial Number **30:12:54:E6:00:0D:B1:2C:51:E9:F8:A6:30:43:DF:24**

**Issued By**

Common Name (CN) COMODO ECC Domain Validation Secure Server CA 2  
Organization (O) COMODO CA Limited  
Organizational Unit (OU) <Not Part Of Certificate>

**Period of Validity**

Begins On **Saturday, November 18, 2017**  
Expires On **Monday, May 28, 2018**

**Fingerprints**

SHA-256 Fingerprint **AF:00:C5:9E:2D:EA:BD:7F:37:26:4E:F1:78:82:05:63:B1:3B:5C:D8:13:AD:55:CC:61:68:D8:40:31:B4:90:1A**  
SHA1 Fingerprint **8D:9F:5C:03:60:A2:07:59:67:82:A3:87:54:CC:3C:29:DC:AF:81:84**

# Content Delivery Networks (CDNs)

---

CDN: geographically distributed network of proxy servers

- Cache static content closer to the requester
- Improve latency
- Decrease network congestion
- Improve reliability and availability
  - DDOS protection
- Cloudflare, Akamai, CloudFront, etc

Mess up our nice security abstractions

- Now Alice deliberately wants her CDN to impersonate her to Bob!

## Content Delivery Networks (CDNs)

---

Bob wants to connect to [www.fbi.gov](http://www.fbi.gov)

Bob's browser attempts to get the corresponding IP address via DNS

Because FBI used Cloudflare CDN, DNS resolves to a Cloudflare server

But Bob's browser thinks it's talking to fbi.gov

Cloudflare needs to convince Bob's browser that it's really FBI

# Content Delivery Networks (CDNs)

Deputized via “Subject Alternate Name” field

- “Yeah, I’m cloudflaressl.com, but I’m authorized to communicate on behalf fbi.gov”



Who decides whether a CDN can get a given Subject Alternate Name in its cert?

Certificate Viewer: "ssl538122.cloudflaressl.com"

General Details

**Certificate Hierarchy**

- COMODO ECC Certification Authority
- COMODO ECC Domain Validation Secure Server CA 2
- ssl538122.cloudflaressl.com

**Certificate Fields**

- Certificate Basic Constraints
- Extended Key Usage
- Certificate Policies
- CRL Distribution Points
- Authority Information Access
- Certificate Subject Alt Name**
- Certificate Signature Algorithm
- Certificate Signature Value

**Field Value**

```
Not Critical
DNS Name: ssl538122.cloudflaressl.com
DNS Name: *.fbi.gov
DNS Name: fbi.gov
```

USERTrust RSA Certification Authority

↳ InCommon RSA Server CA

↳ cse.ucsd.edu

 **cse.ucsd.edu**

Issued by: InCommon RSA Server CA

Expires: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time

✓ This certificate is valid

▼ Details

Subject Name: \_\_\_\_\_

Country: US

Postal Code: 92093

State/Province: CA

Locality: La Jolla

Street Address: 9500 Gilman Drive

Organization: University of California, San Diego

Organizational Unit: UCSD

Common Name: cse.ucsd.edu

Issuer Name: \_\_\_\_\_

Country: US

State/Province: MI

Locality: Ann Arbor

Organization: Internet2

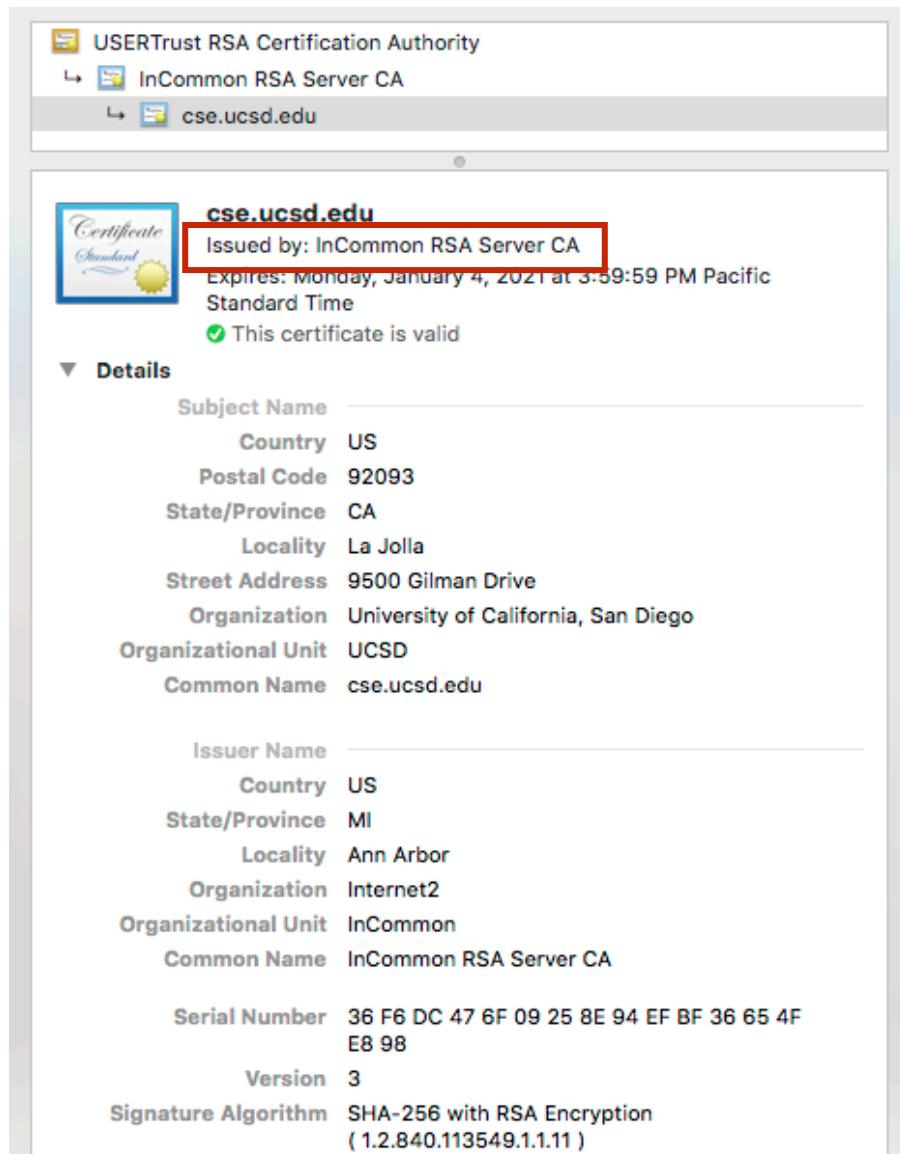
Organizational Unit: InCommon

Common Name: InCommon RSA Server CA

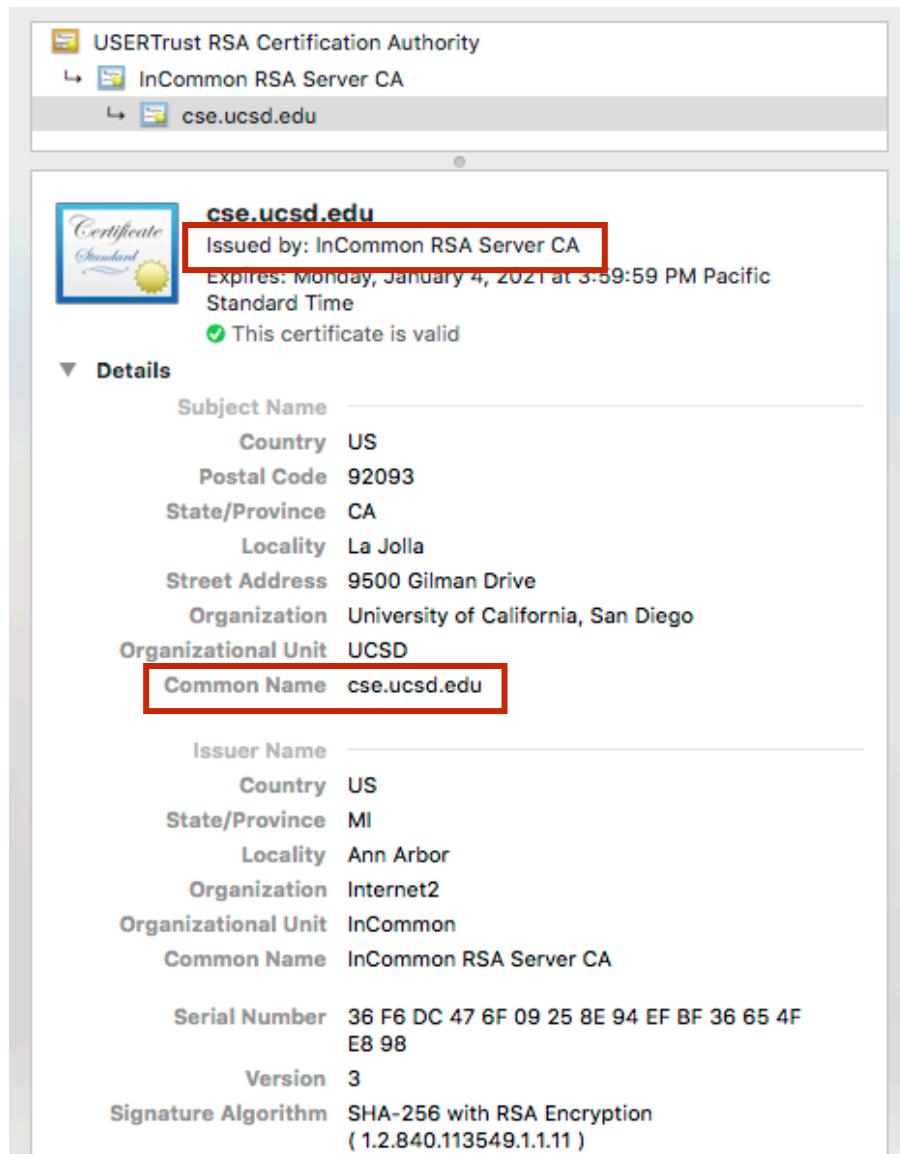
Serial Number: 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F  
E8 98

Version: 3

Signature Algorithm: SHA-256 with RSA Encryption  
( 1.2.840.113549.1.1.11 )



Who are we trusting? 



Who are we trusting?

Who is this cert for?

Key ID 1E 05 A3 77 8F 6C 96 E2 5B 87 4B A6 B4 86 AC  
71 00 0C E7 38

Extension Subject Alternative Name ( 2.5.29.17 )

Critical NO

DNS Name cse.ucsd.edu

DNS Name cs.ucsd.edu

DNS Name www-cs.ucsd.edu

DNS Name www-cse.ucsd.edu

DNS Name www.cs.ucsd.edu

DNS Name www.cse.ucsd.edu

Extension Certificate Policies ( 2.5.29.32 )

Critical NO

Policy ID #1 ( 1.3.6.1.4.1.5923.1.4.3.1.1 )

Qualifier ID #1 Certification Practice Statement ( 1.3.6.1.5.5.7.2.1 )

CPS URI [https://www.incommon.org/cert/repository/cps\\_ssl.pdf](https://www.incommon.org/cert/repository/cps_ssl.pdf)

Policy ID #2 ( 2.23.140.1.2.2 )

Extension CRL Distribution Points ( 2.5.29.31 )

Critical NO

URI <http://crl.incommon-rsa.org/InCommonRSAServerCA.crl>

Extension Certificate Authority Information Access  
( 1.3.6.1.5.5.7.1.1 )

Critical NO

Method #1 CA Issuers ( 1.3.6.1.5.5.7.48.2 )

URI [http://crt.usertrust.com/InCommonRSAServerCA\\_2.crt](http://crt.usertrust.com/InCommonRSAServerCA_2.crt)

Method #2 Online Certificate Status Protocol  
( 1.3.6.1.5.5.7.48.1 )

URI <http://ocsp.usertrust.com>

## Who is this cert for?

Issuer Name: \_\_\_\_\_

Country: US

State/Province: MI

Locality: Ann Arbor

Organization: Internet2

Organizational Unit: InCommon

Common Name: InCommon RSA Server CA

Serial Number: 36 F6 DC 47 6F 09 25 8E 94 EF BF 36 65 4F  
E8 98

Version: 3

Signature Algorithm: SHA-256 with RSA Encryption  
( 1.2.840.113549.1.1.11 )

Parameters: None

Not Valid Before: Thursday, January 4, 2018 at 4:00:00 PM Pacific Standard Time

Not Valid After: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time

**Public Key Info**

Algorithm: RSA Encryption ( 1.2.840.113549.1.1.1 )

Parameters: None

Public Key: 256 bytes : FA F9 1A 08 92 86 9C 7B ...

Exponent: 65537

Key Size: 2,048 bits

Key Usage: Encrypt, Verify, Wrap, Derive

Signature: 256 bytes : 6F 62 36 46 B7 43 28 04 ...

Extension: Key Usage ( 2.5.29.15 )

Critical: YES

Usage: Digital Signature, Key Encipherment

CSE's pub key info

Key ID 1E 05 A3 77 8F 6C 96 E2 5B 87 4B A6 B4 86 AC  
71 00 0C E7 38

Extension Subject Alternative Name ( 2.5.29.17 )  
Critical NO

DNS Name cse.ucsd.edu  
DNS Name cs.ucsd.edu  
DNS Name www-cs.ucsd.edu  
DNS Name www-cse.ucsd.edu  
DNS Name www.cs.ucsd.edu  
DNS Name www.cse.ucsd.edu

Extension Certificate Policies ( 2.5.29.32 )  
Critical NO

Policy ID #1 ( 1.3.6.1.4.1.5923.1.4.3.1.1 )  
Qualifier ID #1 Certification Practice Statement ( 1.3.6.1.5.5.7.2.1 )  
CPS URI [https://www.incommon.org/cert/repository/cps\\_ssl.pdf](https://www.incommon.org/cert/repository/cps_ssl.pdf)  
Policy ID #2 ( 2.23.140.1.2.2 )

Extension CRL Distribution Points ( 2.5.29.31 )  
Critical NO  
URI <http://crl.incommon-rsa.org/InCommonRSAServerCA.crl>

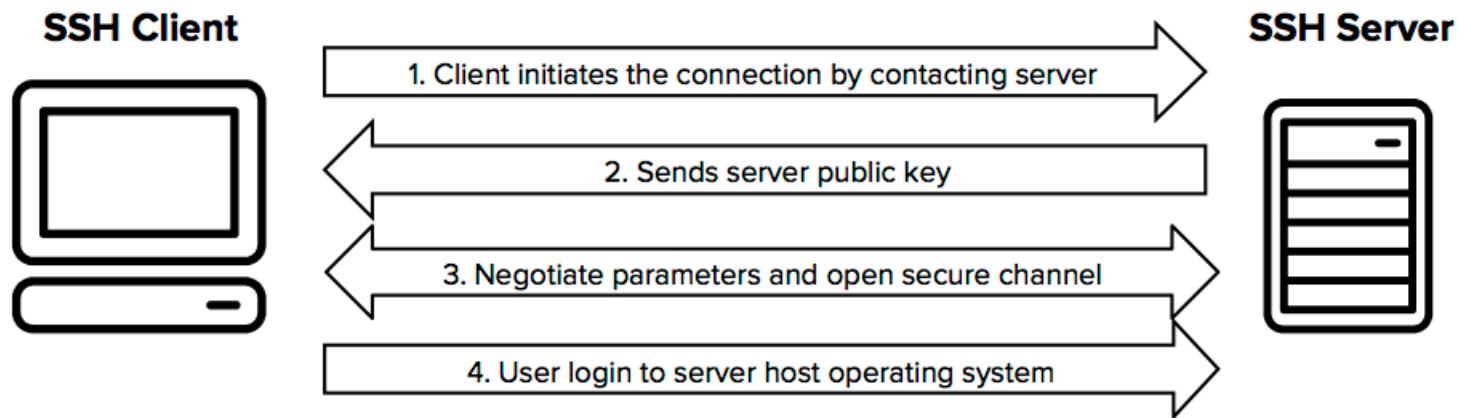
Extension Certificate Authority Information Access ( 1.3.6.1.5.5.7.1.1 )  
Critical NO  
Method #1 CA Issuers ( 1.3.6.1.5.5.7.48.2 )  
URI [http://crt.usertrust.com/InCommonRSAServerCA\\_2.crt](http://crt.usertrust.com/InCommonRSAServerCA_2.crt)  
Method #2 Online Certificate Status Protocol ( 1.3.6.1.5.5.7.48.1 )  
URI <http://ocsp.usertrust.com>

Where we should  
check for revocation  
information

## Another approach: Secure Shell (SSH)

---

*“Secure Shell (SSH) provides a secure channel over an unsecured network, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH.”*

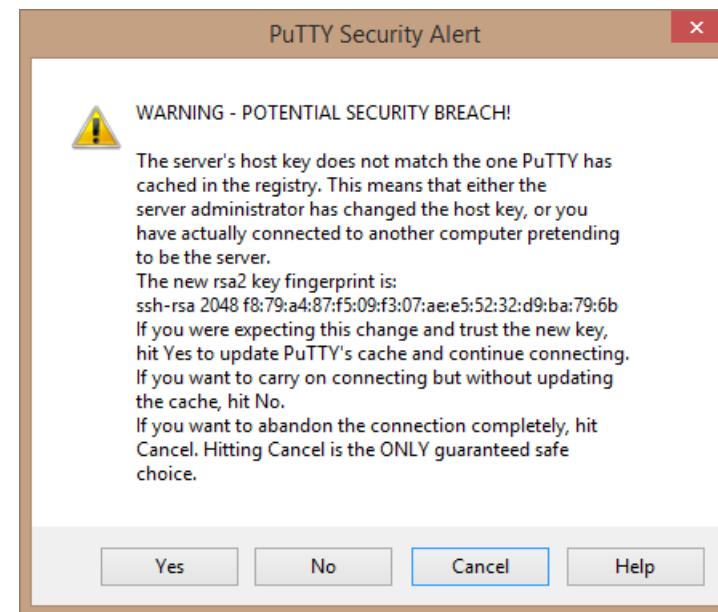


# Another approach: Secure Shell (SSH)

No trusted authorities

- Trust on First Use

Basically certificate pinning



<https://linode.com/docs/databases/oracle/securely-administer-oracle-xe-with-an-ssh-tunnel/>  
<https://software.intel.com/en-us/node/734703>

# Summary

---

Public key crypto is a powerful tool

- Underlies https, ssh, virtually all software updates, etc...
- But doesn't solve the key distribution problem

Certificate authorities (CA) occupy key (and trusted) role

- Third-party attestation of identity or access
- Have become hacking targets
  - 2011: Comodo & Diginotar issued fraudulent certs for Hotmail, Gmail, Skype, Yahoo Mail, Firefox...
  - 2013: TurkTrust issued cert for gmail
  - 2014: Indian Nic issued certs for Google and Yahoo!
  - 2016: WoSign issued cert for GitHub

Ongoing effort to police CAs – Certificate Transparency

- Make a public, searchable log of every new certificate minted... google can go check if anyone else got a cert that covers google.com or gmail.com

# Enjoy Thanksgiving Break!

---

No discussion section this week due to the break

Next week:

- Malware