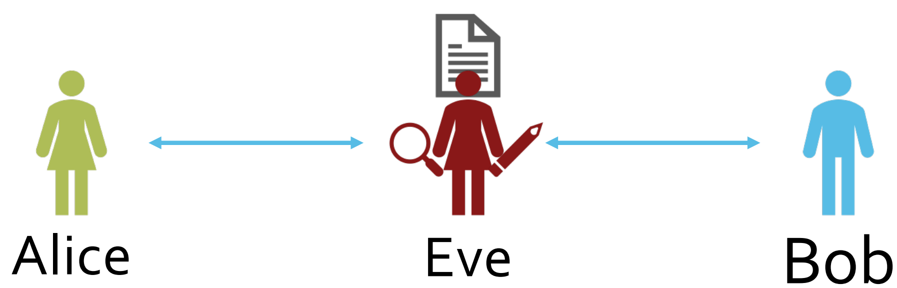
# CSE 127 Computer Security

Stefan Savage, Fall 2024, Lecture 10

## 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

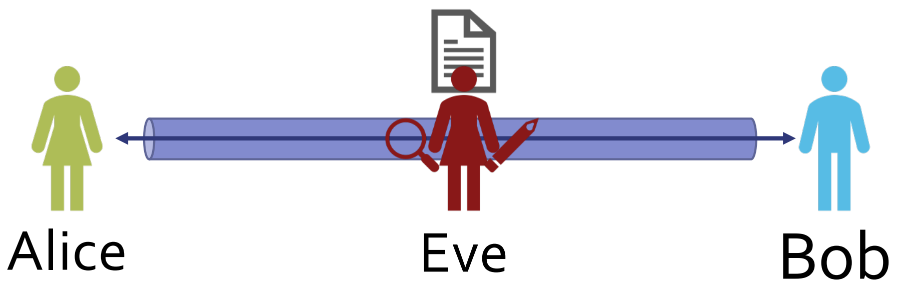
Alice wants to send (a plaintext) *m* to Bob, via a channel that is controlled by Eve.

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)

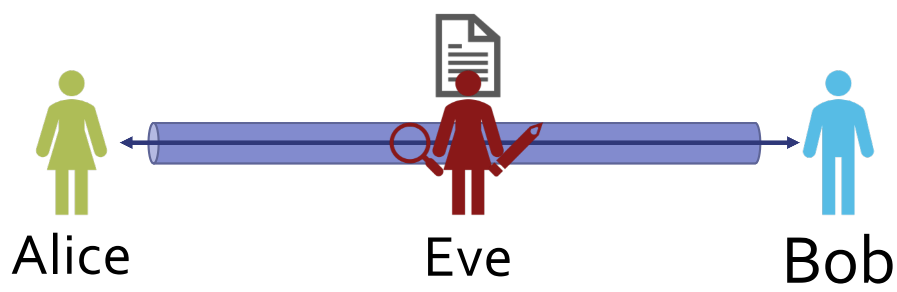
If successful, Eve cannot see plaintext contents inside the pipe, or modify them <u>without detection</u>.



# 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



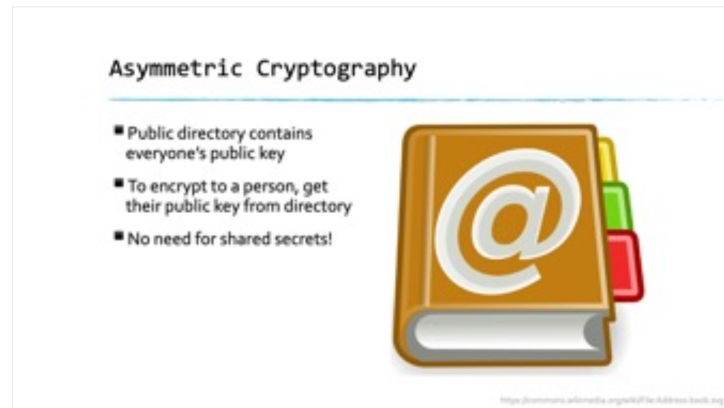
# Public Key Infrastructure (PKI)

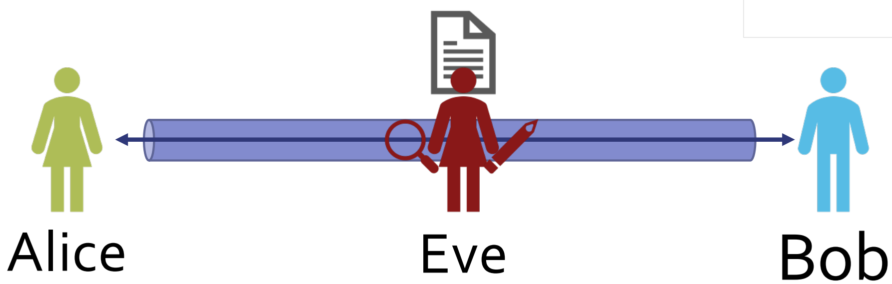
# Using Cryptography

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".



# Getting Public Keys

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

Alice can send an unencrypted message to Bob:

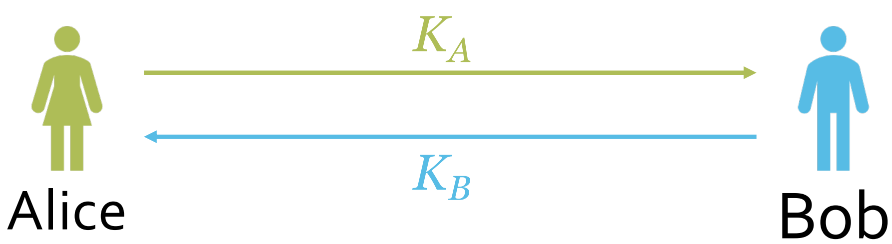
- "Hey, send me your public key. Here's mine."

Bob sends Alice his public key.

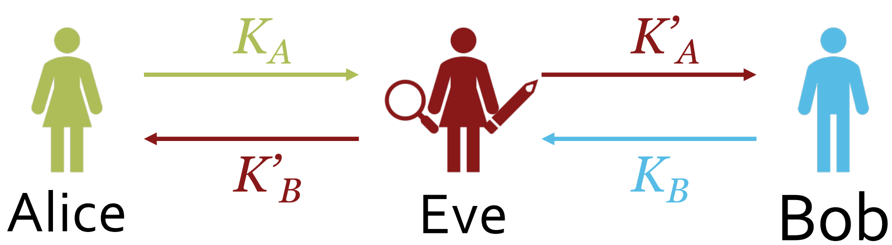
They communicate securely ever after?

# Getting Public Keys

What they want to happen



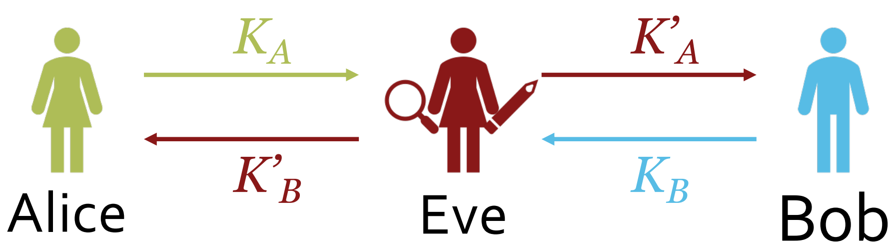

What might happen instead



# 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.



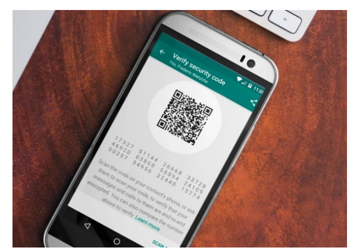
# Getting Public Keys

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:

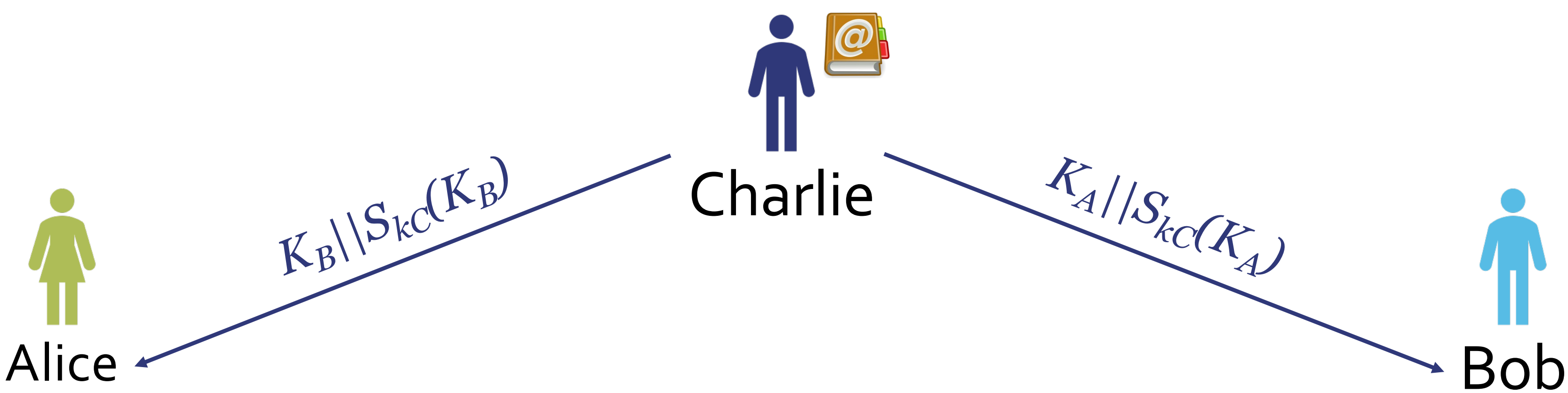
- We are back to pair-wise key establishment
- Alice and Bob need to meet
- Impractical to meet and verify key of everyone you talk to

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 trustworthyness 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?

https://xkcd.com/364/

# 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

| | 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 SA CIF A82743287 | Chambers of Commerce Root | 0 | 2003 Sep 30 | 2037 Sep 30 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | Spain |
| 3 | AC Camerfirma, S.A. | AC Camerfirma S.A. | Chambers of Commerce Root - 2008 | 00a3da427ea4b1aeda | 2008 Aug 01 | 2038 Jul 31 | RSA 4096 bits | sha1WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.17326.10.14.2.1.2 | Spain |
| 4 | AC Camerfirma, S.A. | AC Camerfirma SA CIF A82743287 | Global Chambersign Root | 0 | 2003 Sep 30 | 2037 Sep 30 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | Spain |
| 5 | AC Camerfirma, S.A. | AC Camerfirma S.A. | Global Chambersign Root - 2008 | 00c9cdd3e9d57d23ce | 2008 Aug 01 | 2038 Jul 31 | RSA 4096 bits | sha1WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.17326.10.8.12.1.2 | Spain |
| 6 | Actalis | Actalis S.p.A./03358520967 | Actalis Authentication Root CA | 570a119742c4e3cc | 2011 Sep 22 | 2030 Sep 22 | RSA 4096 bits | sha256WithRSAEncryption | Websites;Email | 1.3.159.1.17.1 | Italy |
| 7 | Amazon Trust Services | Amazon | Amazon Root CA 1 | f8c0a39e2f0788a43e696365bca | 2015 May 26 | 2038 Jan 17 | RSA 2048 bits | sha256WithRSAEncryption | Websites;Email | 2.23.140.1.1 | USA, Global |
| 8 | Amazon Trust Services | Amazon | Amazon Root CA 2 | 5869f0a0fe58678f85b26bb8a37 | 2015 May 26 | 2040 May 26 | RSA 4096 bits | sha384WithRSAEncryption | Websites;Email | 2.23.140.1.1 | USA, Global |
| 9 | Amazon Trust Services | Amazon | Amazon Root CA 3 | 736663f3b0b9ad9e89e7603f24a | 2015 May 26 | 2040 May 26 | EC secp256r1 | ecdsaWithSHA256 | Websites;Email | 2.23.140.1.1 | USA, Global |
| 10 | Amazon Trust Services | Amazon | Amazon Root CA 4 | b104c2943e5717b7b2cc81ac10e | 2015 May 26 | 2040 May 26 | EC secp384r1 | ecdsaWithSHA384 | Websites;Email | 2.23.140.1.1 | USA, Global |
| 11 | Amazon Trust Services | Starfield Technologies, Inc. | Starfield Services Root Certificate Authority | 0 | 2009 Sep 01 | 2037 Dec 31 | RSA 2048 bits | sha256WithRSAEncryption | Websites | 2.23.140.1.1 | USA, Global |
| 12 | Asseco Data Systems S.A. ( | Unizeto Sp. z o.o. | Certum CA | 10020 | 2002 Jun 11 | 2027 Jun 11 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | Poland |
| 13 | Asseco Data Systems S.A. ( | Unizeto Technologies S.A. | Certum Trusted Network CA | 0444c0 | 2008 Oct 22 | 2029 Dec 31 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | 1.2.616.1.113527.2.5.1.1 | Poland |
| 14 | Asseco Data Systems S.A. ( | Unizeto Technologies S.A. | Certum Trusted Network CA 2 | d04a4f250fc93237fcaa5e128de9 | 2011 Oct 06 | 2046 Oct 06 | RSA 4096 bits | sha512WithRSAEncryption | Websites;Email | 1.2.616.1.113527.2.5.1.1 | Poland |
| 15 | Atos | Atos | Atos TrustedRoot 2011 | 5c33cb622c5fb332 | 2011 Jul 07 | 2030 Dec 31 | RSA 2048 bits | sha256WithRSAEncryption | Websites;Email | Not EV | Germany, Europe |
| 16 | Autoridad de Certificacion | Firmaprofesional | Autoridad de Certificacion Firmaprofesiona | 53ec3beefbb2485f | 2009 May 20 | 2030 Dec 31 | RSA 4096 bits | sha1WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.13177.10.1.3.10 | Spain |
| 17 | Buypass | Buypass AS-983163327 | Buypass Class 2 Root CA | 2 | 2010 Oct 26 | 2040 Oct 26 | RSA 4096 bits | sha256WithRSAEncryption | Websites | Not EV | Norway, Europe |
| 18 | Buypass | Buypass AS-983163327 | Buypass Class 3 Root CA | 2 | 2010 Oct 26 | 2040 Oct 26 | RSA 4096 bits | sha256WithRSAEncryption | Websites | 2.16.578.1.26.1.3.3 | Norway, Europe |
| 19 | CerticÃ¡mara | Sociedad Cameral de CertificaciÃ³n | AC RaÃz CerticÃ¡mara S.A. | e52937be015e357f0698ccbec0c | 2006 Nov 27 | 2030 Apr 02 | RSA 4096 bits | sha1WithRSAEncryption | Email | Not EV | Colombia and Andean Regio |
| 20 | Certinomis / Docapost | Certinomis | Certinomis - Root CA | 1 | 2013 Oct 21 | 2033 Oct 21 | RSA 4096 bits | sha256WithRSAEncryption | Websites | Not EV | France |
| 21 | certSIGN | certSIGN | certSIGN ROOT CA | 2.00605E+11 | 2006 Jul 04 | 2031 Jul 04 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | Not EV | Romania |
| 22 | China Financial Certificatio | China Financial Certification Author | CFCA EV ROOT | 184accd6 | 2012 Aug 08 | 2029 Dec 31 | RSA 4096 bits | sha256WithRSAEncryption | Websites | 2.16.156.112554.3 | China |
| 23 | Chunghwa Telecom | Chunghwa Telecom Co., Ltd. | Chunghwa Telecom Co., Ltd. - ePKI Root Ce | 165475cafb897005ee406d2bc9d | 2004 Dec 20 | 2034 Dec 20 | RSA 4096 bits | sha1WithRSAEncryption | Websites;Email | Not EV | Taiwan |
| 24 | Comodo CA | Comodo CA Limited | AAA Certificate Services | 1 | 2004 Jan 01 | 2028 Dec 31 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | Not EV | USA, UK, Global |
| 25 | Comodo CA | AddTrust AB | AddTrust Class 1 CA Root | 1 | 2000 May 30 | 2020 May 30 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | USA, UK, Global |
| 26 | Comodo CA | AddTrust AB | AddTrust External CA Root | 1 | 2000 May 30 | 2020 May 30 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 27 | Comodo CA | COMODO CA Limited | COMODO Certification Authority | 8a8265e00b02ee3e350246e53d | 2006 Dec 01 | 2029 Dec 31 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 28 | Comodo CA | COMODO CA Limited | COMODO ECC Certification Authority | faa62007050544c019e9b63992a | 2008 Mar 06 | 2038 Jan 18 | EC secp384r1 | ecdsaWithSHA384 | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 29 | Comodo CA | COMODO CA Limited | COMODO RSA Certification Authority | 9cadb636fe01ff74ed85b03869d | 2010 Jan 19 | 2038 Jan 18 | RSA 4096 bits | sha384WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 30 | Comodo CA | The USERTRUST Network | USERTrust ECC Certification Authority | 9c55a94c5d27156decd8980cc26 | 2010 Feb 01 | 2038 Jan 18 | EC secp384r1 | ecdsaWithSHA384 | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 31 | Comodo CA | The USERTRUST Network | USERTrust RSA Certification Authority | d30fca3ca51a81bbc640e35032d | 2010 Feb 01 | 2038 Jan 18 | RSA 4096 bits | sha384WithRSAEncryption | Websites;Email | 1.3.6.1.4.1.6449.1.2.1.5.1 | USA, UK, Global |
| 32 | Comodo CA | The USERTRUST Network | UTN-USERFirst-Client Authentication and E | c8b500024b411d336252567c989 | 1999 Jul 09 | 2019 Jul 09 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | USA, UK, Global |
| 33 | ComSign | ComSign | ComSign CA | 68314558cea7b63e5fc34877744 | 2004 Mar 24 | 2029 Mar 19 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | Israel |
| 34 | Consorci AdministraciÃ³ O | Agencia Catalana de Certificacio (NI | EC-ACC | debd421de14a862ac04f3ddc401 | 2003 Jan 07 | 2031 Jan 07 | RSA 2048 bits | sha1WithRSAEncryption | Websites | Not EV | Catalunya in Spain |
| 35 | Cybertrust Japan / JCSI | Japan Certification Services, Inc. | SecureSign RootCA11 | 1 | 2009 Apr 08 | 2029 Apr 08 | RSA 2048 bits | sha1WithRSAEncryption | Websites | Not EV | Japan |
| 36 | D-TRUST | D-Trust GmbH | D-TRUST Root CA 3 2013 | 0fddac | 2013 Sep 20 | 2028 Sep 20 | RSA 2048 bits | sha256WithRSAEncryption | Email | Not EV | Germany, Europe, Global |
| 37 | D-TRUST | D-Trust GmbH | D-TRUST Root Class 3 CA 2 2009 | 0983f3 | 2009 Nov 05 | 2029 Nov 05 | RSA 2048 bits | sha256WithRSAEncryption | Websites | Not EV | Germany, Europe, Global |
| 38 | D-TRUST | D-Trust GmbH | D-TRUST Root Class 3 CA 2 EV 2009 | 0983f4 | 2009 Nov 05 | 2029 Nov 05 | RSA 2048 bits | sha256WithRSAEncryption | Websites | 1.3.6.1.4.1.4788.2.202.1 | Germany, Europe, Global |
| 39 | Deutscher Sparkassen Ver | Deutscher Sparkassen Verlag GmbH | S-TRUST Universal Root CA | b23405b64d4ed25dad9d61e1e | 2013 Oct 22 | 2038 Oct 21 | RSA 2048 bits | sha256WithRSAEncryption | Email | Not EV | Germany |
| 40 | Deutscher Sparkassen Ver | TC TrustCenter GmbH | TC TrustCenter Class 3 CA II | a4700010002e5a05dd63f0051bf | 2006 Jan 12 | 2025 Dec 31 | RSA 2048 bits | sha1WithRSAEncryption | Email | Not EV | Germany |
| 41 | Dhimyotis / Certigna | Dhimyotis | Certigna | 00fedce3010fc948ff | 2007 Jun 29 | 2027 Jun 29 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | Not EV | France, Europe |
| 42 | DigiCert | Baltimore | Baltimore CyberTrust Root | 020000b9 | 2000 May 12 | 2025 May 12 | RSA 2048 bits | sha1WithRSAEncryption | Websites;Email | Not EV | USA, Global |
| 43 | DigiCert | Cybertrust, Inc | Cybertrust Global Root | 0400000000010f85aa2d48 | 2006 Dec 15 | 2021 Dec 15 | RSA 2048 bits | sha1WithRSAEncryption | Websites | 1.3.6.1.4.1.6334.1.100.1 | USA, Global |
| 44 | DigiCert | DigiCert Inc | DigiCert Assured ID Root CA | e0e517d846fe8fe560fc1bf03039 | 2006 Nov 10 | 2031 Nov 10 | RSA 2048 bits | sha1WithRSAEncryptio | | | |
| 45 | DigiCert | DigiCert Inc | DigiCert Assured ID Root G2 | 1c3ad63967ea6723bfc3af9af44b | 2013 Aug 01 | 2038 Jan 15 | RSA 2048 bits | sha256WithRSAEncrypt | | | |
| 46 | DigiCert | DigiCert Inc | DigiCert Assured ID Root G3 | bafa1ddfa0b54944afcd24a06cec | 2013 Aug 01 | 2038 Jan 15 | EC secp384r1 | ecdsaWithSHA384 | | | |
| 47 | DigiCert | DigiCert Inc | DigiCert Global Root CA | 056904246b1a1756ac95991c74a | 2006 Nov 10 | 2031 Nov 10 | RSA 2048 bits | sha1WithRSAEncryptio | | | |
| 48 | DigiCert | DigiCert Inc | DigiCert Global Root G2 | 1e6a711a9a0bb2864b11d09fae5 | 2013 Aug 01 | 2038 Jan 15 | RSA 2048 bits | sha256WithRSAEncrypt | | | |

IncludedCACertificateReport

Ready

Count: 153

# Certificate Authorities (2021)

Mozilla

- ~143 root certificates
- https://wiki.mozilla.org/CA/Included_Certificates

iOS

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

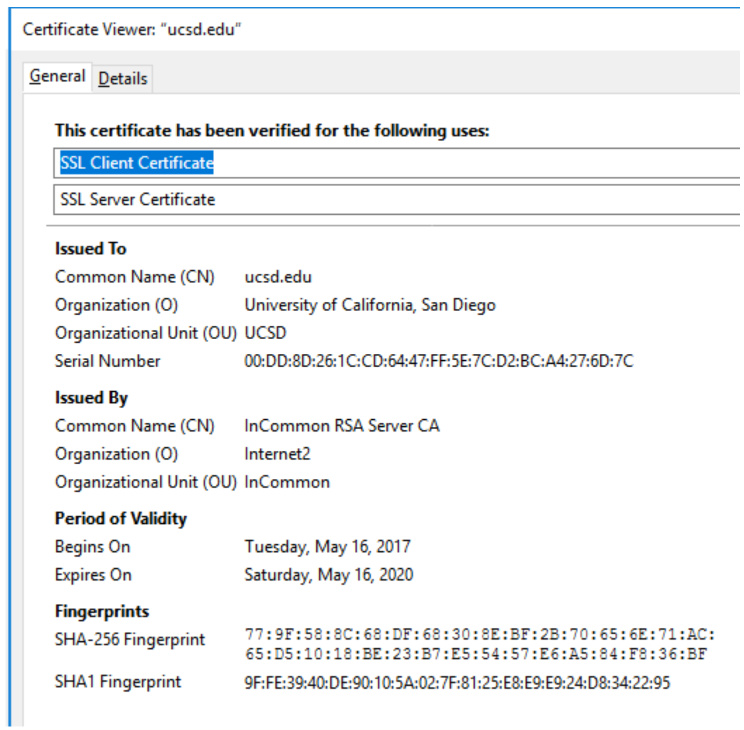
Microsoft

- ~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)

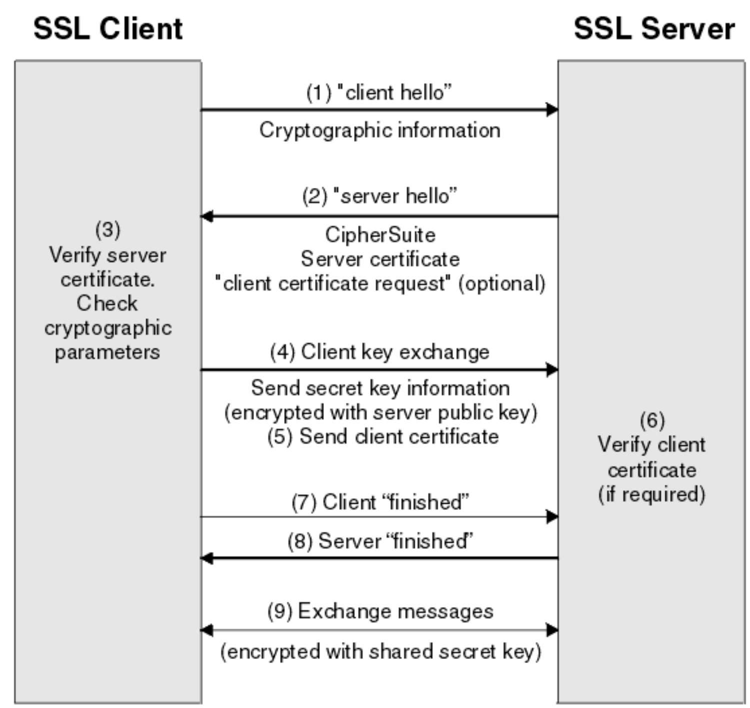
This is what makes https:// work

"*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




https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm

# 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)
  - 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

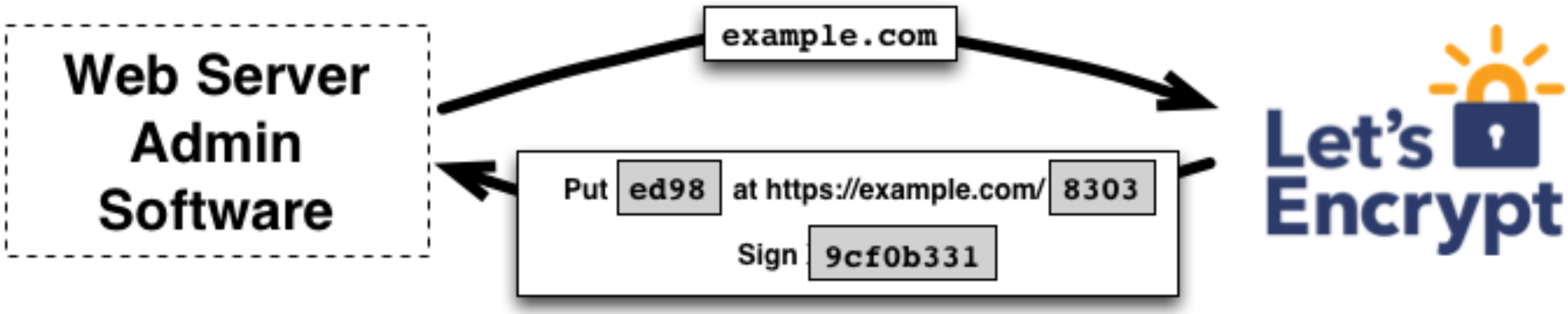
Which CA can issue a certificate for mycompany.com?

For fbi.gov?

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

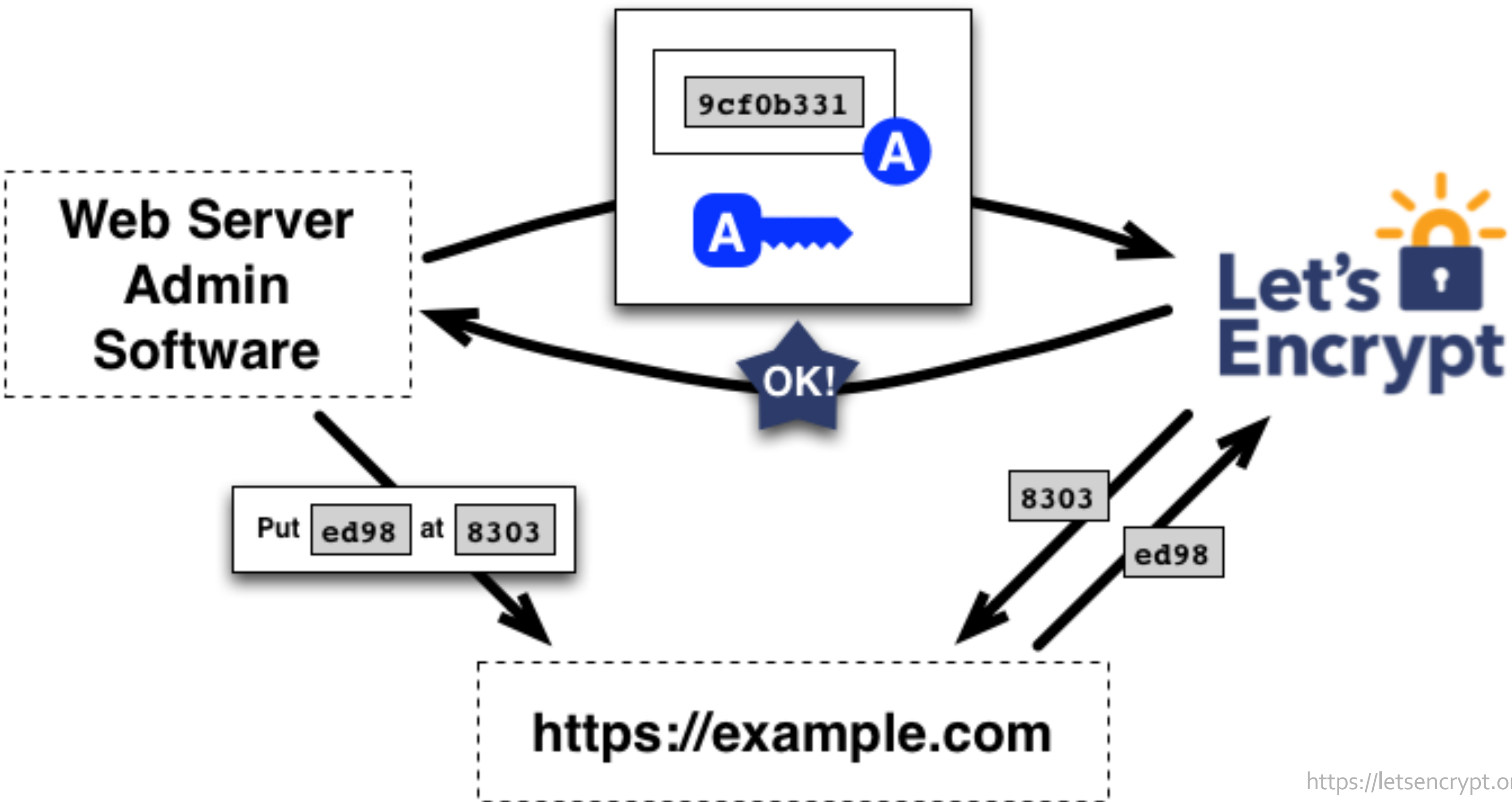
- Traditionally
  - Provide tangible evidence of **ownership** (e.g., corp documents via FAX) and pay a fee
- Today, a new model dominates
  - Provide evidence of **control** over site domain name

# Let’s Encrypt




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

# Let’s Encrypt



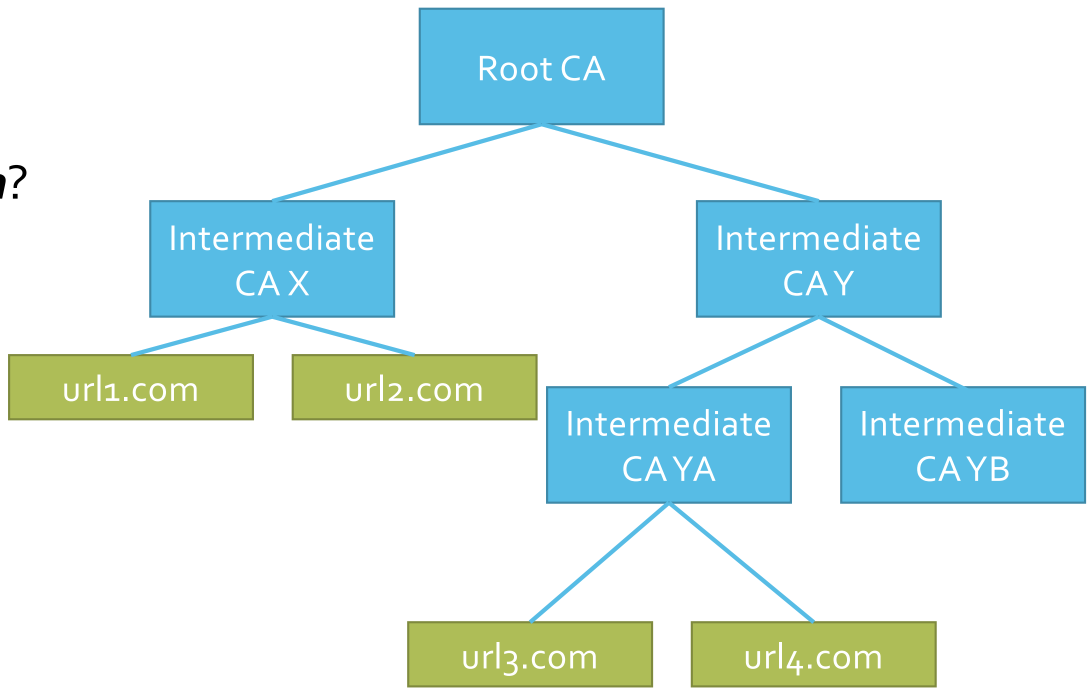

https://letsencrypt.org/how-it-works/

# Certificate Authorities

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

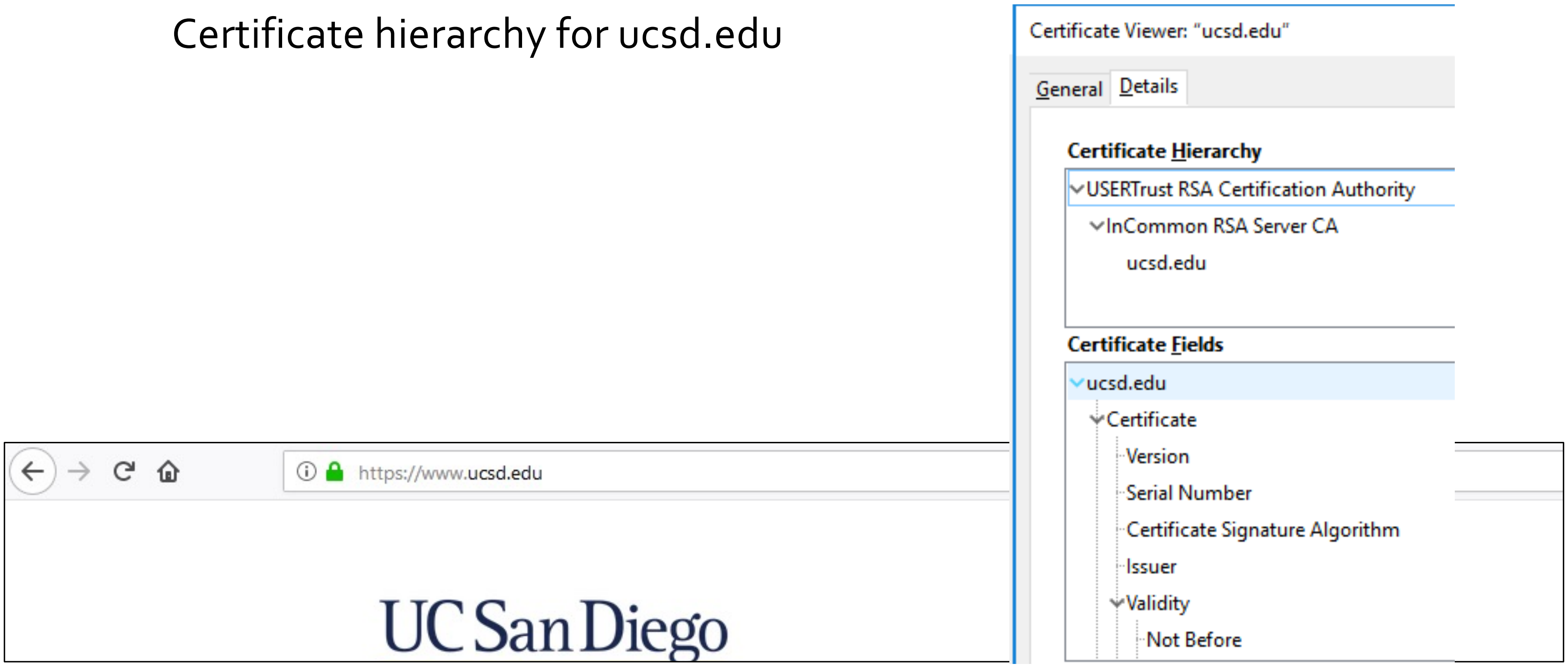
***Certificate Hierarchy***

***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



# 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

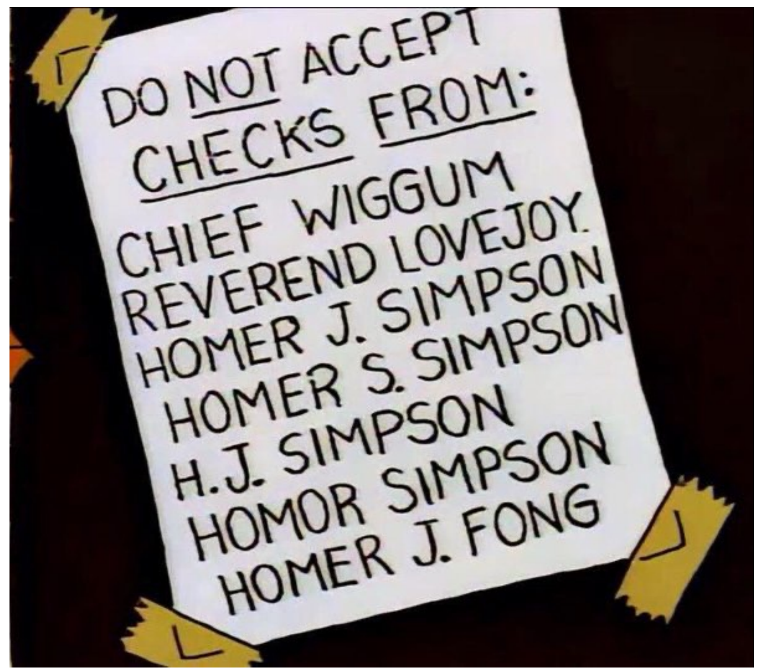
- Alice does not need private key to do this, can authenticate herself through other means (e.g. login to CA service)

# 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

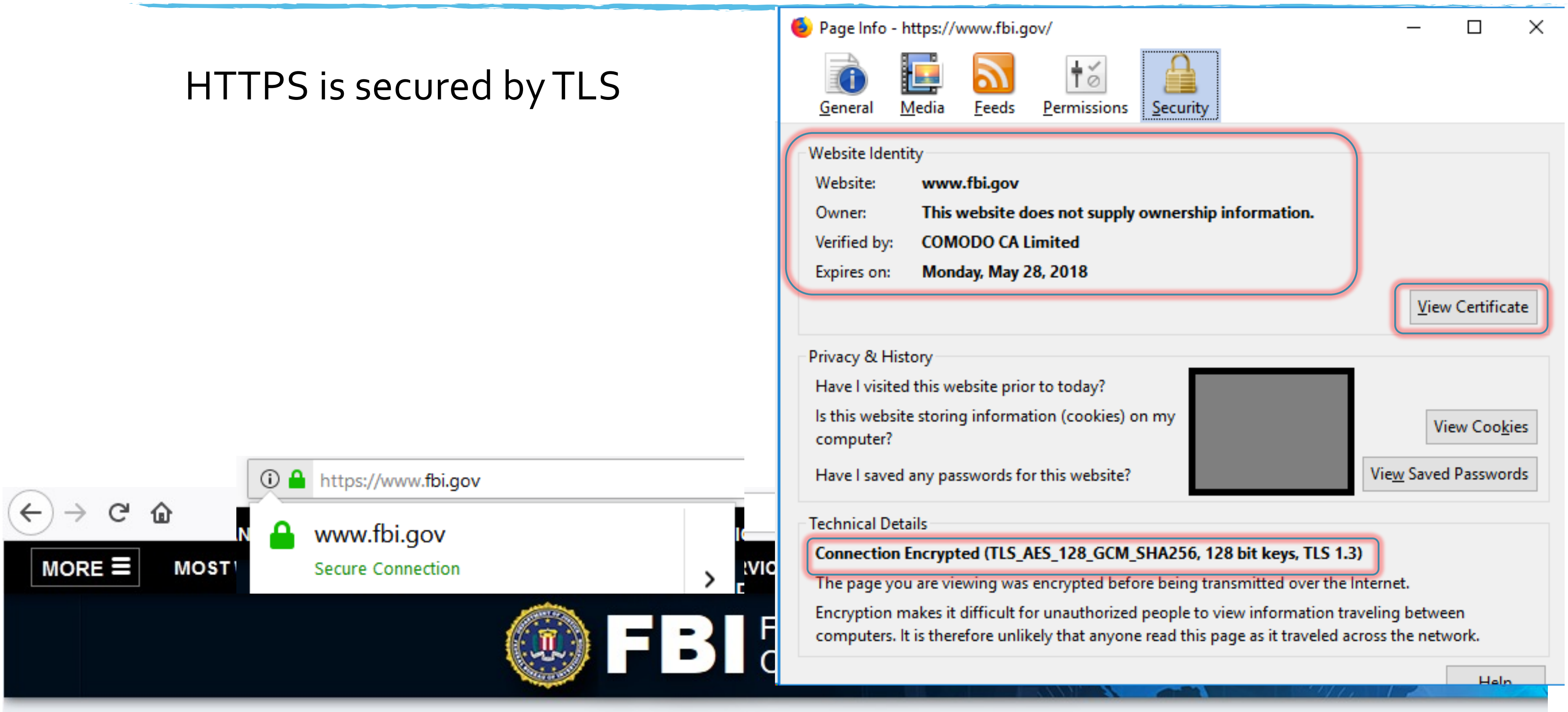
In practice: browsers used to check CRLs/OSCP, but most don't now by default

- 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)

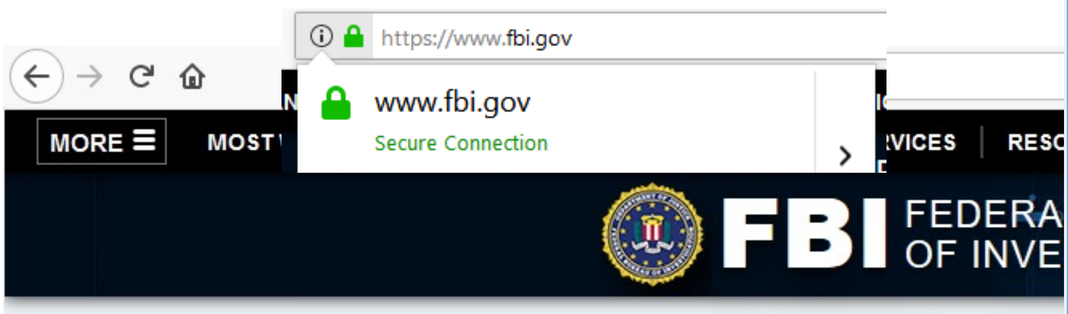
# Some additional complexity: CDNs

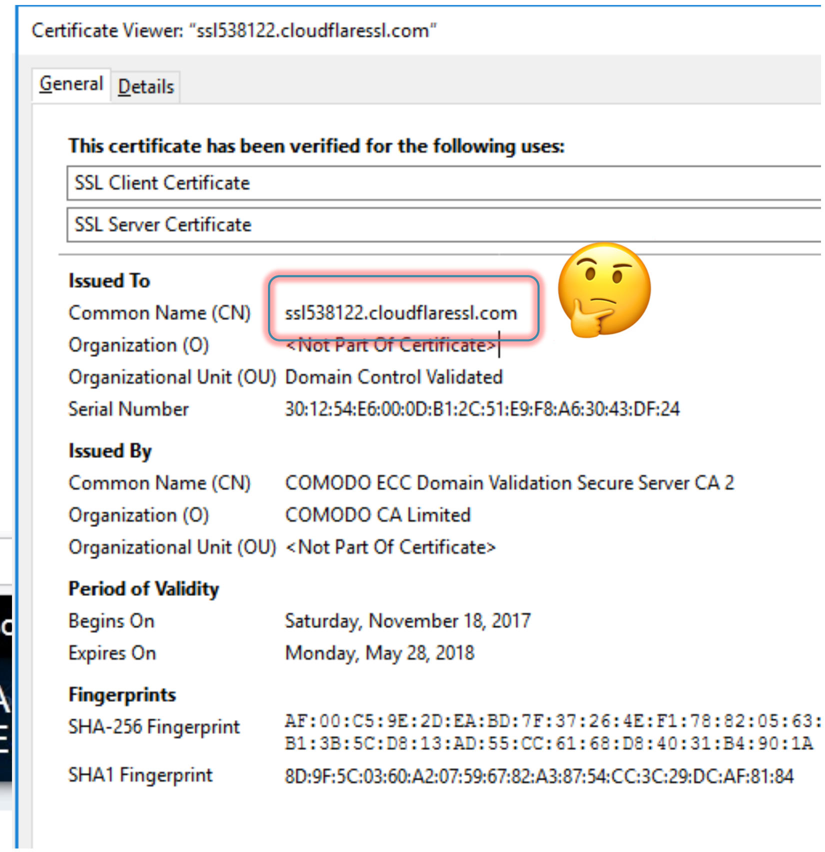
HTTPS is secured by TLS



# Some additional complexity: CDNs

HTTPS is secured by TLS



# 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

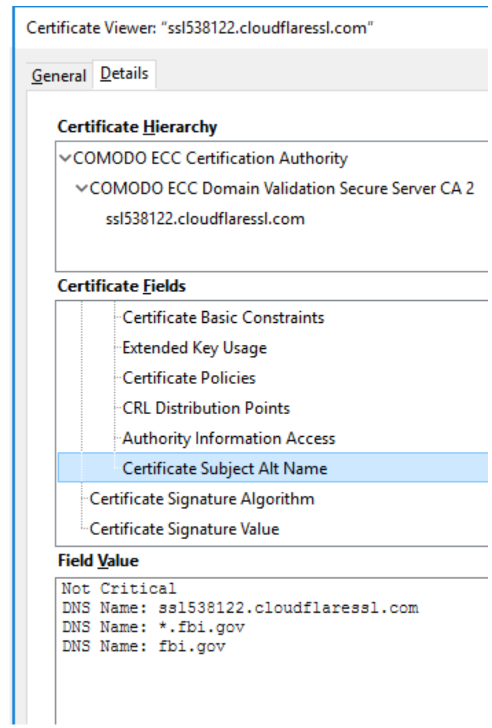
Cloudflare needs to convince Bob’s browser that it’s really FBI

# 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?

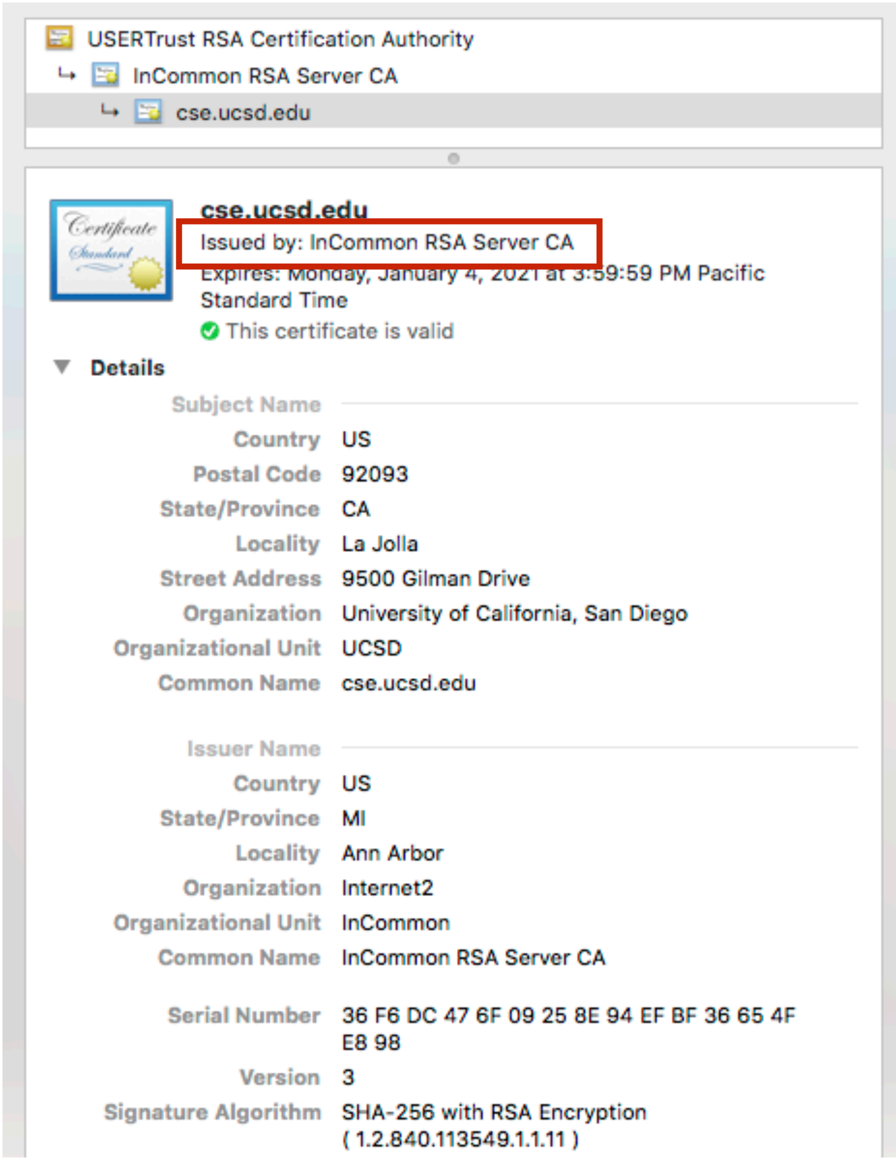


USERTrust RSA Certification Authority
↳ InCommon RSA Server CA
↳ cse.ucsd.edu

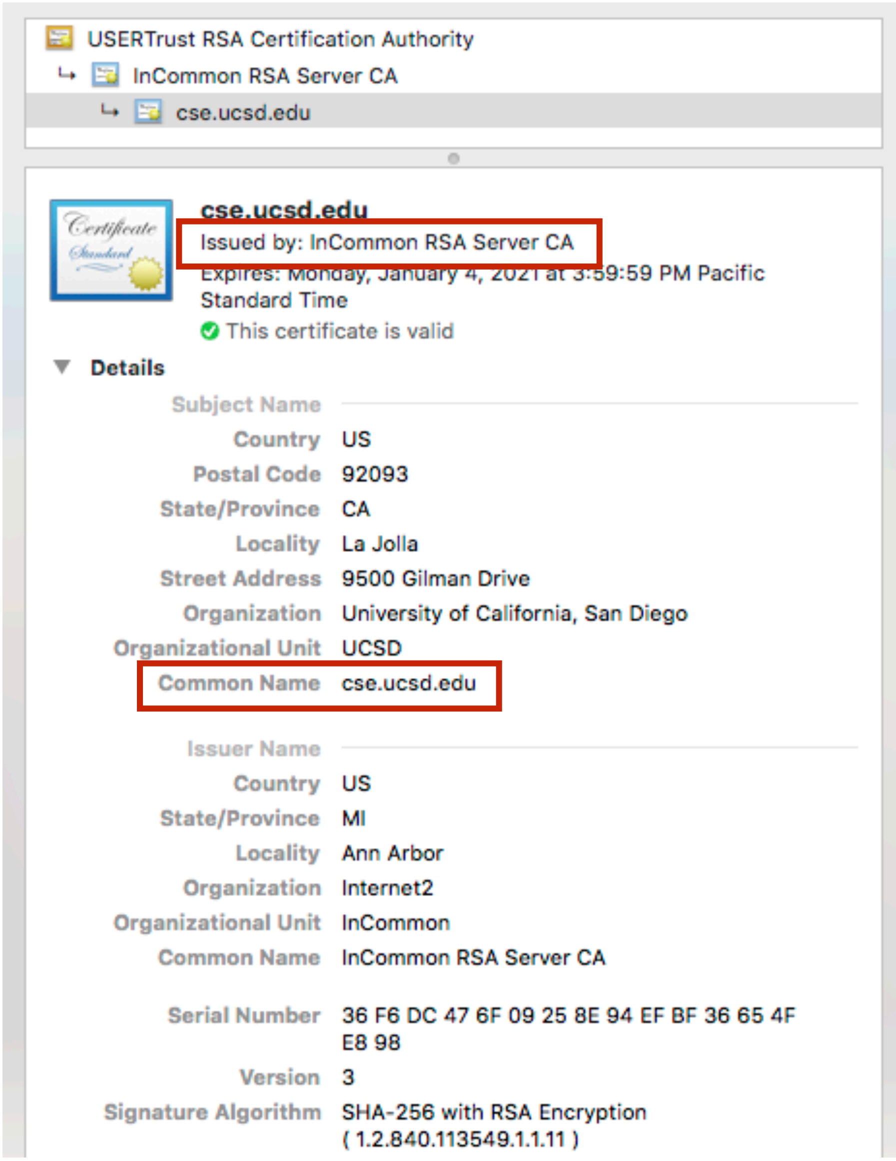
**cse.ucsd.edu**
Issued by: InCommon RSA Server CA
Expires: Monday, January 4, 2021 at 3:59:59 PM Pacific Standard Time
This certificate is valid

**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 |
| 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 |
| 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 |

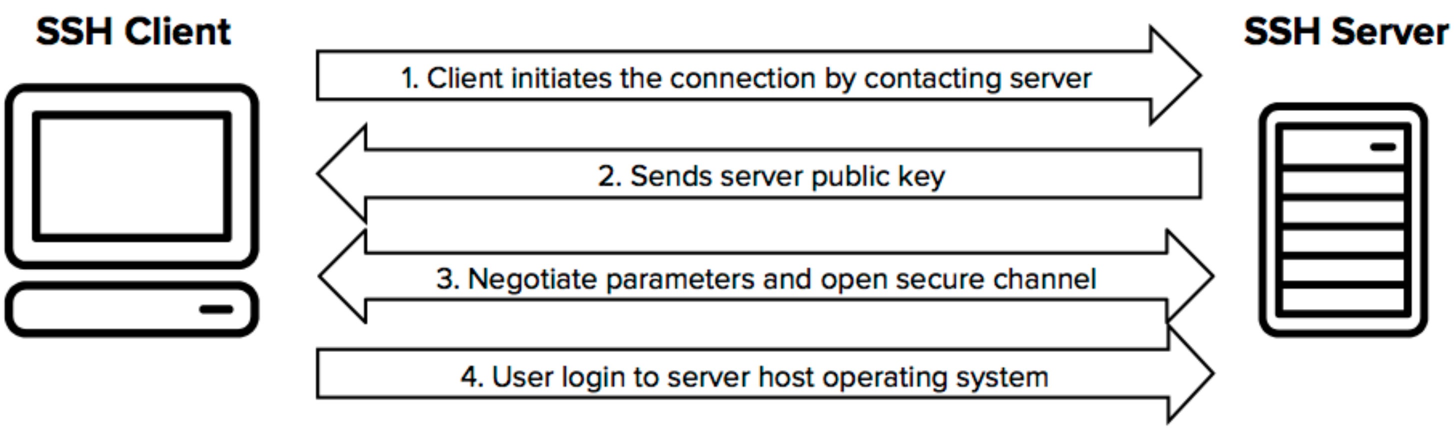
CSE’s pub key info

| | |
|---|---|
| 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 |
| 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 |
| 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."*



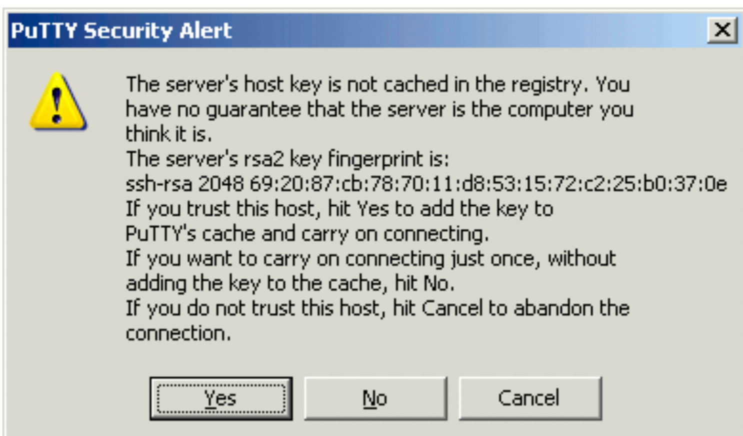

https://www.ssh.com/ssh/

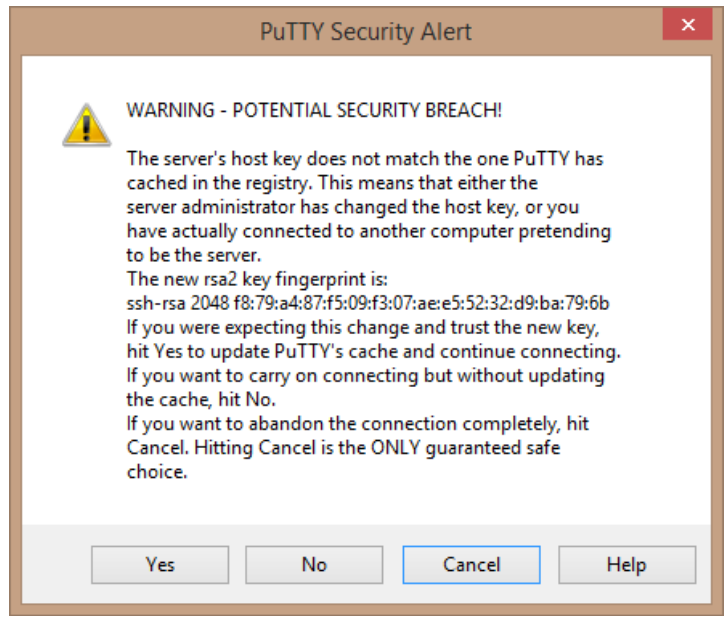
# Another approach: Secure Shell (SSH)

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 issueed 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

# Next time

Starting Web Security