



Certificates

Lecture 06

Software Security Engineering

Winter 2023

Thompson Rivers University

Public Key Problems

- public-key crypto lets us secure communication
 - confidentiality, integrity, authenticity, non-repudiation
- but it requires that the public keys are authentic
 - you still need an authentic channel for that
- this is a hard practical problem
 - you've never met Google before logging into G-Mail, you just somehow got its key

Alice and Bob

- Alice and Bob both have their own public and private keys
- Alice and Bob have never met
- Alice needs Bob's public key to encrypt
 - she asks Bob over an insecure channel
 - she gets a public Key
 - what can go wrong?

Alice needs a way to validate the key without any bits being exchanged over an authentic channel.

You cannot bootstrap trust. It has to start somewhere.

You cannot bootstrap trust. It has to start somewhere.

With a kernel of trust you can exchange the key.

You cannot bootstrap trust. It has to start somewhere.

With a kernel of trust you can exchange the key.

With the key you can exchange everything else.

Solution 1: trust on first use (TOFU)

Solution 1: trust on first use (TOFU)

This is actually used a lot and is called TOFU.

I get something that claims to be Bob's public key.

I get something that claims to be Bob's public key.

I assume it is and **only trust that one**

I get something that claims to be Bob's public key.

I assume it is and **only trust that one**

I am suspicious if it ever changes

I get something that claims to be Bob's public key.

I assume it is and **only trust that one**

I am suspicious if it ever changes

I am safe unless I was being attacked **at that first time.**

I get something that claims to be Bob's public key.

I assume it is and **only trust that one**

I am suspicious if it ever changes

I am safe unless I was being attacked **at that first time.**

I can always validate Bob's public key later if I meet Bob.

```
sina@sina:~$ ssh sina.keshvadi1@136.159.5.27
```

```
The authenticity of host '136.159.5.27 (136.159.5.27)' can't be established.  
ED25519 key fingerprint is SHA256:/JovPeFakPNhorFWgl/9qdFmGSH79/2LNxikVY3Ci/g.
```

```
This host key is known by the following other names/addresses:
```

```
~/.ssh/known_hosts:1: [hashed name]
```

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? 
```

Solution 2: centralized on-demand service

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Replay attacks?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Replay attacks?

DoS attacks?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Replay attacks?

DoS attacks?

Corrupt service?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Replay attacks?

DoS attacks?

Corrupt service?

Does this remind you of anything?

Solution 2: centralized on-demand service

Alice ask Service for Bob's key (or to validate it)

would this be practical?

MITM attacks?

Replay attacks?

DoS attacks?

Corrupt service?

Does this remind you of anything? **Trusted Third Party (TTP)?**

Certificates

- a statement about a public key
 - “this certifies that KEY XYZ belongs to Bob. Yours most sincerely,
Trent”
- Bob sends his public key to Trent over an authentic channel
- Trent prepares a document stating Bob owns the key
- Trent signs the document with Trent’s private key
- Trent appends this signature to the document and gives the result to Bob

Bob can show this to anyone without involving Trent!

Bob can show this to anyone without involving Trent!

Alice can verify this without asking Trent,
she only needs his public key!

Bob can show this to anyone without involving Trent!

Alice can verify this without asking Trent,
she only needs his public key!

Does this remind you of anything?

Bob can show this to anyone without involving Trent!

Alice can verify this without asking Trent,
she only needs his public key!

Does this remind you of anything? **Ticket?**

Protocol

- Alice has Trent's public key (assume for now)
- Alice contacts Bob
- Bob gives Alice the certificate signed by Trent
- Alice checks that the signature is valid using Trent's public key
- If Trent is honest, then that is Bob's public key

In practice, the document is called a **digital certificate**
or a **cert** for short

In practice, the document is called a **digital certificate**
or a **cert** for short

Trent is called a **certificate authority**
or a **CA** for short

what can go wrong?

what can go wrong?

Alice doesn't have authentic key for Trent
(either bad in the first place, or changed)

Alice is not able to validate the cert

what can go wrong?

Alice doesn't have authentic key for Trent
(either bad in the first place, or changed)

Eve pretended to be Bob and Trent gives her a "Bob" cert
(this is not a hypothetical risk, it has happened several times)

what can go wrong?

Alice doesn't have authentic key for Trent
(either bad in the first place, or changed)

Eve pretended to be Bob and Trent gives her a "Bob" cert
(it's one thing for everyone to know Trent,
another for Trent to know everyone)

What can go wrong?

What if Trent is Eve?

What can go wrong?

What if Trent is Eve?

What if Eve breaks into Trent's computers?

How do you know if the whole certificate system works?



https://

That's the user's entire interaction with the security.

That's the user's entire interaction with the security.

It's called a **security indicator** (LEAST SURPRISE)

That's the user's entire interaction with the security.

It means encryption is securing the connection
and the website provided a valid certificate.

That's the user's entire interaction with the security.

It means encryption is securing the connection
and the website provided a valid certificate.

What does that mean though?

That's the user's entire interaction with the security.

It means encryption is securing the connection
and the website provided a valid certificate.

What does that mean though?

Did Trent meet Bob face to face?

But even with the lock, what does it mean?

It means encryption is securing the connection
and the website provided a valid certificate.

What does that mean though?

Did Trent meet Bob face to face?

Who is Trent?

CA's Duties

- Bob claims that bob.com is his
- Bob wants to use PK as a public key for it
- What checks are required before issuing cert by CA?
 - Bob needs to prove he controls the PK and have private key
 - Bob need to prove he control the bob.com
 - (this is the basic level of certification. More later in this lecture)

Alice's Duties

- Bob sends Alice: “Bob-signed(Trent-signed(cert))”
 - cert claims “bob.com’s signing key is PK”
- What checks are required before trusting cert?
 - check Bob’s signature is valid using PK from the cert
 - trust the Trent’s signature is valid
 - bob.com is where she wants to go and this is not a cert for something else

Failing to check that cert is actually for Bob!

Failing to check that cert is actually for Bob!

Researchers discovered that poorly designed APIs used in SSL implementations failed to check the cert matched the sender.

Many critical non-browser software packages such as Amazon's EC2 Java library, Amazon's and PayPal's merchant SDKs, Trillian and AIM instant messaging software, popular integrated shopping cart software packages, Chase mobile banking software, and several Android applications and libraries.

SSL connections from these programs and many others are vulnerable to a man in the middle attack

Failing to check the cert!

```
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
err = sslRawVerify(...);

fail:    return err
```

Secure Coding - Use {} by default

Let's say that Bob's private key is stolen.

Let's say that Bob's private key is stolen.

What's the worst that can happen?

Let's say that Bob's private key is stolen.

What's the worst that can happen?

Eve is able to imposter Bob forever

A fake certificate can trick a system into thinking it is communicating with a trusted entity when it is actually communicating with an untrusted entity.

This can lead to the release of sensitive information, such as passwords or other confidential data, to the attacker.

It can also lead to the installation of malicious software on the system.

A fake certificate undermines the trust that is essential for secure communication and should be avoided at all costs.

Let's say that Bob's private key is stolen.

What's the worst that can happen?

Eve is able to imposter Bob forever

How can we stop this?

Certification Revocation

- revoke means no longer trust this cert
 - keys can get stolen, or suspected stolen
 - also Bob changes companies
 - Bob wants to use a new key instead
- certs are just a signed statement
- how to remove trust once issued?

Cert Revocation List

- CRL are lists of bad certs.
- periodically given out to parties, e.g., weekly
- can be pushed to parties or posted to specific place

Certificate Viewer: *.google.com

General **Details**

Certificate Hierarchy

- ▼ Builtin Object Token:GTS Root R1
 - ▼ GTS CA 1C3
 - *.google.com**

Certificate Fields

- Certificate Basic Constraints
- Certificate Subject Key ID
- Certification Authority Key ID
- Authority Information Access
- Certificate Subject Alternative Name
- Certificate Policies
- CRL Distribution Points**
- OID.1.3.6.1.4.1.11129.2.4.2
- Certificate Signature Algorithm

Field Value

Not Critical
URI: <http://crls.pki.goog/gts1c3/QqFxbi9M48c.crl>

Export...

Certs Expire

- gives upper bound on use of stolen key
- keeps cert authorities with customers
 - Certification Authorities charge a fee for their certificates
 - the cost varies based on the services
- stops **revocation lists** from growing forever

Certificate Viewer: *.google.com



General Details

Issued To

Common Name (CN)	*.google.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	GTS CA 1C3
Organization (O)	Google Trust Services LLC
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Monday, January 9, 2023 at 12:16:58 AM
Expires On	Monday, April 3, 2023 at 1:16:57 AM

Fingerprints

SHA-256 Fingerprint	15 96 F0 86 4C 59 46 C6 A5 59 B0 10 F6 AD 47 8D 1B EA C6 D2 4A D6 D4 33 CE 8F C0 A0 AE 41 09 6D
SHA-1 Fingerprint	B4 07 40 B9 3D 18 39 3B 42 C4 2A E8 EA 5E FA 14 2B 51 6D 4C

CRL Deltas

- instead of publishing the whole CRL, give updates (deltas)
- requires active involvement to keep up to date

Online Status Checking

- use an always online party check if a cert is valid
 - check done by Alice at the time of use
 - online certificate status protocol (OCSP)
- what does this cost?
 - Tom (new Certificate checking system) is involving
 - Tom could be your ISP

OCSP-stapling

- periodically get a signature from the online party with a timestamp
 - cert X is still not revoked at time Y
- check now done by Bob
 - CA load substantially reduced for popular sites
- does this remind you of another protocol?

What does OCSP provide that CRLs don't?

Short-lived Certs

- make all certs only valid for a week
- exposure time is bounded to this low amount
- need to contact the CA to get new certs

Short-lived certs seem equivalent to CRL and
OCSP-stapling but they differ in a failure condition.

How?

Short-lived certs seem equivalent to CRL and OCSP-stapling but they differ in a failure condition.

How?

CRL and OCSP requires a trusted third party but short-lived certs avoid this

Certs in Practice

- certs are used for TLS
 - transport layer security
 - this is the de facto means to secure web traffic
 - puts the S in HTTPS
 - S is for secure
 - topic of next lecture
- certs deliver a website's public key to a browser
 - authentic delivery of public key for Bob
 - creates authentic channel from Alice to Bob

Alice goes to bob.com and gets a cert
for a public key that the owner of
bob.com has the private key for.

For web, this is all done in the browser.

For web, this is all done in the browser.

The browser is responsible for checking if a cert is valid
by checking the fields, CRLs, etc.




https://



Since 2016, more web traffic is over HTTPS than HTTP



Since 2016, more web traffic is over HTTPS than HTTP
the lock has become more normal, and browsers are now
warning on insecure pages

example.com x

 **Your connection to this site is not secure**

You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers. [Learn more](#)

 Cookies 4 in use 

 Site settings 

Please log in to your account.

This connection is not secure. Logins entered here could be compromised.



Learn More

Password



p2146r2-1.pdf

File not downloaded: Potential security risk. — open-std.org — 09:38

Three types of validation for certs.

Domain Validated (DV)

- Bob gives a public key to the CA and claims bob.com
- sends an email to admin@bob.com
 - a challenge, e.g., random number to sign with key
- purported owner proves control over the domain by
 - posting DNS TXT records to bob.com
 - putting some random number on bob.com/ca challenge.html
- no proof that there's anyone named Bob related to it
 - could be a rogue employee with webmaster access
- can be fully automated

Organization Validated (OV)

- also checks a business/organization behind the key
- e.g., look up business in a public directory and call them
- exact practice depends on the CA's certificate practice statement
- this extra information then is part of cert
 - but the user still only sees the lock icon

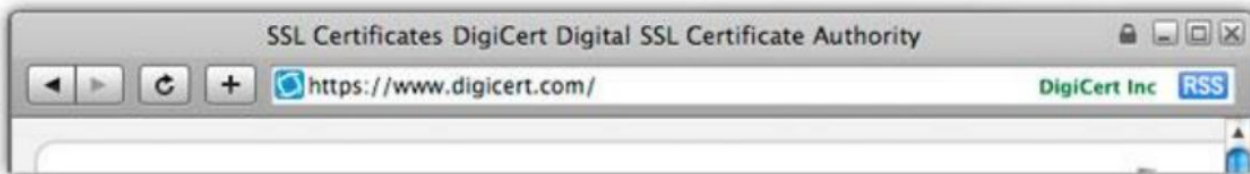
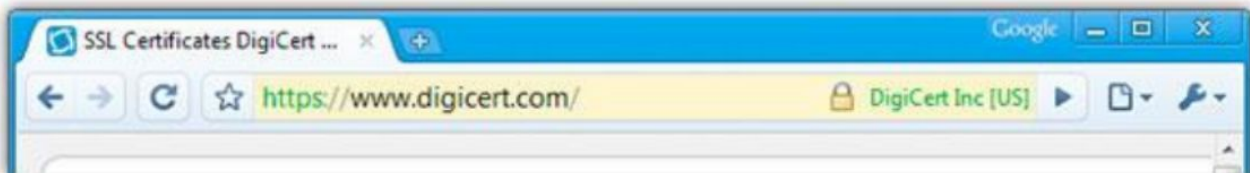
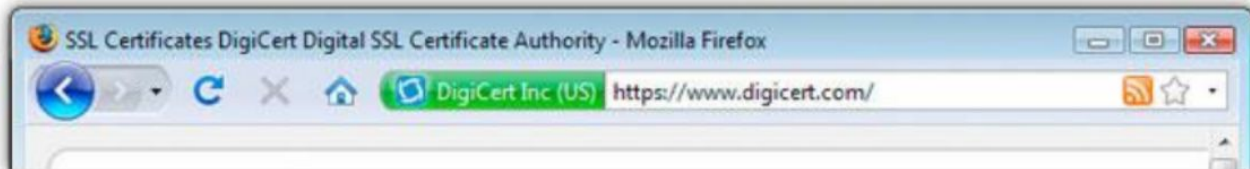
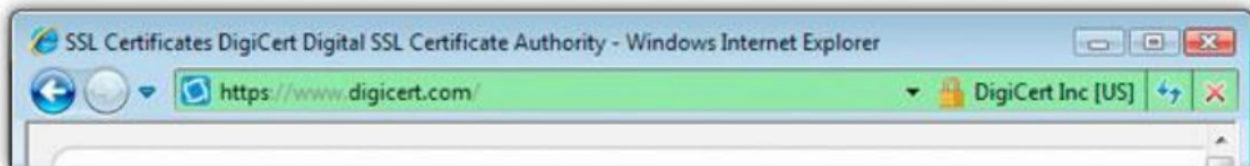
Extended Validation (EV)

- use of government database to confirm existence of legal entities named as Subject
- EV cert issuers are audited, have governance
 - certificate requests must be approved by a human lawyer
- motivated by low confidence DV certs that can be given to phishing websites
 - resulted in same visual experience as a legit site
 - e.g., lock icon, secure browser bar, etc.

ing Started {0 Cross Reference: /fr...

⋮

moz://a



Security

td.com

Connection is secure

Your information (for example, passwords or credit card numbers) is private when it is sent to this site.

[Learn more](#)

Certificate is valid

Issued to: The Toronto-Dominion Bank [CA]

Small Business

Commercial

Bank

America's Most Convenient Bank®



Extended Validation

- this has since stop
 - May 2018, Google removed it from Chrome
 - other browsers soon followed
- this seemed like a good idea, so why did it stop?

Extended Validation Drop Reason

- user studies and A/B testing which showed they were ineffective
 - users do not appear to make secure choices (such as not entering password or credit card information) when the UI is altered or removed
- interfered with the bias towards neutralization of HTTPS
 - secure should be the norm, and insecure is increasing being treated as hostile
- could be hacked with similar business names

Back in 2008

- Sotirov et al. collected 30,000 website certificates
 - 9,000 of them were signed using MD5 hash
 - we generally sign hashes of messages, not messages
 - 97% of those were issued by RapidSSL
 - others were FreeSSL, TrustCenter, RSA Data Security, Thawte, verisign.co.jp
- what is wrong with signing an MD5 hash?

MD5 has known collisions, since 2004.

MD5 has known collisions, since 2004.

A signature on one message means that all other messages with the same hash will appear to also be signed.

MD5 has known collisions, since 2004.

A signature on one message means that all other messages with the same hash will appear to also be signed.

Having a collision can suggest a technique to make more.

MD5 has known collisions, since 2004.

A signature on one message means that all other messages with the same hash will appear to also be signed.

Having a collision can suggest a technique to make more.

Collisions are typically not found by brute force alone.

**set by
the CA**

serial number	chosen prefix (difference)	serial number
validity period		validity period
real cert domain name		rogue cert domain name
real cert RSA key	Hash to the same MD5 value!	???
	collision bits (computed)	
X.509 extensions	identical bytes (copied from real cert)	X.509 extensions
signature		signature

Valid for both certificates!

Finding Collisions takes about 1–2 days
on a cluster of 200 PS3s



How do we trust Certificate Authorities?

You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
▼ AC Camerfirma S.A.	
Chambers of Commerce Root - 2008	Builtin Object Token
Global Chambersign Root - 2008	Builtin Object Token
▼ AC Camerfirma SA CIF A82743287	
Camerfirma Chambers of Commerce Root	Builtin Object Token
Camerfirma Global Chambersign Root	Builtin Object Token
▼ ACCV	
ACCVRAIZ1	Builtin Object Token
▼ Actalis S.p.A./03358520967	
Actalis Authentication Root CA	Builtin Object Token
▼ AddTrust AB	
PositiveSSL CA 2	Software Security Device
COMODO RSA Certification Authority	Software Security Device
COMODO ECC Certification Authority	Software Security Device
USERTrust RSA Certification Authority	Software Security Device
▼ AffirmTrust	
AffirmTrust Commercial	Builtin Object Token
AffirmTrust Networking	Builtin Object Token
AffirmTrust Premium	Builtin Object Token
AffirmTrust Premium ECC	Builtin Object Token
AffirmTrust Certificate Authority - OV1	Software Security Device
▼ Agencia Catalana de Certificacio (NIF Q-0801176-I)	
FC-ACC	Builtin Object Token

View...

Edit Trust...

Import...

Export...

Delete or Distrust...

OK

You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
> AC Camerfirma S.A.	
> AC Camerfirma SA CIF A82743287	
> ACCV	
> Actalis S.p.A./03358520967	
> AddTrust AB	
> AffirmTrust	
> Agencia Catalana de Certificacio (NIF Q-0801176-I)	
> Amazon	
> Atos	
> Autoridad de Certificacion Firmaprofesional CIF A62634068	
> Baltimore	
> Buypass AS-983163327	
> certSIGN	
> CERTSIGN SA	
> China Financial Certification Authority	
> Chunghwa Telecom Co., Ltd.	
> Comodo CA Limited	
> Cybertrust, Inc	
> D-Trust GmbH	
> Deutsche Telekom AG	
> DFN-Verein	
> Dhimyotis	
> DigiCert Inc	

View...

Edit Trust...

Import...

Export...

Delete or Distrust...

OK

Certificate Name

Security Device

ES

- › D-Trust GmbH
- › Deutsche Telekom AG
- › DFN-Verein
- › Dhimyotis
- › DigiCert Inc
- › Digital Signature Trust Co.
- › Disig a.s.
- › E-Tuğra EBG Bilişim Teknolojileri ve Hizmetleri A.Ş.
- › eMudhra Inc
- › eMudhra Technologies Limited
- › Entrust, Inc.
- › Entrust.net
- › Equifax
- › FNMT-RCM
- › GeoTrust Inc.
- › GlobalSign
- › GlobalSign nv-sa
- › GoDaddy.com, Inc.
- › Google Trust Services LLC
- › GTE Corporation
- › GUANG DONG CERTIFICATE AUTHORITY CO.,LTD.
- › Hellenic Academic and Research Institutions Cert. Authority
- ▼ Honk Kong Post

View Edit Trust Import Export Release Distrust

You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
> Hellenic Academic and Research Institutions Cert. Authority	
> Hongkong Post	
> IdenTrust	
> Internet Security Research Group	
> IZENPE S.A.	
> Japan Certification Services, Inc.	
> Krajowa Izba Rozliczeniowa S.A.	
> Microsec Ltd.	
> Microsoft Corporation	
> NetLock Kft.	
> Network Solutions L.L.C.	
> QuoVadis Limited	
> SECOM Trust Systems CO.,LTD.	
> SECOM Trust.net	
> SecureTrust Corporation	
> Sociedad Cameral de Certificación Digital - Certicámara S.A.	
> Sonera	
> SSL Corporation	
> Staat der Nederlanden	
> Starfield Technologies, Inc.	
> StartCom Ltd.	
> SwissSign AG	
√ Svmantec Corporation	

View

Edit Trust

Import

Export

Delete or Distrust

Certificate Name

Security Device

- > Staat der Nederlanden
- > Starfield Technologies, Inc.
- > StartCom Ltd.
- > SwissSign AG
- > Symantec Corporation
- > T-Systems Enterprise Services GmbH
- > TAIWAN-CA
- > TeliaSonera
- > thawte, Inc.
- > The Go Daddy Group, Inc.
- > The USERTRUST Network
- > TrustCor Systems S. de R.L.
- > Trustis Limited
- > Trustwave Holdings, Inc.
- > Türkiye Bilimsel ve Teknolojik Arastirma Kurumu - TUBITAK
- > UniTrust
- > Unizeto Sp. z o.o.
- > Unizeto Technologies S.A.
- > Verein zur Foerderung eines Deutschen Forschungsnetzes e. V.
- > VeriSign, Inc.
- > WISEKey
- > XRamp Security Services Inc

Subject Name

Country TR
Locality Gebze - Kocaeli
Organization Türkiye Bilimsel ve Teknolojik Arastirma Kurumu - TUBITAK
Organizational Unit Kamu Sertifikasyon Merkezi - Kamu SM
Common Name TUBITAK Kamu SM SSL Kok Sertifikasi - Surum 1

Issuer Name

Country TR
Locality Gebze - Kocaeli
Organization Türkiye Bilimsel ve Teknolojik Arastirma Kurumu - TUBITAK
Organizational Unit Kamu Sertifikasyon Merkezi - Kamu SM
Common Name TUBITAK Kamu SM SSL Kok Sertifikasi - Surum 1

Validity

Not Before 11/25/2013, 1:25:55 AM (Mountain Standard Time)
Not After 10/25/2043, 2:25:55 AM (Mountain Standard Time)

Public Key Info

Algorithm RSA
Key Size 2048
Exponent 65537
Modulus AF:75:30:33:AA:BB:6B:D3:99:2C:12:37:84:D9:8D:7B:97:80:D3:6E:E...

Any certificate signed by any of these certs is accepted as completely valid

Any certificate signed by any of these certs is accepted as
completely valid

i.e., gets the lock icon and no warnings.

Any certificate signed by any of these certs is accepted as
completely valid

i.e., gets the lock icon and no warnings.

There's no scale or proportion of trust for these CAs.

Any certificate signed by any of these certs is accepted as
completely valid

i.e., gets the lock icon and no warnings.

There's no scale or proportion of trust for these CAs.

What can go wrong?



Security Warning: Do you trust the Russian government?

Firefox has detected that your connection to this website is probably not secure. If you are attempting to access or transmit sensitive data, you should **stop** this task, and try again using a **different Internet connection**.

Firefox has detected a potential security problem while trying to access www.bankofamerica.com, a website visited at least 131 times in the past by persons using this computer.

In these previous browsing sessions, www.bankofamerica.com provided a security certificate verified by a company in the **United States**.

However, this website is now presenting a different security certificate verified by a company based in **Russia**.

If you do not trust the government of Russia with your private data, or think it unlikely that Bank of America would obtain a security certificate from a company based there, this could be a sign that someone is attempting to intercept your secure communications.

[Click here](#) to learn more about security certificates and this potentially risky situation.

If you trust the government of Russia and companies located there to protect your privacy and security, [click here](#) to accept this new certificate and continue with your visit to the site.

Get me out of here!

Source: "Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL" by Christopher Soghoian and Sid Stamm

The attacker who penetrated the Dutch CA DigiNotar last year had complete control of all eight of the company's certificate-issuing servers during the operation and he may also have issued some rogue certificates that have not yet been identified. The final report from a security company commissioned to investigate the DigiNotar attack shows that the compromise of the now-bankrupt certificate authority was much deeper than previously thought.

Iranian activists feel the chill as hacker taps into e-mails

BY SOMINI SENGUPTA

He claims to be 21 years old, a student of software engineering in Tehran who reveres Ayatollah Ali Khamenei and despises dissidents in his country.

He sneaked into the computer systems of a security firm on the outskirts of Amsterdam. He created fake credentials that could allow someone to spy on Internet connections that appeared to be secure. He then shared that bounty with people he declines to identify.

The fruits of his labor are believed to have been used to tap into the online communications of as many as 300,000 unsuspecting Iranians this summer. What is more, he punched a hole in an

online security mechanism that is trusted by Internet users all over the world.

Comodohacker, as he calls himself, insists that he acted on his own and is unperturbed by the notion that his work might have been used to spy on anti-government compatriots.

"I'm totally independent," he said in an e-mail exchange with The New York Times. "I just share my findings with some people in Iran. They are free to do anything they want with my findings and things I share with them, but I'm not responsible."

In the annals of Internet attacks, this is most likely to go down as a moment of reckoning. For activists, it shows the
HACKER, PAGE 17

Neglect by Certificate Authorities can have a significant impact on
human lives!

Wait!

IT GETS BETTER!

That enormous list of CAs are known as root CAs.

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

So Turktrust signs for someone you never heard of

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

So Turktrust signs for someone you never heard of
who signs for someone else

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

So Turktrust signs for someone you never heard of

who signs for someone else

who signs for someone else

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

So Turktrust signs for someone you never heard of

who signs for someone else

who signs for someone else

who signs that some random public key you've never seen

before is Bob's.

That enormous list of CAs are known as root CAs.

CAs sign certificates for other CAs.

So Turktrust signs for someone you never heard of

who signs for someone else

who signs for someone else

who signs that some random public key you've never seen
before is Bob's.

And it gets the lock icon.

Demo

```
> openssl s_client -showcerts -connect tru.ca:443
```

TURKTRUST, a certificate authority in Mozilla's root program, mis-issued two intermediate certificates to customers. TURKTRUST has scanned their certificate database and log files and confirmed that the mistake was made for only two certificates.

TURKTRUST, a certificate authority in Mozilla's root program, mis-issued two intermediate certificates to customers. TURKTRUST has scanned their certificate database and log files and confirmed that the mistake was made for only two certificates.

Mozilla is actively revoking trust for the two mis-issued certificates which will be released to all supported versions of Firefox in the next update.

TURKTRUST accidentally issued **intermediary CA** certs.

TURKTRUST accidentally issued **intermediary CA** certs.

Those are the ones in the middle, and are
just as good as the root.

TURKTRUST accidentally issued **intermediary CA** certs.

Those are the ones in the middle, and are
just as good as the root.

Just because a country doesn't have a root cert,
doesn't mean they don't have an intermediate one.

TURKTRUST accidentally issued **intermediary CA** certs.

Those are the ones in the middle, and are
just as good as the root.

Just because a country doesn't have a root cert,
doesn't mean they don't have an intermediate one.

There's thousands of intermediaries from
non-democracies and private companies
(including defense contractors)

Also, the MD5 collision issue can be used to create
intermediary CA certs!

serial number	chosen prefix (difference)	rogue CA cert
validity period		rogue CA RSA key
real cert domain name		rogue CA X.509 extensions ← CA bit!
real cert RSA key	collision bits (computed)	Netscape Comment Extension (contents ignored by browsers)
X.509 extensions	identical bytes (copied from real cert)	signature
signature		signature

This means that it is a master key!

This means that it is a master key!

A network attacker to easily forge fake
certificates for any website!

This means that it is a master key!

A network attacker to easily forge fake
certificates for any website!

Users will get wrong public key and
not have any indication something is wrong.

if an attacker become a CA,
it can break everyone access to the Internet
it can MITM attack everyone

if an attacker become a CA,
it can break everyone access to the Internet
it can MITM attack everyone

All it takes is someone accidentally make you a CA,
or use MD5 collision vulnerability



https://

Certificate



General

Details

Certification Path

Certification path



Equifax Secure Global eBusiness CA-1



MD5 Collisions Inc. (<http://www.phreedom.org/md5>)



127.0.0.1

Certificate



General | Details | Certification Path

Certification path

- Equifax Secure Global eBusiness CA-1
 - MDS Collisions Inc. (<http://www.phreedom.org/md5>)
 - `i.broke.the.internet.and.all.i.got.was.this.t-shirt.phreedom`

[View Certificate](#)

Certificate status:

This certificate is OK.

OK

Browsers trust too many CA.

The security of HTTPS is only as strong as the practices of the least trustworthy/competent CA.

WEAKEST LINK

Fake certs is probably the most practical
way to break Internet security but...

Fake certs is probably the most practical
way to break Internet security but...
it is clear if the attack gets done.

Fake certs is probably the most practical
way to break Internet security but...

it is clear if the attack gets done.

Public key signatures provided **non-repudiability**
so if I sign a bad cert I can't undo it.

Fake certs is probably the most practical way to break Internet security but...

it is clear if the attack gets done.

Public key signatures provided **non-repudiability** so if I sign a bad cert I can't undo it.

If I'm the kind of CA that gives out bad certs then I'll stop being in the CA club.

Certificate Transparency (CT)

After Diginotar, Google employees wanted to create an open source framework for detecting mis-issued certificates.

After Diginotar, Google employees wanted to create an open source framework for detecting mis-issued certificates.

idea: log all new certificates from a CA

System was voluntary at first.

System was voluntary at first.

In 2015, Chrome required CT
logging for all new EV certs

System was voluntary at first.

In 2015, Chrome required CT
logging for all new EV certs

i.e., would reject cert if
it did not appear in logs.

System was voluntary at first.

In 2015, Chrome required CT logging for all new EV certs

i.e., would reject cert if it did not appear in logs.

In 2016, required CT for all certs from Symantec (Norton) ([Link](#))

System was voluntary at first.

In 2015, Chrome required CT logging for all new EV certs

i.e., would reject cert if it did not appear in logs.

In 2016, required CT for all certs from Symantec (Norton)

(they had issued 187 certificates without the domain owner's knowledge)

System was voluntary at first.

In 2015, Chrome required CT logging for all new EV certs

i.e., would reject cert if it did not appear in logs.

In 2016, required CT for all certs from Symantec (Norton)

(they had issued 187 certificates without the domain owner's knowledge)

In 2018, all certs.

View the logs: <https://crt.sh/>

How do we get certs?

- pay for one of the trusted authorities to give you one.
- use a **self-signed cert**
 - “Bob’s public key is XXX signed by XXX”
 - only for backwards compatibility
 - you sign your key with your own key
 - still not an authentic channel but what does it stop?



Secure Connection Failed

www.riseup.net uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.
The certificate is only valid for admin.riseup.net

(Error code: sec_error_unknown_issuer)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

[Or you can add an exception...](#)




There is a problem with this website's security certificate.

The security certificate presented by this website was not issued by a trusted certificate authority.

The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

-  [Click here to close this webpage.](#)
-  [Continue to this website \(not recommended\).](#)
-  [More information](#)



The site's security certificate is not trusted!

You attempted to reach **172.20.0.1**, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.

You should not proceed, **especially** if you have never seen this warning before for this site.


Proceed anyway

Back to safety

▶ [Help me understand](#)

Insecure Connection

https://127.0.0.1:8000

 Your connection is not secure

The owner of 127.0.0.1 has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

[Learn more...](#)

[Go Back](#) [Advanced](#)

Report errors like this to help Mozilla identify and block malicious sites

What is the trust model being used for circumvention?

This alarm bell design is good, but it incentivizes **not using security** because not using security generally had no alarm bells! (Think about threat model.)

It should be as hard or worse to use insecure sites.

It should be as hard or worse to use insecure sites.

Best case of a self-signed cert: it's the real cert.

It should be as hard or worse to use insecure sites.

Best case of a self-signed cert: it's the real cert.

Worst case of a self-signed cert: not using security.

It should be as hard or worse to use insecure sites.

Best case of a self-signed cert: it's the real cert.

Worst case of a self-signed cert: not using security.

Fake cert means you are being actively man-in-the-middle.

It should be as hard or worse to use insecure sites.

Best case of a self-signed cert: it's the real cert.

Worst case of a self-signed cert: not using security.

Fake cert means you are being actively man-in-the-middle.

No cert means any passive attacker can read your traffic.

It should be as hard or worse to use insecure sites.

Best case of a self-signed cert: it's the real cert.

Worst case of a self-signed cert: not using security.

Fake cert means you are being actively man-in-the-middle.

No cert means any passive attacker can read your traffic.

as well as actively modify!



LINUX FOUNDATION COLLABORATIVE PROJECTS

[Documentation](#)

[Get Help](#)

[Donate](#) ▼

[About Us](#) ▼

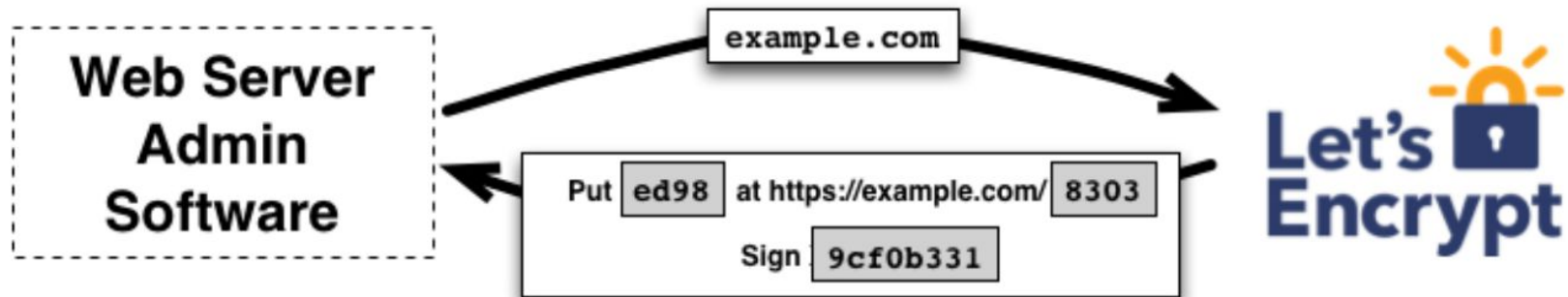
Let's Encrypt is a **free, automated, and open**
Certificate Authority.

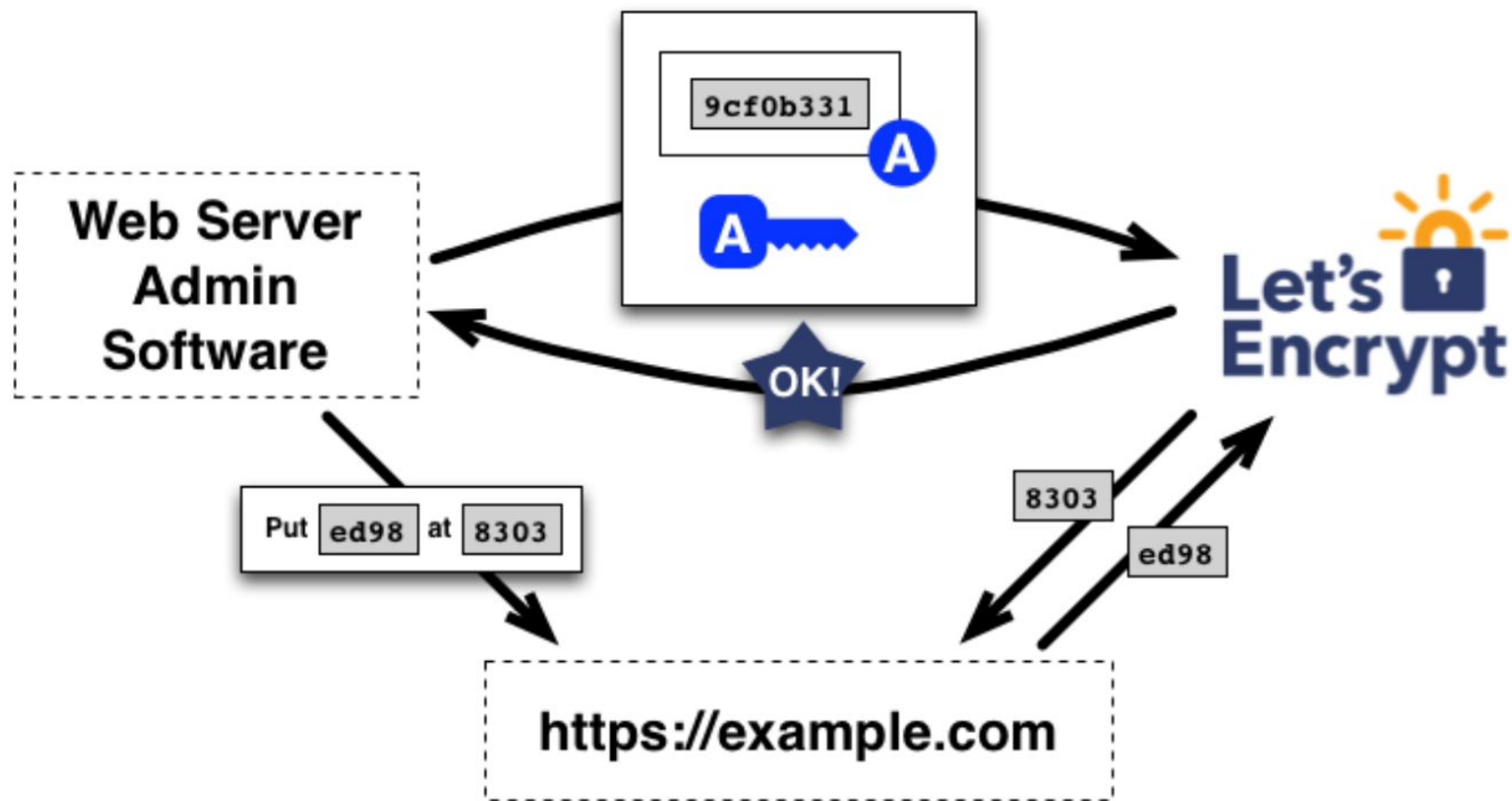
[Get Started](#)

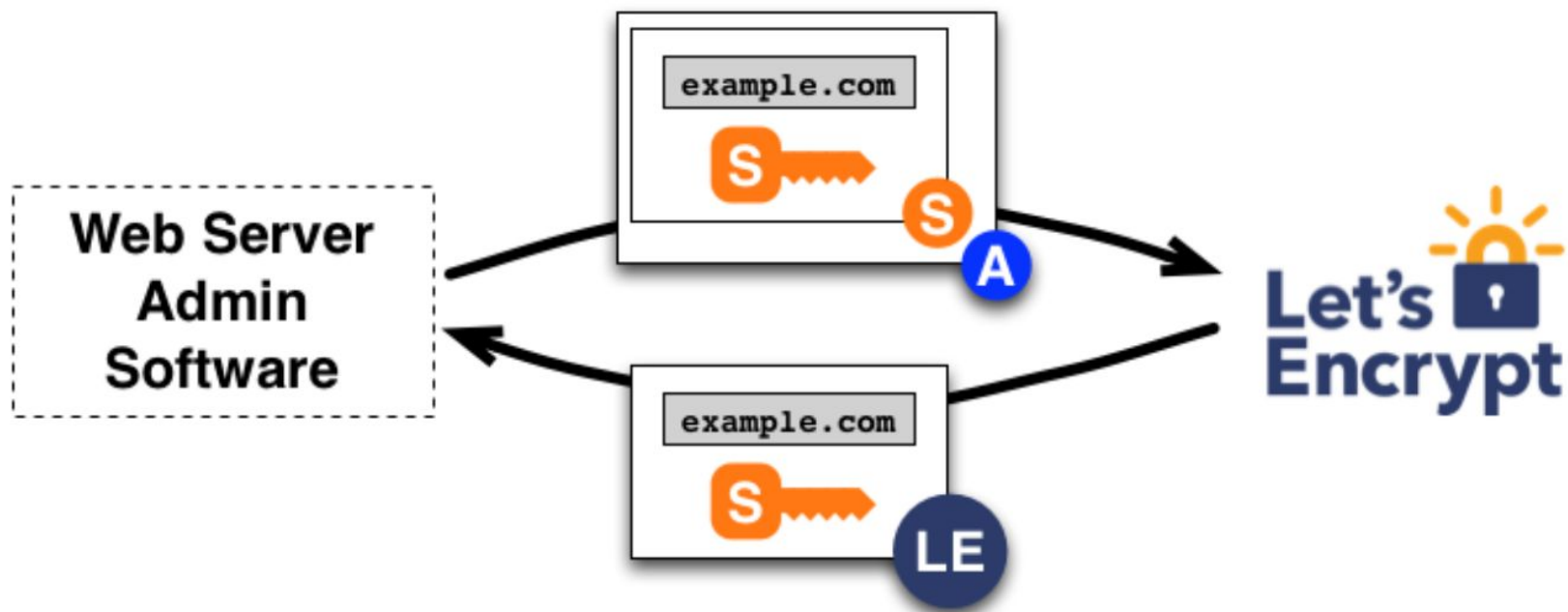
[Donate](#)

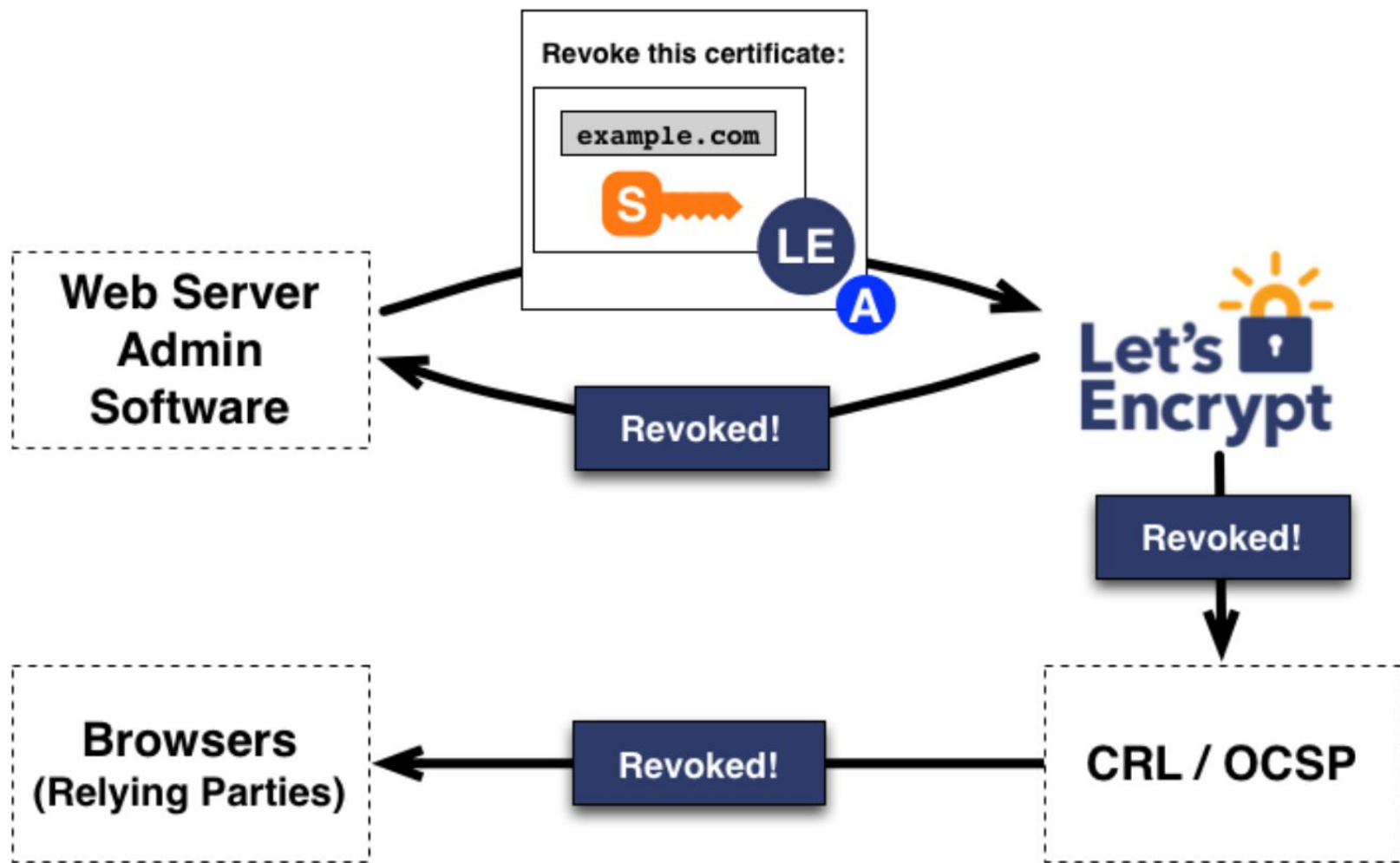
Let's Encrypt

- free automated open cert signing
 - only does DV, not OV or EV
 - supported by donations and volunteers
- allows anyone with just a webpage to have a nice signed cert
 - browsers trust the letsencrypt cert
 - avoids the warning alarms for self signed certs
 - avoids not using encryption









Let's Encrypt

- started in 2014 by EFF and backed by Akamai, Google, Facebook, Mozilla, and more
- has now signed 2.1 B certs for 265 million unique domains (2021)
 - largest certificate issuer in the world
- 83% of all firefox traffic in 2021 is HTTPS (secured)
 - it was 67% in 2017
 - it was 25% in 2013
- it used to be hard and expensive to get a cert

