

Security

Geoff Huston
Chief Scientist, APNIC



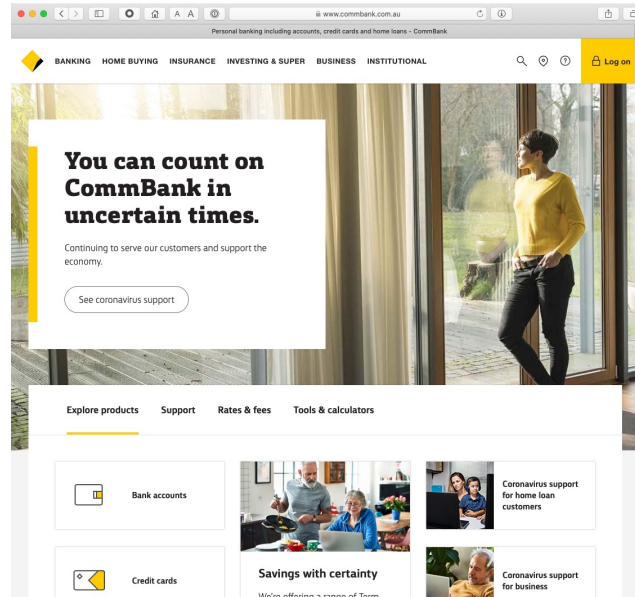
~~Security~~

insecurity!

Geoff Huston
Chief Scientist, APNIC



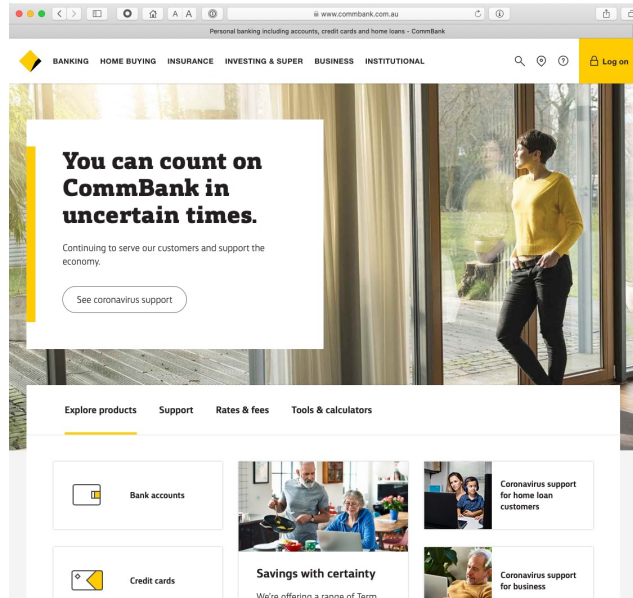
Which Bank?



Let's start with a simple example:

Why should you pass your account and password to this web site? It might look like your bank, but frankly it could just as easily be a fraudulent site intended to steal your banking credentials. Why should you trust what you see on the screen?

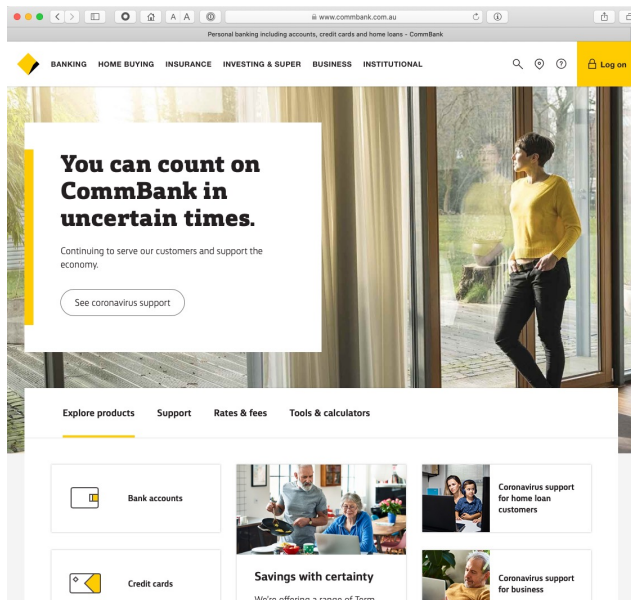
Which Bank? My Bank!



Ok – its not a random example. It's the online bank I use! But the same question is still there. Why should I trust this web page?

Which Bank? My Bank!

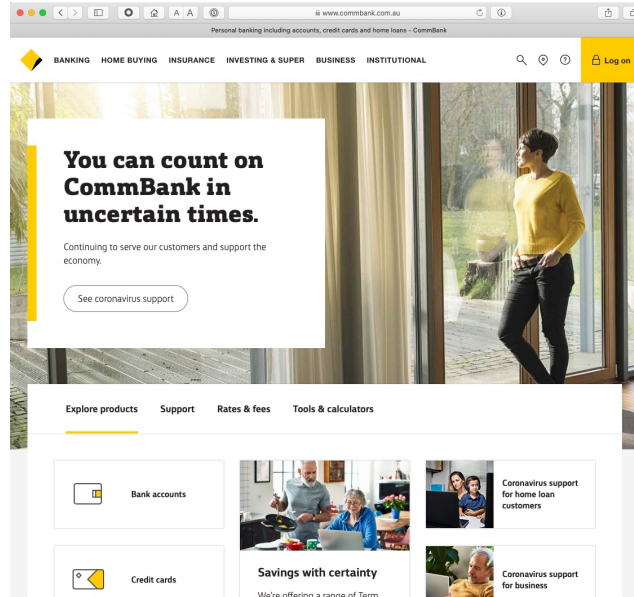
i hope!



Security on the Internet

How do you know that you are really going to where you thought you were going to?

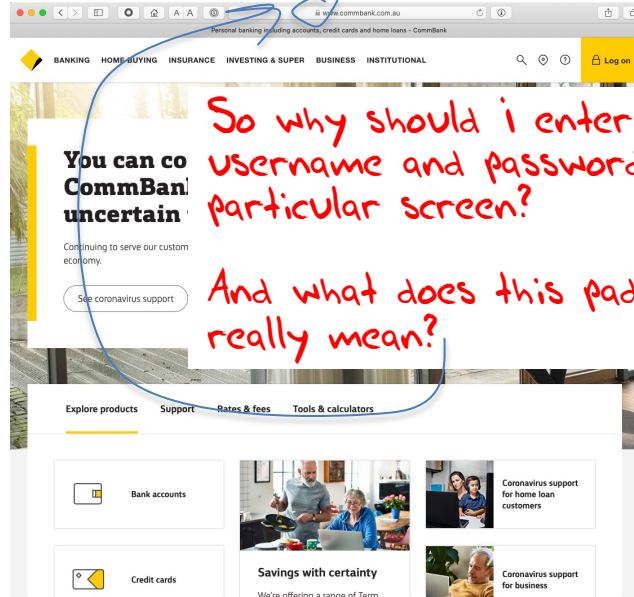
its trivial to mock up a web page to look like another



Security on the Internet

How do you know that you are really going to where you thought you were going to?

its trivial to mock up a web page to look like another



So why should i enter my username and password into this particular screen?

And what does this padlock icon really mean?

Opening the Connection: First Steps



Client:

DNS Query:

www.commbank.com.au?



DNS Response:

23.215.58.96

TCP Session:

TCP Connect 23.215.58.96, port 443



Hang on...

```
$ dig -x 23.215.58.96 +short
```

```
a23-215-58-96.deploy.static.akamaitechnologies.com.
```

Hang on...

```
$ dig -x 23.215.58.96 +short  
a23-215-58-96.deploy.static.akamaitechnologies.com.
```

That's **not** an IP addresses that was allocated to the Commonwealth Bank!

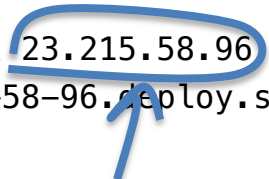
The Commonwealth Bank of Australia has the address blocks

140.168.0.0 - 140.168.255.255 and

203.17.185.0 - 203.17.185.255

Hang on...

```
$ dig -x 23.215.58.96 +short  
a23-215-58-96.deploy.static.akamaitechnologies.com.
```



That's an Akamai IP address

And I'm NOT a customer of the Internet Bank of Akamai!

Why should my browser trust that 23.215.58.96 is really the authentic web site for the Commonwealth Bank of Australia, and not some dastardly evil scam designed to steal my passwords and my money?

And why should I trust my browser?

The major question...

How does my browser tell the difference between an intended truth and a dastardly lie?

Public Key Cryptography

Pick a **pair** of keys such that:

- Messages encoded with one key can only be decoded with the other key
- Knowledge of the value of one key does not infer the value of the other key
- Make one key **public**, and keep the other a closely guarded **private** secret



This is important

So I will repeat it:

- Using public/private key cryptography requires a pair of keys (A,B) such that:
 - Anything encrypted using key A can ONLY be decrypted using key B, and no other key
 - Anything encrypted using key B can ONLY be decrypted using key A, and no other key
 - Knowing the value of one key WILL NOT let you work out the value of the other key anytime soon!

This form of asymmetric cryptography lies at the heart of the Internet's security framework

Public/Private Key Pairs

If I have a copy of your PUBLIC key, and you encrypt a message with your PRIVATE key, and I can decrypt the message using your public key

- I know no one has tampered with your original message
- I am confident that no one else has seen the contents of the message while it was passed through the network
- And I know it was you that sent it.
- And you can't deny it.

Public Key Certificates

But how do I know this is YOUR public key?

— And not the public key of some dastardly evil agent pretending to be you?

- I don't know you
- I've never met you
- So I have absolutely no clue if this public key value is yours or not!

Public Key Certificates

What if I 'trust' an intermediary*?

- Who has contacted you and validated your identity and conducted a 'proof of possession' test that you have control of a private key that matches your public key
- Then if the intermediary signs an attestation that this is your public key (with their private key) then I would be able to trust this public key
- This 'attestation' takes the form of a "public key certificate"

** If you have ever used "public notaries" to validate a document, then this is a digital equivalent*

Entrust Root Certification Authority - G2

Entrust Certification Authority - L1M

www.commbank.com.au



www.commbank.com.au

Issued by: Entrust Certification Authority - L1M

Expires: Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

✓ This certificate is valid

> Trust

> Details

Subject Name

Country or Region AU

County New South Wales

Locality Sydney

Inc. Country/Region AU

Organisation Commonwealth Bank of Australia

Business Category Private Organization

Serial Number 48 123 123 124

Common Name www.commbank.com.au

Issuer Name

Country or Region US

Organisation Entrust, Inc.

Organisational Unit See www.entrust.net/legal-terms

Organisational Unit (c) 2014 Entrust, Inc. - for authorized use only

Common Name Entrust Certification Authority - L1M

Serial Number 24 F5 40 B3 F7 9F 29 57 72 A0 F1 1C 6F 3D E7 AB

Version 3

Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)

Parameters None

Not Valid Before Wednesday, 30 March 2022 at 10:59:12 am Australian Eastern Daylight Time

Not Valid After Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Public Key 256 bytes: BF 7E 21 BA 6C E0 A1 9D ...

Exponent 65537

Key Size 2,048 bits

Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes: C3 28 89 A4 13 51 B0 8A ...

I trust that this is the web site of the Commonwealth Bank because I used the Commonwealth Bank's public key to set up the encrypted connection to the server.

And I can trust that this is the commonwealth Bank's public key because I trust that Entrust has performed a number of checks before issuing a public key certificate for this public key

And another example

- Lets take www.apnic.net and look at that certificate

← → ↻ 🏠

🔒 apnic.net

📄 ☆ 6% 🌐 🇸🇬 📄 📄 📄 📄 📄 📄 📄 📄

WHOIS & WEBSITE 🔍 ADVANCED WHOIS MAKE A PAYMENT LOGIN 📄

APNIC

Get IP ▾ Manage IP ▾ Training ▾ Events ▾ Insights ▾ Community ▾ Blog Help Centre About ▾ Contact

Your IP address: 2001:8003:1dec:da00:1820:acf8:4d4c:b5ed

GET IP ADDRESSES

TRANSFER IP ADDRESSES

GO IPv6

PARTICIPATE

WHAT IS APNIC?

Feedback

APNIC is the Regional Internet Registry administering IP addresses for the Asia Pacific

Events

Vulnerability Reporting Program on a Shoestr...

Tech matters

Homepage highlight



And another

- Let's look at my own web site, with its certificate issued by Let's Encrypt



ISP Articles Papers

Recent Articles

The Path to Reso

MAY 2022

Using the DNS without directly
an approach that is totally ali
might be useful to ask: How d
resolverless form of DNS nam
to whom does it make sense?

Are we there yet?

MAY 2022

This transition to IPv6 has bee
there was any urgency that w
prospect of IPv4 address exha
exhaustion for a decade now.
question: How much longer is
[More...](#)

Using LEOs and GEOs

APRIL 2022

ISRG Root X1

R3

potaroo.net

potaroo.net

Issued by: R3

Expires: Tuesday, 23 August 2022 at 9:35:20 am Australian Eastern Standard Time

✓ This certificate is valid

> Trust

▼ Details

Subject Name

Common Name potaroo.net

Issuer Name

Country or Region US

Organisation Let's Encrypt

Common Name R3

Serial Number 03 3B A0 FB 2C 1D B5 D0 87 0B CF BE 24 69 5A 20 A8 D4

Version 3

Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Not Valid Before Wednesday, 23 August 2022 at 9:35:21 am Australian Eastern Standard Time

Not Valid After Wednesday, 23 August 2022 at 9:35:21 am Australian Eastern Standard Time

Public Key

Algorithm RSA Encryption (2.840.113549.1.1.1)

Parameters None

Public Key 256 bytes: BF 24 1A 56 39 86 01 30 ...

Exponent 65537

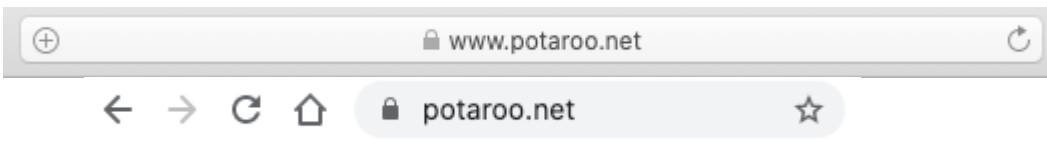
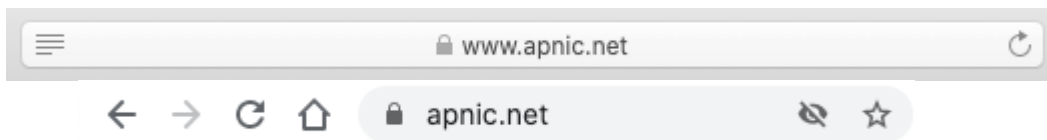
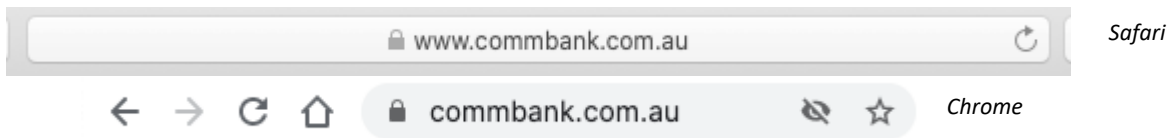
Key Size 2,048 bits

Key Usage Encrypt, Verify, Wrap, Derive

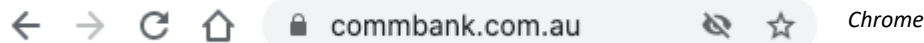
Signature 256 bytes: 26 E3 D0 AE 4F A9 64 7F ...

This certificate binds a public key to a domain name without any attestation to the identity of the name "holder"

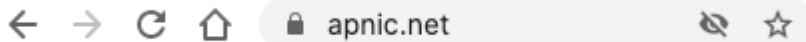
Spot the Difference



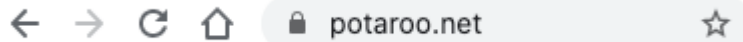
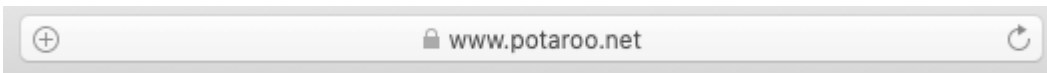
Spot the Difference



This web site's certificate was issued to an organisation called the "Commonwealth Bank of Australia" located in Sydney, Australia



This web site's certificate was issued to "Cloudflare Inc" located in San Francisco, USA!!



This web site's certificate says *nothing* about the entity that holds the public key associated with this domain

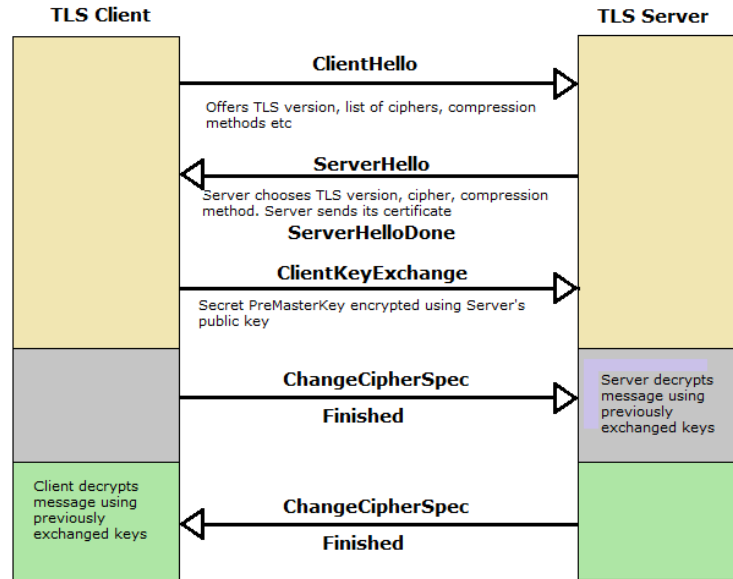
Spot the Difference

- The certification processes taken to issue the certificate were different in each of these cases.
 - One confirmed the identity of the public key holder as well as their association with the domain name
 - The second used a proxy agent and there is no association between the entity domain name that is certified here and the proxy agent
 - The third simply associates a public key with a domain name without any form of identification of the holder of the domain name
- They all have different levels of trustworthiness, yet they all display to the user in exactly the same way
 - Because when we tried to differentiate these different levels of trust (such as painting the padlock icon in green) nobody understood what was going on and nobody cared anyway!

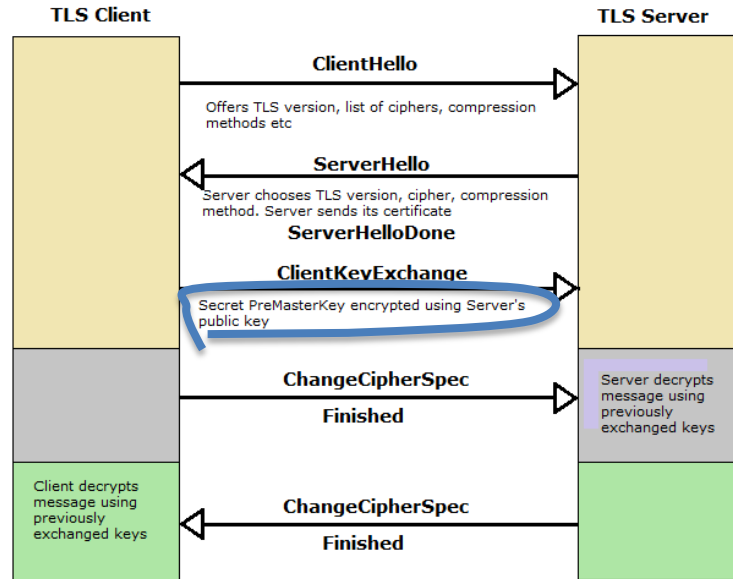
Moving on...

- Ok, so the certificate system is a mess, but TLS still works, right?
- So lets look at the way TLS sets of a secure session

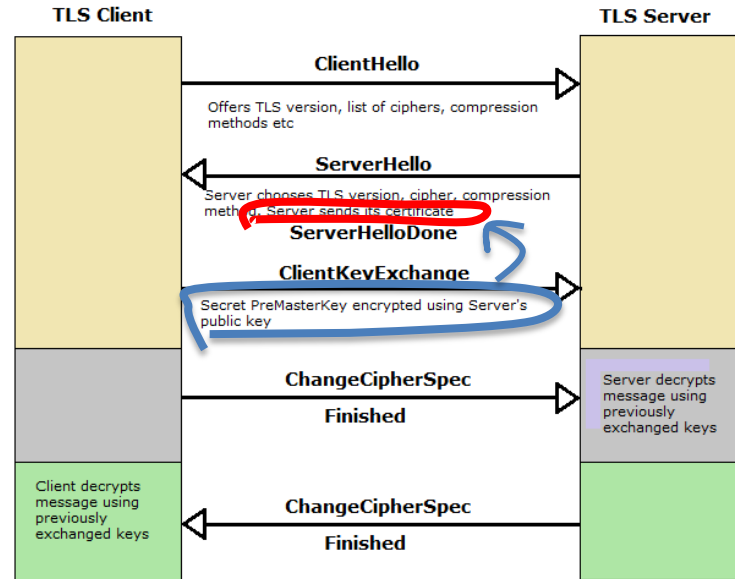
Secure Connections using TLS



Secure Connections using TLS



Secure Connections using TLS



Entrust Root Certification Authority - G2

Entrust Certification Authority - L1M

www.commbank.com.au



www.commbank.com.au

Issued by: Entrust Certification Authority - L1M

Expires: Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

✓ This certificate is valid

> Trust

▼ Details

Subject Name

Country or Region AU

County New South Wales

Locality Sydney

Inc. Country/Region AU

Organisation Commonwealth Bank of Australia

Business Category Private Organization

Serial Number 48 123 123 124

Common Name www.commbank.com.au

Issuer Name

Country or Region US

Organisation Entrust, Inc.

Organisational Unit See www.entrust.net/legal-terms

Organisational Unit (c) 2014 Entrust, Inc. - for authorized use only

Common Name Entrust Certification Authority - L1M

Serial Number 24 F5 40 B3 F7 9F 29 57 72 A0 F1 1C 6F 3D E7 AB

Version 3

Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Not Valid Before Wednesday, 30 March 2022 at 10:59:12 am Australian Eastern Daylight Time

Not Valid After Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

Parameters None

Public Key 256 bytes: BF 7E 21 BA 6C E0 A1 9D ...

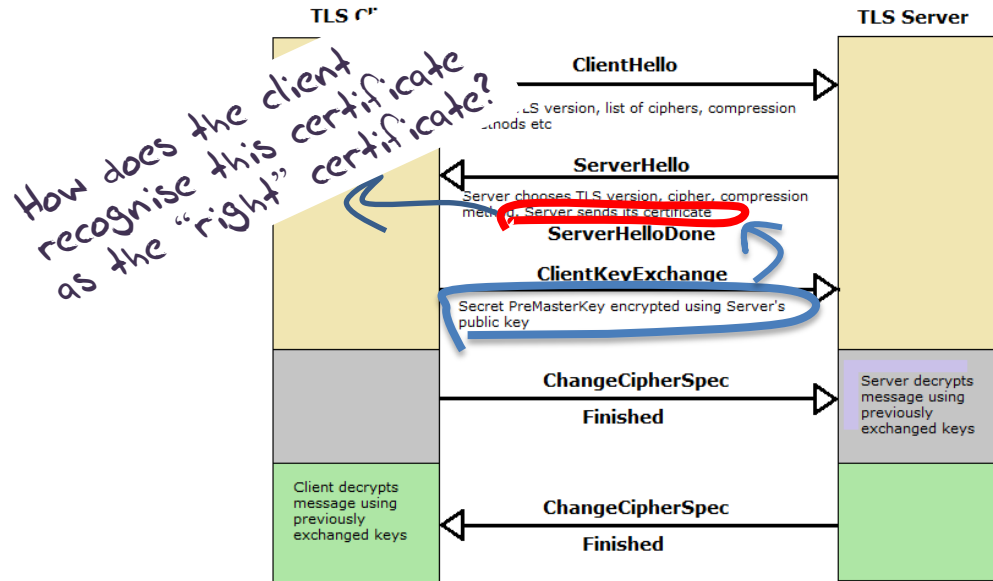
Exponent 65537

Key Size 2,048 bits

Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes: C3 28 89 A4 13 51 B0 8A ...

Secure Connections using TLS



Entrust Root Certification Authority - G2

Entrust Certification Authority - L1M

www.commbank.com.au



www.commbank.com.au

Issued by Entrust Certification Authority - L1M

Expires: Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

✓ This certificate is valid

> Trust

✓ Details

Subject Name
Country or Region AU
Country New South Wales
Locality Sydney
Inc. Country/Region AU
Organisation Commonwealth Bank of Australia
Business Category Private Organization
Serial Number 48 123 123 124
Common Name www.commbank.com.au

Issuer Name
Country or Region US
Organisation Entrust, Inc.
Organisational Unit See www.entrust.net/legal-terms
Organisational Unit (c) 2014 Entrust, Inc. - for authorized use only
Common Name Entrust Certification Authority - L1M

Serial Number 24 F5 40 B3 F7 9F 29 57 72 A0 F1 1C 6F 3D E7 AB
Version 3
Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)
Parameters None

Not Valid Before Wednesday, 30 March 2022 at 10:59:12 am Australian Eastern Daylight Time
Not Valid After Saturday, 29 April 2023 at 9:59:12 am Australian Eastern Standard Time

Public Key Info
Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters None
Public Key 256 bytes: BF 7E 21 BA 6C E0 A1 9D ...
Exponent 65537
Key Size 2,048 bits
Key Usage Encrypt, Verify, Wrap, Derive
Signature 256 bytes: C3 28 89 A4 13 51 B0 8A ...

? How did my browser know that this is a "valid" cert?

Domain Name Certification

- The Commonwealth Bank of Australia has generated a key pair
- And they passed a certificate signing request to a company called “Entrust” in the US
- Who was willing to vouch (in a certificate) that the entity is called the Commonwealth Bank of Australia and they have control of the the domain name www.commbank.com.au and they have a certain public key
- So, if I can associate this public key with a connection then I have a high degree of confidence that I’ve connected to an entity that is able to demonstrate knowledge of the private key for www.commbank.com.au, as long as I am prepared to trust Entrust and the certificates that they issue
- And I’m prepared to trust them because Entrust NEVER lie!

Domain Name Certification

- The Commonwealth Bank of Australia has generated a key pair
- And they passed a certificate signing request to a company called “Entrust” in the US
- Who was willing to vouch (in a certificate) that the entity is called the Commonwealth Bank of Australia and they have control of the the domain name www.commbank.com.au and they have a certain public key
- So, if I can associate this public key with a connection then I have a high degree of confidence that I’ve connected to an entity that is able to demonstrate knowledge of the private key for www.commbank.com.au, as long as I am prepared to trust Entrust and the certificates that they issue
- And I’m *How do i know that? Why should i trust them?* because Entrust NEVER lie!

Local Trust

The cert i'm being asked to trust was issued by a certification authority that my browser already trusts - so i trust that cert!

Keychain Access

Default Keychains
login
iCloud

System Keychains
Directory Servi...
System
System Roots

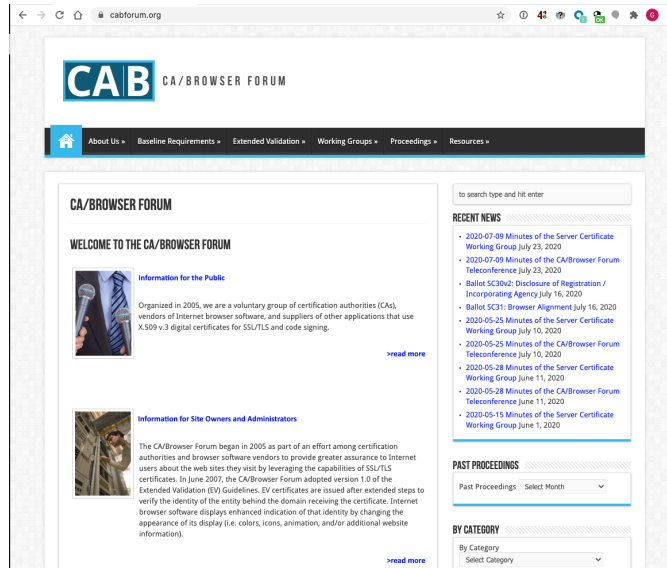
All Items Passwords Secure Notes My Certificates Keys **Certificates**

 **Entrust Root Certification Authority**
Root certificate authority
Expires: Saturday, 28 November 2026 at 7:53:42 am Australian Eastern Daylight Time
This certificate is valid

Name	Kind	Expires	Keychain
D-TRUST Root CA 3 2013	certificate	20 Sep 2026 at 6:25:51 pm	System Roots
D-TRUST Root Class 3 CA 2 2009	certificate	5 Nov 2029 at 7:35:58 pm	System Roots
D-TRUST Root Class 3 CA 2 EV 2009	certificate	5 Nov 2029 at 7:50:46 pm	System Roots
Developer ID Certification Authority	certificate	2 Feb 2027 at 9:12:15 am	System Roots
DigiCert Assured ID Root CA	certificate	10 Nov 2031 at 11:00:00...	System Roots
DigiCert Assured ID Root G2	certificate	15 Jan 2038 at 11:00:00...	System Roots
DigiCert Assured ID Root G3	certificate	15 Jan 2038 at 11:00:00...	System Roots
DigiCert Global Root CA	certificate	10 Nov 2031 at 11:00:00...	System Roots
DigiCert Global Root G2	certificate	15 Jan 2038 at 11:00:00...	System Roots
DigiCert Global Root G3	certificate	15 Jan 2038 at 11:00:00...	System Roots
DigiCert High Assurance EV Root CA	certificate	10 Nov 2031 at 11:00:00...	System Roots
DigiCert Trusted Root G4	certificate	15 Jan 2038 at 11:00:00...	System Roots
E-Tugra Certification Authority	certificate	3 Mar 2023 at 11:09:48 pm	System Roots
Echoworx Root CA2	certificate	7 Oct 2030 at 9:49:13 pm	System Roots
emSign ECC Root CA - G3	certificate	19 Feb 2043 at 5:30:00 am	System Roots
emSign Root CA - G1	certificate	15 Feb 2018 at 5:00:00 am	System Roots
Entrust Root Certification Authority	certificate	28 Nov 2026 at 7:53:42 a...	System Roots
Entrust Root Certification Authority - EC1	certificate	19 Dec 2037 at 2:55:56 am	System Roots
Entrust Root Certification Authority - G2	certificate	8 Dec 2030 at 4:55:54 am	System Roots
Entrust Root Certification Authority - G4	certificate	27 Dec 2037 at 10:41:16...	System Roots
Entrust.net Certification Authority (2048)	certificate	25 Jul 2029 at 12:15:12 am	System Roots
ePKI Root Certification Authority	certificate	20 Dec 2034 at 1:31:27 pm	System Roots
GDCA TrustAUTH R5 ROOT	certificate	1 Jan 2041 at 2:59:59 am	System Roots
GeoTrust Primary Certification Authority	certificate	17 Jul 2036 at 9:59:59 am	System Roots
GeoTrust Primary Certification Authority - G2	certificate	19 Jan 2038 at 10:59:59...	System Roots
GeoTrust Primary Certification Authority - G3	certificate	2 Dec 2037 at 10:59:59 am	System Roots
Global Chambersign Root	certificate	1 Oct 2037 at 2:14:18 am	System Roots
Global Chambersign Root - 2008	certificate	31 Jul 2038 at 10:31:40 pm	System Roots
GlobalSign	certificate	19 Jan 2038 at 2:14:07 pm	System Roots
GlobalSign	certificate	19 Jan 2038 at 2:14:07 pm	System Roots
GlobalSign	certificate	18 Mar 2029 at 9:00:00 pm	System Roots
GlobalSign	certificate	10 Dec 2034 at 11:00:00...	System Roots
GlobalSign Root CA	certificate	28 Jan 2028 at 11:00:00...	System Roots
GlobalSign Root E46	certificate	20 Mar 2046 at 11:00:00...	System Roots
GlobalSign Root R46	certificate	20 Mar 2046 at 11:00:00...	System Roots
GlobalSign Secure Mail Root E45	certificate	18 Mar 2045 at 11:00:00...	System Roots
GlobalSign Secure Mail Root R45	certificate	18 Mar 2045 at 11:00:00...	System Roots
Go Daddy Class 2 Certification Authority	certificate	30 Jun 2034 at 3:06:20 am	System Roots

Local Trust

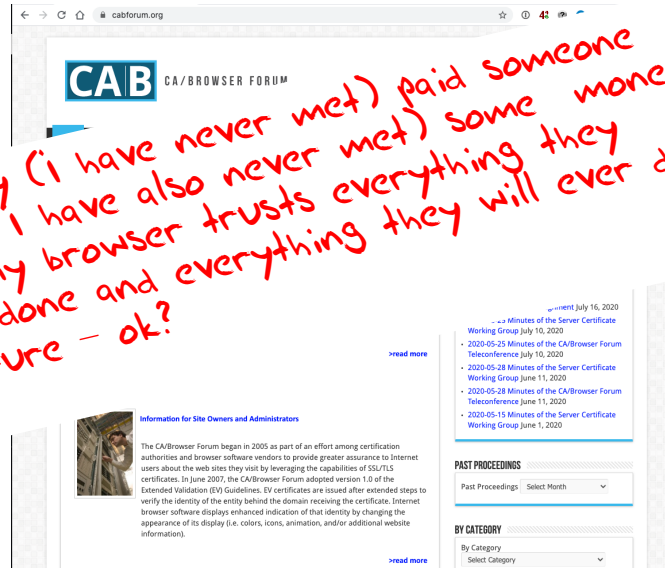
These Certificate Authorities are listed in my computer's trust set because they claim to operate according to the practices defined by the CAB industry forum (of which they are a member) and they **never** lie!



Local Trust

These Certificate Authorities are listed in my computer's trust set because they claim to operate according to the practices defined by the CAB industry forum (of which they are a member) and they **never** lie!

So somebody (i have never met) paid someone else (whom i have also never met) some money and then my browser trusts everything they have ever done and everything they will ever do in the future - ok?



Local Trust or Local Credulity*?

Wow!

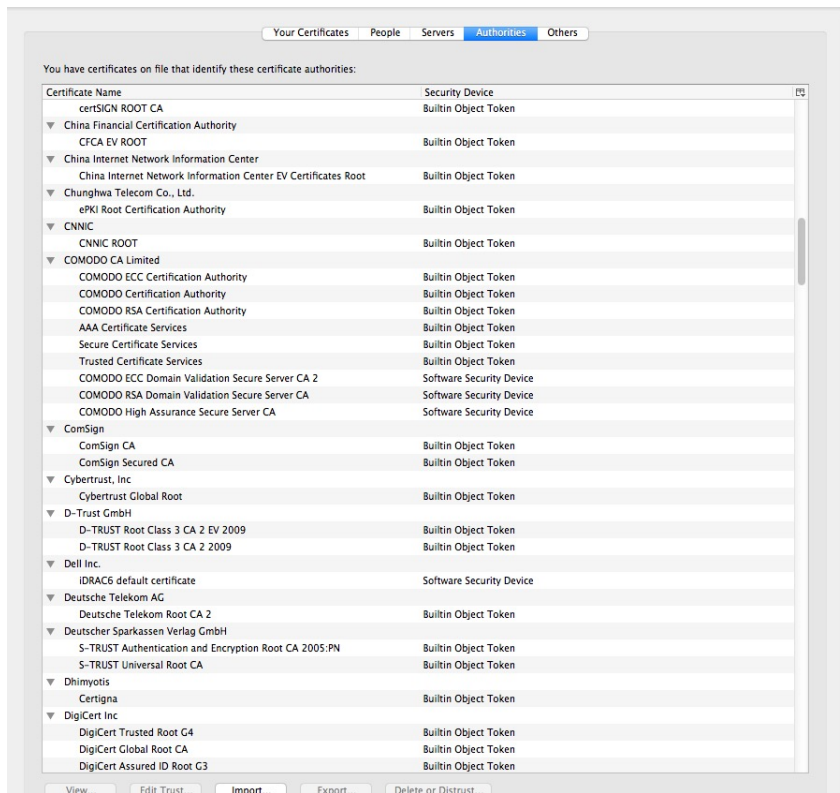
Are they **all** trustable?

* cre·du·li·ty

/krəˈd(y)ŋlədə/

noun

a tendency to be too ready to believe that something is real or true.



Local Credulity

Wow!

Are they **all** trustable?

Evidently Not!

The screenshot shows a web browser window with a sidebar on the left listing various certificate authorities (CAs). The main content area displays a blog post titled "Maintaining digital certificate security" by Adam Langley, Security Engineer, dated Monday, March 23, 2015. The blog post discusses an incident where unauthorized digital certificates were issued for several Google domains by an intermediate certificate authority held by a company called MCS Holdings. The post mentions that CNNIC is included in all major root stores and so the misissued certificates would be trusted by almost all browsers and operating systems. The post also mentions that the certificates were issued by an intermediate certificate authority apparently held by a company called MCS Holdings. This intermediate certificate was issued by CNNIC. The post further states that CNNIC is included in all major root stores and so the misissued certificates would be trusted by almost all browsers and operating systems. Chrome on Windows, OS X, and Linux, ChromeOS, and Firefox 33 and greater would have rejected these certificates because of public-key pinning, although misissued certificates for other sites likely exist. The post concludes by stating that they promptly alerted CNNIC and other major browsers about the incident, and they blocked the MCS Holdings certificate in Chrome with a CRLSet push. CNNIC responded on the 22nd to explain that they had contracted with MCS Holdings on the basis that MCS would only issue certificates for domains that they had registered. However, rather than keep the private key in a suitable HSM, MCS installed it in a man-in-the-middle proxy. These devices intercept secure connections by masquerading as the intended destination and are sometimes used by companies to intercept their employees' secure traffic for monitoring or legal reasons. The employees' computers normally have to be configured to trust a proxy for it to be able to do this. However, in this case, the presumed proxy was given the full authority of a public CA, which is a serious breach of the CA system. This situation is similar to a failure by ANSSI in 2013.

Certificate Name: certSIGN ROOT CA, Security Device: Built-in Object Token

China Financial Certification Authority, CFCA EV ROOT, Built-in Object Token

China Internet Network Information Center, China Internet Network Information Center EV Certificates Root, Built-in Object Token

Chunghwa Telecom, Built-in Object Token

CNNIC, CNNIC ROOT, COMODO CA Limited, COMODO ECC Certificate, COMODO Certificate, COMODO RSA Certificate, AAA Certificate, Secure Certificate, Trusted Certificate, COMODO ECC Certificate, COMODO RSA Certificate, COMODO High

ComSign, ComSign CA, ComSign Secure

Cybertrust, Inc, Cybertrust Global

D-Trust GmbH, D-TRUST Root Certificate, D-TRUST Root Certificate

Dell Inc., iDRAC6 default

Deutsche Telekom, Deutsche Telekom

Deutscher Sparkassen, S-TRUST Authority, S-TRUST Universal

Dhimyotis, Certigna

DigiCert Inc, DigiCert Trusted, DigiCert Global, DigiCert Assure

View... Ed

Maintaining digital certificate security

Posted: Monday, March 23, 2015

Posted by Adam Langley, Security Engineer

On Friday, March 20th, we became aware of unauthorized digital certificates for several Google domains. The certificates were issued by an intermediate certificate authority apparently held by a company called MCS Holdings. This intermediate certificate was issued by CNNIC.

CNNIC is included in all major root stores and so the misissued certificates would be trusted by almost all browsers and operating systems. Chrome on Windows, OS X, and Linux, ChromeOS, and Firefox 33 and greater would have rejected these certificates because of public-key pinning, although misissued certificates for other sites likely exist.

We promptly alerted CNNIC and other major browsers about the incident, and we blocked the MCS Holdings certificate in Chrome with a CRLSet push. CNNIC responded on the 22nd to explain that they had contracted with MCS Holdings on the basis that MCS would only issue certificates for domains that they had registered. However, rather than keep the private key in a suitable HSM, MCS installed it in a man-in-the-middle proxy. These devices intercept secure connections by masquerading as the intended destination and are sometimes used by companies to intercept their employees' secure traffic for monitoring or legal reasons. The employees' computers normally have to be configured to trust a proxy for it to be able to do this. However, in this case, the presumed proxy was given the full authority of a public CA, which is a serious breach of the CA system. This situation is similar to a failure by ANSSI in 2013.

Evidently Not!

Are they **all** trustable?

COMODO CA Limited

The real security issue behind the Comodo hack

News of an Iranian hacker **duping** certification authority Comodo into issuing digital certificates to one or more unauthorized parties has caused an uproar in the IT community, moving some critics to call for Microsoft and Mozilla to remove Comodo as a trusted root certification authority from the systems under their control. Though the hacker managed his way off first compromising a site containing a hard-coded logon name and password, then generating certificates for several well-known sites, including Google, Live.com, Skype, and Yahoo, I'm not bothered by the

Never?

Well, hardly ever

ars

TECHNICA

q

BIZ & IT

TECH

SCIENCE

POLICY

CARS

GAMING & CULTURE

FORUMS

≡

ST

RISK ASSESSMENT —

Already on probation, Symantec issues more illegit HTTPS certificates

At least 108 Symantec certificates threatened the integrity of the encrypted Web.

DAN GOODIN · 1/21/2017, 8:40 AM



Enlarge

62

A security researcher has unearthed evidence showing that three browser-trusted certificate authorities (CAs) owned and operated by Symantec improperly issued more than 100 unvalidated **transport layer security** certificates. In some cases, those certificates made it possible to spoof HTTPS-protected websites.

<http://arstechnica.com/security/2017/01/already-on-probation-symantec-issues-more-illegit-https-certificates/>

Misissued/Suspicious Symantec Certificates

Andrew Ayer | Thu, 19 Jan 2017 13:47:06 -0800

I. Misissued certificates for example.com

On 2016-07-14, Symantec misissued the following certificates for example.com:

<https://crt.sh/?sha256=A8F14F52CC1282D7153A13316E7DA39E6AE37B1A10C16288B9024A9B9DC3C4C6>

<https://crt.sh/?sha256=8B5956C57FDC720B6907A4B1BC8CA2E46CD90EAD5C061A426CF48A6117BFBEA>

<https://crt.sh/?sha256=94482136A1400BC3A1136FECA3E79D4D200E03DD20B245D19F0E78B5679EAF48>

<https://crt.sh/?sha256=C69AB94C1B20E6FC7861C67476CADD1DAE7A8DCF6E23E15311C2D2794BFCDD1>

I confirmed with ICANN, the owner of example.com, that they did not authorize these certificates. These certificates were already revoked at the time I found them.

II. Suspicious certificates for domains containing the word "test"

On 2016-11-15 and 2016-10-26, Symantec issued certificates for various domains containing the word "test" which I strongly suspect were misissued:

Well, hardly ever



Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Distrust of the Symantec PKI: Immediate action needed by site operators

March 7, 2018

Posted by Devon O'Brien, Ryan Sleevi, Emily Stark, Chrome security team

We [previously announced](#) plans to deprecate Chrome's trust in the Symantec certificate authority (including Symantec-owned brands like Thawte, VeriSign, Equifax, GeoTrust, and RapidSSL). This post outlines how site operators can determine if they're affected by this deprecation, and if so, what needs to be done and by when. Failure to replace these certificates will result in site breakage in upcoming versions of major browsers, including Chrome.

Chrome 66

If your site is using a SSL/TLS certificate from Symantec that was issued before June 1, 2016, it will stop functioning in Chrome 66, which could already be impacting your users.

If you are uncertain about whether your site is using such a certificate, you can preview these changes in [Chrome Canary](#) to see if your site is affected. If connecting to your site displays a certificate error or a warning in DevTools as shown below, you'll need to replace your certificate. You can get a new certificate from any [trusted CA](#), including Digicert, which recently acquired Symantec's CA business.

These are isolated events

No they're not:

<https://www.feistyduck.com/ssl-tls-and-pki-history/>



[HOME](#) [BOOKS](#) [TRAINING](#) [NEWSLETTER](#) [RESOURCES](#)

SSL/TLS and PKI History

A comprehensive history of the most important events that shaped the SSL/TLS and PKI ecosystem. Based on [Bulletproof TLS and PKI](#), by Ivan Ristić. [Tweet](#)

Last updated in February 2022.

1994

SSL v2

Netscape develops SSL v2, an encryption protocol designed to support the Web as a hot new commerce platform. This first secure protocol version shipped in Netscape Navigator 1.1 in March 1995.



November 1994

With unpleasant consequences when it all
goes wrong

With unpleasant consequences when it all
goes wrong



BORDER GATEWAY PROTOCOL ATTACK —

Suspicious event hijacks Amazon traffic for 2 hours, steals cryptocurrency

Almost 1,300 addresses for Amazon Route 53 rerouted for two hours.

DAN GOODIN - 4/25/2018, 5:00 AM



Amazon

123



Amazon lost control of a small number of its cloud services IP addresses for two hours on Tuesday morning when hackers exploited a known Internet-protocol weakness that let them to redirect traffic to rogue destinations. By subverting Amazon's domain-resolution service, the attackers masqueraded as cryptocurrency website MyEtherWallet.com and stole about \$150,000 in digital coins from unwitting end users. They may have targeted other Amazon customers as well.

The incident, which started around 6 AM California time, hijacked roughly 1,300 IP addresses, Oracle-owned Internet Intelligence [said on Twitter](#). The malicious redirection was caused by fraudulent routes that were announced by [Columbus, Ohio-based eNet](#), a large Internet service provider that is referred to as autonomous system 10297. Once in place, the eNet announcement caused Hurricane Electric and possibly Hurricane Electric customers and other eNet peers to send traffic over the same unauthorized routes. The 1,300 addresses belonged to [Route 53](#), Amazon's domain name system service

The attackers managed to steal about \$150,000 of currency from MyEtherWallet users,

What's going wrong here?

What's going wrong here?

- The TLS handshake cannot specify WHICH CA should be used by the client to validate the digital certificate that describes the server's public key
- The result is that your browser will allow ANY CA to be used to validate a certificate!

What's going wrong here?

- The TLS handshake cannot specify WHICH CA should be used by the client to validate the digital certificate that describes the server's public key
- The result is that your browser will allow ANY CA to be used to validate a certificate!

WOW! That's awesomely bad!

What's going wrong here?

- The TLS handshake cannot specify WHICH CA



Here's a lock – it might be the lock on your front door for all i know.

The lock might LOOK secure, but don't worry – literally ANY key can open it!

validate a certificate!

WOW! That's awesomely bad!

What's going wrong here?

- There is no incentive for quality in the CA marketplace
- Why pay more for any certificate when the entire CA structure is only as strong as the weakest CA
- And your browser trusts a LOT of CAs!
 - About 60 – 100 CA's
 - About 1,500 Subordinate RA's
 - Operated by 650 different organisations

See the EFF SSL observatory

<http://www.eff.org/files/DefconSSLiverse.pdf>

In a Commercial Environment

Where CA's compete with each other for market share

And quality offers no protection

Then what 'wins' in the market?



?

In a Commercial Environment

Where CA's compete with each other for market share
And quality offers no protection
Then what 'wins' in the market?

Sustainable
Resilient

Secure
Privacy
Trusted



cheap!

But its all OK

Really.

- Because 'bad' certificates can be revoked
- And browsers **always** check revocation status of certificates before they trust them

Always?

Ok - Not Always.
Some do.
Sometimes.

Platform	Chrome	Firefox	Opera	Safari	Edge
Mac OS X 10.15.3	YES 80.0.3987.132	YES 73.0.1	YES 67.0.3575.53	YES 13.0.5	
iOS 13.3.1	YES 80.0.3987.95	YES 23.0	NO 16.0.15	YES 13.3.1	
Android 10	NO 80.0.3987.132	NO 68.6.0	NO 56.1		
Windows 10	NO 80.0.3987.132	YES 74.0	NO 67		YES 44.18362

Table 1 – Browser Revocation Status

So we can't count on revocation

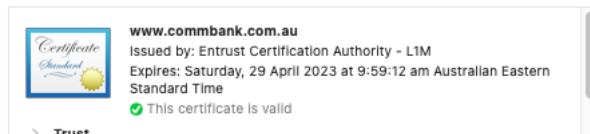
- If we can't revoke certificates then we need to reduce certificate lifetimes

So we can't count on revocation

- If we can't revoke certificates then we need to reduce certificate lifetimes
- But we are not doing that!

emSign ECC Root CA - G3	certificate	19 Feb 2043 at 5:30:00 am	System Roots
emSign Root CA - G1	certificate	19 Feb 2043 at 5:30:00 am	System Roots
Entrust Root Certification Authority	certificate	28 Nov 2026 at 7:53:42 a...	System Roots
Entrust Root Certification Authority - EC1	certificate	10 Dec 2037 at 2:55:36 am	System Roots

Yes, 2026!!!



Yes, 2023!!!

So we can't count on revocation

- If we can't revoke certificates then we need to reduce certificate lifetimes
- What's a “safe” certificate lifetime?

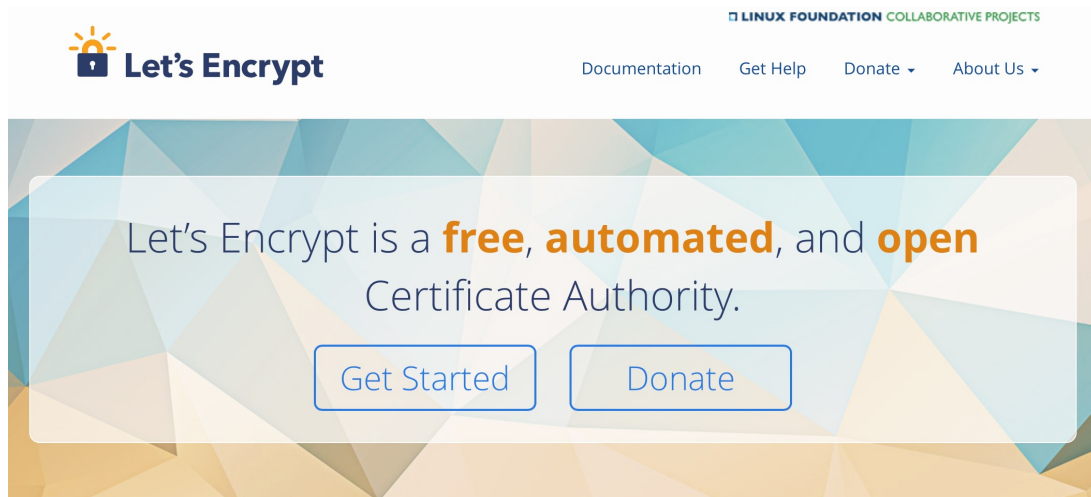


So we can't count on revocation

- If we can't revoke certificates then we need to reduce certificate lifetimes
- What's a "safe" certificate lifetime?
- If we want 2 hours or less then we need to think hard about how to achieve this

How can we make certificates better?

Option A: Take all the money out of the system!



How can we make certificates better?

Option A: Take all the money out of the system!



How can we make certificates better?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json

transport_security_state_static.json

```
1 // Copyright (c) 2012 The Chromium Authors. All rights reserved.
2 // Use of this source code is governed by a BSD-style license that can be
3 // found in the LICENSE file.
4
5 // This file contains the HSTS preloaded list in a machine readable format.
6
7 // The top-level element is a dictionary with two keys: "pinsets" maps details
8 // of certificate pinning to a name and "entries" contains the HSTS details for
9 // each host.
10 //
11 // "pinsets" is a list of objects. Each object has the following members:
12 //   name: (string) the name of the pinset
13 //   static_spki_hashes: (list of strings) the set of allowed SPKIs hashes
14 //   bad_static_spki_hashes: (optional list of strings) the set of forbidden
15 //     SPKIs hashes
16 //   report_uri: (optional string) the URI to send violation reports to;
17 //     reports will be in the format defined in RFC 7469
18 //
19 // For a given pinset, a certificate is accepted if at least one of the
20 // "static_spki_hashes" SPKIs is found in the chain and none of the
21 // "bad_static_spki_hashes" SPKIs are. SPKIs are specified as names, which must
22 // match up with the file of certificates.
23 //
```

How can we make certificates better?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch?cs=src/net/transport_security_state_static.json

its not a totally insane idea -- until you realise that it appears to be completely unscalable!

its just Google protecting itself and no one else

```
transport_security_state_static.json
// This source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// This file contains the HSTS preloaded list in a machine readable format.
6
7 // The top-level element is a dictionary with two keys: "pinsets" maps details
8 // of certificate pinning to a name and "entries" contains the HSTS details for
9 // each host.
10 //
11 // "pinsets" is a list of objects. Each object has the following members:
12 //   name: (string) the name of the pinset
13 //   static_spki_hashes: (list of strings) the set of allowed SPKIs hashes
14 //   bad_static_spki_hashes: (optional list of strings) the set of forbidden
15 //     SPKIs hashes
16 //   report_uri: (optional string) the URI to send violation reports to;
17 //     reports will be in the format defined in RFC 7469
18 //
19 // For a given pinset, a certificate is accepted if at least one of the
20 // "static_spki_hashes" SPKIs is found in the chain and none of the
21 // "bad_static_spki_hashes" SPKIs are. SPKIs are specified as names, which must
22 // match up with the file of certificates.
23 //
```

How can we make certificates better?

O it's not a totally insane idea -- until you realise that it appears to be completely unscalable!

its just Google protecting itself and no one else

<https://code.google.com/p/chromium/codesearch#chromium/src/net/http/tree-secure-static-ic>



INFOWORLD TECH WATCH

By Fahmida Y. Rashid, Senior Writer, InfoWorld | JAN 30, 2017

About | RSS

Informed news analysis every weekday

Google moves into the Certificate Authority business

Google doesn't seem to trust the current system, as it has launched its own security certificates

```
17 //      reports will be in the format defined in RFC 7469
18 //
19 // For a given pinset, a certificate is accepted if at least one of the
20 // "static_spki_hashes" SPKIs is found in the chain and none of the
21 // "bad_static_spki_hashes" SPKIs are. SPKIs are specified as names, which must
22 // match up with the file of certificates.
23 //
```

How can we make certificates better?

Option C: Certificate Transparency

Google Transparency Report

[Overview](#) [Certificates](#)

HTTPS encryption on the web

Certificate transparency

In order to provide encrypted traffic to users, a site must first apply for a certificate from a trusted Certificate Authority (CA). This certificate is then presented to the browser to authenticate the site the user is trying to access. In recent years, due to structural flaws in the HTTPS certificate system, certificates and issuing CAs have proven vulnerable to compromise and manipulation. Google's Certificate Transparency project aims to safeguard the certificate issuance process by providing an open framework for monitoring and auditing HTTPS certificates.

Use the search bar below to look up all of a domain's certificates that are present in [active public certificate transparency logs](#). Site owners can search this site for domain names they control to ensure there have been no incorrect issuances of certificates referencing their domains.

Google encourages all CAs to write the certificates they issue to publicly verifiable, append-only, tamper-proof logs. In the future, Chrome and other browsers may decide not to accept certificates that have not been written to such logs.

As of May 6, 2020, there have been 9,178,649,366 entries made to the set of Certificate Transparency logs that Google monitors.

[Learn more about the Certificate Transparency Project](#)

Search certificates by hostname

☒ include subdomains

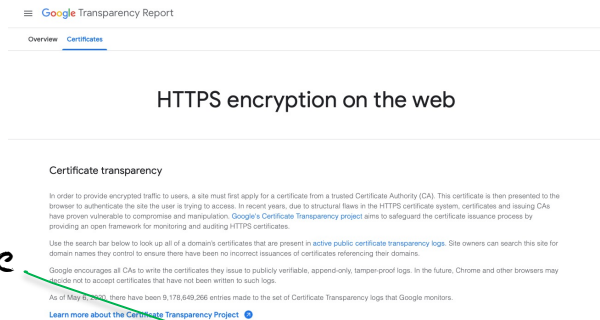
Current status:

Issuer	# issued
C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3	36 Filter

Subject	Issuer	# DNS names	Valid from	Valid to	# CT logs
*.potaroo.net	Let's Encrypt Authority X3	1	Mar 29, 2020	Jun 27, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Oct 21, 2019	Jan 19, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Aug 22, 2019	Nov 20, 2019	6 See details

How can we make certificates better?

Option C: Certificate Transparency



This is true

In order to provide encrypted traffic to users, a site must first apply for a certificate from a trusted Certificate Authority (CA). This certificate is then presented to the browser to authenticate the site the user is trying to access. In recent years, due to structural flaws in the HTTPS certificate system, certificates and issuing CAs have proven vulnerable to compromise and manipulation. Google's Certificate Transparency project aims to safeguard the certificate issuance process by providing an open framework for monitoring and auditing HTTPS certificates.

Current status:

Issuer					# issued
C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3					36 Filter
Subject	Issuer	# DNS names	Valid from	Valid to	# CT logs
*.potaroo.net	Let's Encrypt Authority X3	1	Mar 29, 2020	Jun 27, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Oct 21, 2019	Jan 19, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Aug 22, 2019	Nov 20, 2019	6 See details

This is a fail

How can we maske certificates better?

Option C: Certificate Transparency

Google Transparency Report

Overview Certificates

HTTPS encryption on the web

Certificate transparency

In order to provide encrypted traffic to users, a site must first apply for a certificate from a trusted Certificate Authority (CA). This certificate is then presented to the browser to authenticate the site the user is trying to access. In recent years, due to structural flaws in the HTTPS certificate system, certificate issuing CAs have proven vulnerable to compromise and manipulation. Google's Certificate Transparency project aims to safeguard the certificate issuance process by providing an open framework for monitoring and auditing HTTPS certificates.

Use the search bar below to look up all of a domain's certificates that are present in [active public certificate logs](#). This site will search this site for certificates issued by all CAs to ensure there have been no incorrect issuances of certificates. Google encourages all CAs to write their certificates to public logs to ensure transparency. Google also provides tamper-proof logs. In the future, Google may offer other options to decide not to accept certificates that are not in public logs.

[Learn more about the Certificate Transparency project.](#)

Search certificates by hostname

www.potaroo.net

☒ include subdomains

Filter status:

Issuer	# issued
C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3	36 Filter

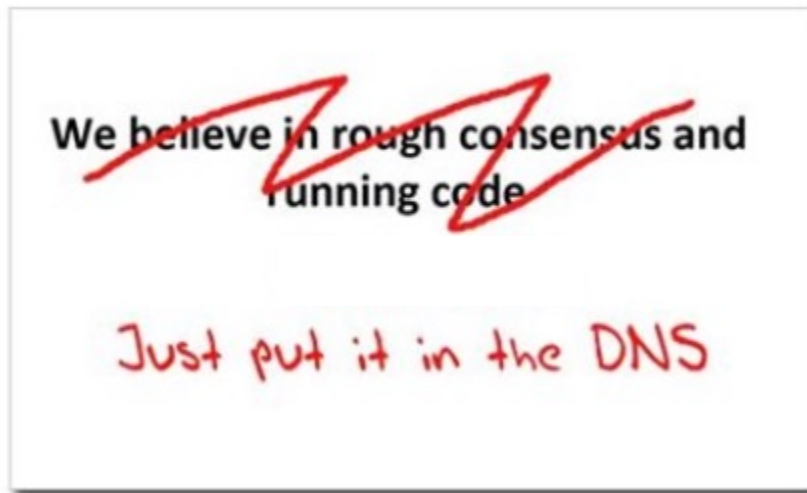
Subject	Issuer	# DNS names	Valid from	Valid to	# CT logs
*.potaroo.net	Let's Encrypt Authority X3	1	Mar 29, 2020	Jun 27, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Oct 21, 2019	Jan 19, 2020	4 See details
www.potaroo.net	Let's Encrypt Authority X3	1	Aug 22, 2019	Nov 20, 2019	6 See details

its just so broken
These transparency logs are a case of same week service in a millisecond world -- Assuming anyone looks in the first place!

Cert Transparency is probably worse than a placebo!

How can we make certificates better?

Option D: Use the DNS!



Seriously? The DNS?

Where better to find out the public key associated with a DNS-named service than to look it up in the DNS?

- Why not query the DNS for the HSTS record?
- Why not query the DNS for the issuer CA?
- Why not query the DNS for the hash of the domain name cert?
- Why not query the DNS for the hash of the domain name public key?

Seriously? The DNS?

Where better to find out the public key associated with a DNS-named service than to look it up in the DNS?

- Why not query the DNS for the HSTS?
- Why not query the DNS for the CA?
- Why not query the DNS for the hash of the domain name cert?
- Why not query the DNS for the hash of the domain name public key?

Who needs CA's anyway?!

DANE

- Using the DNS to associated domain name public key certificates with domain name

[Docs] [txt|pdf] [draft-ietf-dane-p...] [Diff1] [Diff2] [Errata]

Updated by: [7218](#), [7671](#) PROPOSED STANDARD

Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6698
Category: Standards Track
ISSN: 2070-1721

P. Hoffman
VPN Consortium
J. Schlyter
Kirei AB
August 2008

The DNS-Based Authentication of Names Transport Layer Security

Abstract

Encrypted communication on the Internet often uses Transport Layer Security (TLS). This document improves on that situation by enabling the administrators of domain names to specify the keys used in that domain's TLS servers. This requires matching improvements in TLS client software, but no change in TLS server software.

Status of This Memo

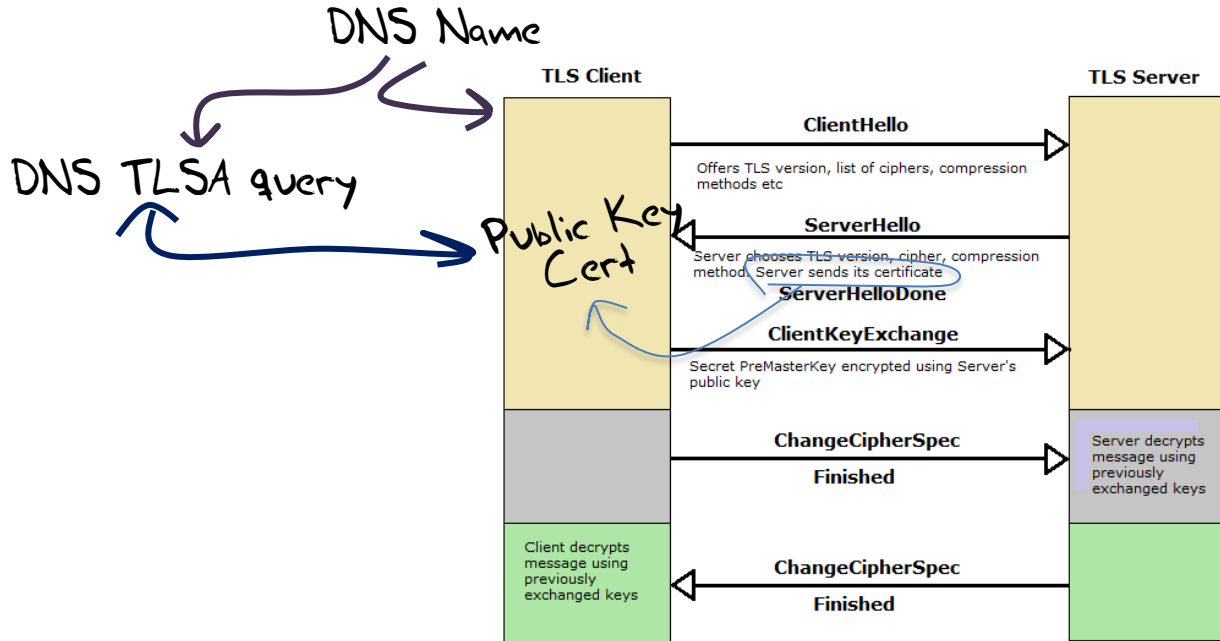
This is an Internet Standards Track document.

RFC 6698 -- You should read this!

TLS with DANE

- Client receives server cert in Server Hello
 - *Client lookups the DNS for the TLSA Resource Record of the domain name*
 - *Client validates the presented certificate against the TLSA RR*
- Client performs Client Key exchange

TLS Connections



Just one problem...

- The DNS is full of liars and lies!
- And this can compromise the integrity of public key information embedded in the DNS
- Unless we fix the DNS we are no better off than before with these TLSA records!

Just one response...

- We need to allow users to **validate** DNS responses for themselves
- And for this we need a Secure DNS framework
- Which we have – and it's called **DNSSEC**!

DANE + DNSSEC

- Query the DNS for the TLSA record of the domain name and ask for the DNSSEC signature to be included in the response
- Validate the signature to ensure that you have an unbroken signature chain to the root trust point
- At this point you can accept the TLSA record as the authentic record, and set up a TLS session based on this data

DANE + DNSSEC

- Query the DNS for the TLSA record of the domain name and ask for the DNSSEC signature to be included in the response
- Validate the signature to ensure that you have an unbroken signature chain to the root
- At this point you can **Yes, but No!** use the TLSA record as the authentic record, and set up a TLS session based on this data

DANE + DNSSEC

ImperialViolet

DNSSEC authenticated HTTPS in Chrome (16 Jun 2011)

Update: this has been removed from Chrome due to lack of use.

DNSSEC validation of HTTPS sites has been hanging around in Chrome for nearly a year now. But it's now enabled by default in the current canary and dev channels of Chrome and is on schedule to go stable with Chrome 14. If you're running a canary or dev channel (and you need today's dev channel release: 14.0.794.0) then you can go to <https://dnssec.imperialviolet.org> and see a DNSSEC signed site in action.



DNSSEC stapled certificates (and the reason that I use that phrase will become clear in a minute) are aimed at sites that currently have, or would use, self-signed certificates and, possibly, larger organisations that are Chrome based and want certificates for internal sites without having to bother with installing a custom root CA on all the client devices. Suggesting that this heralds the end of the CA system would be utterly inaccurate. Given the deployed base of software, all non-trivial sites will continue to use CA signed certificates for decades, at least. DNSSEC signing is just a gateway drug to better transport security.

DANE validation can be
SO SLOW!

Or...

Faster validation?

[\[Docs\]](#) [\[txt|pdf\]](#) [\[draft-ietf-dnso...\]](#) [\[Tracker\]](#) [\[Diff1\]](#) [\[Diff2\]](#)

EXPERIMENTAL

Internet Engineering Task Force (IETF)
Request for Comments: 7901
Category: Experimental
ISSN: 2070-1721

P. Wouters
Red Hat
June 2016

CHAIN Query Requests in DNS

Abstract

This document defines an EDNS0 extension that can be used by a security-aware validating resolver configured to use a forwarding resolver to send a single query, requesting a complete validation path along with the regular query answer. The reduction in queries potentially lowers the latency and reduces the need to send multiple queries at once. This extension mandates the use of source-IP-verified transport such as TCP or UDP with EDNS-COOKIE, so it cannot be abused in amplification attacks.

Status of This Memo

Or ... Look! No DNS!

- Server packages server cert, TLSA record and the DNSSEC credential chain in a single bundle
- Client receives bundle in Server Hello
 - *Client performs validation of TLSA Resource Record using the supplied DNSSEC signatures plus the local DNS Root Trust Anchor without performing any DNS queries*
 - *Client validates the presented certificate against the TLSA RR*
- Client performs Client Key exchange

Doing a better job

We could do a **far** better job at Internet Security by moving on from X.509 public key certificates:

- Publishing DNSSEC-signed zones

- Publishing DANE TLSA records

- Using DNSSEC-validating resolution

- Using TLSA records to guide TLS Key Exchange

- Stapling the TLSA + sig bundle into TLS

Doing a better job

We could do a **far** better job
X.509 public keys

*But nothing has happened for
more than a decade!*

oving on from

L

U

Why not?

... TLS Key Exchange

St

... RSA + sig bundle into TLS

Why is this so hard?

Why is this so hard?

We have different goals

- Some people want to provide strong hierarchical controls on the certificates and keys because it entrenches their role in providing services
- Some want to do it because it gives them a point of control to intrude into the conversations of their citizens
- Others want to exploit weaknesses in the system to leverage a competitive advantage
- Some people think users prefer faster applications, even if they have security weaknesses
- Others think users are willing to pay a time penalty for better authentication controls

Why is this so hard?

Because there are so many moving parts?

- In a system that is constructed upon the efforts of multiple systems and multiple providers we are relying on someone in charge to orchestrate the components to as working whole



Saturn V Launch Vehicle

Three stage rocket, each built by a different contractor

Each of whom used multiple subcontractors

3 million components

Each supplied by the lowest bidder!

Why is this so hard?

Because we are relying on the market to provide coherence and consistency of orchestration across providers?

- And perhaps that's the key point here
- Loosely coupled systems will always present windows of vulnerability
 - Routing integrity
 - Name registration
 - Name certification
 - Service control
- Effective defence involves not only component defence but also in defending the points of interaction between components
- And we find this very hard to achieve when the market itself is the orchestration agent

Users and Trust

- Users just want to be able to trust that the websites and services that they connect to and share their credentials, passwords and content with are truly the ones they expected to be using without first studying for a PhD in Network Operational Security
- Somehow we're missing that simple objective and we've interposed complexity and adornment that have taken on a life of their own and are in fact eroding trust
- And that's bad!
- **If we can't trust our communications infrastructure, then we don't have a useful communications infrastructure.**

What a dysfunctional mess we've created!

Can we make it better?

We could do a better job if we knew what we wanted

Can we make it better?

We could do a better job if we knew what we wanted

- Single point of trust for EVERYTHING (DNSSEC)

or

- Many points of trust in a highly distributed framework (Web PKI)

Can we make it better?

We could do a better job if we knew what we wanted

- Highly robust validation performed by the client

or

- Fast!

Can we make it better?

We could do a better job if we knew what we wanted

- Single common secure credential infrastructure

or

- Application-specific credentials

Can we make it better?

- Yes, if we could only agree on what we want in the first place!
- And we just can't agree on that!

That's it!

Questions?