Security

Geoff Huston
Chief Scientist, APNIC
Insecurity

Geoff Huston
Chief Scientist, APNIC
Which Bank?

You can count on CommBank in uncertain times.

Continuing to serve our customers and support the economy.

See coronavirus support
You can count on CommBank in uncertain times.

Continuing to serve our customers and support the economy.

See coronavirus support

Explore products  Support  Rates & fees  Tools & calculators

Bank accounts

Credit cards

Savings with certainty

Coronavirus support for home loan customers

Coronavirus support for business

Which Bank? My Bank!
<table>
<thead>
<tr>
<th>Explore products</th>
<th>Support</th>
<th>Rates &amp; fees</th>
<th>Tools &amp; calculators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank accounts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit cards</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can count on CommBank in uncertain times.

Continuing to serve our customers and support the economy.

See coronavirus support

Coronavirus support for home loan customers

Savings with certainty

View all terms and conditions

Coronavirus support for business

I hope!
Security on the Internet

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

It's 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?

It's trivial to mock up a web page to look like another.

So why should I enter my username and password into this particular screen?
Opening the Connection: First Steps

Client:

*DNS Query:*

www.commbank.com.au?

*DNS Response:*

23.214.88.32

*TCP Session:*

TCP Connect 23.214.88.32, port 443
Hang on...

$ dig -x 23.214.88.32 +short
Hang on...

$ dig -x 23.214.88.32 +short

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 23.214.88.32 +short

That’s an Akamai IP address

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

Why should my browser trust that 23.214.88.32 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?
Security on the Internet

Also, how can you keep your session a secret from wire(less) snoopers?
It's all about cryptography
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
The Power of Primes

\[(m^e)^d \equiv m \pmod{n}\]

As long as \(d\) and \(n\) are relatively large, and \(n\) is the product of two large prime numbers, then finding the value of \(d\) when you already know the values of \(e\) and \(n\) is computationally expensive.
Why is this important?

Because much of the foundation of Internet security rests upon this prime number relationship

Because prime number factorization still involves enumeration

And cryptography is still about getting the defender to perform just enough work to make the attacker’s task so much greater that its infeasible
Back to Public/Private Key Pairs

• If I have a copy of your PUBLIC key,
• And you encrypt a message with your PRIVATE key,
• Then I can decrypt the message.

• 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”
Public Key Certificates

• If the intermediary signs an attestation that this is a public key (with their private key) then
  – I trust this intermediary
  – And this intermediary has said that this is your public key
  – Then I can trust that this is your public key
• This ‘attestation’ takes the form of a “public key certificate”
<table>
<thead>
<tr>
<th>Subject Name</th>
<th>Private Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Category</td>
<td></td>
</tr>
<tr>
<td>Inc. Country/Region</td>
<td>AU</td>
</tr>
<tr>
<td>Serial Number</td>
<td>123 123 124</td>
</tr>
<tr>
<td>Country or Region</td>
<td>AU</td>
</tr>
<tr>
<td>State/Province</td>
<td>New South Wales</td>
</tr>
<tr>
<td>Locality</td>
<td>SYDNEY</td>
</tr>
<tr>
<td>Organisation</td>
<td>Commonwealth Bank of Australia</td>
</tr>
<tr>
<td>Organisational Unit</td>
<td>CBA Business System Hosting</td>
</tr>
<tr>
<td>Common Name</td>
<td><a href="http://www.commbank.com.au">www.commbank.com.au</a></td>
</tr>
</tbody>
</table>

**Issuer Name**
- US

**Organisation**
- DigiCert Inc

**Organisational Unit**
- www.digicert.com

**Common Name**
- DigiCert SHA2 Extended Validation Server CA

**Serial Number**
- 03 1A 62 D5 68 8B 27 9F 00 80 A9 D3 98 4F 41 66

**Version**
- 3

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

**Parameters**
- None

**Not Valid Before**
- Thursday, 25 June 2020 at 10:00:00 am Australian Eastern Standard Time

**Not Valid After**
- Saturday, 23 July 2022 at 10:00:00 pm Australian Eastern Standard Time

**Public Key Info**
- Algorithm: RSA Encryption (1.2.840.113549.1.1.11)
- Parameters: None
- Public Key: 256 bytes: C5 48 B6 88 2D 3F 67 3C ...
- Exponent: 65537
- Key Size: 2048 bits
No all certificates are the same - This certificate binds a public key to a set of names without any attestation to the identity of the name “holder”.

This certificate binds a public key to a set of names without any attestation to the identity of the name "holder".
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.
Moving on...

- Ok, so the certificate system is a total mess, but TLS still works, right?
Secure Connections using TLS

1. **ClientHello**
   - Offers TLS version, list of ciphers, compression methods etc.

2. **ServerHello**
   - **ServerHelloDone**

3. **ClientKeyExchange**
   - Secret PremasterKey encrypted using Server's public key.

4. **ChangeCipherSpec**
   - Client decrypts message using previously exchanged keys.

5. **Finished**
   - Server decrypts message using previously exchanged keys.

6. **ChangeCipherSpec**
   - Finished.

https://rhsecurity.wordpress.com/tag/tls/
Secure Connections using TLS

- **ClientHello**: Offers TLS version, list of ciphers, compression methods etc.
- **ServerHelloDone**
- **ClientKeyExchange**: Secret PreMasterKey encrypted using Server's public key.
- **ChangeCipherSpec**
- **Finished**
- **ChangeCipherSpec**
- **Finished**

Client decrypts message using previously exchanged keys.

Server decrypts message using previously exchanged keys.

https://rhsecurity.wordpress.com/tag/tls/
Secure Connections using TLS

`ClientHello`:
- Offers TLS version, list of ciphers, compression methods etc.

`ServerHello`:
- Server chooses TLS version, cipher, compression method, Server sends its certificate

`ClientKeyExchange`:
- Secret PreMasterKey encrypted using Server's public key

`ChangeCipherSpec`

`Finished`
- Client decrypts message using previously exchanged keys

Server decrypts message using previously exchanged keys

https://rhsecurity.wordpress.com/tag/tls/
Secure Connections using TLS

How does the client recognise this certificate as the “right” certificate?
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 “DigiCert Inc” 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 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 DigiCert and the certificates that they issue
• And I am prepared to trust them because DigiCert 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 “DigiCert Inc” 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 DigiCert and the certificates that they issue
• And I am prepared to trust them because DigiCert NEVER lie!

How do I know that? Why should I trust them?
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!
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?

* credulity
/ˈkredələt/ n
a tendency to be too ready to believe that something is real or true.
Local Credulity

Wow!

Are they all trustable?

Evidently Not!

Maintaining digital certificate security

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.
Local Credulity

Wow!

Are they all trustable?

Evidently Not!
Never?
Well, hardly ever


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

Misissued/Suspicious Symantec Certificates

Andrew Ayer | Thu, 19 Jan 2017 12:47:06 -0600

I. Misissued certificates for example.com

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

https://crt.sh/?sha256=ABF14F52CC12B2D7353A1331887DA39E88E37FR1A10C1828BB624A9BB9C3C4C6

https://crt.sh/?sha256=8B9565E17FC7E72C6E6D97A481BEC8A2E46C508EAD5081A426CF48A617B9F8A

https://crt.sh/?sha256=964B21386A140D53A1186F2E4A82D432D19FDC87B686A19DAF48

https://crt.sh/?sha256=699AO04C1DB20EF7BC1C67476CADD1DA97AB9DCE5B23B1531C2D2794BPC011

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:

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.
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 Sleeter, 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.
With unpleasant consequences when it all goes wrong
Iranian activists feel the chill after hacker taps into e-mails

BY SOMINI SENGUPTA

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

Recent stock market history, page 16

Recent stock market history, page 16

When it all goes wrong with unpleasant consequences

International Herald Tribune

Sep 13, 2011 Front Page
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 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 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?

• 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?

Sustainable, Trusted, Resilient, Secure, Privacy, Trusted?
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 it's all OK

Really.

- Because ‘bad’ certificates can be revoked
- And browsers *always* check revocation status of certificates
Always?
Ok – Not Always. Some do. Sometimes.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Opera</th>
<th>Safari</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OS X 10.15.3</td>
<td>YES 80.0.3987.132</td>
<td>YES 73.0.1</td>
<td>YES 67.0.3575.53</td>
<td>YES 13.0.5</td>
<td></td>
</tr>
<tr>
<td>iOS 13.3.1</td>
<td>YES 80.0.3987.95</td>
<td>YES 23.0</td>
<td>NO 16.0.15</td>
<td>YES 13.3.1</td>
<td></td>
</tr>
<tr>
<td>Android 10</td>
<td>NO 80.0.3987.132</td>
<td>NO 68.6.0</td>
<td>NO 56.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows 10</td>
<td>NO 80.0.3987.132</td>
<td>YES 74.0</td>
<td>NO 67</td>
<td>YES 44.18362</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 – Browser Revocation Status

https://www.potaroo.net/ispcol/2020-03/revocation.html
How can we fix this?

Option A: Take all the money out of the system!
How can we fix this?

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

Will the automation of the Cert issuance coupled with a totally free service make the overall environment more or less secure? I think we already know the answer!
How can we fix this?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json
How can we fix this?

Option B: White Listing and Pinning with HSTS

https://code.google.com/p/chromium/source/browse/src/net/http/transport_security_state_static.json

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

It's just Google protecting itself and no one else.
How can we fix this?

Option B: White listing and Pinning with HSTS

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

It's just Google protecting itself and no one else.

Google moves into the Certificate Authority business

Google doesn't seem to trust the current system, as it has launched its own security certificates
How can we fix this?

Option C: Certificate Transparency

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 for use in making secure transactions. 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 9,796,493,262 entries made to the Certificate Transparency logs that Google monitors.

Learn more about the Certificate Transparency Project.
How can we fix this?

Option C: 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.

This is true
How can we fix this?

Option C: Certificate Transparency

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 fix this?

Option D: Use the DNS!

We believe in rough consensus and running code.

Just put it in 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 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?
DANE

• Using the DNS to associate domain name public key certificates with domain name

Abstract

Encrypted communication over the Internet often uses Transport Layer Security (TLS) and depends on third parties to certify the keys used. This document improves upon 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.
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

TLS Client

- ClientHello
  - Offers TLS version, list of ciphers, compression methods etc
- ServerHello
  - Server chooses TLS version, cipher, compression method, Server sends its certificate
- ServerHelloDone
- ClientKeyExchange
  - Secret PreMasterKey encrypted using Server’s public key
- ChangeCipherSpec
- Finished

TLS Server

- Server decrypts message using previously exchanged keys
- ChangeCipherSpec
- Finished

DNS Name

TLSA query

Public Key Cert
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 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.

Yes, but No!
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.
Or...

Faster validation?
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 DNSEC 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:
  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 at Internet Security:

- 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

But nothing has happened for more than a decade!

Why not?
Why is change 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 conversation
– Others want to exploit weaknesses in the system to leverage a competitive advantage
– Some people think users prefer faster applications even if they have weaknesses
– Others think users are willing to pay a time penalty for better authentication controls
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 are missing that simple objective and have 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!
That’s it!

Questions?