

# Security and Stuff

Geoff Huston


APNIC

# What I'm working on at the moment..

EnglishالعربيةEspañolFrançaisРусский中文

Search ICANN.org

Log In | Sign Up

GET STARTEDNEWS & MEDIAPOLICYPUBLIC COMMENTRESOURCESCOMMUNITYIANA STEWARDSHIP & ACCOUNTABILITY

## Design Team Review of Plan for DNS Root Zone KSK Change

Follow Updates

**Open Date**  
6 Aug 2015 23:59 UTC

**Close Date**  
5 Oct 2015 23:59 UTC

**Staff Report Due**  
19 Oct 2015 23:59 UTC

✓

Comments Closed

✓

Report of Public Comments

✓

Contents

[Brief Overview](#)  
[Comments Forum](#)  
[Report of Public Comments](#)  
[Section I: Description, Explanation & Purpose](#)  
[Section II: Background](#)  
[Section III: Relevant Resources](#)  
[Section IV: Additional Information](#)  
[Section V: Reports](#)  
[Staff Contact](#)

### Brief Overview

*Purpose:* This public comment proceeding seeks to review the Design Team's findings to date related to issues and plans for changing the cryptographic key used to originate the DNSSEC chain of trust.

*Current Status:* The Design Team has generated a preliminary report and will accept wider review.

*Next Steps:* After the public comment proceeding, the Design Team will finalize its report and plan for changing the cryptographic key.

[Report of Public Comments](#)

### Section I: Description, Explanation, and Purpose

A design team consisting of seven independent DNS experts has produced a report examining previously proposed schemes for changing the DNSSEC root zone KSK, along with considerations related to Internet realities, in preparation for finalizing plans to change the current Root Zone KSK.

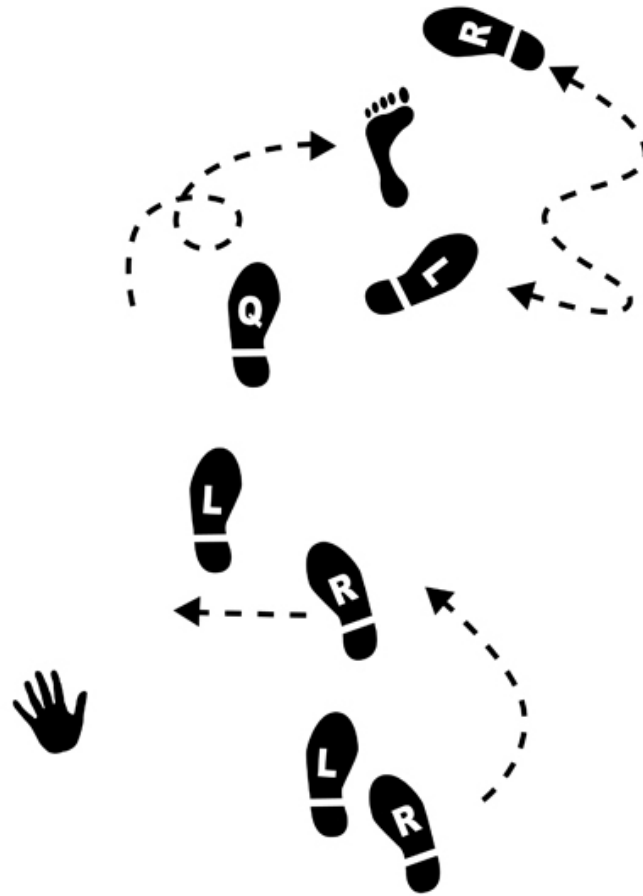
### Section II: Background

In 2010, the Root Zone Management Partners (ICANN, Verisign, and NTIA) introduced the DNS Security Extensions to the operational root zone. After five years of operation, there is a requirement to change the top most cryptographic key in the hierarchy, the key called the Root Zone Key Signing Key. The challenge is to ensure that all copies of the publicly distributed key

# Why is this important?

- Rolling the value of the Key Signing key of the DNS is perhaps one of the more esoteric aspects of the management of Internet infrastructure
- So why should you care that this is done well?
- And what's the problem if it all goes wrong?

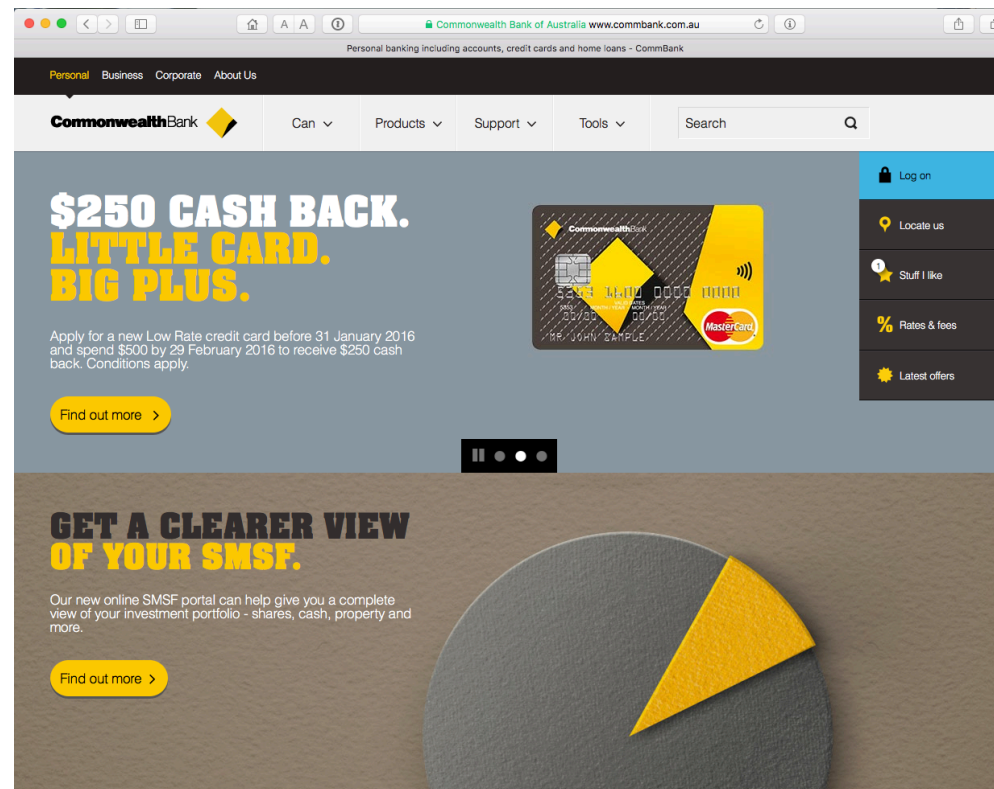
Lets take a step back





# Security on the Internet

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



# Connection Steps

Client:

DNS Query:

www.commbank.com.au?

DNS Response

104.97.235.12

TCP Session:

TCP Connect 104.97.235.12, port 443

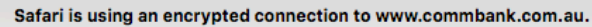
# Hang on...

```
$ dig -x 104.97.235.12 +short  
a104-97-235-12.deploy.static.akamaitechnologies.com.
```

That's not an IP addresses that was allocated to the Commonwealth Bank.  
The Commonwealth Bank of Australia has 140.168.0.0 - 140.168.255.255  
and 203.17.185.0 - 203.17.185.255

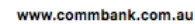
So why should my browser trust that 104.97.235.12 is really the “proper”  
web site for the Commonwealth Bank of Australia and not some dastardly  
evil scam?


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



Symantec Corporation has identified [www.commbank.com.au](http://www.commbank.com.au) as being owned by Commonwealth Bank of Australia in SYDNEY, New South Wales, AU.

↳  [www.commbank.com.au](http://www.commbank.com.au)



Issued by: Symantec Class 3 EV SSL CA - G3  
Expires: Saturday, 27 February 2016 at 10:59:59 AM Australian Eastern Daylight Time  
 This certificate is valid

## ▼ Details

Public Key Info	
Algorithm	RSA Encryption ( 1.2.840.113549.1.1.1 )
Parameters	none
Public Key	256 bytes : CA B4 74 93 E8 00 22 10 ...
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes : 95 32 C3 F0 62 F1 F8 F1 ...

[Hide Certificate](#)

OK

 Latest offers

Find out more >

## FAMILIAR BANKING FOR UNFAMILIAR

Personal

Business

Corporate

Commonwealth Bank of Australia

GET A C OF YOU

Our new online SMSF view of your investments more.

Find out more >

FAMILIAR BANKING FOR UNFAMILIAR

Commonwealth Bank of Australia www.commbank.com.au

Personal banking including accounts, credit cards and home loans - CommBank

Safari is using an encrypted connection to www.commbank.com.au.

Encryption with a digital certificate keeps information private as it's sent to or from the https website www.commbank.com.au.

Symantec Corporation has identified www.commbank.com.au as being owned by Commonwealth Bank of Australia in SYDNEY, New South Wales, AU.

VeriSign Class 3 Public Primary Certification Authority - G5

Symantec Class 3 EV SSL CA - G3

www.commbank.com.au

Certificate

www.commbank.com.au

Issued by: Symantec Class 3 EV SSL CA - G3

Expires: Saturday, 27 February 2016 at 10:59:59 AM Australian Eastern Daylight Time

✓ This certificate is valid

Trust

Details

Subject Name

Inc. Country AU

Business Category Private Organization

Serial Number 123 123 124

Country AU

Postal Code 2000

State/Province New South Wales

Locality SYDNEY

Street Address 201 SUSSEX S T

Organization Commonwealth Bank of Australia

Organizational Unit CBA Business System Hosting

Common Name www.commbank.com.au

Issuer Name

Country US

Organization Symantec Corporation

Organizational Unit Symantec Trust Network

Common Name Symantec Class 3 EV SSL CA - G3

Serial Number 1A 9F E9 4B 03 9D E2 9A B6 15 56 69 60 3E 98 AE

Version 3

Signature Algorithm SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.1 )

Parameters none

Not Valid Before Monday, 4 May 2015 at 10:00:00 AM Australian Eastern Standard Time

Not Valid After Saturday, 27 February 2016 at 10:59:59 AM Australian Eastern Daylight Time

Public Key Info

Algorithm RSA Encryption ( 1.2.840.113549.1.1.1 )

Parameters none

Public Key 256 bytes : CA B4 74 93 E8 00 22 10 ...

Exponent 65537

Key Size 2048 bits

Key Usage Encrypt, Verify, Wrap, Derive

Signature 256 bytes : 95 32 C3 F0 62 F1 F8 F1 ...

Hide Certificate

OK

Log on

Locate us

1 Stuff I like

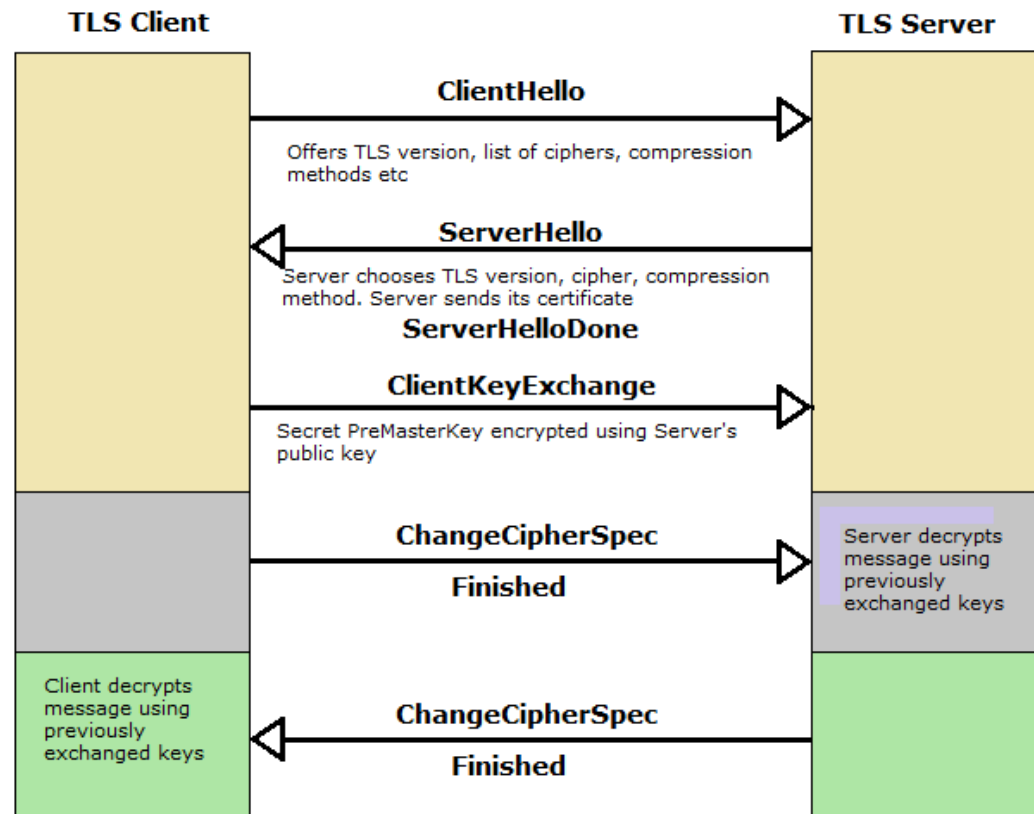
% Rates & fees

Latest offers

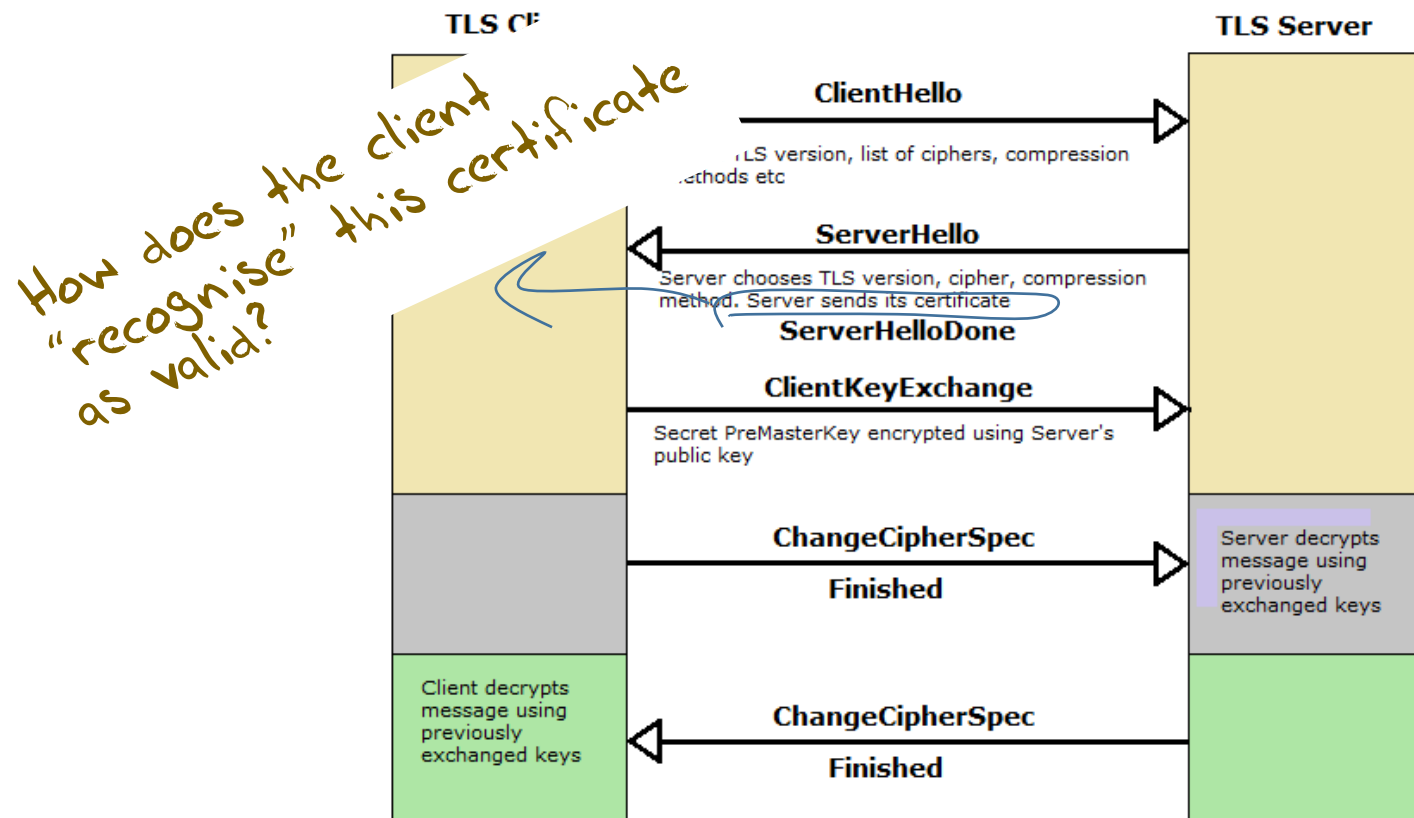
# Domain Name Certification

- The Commonwealth Bank of Australia has generated a key pair
- And they passed a certificate signing request to a company called “Verisign”
- Who is willing to vouch (in a certificate) that the entity who goes by the domain name of [www.commbank.com.au](http://www.commbank.com.au) has a certain public key value
- So if I can associate this public key with a connection then I have a high degree of confidence that I’ve connected to [www.commbank.com.au](http://www.commbank.com.au), as long as I am prepared to trust Verisign and the certificates that they issue

# TLS Connections

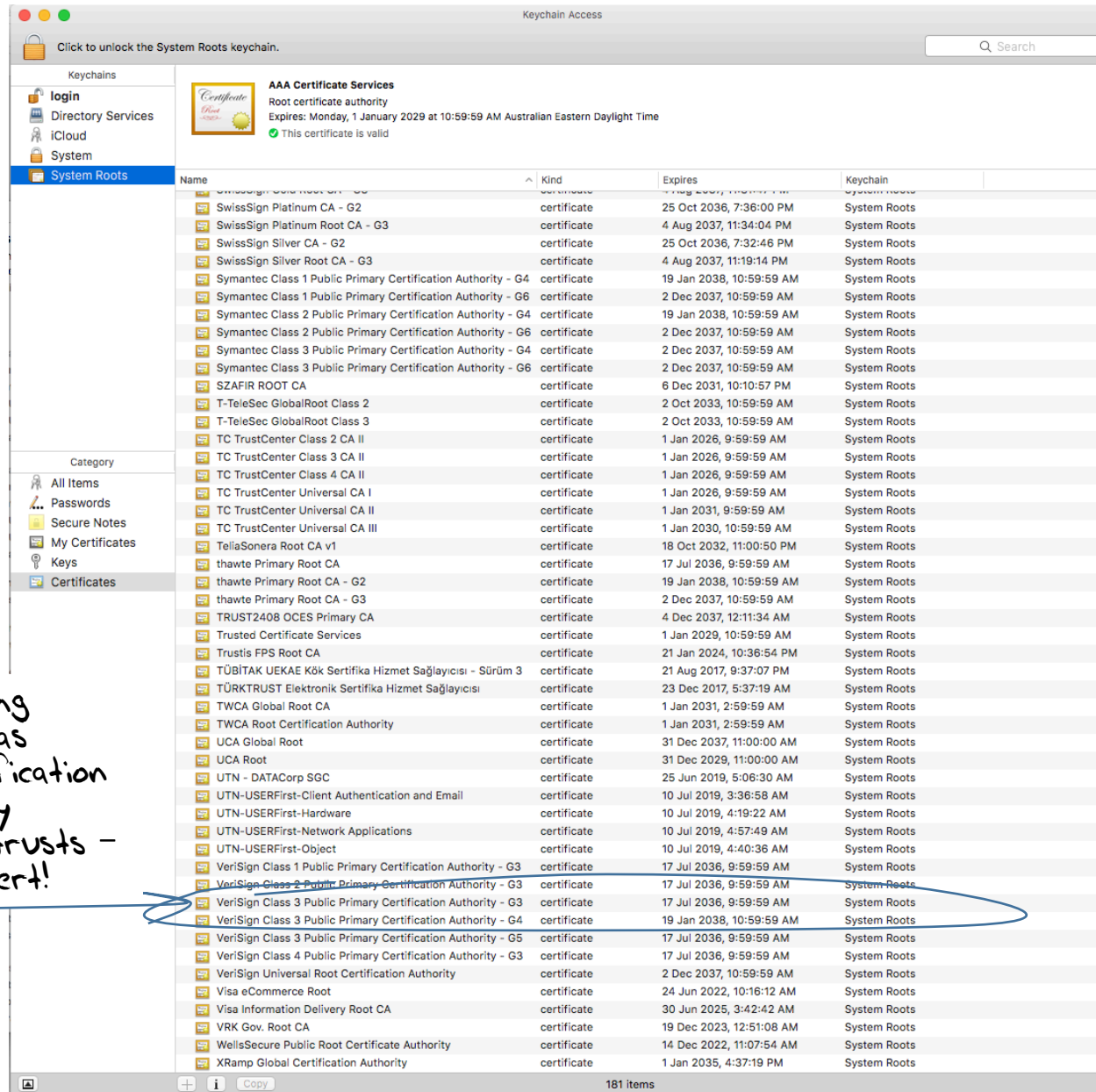


# TLS Connections





# Local Trust

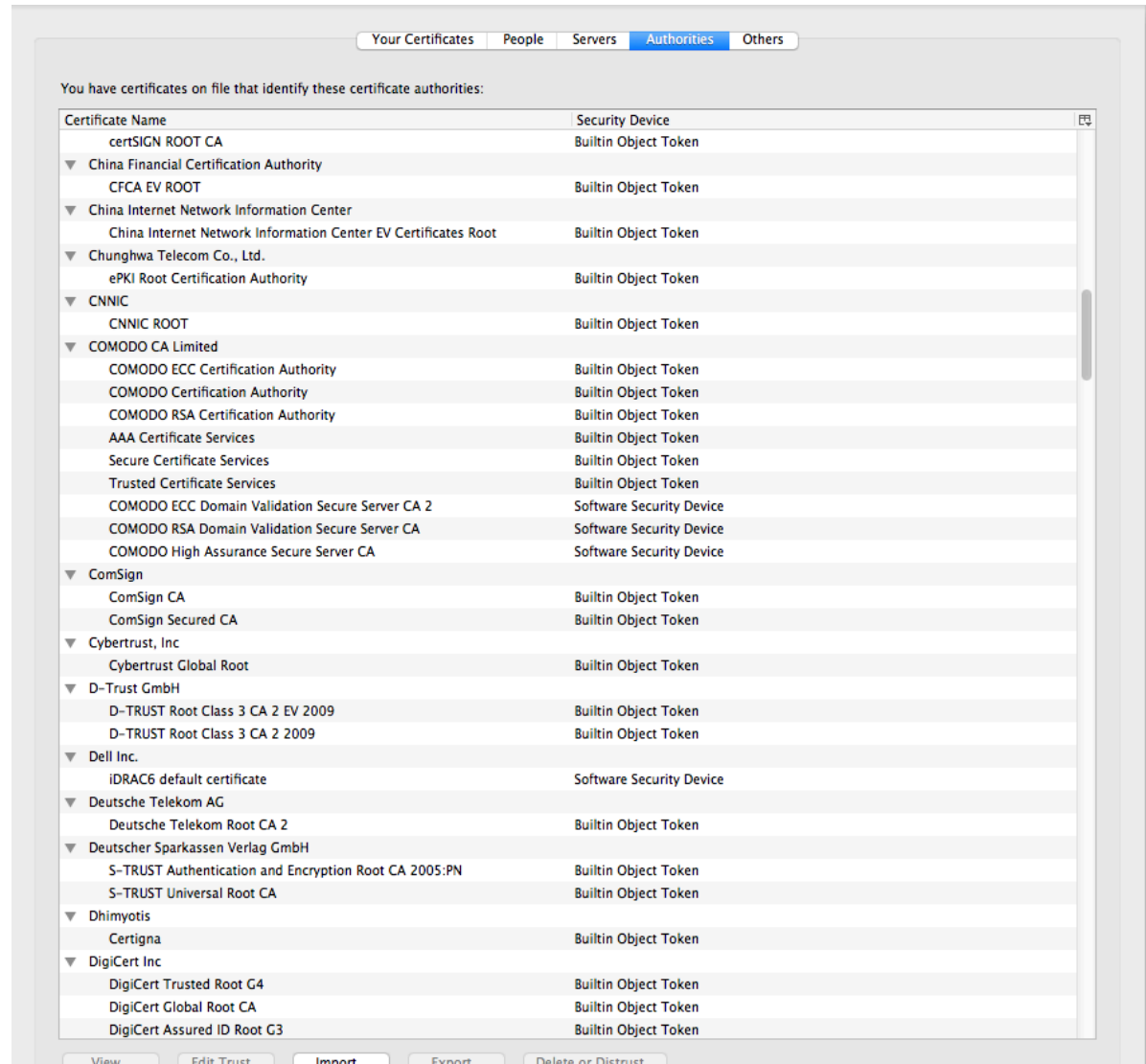


The cert i'm being asked to trust was issued by a certification authority that my browser already trusts - so i trust that cert!

# Local Trust

That's a big list of people to Trust

Are they all trustable?



# Local Trust

That's a big list of people to Trust

Are they all trustable?

*Evidently Not!*

Your Certificates People Servers Authorities Others

You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
certSIGN ROOT CA	Builtin Object Token
▼ China Financial Certification Authority	
CFCA EV ROOT	Builtin Object Token
▼ China Internet Network Information Center	
China Internet Network Information Center EV Certificates Root	Builtin Object Token
▼ Chunghwa Telecom	
ePKI Root Certif	
▼ CNNIC	
CNNIC ROOT	
COMODO CA Limit	
COMODO ECC C	
COMODO Certif	
COMODO RSA C	
AAA Certificate	
Secure Certificat	
Trusted Certificat	
COMODO ECC D	
COMODO RSA D	
COMODO High	
▼ ComSign	
ComSign CA	
ComSign Secure	
▼ Cybertrust, Inc	
Cybertrust Glob	
▼ D-Trust GmbH	
D-TRUST Root C	
D-TRUST Root C	
▼ Dell Inc.	
iDRAC6 default	
▼ Deutsche Telekom	
Deutsche Telekom	
▼ Deutscher Sparkas	
S-TRUST Authen	
S-TRUST Univer	
▼ Dhimytotis	
Certigna	
▼ DigiCert Inc	
DigiCert Truste	
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.

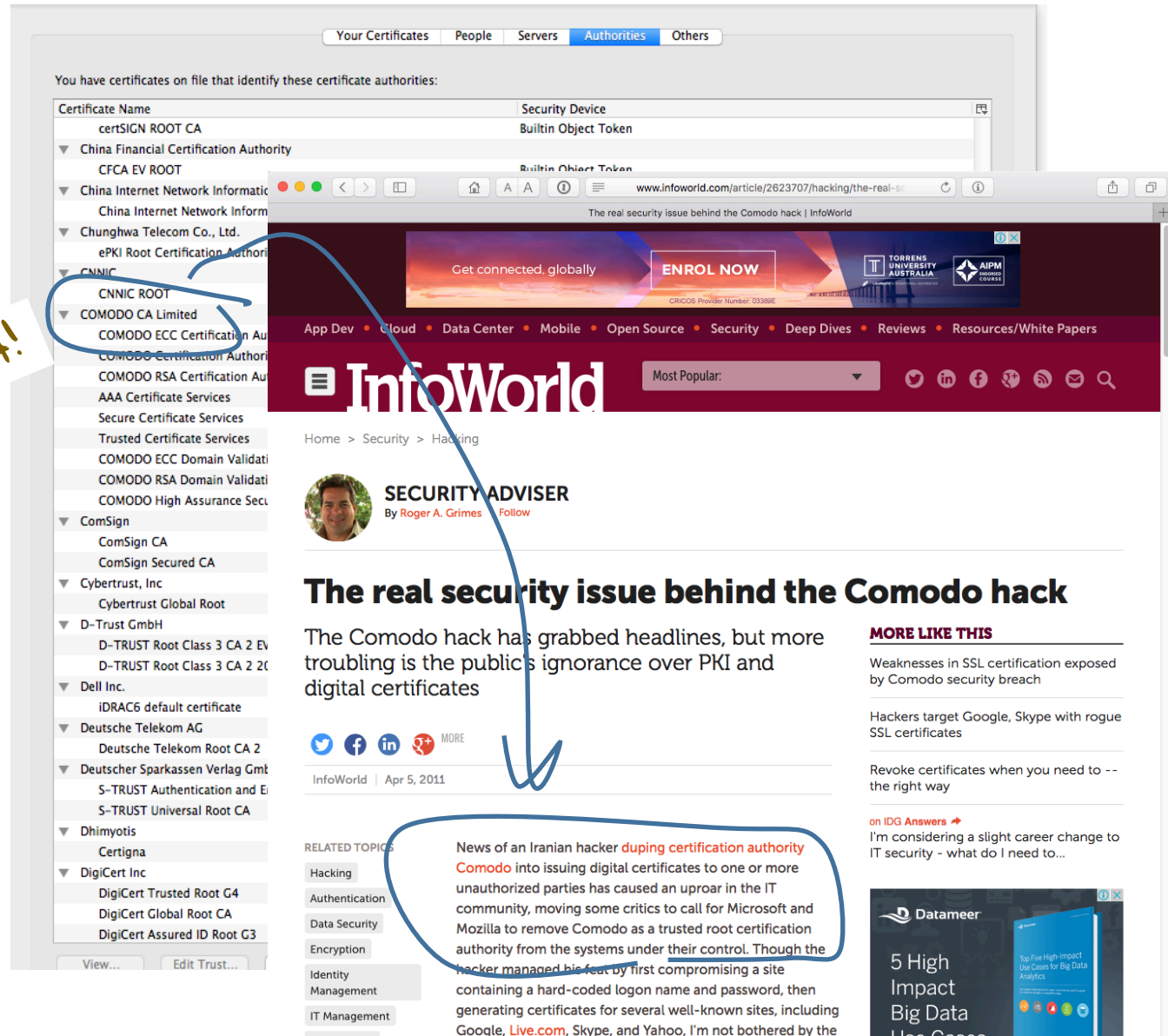
# Local Trust

That's a big list of people to Trust

## Are they all trustable?

rustable?

*Evidently Not!*



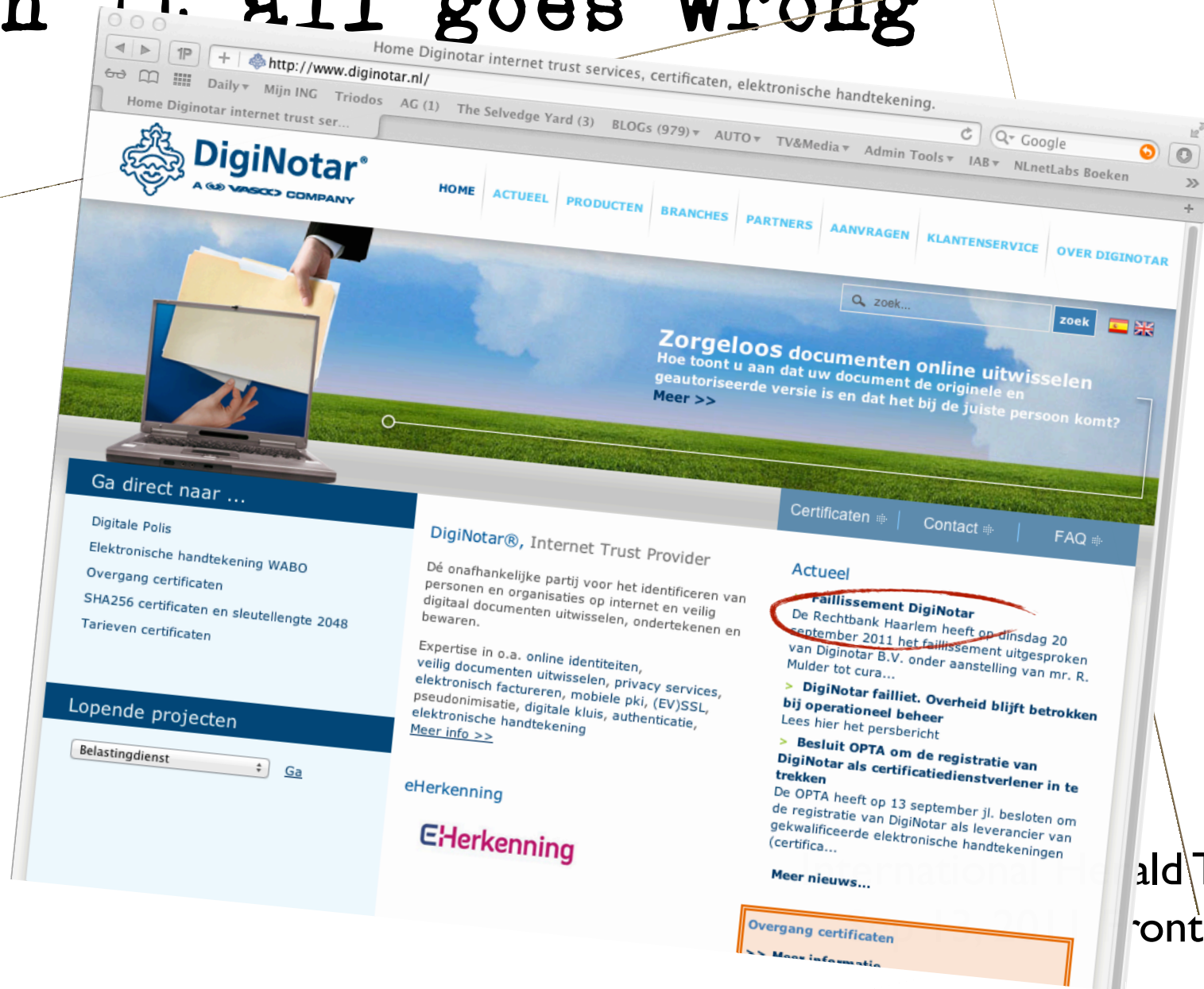
With unpleasant consequences  
when it all goes wrong

**With unpleasant consequences  
when it all goes wrong**

International Herald Tribune  
Sep 13, 2011 Front Page



With unpleasant consequences  
when it all goes wrong



ald Tribune  
ront Page

# What's going wrong here?

- The TLS handshake cannot specify WHICH CA should be used to validate the digital certificate
- 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.
- *WOW! That's astonishingly bad!* 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.
- **WOW! That's astonishingly bad!** Any CA will allow ANY CA to be used to validate



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

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

# 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

Than what 'wins' in the market?



Sustainable  
Resilient

Secure

Privacy

Trusted

cheap!

# Where now?

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



The screenshot shows the Let's Encrypt website homepage. At the top, there is a navigation bar with the Let's Encrypt logo on the left and links for Blog, Technology, Sponsors, Support, and About on the right. The main content area features a large, colorful geometric background with a central white box containing the text: "Let's Encrypt is a new Certificate Authority: It's free, automated, and open. In Limited Beta". Below this, there are two sections: "FROM OUR BLOG" and "MAJOR SPONSORS". The "FROM OUR BLOG" section includes a date "Nov 12, 2015", a link "Public Beta: December 3, 2015", and a paragraph stating that Let's Encrypt will enter Public Beta on December 3rd, 2015, and that systems will be open to anyone who requests a certificate. The "MAJOR SPONSORS" section displays logos for Mozilla, Akamai, Cisco, EFF, IdenTrust, and the Internet Society.

**Let's Encrypt** LINUX FOUNDATION COLLABORATIVE PROJECTS

Blog Technology Sponsors Support About

Let's Encrypt is a new Certificate Authority:  
**It's free, automated, and open.**  
In Limited Beta

**FROM OUR BLOG**

Nov 12, 2015  
[Public Beta: December 3, 2015](#)  
Let's Encrypt will enter Public Beta on December 3rd, 2015. Once we've entered Public Beta our systems will be open to anyone who would like to request a certificate.  
[Read more](#)

**MAJOR SPONSORS**

mozilla Akamai CISCO  
EFF IdenTrust Internet Society

# Where now?

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?

We're probably going to find out real soon!

The screenshot shows the Let's Encrypt website header with the Linux Foundation Collaborative Projects logo and navigation links (Blog, Technology, Sponsors, Support, About). The main content area features a large, semi-transparent box with handwritten text. Below this, the 'FROM OUR BLOG' section includes a post dated Nov 12, 2015, titled 'Public Beta: December 3, 2015', with a brief description of the public beta launch and a 'Read more' link. The 'MAJOR SPONSORS' section lists logos for Mozilla, Akamai, Cisco, EFF, IdenTrust, and the Internet Society.

LINUX FOUNDATION COLLABORATIVE PROJECTS

Let's Encrypt

Blog Technology Sponsors Support About

Let's Encrypt is a new Certificate Authority:  
It's free, automated and open.  
In Limited Beta

FROM OUR BLOG

Nov 12, 2015

[Public Beta: December 3, 2015](#)

Let's Encrypt will enter Public Beta on December 3rd, 2015. Once we've entered Public Beta our systems will be open to anyone who would like to request a certificate.

[Read more](#)

MAJOR SPONSORS

mozilla Akamai CISCO

EFF IdenTrust Internet Society

# Where now?

Option B: White Listing and Pinning with HSTS

[https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport\\_security\\_state\\_static.json](https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json)

# Where now?

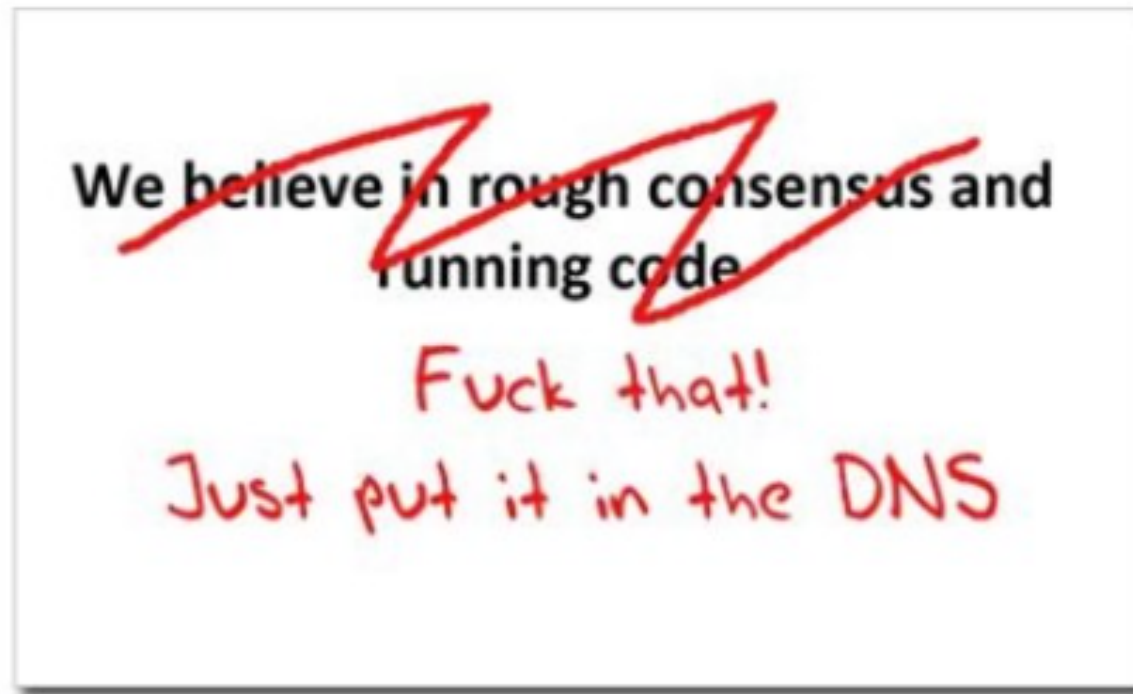
Option B: White Listing and Pinning with HSTS

[https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport\\_security\\_state\\_static.json](https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json)  
This approach appears to be completely unscalable!



# Where now?

Option C: Use the DNS!



# Seriously

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

# Seriously

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

- Why not query the DNS for the HSTS record (pinning record)?

# Seriously

Where better to find out the public key associated with a DNS name 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?

# Seriously

Where better to find out the public key associated with a DNS name 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?

# Seriously

Where better to find out the public key associated with a DNS name 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 domain name public key cert as a simple self-signed cert?

# Seriously

Where better to find out the public key associated with a DNS name than to query 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 cert?
- Why not query the DNS for the domain name public key cert as a simple self-signed cert?

*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 2012

**The DNS-Based Authentication of Named Entities (DANE)  
Transport Layer Security (TLS) Protocol: TLSA**

Abstract

Encrypted communication on the Internet often uses Transport Layer Security (TLS), which depends on third parties to certify the keys used. 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.



# DANE

## TLSA RR

### 2.3. TLSA RR Examples

An example of a hashed (SHA-256) association of a PKIX CA certificate:

```
_443._tcp.www.example.com. IN TLSA (  
  0 0 1 d2abde240d7cd3ee6b4b28c54df034b9  
        7983ald16e8a410e4561cb106618e971 )
```

CA Cert Hash

An example of a hashed (SHA-512) subject public key association of a PKIX end entity certificate:

```
_443._tcp.www.example.com. IN TLSA  
  1 1 2 92003ba34942dc74152e2f2c408d29ec  
        a5a520e7f2e06bb944f4dca346baf63c  
        1b177615d466f6c4b71c216a50292bd5  
        8c9ebdd2f74e38fe51ffd48c43326cbc )
```

EE Cert Hash

An example of a full certificate association of a PKIX trust anchor:

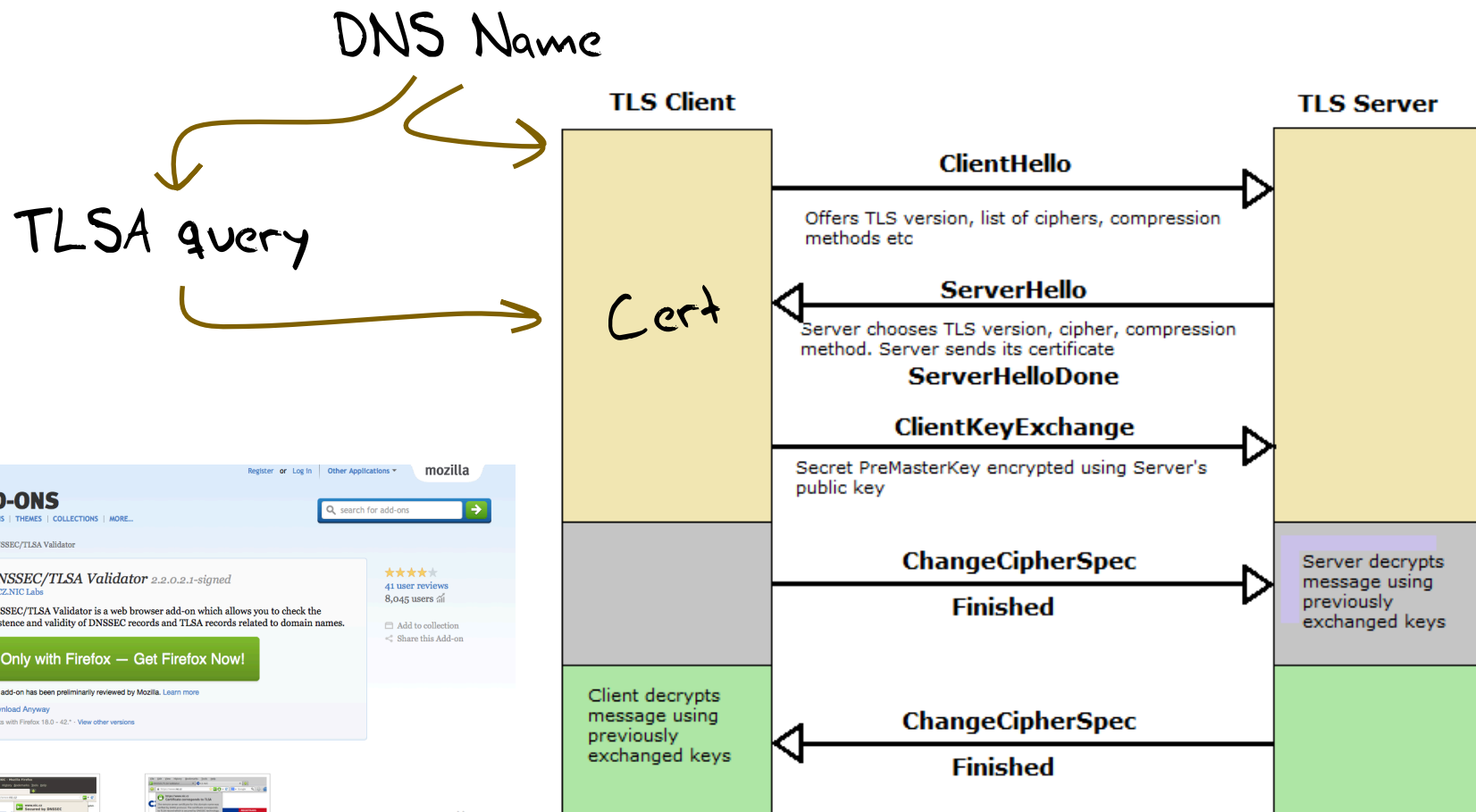
```
_443._tcp.www.example.com. IN TLSA  
  2 0 0 30820307308201efa003020102020... )
```

Trust Anchor

# 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



ADD-ONS  
EXTENSIONS | THEMES | COLLECTIONS | MORE...

Extensions » DNSSEC/TLSA Validator

**DNSSEC/TLSA Validator** 2.2.0.2.1-signed  
by CZ.NIC Labs

41 user reviews  
8,045 users all

Only with Firefox — Get Firefox Now!

This add-on has been preliminarily reviewed by Mozilla. [Learn more](#)

[Download Anyway](#)  
Works with Firefox 18.0 - 42.\* · [View other versions](#)

[Add to collection](#)  
[Share this Add-on](#)

[Support E-mail](#)

**About this Add-on**

DNSSEC/TLSA Validator allows you to check the existence and validity of DNS Security Extensions (DNSSEC) signed records. If a valid DNSSEC chain related to the domain is found the plug-in will also check for the existence of Transport Layer Security Association (TLSA) records. TLSA records store hashes of remote server TLS/SSL certificates. The authenticity of a TLS/SSL certificate for a domain name is verified by the DANE protocol (RFC 6698). DNSSEC and TLSA validation results are displayed by using several icons. Clicking on a given icon symbol reveals more detailed information.

DNSSEC/TLSA Validator uses external libraries to resolve and validate DNSSEC/TLSA signatures and to verify HTTPS server certificates. More info is available on the [www.dnssec-validator.cz](http://www.dnssec-validator.cz) page.

[Add-on home page](#)  
[Support site](#)  
[Support E-mail](#)

Version 2.2.0.2.1-signed Info  
Last Updated: May 15, 2015  
Released under GNU General Public License, version 3.0

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

# DNSSEC Interlocking Signatures

. (root)

- . Key-Signing Key – signs over
  - . Zone-Signing Key – signs over
    - DS for .com (Key-Signing Key)

.com

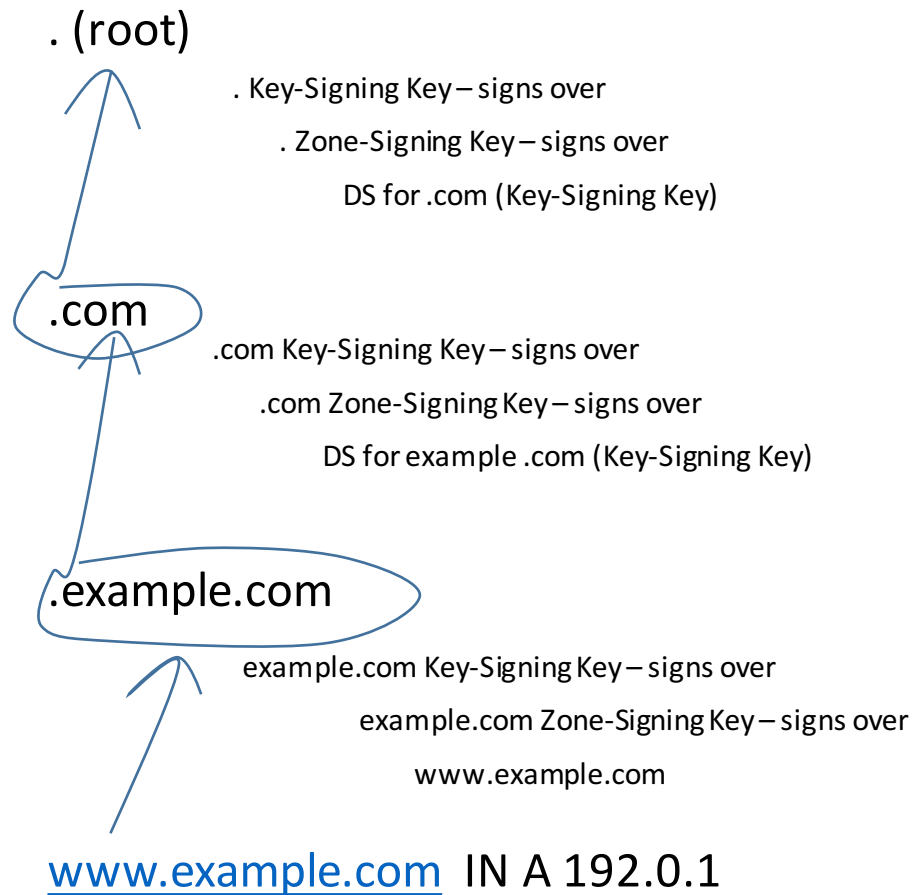
- .com Key-Signing Key – signs over
  - .com Zone-Signing Key – signs over
    - DS for example .com (Key-Signing Key)

.example.com

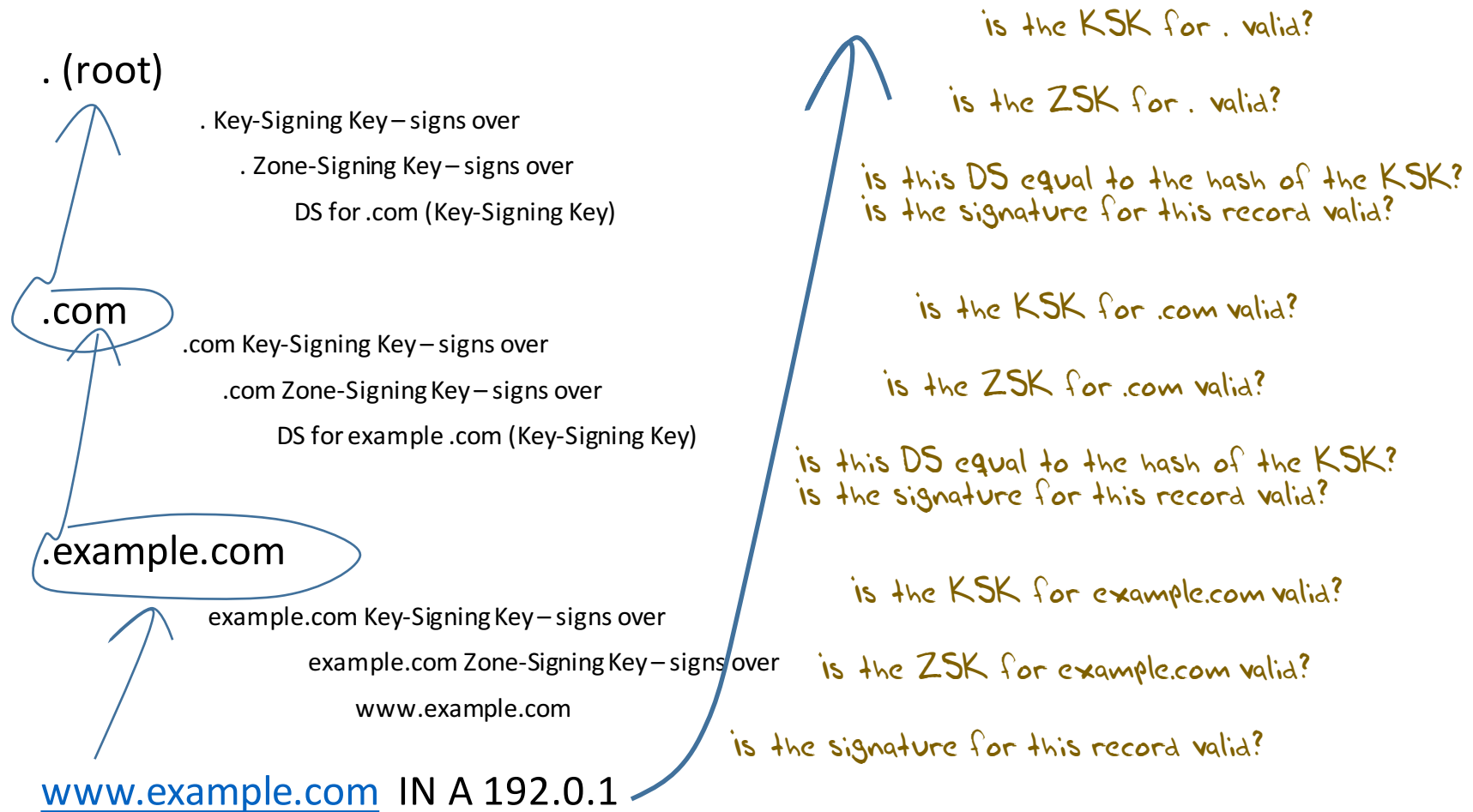
- example.com Key-Signing Key – signs over
  - example.com Zone-Signing Key – signs over
    - www.example.com

www.example.com

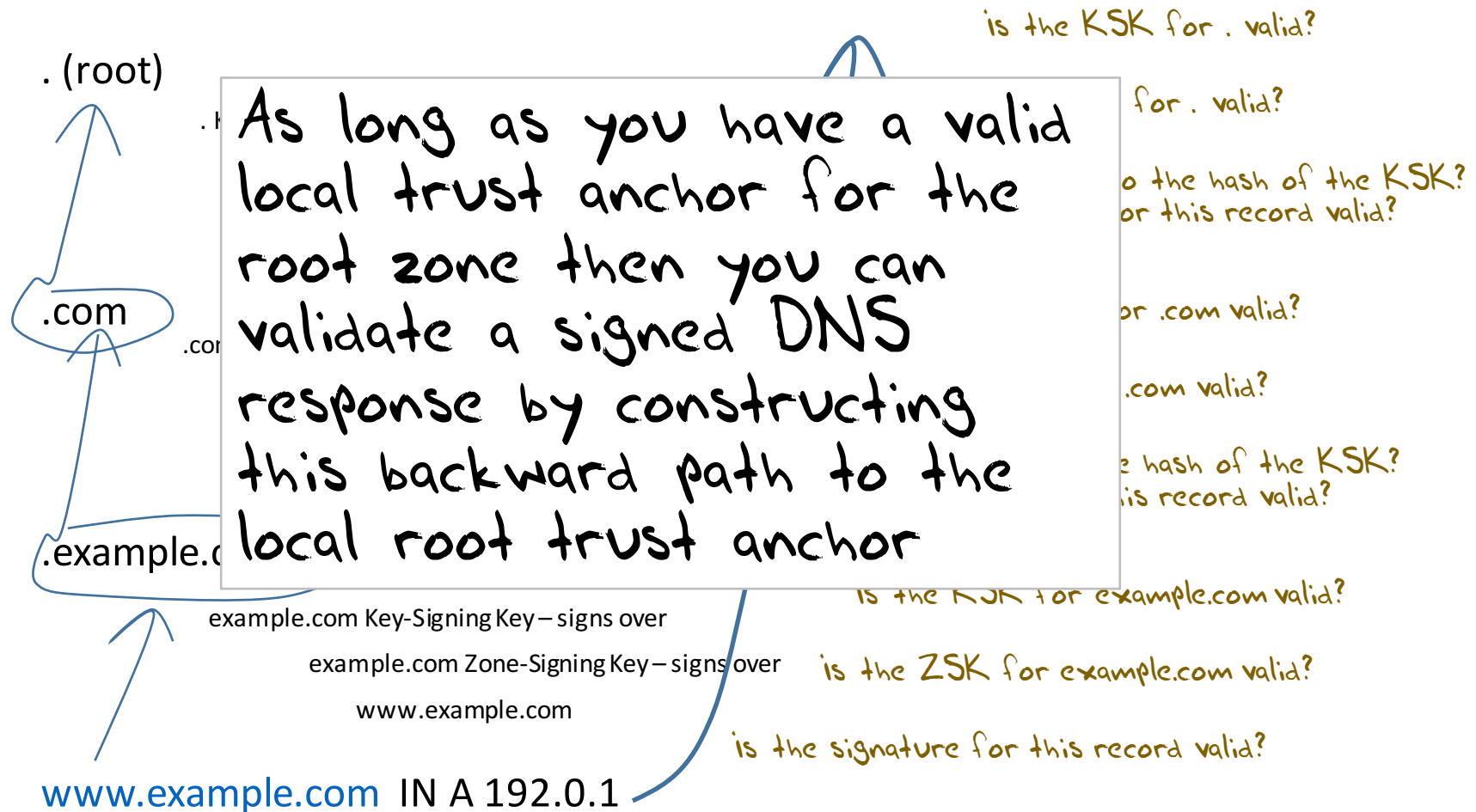
# DNSSEC Interlocking Signatures



# DNSSEC Interlocking Signatures



# DNSSEC Interlocking Signatures



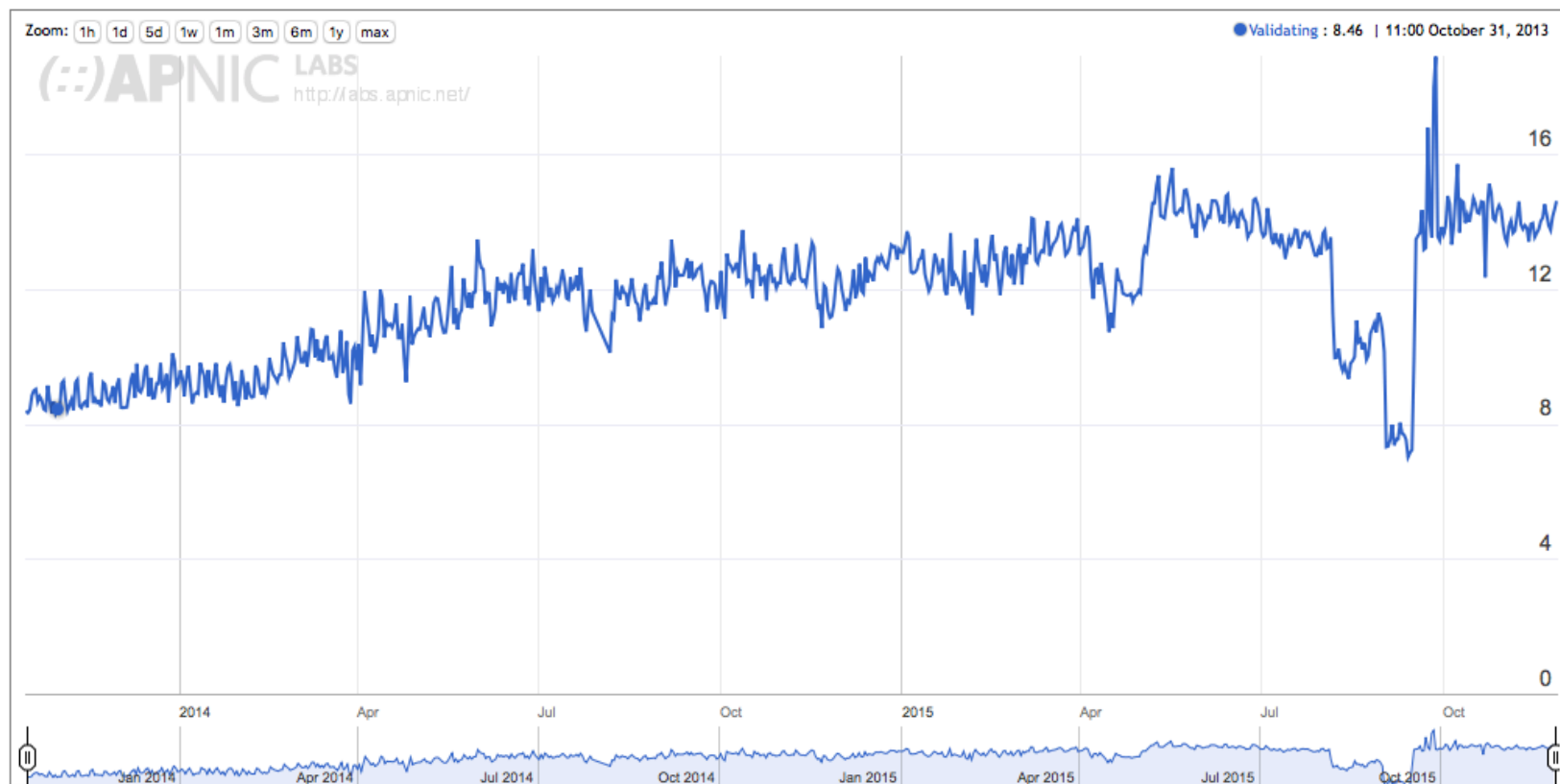


Do we do DNSSEC  
Validation?

# Do we do DNSSEC Validation?

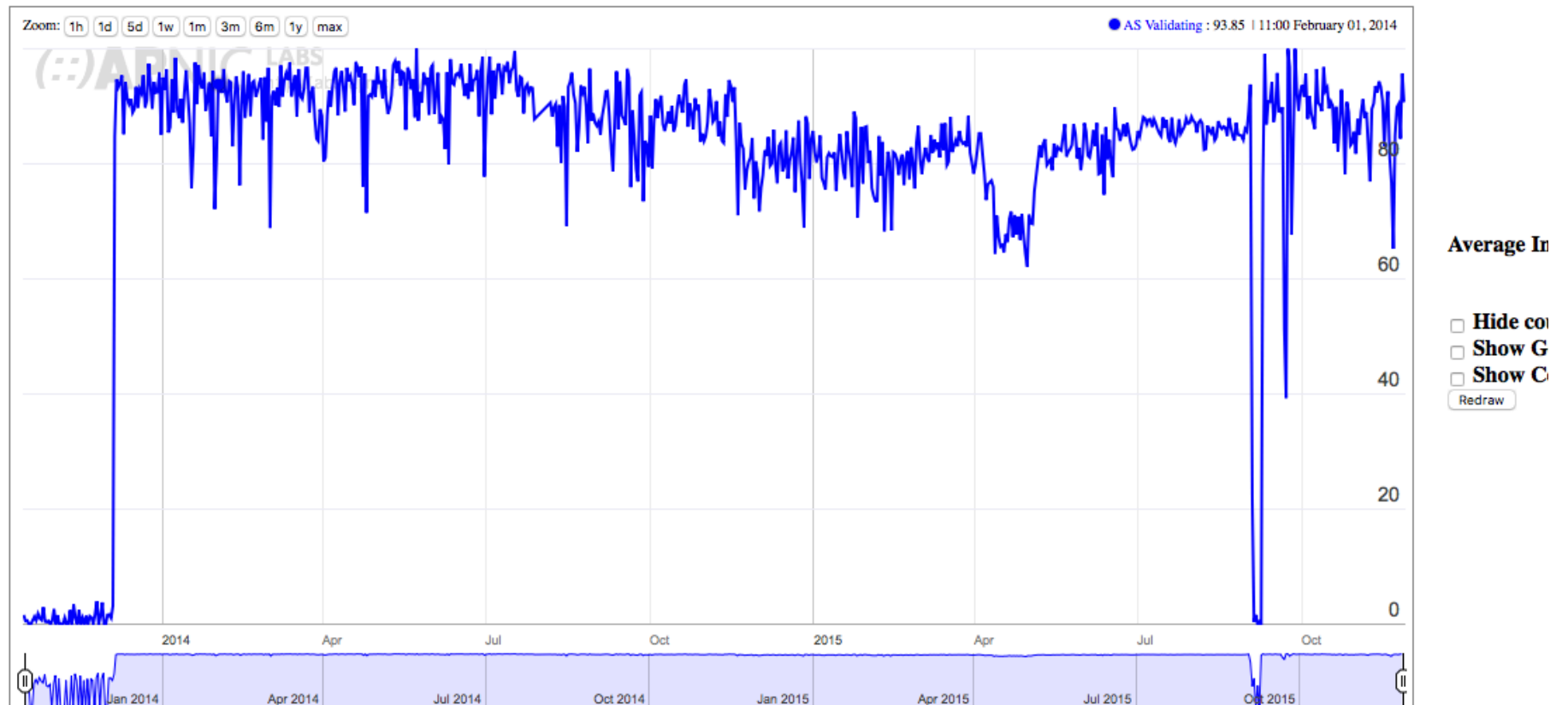
- Surprisingly, there is a lot of it out there!

## Use of DNSSEC Validation for World (XA)



# Optus has been running it for 2 years!

## DNSSEC Country Deployment for AS4804: MPX-AS Microplex PTY LTD, Australia (AU)



# So we're done - right?

- Um – well, if we're not done, we're well on the way!

# So we're done - right?

- Um – well, if we're not done, we're well on the way!

But:

- We need to improve the use of DNSSEC validation in resolvers
- We need to load DNSSEC validation as a library for applications to use directly
- And we need to improve our day-to-day operational practices in managing DNSSEC
- And hopefully that will clear the path for the widespread adoption of DANE
- Because we have no other way to nail down the CA pinning problem in a reliable and secure manner

# Operational Practices?

- Key Management
  - Registration of the DS record in the parent zone
  - Regular Key rotation

# Rolling Keys

## Rolling a ZSK for a zone:

(the issue here is that you need to be aware that resolvers will cache data, so any sudden move may isolate you from the net!)

- Add the new ZSK to the DNSKEY RRset for the zone  
(and sign across it with the KSK)
- Pause for breath (or at least a TTL)
- Remove the zone's old RRsigs (signed by the outgoing ZSK) and replace them with RRsigs signed by the new ZSK
- Pause for another breath
- Remove the old ZSK from the DNSKEY RRset

# Rolling Keys

Rolling a KSK for a zone:

- Add the new KSK to the DNSKEY RRset for the zone  
(and sign across it with both the old and new KSKs)
- Pause for breath (or at least a TTL)
- Replace the parent's DS record for this zone with the DS record for the new KSK
- Pause for another breath (TTL)
- Remove the old KSK (and its RRSIG) from the DNSKEY RRset



# But What about the Root Keys?

- The Root Key ZSK is just like any other ZSK
  - it's rolled every three months
  - And nobody appears to have a problem with this!

# But What about the Root Keys?

- The Root Zone KSK is different

# Why is the Root Zone KSK different?

- The KSK Public Key is used as the DNSSEC Validation trust anchor
  - This key is the root of all trust in the DNSSEC framework
  - It is distributed everywhere as “configuration data”
  - Most of the time the KSK itself is kept offline in highly secure facilities

# But What about the Root Keys?

- The Root Zone KSK is different
- There is no “parent authority”
- And there is no real way to disseminate a new KSK other than using the DNS itself
- So rolling the KSK means that we have to use an “old signs new” approach to transitive trust  
(RFC 5011)
- And there is no Plan B here!

# Five Years Ago...



[MAIN MENU](#) [MY STORIES: 25](#) [FORUMS](#) [SUBSCRIBE](#) [JOBS](#) [ARS CONSORTIUM](#)

## RISK ASSESSMENT / SECURITY & HACKTIVISM

### DNS root zone finally signed, but security battle not over

The root of the DNS hierarchy is now protected with a cryptographic signature ...

by Iljitsch van Beijnum - Jul 16, 2010 11:28pm CEST

[Share](#) [Tweet](#) [13](#)

Yesterday, the DNS root zone was signed. This is an important step in the [deployment of DNSSEC](#), the mechanism that will finally secure the DNS against manipulation by malicious third parties.

## ICANN's First DNSSEC Key Ceremony for the Root Zone

in [f](#) [t](#) [d](#) [e](#) [m](#) [+](#)

The global deployment of [Domain Name System Security Extensions \(DNSSEC\)](#) will achieve an important milestone on June 16, 2010 as [ICANN](#) hosts the first production [DNSSEC](#) key ceremony in a high security data centre in Culpeper, VA, outside of Washington, DC.

## Schneier on Security

[Blog](#)

[Newsletter](#)

[Books](#)

[Essays](#)

[News](#)

[Schedule](#)

[Crypto](#)

[About Me](#)

[← Pork-Filled Counter-Islamic Bomb Device](#)

[Security Vulnerabilities of Smart Electricity Meters](#)

### DNSSEC Root Key Split Among Seven People

The DNSSEC root key has been [divided](#) among seven people:

Part of ICANN's security scheme is the Domain Name System Security, a security protocol that ensures Web sites are registered and "signed" (this is the security measure built into the Web that ensures when you go to a URL you arrive at a real site and not an identical pirate site). Most major servers are a part of DNSSEC, as it's known, and during a major international attack, the system might sever connections between important servers to contain the damage.

, VA - location of first [DNSSEC](#) key signing ceremony

# The Eastern KSK Repository



Secure data center in Culpeper, VA - location of first DNSSEC key signing ceremony

# The Western KSK Repository



El Segundo, California \*



# The Ultra Secret Third KSK Repository in Amsterdam





# Five Years Ago...

Root DNSSEC Design Team

F. Ljunggren  
Kirei  
T. Okubo  
VeriSign  
R. Lamb  
ICANN  
J. Schlyter  
Kirei  
May 21, 2010

## DNSSEC Practice Statement for the Root Zone KSK Operator

### Abstract

This document is the DNSSEC Practice Statement (DPS) for the Root Zone Key Signing Key (KSK) Operator. It states the practices and provisions that are used to provide Root Zone Key Signing and Key Distribution services. These include, but are not limited to: issuing, managing, changing and distributing DNS keys in accordance with the specific requirements of the U.S. Department of Commerce.

Root Zone KSK Operator DPS

May 2010

### 6.3. Signature format

The cryptographic hash function used in conjunction with the signing algorithm is required to be sufficiently resistant to preimage attacks during the time in which the signature is valid.

The RZ KSK signatures will be generated by encrypting SHA-256 hashes using RSA [RFC5702].

### 6.4. Zone signing key roll-over

ZSK rollover is carried out quarterly automatically by the Root Zone ZSK Operator's system as described in the Root Zone ZSK Operator's DPS.

### 6.5. Key signing key roll-over

Each RZ KSK will be scheduled to be rolled over through a key ceremony as required, or after 5 years of operation.

RZ KSK roll-over is scheduled to facilitate automatic updates of resolvers' Trust Anchors as described in RFC 5011 [RFC5011].

After a RZ KSK has been removed from the key set, it will be retained after its operational period until the next scheduled key ceremony, when the private component will be destroyed in accordance with section 5.2.10.

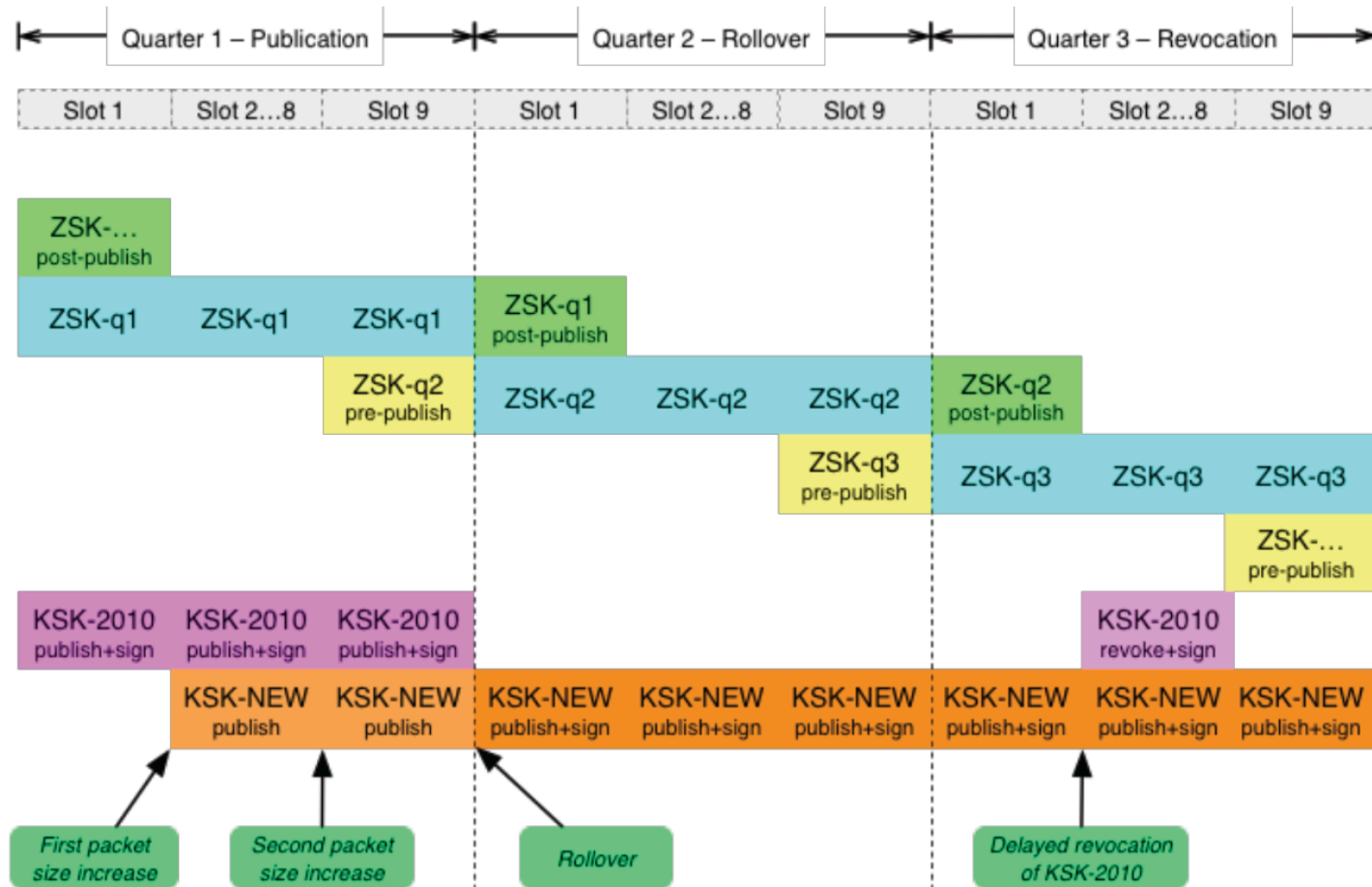
# Rolling the KSK?

- All DNS resolvers that perform validation of DNS responses use a local copy of the KSK
- They will need to load a new KSK public key and replace the existing trust anchor with this new value at the appropriate time
- This key roll could have a public impact, particularly if DNSSEC-validating resolvers do not load the new KSK
  - These resolvers will go dark and will not resolve signed responses

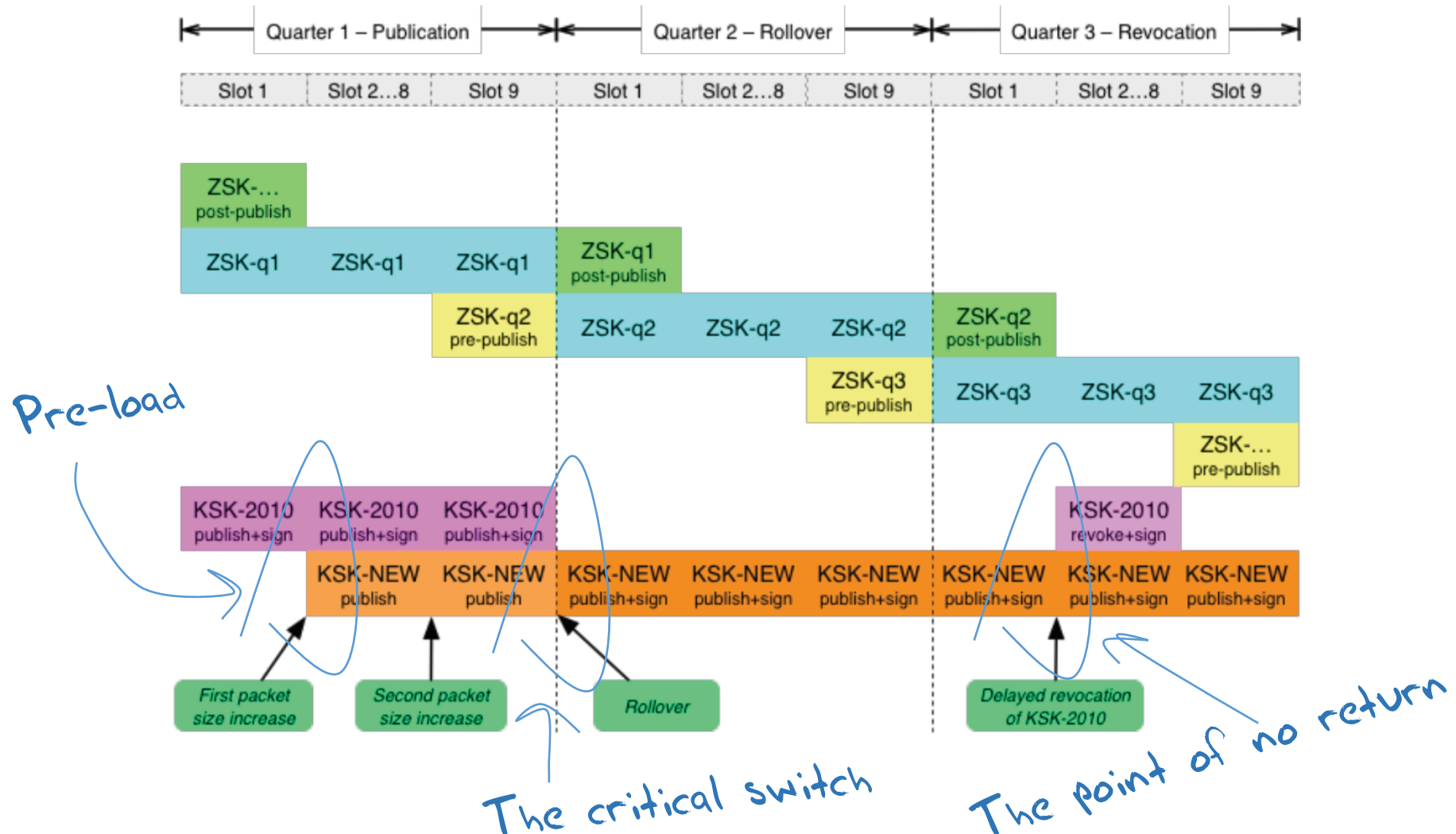
# Easy, Right?

- Publish a new KSK and include it in DNSKEY responses
- Use the new KSK to sign the ZSK, as well as the old KSK signature
  - Resolvers use old-signs-over-new to pick up the new KSK, validate it using the old KSK, and replace the local trust anchor material with the new KSK
- Withdraw the old signature signed via the old KSK
- Revoke the old KSK

# The RFC5011 Approach



# The RFC5011 Approach



# Just Like Last Time?

## Roll Over and Die?

February 2010

**George Michaelson  
Patrik Wallström  
Roy Arends  
Geoff Huston**

In this month's column I have the pleasure of being joined by George Michaelson, Patrik Wallström and Roy Arends to present some critical results following recent investigations on the behaviour of DNS resolvers with DNSSEC. It's a little longer than usual, but I trust that its well worth the read.

-- *Geoff*

It is considered good security practice to treat cryptographic keys with a healthy level of respect. The conventional wisdom appears to be that the more material you sign with a given private key the more clues you are leaving behind that could enable some form of effective key guessing. As RFC4641 states: "the longer a key is in use, the greater the probability that it will have been compromised through carelessness, accident, espionage, or cryptanalysis." Even though the risk is considered slight if you have chosen to use a decent key length, RFC 4641 recommends, as good operational practice, that you should "roll" your key at regular intervals. Evidently it's a popular view that fresh keys are better keys!

The standard practice for a "staged" key rollover is to generate a new key pair, and then have the two public keys co-exist at the publication point for a period of time, allowing relying parties, or clients, some period of time to pick up the new public key part. Where possible during this period, signing is performed twice, once with each key, so that the validation test can be performed using either key. After an appropriate interval of parallel operation the old key pair can be deprecated and the new key can be used for signing.

This practice of staged rollover as part of key management is used in X.509 certificates, and is also used in signing the DNS, using DNSSEC. A zone operator who wants to roll the DNSSEC key value would provide notice of a pending key change, publish the public key part of a new key pair, and then use the new and old private keys in parallel for a period. On the face of it, this process sounds quite straightforward.

What could possibly go wrong?

# But that was then...

## And this is now:

- Resolvers are now not so aggressive in searching for alternate validation paths when validation fails  
(as long as resolvers keep their code up to date, which everyone does – right?)
- And now we ***all*** support RFC5011 key roll processes
- And ***everyone*** can cope with large DNS responses

So all this will go without a hitch

Nobody will even notice the KSK roll at the root

# But that was then...

## And this is now:

- Resolvers are now not so aggressive in searching for alternate validation paths when validation fails  
(as long as the path is up to date, which everyone does – right?)
- And now
- And **every**
- 1 key roll processes
- large DNS responses

So all this will go without a hitch

Nobody will even notice the KSK roll at the root



# What we all should be concerned about...

That resolvers who validate DNS responses will fail to pick up the new DNS root key automatically

- i.e. they do not have code that follows RFC5011 procedures for the introduction of a new KSK

The resolvers will be unable to receive the larger DNS responses that will occur during the dual signature phase of the rollover

# Technical Concerns

- Some DNSSEC validating resolvers do not support RFC5011
  - How many resolvers may be affected in this way?
  - How many users may be affected?
  - What will the resolvers do when validation fails?
    - Will they perform lookup 'thrashing'?
  - What will users do when resolvers return SERVFAIL?
    - How many users will redirect their query to a non-validating resolver?

# Technical Concerns

- Some DNSSEC validating resolvers do not support RFC5011

- How many resolvers may be affected?

- How many

*Really hard to test this in the wild with recursive resolvers*

recursion tails?

'lookup thrashing'

What will users do when resolvers return SERVFAIL?

- How many users will redirect their query to a non-validating resolver

# Some Observations - 1

There is a LOT of DNSSEC validation out there!

- 87% of all queries have DNSSEC-OK set
- 30% of all DNSSEC-OK queries attempt to validate the response
- 25% of end users are using DNS resolvers that will validate what they are told
- 12% of end users don't believe bad validation news and turn to other non-validating resolvers when validation fails.

# Some Observations - 2

ECDSA is viable – sort of

- 1 in 5 clients who use resolvers that validate RSA-signed responses are unable to validate the same response when signed using ECDSA
- But they fail to “unsigned” rather than “invalid” so it’s a (sort of) safe fail

# Some Observations - 3

The larger DNS responses will probably work, but not for everyone

- The “fall back to TCP” will rise to 6% of queries when the response size get to around 1,350 octets
- But around 16% of visible resolvers appear not to use TCP at all
- So the DNS failure rate appears to rise by around 1 - 2 % of end users

BUT .org currently runs at 1,650 octets and nobody is screaming failure

- So it will probably work

# Some Observations - 4

We can't measure automated key take up

- We can't see how many resolvers fail to use RFC5011 notices to pick up the new KSK as a Trust Anchor in advance
- We will only see it via failure on key roll

# Where are we?

- A key roll of the Root Zone KSK will cause some resolvers to fail:
  - Resolvers who do not pick up the new key in the manner described by RFC5011
  - Resolvers who cannot receive a DNS response of ~1,300 octets
- Many users who use these failing resolvers will just switch over to use a non-validating resolver
- A small pool of users will be affected with no DNS



# What can I do?

Check your recursive resolver config!

# Good Dog!

```
# // recursive resolver configuration - Bind
...
managed-keys {
    . initial-key 257 3 5 "AwEAAfdqNV
      JMRMzrppU1WnNW0PWrGn4x9dPg
...
    =,,i };
```

# Bad Dog!

```
# // recursive resolver configuration - Bind
...
trusted-keys {
    . 257 3 5 "AwEAAfdqNV
        JMRMzrppU1WnNW0PWrGn4x9dPg
...
    =,,i };
```

**Thanks !**