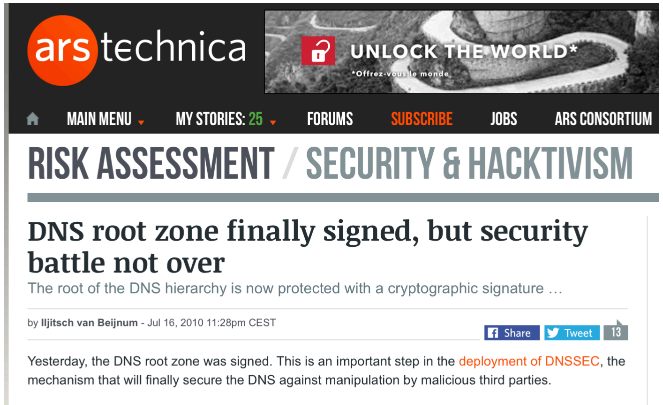
Tyre Kicking the DNS

Testing Transport Considerations of Rolling Roots

Geoff Huston
APNIC

Five Years Ago



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 …

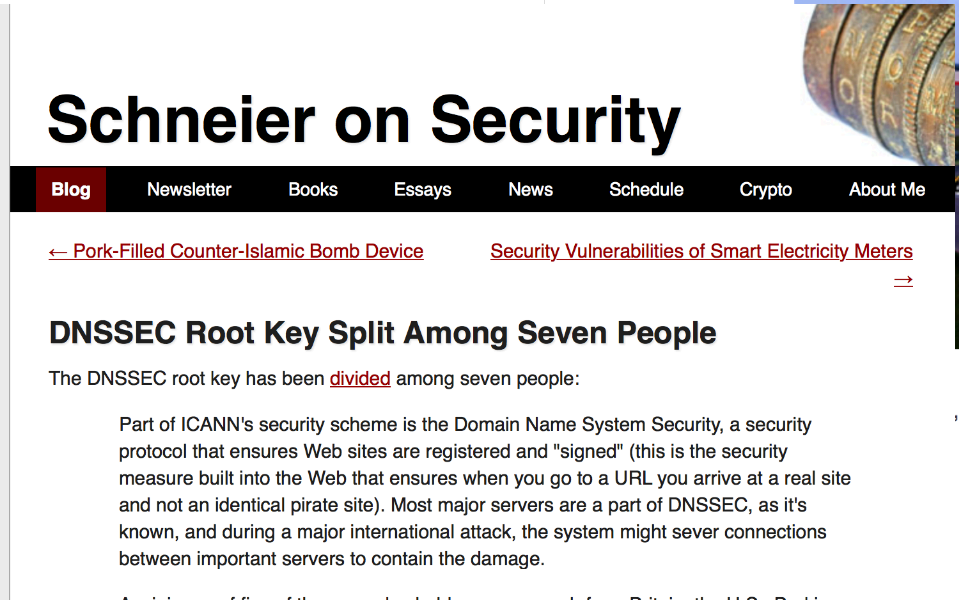
by Iljitsch van Beijnum - Jul 16, 2010 11:28pm CEST

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

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.

, VA - location of first DNSSEC key signing ceremony



Schneier on Security

← 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.

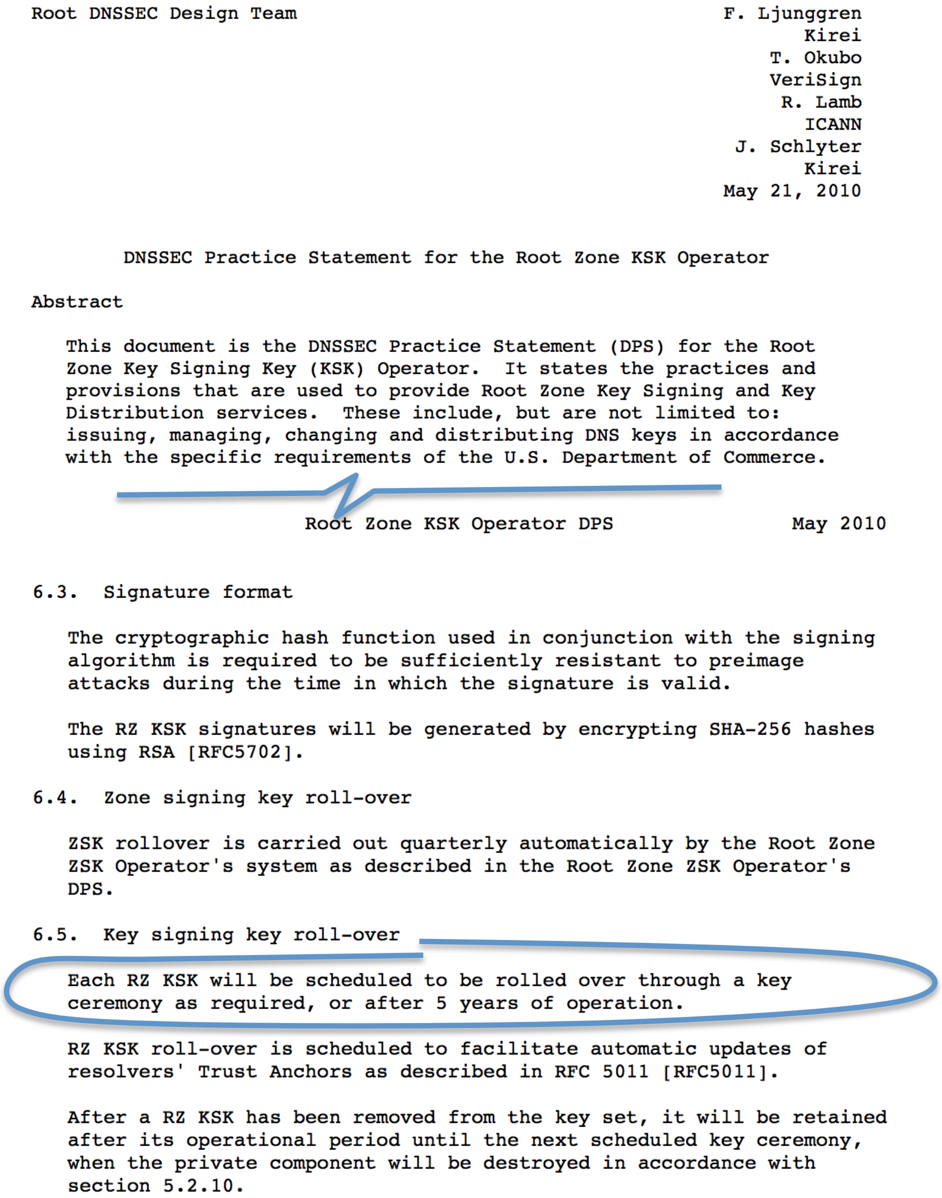
The US KSK Repository

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

The Amsterdam KSK Repository

George Michaelson

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.

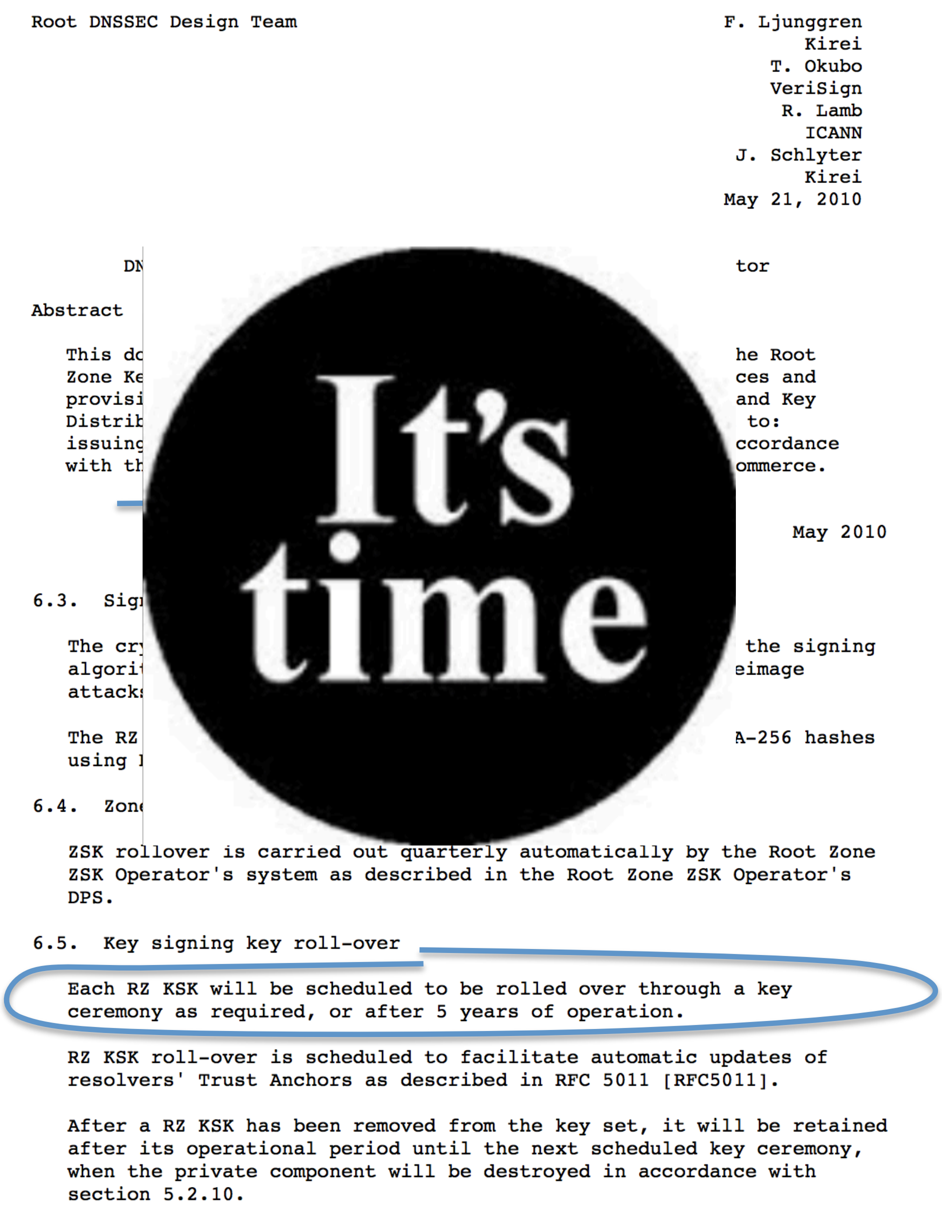
6.5. Key signing key roll-over

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.

Five Years Ago…



```
Root DNSSEC Design Team                                 F. Ljunggren
                                                               Kirei
                                                            T. Okubo
                                                            VeriSign
                                                             R. Lamb
                                                               ICANN
                                                         J. Schlyter
                                                               Kirei
                                                        May 21, 2010

        DN                                                  tor

Abstract

   This do                                                 he Root
   Zone Ke                                                 ces and
   provisi                                                 and Key
   Distrib                                                  to:
   issuing                                                 ccordance
   with th                                                 ommerce.

                                                              May 2010

6.3.  Sig

   The cry                                                  the signing
   algorit                                                 eimage
   attacks

   The RZ                                                  A-256 hashes
   using 

6.4.  Zon

   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.
```

Easy, Right?

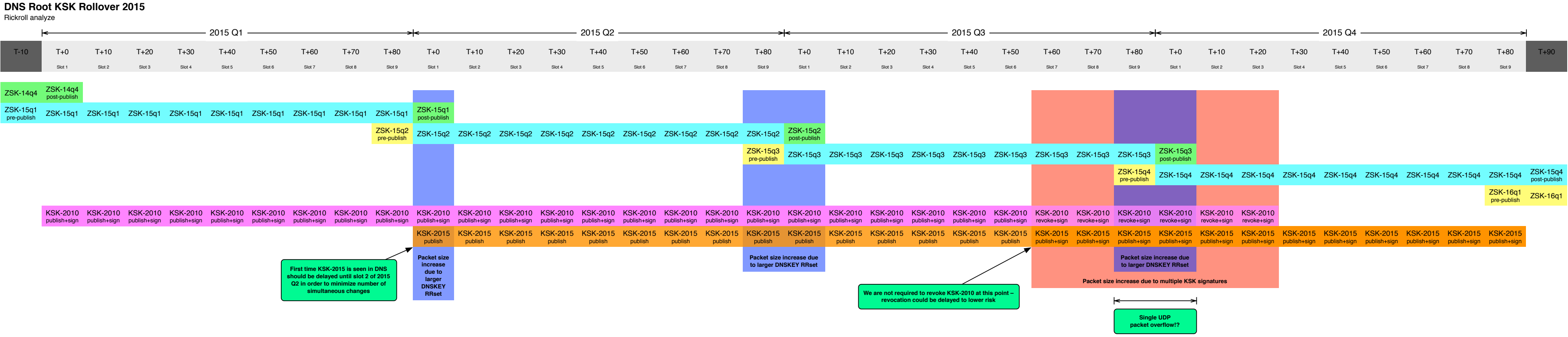
- Publish a new KSK and include it in DNSKEY responses
- Use the new KSK to sign the ZSK (as well as the old KSK signature)
- Withdraw the old signature signed via the old KSK
- Revoke the old KSK

Easy, Right?

- Publish a new KSK and include it in DNSKEY responses
- Use the ne[illegible] as well as the old [illegible]
- [illegible] old signature signed via the old [illegible]
- Revoke the old KSK

RFC 5011!

Easy, Right?



DNS Root KSK Rollover 2015

Rickroll analyze

2015 Q1

T-10	T+0 Slot 1	T+10 Slot 2	T+20 Slot 3	T+30 Slot 4	T+40 Slot 5	T+50 Slot 6	T+60 Slot 7	T+70 Slot 8	T+80 Slot 9
ZSK-14q4	ZSK-14q4 post-publish								
ZSK-15q1 pre-publish	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1	ZSK-15q1
									ZSK-15q2 pre-publish
	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign

2015 Q2

T+0 Slot 1	T+10 Slot 2	T+20 Slot 3	T+30 Slot 4	T+40 Slot 5	T+50 Slot 6	T+60 Slot 7	T+70 Slot 8	T+80 Slot 9
ZSK-15q1 post-publish								
ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2	ZSK-15q2
								ZSK-15q3 pre-publish
KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign
KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish

2015 Q3

T+0 Slot 1	T+10 Slot 2	T+20 Slot 3	T+30 Slot 4	T+40 Slot 5	T+50 Slot 6	T+60 Slot 7	T+70 Slot 8	T+80 Slot 9
ZSK-15q2 post-publish								
ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3	ZSK-15q3
								ZSK-15q4 pre-publish
KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 publish+sign	KSK-2010 revoke+sign	KSK-2010 revoke+sign	KSK-2010 revoke+sign
KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign

2015 Q4

T+0 Slot 1	T+10 Slot 2	T+20 Slot 3	T+30 Slot 4	T+40 Slot 5	T+50 Slot 6	T+60 Slot 7	T+70 Slot 8	T+80 Slot 9	T+90
ZSK-15q3 post-publish									
ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4	ZSK-15q4 post-publish
								ZSK-16q1 pre-publish	ZSK-16q1
KSK-2010 revoke+sign	KSK-2010 revoke+sign	KSK-2010 revoke+sign							
KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	KSK-2015 publish+sign	

First time KSK-2015 is seen in DNS should be delayed until slot 2 of 2015 Q2 in order to minimize number of simultaneous changes

Packet size increase due to larger DNSKEY RRset

Packet size increase due to larger DNSKEY RRset

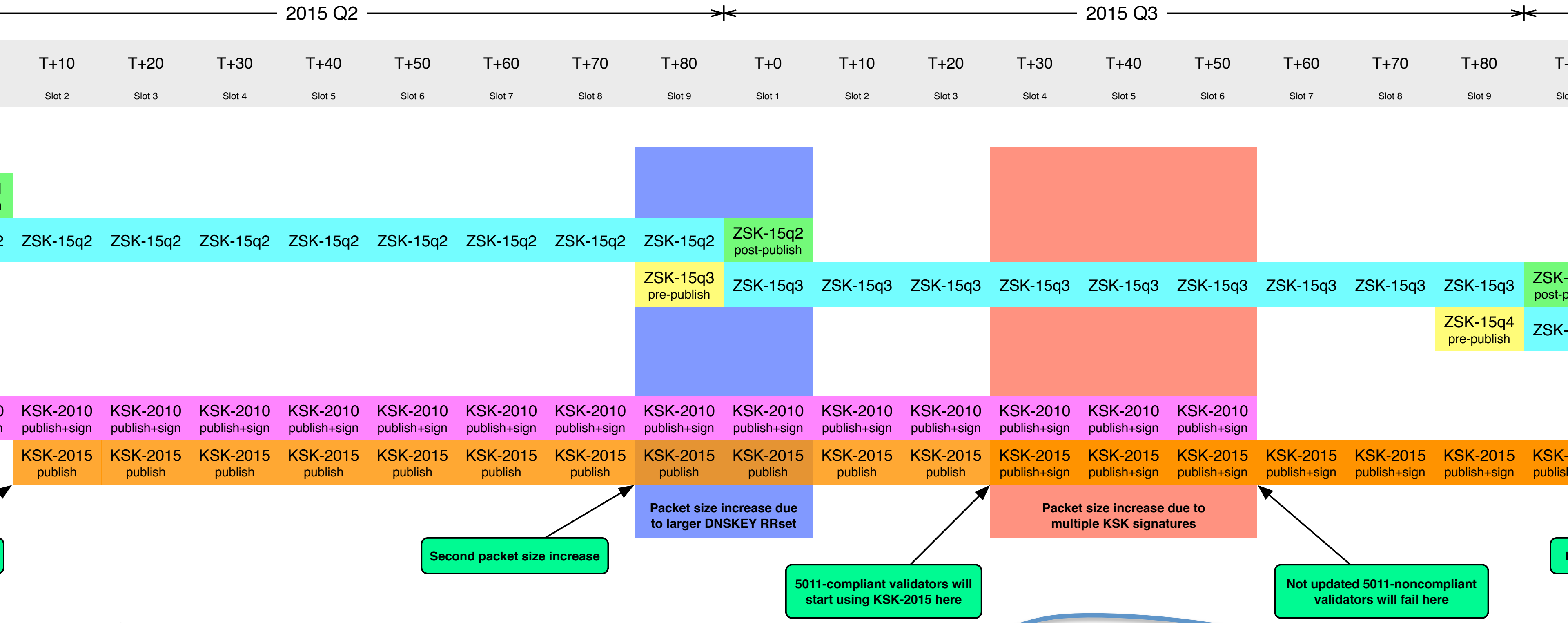
We are not required to revoke KSK-2010 at this point – revocation could be delayed to lower risk

Packet size increase due to larger DNSKEY RRset

Packet size increase due to multiple KSK signatures

Single UDP packet overflow!?

Easy, Right?



Packet size increase due to larger DNSKEY RRset

Packet size increase due to multiple KSK signatures

Second packet size increase

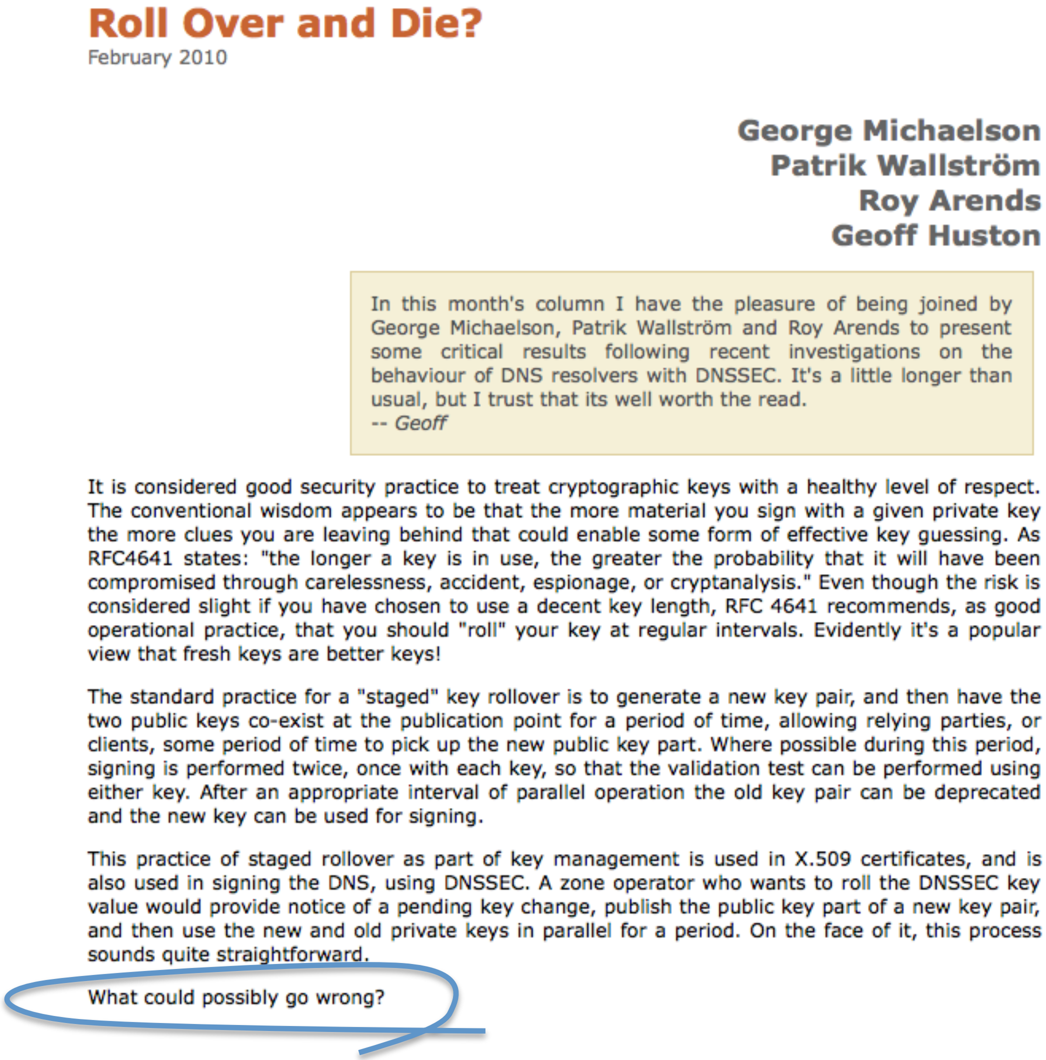
5011-compliant validators will start using KSK-2015 here

Not updated 5011-noncompliant validators will fail here

1024 bit ZSK	1011	1158	1011	1297	736	883
2048 bit ZSK	1139	1414	1139	1425	864	1139

. / IN / DNSKEY response size

We've (sort of) done it before



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

Truly ruly!

But that was then…

And this is now:

- Resolvers are now not so aggressive in searching for alternate validation path hen validation fails
  - (as long a up to date, which everyone does – rig
- And now w 011 key roll processes
- And ***everyo*** an cope with large DNS responses

Not!

So all this will go without a hitch

Nobody will even notice the KSK roll at the root

Truly ruly!

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

What can be tested …

~~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~~

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

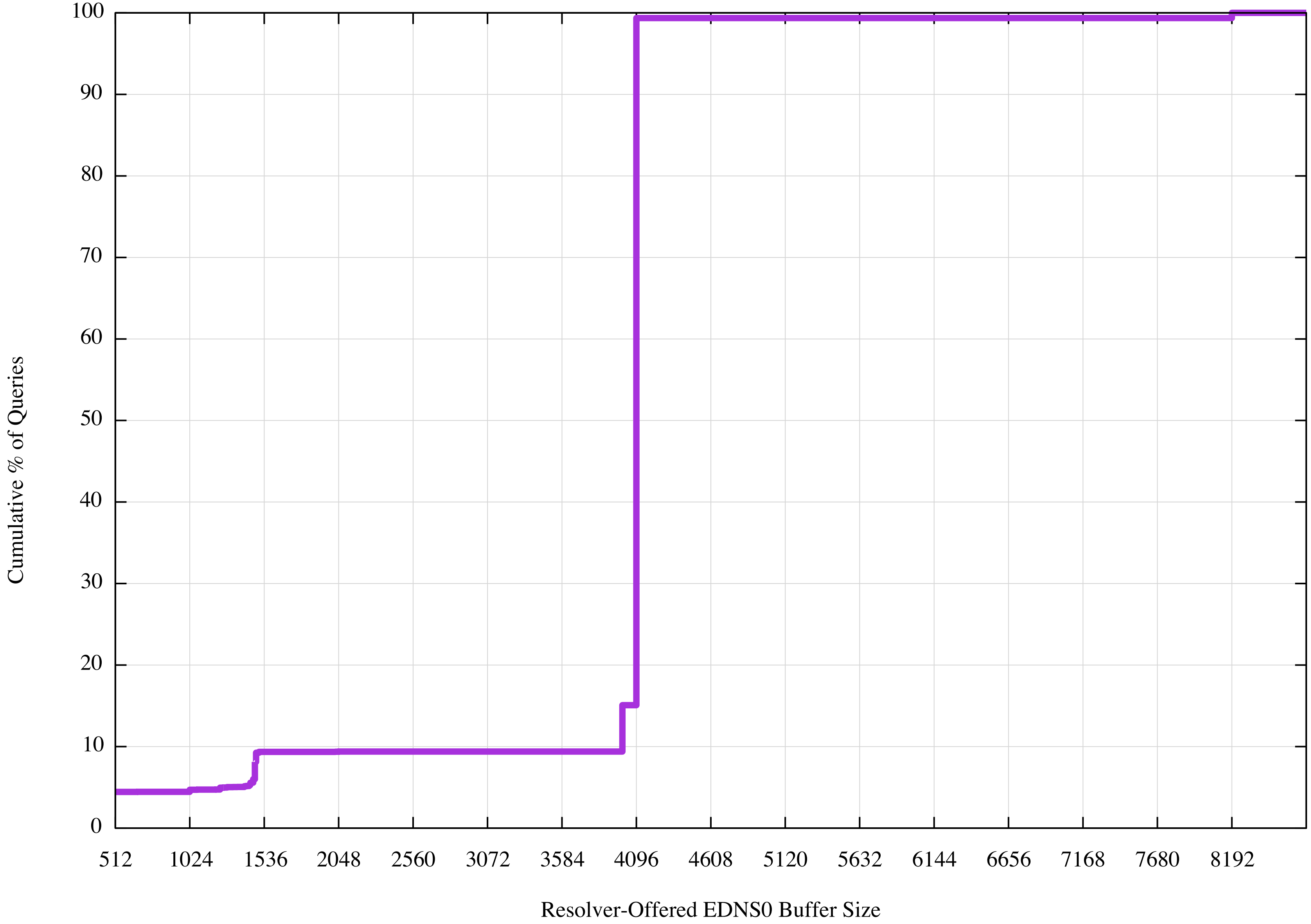
So we've been testing

- We are interested in sending DNSSEC-aware DNS resolvers a response that is much the same size as that being contemplated in a KSK key roll
- And seeing whether they got the response

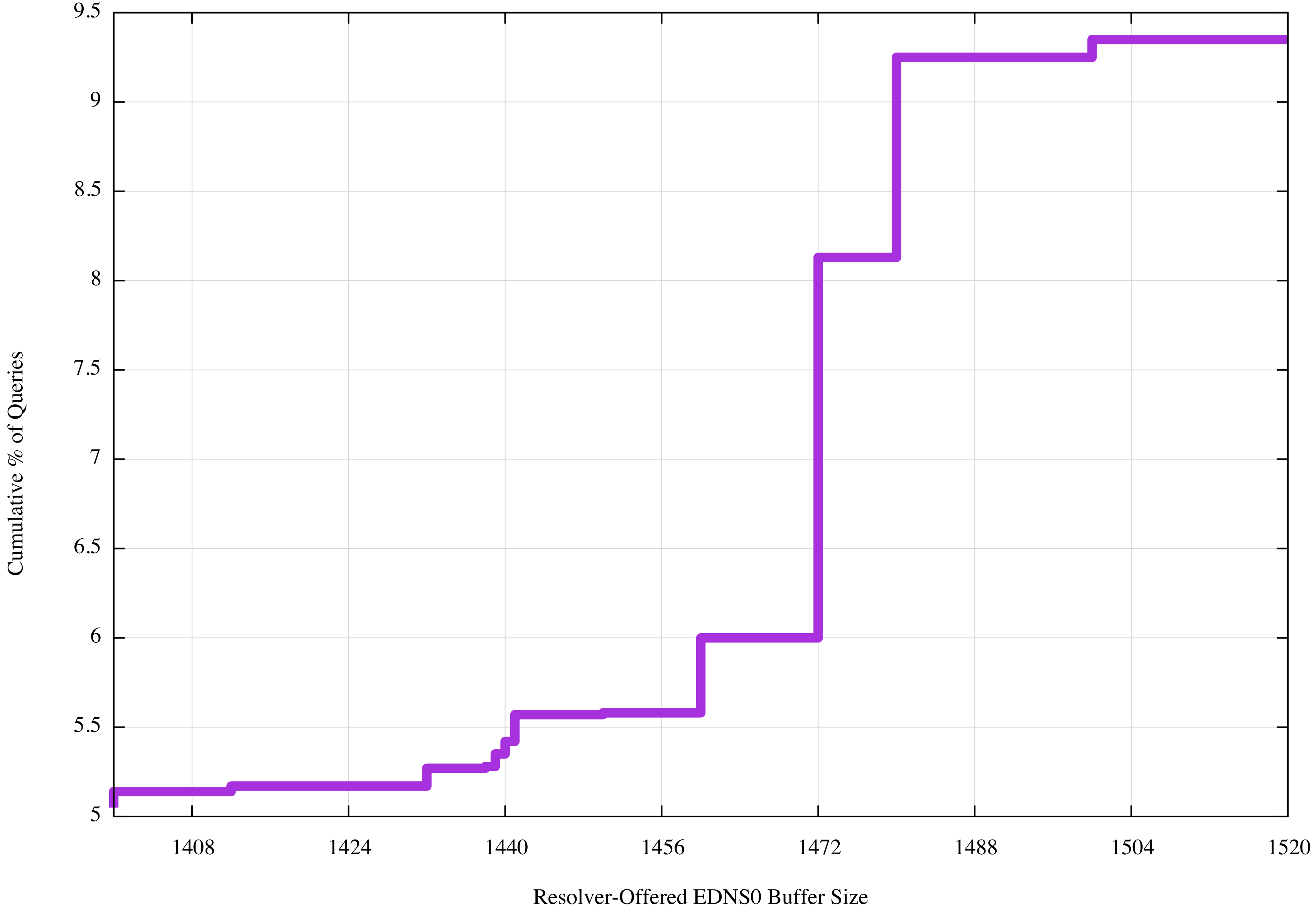
Some Interesting Sizes

8 octets	UDP pseudo header size
20 octets	IPv4 packet header
40 octets	maximum size of IPv4 options in an IPv4 IP packet header
40 octets	IPv6 packet header
512 octets	the maximum DNS payload size that must be supported by DNS
560 octets	the maximum IPv4 packet size that must be supported by IPv4 DNS UDP systems
576 octets	the largest IP packet size (including headers) that must be supported by IPv4 systems
913 octets	the size of the current root priming response with DNSSEC signature
1,232 octets	the largest DNS payload size of an unfragmentable IPv6 DNS UDP packet
1,280 octets	the smallest unfragmented IPv6 packet that must be supported by all IPv6 systems
1,425 octets	**the largest size of a ./IN/DNSKEY response with a 2048 bit ZSK**
1,452 octets	the largest DNS payload size of an unfragmented Ethernet IPv6 DNS UDP packet
1,472 octets	the largest DNS payload size of an unfragmented Ethernet IPv4 DNS UDP packet
1,500 octets	the largest IP packet supported on IEEE 802.3 Ethernet networks

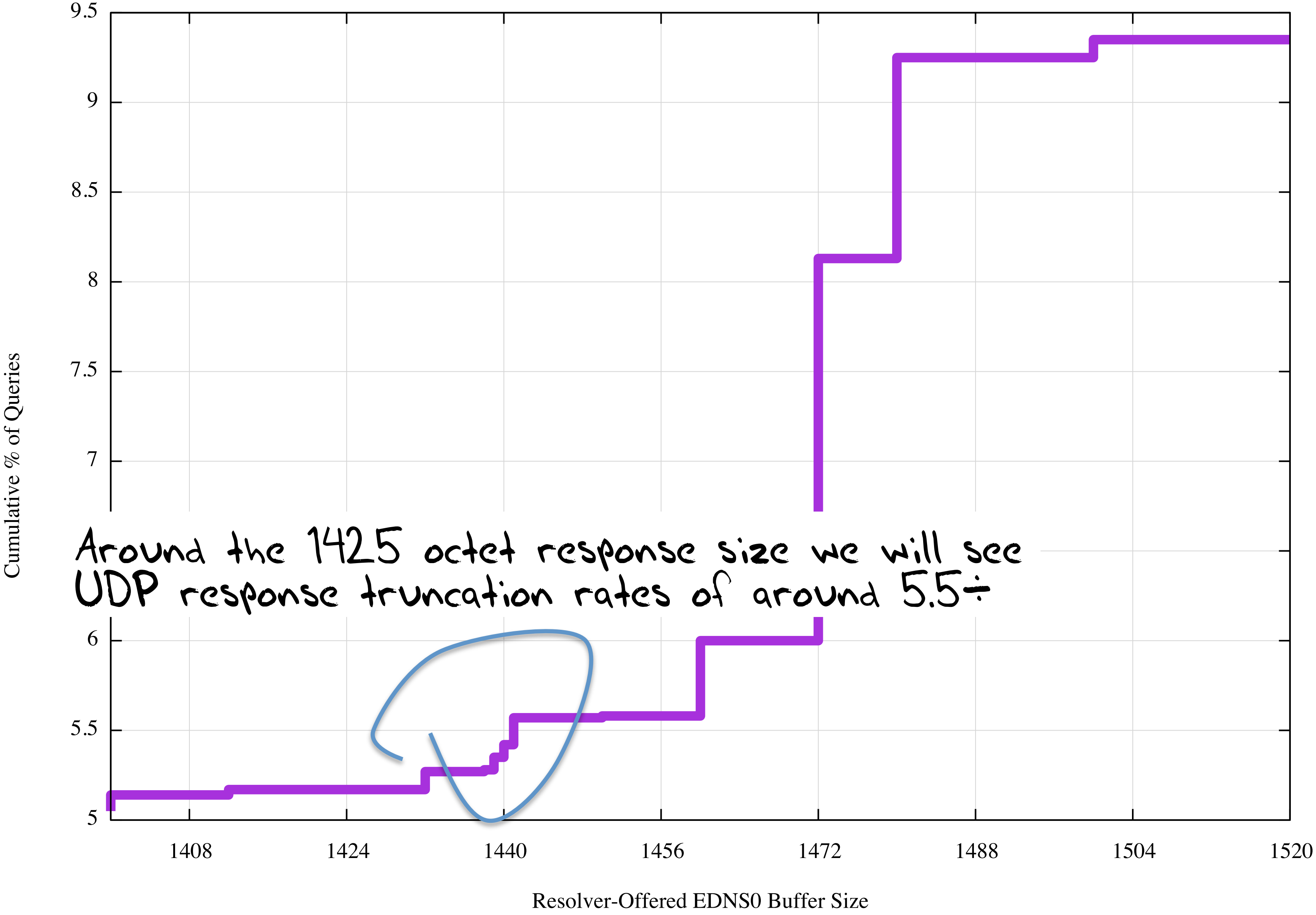
EDNS(0) UDP Buffer sizes



EDNS(0) UDP Buffer sizes



EDNS(0) UDP Buffer sizes



Around the 1425 octet response size we will see UDP response truncation rates of around 5.5%

The Test Method

We are using a mechanism to measure the Internet from the “edge”:

- We use an ad with an active script element
- When a browser receives an impression of the ad the script is activated
- The script fetches a small number (5) of 1x1 blots, and then fetches a final blot to tell us which ones it actually received
- As long as every DNS name in the URLs of these blots is unique, then DNS and Web proxies can’t interfere!
- Our servers see the DNS queries and the Web fetches
- We can infer client-side behaviours based on these observations

* Acknowledgement and thanks to Google for supporting this work

The Test

- We are interested in resolvers who are DNSSEC aware (queries that contain the EDNS0 option with DNSSEC OK flag set on)
- We would like to test larger responses:
  - 1,440 octets of DNS payload
- We would like to test a couple of crypto protocols
  - RSA
  - ECDSA

Testing

- We are interested in those resolvers that are retrieving DNSSEC signature data, so we are looking for queries that include EDNS0 and DNSSEC OK flag set
- How many resolver queries have DNSSEC OK set?

EDNS(0) DNSSEC OK Set

76,456,053 **queries**

63,352,607 queries with EDNS(0) and DNSSEC OK set

= 83% of queries

777,371 **resolvers**

649,304 resolvers with EDNS(0) and DNSSEC OK set

= 84% of resolvers

Large Responses

How well are 1,440 octet DNS responses handled when compared to much smaller responses?

1,440 octet RSA-signed Responses

9,113,215 tests

7,769,221 retrieved the 1x1 blot (85%)

2,644,351 queried for the DS record

849,340 queried for the DS record (but no blot fetch)

494,581 timed out (but no blot fetch)

72 appeared to fail the DNS

1,440 octet RSA-signed Responses

9,113,215 tests

7,769,221 retrieved the 1x1 blot

2,644,351 queried f

8 the DS record (but no blot)

494,581 timed out (but no blot)

72 appeared to fail the DNS

Some 5% of experiments did not run through to completion.
For an online ad, this is not an unexpected outcome

1,440 octet RSA-signed Responses

9,113,215 tests

7,769,221 retrieved the 1x1 b

2,644,351 queri

84 cord (but no blot)

4 t (but no blot)

72 appeared to fail the DNS

This is measuring USERS not RESOLVERS. Users often use local configurations of 2 or more resolvers, and problems in one resolver appear to be fixed by the other resolver(s).

Small vs Large

What happens when the response size grows above 1,472 octets?

1,440 Octets Payload

Experiments:	6,542,993
Web Fetch:	5,880,921
DS Fetch:	181,610
Timeout:	480,415
DNS Fail:	47

1,770 Octets Payload

Experiments:	6,566,645
Web Fetch:	5,992,617
DS Fetch:	167,119
Timeout:	401,831
DNS Fail:	5,078

ECDSA vs RSA

The spec says that when a resolver encounters a zone signed only with algorithms that are not supported by the resolver then it will treat the zone as unsigned and not proceed with validation

Most resolvers determine the zone's signing algorithms from the DS record

What happens when we compare a 1,440 octet response signed by RSA and a 1,440 octet response signed by ECDSA?

1,440 octet ECDSA-signed Responses

9,137,436 tests

7,766,572 retrieved the 1x1 blot

2,644,564 queried for the DS record

  860,163 queried for the DS record (but no blot)

  505,045 timed out (but no blot!)

    5,656 appeared to fail the DNS

1,440 octet ECDSA-signed Responses

9,137,436 tests

7,766,572 retrieved the 1x1

2,644,564 queri

86 ord (but no blot)

5 (but no blot!)

appeared to fail the DNS

This is larger than RSA failure rates, but is still a small proportion of users affected. Some resolvers appear to have problems when presented with an unknown crypto protocol.

IPv4 vs IPv6

Do resolvers prefer IPv4 over IPv6?

Total Queries:	47,826,735
Queries over V6:	394,816
Number of Resolvers:	109,725
Number of Resolvers using IPv6 for queries:	2,849

IPv4 vs IPv6

Do resolvers prefer IPv4 over IPv6?

Total Queries: 47,8

Queries ov

in a Dual Stack environment 1% of queries and 3% of resolvers use IPv6.

solvers: 109,725

What if the server was IPv6 only?

Number of Resolvers

using IPv6 for queries: 2,849

Some Observations

There is a LOT of DNSSEC validation out there

- 87% of all queries have DNSSEC-OK set
- 30% of all DNSSEC-OK resolvers 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

There is very little V6 being used out there

- 1% of queries use IPv6 as the transport protocol when given a dual stack name server

It seems that when given a choice:

Browsers prefer IPv6

Resolvers prefer IPv4

Some Observations

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

Can it work?

If we stick to RSA and keep response sizes at or below 1,440 octets then there appears to be no obvious user impact in terms of packet size

- Some resolvers may get stuck, but users appear to use multiple resolvers

Questions?

Geoff Huston
George Michaelson