Rolling with Confidence:
Managing the Complexity of DNSSEC Operations

Moritz Müller[1,2], Taejoong Chung[3], Roland van Rijswijk-Deij[2], Alan Mislove[3]

[1]SIDN, [2]University of Twente, [3]Northeastern University

RIPE 76 | 2018-05-16

Northeastern UNIVERSITY OF TWENTE. SIDN LABS

About SIDN

- Registry of the Dutch ccTLD *.nl*
- More than 5,8 million registered domains
- More than 3 million signed with DNSSEC
- SIDN Labs is its research department
  - Goal: increase the security and stability of *.nl* and the Internet overall
  - 7 team members + interns

“key rollovers are
a fact of life
when using DNSSEC”

from RFC 6781

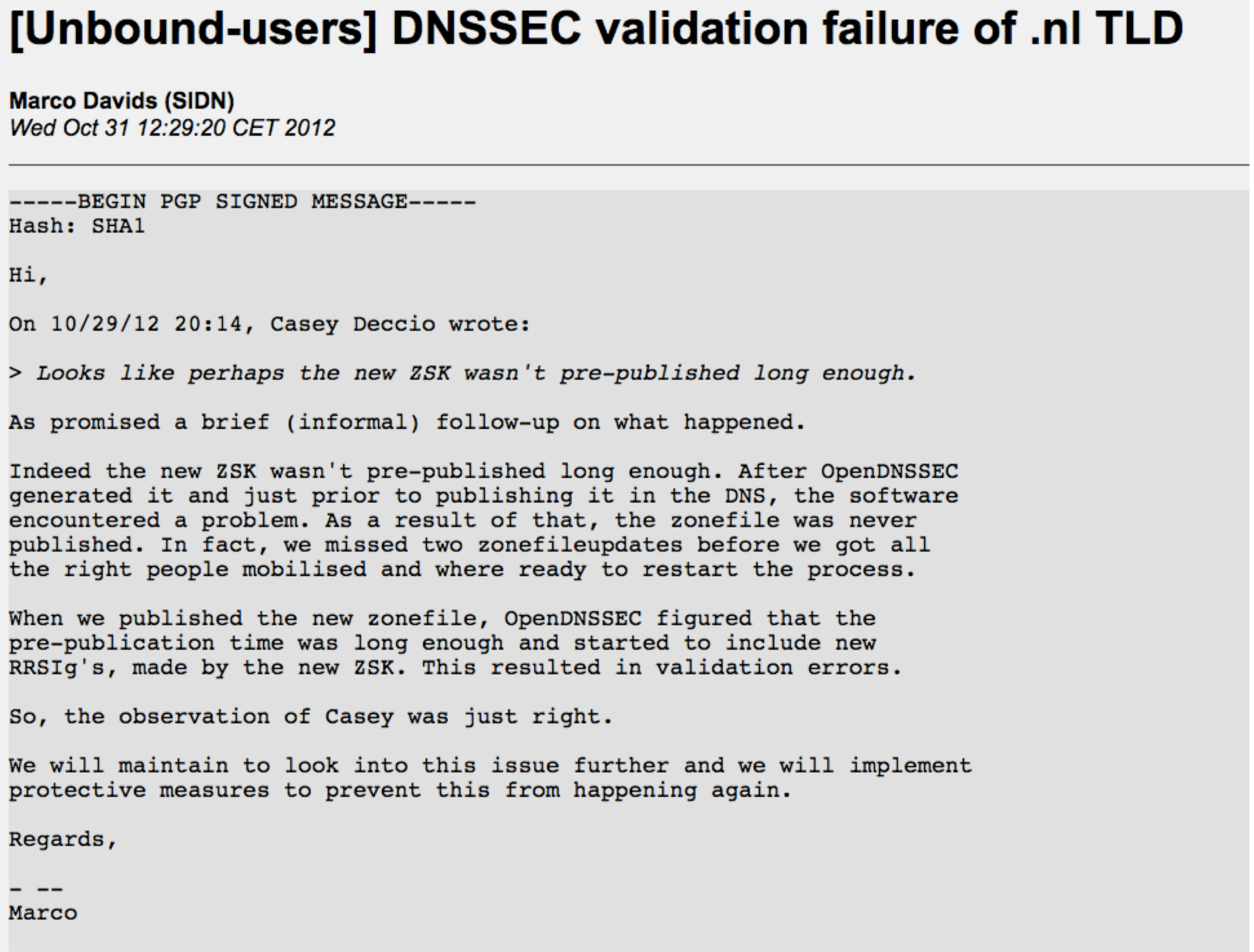
- ZSK Rollovers
- KSK Rollovers
- Algorithm Rollovers

Algorithm Rollover Stages

```
--------------------------------------------------------------------------------------------------------------------------------
 initial              new RRSIGs           new DNSKEY             new DS               DNSKEY removal       RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 --------------------------------------------------------> SOA_1 -------------------------------------------------------->
 RRSIG_par(SOA) -----------------------------------------------> RRSIG_par(SOA) ----------------------------------------------->
 DS_K_1 -------------------------------------------------------> DS_K_2 ------------------------------------------------------->
 RRSIG_par(DS_K_1) --------------------------------------------> RRSIG_par(DS_K_2) -------------------------------------------->

Child:
 SOA_0                SOA_1                SOA_2                 -------------------> SOA_3                SOA_4
 RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)       -------------------> RRSIG_Z_10(SOA)
                      RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)       -------------------> RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)

 DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1            ------------------->
                                           DNSKEY_K_2            -------------------> DNSKEY_K_2           DNSKEY_K_2
 DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10           ------------------->
                                           DNSKEY_Z_11           -------------------> DNSKEY_Z_11          DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)     ------------------->
                                           RRSIG_K_2(DNSKEY)     -------------------> RRSIG_K_2(DNSKEY)    RRSIG_K_2(DNSKEY)
 --------------------------------------------------------------------------------------------------------------------------------
```

from RFC 6781, Figure 8

Rollovers can be risky



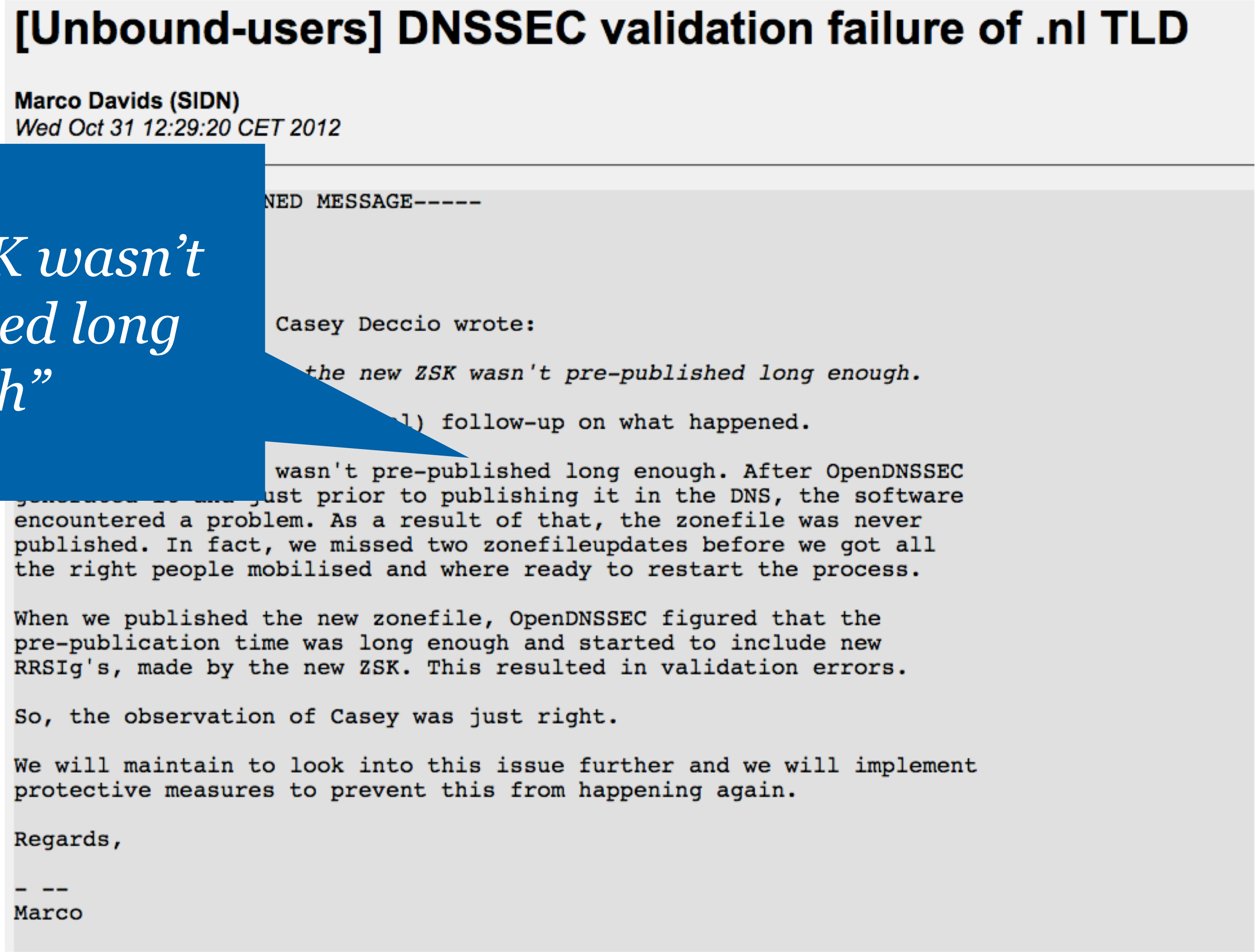
[Unbound-users] DNSSEC validation failure of .nl TLD

Marco Davids (SIDN)
Wed Oct 31 12:29:20 CET 2012

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hi,

On 10/29/12 20:14, Casey Deccio wrote:

> Looks like perhaps the new ZSK wasn't pre-published long enough.

As promised a brief (informal) follow-up on what happened.

Indeed the new ZSK wasn't pre-published long enough. After OpenDNSSEC
generated it and just prior to publishing it in the DNS, the software
encountered a problem. As a result of that, the zonefile was never
published. In fact, we missed two zonefileupdates before we got all
the right people mobilised and where ready to restart the process.

When we published the new zonefile, OpenDNSSEC figured that the
pre-publication time was long enough and started to include new
RRSIg's, made by the new ZSK. This resulted in validation errors.

So, the observation of Casey was just right.

We will maintain to look into this issue further and we will implement
protective measures to prevent this from happening again.

Regards,

- --
Marco
```

Rollovers can be risky

”the new ZSK wasn’t pre-published long enough”

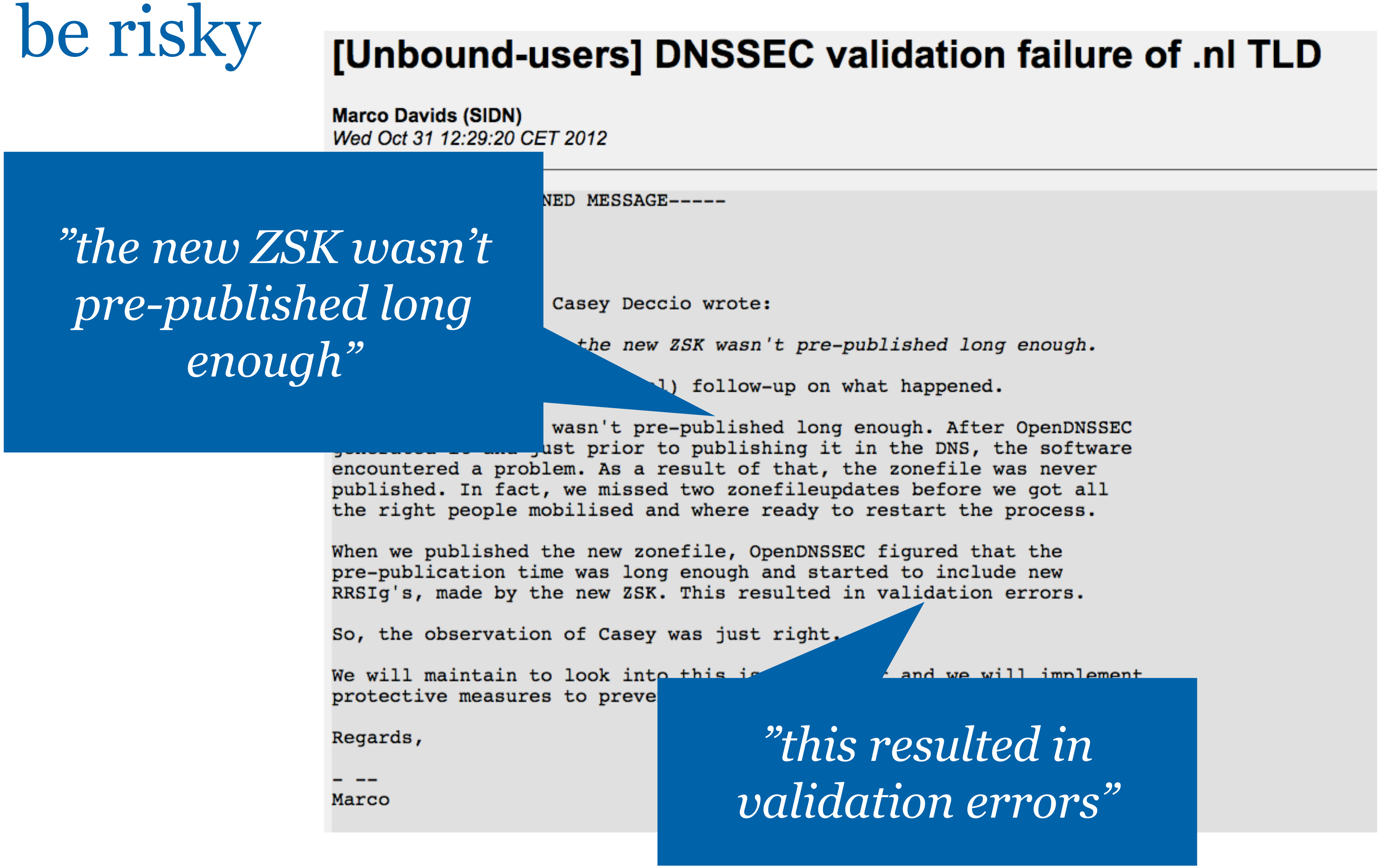


[Unbound-users] DNSSEC validation failure of .nl TLD

Marco Davids (SIDN)
Wed Oct 31 12:29:20 CET 2012

```
NED MESSAGE-----

 Casey Deccio wrote:

 the new ZSK wasn't pre-published long enough.

l) follow-up on what happened.

 wasn't pre-published long enough. After OpenDNSSEC
ust prior to publishing it in the DNS, the software
encountered a problem. As a result of that, the zonefile was never
published. In fact, we missed two zonefileupdates before we got all
the right people mobilised and where ready to restart the process.

When we published the new zonefile, OpenDNSSEC figured that the
pre-publication time was long enough and started to include new
RRSIg's, made by the new ZSK. This resulted in validation errors.

So, the observation of Casey was just right.

We will maintain to look into this issue further and we will implement
protective measures to prevent this from happening again.

Regards,

- --
Marco
```

Rollovers can be risky



[Unbound-users] DNSSEC validation failure of .nl TLD

Marco Davids (SIDN)
Wed Oct 31 12:29:20 CET 2012

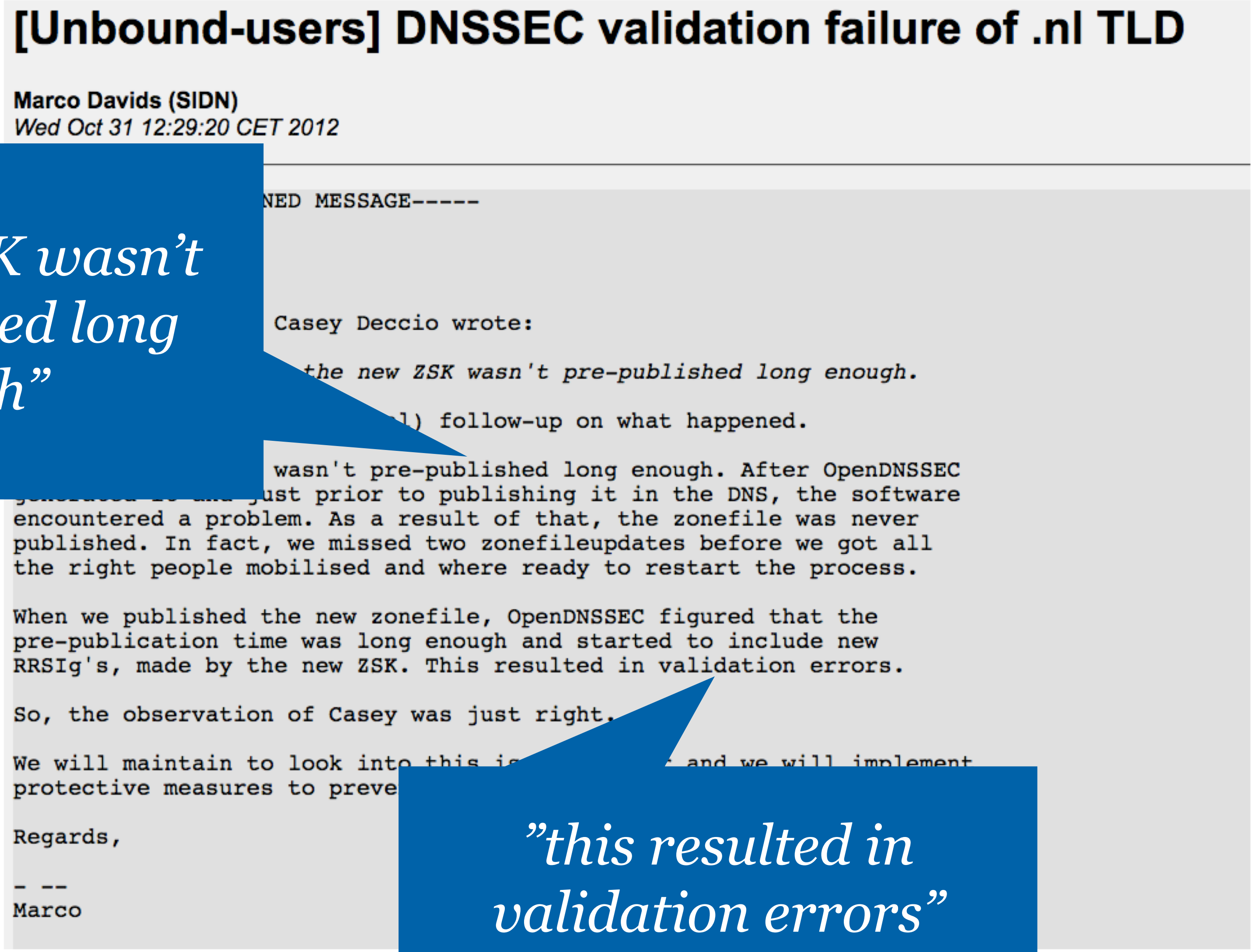
```
NED MESSAGE-----

 Casey Deccio wrote:

the new ZSK wasn't pre-published long enough.

l) follow-up on what happened.

 wasn't pre-published long enough. After OpenDNSSEC
ust prior to publishing it in the DNS, the software
encountered a problem. As a result of that, the zonefile was never
published. In fact, we missed two zonefileupdates before we got all
the right people mobilised and where ready to restart the process.

When we published the new zonefile, OpenDNSSEC figured that the
pre-publication time was long enough and started to include new
RRSIg's, made by the new ZSK. This resulted in validation errors.

So, the observation of Casey was just right.

We will maintain to look into this i                and we will implement
protective measures to preve

Regards,

- --
Marco
```

”the new ZSK wasn’t pre-published long enough”

”this resulted in validation errors”

Northeastern UNIVERSITY OF TWENTE. SIDN LABS

Rollovers can be risky



[Unbound-users] DNSSEC validation failure of .nl TLD

Marco Davids (SIDN)
Wed Oct 31 12:29:20 CET 2012

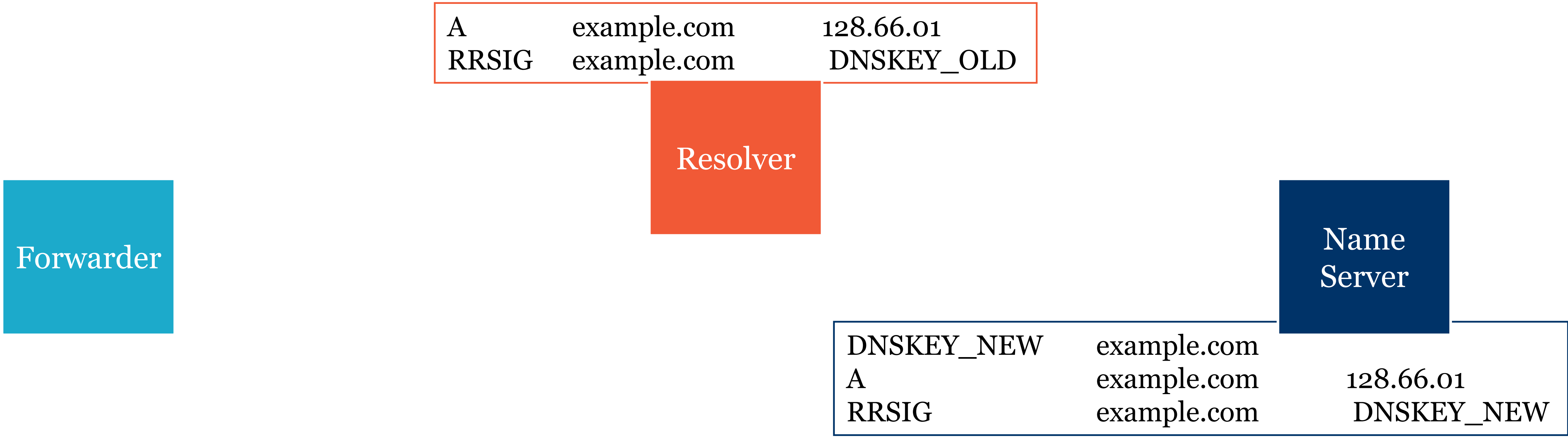
```
NED MESSAGE-----

 Casey Deccio wrote:

 the new ZSK wasn't pre-published long enough.

l) follow-up on what happened.

 wasn't pre-published long enough. After OpenDNSSEC
just prior to publishing it in the DNS, the software
encountered a problem. As a result of that, the zonefile was never
published. In fact, we missed two zonefileupdates before we got all
the right people mobilised and where ready to restart the process.

When we published the new zonefile, OpenDNSSEC figured that the
pre-publication time was long enough and started to include new
RRSIg's, made by the new ZSK. This resulted in validation errors.

So, the observation of Casey was just right.

We will maintain to look into this i          and we will implement
protective measures to preve

Regards,

- --
Marco
```

”the new ZSK wasn’t pre-published long enough”

”this resulted in validation errors”

It’s all about the right timing

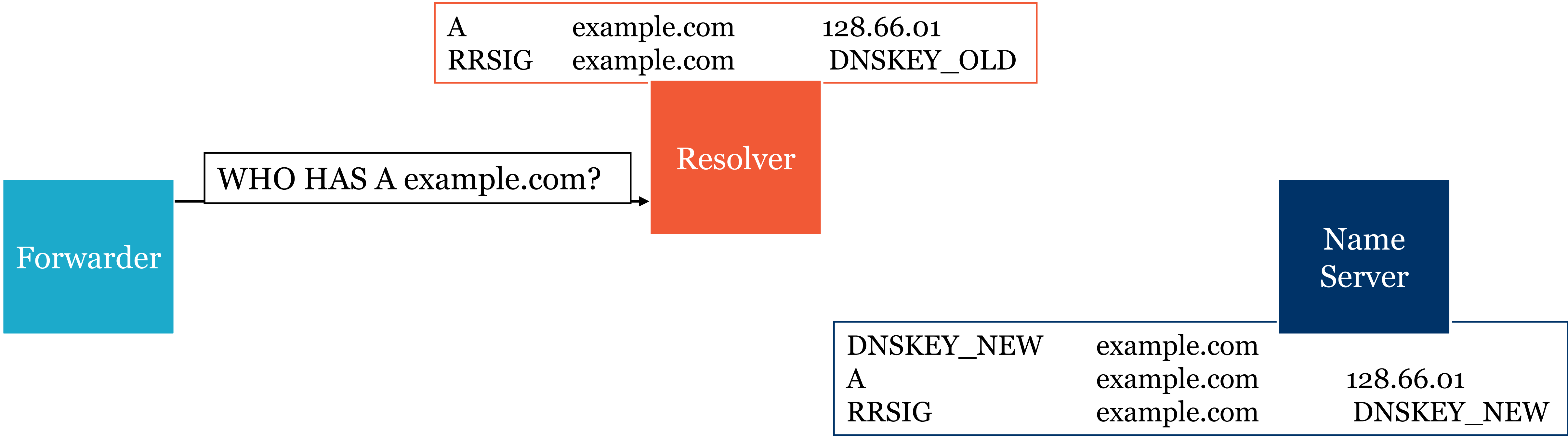
Timing of Rollovers



A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers



A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

Resolver

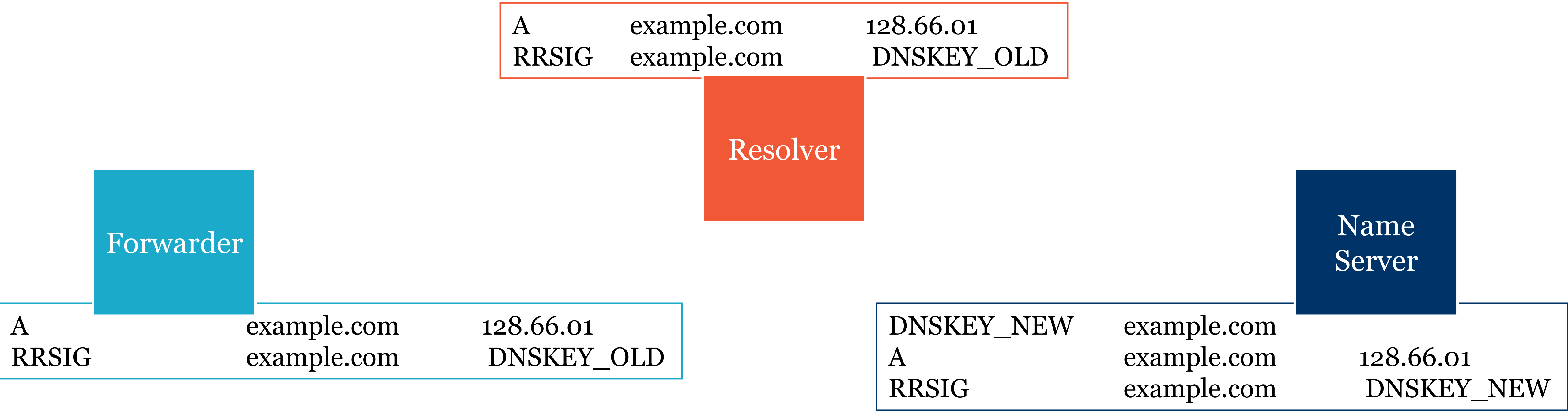
WHO HAS A example.com?

Forwarder

Name Server

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers



Resolver

A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

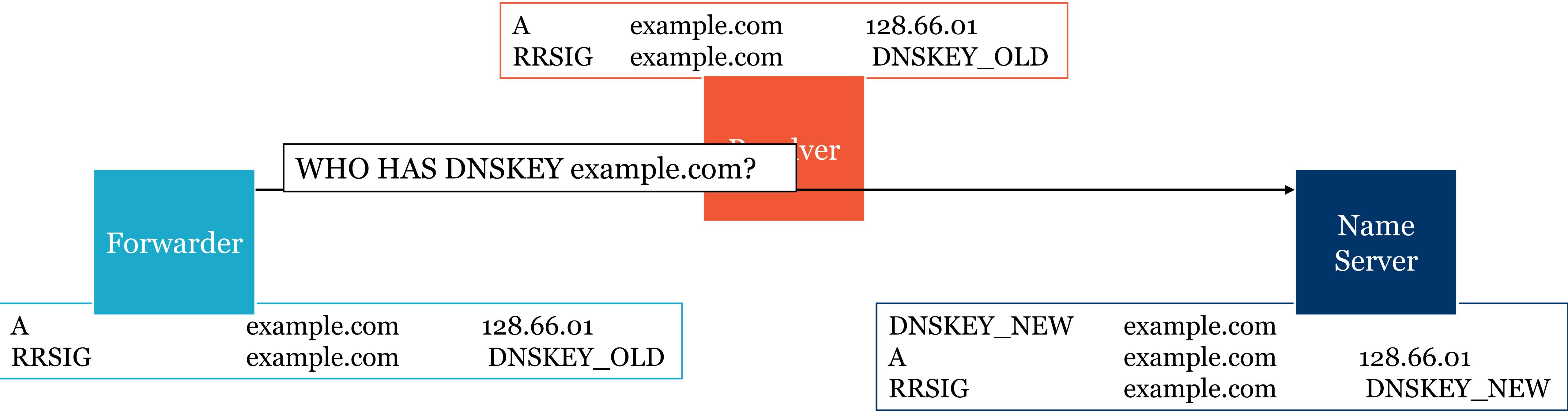
Forwarder

A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

Name Server

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers

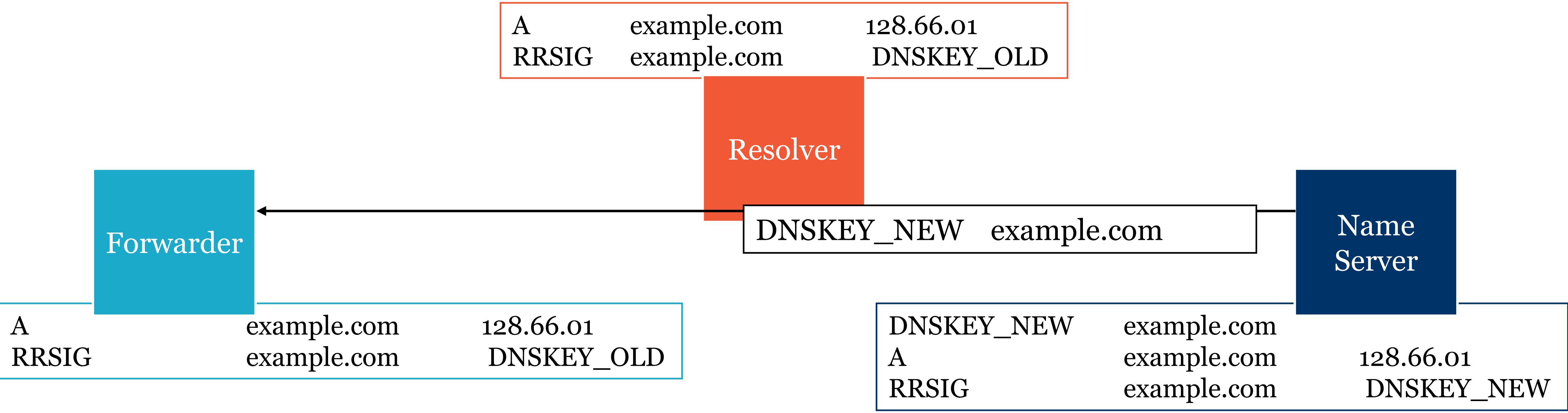


A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers



A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

Resolver

Forwarder

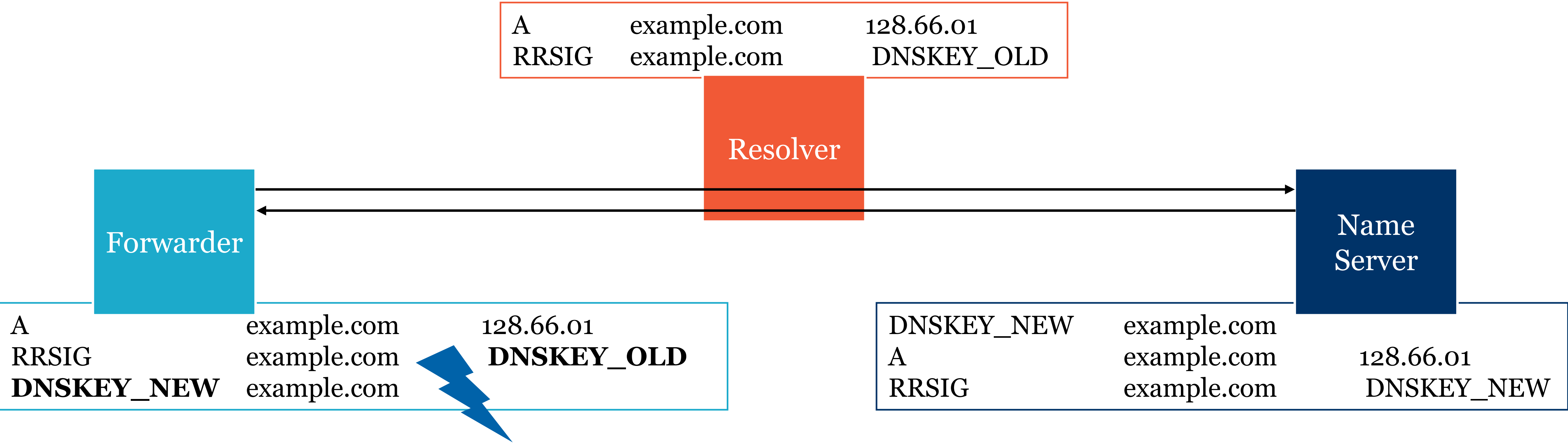
DNSKEY_NEW example.com

Name Server

A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers



A	example.com	128.66.01
RRSIG	example.com	DNSKEY_OLD

A	example.com	128.66.01
RRSIG	example.com	**DNSKEY_OLD**
DNSKEY_NEW	example.com	

DNSKEY_NEW	example.com	
A	example.com	128.66.01
RRSIG	example.com	DNSKEY_NEW

Timing of Rollovers

	Publication Delay	**Propagation Delay**
Description	Time it takes until every name server is in sync	Time it takes until resolvers have picked up the new state
Period	Seconds to minutes	Minutes, hours, or even days

Algorithm Rollover Stages

```
--------------------------------------------------------------------------------------------------------------------------------
 initial              new RRSIGs           new DNSKEY             new DS               DNSKEY removal       RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 --------------------------------------------------------> SOA_1 -------------------------------------------------------->
 RRSIG_par(SOA) -----------------------------------------------> RRSIG_par(SOA) ----------------------------------------------->
 DS_K_1 -------------------------------------------------------> DS_K_2 ------------------------------------------------------->
 RRSIG_par(DS_K_1) --------------------------------------------> RRSIG_par(DS_K_2) ------------------------------------------->

Child:
 SOA_0                SOA_1                SOA_2                 -------------------> SOA_3                SOA_4
 RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)       -------------------> RRSIG_Z_10(SOA)
                      RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)       -------------------> RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)

 DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1            ------------------->
                                           DNSKEY_K_2            -------------------> DNSKEY_K_2           DNSKEY_K_2
 DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10           ------------------->
                                           DNSKEY_Z_11           -------------------> DNSKEY_Z_11          DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)     ------------------->
                                           RRSIG_K_2(DNSKEY)     -------------------> RRSIG_K_2(DNSKEY)    RRSIG_K_2(DNSKEY)
 --------------------------------------------------------------------------------------------------------------------------------
```

from RFC 6781, Figure 8

Algorithm Rollover Stages

5 Stages

```
--------------------------------------------------------------------------------------------------------------------------------
 initial              new RRSIGs           new DNSKEY             new DS               DNSKEY removal       RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 --------------------------------------------------------> SOA_1 ------------------------------------------------------->
 RRSIG_par(SOA) -----------------------------------------------> RRSIG_par(SOA) ---------------------------------------------->
 DS_K_1 -------------------------------------------------------> DS_K_2 ------------------------------------------------------>
 RRSIG_par(DS_K_1) --------------------------------------------> RRSIG_par(DS_K_2) ------------------------------------------->

Child:
 SOA_0                SOA_1                SOA_2                 -------------------> SOA_3                SOA_4
 RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)       -------------------> RRSIG_Z_10(SOA)
                      RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)       -------------------> RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)

 DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1            ------------------->
                                           DNSKEY_K_2            -------------------> DNSKEY_K_2           DNSKEY_K_2
 DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10           ------------------->
                                           DNSKEY_Z_11           -------------------> DNSKEY_Z_11          DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)     ------------------->
                                           RRSIG_K_2(DNSKEY)     -------------------> RRSIG_K_2(DNSKEY)    RRSIG_K_2(DNSKEY)
 --------------------------------------------------------------------------------------------------------------------------------
```

from RFC 6781, Figure 8

Algorithm Rollover Stages

5 Stages

```
--------------------------------------------------------------------------------------------------------------------------------
 initial              new RRSIGs           new DNSKEY             new DS               DNSKEY removal       RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 --------------------------------------------------------> SOA_1 -------------------------------------------------------->
 RRSIG_par(SOA) -----------------------------------------------> RRSIG_par(SOA) ----------------------------------------------->
 DS_K_1 -------------------------------------------------------> DS_K_2 ------------------------------------------------------->
 RRSIG_par(DS_K_1) --------------------------------------------> RRSIG_par(DS_K_2) -------------------------------------------->

Child:
 SOA_0                SOA_1                SOA_2                 -------------------> SOA_3                SOA_4
 RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)       -------------------> RRSIG_Z_10(SOA)
                      RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)       -------------------> RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)

 DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1            ------------------->
                                           DNSKEY_K_2            -------------------> DNSKEY_K_2           DNSKEY_K_2
 DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10           ------------------->
                                           DNSKEY_Z_11           -------------------> DNSKEY_Z_11          DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)     ------------------->
                                           RRSIG_K_2(DNSKEY)     -------------------> RRSIG_K_2(DNSKEY)    RRSIG_K_2(DNSKEY)
 -------------------------------------------------------------------------------------------------------------------------------
```

Wait for delays

from RFC 6781, Figure 8

Algorithm Rollover Stages

5 Stages

Interaction with parent

```
--------------------------------------------------------------------------------------------------------------------------------
 initial              new RRSIGs           new DNSKEY             new DS               DNSKEY removal       RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 --------------------------------------------------------> SOA_1 -------------------------------------------------------->
 RRSIG_par(SOA) -----------------------------------------------> RRSIG_par(SOA) ----------------------------------------------->
 DS_K_1 -------------------------------------------------------> DS_K_2 ------------------------------------------------------->
 RRSIG_par(DS_K_1) --------------------------------------------> RRSIG_par(DS_K_2) ------------------------------------------->

Child:
 SOA_0                SOA_1                SOA_2                 -------------------> SOA_3                SOA_4
 RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)       -------------------> RRSIG_Z_10(SOA)
                      RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)       -------------------> RRSIG_Z_11(SOA)      RRSIG_Z_11(SOA)

 DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1            ------------------->
                                           DNSKEY_K_2            -------------------> DNSKEY_K_2           DNSKEY_K_2
 DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10           ------------------->
                                           DNSKEY_Z_11           -------------------> DNSKEY_Z_11          DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)     ------------------->
                                           RRSIG_K_2(DNSKEY)     -------------------> RRSIG_K_2(DNSKEY)    RRSIG_K_2(DNSKEY)
 --------------------------------------------------------------------------------------------------------------------------------
```

Wait for delays

from RFC 6781, Figure 8

The Conservative Algorithm Rollover

- Some old Unbound resolvers expect one signature for each algorithm in the zone apex
- If not, they suspect a downgrade attack
- and fail validation :-(

The Conservative Algorithm Rollover

- Some old Unbound resolvers expect one signature for each algorithm in the zone apex
- If not, they suspect a downgrade attack
- and fail validation :-(

- We've tested this:
  - Out of 10.000 RIPE Atlas probes only 6 failed :-)

The .se Algorithm Rollover

- .se has 1.4 Million registered domains
- > 50% signed with DNSSEC
- ~ 70% of Swedish users rely on validating resolvers
- First algorithm rollover ever:
  - From RSA/SHA-1 to RSA/SHA-256

3 Measurement Types

- Monitor publication delay
- Monitor propagation delay
- Monitor the trust chain

Algorithm Rollover Stages

```
--------------------------------------------------------------------------------------------------------------------------
 initial             new RRSIGs          new DNSKEY            new DS             DNSKEY removal      RRSIGs removal
--------------------------------------------------------------------------------------------------------------------------
Parent:
 SOA_0 -----------------------------------------------------> SOA_1 ----------------------------------------------------->
 RRSIG_par(SOA) --------------------------------------------> RRSIG_par(SOA) -------------------------------------------->
 DS_K_1 ----------------------------------------------------> DS_K_2 ---------------------------------------------------->
 RRSIG_par(DS_K_1) -----------------------------------------> RRSIG_par(DS_K_2) ----------------------------------------->

Child:
 SOA_0               SOA_1               SOA_2                -------------------->SOA_3               SOA_4
 RRSIG_Z_10(SOA)     RRSIG_Z_10(SOA)     RRSIG_Z_10(SOA)      -------------------->RRSIG_Z_10(SOA)
                     RRSIG_Z_11(SOA)     RRSIG_Z_11(SOA)      -------------------->RRSIG_Z_11(SOA)     RRSIG_Z_11(SOA)

 DNSKEY_K_1          DNSKEY_K_1          DNSKEY_K_1           -------------------->
                                         DNSKEY_K_2           -------------------->DNSKEY_K_2          DNSKEY_K_2
 DNSKEY_Z_10         DNSKEY_Z_10         DNSKEY_Z_10          -------------------->
                                         DNSKEY_Z_11          -------------------->DNSKEY_Z_11         DNSKEY_Z_11
 RRSIG_K_1(DNSKEY)   RRSIG_K_1(DNSKEY)   RRSIG_K_1(DNSKEY)    -------------------->
                                         RRSIG_K_2(DNSKEY)    -------------------->RRSIG_K_2(DNSKEY)   RRSIG_K_2(DNSKEY)
 --------------------------------------------------------------------------------------------------------------------------
```

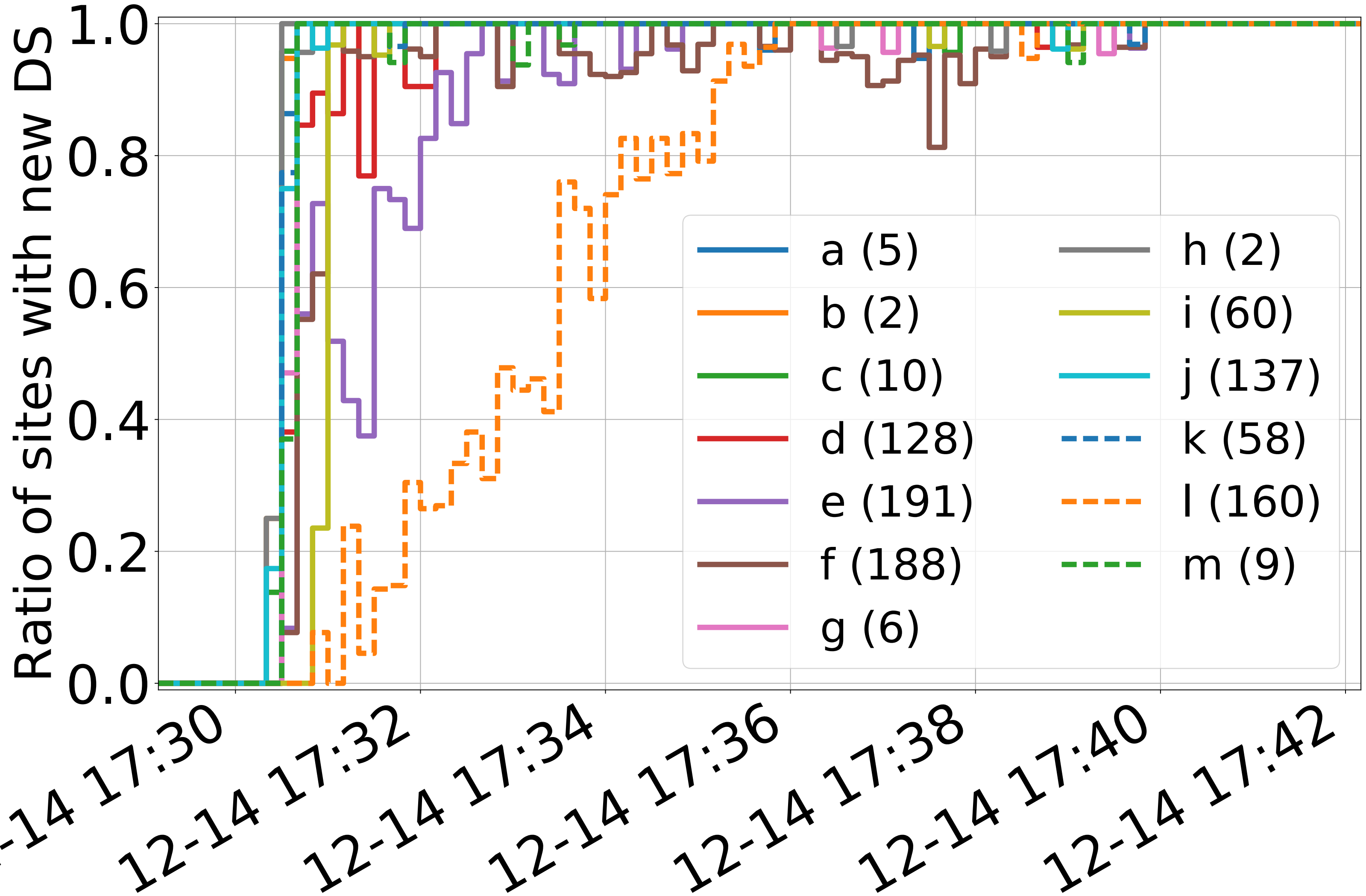
from RFC 6781, Figure 8

Publication Delay

- Using 10.000 RIPE Atlas probes
- Query the authoritative NSes directly

Publication Delay

- Using 10.000 RIPE Atlas probes
- Query the authoritative NSes directly



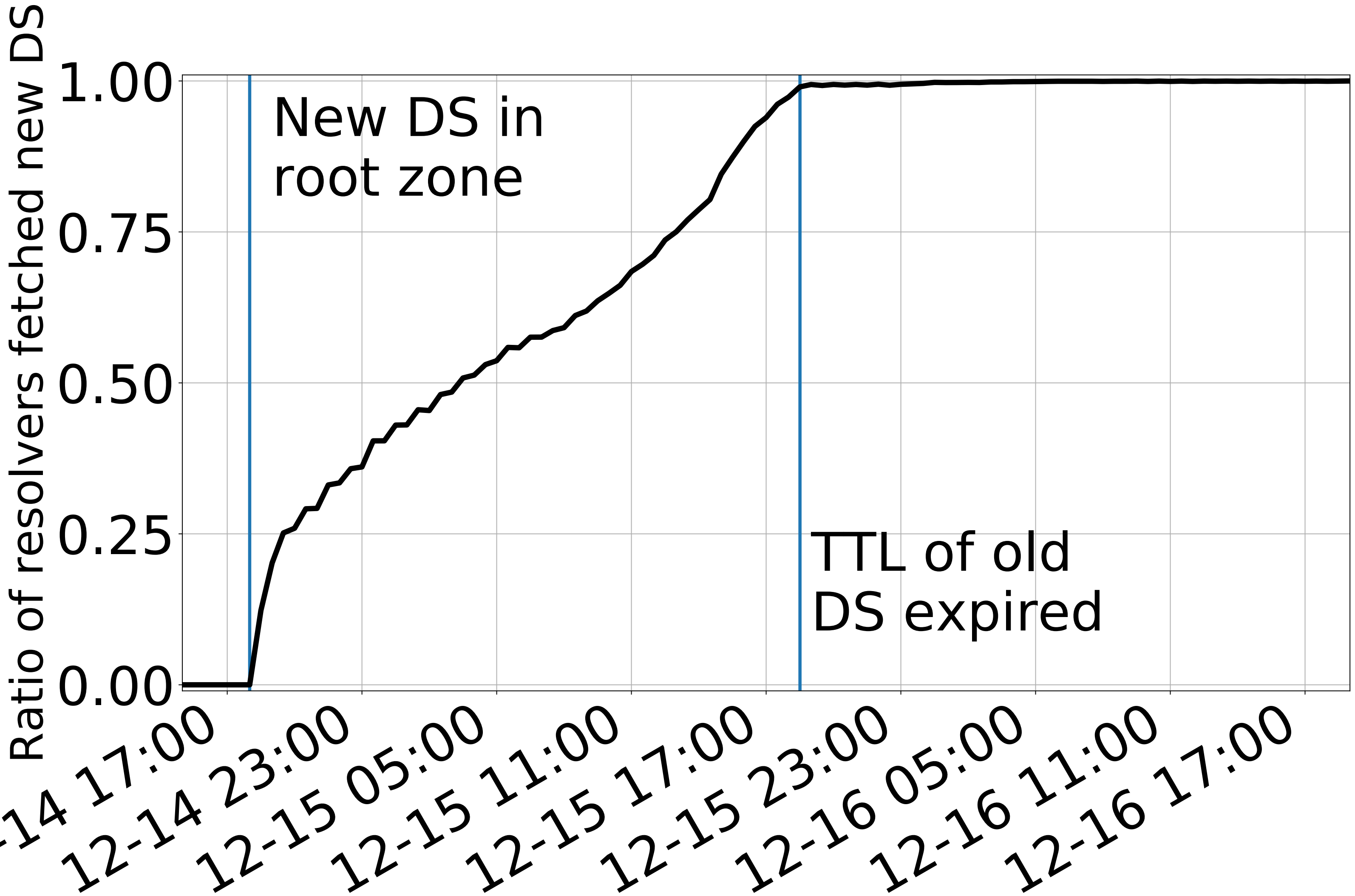
Northeastern UNIVERSITY OF TWENTE. SIDN LABS

Propagation Delay

- Using 10.000 RIPE Atlas probes
- Query for the new record using the probe's resolver

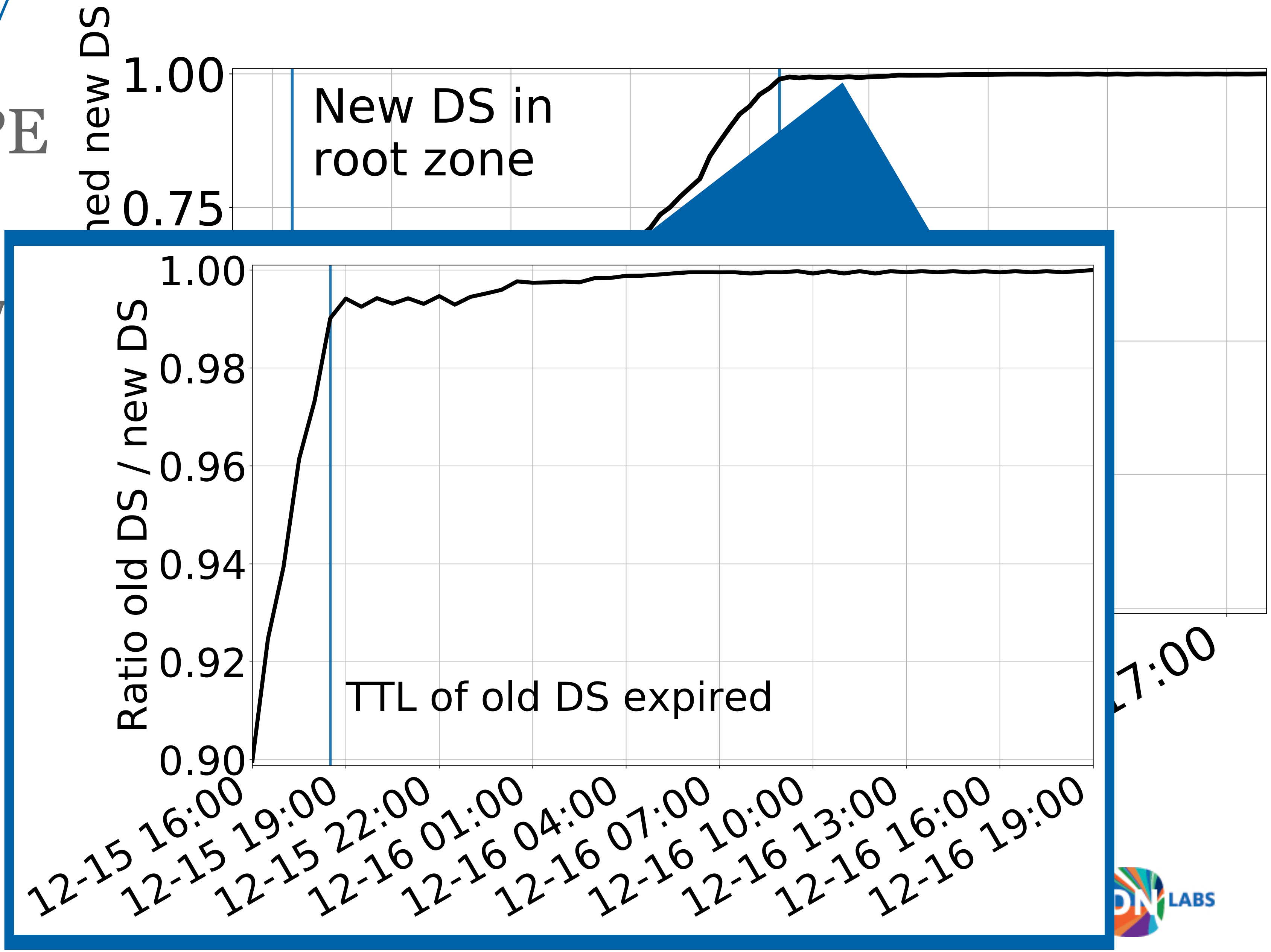
Propagation Delay

- Using 10.000 RIPE Atlas probes
- Query for the new record using the probe's resolver



Propagation Delay

- Using 10.000 RIPE Atlas probes
- Query for the new record using the probe’s resolver



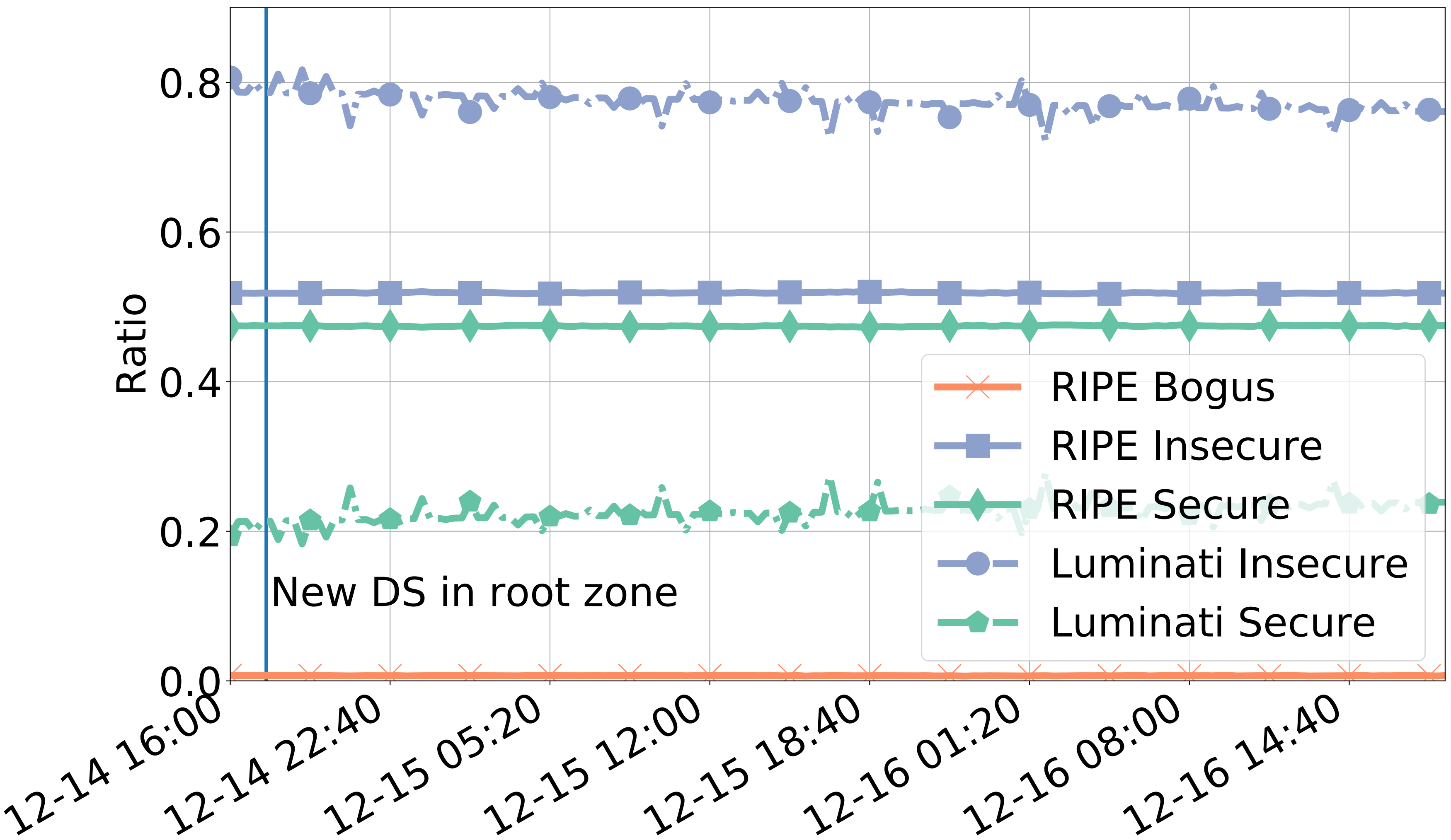
Timing of the Stage

• Publication delay:	~ 10 minutes
• Propagation delay:	~ 48 hours
• Move to next stage after:	~ 48 hours, 10 minutes

Monitor the Trust Chain

- Using 10.000 RIPE Atlas probes
- Luminati Network
- >46.000 VPs, > 8.000 behind validating resolvers
- Test-domains with valid and bogus records
- Which gives us three resolver states:
  - Validating, non-validating and bogus

Monitor the Trust Chain



Summary

- .se rollover was successful
- Conservative algorithm rollover not necessary
- Take your time

Monitor your own Rollover

- Measurements described at sidnlabs.nl
- Tool to automate the rollover available soon
- Detailed paper available soon (if it gets accepted)
- More information about the .se rollover:
  - Preparation
  - Lessons learned

Thanks

- to IIS, the operators of .se
- to RIPE

Thanks

- to IIS, the operators of .se
- to RIPE

Questions?

Moritz Müller

moritz.muller@sidn.nl

@moritzcm_