

DNSSEC signer migration

A step-by-step guide

Author(s): Roland van Rijswijk - Deij

Version: 1.0

Date: August 2012

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Motivation	3
1.3	Revision history	4
2	Signer migration process	5
2.1	Introduction	5
2.2	Notation	5
2.3	Assumptions	5
2.4	Guiding principles	6
2.5	Step 1: starting situation	7
2.6	Step 2: cleaning up the source signer	8
2.7	Step 3: configuring the destination signer	9
2.8	Step 4: introduce source keys on destination signer	10
2.9	Step 5: introduce destination keys on source signer	11
2.10	Step 6: create a fully cross-signed zone	12
2.11	Step 7: switch the DS record	13
2.12	Step 8: switch over to the destination signer	14
2.13	Step 9: resume automated key management	15
Appendix A	Worksheet	16
Appendix B	Using tools to help with the migration	18
B.1	Introduction	18
B.2	Example output of DNSViz	18

1 Introduction

1.1 Purpose

The purpose of this document is to describe the steps required to perform a DNSSEC signer migration. By documenting this process, we hope to help others who are facing a similar issue. Furthermore, we believe that this process can also easily be extended to serve as a template for DNS operator migration.

This document and the scenarios contained in it have been composed with the assistance of OpenFortress (<http://www.openfortress.nl/>).

1.2 Motivation

In 2010 SURFnet deployed automated DNSSEC signing in its “SURFdomeinen” managed DNS service. One of the design decisions that we made during the implementation of DNSSEC in this service was to use “shared keys” (i.e. sharing the same set of cryptographic keys for multiple DNS zones, instead of using separate keys for each individual zone).

There were two reasons for this. First, the HSMs that we selected for secure key storage only support storing a limited number of objects. Secondly, we saw this as a way to simplify key management by only using one set of keys per user (in our case connected institutions such as a university or teaching hospital). Another decision that we made early on was to use OpenDNSSEC as software suite for our deployment.

Using shared keys has a number of drawbacks:

- It is harder to migrate zones to a different DNS operator due to the fact that this requires stopping key rollover for the zone; in the case of shared keys this affects all zones that share a key set
- The state of a zone cannot develop independently; the zone with the highest overall TTL determines the speed at which key rollovers can proceed and adding a zone with a higher overall TTL may pose a problem
- Using shared keys creates an artificial tie between zones that are otherwise independent of one another
- KSK rollover is problematic for the case where a large number of zones shares one key set because of the number of parent interactions and possibly because of differing policies on the parent side when different top-level domain registries are involved
- Synchronised key rollover as is the case with shared keys may impose additional load on the infrastructure because caches will be sending more requests for all zones for which the keys are rolled

On top of that, the OpenDNSSEC version that we are currently using (version 1.1) does not have optimal support for shared keys.

For this reason, we have decided to upgrade our HSMs to a version that supports storage of a much larger number of keys and to simultaneously migrate away from using shared keys. To do this in one go, we are going to migrate to new signers systems that are also running a newer version of

OpenDNSSEC (version 1.3). Effectively, this means that we are performing a signer migration or signer rollover.

1.3 Revision history

Version	Date	Author	Changes
0.1	2011-09-19	R.M. van Rijswijk-Deij	Initial version
0.2	2011-11-09	R.M. van Rijswijk-Deij	Reworked after review by OpenFortress
0.3	2012-07-27	R.M. van Rijswijk-Deij	Incorporated lessons-learned from the actual migration
1.0	2012-08-10	R.M. van Rijswijk-Deij	Finalised after review by OpenFortress

2 Signer migration process

2.1 Introduction

In this chapter we will describe the signer migration process step by step. We will show the process from two perspectives. The first perspective is that of the *source signer*, by which we mean the signer that is currently actively signing a zone. The second perspective is that of the *destination signer*, i.e. the signer system to which we want to migrate responsibility for signing a zone.

We have included a worksheet in Appendix A that can be used to record useful data required during the migration process. We recommend that you print a copy of this worksheet before performing the migration process for a zone.

2.2 Notation

Each step is illustrated using a diagram. For these diagrams, we have used consistent arrow types and colour coding:

- Blue indicates (part of) the chain of trust that uses the active key set on the *source signer*
- Red indicates (part of) the chain of trust that uses the active key set on the *destination signer*
- Grey indicates (part of) a chain of trust that is no longer used or published
- An arrow indicates a validation based on the object at the origin of the arrow for the object that the arrow points to (in most cases this implies a signature)
- A dashed line in an arrow indicates that this relationship may exist but does not necessarily have to exist
- “Self-references” (i.e. KSK signatures over the KSK) are omitted for reasons of clarity

To further clarify the diagrams independent of the colour coding we have included subscripts for each object. The following subscripts are used:

- *src* – indicates that this object belongs to or originates from the *source signer*
- *dst* – indicates that this object belongs to or originates from the *destination signer*
- *old* – indicates that this object is an old key that is no longer actively being used to sign data
- *act* – indicates that this object is a key that is actively being used to sign data
- *pre* – indicates that this object is a key that is being pre-published and may be the next in line for a key rollover

Finally, we always show the situation on both the *source* as well as the *destination* signer. The signer that is authoritatively serving the zone has thicker lines and a **bold** heading at the top of the diagram.

2.3 Assumptions

At the start of this process, the following conditions are assumed to be true:

- There is no KSK rollover in progress on the *source signer*

- The validity of the signatures is significantly longer than the maximum TTL in the zone that is to be migrated (at least 4 times as long)
- Source and destination signer function as a hidden primary (i.e. are not authoritative name servers that are part of the NS-set for the domain); consequently, no change to the NS-set for any signed domains is required to facilitate the migration
- There is a means to switch signers (e.g. reconfiguring the public primary name server to source the zone data from a different signer)
- Key management and zone signing are separate processes (as is for instance the case when using OpenDNSSEC)
- A split-key policy is used (i.e. with separate ZSK and KSK)
- The parent zone allows a single DS for a secure delegation

2.4 Guiding principles

The following guiding principles were used while designing this process:

- The DNSKEY RRset may only consist of keys that are active on one or both signers
- Manual zone editing shall be kept to a minimum
- The migration must take place as quickly as possible

2.5 Step 1: starting situation

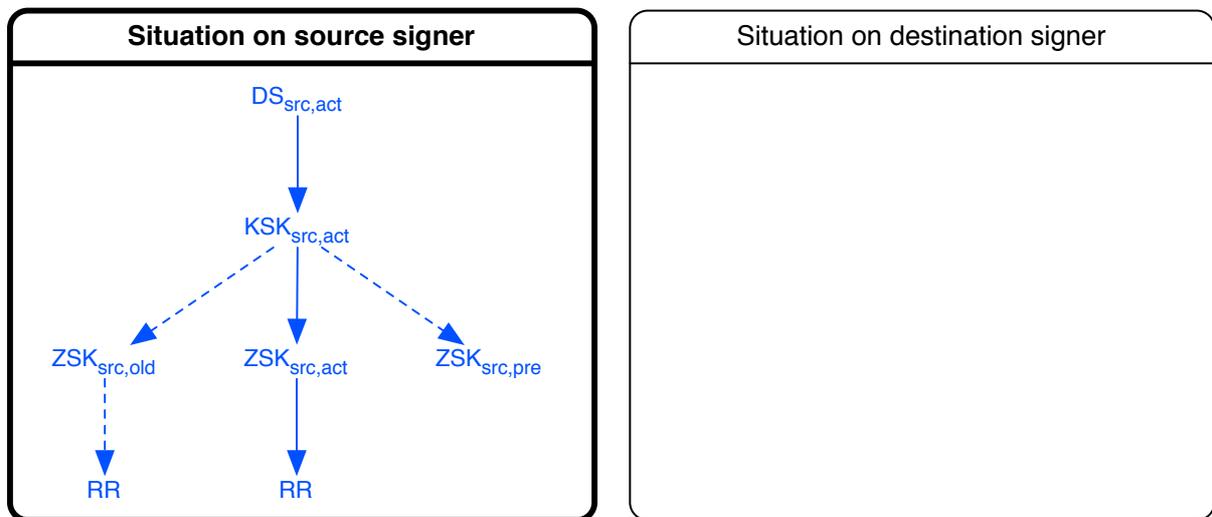


Figure 1 – Step 1: starting situation

Figure 1 shows the starting situation on the *source signer*. The starting situation shown is a typical snapshot of the state of a signer that uses the “ZSK pre-publication” rollover strategy in which a ZSK is pre-published before it is made active and in which old signatures are gradually rolled to the new key during a key rollover.

The diagram shows the active DS in the parent zone at the top. This DS “certifies” the active KSK for the zone, depicted below it. In turn, this KSK signs the entire DNSKEY set, which will include an active ZSK (depicted in the middle, directly below the KSK), and may include an old ZSK (which is still being published because there are still signatures in use that depend on it) and may include a new pre-published ZSK.

The authoritative name servers are publishing zone data that is output by the *source signer* at this moment in the process. This continues to be the case until the document explicitly mentions that this changes.

Record the characteristics such as the maximum zone TTL and the SOA serial number of the zone. You can use the worksheet included in Appendix A to do this.

2.6 Step 2: cleaning up the source signer

The purpose of this step is to clean up the *source signer* and to transition it to the situation shown in Figure 2.

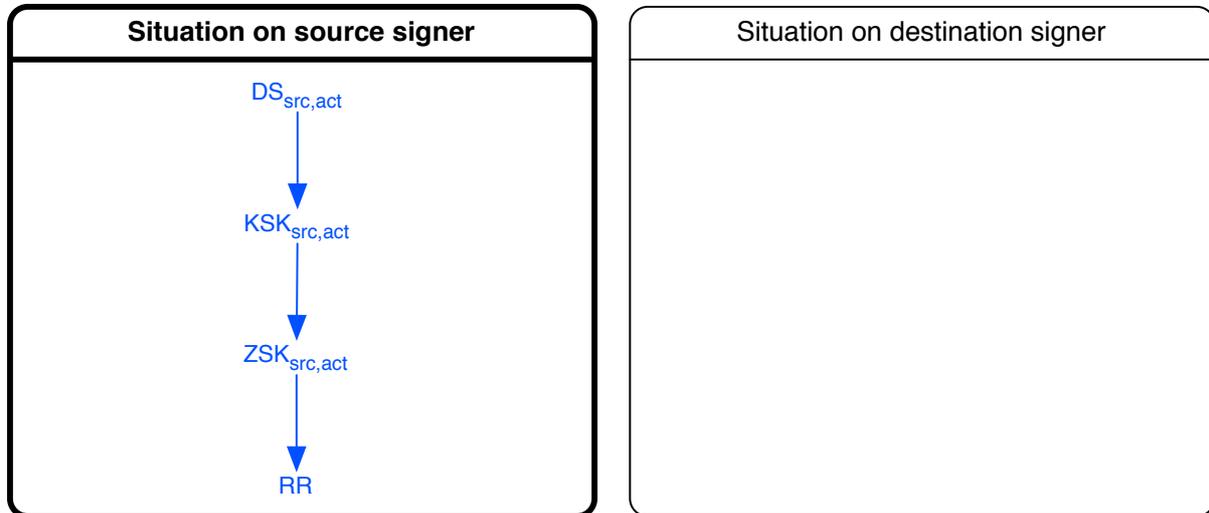


Figure 2 - Situation after step 2

To reach this state, the following sub-steps need to be taken:

1. Stop automated uploads of the input zone to the *destination signer*
2. Stop active key management (when using OpenDNSSEC this means stopping the “enforcer” component)
3. Alter the signer configuration such that only the active ZSK is published as part of the DNSKEY set (when using OpenDNSSEC this means manually editing the signer configuration for the zone that is being migrated)
4. Force the signer to generate a zone that only contains fresh signatures for the active ZSK (when using OpenDNSSEC this means deleting the signed zone and all intermediate files and re-running the signer)
5. Set the SOA serial number in the input zone such that it is higher than the currently published zone’s SOA serial number
6. Re-start the signer and make sure that a new zone with the new DNSKEY set and fresh signatures is published
7. Wait $\max\text{TTL}(\text{zone})$ after all authoritative name servers have received the new zone data for the new signatures to propagate to caches

Note: a side effect of this step is that all signatures have the maximum validity time at the start of the migration process.

2.7 Step 3: configuring the destination signer

The purpose of this step is to configure the *destination signer* and to transition it to the situation as shown in Figure 3.

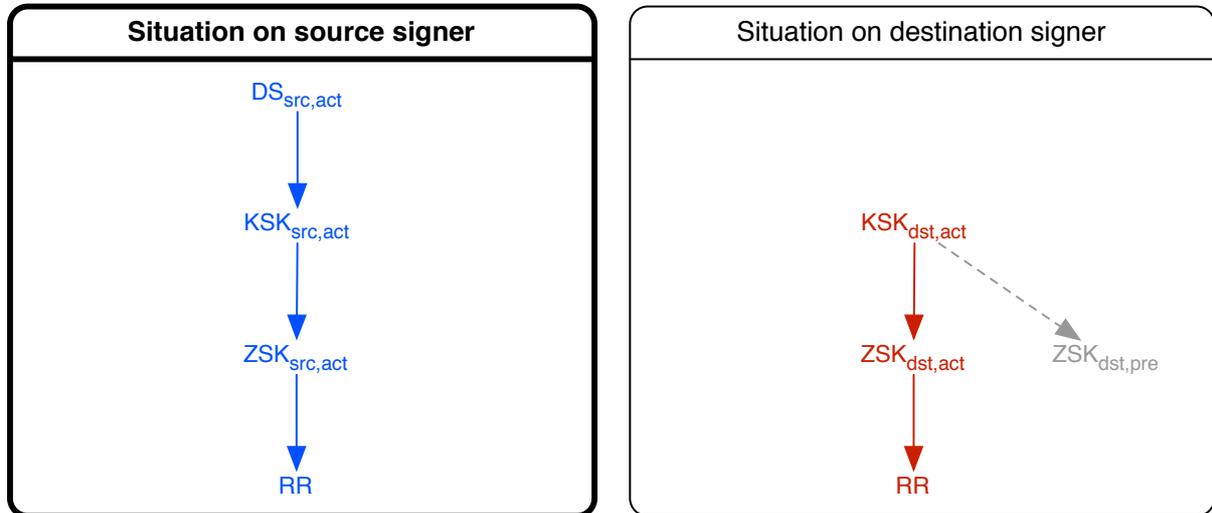


Figure 3 - Situation after step 3

To reach this situation, the following sub-steps need to be taken:

1. Configure the zone to be migrated on the new signer
2. Launch automated key management on the new signer but do not yet launch zone signing (when using OpenDNSSEC this means launching the “enforcer” component but not yet launching the “signer” component)
3. Once the signer configuration has been created by the automated key manager, you must stop automated key management again
4. Edit the signer configuration such that no new ZSK is pre-published if the key manager generated and configured such a key
5. Run the signer and allow it to generate an output zone with the trust chain as shown in Figure 3; stop the signer afterwards (or, alternatively, run the signer once if this is possible)

2.8 Step 4: introduce source keys on destination signer

The purpose of this step is to introduce the DNSKEY records for the active keys from the *source signer* on the *destination signer* and to get an RRSIG signature for the new DNSKEY set this will result in using the active keys on the *destination signer*. The end situation of this step is shown in Figure 4.

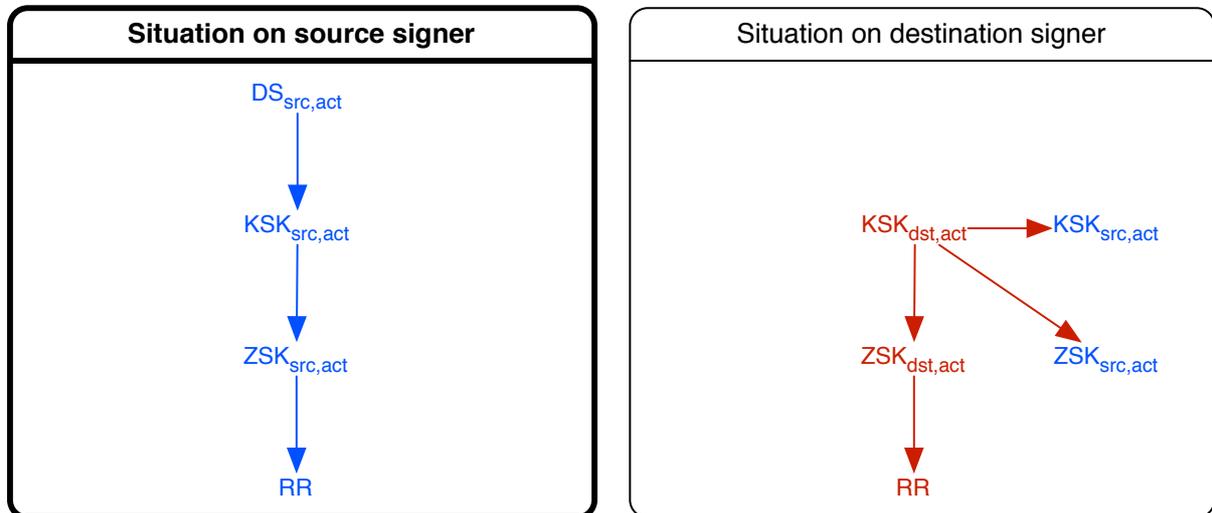


Figure 4 - Situation after step 4

To reach this situation, the following sub-steps need to be taken:

1. Stop automated uploads of the input zone to the *destination signer*
2. Save an unmodified copy of the input zone on the *destination signer*; this unmodified copy will be used at a later stage in the process
3. Alter the input zone on the *destination signer* to include the DNSKEY record for the active KSK from the *source signer*
4. Alter the input zone on the *destination signer* to include the DNSKEY record for the active ZSK from the *source signer*
5. Allow the signer software to run on the *destination signer* and ensure that the output zone includes a valid signature over the entire DNSKEY set, which should include the active KSK and ZSK from both the *source* and the *destination* signer; stop the signer after the zone has been generated (or, alternative, run the signer once if possible)
6. Save the RRSIG made with the active KSK on the *destination signer* over the DNSKEY RRset; you will need this record in the following steps

2.9 Step 5: introduce destination keys on source signer

The purpose of this step is to introduce the DNSKEY records for the active keys from the *destination signer* on the *source signer*. The situation at the end of this step is shown in Figure 5.

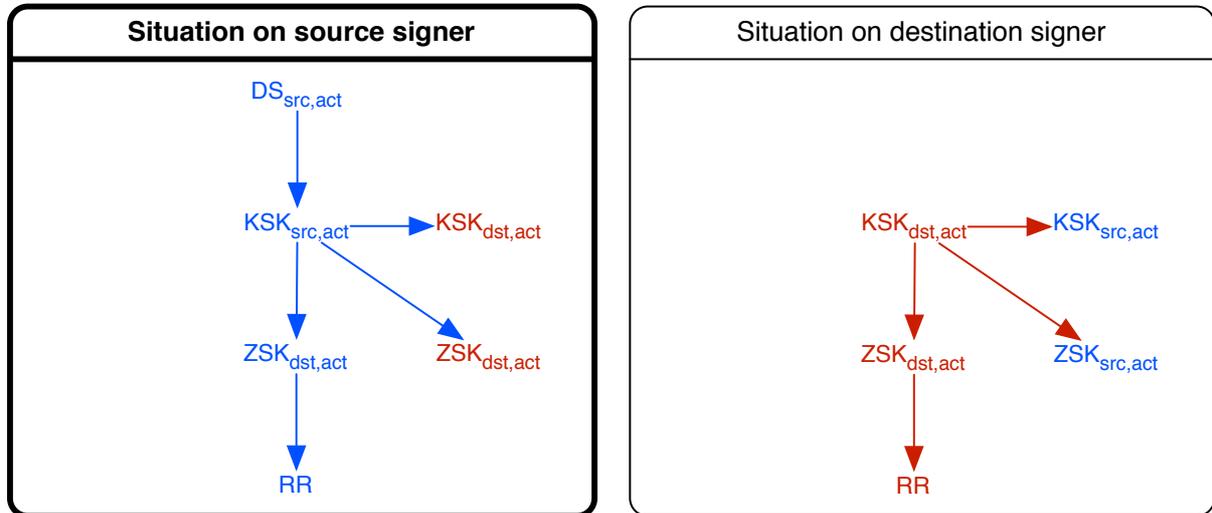


Figure 5 - Situation after step 5

To reach this situation, the following sub-steps need to be taken:

1. Stop the signer software on the *source signer* and ensure that zone publication is temporarily halted (i.e. no new versions of the zone are pushed from the signer to the authoritative name servers)
2. Alter the input zone on the *source signer* to include the DNSKEY record for the active KSK from the *destination signer*
3. Alter the input zone on the *source signer* to include the DNSKEY record for the active ZSK from the *destination signer*
4. Set the SOA serial number in the input zone such that it is higher than the SOA serial number of the zone that was published in step 2 (§2.6)
5. Allow the signer software to run once on the *source signer* and ensure that the output zone includes a valid signature over the entire DNSKEY set, which should include the active KSK and ZSK from both the *source* and the *destination* signer
6. **Do NOT resume zone publication (i.e. do not push new zones from the signer to the authoritative name servers)**

2.10 Step 6: create a fully cross-signed zone

The purpose of this step is to create a fully cross-signed zone that includes the key set from the *source* as well as the *destination* signer. The situation at the end of this step is shown in Figure 6.

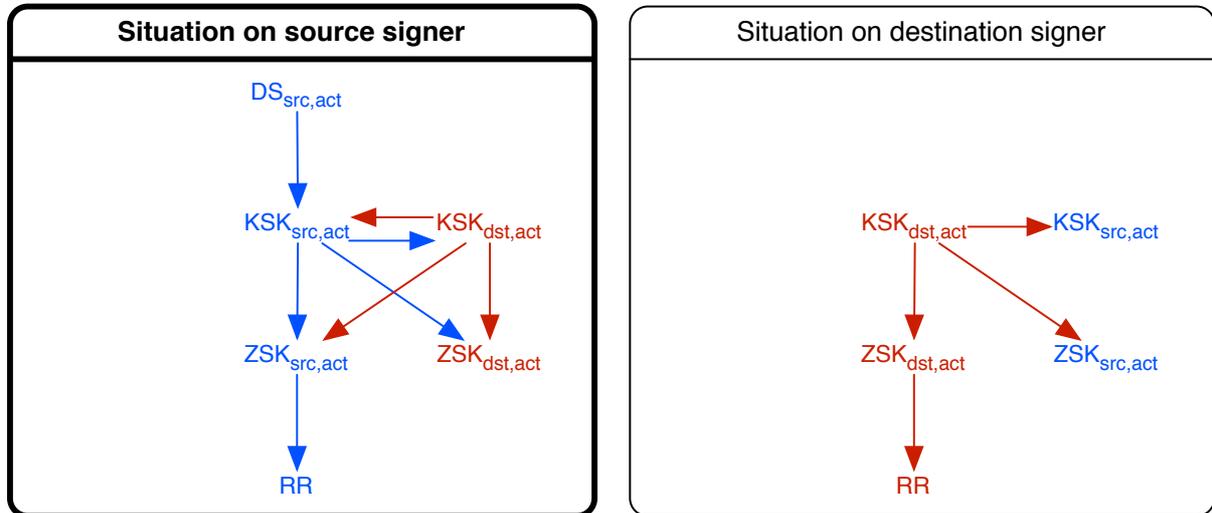


Figure 6 - Situation after step 6

To reach this situation, the following sub-steps need to be taken:

1. Edit the **output** (i.e. the signed) zone on the *source signer* and include the RRSIG made with the active KSK from the *destination signer* over the DNSKEY set saved at the end of step 4
2. Publish this edited zone on the authoritative name servers
3. **Do NOT restart the signer component on the source signer**
4. Wait TTL(DNSKEY) for the DNSKEY set and the associated signatures to propagate to caches

2.11 Step 7: switch the DS record

The purpose of this step is to switch the DS for the zone to point to the KSK of the *destination signer*. The situation at the end of this step is shown in Figure 7.

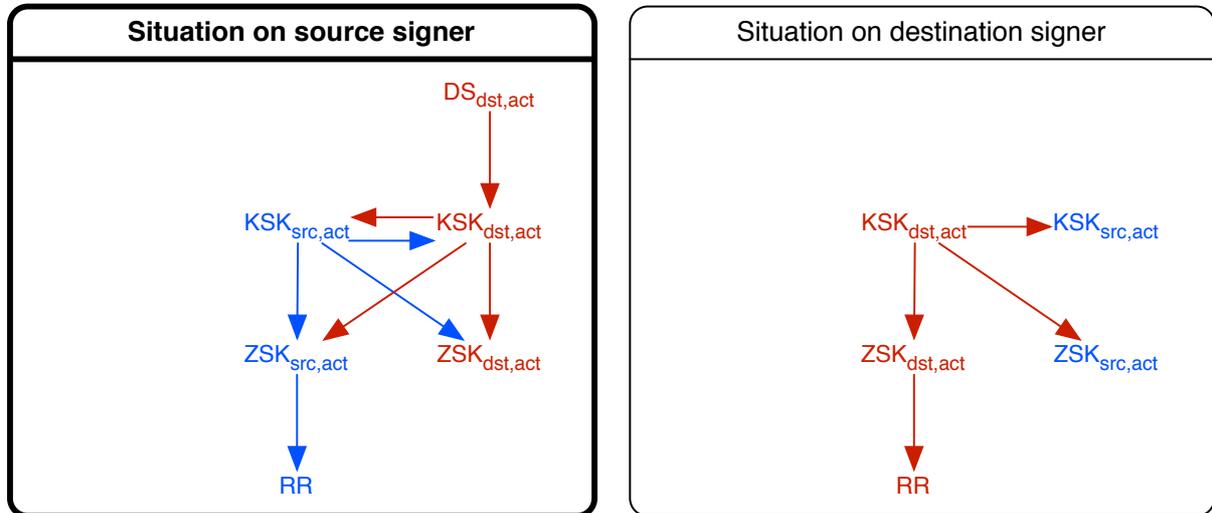


Figure 7 - Situation after step 7

To reach this situation, the following sub-steps need to be taken:

1. Contact the parent zone (registry) to submit the new DS and to retract the old DS (some registries may allow you to do this in one interaction, others will have different policies for handling this situation)
2. Wait for the new DS to appear in the parent zone
3. Wait for the old DS to disappear from the parent zone
4. Wait TTL(DS) to allow the DS exchange to propagate to caches

2.12 Step 8: switch over to the destination signer

The purpose of this step is to completely switch over to the *destination signer*. At the end of this step, continuous zone signing has been restarted and will only take place on the *destination signer*, zone publication will have been resumed and will use the output from the *destination signer*. The situation at the end of this step is shown in Figure 8 (note that the situation on the *source signer* is greyed-out since the *source signer* will no longer be in use for the zone that is being migrated by the end of this step).

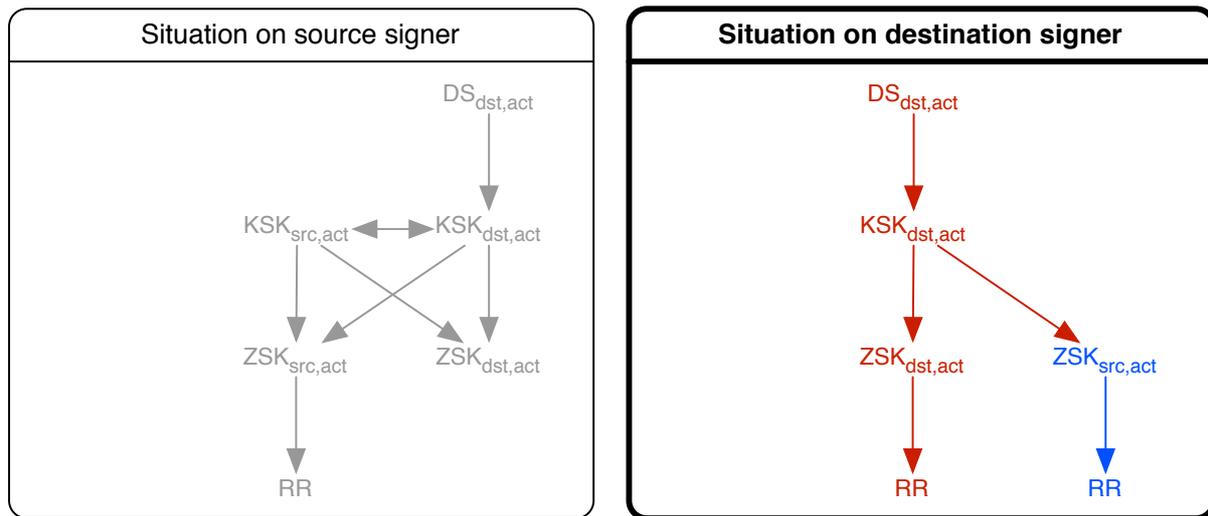


Figure 8 - Situation after step 8

To reach this situation, the following sub-steps need to be taken:

1. Make the latest clean (i.e. unmodified by the steps of this process) input zone available as input on the *destination signer*
2. Add the DNSKEY record for the active ZSK from the *source signer* to the input zone
3. Set the SOA serial number such that it is higher than the SOA serial number of the currently published zone
4. Remove any signed zone output and intermediate data on the *destination signer*
5. Restart automated signing on the *destination signer*
6. **Do NOT yet restart automated key management or zone upload, this will be done later**
7. Reconfigure your authoritative name servers to source their data from the *destination signer*
8. Wait for the newly signed zone from the *destination signer* to be published on your authoritative name servers
9. Wait $\text{maxTTL}(\text{zone})$ for all the signatures created with the active ZSK from the *source signer* to disappear from caches and be replaced by signatures created with the active ZSK from the *destination signer*

Note that the active ZSK from the *source signer* needs to be published because caches will still contain signatures over resource record sets created with this key. It is therefore still part of the trust chain.

2.13 Step 9: resume automated key management

The purpose of this step is to remove the DNSKEY record for the active ZSK from the *source signer* from the input zone and to resume automated key management. Once this step has taken place, the migration is complete. The situation at the end of this step is shown in Figure 9.

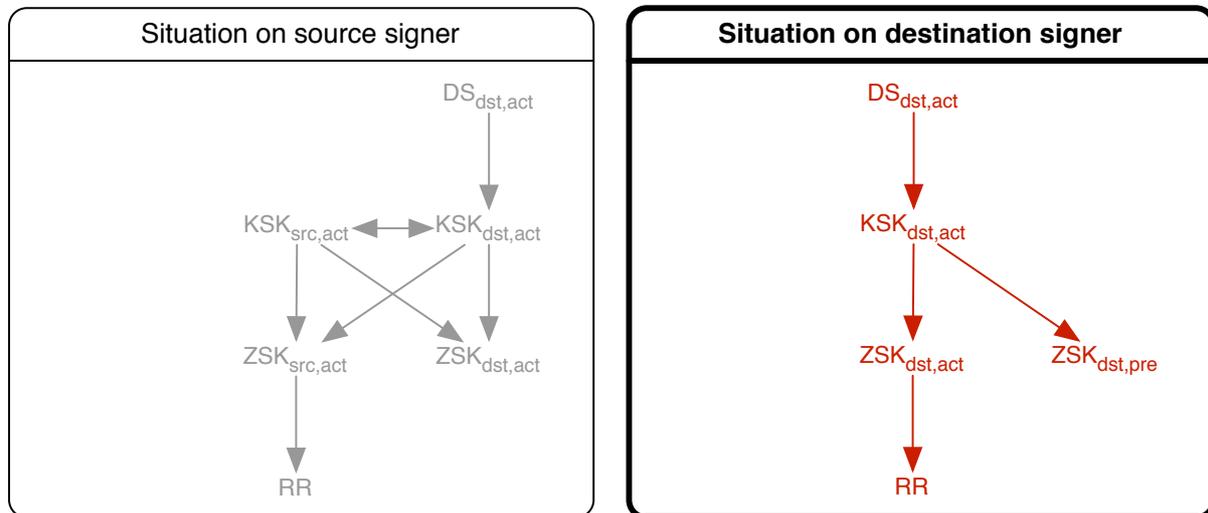


Figure 9 - Situation after step 9

To reach this situation, the following sub-steps need to be taken:

1. Remove the active ZSK from the *source signer* from the input zone
2. If required, update the SOA serial number in your backend system such that it is higher than the SOA serial number that is currently published
3. Resume automated uploads of the input zone to the *destination signer*; if you still have zones that are signed and published by the *source signer* then you can now also resume automated upload of input zones to the *source signer*
4. Resume automated key management on the *destination signer* (in case OpenDNSSEC is used, you can now restart the enforcer component); if you still have zones that are signed and published by the *source signer* you can also resume automated key management on the *source signer*

At the end of this step, the migration is complete.

Appendix A Worksheet

Zone name:

Key identifiers

	<i>source signer</i>	<i>destination signer</i>
Active KSK:	<input type="text"/>	<input type="text"/>
Active ZSK:	<input type="text"/>	<input type="text"/>
Old ZSK (optional):	<input type="text"/>	<input type="text"/>
Pre-published ZSK (optional):	<input type="text"/>	<input type="text"/>

Zone characteristics

No KSK rollover in progress

No ZSK rollover in progress

Maximum TTL:	<input type="text"/>	Signature validity:	<input type="text"/>
DNSKEY set TTL:	<input type="text"/>	Starting SOA:	<input type="text"/>
DS TTL:	<input type="text"/>		

Notes for 2.6 – Step 2: cleaning up the source signer

Upload of the zone to the source signer stopped	at:	<input type="text"/>	:	<input type="text"/>	h
Active key management on the source signer stopped	at:	<input type="text"/>	:	<input type="text"/>	h
SOA serial for zone with fresh signatures	#:	<input type="text"/>			
Zone with fresh signatures and only active KSK published	at:	<input type="text"/>	:	<input type="text"/>	h
Wait maximum TTL	until:	<input type="text"/>	:	<input type="text"/>	h

Notes for 2.7 – Step 3: configuring the destination signer

New key set generated on destination signer, key IDs noted above

Notes for 2.8 – Step 4: introduce source keys on destination signer

Upload of the input zone to the destination signer has been stopped

Copy of unmodified input zone saved as:

RRSIG over DNSKEY set saved as:

Notes for 2.9 – Step 5: introduce destination keys on source signer

Zone publication from the *source signer* has been stopped at:

Zone signing on the *source signer* has been stopped at:

SOA serial number of the zone created in this step #:

Notes for 2.10 – Step 6: create a fully cross-signed zone

Fully cross-signed zone published at:

Wait TTL(DNSKEY) until:

Notes for 2.11 – Step 7: switch the DS record

New DS published in parent zone at:

Old DS published in parent zone at:

Wait TTL(DS) until:

Notes for 2.12 – Step 8: switch over to the destination signer

ZSK from source signer included in clean input zone on destination signer

SOA serial for zone created in this step #:

Zone from destination signer published at:

Wait maximum TTL until:

Notes for 2.13 – Step 9: resume automated key management

Upload of the zone to the destination signer resumed at:

Active key management on the destination signer resumed at:

Appendix B Using tools to help with the migration

B.1 Introduction

During our migration we used two tools to help us check that our zones remained valid:

- DNSViz – this tool, developed by Sandia National Laboratories in the US, visualises the chain of trust for a DNSSEC-signed domain. It shows very clearly which keys are in use and what trust chains exist. B.2 shows an example of the output of DNSViz for one of the domains we migrated. DNSViz can be found at <http://dnsviz.net/>
- DNSCheck – this tool, developed by IIS - the registry for the Swedish ccTLD .se, takes a domain name as input and validates all aspects of the DNS structure for a domain including valid DNSSEC data. DNSCheck can be found at <http://dnscheck.iis.se/>

B.2 Example output of DNSViz

In preparation for our actual migration, we migrated a test domain, surfnt.org, from our old infrastructure to our new infrastructure. Below, we have included three diagrams produced by DNSViz that show some of the steps in the migration process:

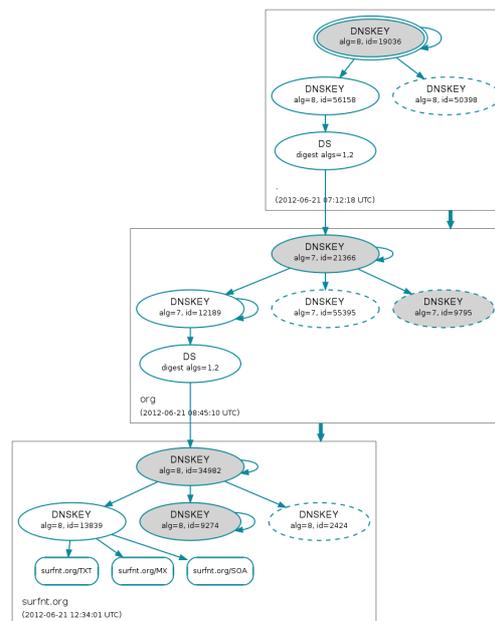


Figure 10 - Situation for surfnt.org with a cross-signed zone

Figure 10 above shows the situation after Step 6: create a fully cross-signed zone, described in section 2.10. The bottom rectangle in the diagram shows the zone data for surfnt.org. The grey ellipses are the KSKs, where the KSK with ID 34982 is the KSK from the *source signer* and the KSK with ID 9274 is the KSK from the *destination signer*. Likewise, the ZSK (white ellipses) with ID 13839 is the ZSK from the *source signer* and the ZSK with ID 2424 is the ZSK from the *destination signer*. Note that, unfortunately, DNSViz does not show the signatures created with the KSK from the *destination signer* over the key set.

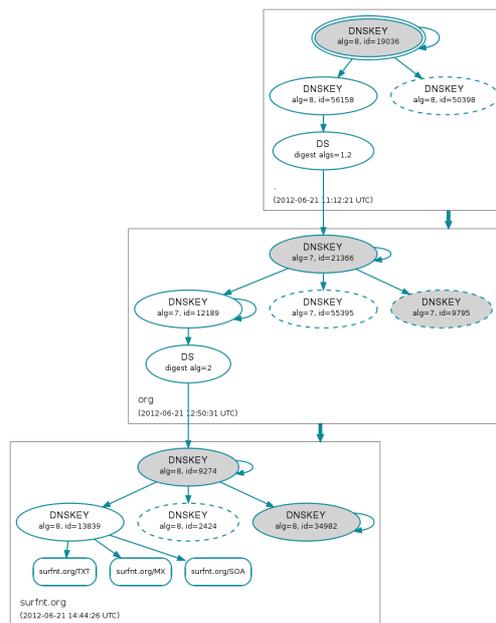


Figure 11 - DS has now changed to point to KSK from destination signer

5. Figure 11 above shows the situation after

Step 7: switch the DS record described in section 4. The KSK with ID 9274 from the *destination signer* is now the secure entry point into the zone. Note that the ZSK in use is still the ZSK with ID 13839 from the *source signer*.

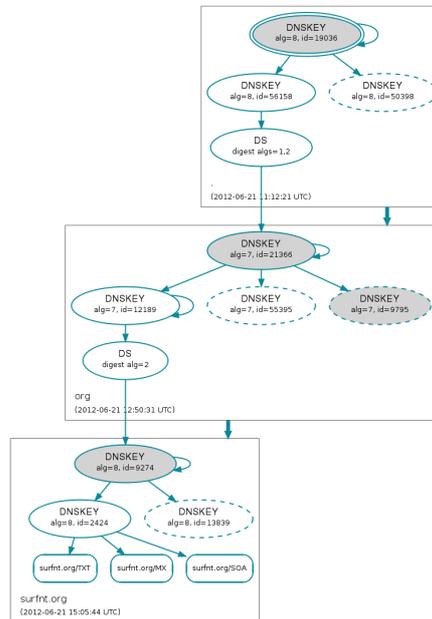


Figure 12 - Situation after migration to the destination signer

5. Finally, Figure 12 above shows the situation after

Step 8: switch over to the destination signer described in section 4. In the diagram, the chain of trust is now formed by the KSK with ID 9274 and the ZSK with ID 2424 both of which are from the *destination signer*. Note that the ZSK with ID 13839 from the *source signer* is still in the key set at this stage since signatures made with this key may still exist in caches. At the end of the migration, this ZSK will also be gone.