Domain Name System Operations
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2011

W. Mekking
NLnet Labs

February 25, 2011

DNSSEC Key Timing Considerations Follow-Up
draft-mekking-dnsop-dnssec-key-timing-bis-00

Abstract

This document describes issues surrounding the timing of events in enforcing key policy within DNSSEC. It presents timelines for various key rollovers and changes into the policy with respect to the key signing scheme. It explicitly identifies the relationships between the various parameters affecting the process.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2011.

Copyright Notice

Table of Contents

1. Introduction

A zone is managed according to a given security policy. Such a policy may enforce DNSSEC keys to be used and for how long. When enforcing a lifetime on DNSSEC keys, key rollovers must to take place. In addition, changes in the policy may trigger certain key rollover events. Key rollovers are time critical, multiple steps processes. This document describes issues surrounding the timing of events in the rolling of DNSSEC keys.

[MM: Editorarial comments are indicated by square brackets and editor initials]

1.1. Key Rollover Considerations

A key is used with a purpose: An operational decision has been made to secure the zone with DNSSEC. That decision leads to a key being created, published in the zone and used for signing. Policy may enforce a lifetime on keys. As a result, current active keys need to be replaced with new keys. The new key becomes active, while the current key is retired. The keys need to be introcuded into and removed from the zone at the appropriate times. Considerations that must be taken into account are:

- Speed: A rollover should occur as fast and simple as possible. However, DNSSEC records are not only held at the authoritative nameserver, they are also cached at client resolvers. The data on these systems can be interlinked, meaning a validating may try to validate a signature retrieved from a cache with a key obtained separately. The rollover process needs to happen in such a way that at all times through the rollover the information is consistent.

- Size of the zone and the DNS response: A rollover can be speed up by introducing the DNSSEC records prematurely. However, adding arbitrary signatures increases the size of your zone and DNS responses significantly. To keep the sizes of the zone and responses as small as possible, you might want to consider to introduce the DNSSEC records only when they are required, For the same reason, dead keys and signatures must be removed periodically.

- Size of the DNSKEY RRset and the priming response: You can choose to keep the size of the DNSKEY RRset to a minimum, to make priming responses smaller in size. The larger the packet, the more resolvers may have problems retrieving the response. Other responses may have more signatures, since the initial size is relatively small. The DNSKEY RRset is usually already quite large

and should not grow too much anymore.

- Interactions with the Parent: A KSK sometimes needs its corresponding DS record to be published at the parent zone, while its predecessor needs to remove its DS record from the parent zone. Such a request requires additional operational work and can be a sufficient delay. Ideally, the interactions with the parent is kept to a minimum.

1.1.1. Key Goals

We have identified three different goals for a key:

- Activate key: Make validating resolvers use the key's associated information to perform authentication.
- Remove key: Make validating resolvers forget about the key's associated information.
- Stand-by key: Pre-publish information for this key to speed up a future (unscheduled) rollover.

Each key rollover and change in key signing scheme can now be described by one or more goals that are put on a key.

1.2. Terminology

The terminology used in this document is as defined in [RFC4033] and [RFC5011].

2. Key Definitions

2.1. Key Types

Keys can be used to authenticate information within the zone. Such keys are said to be ZSKs. In addition, keys can be used to authenticate the DNSKEY RRset in the zone. These keys are said to be KSKs. Keys can be marked to be ZSK and KSK at the same time, for example in a Single Type Signing Scheme (STSS).

Despite that ZSK and KSK only describe the usage of a key, the terms are often used for identifiying a key. Thus when we talk about a ZSK, we actually mean that the key is used as ZSK. In the same spirt, a KSK is a key that is used as KSK.

DNSSEC recognises the classification of keys with its SEP bit set and not set. Usually if a key is used as KSK, the SEP bit is set. However, draft-ietf-dnsext-dnssec-bis-updates [dnssec-bis] says that

a SEP bit setting has no effect on how a DNSKEY may be used. Policy determines whether the bit should be set, depending on the key's usage.

2.2. Key States Unraveled

We use unraveled key states to seperately represent the key and its associated information. There can be up to three pieces of key associated information: the public key (in DNSKEY format), its created signatures (the RRSIG records) and the secured delegation (the corresponding DS record). The state of the piece of information is defined by 'RRtype State'.

Key conditions are essentially what are called key states in draft-ietf-dnsop-dnssec-key-timing [key-timing]. A key can have multiple conditions at the same time.

A piece of information may exist in up to two places: it can be present in the corresponding zone and it may live in resolver caches. This is true for every piece of associated information. Therefore, all of the three pieces of information follow the same state diagram:

Generated --> Introduced --> Propagated --> Withdrawn --> Dead.

Generated: The information has been generated, but is not available in the zone. In this state, no resolvers are able to fetch this information.
- The key condition is said to be Generated, if no information has passed the Introduced state yet.

Introduced: The information is introduced and, as a result, may be available in the zone. In this state, there may be resolvers that fetch this information.
- The key condition is said to be Published if it has its DNSKEY state in Introduced.
- The key condition is said to be Active if it has its RRSIG state in Introduced (for ZSKs).
- The key condition is said to be Submitted, or ActiveDS, if it has its DS state in Introduced (for KSKs).

Propagated: The information is available in the zone and enough time has passed to have it propagated into all resolver caches. As a result, all resolvers fetch this information from cache of from the authoritative name server.
- The key condition is said to be Known if it has its DNSKEY state in Propagated.
- The key condition is said to be Safe if it has its RRSIG state in Propagated (for ZSKs).

- The key condition is said to be SafeDS if it has its DS state in Propagated (for KSKs).

Withdrawn: The information is being withdrawn from the zone, but may still be available in the zone. In this state, the information can still live in resolver caches.
- The key condition is said to be Removed if it has its DNSKEY state in Withdrawn.
- The key condition is said to be Retired if it has its RRSIG state in Withdrawn (for ZSKs).
- The key condition is said to be RetiredDS if it has its DS state in Withdrawn (for KSKs).

Dead: The information is not available in the zone anymore and enough time has passed to have it expire from all resolver caches.
- The key condition is said to be Forgotten if it has its DNSKEY state in Dead.
- The key condition is said to be Expired if it has its RRSIG state in Dead (for ZSKs).
- The key condition is said to be ExpiredDS if it has its DS state in Dead (for KSKs).

A key state can now be represented as the triplet (DNSKEY State, RRSIG State, DS State). For example:

S(Kc) = (DNSKEY Propagated, RRSIG Introduced, DS Generated)

tells us that key Kc is published in the zone and all the resolvers that have a copy of the DNSKEY RRset, have one that includes Kc. In other words, Kc is said to be Known. In addition, the key is Active as it is being used for signing RRsets: RRSIG records made with Kc have been introduced in the zone. However, there may still be some resolver caches that are unaware of these signatures. Finally, the corresponding DS record is said to be Generated and has thus not yet been submitted to the parent.

For convenience, we can represent a ZSK as a tuple (DNSKEY State, RRSIG State), because the DS record is only used with KSKs. And we can represent a KSK as a tuple (DNSKEY State, DS State), because the RRSIG state only refers to ZSKs. The RRSIG record over the DNSKEY RRset should be published at the same time when the corresponding DNSKEY record is published. Therefore, both records will propagate and expire at the same time from resolver caches.

2.3. Delay Timings

For every change we make in the zone, we have to take into account several delays.

Software Delay (Dsfw): The time it takes for the software to introduce the new information in the zone. This delay can vary alot depending on the information that needs to be introduced. One can imagine that the software needs more time to sign a complete zone than when it pre-publishes a DNSKEY record. [MM: Dsfw maps to Dsgn from the key-timing draft]

Propagation Delay (Dprp): The time it takes for any change introduced at the master to replicate to all slave servers.

TTL Delay (Dttl): The time it takes to expire the previous information from the resolver caches. This delay depends on what RRsets need to expire from the caches. If not explicitly mentioned otherwise, Dttl is considered the maximum TTL of the information that needs to expire from caches. Otherwise, Dttl(RRtype) shows which specific RRsets need to expire. [MM: TTL terminology in key-timing draft: TTLds, TTLkey, TTLkeyC, TTLsoa, TTLsoaC, TTLsoaP, TTLsig)]

Registration Delay to the Parent (Dreg): The time it takes to get the DS record to be placed into the parent zone, after it is submitted.

Propagation Delay of the Parent (DprpP): The time it takes for any change introduced at the parent master to replicate to all parent slave servers.

Despite these delays may vary for the different rollover methods, we can identify the propagation delay to the caches as:

```
DcacheZ = Dsfw + Dprp + Dttl
DcacheK = Dsfw + Dprp + Dttl(DNSKEY)
DcacheP = Dreg + DprpP + Dttl(DS)
```

where DcacheZ is the propagation delay to the caches for information published in our zone, DcacheK is the propagation delay to the caches for our DNSKEY RRset and DcacheP is the propagation delay for information published in our parent zone.

3. Key Rollovers

There are many different key rollover methods. In Section 1.1, we have seen that there are several properties to prefer one method over the other. Though there are many different type of key rollovers, all methods share the same goal. There is a current key (Kc) that needs to become Forgotten-Retired and a successor key (Ks) that needs to become Known-Safe.

3.1. ZSK Rollovers

The two most common rollover methods for ZSKs are Double-Signature and Pre-Publication. Both are described in RFC4641 [RFC4641]. draft-ietf-dnsop-dnssec-key-timing [key-timing] also introduces ZSK Double-RRSIG rollover. Double-Signature is the fastest way to rollover a ZSK. Pre-Publication minimizes the number of signatures over the RRsets in the zone and responses. Double-RRSIG keeps the size of the DNSKEY RRset to a minimum.

3.1.1. Double-Signature

This involves introducing the new key into the zone and using it to create additional RRSIG records; the old key and existing RRSIG records are retained. During the period in which the zone is being signed, client resolvers are always able to validate RRSIGs: any combination of old and new DNSKEY RRset and RRSIG allows at least one signature to be validated.

Once the signing process is complete and enough time has elapsed to allow all old information to expire from caches, the old key and signatures can be removed from the zone. As before, during this period any combination of DNSKEY RRset and RRSIG will allow validation of at least one signature.

Double-Signature is the fastest way to rollover to a new key, since all new information is published right away. The drawback of this method is a noticeable increase in the size of the DNSSEC data, affecting both the overall size of the zone and the size of the responses.

Only when Ks is said to be Known, e.g. the DNSKEY record of Ks is known to all validating resolvers, we can remove the signatures made with Kc. And only when we can ensure that all validators only use the information of Ks for authentication, we can remove the DNSKEY record for Kc. In other words, Ks needs to be Known and Safe, before we can remove Kc. Thus, we first have to introduce all new information into the zone. Once all has been propagated, we can withdraw all information of Kc from the zone.

The timeline diagram is shown below:

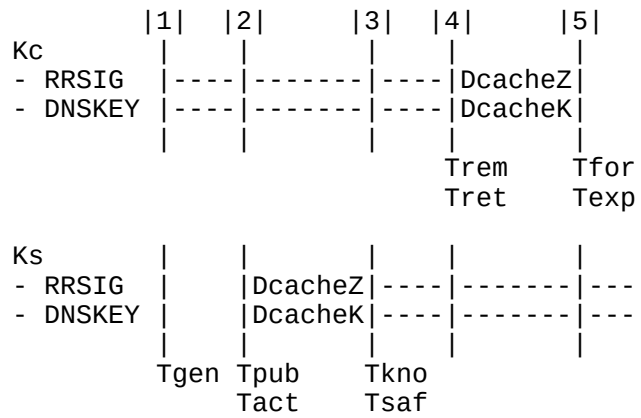

Figure: ZSK Double-Signature Rollover.

Event 1: Key Ks is generated at the generate time (Tgen).

S(Ks) = (DNSKEY Generated, RRSIG Generated)
C(Ks) = Generated

Event 2: Key Ks is added to the DNSKEY RRset and is immediately used to sign the zone; existing signatures in the zone are not removed. This is Ks' publish time (Tpub) and Ks is said to be Published. It is also Ks' active time (Tact), the time when Ks is said to be Active. Because the Double-Signature rollover is in place, we now temporarily have two active keys.

Tpub(Ks) >= Tgen(Ks), Tact(Ks) == Tpub(Ks)

S(Ks) = (DNSKEY Introduced, RRSIG Introduced)
C(ks) = Published Active

Event 3: The information for Ks must be published long enough to ensure that the information have reached all validating resolvers that may have RRsets from this zone cached. At the point in time that the DNSKEY RRset including Ks has been propagated and Ks is said to be Known (Tkno). At the point in time that the other RRsets including a signature of Ks have been propagated (Tsaf), Ks is said to be Safe.

Tkno(Ks) >= Tpub(Ks) + DcacheK
Tsaf(Ks) >= Tact(Ks) + DcacheZ

S(Ks) = (DNSKEY Propagated, RRSIG Propagated)

C(ks) = Known Safe

Note that we could already retire Kc, i.e. stop signing with Kc, after DcacheK. It does not matter if not all signatures of Ks have been Propagated, since the resolver can validate RRsets with both Kc and Ks. If the validator fetches a RRset from the cache, it uses the DNSKEY of Kc for validation. Otherwise, it can use the DNSKEY of Ks.

Event 4: Once we have a successor key that is said to be Propagated, we can retire Kc. This is Kc' retire time (Tret) and Kc is said to be Retired. And once we have a successor key that is said to be Safe, we can remove Kc. Therefore, it is also Kc' removal time (Trem), the time that Kc is said to be Removed.

Tret(Kc) >= Tkno(Ks)
Trem(Kc) >= MAX(Tsaf(Ks), TsafDS(Ks))

S(Kc) = (DNSKEY Withdrawn, RRSIG Withdrawn)
C(kc) = Removed Retired

Event 5: From the perspective of the authoritative server, the rollover is complete. After some delay, Kc and its signatures have expired from the caches. This delay is the maximum of DcacheZ, DcacheK. This is Tfor, the time that the key is said to be Forgotten and Texp, the time that the key is said to be Expired.

Tfor(Kc) >= Trem(Kc) + DcacheK
Texp(Kc) >= Tret(Kc) + DcacheZ

S(Kc) = (DNSKEY Dead, RRSIG Dead)
C(Kc) = Forgotten Expired

3.1.2. Pre-Publication

With Pre-Publication, the new key is introduced into the DNSKEY RRset, leaving the existing keys and signatures in place. This state of affairs remains in place for long enough to ensure that any DNSKEY RRsets cached in client validating resolvers contain both keys. At that point signatures created with the old key can be replaced by those created with the new key, and the old signatures can be removed. During the re-signing process it doesn't matter which key an RRSIG record retrieved by a client was created with; clients with a cached copy of the DNSKEY RRset will have a copy containing both the old and new keys.

Once the zone contains only signatures created with the new key, there is an interval during which RRSIG records created with the old key expire from client caches. After this, there will be no

signatures anywhere that were created using the old key, and it can can be removed from the DNSKEY RRset.

Pre-Publication is more complex than Double-Signature - introduce the new key, approximately one TTL later sign the records, and approximately one TTL after that remove the old key. It also takes more time than the Double-Signature method. The delay is because we don't want to publish signatures of both keys at the same time. As an advantage, the amount of DNSSEC data is kept to a minimum which reduces the impact on performance.

The timeline diagram looks like this:

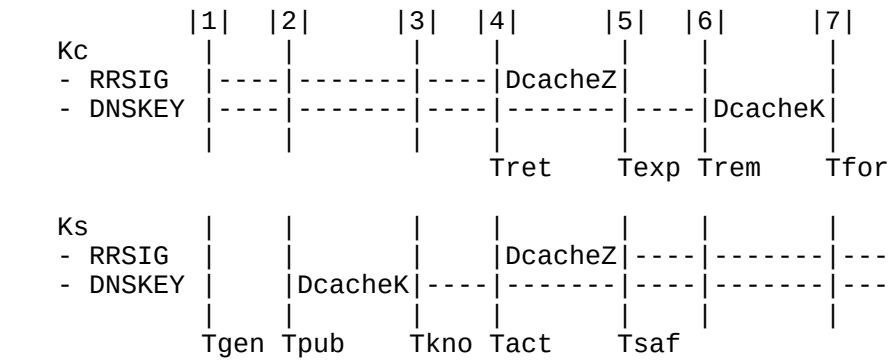

Figure: ZSK Pre-Publication Rollover.

Event 1: Key Ks is generated at the generate time (Tgen).

S(Ks) = (DNSKEY Generated, RRSIG Generated)
C(Ks) = Generated

Event 2: The DNSKEY record of Ks is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the current KSK. The time at which this occurs is Ks' publication time (Tpub), and the key is now said to be Published. Note that the key is not yet used to sign records.

Tpub(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Introduced, RRSIG Generated)
C(Ks) = Published

Event 3: Before Ks can be used, the DNSKEY record for Ks must be published for long enough (DcacheK) to guarantee that any resolver that has a copy of the DNSKEY RRset also includes this key. In other words, that any prior cached information about the DNSKEY RRset has

expired. After this delay, the key is said to be Known and could be used to sign records. The time at which this event occurs is Tkno, which is given by:

Tkno(Ks) >= Tpub(Ks) + DcacheK

S(Ks) = (DNSKEY Propagated, RRSIG Generated)
C(Ks) = Known

Event 4: At some point in time, the decision is made to actually start signing the zone using the successor key. This will be when the current key has been in use for an interval equal to the ZSK lifetime. This point in time is Ks' active time (Tact), the time that Ks is said to be Active. It is also Kc' retire time (Tret), the time that Kc is said to be Retired.

Tact(Ks) >= Tkno(Ks), Tret(Kc) == Tact(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Withdrawn)
C(Kc) = Known Retired
S(Ks) = (DNSKEY Propagated, RRSIG Introduced)
C(Ks) = Known Active

Event 5: Kc needs to be retained in the zone whilst any RRSIG records created by the retired key are still published in the zone or held in resolver caches. In other words, Kc should be retained in the zone until all RRSIG records created by Ks have been propagated. This time is Ks' safe time (Tsaf), the time that Ks is considered to be Safe. Consequently, at the same time Kc is considered to be Expired.

Tsaf(Ks) >= Tact(Ks) + DcacheZ

S(Kc) = (DNSKEY Propagated, RRSIG Dead)
C(Kc) = Known Expired
S(Ks) = (DNSKEY Propagated, RRSIG Propagated)
C(Ks) = Known Safe

Event 6: When all new signatures have been propagated, Kc can be removed from the zone and the DNSKEY RRset re-signed with the current KSK. This time is Kc' removal time (Trem), the time that Kc is considered to be Removed.

Trem(Kc) >= Tsaf(Ks)

S(Kc) = (DNSKEY Withdrawn, RRSIG Dead)
C(Kc) = Removed Expired

Event 7: From the perspective of the authoritative server, the

rollover is complete. After some delay, The DNSKEY record for Kc has expired from the caches. This is Tfor, and the key is said to be Forgotten.

Tfor(Kc) >= Trem(Kc) + DcacheK

S(Kc) = (DNSKEY Dead, RRSIG Dead)
C(Kc) = Forgotten Expired

3.1.3. Double-RRSIG

This involves introducing the new signatures first, while existing signatures are being retained. This state of affairs remains in place for long enough to ensure that all RRsets cached in client validating resolvers contain two signatures. The DNSKEY RR can now be switched. For the period of time before the predecessor key has been expired from all caches, it does not matter if the validator uses the cached key or the successor key that is in the zone. Both corresponding signatures can be retrieved from the cache or from the name server.

Once the signing process is complete and enough time has elapsed to allow all old information to expire from caches, the old signatures can be removed from the zone. As before, during this period any combination of DNSKEY RRset and RRSIG will allow validation of at least one signature.

Double-RRSIG is also more complex than Double-Signature - first introducing the signatures, then switch the key and finally remove the olds signatures. It also takes more time than the Double-Signature method. The delay is because we cannot publish the public data of both keys at the same time. As an advantage, the DNSKEY RRset is kept to a minimum which reduces the impact on priming performance.

The timeline diagram is shown below:

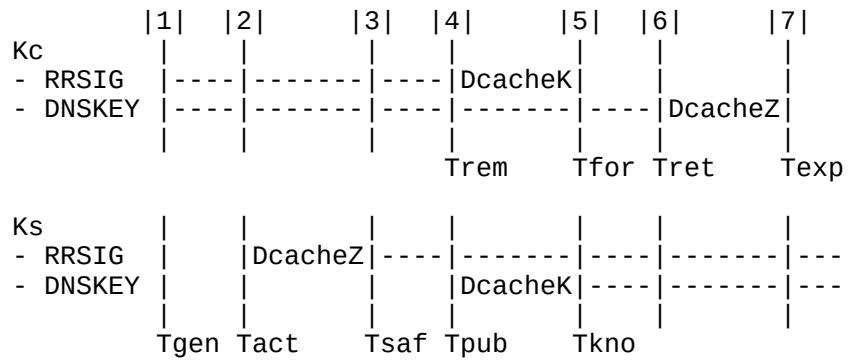

Figure: ZSK Double-RRSIG Rollover.

Event 1: Key Ks is generated at the generate time (Tgen).

S(Ks) = (DNSKEY Generated, RRSIG Generated)
C(Ks) = Generated

Event 2: The zone is signed with Ks but existing signatures are retained. The DNSKEY RR for Ks remains unpublished. The time at which this occurs is Ks' active time (Tact), and the key is now said to be Active.

Tact(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Generated, RRSIG Introduced)
C(Ks) = Active

Event 3: Before we can switch the DNSKEY from Kc to Ks, the signatures of Ks must be published for long enough (DcacheZ) to guarantee that any resolver that has a copy of any RRset, also has both signatures. In other words, that any cached information is double signed. After this delay, the key is said to be Safe. The time at which this event occurs is Tsaf, which is given by:

Tsaf(Ks) >= Tact(Ks) + DcacheZ

S(Ks) = (DNSKEY Generated, RRSIG Propagated)
C(Ks) = Safe

Event 4: At some point in time, the decision is made to publish Ks. This point in time is Ks' publish time (Tpub), the time that Ks is said to be Published. At the same time, the DNSKEY RR for Kc is removed from the zone, and Kc is said to be Removed.

Tpub(Ks) >= Tsaf(Ks), Trem(Kc) == Tpub(Ks)

S(Kc) = (DNSKEY Removed, RRSIG Propagated)
C(Kc) = Removed Safe
S(Ks) = (DNSKEY Introduced, RRSIG Propagated)
C(Ks) = Published Safe

Event 5: The signatures of Kc need to be retained in the zone until the DNSKEY RR has expired from all resolver caches. When this happens, Ks is said to be Known (Tkno) and Kc is said to be Forgotten (Tfor).

Tfor(Kc) >= Trem(Kc) + DcacheK
Tkno(Ks) >= Tpub(Ks) + DcacheK

S(Kc) = (DNSKEY Dead, RRSIG Propagated)
C(Kc) = Forgotten Safe
S(Ks) = (DNSKEY Propagated, RRSIG Propagated)
C(Ks) = Known Safe

Event 6: The signatures of Kc can be removed when the DNSKEY RR for Ks has been propagated. This time is Kc' retire time (Tret), the time that Kc is considered to be Retired.

Tret(Kc) >= Tsaf(Ks)

S(Kc) = (DNSKEY Dead, RRSIG Withdrawn)
C(Kc) = Forgotten Retired

Event 7: From the perspective of the authoritative server, the rollover is complete. After some delay, all signatures of Kc have expired from the caches. This is Texp, and the key is said to be Expired.

Texp(Kc) >= Tret(Kc) + DcacheZ

S(Kc) = (DNSKEY Dead, RRSIG Dead)
C(Kc) = Forgotten Expired

3.2. KSK Rollovers

The most common rollover method for KSKs is Double-Signature, described in RFC4641 [RFC4641]. Two more methods are identified in draft-ietf-dnsop-dnssec-key-timing [key-timing]: Double-DS and Double-RRset. Double-RRset is the fastest way to rollover a KSK, while Double-Signature minimizes the number of required interactions to the parent, and Double-DS keeps your DNSKEY RRset as small as possible.

Note that with these type of rollovers, we do not have to worry whether the information within the zone is authentic. We assume that there exists one or more ZSKs in the DNSKEY RRset that takes care of this during the rollover.

3.2.1. Double-RRset

With Double-RRset, the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key, and the new DS record added to the parent zone. After waiting a suitable interval for the old DS and DNSKEY RRsets to expire from validating resolver caches, the old DNSKEY and DS record are removed.

Only when Ks is said to be Known, e.g. the DNSKEY record of Ks is known to all validating resolvers, we can remove the DS record of Kc. And only when can ensure that all validators can use the DS record for Ks to build the secure chain of trust, we can remove the DNSKEY record of Kc. In other words, Ks needs to be Known and SafeDS. Thus, we first have to introduce all new information into the zone. Once all has been propagated, we can withdraw all information of Kc from the zone.

The timeline diagram looks like this:

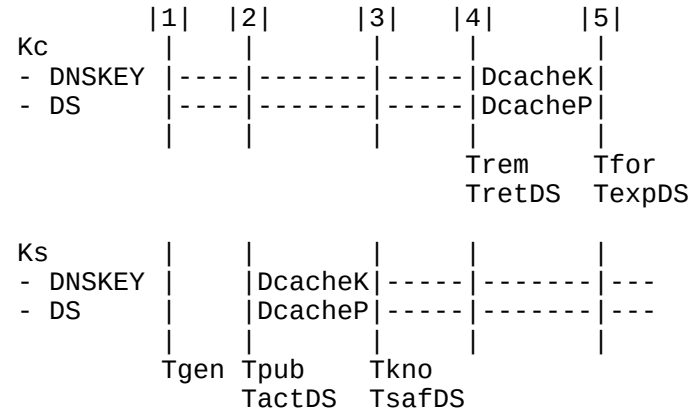

Figure: KSK Double-RRset Rollover.

Event 1: Ks is generated at time Tgen.

S(Ks) = (DNSKEY Generated, DS Generated)
C(Ks) = Generated

Event 2: Ks is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by all currently active KSKs (including

Kc and Ks). In addition, the DS record is submitted to the parent. This is Ks' publish time (Tpub), the time that Ks is said to be Published. It is also Ks' submit time (TactDS), the time that the DS record for Ks is Submitted (ActiveDS).

Tpub(ks) >= Tgen(Ks), TactDS(Ks) == Tpub(Ks)

S(Ks) = (DNSKEY Introduced, DS Introduced)
C(ks) = Published ActiveDS

After the registration delay, the DS is published in the parent.

Event 3: The information for Ks must be published long enough to ensure that the information have reached all validating resolvers that may have the DNSKEY or DS RRset from this zone cached. At the point in time that the DNSKEY RRset including Ks has been propagated (Tkno), Ks is said to be Known. At the point in time that the DS RRset of Ks has been propagated (Tsaf), Ks is said to be SafeDS.

Tkno(Ks) >= Tpub(Ks) + DcacheK, TsafDS(Ks) >= TactDS(Ks) + DcacheP

S(Ks) = (DNSKEY Propagated, DS Propagated)
C(Ks) = Known SafeDS

Note that we could already send the request to the parent to withdraw the DS record of Kc after DcacheK. It does not matter if the DS record for Ks has not yet been propagated, since the resolver can authenticate the DNSKEY RRset with both Kc and Ks. If the validator fetches a DS RRset from the cache, it uses Kc. Otherwise, it can use Ks.

Event 4: Once we have a successor key that is said to be Known, we can withdraw the DS record for Kc. This is Kc' retire time (Tret), the time that Kc is said to be RetiredDS. If Ks is also said to be SafeDS, we no longer need to retain Kc in the zone. It is also Kc' removal time (Trem), the time that Kc is said to be Removed.

TretDS(Kc) >= Tkno(ks)
Trem(Kc) >= MAX(TsafDS(Ks), Tkno(Ks))

S(Kc) = (DNSKEY Withdrawn, DS Withdrawn)
C(Kc) = Removed RetiredDS

Event 5: From the perspective of the authoritative server, the rollover is complete. After some delay, Kc and its DS have also expired from the caches.

Tfor(Kc) >= Trem(Kc) + DcachK

TexpDS(Kc) >= TretDS(Kc) + DcacheP

S(Kc) = (DNSKEY Dead, DS Dead)
C(Kc) = Forgotten Expired

3.2.2. Double-Signature

With Double-Signature, the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key. After waiting for the old RRset to expire from caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset.

If you want to minimize the number of interactions to the parent, this rollover method is preferred over the Double-RRset method. As a consequence, you have to wait with submitting the DS record of Ks, until it is safe to withdraw the DS record of Kc.

The timing diagram for such a rollover is:

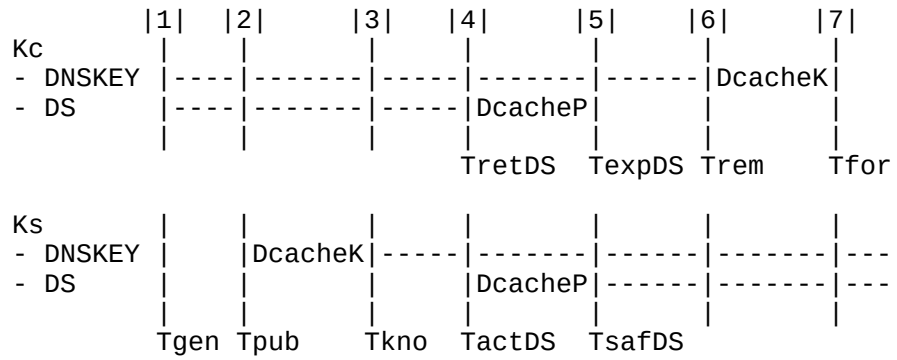

Figure: KSK Double-Signature Rollover.

Event 1: Ks is generated at time Tgen.

S(Ks) = (DNSKEY Generated, DS Generated)
C(Ks) = Generated

Event 2: Ks is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by Ks and all currently actice KSKs (including Kc). This is the publication time (Tpub), the time that Ks is said to be Published.

Tpub(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Introduced, DS Generated)

C(Ks) = Published

Event 3: Before we can submit the corresponding DS, the DNSKEY record for Ks must be published for long enough (DcacheK) to guarantee that any resolver that has a copy of the DNSKEY RRset also includes this key. In other words, that any prior cached information about the DNSKEY RRset has expired. This time is Tkno and Ks is said to be Known.

Tkno(Ks) >= Tpub(Ks) + DcacheK

S(Ks) = (DNSKEY Propagated, DS Generated)
C(Ks) = Known

Event 4: At some later time, the DS RR corresponding to Ks is submitted to the parent zone for publication. In addition, the request has been made to remove the DS RR corresponding to Kc from the parent zone. This time is Ks' submit time (TactDS), the time that Ks is considered to be Submitted. It is also Kc' retire time (TretDS), the time that Kc is considered to be RetiredDS.

TactDS(Ks) >= Tkno(Ks)
TretDS(kc) == TactDS(Kc)

S(Kc) = (DNSKEY Propagated, DS Withdrawn)
C(Ks) = Known RetiredDS
S(Ks) = (DNSKEY Propagated, DS Introduced)
C(Ks) = Known ActiveDS

After the registration delay, the DS is published in the parent.

Event 5: At some time later, all validating resolvers that have the DS RRset cached will have a a copy that includes the new DS record. This is Ks' safe time (TsafDS), the time that the new KSK is said to be SafeDS. Consequently, Kc is said to be ExpiredDS (TexpDS).

TsafDS(Ks) >= TactDS(Ks) + DcacheP
TexpDS(Kc) >= TretDS(Kc) + DcacheP

S(Kc) = (DNSKEY Propagated, DS Dead)
C(kc) = Known ExpiredDS
S(Ks) = (DNSKEY Propagated, DS Propagated)
C(Ks) = Known SafeDS

Event 6: When the new DS record has been propagated, the DNSKEY record of Kc can be removed from the zone. This is Kc' removal time (Trem), the time that Kc is said to be Removed.

Trem(Kc) >= TsafDS(Ks)

S(Kc) = (DNSKEY Withdrawn, DS Dead)
C(Kc) = Removed ExpiredDS

Event 7: From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record for Kc has also expired from the caches.

Tfor(Kc) >= Trem(Kc) + DcacheK

S(Kc) = (DNSKEY Dead, DS Dead)
C(Kc) = Forgotten ExpiredDS

3.2.3. Double-DS

In this case, first the new DS record is published. After waiting for this change to propagate into the caches of all validating resolvers, the KSK is changed. After waiting another interval, during which the old DNSKEY RRset expires from caches, the old DS record is removed.

If you want to keep the size of the DNSKEY RRset to a minimum, this rollover method is preferred over Double-RRset. It does require the additional administrative overhead of two interactions with the parent to roll a KSK.

The timeline diagram looks like this:

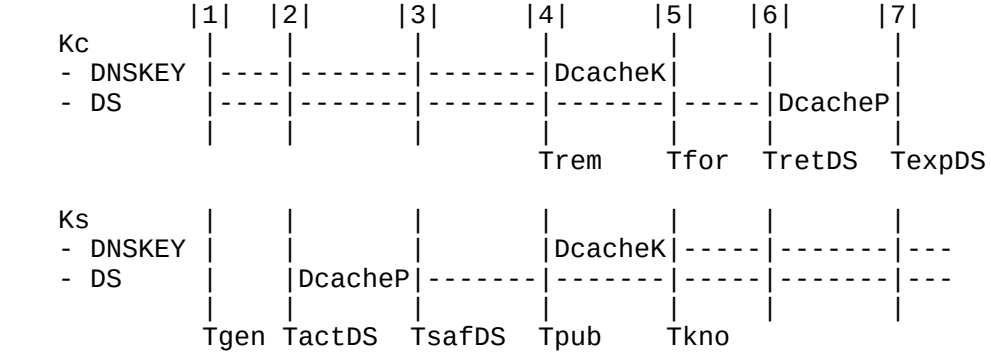

Figure: KSK Double-DS Rollover.

Event 1: Ks is generated at time Tgen.

S(Ks) = (DNSKEY Generated, DS Generated)
C(Ks) = Generated

Event 2: Before we introduce the new key Ks into the zone, we are going to submit the new DS. We can do that, because there exists a valid chain of trust for the same algorithm (Kc). This time is Ks' submit time (TactDS), the time that the DS record for Ks was submitted and is said to be ActiveDS.

TactDS(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Generated, DS Introduced)
C(Kc) = ActiveDS

After some delay, the DS becomes available in the parent zone.

Event 3: Some time later, the new DS RRset has been propagated. This is Ks' safe time (TsafDS), the time that Ks is said to be SafeDS.

TsafDS(Ks) >= TactDS(Ks) + DcacheP

S(Ks) = (DNSKEY Generated, DS Propagated)
C(Ks) = SafeDS

Event 4: Because there are now two trust anchors a resolver can use, we can switch the KSK in the DNSKEY RRset. We stop signing with Kc and sign the DNSKEY RRset with Ks. This time is Ks' publish time (Tpub), the time that Ks is said to be Published. It is also Kc' removal time (Trem), the time that Kc is said to be Removed.

Tpub(Ks) >= TsafDS(Ks)
Trem(Kc) == Tpub(Ks)

S(Kc) = (DNSKEY Withdrawn, DS Propagated)
C(Kc) = Removed SafeDS
S(Ks) = (DNSKEY Introduced, DS Propagated)
C(Ks) = Published SafeDS

Event 5: We have to wait before Kc has been expired from the caches, before we can withdraw the DS record of Kc. When the DNSKEY RRset that includes Kc has been expired, Kc is said to be forgotten and Ks is said to be Known. This happens at Ks' known time, given by:

Tkno(Ks) >= Tpub(Ks) + DcacheK
Tfor(Kc) == Tkno(Ks)

S(Kc) = (DNSKEY Dead, DS Propagated)
C(Kc) = Forgotten SafeDS
S(Ks) = (DNSKEY Propagated, DS Propagated)
C(Ks) = Known SafeDS

Event 6: Now that we have a key Ks that is said to be Propagated and SafeDS, we are ready to withdraw the DS for Kc. We call this Kc' retire time (TretDS), the time that we don't need a secure delegation for Kc anymore.

TretDS(Kc) >= Tkno(Ks)

S(Kc) = (DNSKEY Dead, DS Withdrawn)
C(Kc) = Forgotten RetiredDS

Event 7: From the perspective of the authoritative server, the rollover is complete. After some delay, The DS record for Kc has expired from the caches. This is Texp, given by:

Texp(Kc) >= Tret(Kc) + DcacheP

S(Kc) = (DNSKEY Dead, DS Dead)
C(Kc) = Forgotten ExpiredDS

3.2.4. Interaction with Configured Trust Anchors

Zone managers may want to take into account the possibility that some validating resolvers may have their KSK configured as a trust anchor directly, as described in [RFC5011]. This influences the value of DcacheK, the time to guarantee that any resolver that has a copy of the newest DNSKEY RRset.

3.2.4.1. Adding a KSK

When the new key is introduced, the delay DcacheK between Tpub and Tkno is also subject to the condition:

DcacheK' = MAX(DcacheK, 2 * (queryInterval + x * retryTime) + c)

The right hand side of this expression is two times the Active Refresh time defined in section 2.3 in [RFC5011]. This ensures that the successor key is at least seen twice by 5011-enabled validators. The parameter x is the maximum number of retries that is taken as a safety margin, in case an Active Refresh fails. The parameter c is a constant that can be taken as an additional safety margin.

Most probably, this delays the time when a key is said to be Known.

3.2.4.2. Removing a KSK

When the current key is ready to be removed from the zone, it is instead said to be Revoked. The REVOKE bit is said and the key is published for DcacheK' time:

DcacheK' = MAX(DcacheK, (queryInterval + x * retryTime) + c)

The right hand side of this expression is the Active Refresh time defined in section 2.3 in RFC5011 [RFC5011]. This ensures that the revoked key is at least seen once by 5011-enabled validators.

After that delay, we can guarantee that every 5011-enabled resolver has seen the revoked key and it may be removed from the zone. Another DcacheK delay, the key has fully expired from all the resolver caches.

3.3. Rollovers in a Single Type Signing Scheme

In situations where you use a Single Type Signing Scheme, you can combine one of the ZSK rollover methods with one of the KSK rollover methods. However, not all combinations are possible. The KSK Double-DS rollover is only suitable for combining with the ZSK Double-RRSIG rollover, because both keep the DNSKEY RRset to a minimum size. The other ZSK rollovers require a period where both the current key and its successor are being served at the same time.

The KSK Double-RRset method is suitable with both the other ZSK rollover methods, but does not gain any advantages when combined with the ZSK Pre-Publication method. Therefore, we can leave that combination out. The KSK Double-Signature method is suitable with both the ZSK Double-Signature and the ZSK Pre-Publication method.

To conclude, we can identify four different rollover methods for the Single Type Signing Scheme.

3.3.1. Double-RRset

This is a combination of the ZSK Double-Signature rollover and the KSK Double-RRset rollover. The new KSK is added to the DNSKEY RRset, and all RRsets are then signed with both the old and new key, and the new DS record added to the parent zone. After waiting a suitable interval for the old DS and all zone RRsets to expire from validating resolver caches, the old DNSKEY and DS record are removed.

Double-RRset is the fastest way to replace keys in a Single Type Signing Scheme. However, it does have a lot of disadvantages of - it requires two signatures and two keys during the period of the rollover, as well as two interactions with the parent.

The timeline diagram looks like this:

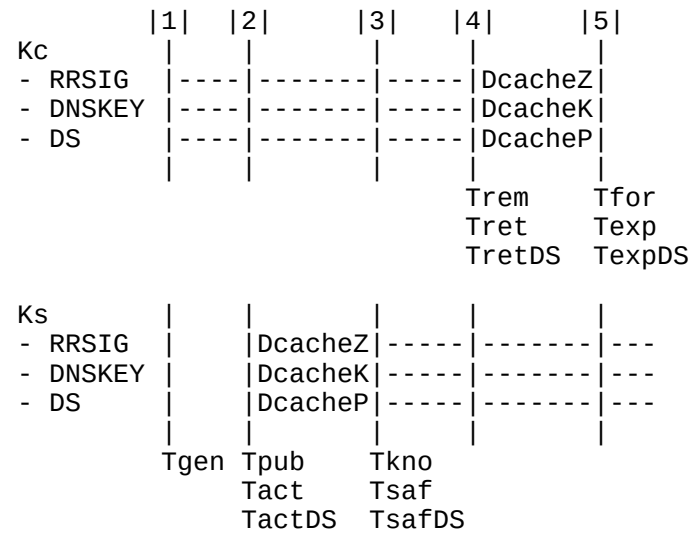

Figure: STSS Double-RRset Rollover.

The rollover method is almost the same as that of the KSK Double-RRset rollover, except we now have to take DcacheZ into account.

3.3.2. Double-Signature

This is a combination of the ZSK Double-Signature rollover and the KSK Double-Signature rollover. The new key is added to the DNSKEY RRset and all RRsets are then signed with both the old and new key. After waiting for the old RRsets to expire from caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the DNSKEY RRset, and all RRsets are signed with the new key only.

This rollover minimizes the number of interactions with the parent zone. However, for the period of the rollover all RRsets are still signed with two keys, so increasing the size of the zone and the size of the response.

The timing diagram for such a rollover is:

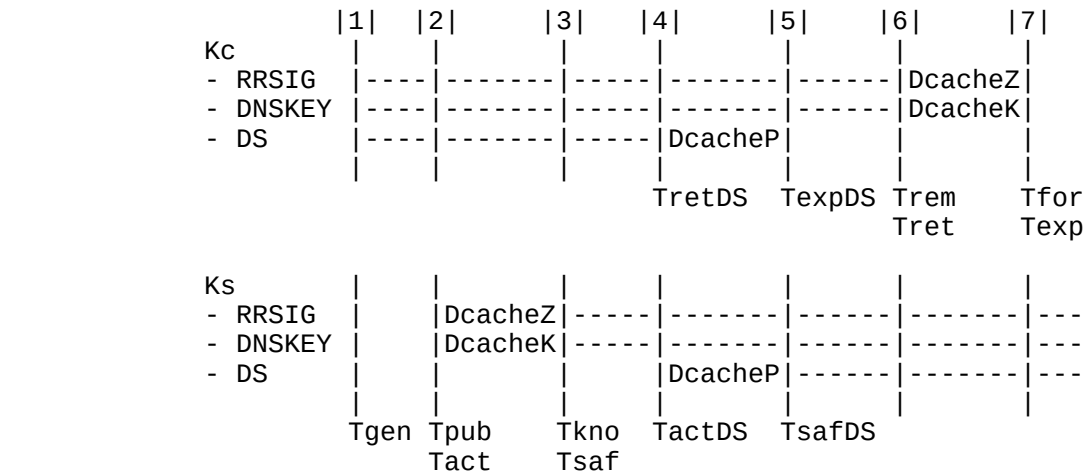

Figure: STSS Double-Signature Rollover.

The rollover diagram is almost the same as that of the KSK Double-Signature rollover, except we now have to take DcacheZ into account.

3.3.3. Pre-Publication

This is a combination of the ZSK Pre-Publication rollover and the KSK Double-Signature rollover and requires only one interaction with the parent. In addition, your non-DNSKEY RRsets require only one signature during the rollover. If speed is not an issue, this rollover method is considered to be the best practice in a Single Type Signing Scheme environment.

The new key is added to the DNSKEY RRset and the DNSKEY RRset is then signed with both the old and new key. Other RRsets will only be signed with the old key. Only after the DS has been switched, the signatures of other RRsets are replaced with that of the new key. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset, and is signed with the new key only.

The timeline diagram looks like this:

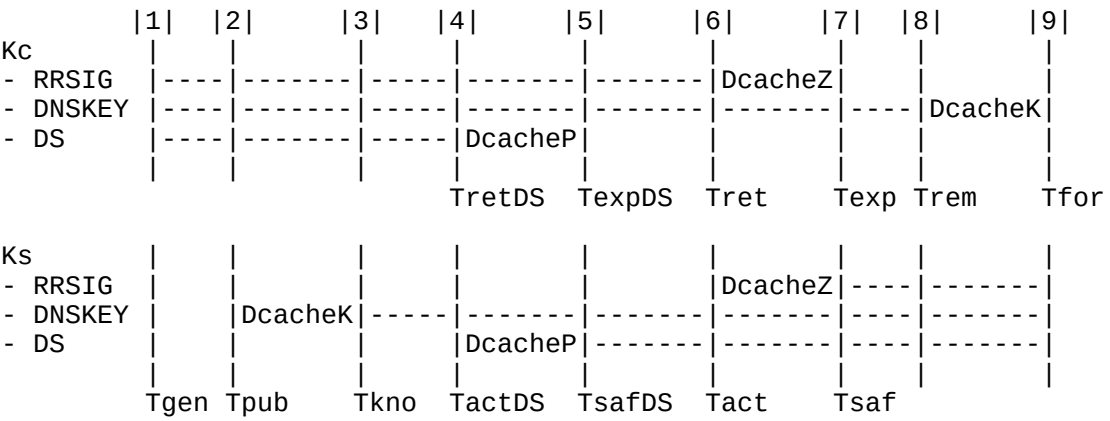

Figure: STSS Pre-Publication Rollover.

Event 1: Key Ks is generated at the generate time (Tgen).

S(Ks) = (DNSKEY Generated, RRSIG Generated, DS Generated)
C(Ks) = Generated

Event 2: The DNSKEY record of Ks is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the Ks and all other current KSKs (including Kc). The time at which this occurs is Ks' publication time (Tpub), and the key is now said to be Published. Note that the key is not yet used to sign other RRsets.

Tpub(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Introduced, RRSIG Generated, DS Generated)
C(Ks) = Published

Event 3: Before we can switch the DS, the DNSKEY record for Ks must be published for long enough (DcacheK) to guarantee that any resolver that has a copy of the DNSKEY RRset also includes this key. After this delay, the key is said to be Known and the DS record may be submitted. The time at which this event occurs is Ks' known time (Tkno), which is given by:

Tkno(Ks) >= Tpub(Ks) + DcacheK

S(Ks) = (DNSKEY Propagated, RRSIG Generated, DS Generated)
C(ks) = Known

Event 4: The time that the DS record of Ks is submitted is at Ks'

submit time (TactDS). Ks is said to be ActiveDS. At the same time, the DS record of Kc is withdrawn (TretDS) and Kc is said to be RetiredDS.

TactDS(Ks) >= Tkno(Ks) TretDS(Kc) == TactDS(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Propagated, DS Withdrawn)
C(Kc) = Known Safe RetiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Generated, DS Introduced)
C(Ks) = Known ActiveDS

Some time later, the new DS RRset is published at the parent.

Event 5: Some time later, we can guarantee that all validating resolvers use the DS RRset that includes a copy of the DS record of DS. At this time, Ks' safe time (TsafDS), Ks is said to be SafeDS. But we still use Kc as ZSK.

TsafDS(Ks) >= TactDS(Ks) + DcacheP
TexpDS(Kc) >= TretDS(Kc) + DcacheP

S(Kc) = (DNSKEY Propagated, RRSIG Propagated, DS Dead)
C(kc) = Known Safe ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Generated, DS Propagated)
C(Ks) = Known SafeDS

Event 6: At some point in time, the decision is made to actually start signing the zone using the successor key. This will be when the current key has been in use for an interval equal to the ZSK lifetime. This point in time is Ks' active time (Tact), the time that Ks is said to be Active. It is also Kc' retire time (Tret), the time that Kc is said to be Retired.

Tact(Ks) >= TsafDS(Ks)
Tret(Kc) == Tact(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Withdrawn, DS Dead))
C(Kc) = Known Retired ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Introduced, DS Propagated))
C(Ks) = Known Active SafeDS

Event 7: Kc needs to be retained in the zone whilst any RRSIG records created by the retired key are still published in the zone or held in resolver caches. In other words, Kc should be retained in the zone until all RRSIG records created by Ks have been propagated. This time is Ks' safe time (Tsaf), the time that Ks is considered to be Safe, and Kc' expiration time (Texp), the time that Kc is considered to be Expired.

Tsaf(Ks) >= Tact(Ks) + DcacheZ
Texp(Kc) == Tsaf(Ks)

S(Kc) = (DNSKEY Propagated, RRSIG Dead, DS Dead)
C(Kc) = Known Expired ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Propagated, DS Propagated)
C(Ks) = Known Safe SafeDS

Event 8: When all new signatures have been propagated, Kc can be removed from the zone and the DNSKEY RRset re-signed with the current KSK. This time is Kc' removal time (Trem), the time that Kc is considered to be Removed.

Trem(Kc) >= Tsaf(Ks)

S(Kc) = (DNSKEY Withdrawn, RRSIG Dead, DS Dead)
C(Kc) = Removed Expired ExpiredDS

Event 9: From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record for Kc has expired from the caches. This is Tfor, the time that the key is said to be Forgotten.

Tfor(Kc) >= Trem(Kc) + DcacheK

S(Kc) = (DNSKEY Dead, RRSIG Dead, DS Dead)
C(Kc) = Forgotten Expired ExpiredDS

3.3.4. Double-DS

This is a combination of the ZSK Double-RRSIG rollover and the KSK Double-DS rollover. This keeps your DNSKEY RRset to a minimum size, but at the cost of double signatures in your zone and double DS at the parent.

The new signatures are added to the zone and the new DS is submitted. Once all signatures and the DS record have been propagated, the DNSKEY is switched. After waiting a further interval for this switch to be reflected in caches, the old signatures are removed and the old DS is withdrawn from the parent zone.

The timeline diagram looks like this:

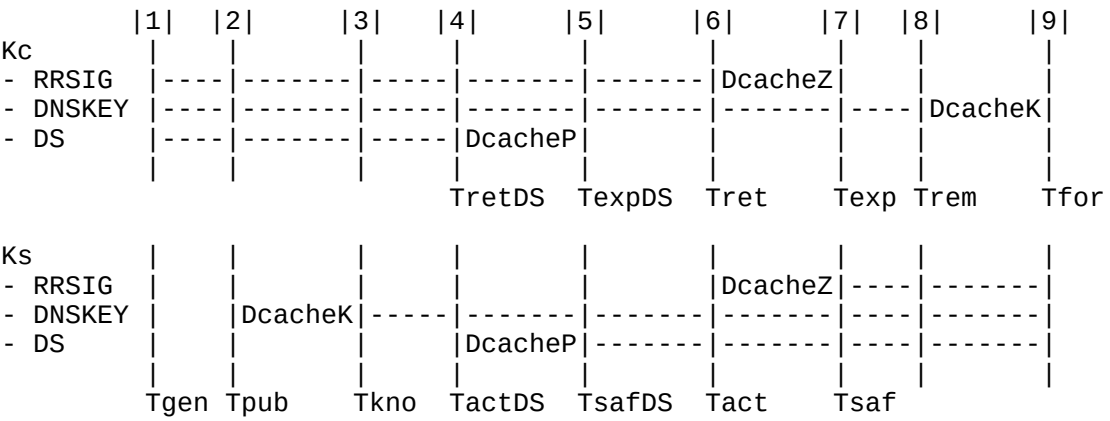

Figure: STSS Double-DS Rollover.

Event 1: Key Ks is generated at the generate time (Tgen).

S(Ks) = (DNSKEY Generated, RRSIG Generated, DS Generated)
C(Ks) = Generated

Event 2: Before we introduce the new key Ks into the zone, we are going to add the new signatures and submit the new DS. This time is Ks' active time (Tact), the time that Ks is said to be Active. It is also Ks' submit time (TactDS), the time that the DS record for Ks was submitted and is said to be ActiveDS.

Tact(Ks) >= Tgen(Ks)
TactDS(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Generated, RRSIG Introduced, DS Introduced)
C(Kc) = Active ActiveDS

After some delay, the DS becomes available in the parent zone.

Event 3: Some time later, the new signatures and the new DS RRset have been propagated. This is Ks' safe time (Tsaf, TsafDS), the time that Ks is said to be Safe and SafeDS.

Tsaf(Ks) >= Tact(Ks) + DcacheZ
TsafDS(Ks) >= TactDS(Ks) + DcacheP

S(Ks) = (DNSKEY Generated, RRSIG Propagated, DS Propagated)
C(Ks) = Safe SafeDS

Event 4: Because there are now two trust anchors a resolver can use, we can switch the KSK in the DNSKEY RRset. This time is Ks' publish time (Tpub), the time that Ks is said to be Published. It is also Kc' removal time (Trem), the time that Kc is removed from the zone.

```
Tpub(Ks) >= MAX(TsafDS(Ks), Tsaf(Ks))
Trem(Kc) == Tpub(Ks)

S(Kc) = (DNSKEY Withdrawn, RRSIG Propagated, DS Propagated)
C(Kc) = Removed Safe SafeDS
S(Ks) = (DNSKEY Introduced, RRSIG Propagated, DS Propagated)
C(Ks) = Published Safe SafeDS
```

Event 5: We have to wait before the signatures of Kc and its corresponding DS record have been expired from the caches, before we can withdraw the DNSKEY record of Kc. When the DNSKEY RRset that includes Kc has been expired, Ks is said to be Known and Kc is said to be Removed. This happens at Ks' known time, given by:

```
Tkno(Ks) >= Tpub(Ks) + DcacheK, Trem(Kc) == Tkno(Ks)

S(Kc) = (DNSKEY Dead, RRSIG Propagated, DS Propagated)
C(Kc) = Forgotten Safe SafeDS
S(Ks) = (DNSKEY Propagated, RRSIG Propagated, DS Propagated)
C(Ks) = Known Safe SafeDS
```

Event 6: Now that we have a key Ks that is said to be Propagated and SafeDS, we are ready to withdraw the signatures and DS for Kc. We call this Kc' retire time (Tret, TretDS), the time Kc is said to be Retired and RetiredDS.

```
Tret(Kc) >= Tkno(Ks)
TretDS(Kc) >= Tkno(Ks)

S(Kc) = (DNSKEY Dead, RRSIG Withdrawn, DS Withdrawn)
C(Kc) = Forgotten Retired RetiredDS
```

Event 7: From the perspective of the authoritative server, the rollover is complete. After some delay, The signatures of Kc and its corresponding DS record have expired from the caches.

```
Texp(Kc) >= Tret(Kc) + DcacheZ
TexpDS(Kc) >= TretDS(Kc) + DcacheP

S(Kc) = (DNSKEY Dead, RRSIG Dead, DS Dead)
C(Kc) = Forgotten Expired ExpiredDS
```

3.4. Stand-by Keys

Although keys will usually be rolled according to some regular schedule, there may be occasions when an emergency rollover is required, e.g. if the active key is suspected of being compromised. The aim of the emergency rollover is to allow the zone to be re-signed with a new key as soon as possible. As a key must be ready to sign the zone, having at least one additional key (a stand-by key) in this state at all times will minimise delay.

In the case of a ZSK, a stand-by key only makes sense with the Pre-Publication method, since with the Double-Signature and Double-RRSIG methods, the stand-by key would be used for signing. The goal is to make the stand-by key Known. This goal is reached at Tkno, step 3 in the Pre-Publication method timeline diagram.

A successor key must always be published soon enough so that the key lifetime of the predecessor key does not exceed. That means that the successor ZSK Ks must at latest be published DcacheK delay before the lifetime of the predecessor ZSK kc has reached:

Tpub(Ks) <= Tact(Kc) + Lzsk - DcacheK

Here, Lzsk is the lifetime of ZSKs according to policy.

In the case of a KSK, a stand-by key only makes sense with the Double-DS method, since in the other cases, the key would be needed to sign the DNSKEY RRset. The goal is to get the stand-by key in the SafeDS condition. This goal is reached at TsafDS, step 3 in the Double-DS method timeline diagram.

The DS record for the successor KSK Ks should be propagated to the caches before the key lifetime of the predecessor KSK Kc exceeds:

TactDS(Ks) <= Tact(Kc) + Lksk - DcacheP

Here, Lksk is the lifetime of KSKs according to policy.

Because a stand-by KSK only makes sense with the Double-DS method, stand-by keys in a STSS is not applicable. This is because the Double-DS method is not easy integratable with one of the ZSK rollover methods.

4. Policy rollover

Besides your scheduled and unscheduled key rollovers, changes in policy may occur. The initial transition is enabling DNSSEC. The counterpart, disabling DNSSEC, is also possible. Two other policy

changes we have encountered are are algorithm rollover and changing signing schemes.

4.1. Enabling DNSSEC

When a zone makes the transition from going insecure to secure, the initial set of keys safely need to be introduced into the zone. The goals of this event is to make a ZSK (Kz) and a KSK (Kk) both Known and Safe.

One must take into account that resolver caches may hold unsigned RRsets. Therefore, validating resolvers should not know about the initial DNSKEY RRset before all unsigned RRsets have been expired from the caches. This means that the zone must be fully signed, before the DS associated with the initial KSK is published. Only if you are afraid that a key scraper fetches your DNSKEY RRset too soon, you should wait with publishing your DNSKEY RRset until enough time has elapsed for all unsigned RRsets to expire from all resolver caches. The ZSK and KSK can be the same key, for example in a Single Type Signing Scheme.

The timeline diagram is shown below:

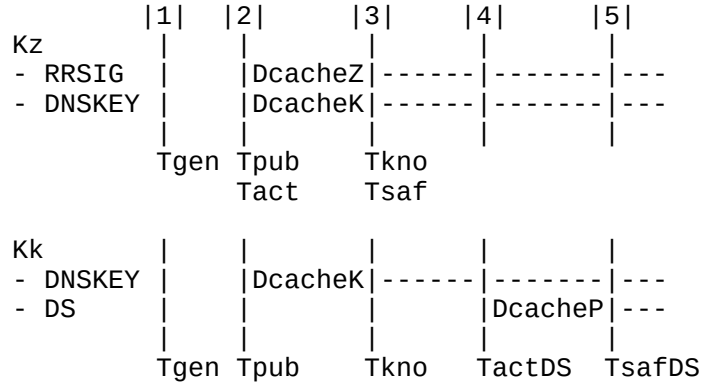

Figure: Enabling DNSSEC.

Event 1: Kk and Kz are generated. We call this Tgen, the time that the keys were Generated (note that Tgen for Kk could be different that Tgen for Kz).

S(Kk) = (DNSKEY Generated, DS Generated)
C(Kk) = Generated
S(Kz) = (DNSKEY Generated, RRSIG Generated)
C(Kk) = Generated

Event 2: The keys are put into the zone and are immediately used for signing. Because there exists no pointer to the fact that our zone is DNSSEC enabled, the DNSKEY and RRSIG records may be introduced at the same time. This is the publish time (Tpub), the time that the keys are Published. It is also Kz' active time (Tact), the time that Kz is said to be Active.

```
Tpub(Kk) >= Tgen(Kk)
Tpub(Kz) >= Tgen(Kz)
Tact(Kz) == Tpub(Kz)

S(Kk) = (DNSKEY Introduced, DS Generated)
C(Kk) = Published
S(Kz) = (DNSKEY Introduced, RRSIG Introduced)
C(Kz) = Published Active
```

Event 3: Before we can submit the DS record, Kz must be considered Known and Safe. Once that has happened, we are done for the ZSK. This time is Kz' known time (Tkno).

```
Tkno(Kk) >= Tpub(Kk) + DcacheP
Tkno(Kk) == Tkno(Kz)
Tsaf(Kz) >= Tact(Kz) + DcacheZ

S(Kk) = (DNSKEY Propagated, DS Generated)
C(Kk) = Known
S(Kz) = (DNSKEY Propagated, RRSIG Propagated)
C(Kz) = Known Safe
```

Because this is the first DNSKEY for this zone, the Dttl for the DNSKEY RRset is Ingc, the negative cache interval from the zone's SOA record, calculated according to RFC2308 [RFC2308] as the minimum of the TTL of the SOA record itself and the MINIMUM field in the record's parameters:

```
Ingc = min(TTL(SOA), MINIMUM)
```

Event 4: Once we are sure of the fact that the DNSKEY RRset and all RRSIG records have reached the caches, we may submit the DS to the parent. We call this TactDS, the time that the DS has been submitted to the parent.

```
TactDS(Kk) >= Tkno(Kk)

S(Kk) = (DNSKEY Propagated, DS Introduced)
C(Kk) = Known ActiveDS
```

Event 5: Some time later, the DS has been published in the parent

zone. Some more time later, all resolvers that have a copy of the DS RRset have one that includes the DS record of Kk.

TsafDS(Kk) >= TactDS(Kk) + DcacheP

S(Kk) = (DNSKEY Propagated, DS Propagated)
C(Kk) = Known SafeDS

Because this is the first DS for this zone, the Dttl for the DS RRset is Ingc, for the same reason as in step 3 for the DNSKEY RRset.

4.2. Disabling DNSSEC

When a zone decides for whatever reason to go back to the Insecure status, the set of keys safely need to be removed from the zone. We assume that there is a KSK (Kk) and a ZSK (Kz) that are Known and Safe. The goals of this event is to make Kk and Kz both Forgotten and Expired.

The timeline diagram is shown below:

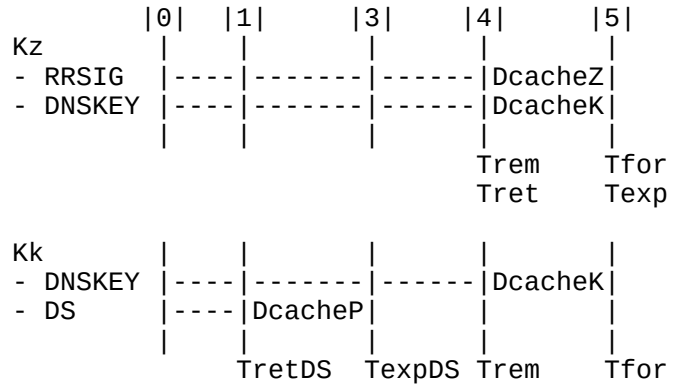

Figure: Disabling DNSSEC.

Event 1: The DS record of Kk needs to be withdrawn. This time is Kk' retire time (TretDS), the time that Kk is said to be RetiredDS.

S(Kk) = (DNSKEY Propagated, DS Withdrawn)
C(Kk) = Known RetiredDS

Event 2: We have to wait until the DS record of Kk has expired from all resolver caches. This time is Kk' expire time (TexpDS), the time that Kk is said to be ExpiredDS.

TexpDS(Kk) >= TretDS(Kk) + DcacheP

S(Kk) = (DNSKEY Propagated, DS Dead)
C(Kk) = Known ExpiredDS

Event 3: Now that we can guarantee that no secure chain of trust to Kk exist anymore, we can retire the ZSK and withdraw both keys. This time is Trem, the time that the keys are removed from the zone.

Trem(Kk) >= TexpDS(Kk)
Trem(Kz) == Trem(Kk)
Tret(Kz) == Trem(kz)

S(Kk) = (DNSKEY Withdrawn, DS Dead)
C(Kk) = Removed ExpiredDS
S(Kz) = (DNSKEY Withdrawn, RRSIG Withdrawn)
C(Kz) = Removed Retired

Event 4: After some delay, all information about the keys have expired from the caches.

Tfor(Kk) >= Trem(Kk) + DcacheK
Tfor(Kz) == Tfor(Kk)
Texp(Kz) >= Tret(Kz) + DcacheZ

S(Kk) = (DNSKEY Dead, DS Dead)
C(Kk) = Forgotten ExpiredDS
S(Kz) = (DNSKEY Dead, RRSIG Dead)
C(Kz) = Forgotten Expired

4.3. Algorithm Rollover

When changing algorithms, you can either add, remove or replace an algorithm. Adding and removing an algorithm follow the same timings as enabling and disabling DNSSEC. Replacing an algorithm can be done with a STSS Double-Signature rollover or a KSK and ZSK Double-Signature Rollover at the same time. [MM: This needs more text, but I am awaiting the discussion about algorithm rollover and how to interpret section 2.2 of RFC 4035]

4.4. KSK-ZSK Split or Single Type Signing Scheme

When changing signing schemes, you should follow the timelines of the most restricting signing scheme. The STSS signing scheme makes some rollover combinations unsuitable, thus it can be considered the most restricted signing scheme. In the case of moving to a KSK-ZSK Split, Ks is used as the successor key in the STSS rollover methods, and it now reflects both the successor ZSK and KSK. In the case of moving

away from a KSK-ZSK Split, Kc is used as the predecessor key in the STSS rollover methods, and it now reflects both the predecessor ZSK and KSK. [MM: This could perhaps also use more explanation.]

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document does not introduce any new security issues beyond those already discussed in RFC4033 [RFC4033], RFC4034 [RFC4034]. RFC4035 [RFC4035] and RFC5011 [RFC5011].

7. Acknowledgements

Special acknowledgments and gratitude go out to Stephen Morris, Johan Ihren and John Dickinson, the authors of the key-timing draft [key-timing]. Significant parts of the text is taken from that document. Especially Section 3.1 and Section 3.2 are largely copied and adjusted to the new introduced terminology from this document.

I also want to acknowledge Yuri Schaeffer, who brought to my attention the idea of key goals (Section 1.1.1) and whose discussions helped to shape this document.

8. Changes with key-timing draft

This document builds further on draft-ietf-dnsop-dnssec-key-timing [key-timing]. The most important changes with respect to that document are:

- Introduced the concept of Rollover Considerations (Speed vs Size vs Interactions), that causes the existence of different key rollover scenarios.

- Introduced the concept of Key Goals.

- Key States are unraveled to represent the status of each piece of information seperately. Provides more flexibility. Used for combining rollover methods in a Single Type Singing Scheme.

- What were Key States in the key-timing draft, are now called Key Conditions. A key can have more than one condition.

- Four new Key Conditions are introduced: Known, Safe, Forgotten and Expired, to represent whether information about the key exist in resolver caches. The key conditions Ready and Dead are deprecated.

- Timelines for STSS Rollovers.

- Timelines for enabling and disabling DNSSEC.

- Text about policy rollover, such as algorithm rollover and changing signing schemes.

9. References

9.1. Informative References

[RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, September 2006.

9.2. Normative References

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC 5011, September 2007.

[dnssec-bis] Weiler, S. and D. Blacka, "Clarifications and Implementation Notes for DNSSECbis", November 2010.

[key-timing] Morris, S., Ihren, J., and J. Dickinson, "DNSSEC Key Timing Considerations", July 2010.

Author's Address

Matthijs Mekking
NLnet Labs
Science Park 140
Amsterdam 1098 XG
The Netherlands

EMail: matthijs@nlnetlabs.nl