Domain Name System Operations
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2012

W. Mekking
NLnet Labs

July 8, 2011

DNSSEC Key Timing Considerations Follow-Up
draft-mekking-dnsop-dnssec-key-timing-bis-02

Abstract

This document describes issues surrounding the timing of events related to DNSSEC policy. It presents timelines for various key rollovers and DNSSEC policy changes regarding the key signing scheme. It explicitly identifies the relationships between the various parameters affecting the rollover process.

This document updates [draft-ietf-dnsop-dnssec-key-timing] [MM: If approved] as it covers timelines for key rollovers in more detail and it covers additional key rollover scenarios, including algorithm rollover and single type key rollovers.

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 January 9, 2012.

Table of Contents

1. Introduction 4
1.1. Key Rollover Considerations 4
1.2. Terminology 5

2. Key Definitions 5
2.1. Key Types 5
2.2. Key States Unraveled 6
2.2.1. Validation Components 6
2.2.1.1. The Public Key Component 6
2.2.1.2. The Signature Component 6
2.2.1.3. The Secure Delegation Component 6
2.2.2. Validation Component States 6
2.3. Key States 8
2.4. Key Goals 10
2.5. Delay Timings 10

3. Key Rollovers 11
3.1. Key Rollover Stages 11
3.2. ZSK Rollovers 13
3.2.1. Double-Signature 13
3.2.2. Pre-Publication 15
3.2.3. Double-RRSIG 18
3.3. KSK Rollovers 21
3.3.1. Double-RRset 21
3.3.2. Double-Signature 24
3.3.3. Double-DS 26
3.3.4. Interaction with Configured Trust Anchors 29
3.3.4.1. Adding a KSK 29
3.3.4.2. Removing a KSK 29
3.4. Rollovers in a Single Type Signing Scheme 29
3.4.1. Double-RRset 30
3.4.2. Double-Signature 31
3.4.3. Pre-Publication 32
3.4.4. Double-DS 35
3.5. Stand-by Keys 38

4. Policy rollover 39
4.1. Enabling DNSSEC 39
4.2. Disabling DNSSEC 41
4.3. Algorithm Rollover 42

4.4. KSK-ZSK Split or Single Type Signing Scheme 43

5. IANA Considerations 43

6. Security Considerations 43

7. Acknowledgements 43

8. Changelog 43
8.1. Changes with key-timing draft 43
8.2. From -00 to -01 44
8.3. From -01 to -02 44

9. References 45
9.1. Informative References 45
9.2. Normative References 45

Appendix A. List of Symbols 45

1. Introduction

DNS was not originally designed with security in mind. The Domain Name System Security Extensions (DNSSEC, [RFC4033], [RFC4034], [RFC4035]) add a security layer that provides data origin authentication and data integrity. A DNS zone that implements DNSSEC must have the ability to replace ("roll") keys. This will be needed for various reasons such as recovery from a key compromise, replacement of key-signing hardware (if used), or even just to implement a policy that requires periodic rollovers.

In addition, a DNS zone may be subject to a given DNSSEC Policy [dps-framework]. Normally, such a policy provides a methodology for key rollover. A DNS operator may choose to change the DNSSEC Policy for a zone, or switch to a different policy. Such a change may also trigger a key rollover scenario to occur.

Key rollovers are time critical, multiple steps processes. This document describes issues surrounding the timing of events in the rolling of DNSSEC keys.

The structure of this document is as follows. In section Section 2, more terminology on keys is provided. The timelines for the various methods of key rollovers are presented in section Section 3. Section Section 4 deals with key rollovers initiated by a change in a DNSSEC Policy.

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

1.1. Key Rollover Considerations

A key rollover involves the replacement of active keys with new keys. In order to avoid the zone being seen as bogus during the transition, there are constraints on the times at which the keys are added to and removed from the zone. DNSSEC records are not only held at the authoritative name server, they are also cached at client validators. The data on these systems can be interlinked, meaning a validator 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.

There exist different flavours of key rollovers. When making a choice which type of rollover to implement, several considerations may be taken into account:

- Size of the zone and the DNS response: Adding signatures increases the zone size and the size of DNS responses significantly. To keep the sizes of the zone and responses as small as possible, the DNSSEC records should be introduced only when they are required. For the same reason, dead keys and signatures must be removed periodically.

- Size of the DNSKEY RRset: Instead of keeping the set of signatures to a minimum, it is also possible to minimize the size of the DNSKEY RRset. This consideration may be of importance in the case where trust anchor priming is an issue.

- Interactions with the Parent: Where a key being replaced has a corresponding DS record in the parent zone, the rollover involves removing it and introducing the DS record corresponding to the new key. Such a process requires communication between the child and parent zones and may require additional operational work. This may lead to a sufficient delay. In the case where the interaction through the child-parent provisioning chain is unpredictable, it is preferred to keep the number of interactions with the parent to a minimum.

1.2. Terminology

The terminology used in this document is as defined in [RFC4033], [RFC4034], [RFC4035] and [RFC5011]. This document also introduces new terms in Section 2.

2. Key Definitions

2.1. Key Types

Keys can be used to authenticate information within the zone. Such keys are said to be Zone Signing Keys (ZSKs). In addition, keys can be used to authenticate the DNSKEY RRset in the zone. These keys are said to be Key Signing Keys (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 identifying a key. However, when this document talks about a ZSK it actually means that the key is used as ZSK (but may also be used as KSK). In the same spirit, a KSK is a key that is used as KSK (but may also be used as ZSK). A key that is used as a KSK is responsible for creating a signature for the DNSKEY RRset. A key that is used as a ZSK is responsible for creating a signatures for all RRsets, except the DNSKEY RRset.

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, 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

In this document, the key states from [key-timing] have been unraveled. Instead of a single state, the state of all information associated with the key is representated separately. This information comprises up to three items called Validation Components: the public key, its created signatures, and the corresponding secure delegation.

2.2.1. Validation Components

2.2.1.1. The Public Key Component

The Public Key (DNSKEY) Component represents the state of the public part of the key. When talking about a KSK, this comprises the DNSKEY record and the RRSIG record for the DNSKEY RRset created with the key, as both the key and signature travel together. In the case of a ZSK, this comprises just the DNSKEY record.

[MM: Is this a safe assumption? Or are there rollover scenarios that benefit to decouple the DNSKEY RR and the RRSIG RR created with the key? For example in STSS environment.]

2.2.1.2. The Signature Component

The Signature (RRSIG) Component represents the state of the private part of the key. This comprises the RRSIG records for all RRsets excluding the DNSKEY RRset.

2.2.1.3. The Secure Delegation Component

The Secure Delegation (DS) Component represents the state of the secure delegation of the key. This comprises the DS record that corresponds to the Public Key Component.

2.2.2. Validation Component States

The consequence of this unraveling is that a single [key-timing] key state now comprises a set of multiple Validation Component states. A Validation Component may exist in up to two places: it can be present in the corresponding zone and it may be known in validator caches. Thus, all Validation Components follow the same state diagram:

Hidden --> Introduced --> Propagated --> Withdrawn --> Dead.

Hidden: The Validation Component is not available in the zone. In this state, no validators are able to fetch this Validation Component.

Introduced: The Validation Component is introduced and, as a result, is available in the zone. If the Validation Component comprises multiple RRs, the introduction may be done incrementally. As a result, the Validation Component that is said to be Introduced may be only partly available in the zone. In this state, there may be validators that fetch this Validation Component from the authoritative name server. However, there may also be validators that have associated information in the cache and don't use the new Validation Component.

Propagated: The Validation Component is available in the zone and enough time has passed to have it propagated into all validator caches. If the Validation Component comprises multiple RRs, it is said to be Propagated if and only if all RRs have been propagated into all validator caches. As a result, all validators fetch this Validation Component from cache or from the authoritative name server.

Withdrawn: The Validation Component is being withdrawn from the zone. If the Validation Component comprises multiple RRs, the withdrawal may be done incrementally. As a result, the Validation Component that is said to be Withdrawn may still be partly available in the zone. In this state, the Validation Component can also still live in validator caches.

Dead: The Validation Component is not available in the zone anymore and enough time has passed to have it expire from all validator caches.

A Key State can now be represented as the tuple (DNSKEY Component State, RRSIG Component State, DS Component State). For example:

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

where S(Kc) is the state of the key (Kc), means that Kc is published in the zone and all the validators that have a copy of the DNSKEY RRset, have one that includes Kc. In addition, the key is being used for signing RRsets: RRSIG records made with Kc have been introduced in the zone. However, there may still be some validator caches that are unaware of these signatures. Finally, the corresponding DS record of Kc is said to be Hidden, meaning it has not yet been submitted to the parent.

For convenience, a ZSK can be represented as a tuple (DNSKEY State, RRSIG State), because the DS record is only used with KSKs. And a KSK can be represented 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 to and expire from validator caches at the same time.

2.3. Key States

During the rolling process, a key moves through different states. Key States are derived from the Validation Component States. For example, if the DNSKEY Component of a key is in the Introduced State, the key is said to be Published. A key can be in multiple states at the same time.

Uninformed: A key is said to be Uninformed, if all Validation Components are in the Hidden state. The key has been created, but has not yet been used for anything.
S(k) = (DNSKEY Hidden, RRSIG Hidden, DS Hidden)

Published: A key is said to be Published if the DNSKEY Component is in the Introduced state. The DNSKEY record is published in the zone, but predecessors DNSKEY RRsets may be held in caches.
S(k) = (DNSKEY Introduced, RRSIG *, DS *)

Active: A key is said to be Active if the RRSIG Component is in the Introduced state (for ZSKs).
S(k) = (DNSKEY *, RRSIG Introduced, DS *)

ActiveDS: A key is said to be ActiveDS, or Submitted, if the DS Component is in the Introduced state (for KSKs).
S(k) = (DNSKEY *, RRSIG *, DS Introduced)

Known: A key is said to be Known if the DNSKEY Component is in the Propagated state. The new key data has been published for long enough to guarantee that any previous versions of it have expired from caches.
S(k) = (DNSKEY Propagated, RRSIG *, DS *)

Safe: A key is said to be Safe if the RRSIG Component is in the Propagated state (for ZSKs). If a validator knows about the key, it is safe to assume that it may expect a signature created with this key.
S(k) = (DNSKEY *, RRSIG Propagated, DS *)

SafeDS: A key is said to be SafeDS if the DS Component is in the Propagated state (for KSKs). If a validator knows about the key, it is safe to assume that it may expect a corresponding DS record for this key.
S(k) = (DNSKEY *, RRSIG *, DS Propagated)

Removed: A key is said to be Removed if the DNSKEY Component is in the Withdrawn state. The key has been removed from the zone.
S(k) = (DNSKEY Withdrawn, RRSIG *, DS *)

Retired: A key is said to be Retired if the RRSIG Component is in the Withdrawn state (for ZSKs). Signatures are removed, or are incrementally being removed, from the zone. When a key is said to be Retired, there may still be caches that hold copies of the signatures.
S(k) = (DNSKEY *, RRSIG Withdrawn, DS *)

RetiredDS: A key is said to be RetiredDS if the DS Component is in the Withdrawn state (for KSKs). The request has been made to withdraw the DS record from the parent zone, but it may take some time before the record is actually removed. When a key is said to be RetiredDS, there may still be caches that hold copies of the DS record.
S(k) = (DNSKEY *, RRSIG *, DS Withdrawn)

Forgotten: A key is said to be Forgotten if the DNSKEY Component is in the Dead state. At this point, no single validator cache should know about this key.
S(k) = (DNSKEY Dead, RRSIG *, DS *)

Expired: A key is said to be Expired if the RRSIG Component is in the Dead state (for ZSKs). At this point, no single validator cache should know about this key's signatures.
S(k) = (DNSKEY *, RRSIG Dead, DS *)

ExpiredDS: A key is said to be ExpiredDS if the DS Component is in the Dead state (for KSKs). At this point, no single validator cache should know about this key's DS record.
S(k) = (DNSKEY *, RRSIG *, DS Dead)

Throughout the document, the states of a key Kc is denoted as P(Kc). For example,

P(Kc) = Known Retired

means that key Kc is considered to be Known, the new key data has been published for long enough to guarantee that any previous versions of it have expired from caches, and it is considered to be

Retired, its signatures are being removed from the zone.

2.4. Key Goals

When performing a key rollover, it is usually intended to introduce a new key into the zone and to remove an existing key from the zone. These intentions can be called Key Goals. A Key Goal is the desire to make a key have certain Key State, as described in Section 2.3. During the lifetime of a key, the following goals may be put on a key:

- Activate key: Make validators use the key's associated information to perform authentication. The goal is reached if the key is said to be Known and Safe(DS).

- Remove key: Make validators forget about the key's associated information. The goal is reached if the key is said to be Forgotten and Expired(DS).

- Stand-by key: Pre-publish information for this key to speed up a future (unscheduled) rollover. In case of a Stand-by ZSK, the goal is reached if the key is said to be Known. In case of a Stand-by KSK, the goal is reached if the key is said to be SafeDS.

2.5. Delay Timings

For every change made in the zone there are time delays that need to be taken into account:

Software Delay (Dsfw): The time it takes for the software to introduce the new information in the zone. This delay can vary a lot 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 validator 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 the values of these delays may vary for the different rollover methods, the propagation delay to the caches can be defined 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 the zone, DcacheK is the propagation delay to the caches for the DNSKEY RRset and DcacheP is the propagation delay for information published in the parent zone.

[MM: Because some timings are unpredictable, it would make more sense to use triggering events instead of timings]

3. Key Rollovers

There are many different key rollover methods. Section 1.1 lists several considerations 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 be removed and a successor key (Ks) that needs to become active.

3.1. Key Rollover Stages

Broadly speaking, any key rollover can be thought of as the following sequence of stages:

Generation: In this stage, a new successor key is generated or derived from a key pool.

Preparation: In this stage, one or more Validation Components of the successor key are published in the zone and propagate through the nameserver network and into caches.

Ready: The Ready Stage always follows the Preparation Stage. The initial Validation Components of the successor key have been published long enough to guarantee that where key validation components for this zone appear in caches, they will include the

components for the successor key.
Not all rollovers go through the Preparation and Ready stages.
The stages exist to facilitate rollover methods where a subset of Validation Components is introduced first and the final Validation Components are changed in an atomic manner.
It is possible that a rollover goes through the Preparation and Ready stages multiple times.

Transition: The final Validation Components are added to the zone:
The DNSKEY record is published in the zone, if it has not yet been introduced in previous Rollover Stages. In addition, a key that acts as ZSK is started to be used to sign RRsets, if it has not yet been started to do so in previous Rollover Stages. A key that acts as a KSK has its corresponding DS record submitted to the parent, if the DS record was not yet submitted in previous Rollover Stages.
During this stage, it is not guaranteed that all RRs can be validated with the successor key information; some may only be able to be validated with with information from the predecessor key.
In some rollover scenario's, the Transition Stage is an atomic operation, where Validation Components of the successor key replace the Validation Components of its predecessor.

Transited: The Transited Stage always follows the Transition Stage.
All the Validation Components of the successor key have been published long enough to guarantee that any cache that holds validation components for a RR in the zone will contain a copy of these components.

Revocation: If the key acts as a KSK, and it is known that the key is used as a [RFC5011] trust anchor, the predecessor key must be published for a period with the REVOKE bit set as a way of notifying validators that may have the key configured as a trust anchor, that is is about to be removed from the zone.

Revoked: The Revoked Stage always follows the Revocation Stage.
The revoked predecessor key has been published long enough to guarantee that RFC5011-aware validators have seen the key being revoked.
Note that if the key is not used as an RFC5011 trust anchor, the rollover will not go through the Revocation and Revoked stages.

Withdrawal: At this point, there may still exist old Validation Components that belong to the predecessor key. Because a successor key is available, it is safe to withdraw all remaining old Validation Components.

Complete: All the Validation Components of the predecessor key have been removed from the zone long enough to guarantee that they have expired from caches.

3.2. ZSK Rollovers

The two most common rollover methods for ZSKs are Double-Signature and Pre-Publication. Both are described in RFC4641 [RFC4641]. [key-timing] also introduces ZSK Double-RRSIG rollover. These three rollover methods are shaped like this because different rollover considerations are being taken into account. Pre-Publication minimizes the number of signatures over the RRsets in the zone and DNS responses. Double-RRSIG keeps the size of the DNSKEY RRset to a minimum. Double-Signature is the fastest way to roll a ZSK, because no considerations are being taken into account.

3.2.1. 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 validators 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.

The successor key Ks needs to be Known and Safe before Kc can be removed. First, all Validation Components of the successor key need to be introduced into the zone. Once all have been propagated, all information of Kc can be withdrawn from the zone.

The timeline diagram is shown below:

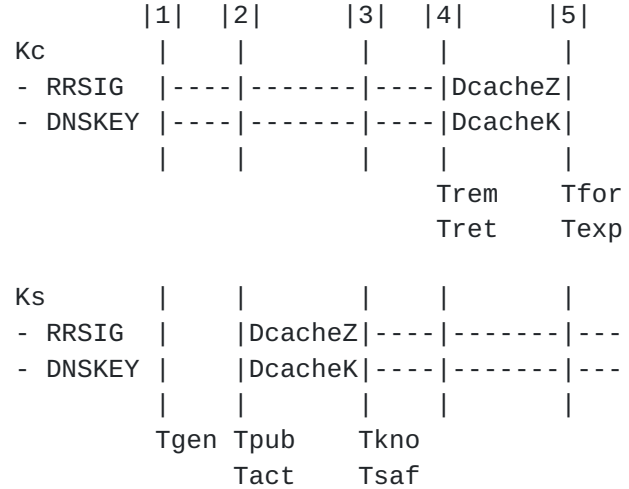

Figure: ZSK Double-Signature Rollover.

Generation Stage: Event 1
Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden)
P(Ks) = Uninformed

With the ZSK Double-Signature Rollover, all new Validation Components of the key Ks are going to be added to the zone and are allowed to propagate into the caches of validators. Thus, there is no need for Preparation and Ready Stages.

Transition Stage: Event 2
Key Ks is added to the DNSKEY RRset and is immediately used to sign the zone; existing signatures in the zone are maintained. This is Ks's publish time (Tpub) and Ks is said to be Published. It is also Ks's active time (Tact), the time when Ks is said to be Active. Because the Double-Signature rollover is in place, there are now temporarily two active keys.

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

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

Transited Stage: Event 3
The information for Ks must be published long enough to ensure that the information have reached all validators 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 signatures 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)
P(ks) = Known Safe

Note that once the DNSKEY RRset containing both Kc and Ks has propagated to all caches, Kc can be retired (i.e. no longer used to sign RRsets). It does not matter if not all signatures of Ks have been Propagated, since the validator 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.

Withdrawal Stage: Event 4
When the successor key Ks is said to be Propagated, Kc can be retired. And once there is a successor key that is said to be Safe, Kc can be removed. This is Kc's retire time (Tret) and Kc is said to be Retired. It is also Kc's removal time (Trem), the time that Kc is said to be Removed.

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

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

Complete Stage: 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)
P(Kc) = Forgotten Expired

3.2.2. 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 validators 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. Although it takes more time than the Double-Signature method, it has the advantage that each RRset is signed with just one key. As a result, it has the advantage that the amount of DNSSEC data is kept to a minimum, reducing the impact on performance.

Only when Ks is said to be Known, Kc may be retired. Signatures may be retired all at once or may be incrementally replaced with signatures of Ks. However, during the transition all RRsets must either be signed with Kc or be signed with Ks. If Ks is considered to be Known and Safe, the DNSKEY record of Kc can be removed.

The timeline diagram looks like this:

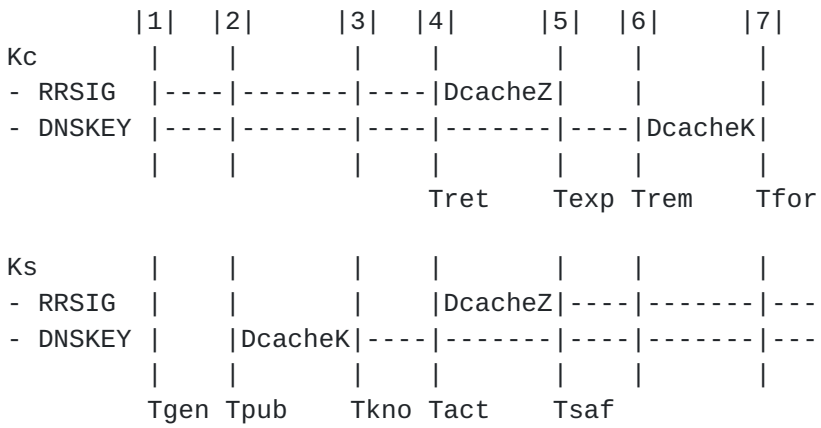

Figure: ZSK Pre-Publication Rollover.

With the ZSK Pre-Publication rollover, the DNSKEY record of Ks needs to be pre-published before the rollover can go into the Transition

Stage.

Generation Stage: Event 1
Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden)
P(Ks) = Uninformed

Preparation Stage: 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's 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 Hidden)
P(Ks) = Published

Ready Stage: Event 3
Before Ks can be used, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator that has a copy of the DNSKEY RRset in its cache 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 Hidden)
P(Ks) = Known

At this point, the rollover is in the Ready Stage.

Transition Stage: 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's active time (Tact), the time that Ks is said to be Active. It is also Kc's 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)
P(Kc) = Known Retired
S(Ks) = (DNSKEY Propagated, RRSIG Introduced)
P(Ks) = Known Active

Transited Stage: 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 validator 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's 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, Texp(Kc) == Tsaf(Ks)

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

Withdrawal Stage: 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's removal time (Trem), the time that Kc is considered to be Removed.

Trem(Kc) >= Tsaf(Ks)

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

Complete Stage: Event 7
From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record of 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)
P(Kc) = Forgotten Expired

3.2.3. Double-RRSIG

[MM: Note it comes down to "signatures, generated with a key, whose public part is not published".]

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 validators 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 old signatures. It also takes more time than the Double-Signature method. The method will be used where it is desired not to 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 disadvantage is that for a period, each RRset returned will be accompanied by two RRSIGs.

When Ks is said to be Safe, the DNSKEY record of Kc may be removed. At the same time that the DNSKEY record of Kc is removed, the DNSKEY record for Ks is introduced. If Ks is considered to be Known and Safe, Kc no longer needs to generate signatures.

The timeline diagram is shown below:

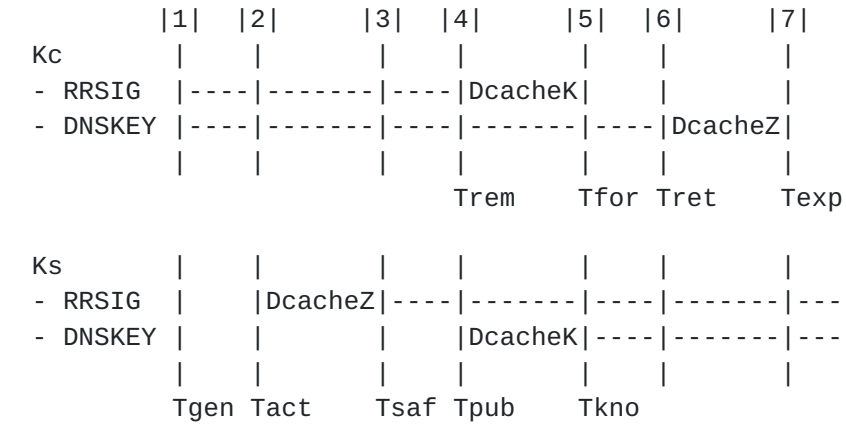

Figure: ZSK Double-RRSIG Rollover.

Generation Stage: Event 1
Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside

world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden)
P(Ks) = Uninformed

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

Tact(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Hidden, RRSIG Introduced)
P(Ks) = Active

Ready Stage: Event 3
Before the DNSKEY from Kc can be switched to Ks, the signatures of Ks must be published for long enough (DcacheZ) to guarantee that any validator 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 Hidden, RRSIG Propagated)
P(Ks) = Safe

Transition Stage: Event 4
At some point in time, the decision is made to publish Ks. This point in time is Ks's publish time (Tpub), the time that Ks is said to be Published. At the same time, the DNSKEY RR of 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)
P(Kc) = Removed Safe
S(Ks) = (DNSKEY Introduced, RRSIG Propagated)
P(Ks) = Published Safe

Transited Stage: Event 5
The signatures of Kc need to be retained in the zone until the DNSKEY RR has expired from all validator 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)
P(Kc) = Forgotten Safe
S(Ks) = (DNSKEY Propagated, RRSIG Propagated)
P(Ks) = Known Safe

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

Tret(Kc) >= Tsaf(Ks)

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

Complete Stage: 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)
P(Kc) = Forgotten Expired

3.3. KSK Rollovers

The most common rollover method for KSKs is Double-Signature, described in RFC4641 [RFC4641]. Two more methods are identified in [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 the DNSKEY RRset as small as possible.

Note that with the KSK rollovers, it is out of scope whether the information within the zone is authentic. It is assumed that there exists one or more ZSKs in the DNSKEY RRset that takes care of this during the rollover.

3.3.1. 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 validator caches, the old DNSKEY and DS record are removed.

Ks needs to be Known and SafeDS, before Kc can be removed. First,

all new information for Ks need to be introduced into the zone. Once all have been propagated, all information of Kc can be withdrawn from the zone.

The timeline diagram looks like this:

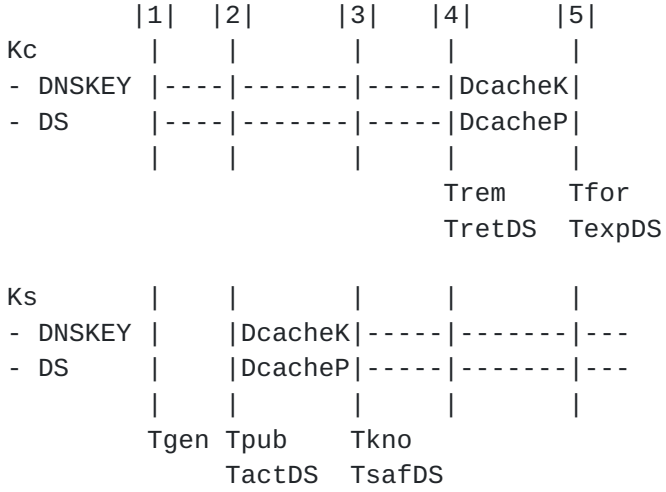

Figure: KSK Double-RRset Rollover.

Generation Stage: Event 1
Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, DS Hidden)
P(Ks) = Uninformed

With the KSK Double-RRset Rollover, all new Validation Components of the key Ks are going to be added to the zone and are allowed to propagate into the caches of validators. Thus, there is no need for a Preparation Stage.

Transition Stage: 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's publish time (Tpub), the time that Ks is said to be Published. It is also Ks's submit time (TactDS), the time that the DS record of Ks is Submitted (ActiveDS).

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

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

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

Transited Stage: Event 3
The information for Ks must be published long enough to ensure that the information have reached all validators 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)
P(Ks) = Known SafeDS

Note that the request to the parent to withdraw the DS record of Kc can already be made after DcacheK. It does not matter if the DS record for Ks has not yet been propagated, since the validator 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.

Withdrawal Stage: Event 4
Once the successor key Ks is said to be Known, the DS record of Kc can be withdrawn. If Ks is also said to be SafeDS, the DNSKEY record of Kc can be removed from the zone. This is Kc's retire time (Tret), the time that Kc is said to be RetiredDS. It is also Kc's 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)
P(Kc) = Removed RetiredDS

Complete Stage: 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)
P(Kc) = Forgotten Expired

3.3.2. 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 the number of interactions to the parent needs to be minimized, this rollover method is preferred over the Double-RRset method. As a consequence, the DS record of Ks can be submitted to the parent only if it is safe to withdraw the DS record of Kc.

When Ks is said to be Known, the DS record can be changed When Ks is considered to be Known and SafeDS, the DNSKEY record of Kc can be removed.

The timing diagram for such a rollover is:

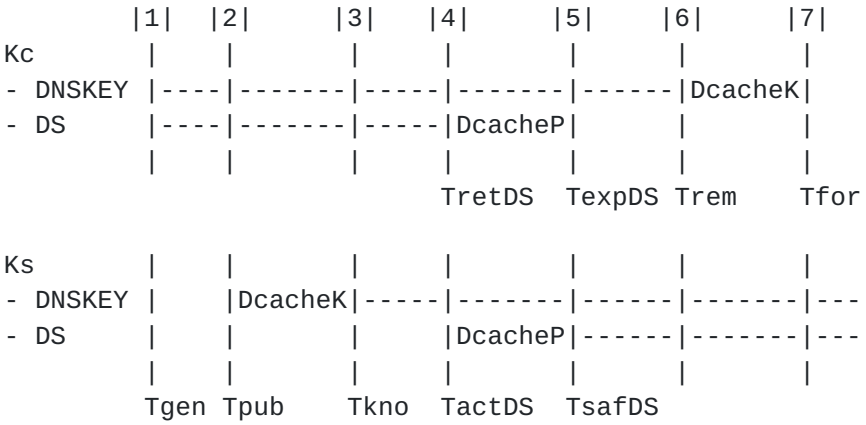

Figure: KSK Double-Signature Rollover.

Generation Stage: Event 1
Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, DS Hidden)
P(Ks) = Uninformed

Preparation Stage: Event 2
Ks is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by Ks and all currently active 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 Hidden)
P(Ks) = Published

Ready Stage: Event 3
Before the corresponding DS can be submitted, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator 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 Hidden)
P(Ks) = Known

Transition Stage: Event 4
At some later time, the DS RR corresponding to Ks is submitted to the parent zone for publication, with a request that it replaces the DS RR corresponding to Kc. This time is Ks's submit time (TactDS), the time that Ks is considered to be Submitted. It is also Kc's 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)
P(Ks) = Known RetiredDS
S(Ks) = (DNSKEY Propagated, DS Introduced)
P(Ks) = Known ActiveDS

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

Transited Stage: Event 5
All validators that have the DS RRset cached will have a a copy that includes the new DS record. This is Ks's 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)
P(kc) = Known ExpiredDS
S(Ks) = (DNSKEY Propagated, DS Propagated)
P(Ks) = Known SafeDS

Withdrawal Stage: Event 6
When the new DS record has been propagated, the DNSKEY record of Kc

can be removed from the zone. This is Kc's removal time (Trem), the time that Kc is said to be Removed.

Trem(Kc) >= TsafDS(Ks)

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

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

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

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

3.3.3. Double-DS

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

If the size of the DNSKEY RRset needs to be minimized, 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.

When Ks is said to be SafeDS, the DNSKEY record of Kc can be removed. At the same time, the DNSKEY record of Ks can be introduced. When Ks is considered to be Known and SafeDS, the DS record of Kc can be removed.

The timeline diagram looks like this:

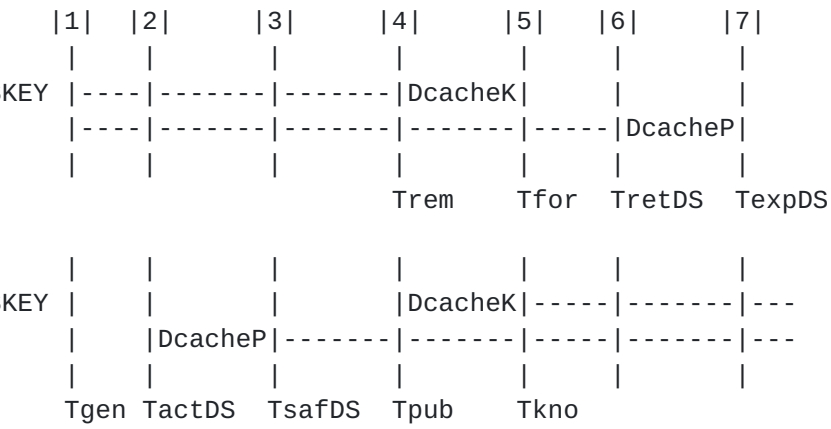

Figure: KSK Double-DS Rollover.

Generation Stage: Event 1
Ks is generated at time Tgen. No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, DS Hidden)
P(Ks) = Uninformed

Preparation Stage: Event 2
Before the new key Ks can be introduced into the zone, the new DS record needs to be submitted. This is allowed, because there exists a valid chain of trust for the same algorithm (with the current key Kc). This is Ks's submit time (TactDS), the time that the DS record of Ks was submitted and is said to be ActiveDS.

TactDS(Ks) >= Tgen(Ks)

S(Ks) = (DNSKEY Hidden, DS Introduced)
P(Kc) = ActiveDS

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

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

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

S(Ks) = (DNSKEY Hidden, DS Propagated)
P(Ks) = SafeDS

Transition Stage: Event 4

Because there are now two trust anchors a validator can use, the DNSKEY record of Kc can be switched with the DNSKEY record of Ks. Kc becomes retired and the DNSKEY RRset is only signed with Ks. This time is Ks's publish time (Tpub), the time that Ks is said to be Published. It is also Kc's 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)
P(Kc) = Removed SafeDS
S(Ks) = (DNSKEY Introduced, DS Propagated)
P(Ks) = Published SafeDS

Transited Stage: Event 5
Before the DS record of Kc can be withdrawn, Kc will have to expire from validator caches. When the DNSKEY RRset that includes Kc has expired from caches, Kc is said to be Forgotten and Ks is said to be Known. This happens at Ks's known time, given by:

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

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

Withdrawal Stage: Event 6
Now that there is a key Ks that is said to be Propagated and SafeDS, the DS record of Kc can be withdrawn. This is Kc's retire time (TretDS), the time that there is no need for a secure delegation for Kc anymore.

TretDS(Kc) >= Tkno(Ks)

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

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

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

S(Kc) = (DNSKEY Dead, DS Dead)

P(Kc) = Forgotten ExpiredDS

3.3.4. Interaction with Configured Trust Anchors

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

3.3.4.1. Adding a KSK

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

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 [RFC5011]. This ensures that the successor key is at least seen twice by RFC5011-aware 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.

3.3.4.2. Removing a KSK

When the current key is ready to be removed from the zone, it is instead going to be revoked. The REVOKE bit is set 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 RFC5011-aware validators.

After that delay, every RFC5011-aware validator has seen the revoked key and the DNSKEY record may be removed from the zone. Another DcacheK delay, the key has fully expired from all the validator caches.

3.4. Rollovers in a Single Type Signing Scheme

Previous sections described the possible ways to roll keys that have one key type (either ZSK or KSK). In situations where a Single Type

Signing Scheme (STSS) is used, one key is used both as ZSK and KSK. This key is responsible for authenticating information within the zone, as well as authenticating the DNSKEY RRset. STSS Rollovers can be constructed by combining a ZSK rollover method with a KSK rollover method. However, not all combinations are possible. For example, the ZSK Double-RRSIG rollover is only suitable for combining with the KSK Double-DS rollover, because both keep the DNSKEY RRset to a minimum size. The other rollovers are ruled out because they require a period where both the DNSKEY record of the current key and its successor are being served at the same time.

The ZSK Pre-Publication method is suitable for combining with the KSK Double-RRset and KSK Double-Signature rollover methods, but does not gain any advantages when combined with the KSK Double-RRSet. In both cases the DNSKEY record needs to be post-published, taking a similar amount of time. However, the KSK Double-RRset requires two interactions with the parent, while the KSK Double-Signature only involves one interaction. Therefore, the ZSK Pre-Publication rollover combined with the KSK Double-RRset is left out of this document.

The ZSK Double-Signature method is suitable for combining with both the KSK Double-RRset and the KSK Double-Signature method.

To conclude, there are four different STSS rollover methods.

3.4.1. Double-RRset

This is a combination of the ZSK Double-Signature rollover and the KSK Double-RRset rollover. The new key is added to the DNSKEY RRset, and all RRsets - including the DNSKEY RRset - 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 validator 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, the disadvantages are that 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:

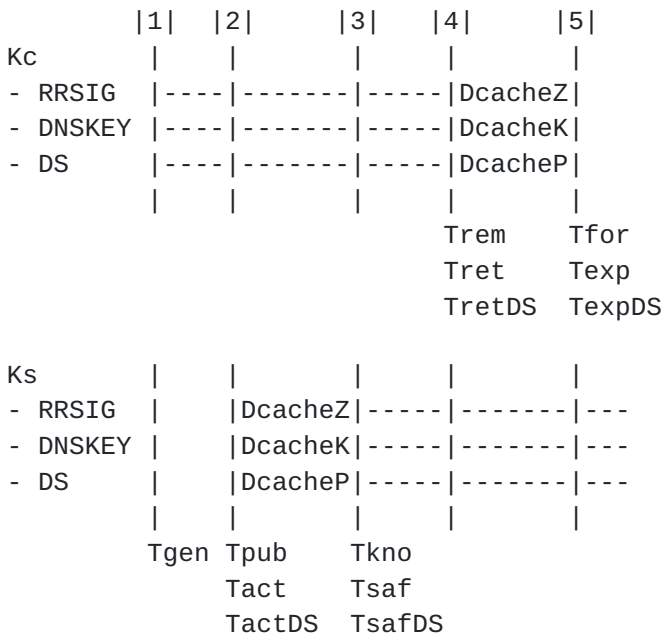

Figure: STSS Double-RRset Rollover.

The rollover method is almost the same as that of the KSK Double-RRset rollover, except now also DcacheZ has to be taken into account.

3.4.2. 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:

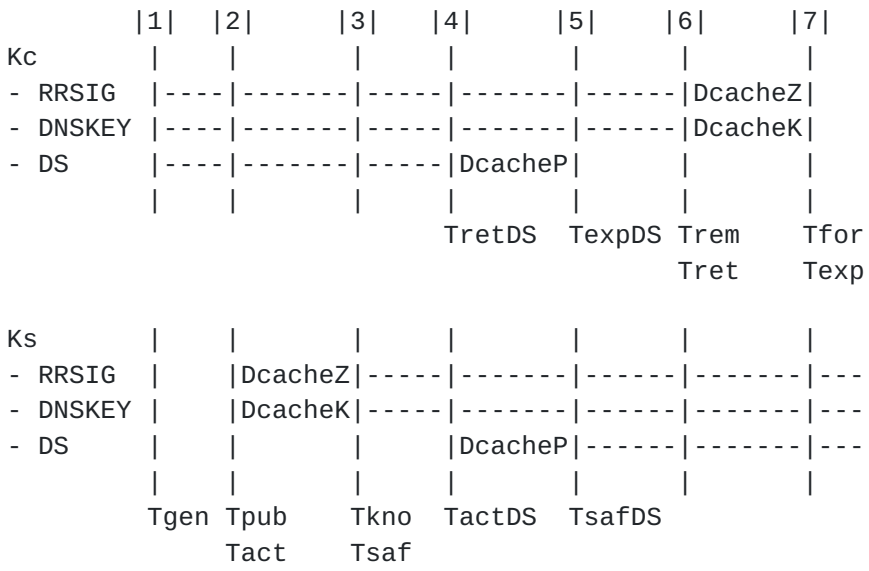

Figure: STSS Double-Signature Rollover.

The rollover method is almost the same as that of the KSK Double-RRset rollover, except now also DcacheZ has to be taken into account.

3.4.3. 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, the non-DNSKEY RRsets require only one signature during the rollover. If speed is not an issue, this rollover method might be the way to go in a STSS environment, since it optimizes in both size and interactions with the parent.

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:

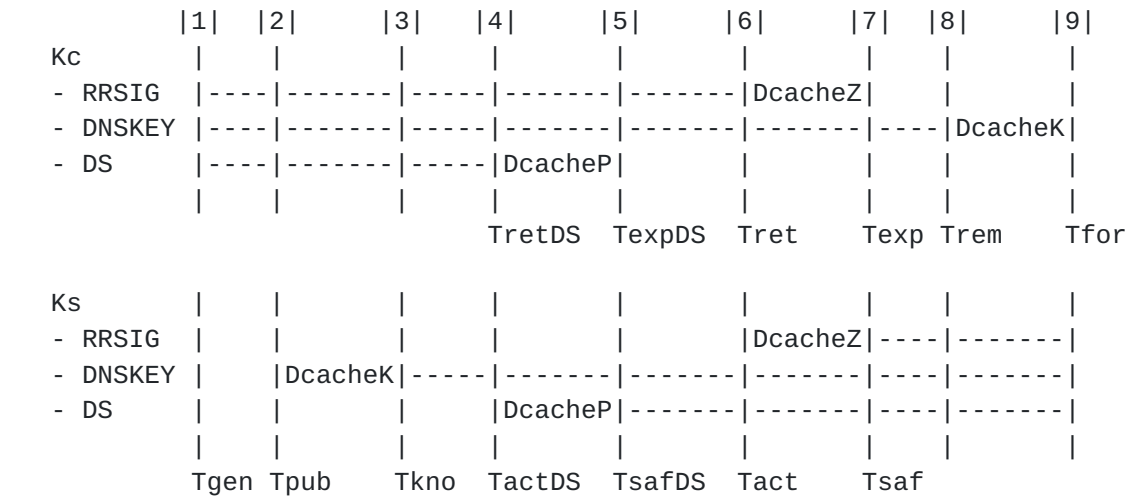

Figure: STSS Pre-Publication Rollover.

Generation Stage: Event 1
Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden, DS Hidden)
P(Ks) = Uninformed

Preparation Stage: 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's 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 Hidden, DS Hidden)
P(Ks) = Published

Ready Stage: Event 3
Before the DS record may be switched, the DNSKEY record of Ks must be published for long enough (DcacheK) to guarantee that any validator 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's known time (Tkno), which is given by:

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

S(Ks) = (DNSKEY Propagated, RRSIG Hidden, DS Hidden)
P(ks) = Known

Preparation Stage (2): Event 4
At Ks's submit time (TactDS), the DS RR corresponding to Ks is submitted to the parent zone for publication, with a request that it replaces the DS RR corresponding to Kc. Ks is said to be ActiveDS. This time is also Kc's retire time (TretDS) and Kc is said to be RetiredDS.

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

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

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

Ready Stage (2): Event 5
All validators use the DS RRset that includes a copy of the DS record of Ks. At this time, Ks's safe time (TsafDS), Ks is said to be SafeDS. But Kc is still used as the ZSK.

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

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

Transition Stage: 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 key lifetime. This point in time is Ks's active time (Tact), the time that Ks is said to be Active. It is also Kc's 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))
P(Kc) = Known Retired ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Introduced, DS Propagated))
P(Ks) = Known Active SafeDS

Transited Stage: 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 validator 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's safe time (Tsaf), the time that Ks is considered to be Safe, and Kc's 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)
P(Kc) = Known Expired ExpiredDS
S(Ks) = (DNSKEY Propagated, RRSIG Propagated, DS Propagated)
P(Ks) = Known Safe SafeDS

Withdrawal Stage: 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's removal time (Trem), the time that Kc is considered to be Removed.

Trem(Kc) >= Tsaf(Ks)

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

Complete Stage: Event 9
From the perspective of the authoritative server, the rollover is complete. After some delay, The DNSKEY record of 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)
P(Kc) = Forgotten Expired ExpiredDS

3.4.4. Double-DS

This is a combination of the ZSK Double-RRSIG rollover and the KSK Double-DS rollover. This keeps the DNSKEY RRset to a minimum size, but at the cost of double signatures in the zone and the necessity of two interactions with 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 record is withdrawn from the parent zone.

The timeline diagram looks like this:

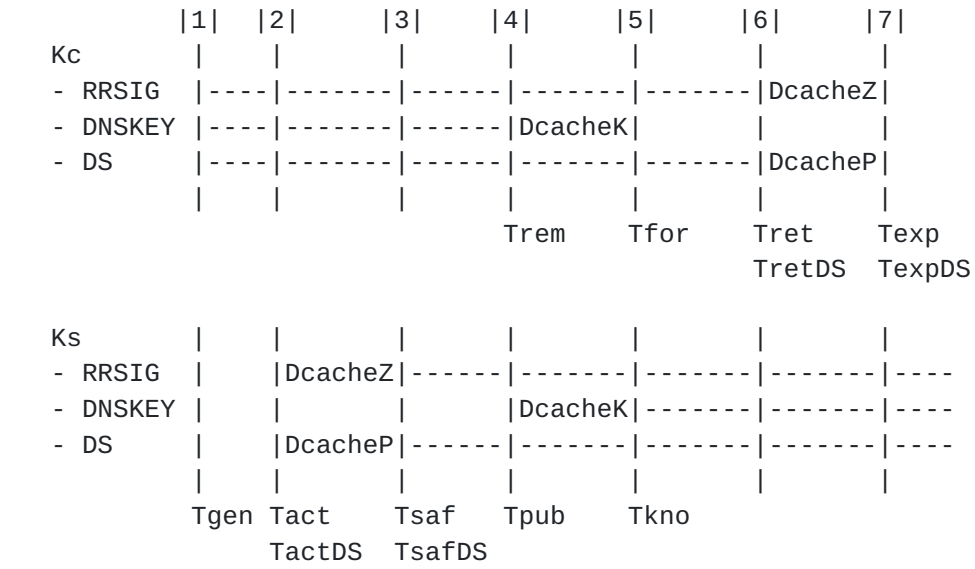

Figure: STSS Double-DS Rollover.

Generation Stage: Event 1
Key Ks is generated at the generate time (Tgen). No information about the key is yet published, it is still unknown to the outside world.

S(Ks) = (DNSKEY Hidden, RRSIG Hidden, DS Hidden)
P(Ks) = Uninformed

Preparation Stage: Event 2
Before the new key Ks can be introduced into the zone, the new signatures must be introduced and the new DS record must be submitted. This time is Ks's active time (Tact), the time that Ks is said to be Active. It is also Ks's submit time (TactDS), the time that the DS record of Ks was submitted and is said to be ActiveDS.

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

S(Ks) = (DNSKEY Hidden, RRSIG Introduced, DS Introduced)
P(Kc) = Active ActiveDS

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

Ready Stage: Event 3
The new signatures and the new DS RRset have been propagated. This is Ks's 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 Hidden, RRSIG Propagated, DS Propagated)
P(Ks) = Safe SafeDS

Transition Stage: Event 4
Because there are now two trust anchors a validator can use, the DNSKEY record of Kc can be switched with the DNSKEY record of Ks. This time is Ks's publish time (Tpub), the time that Ks is said to be Published. It is also Kc's 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)
P(Kc) = Removed Safe SafeDS
S(Ks) = (DNSKEY Introduced, RRSIG Propagated, DS Propagated)
P(Ks) = Published Safe SafeDS

Transited Stage: Event 5
When the signatures of Kc and its corresponding DS record have expired from the caches, the DNSKEY record of Kc can be withdrawn from the zone. 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's known time, given by:

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

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

Withdrawal Stage: Event 6
Ks is said to be Propagated and SafeDS, and the signatures and DS record of Kc may be withdrawn. This is this Kc's 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)
P(Kc) = Forgotten Retired RetiredDS

Complete Stage: 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)
P(Kc) = Forgotten Expired ExpiredDS

3.5. Stand-by Keys

Although keys will usually be rolled according to some regular schedule, there may be occasions where 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 expire. As a consequence, a stand-by 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 state. This goal is reached at TsafDS, step 3 in the Double-DS method timeline diagram.

The DS record for the stand-by KSK Ks should be propagated to the caches before the key lifetime of the predecessor KSK Kc expires:

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 (un)scheduled key rollovers, changes in policy may occur. The initial transition is enabling DNSSEC. The counterpart, disabling DNSSEC, is also possible. Two other examples of policy changes 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.

A zone must be fully signed, before the DS associated with the initial KSK is published. The ZSK and KSK can be the same key, for example in a Single Type Signing Scheme.

The timeline diagram is shown below:

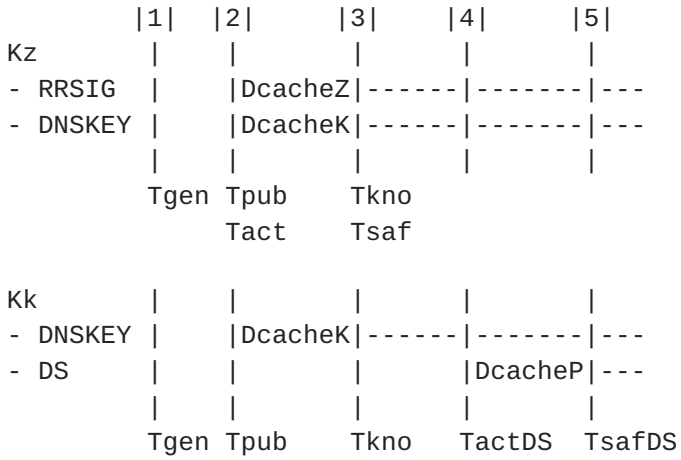

Figure: Enabling DNSSEC.

Generation Stage: Event 1
Kk and Kz are generated. The time when this happens is Tgen, the time that the keys were generated (note that Tgen for Kk could be

different that Tgen for Kz).

S(Kk) = (DNSKEY Hidden, DS Hidden)
P(Kk) = Uninformed
S(Kz) = (DNSKEY Hidden, RRSIG Hidden)
P(Kk) = Uninformed

Preparation Stage: 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 Hidden)
P(Kk) = Published
S(Kz) = (DNSKEY Introduced, RRSIG Introduced)
P(Kz) = Published Active

Ready Stage: Event 3
Before the DS record can be committed, Kz must be considered Known and Safe. 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 Hidden)
P(Kk) = Known
S(Kz) = (DNSKEY Propagated, RRSIG Propagated)
P(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)

Transition Stage: Event 4
The DNSKEY RRset and all RRSIG records have reached the caches, and the DS record can be submitted to the parent. This is TactDS, the

time that the DS has been submitted to the parent.

TactDS(Kk) >= Tkno(Kk)

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

Transited Stage: Event 5
The DS has been published in the parent zone. Some more time later, all validators 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)
P(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. It is assumed that there is a KSK (Kk) and a ZSK (Kz) that are Known and Safe. The goals of this event are to make Kk and Kz both Forgotten and Expired.

The timeline diagram is shown below:

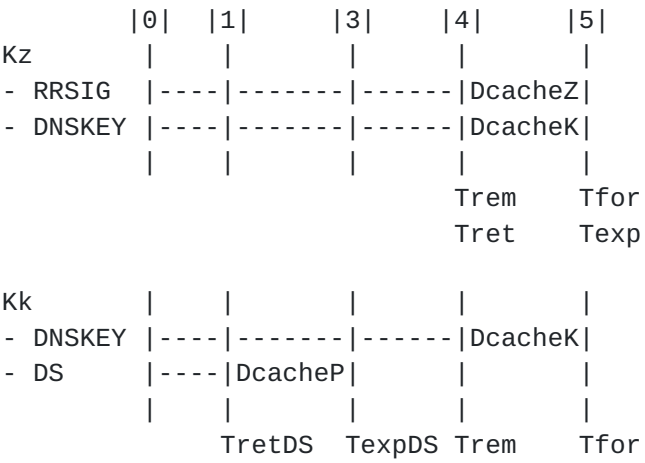

Figure: Disabling DNSSEC.

Transition Stage: 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)
P(Kk) = Known RetiredDS

Transited Stage: Event 2
First, the DS record of Kk must expire from all validator 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)
P(Kk) = Known ExpiredDS

Withdrawal Stage: Event 3
When no secure chain of trust to Kk exists anymore, the DNSKEY records of both keys and all RRSIG records can be removed from the zone. 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)
P(Kk) = Removed ExpiredDS
S(Kz) = (DNSKEY Withdrawn, RRSIG Withdrawn)
P(Kz) = Removed Retired

Complete Stage: 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)
P(Kk) = Forgotten ExpiredDS
S(Kz) = (DNSKEY Dead, RRSIG Dead)
P(Kz) = Forgotten Expired

4.3. Algorithm Rollover

When changing algorithms, it is possible that algorithms are added, removed or replaced. Adding and removing an algorithm follows 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, one should follow the timelines of the most restrive 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 [key-timing]. Significant parts of the text is taken from that document. Especially Section 3.2 and Section 3.3 are largely copied and adjusted to the new introduced terminology from this document.

Also, acknowledgements to Yuri Schaeffer, who brought to the attention the idea of key goals (Section 2.4) and whose discussions helped to shape this document.

8. Changelog

8.1. Changes with key-timing draft

This document builds further on [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.

- Four new Key States are introduced: Known, Safe, Forgotten and Expired, to represent whether information about the key exist in validator caches. The key states 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.

8.2. From -00 to -01

- Initial review Stephen Morris.

- Changed style, removed all first and second person style.

- Key Conditions are now called Key Properties.

- More detailed explanation on Key States Unraveled: Introduced Validation Components and Key Properties, described in different sections.

- Put the correct timeline figure in the section on STSS Double-DS rollover method.

- Review Marc Lampo, Stephen Morris

- Stephen provided text for the section on Key Rollover Stages

8.3. From -01 to -02

- Added a paragraph on document outline.

- Key Properties are now called Key States.

- Renamed Validaton Component State 'Generated' to 'Hidden', renamed Key State 'Generated' to 'Uninformed'.

- Renamed Rollover Stages 'Activation' and 'Activated' to 'Transition' and 'Transited'.

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.

[dps-framework] Ljunggren, F., Eklund-Lowinder, A-M., and T. Okubo, "DNSSEC Policy & Practice Statement Framework", March 2011.

[key-timing] Morris, S., Ihren, J., and J. Dickinson, "DNSSEC Key Timing Considerations", March 2011.

Appendix A. List of Symbols

[MM: To do]

Author's Address

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

EMail: matthijs@nlnetlabs.nl