Internet Engineering Task Force	S. Morris
Internet-Draft	ISC
Intended status: Informational	J. Ihren
Expires: April 28, 2011	Netnod
	J. Dickinson
	Sinodun
	October 25, 2010

TOC

DNSSEC Key Timing Considerations
draft-ietf-dnsop-dnssec-key-timing-01.txt

Abstract

This document describes the issues surrounding the timing of events in the rolling of a key in a DNSSEC-secured zone. It presents timelines for the key rollover and 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 April 28, 2011.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

1.1. Key Rolling Considerations

TOC

When a zone is secured with DNSSEC, the zone manager must be prepared to replace ("roll") the keys used in the signing process. The rolling of keys may be caused by compromise of one or more of the existing keys, or it may be due to a management policy that demands periodic key replacement for security or operational reasons. In order to implement a key rollover, the keys need to be introduced into and removed from the zone at the appropriate times. Considerations that must be taken into account are:

- DNSKEY records and associated information (such as RRSIG records created with the key or the associated DS records) are not only held at the authoritative nameserver, they are also cached at client resolvers. The data on these systems can be interlinked, e.g. a validating resolver may try to validate a signature retrieved from a cache with a key obtained separately.

- A query for the key RRset returns all DNSKEY records in the zone. As there is limited space in the UDP packet (even with EDNS0 support), dead key records must be periodically removed. (For the same reason, the number of stand-by keys in the zone should be restricted to the minimum required to support the key management policy.)

- Zone "boot-strapping" events, where a zone is signed for the first time, can be common in configurations where a large number of zones are being served. Procedures should be able to cope with the introduction of keys into the zone for the first time as well as "steady-state", where the records are being replaced as part of normal zone maintenance.

- To allow for an emergency re-signing of the zone as soon as possible after a key compromise has been detected, stand-by keys (additional keys over and above those used to sign the zone) need to be present.

Management policy, e.g. how long a key is used for, also needs to be considered. However, the point of key management logic is not to ensure that a "rollover" is completed at a certain time but rather to ensure that no changes are made to the state of keys published in the zone until it is "safe" to do so ("safe" in this context meaning that at no time during the rollover process does any part of the zone ever go bogus). In other words, although key management logic enforces policy, it may not enforce it strictly.

1.2. Types of Keys

TOC

Although DNSSEC validation treats all keys equally, [RFC4033] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “DNS Security Introduction and Requirements,” March 2005.) recognises the broad classification of zone-signing keys (ZSK) and key-signing keys (KSK). A ZSK is used to authenticate information within the zone; a KSK is used to authenticate the key set in the zone. The main implication for this distinction concerns the consistency of information during a rollover. During operation, a validating resolver must use separate pieces of information to perform an authentication. At the time of authentication, each piece of information may be in the validating resolver's cache or may need to be retrieved from the authoritative server. The rollover process needs to happen in such a way that at all times through the rollover the information is consistent. With a ZSK, the information is the RRSIG (plus associated RRset) and the DNSKEY. These are both obtained from the same zone. In the case of the KSK, the information is the DNSKEY and DS RRset with the latter being obtained from a different zone.
There are similarities in the rolling of ZSKs and KSKs: replace the RRSIG (plus RR) by the DNSKEY and replace the DNSKEY by the DS, and the ZSK rolling algorithms are virtually the same as the KSK algorithms. However, there are a number of differences, and for this reason the two types of rollovers are described separately in this document.

1.3. Terminology

TOC

The terminology used in this document is as defined in [RFC4033] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “DNS Security Introduction and Requirements,” March 2005.) and [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.).
A large number of symbols are used to identify times, intervals, etc. All are listed in Appendix A (List of Symbols).

---

2. Rollover Methods

TOC

TOC

2.1. ZSK Rollovers

A ZSK can be rolled in one of three ways:

*Pre-Publication. Described in [RFC4641] (Kolkman, O. and R. Gieben, "DNSSEC Operational Practices," September 2006.), 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 removed. During the re-signing process (which may or may not be atomic depending on how the zone is managed), 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.

*Double-Signature. Also mentioned in [RFC4641] (Kolkman, O. and R. Gieben, "DNSSEC Operational Practices," September 2006.), 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 (again, the signing process may not be atomic), 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-RRSIG. Strictly speaking, the use of the term "Double-Signature" above is a misnomer as the method is not only double signature, it is also double key as well. A true Double-Signature method (here called the Double-RRSIG method) involves introducing new signatures in the zone (while still retaining the old ones) but not the new key.

  Once the signing process is complete and enough time has elapsed to ensure that all caches that may contain an RR and associated

RRSIG to have a copy of both signatures, the ZSK is changed. After a further interval during which the old DNSKEY RRset expires from caches, the old signatures are removed from the zone.

Of three methods, Double-Signature is the simplest conceptually - introduce the new key and new signatures, then approximately one TTL later remove the old key and signatures. 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.
Pre-Publication is more complex - introduce the new key, approximately one TTL later sign the records, and approximately one TTL after that remove the old key. However, the amount of DNSSEC data is kept to a minimum which reduces the impact on performance.
The Double-RRSIG combines the increase in data volume of the Double-Signature method with the complexity of Pre-Publication. It has few (if any) advantages and a description is only included here for completeness.

2.2. KSK Rollovers

TOC

In the ZSK case the issue for the validating resolver is to ensure that it has access to the ZSK that corresponds to a particular signature. In the KSK case this can never be a problem as the KSK is only used for one signature (that over the DNSKEY RRset) and both the key the signature travel together. Instead, the issue is to ensure that the KSK is trusted.
Trust in the KSK is either due to the existence of an explicitly configured trust anchor in the validating resolver or DS record in the parent zone (which is itself trusted). If the former, [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.) timings will be needed to roll the keys. If the latter, the rollover algorithm will need to involve the parent zone in the addition and removal of DS records, so timings are not wholly under the control of the zone manager. (The zone manager may elect to include [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.) timings in the key rolling process so as to cope with the possibility that the key has also been explicitly configured as a trust anchor.)
It is important to note that this does not preclude the development of key rollover logic; in accordance with the goal of the rollover logic being able to determine when a state change is "safe", the only effect of being dependent on the parent is that there may be a period of waiting for the parent to respond in addition to any delay the key rollover logic requires. Although this introduces additional delays, even with a parent that is less than ideally responsive the only effect

will be a slowdown in the rollover state transitions. This may cause a policy violation, but will not cause any operational problems.
Like the ZSK case, there are three methods for rolling a KSK:

*Double-Signature: Also known as Double-DNSKEY, 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. (The name "Double-Signature" is used because, like the ZSK method of the same name, the new key is introduced and immediately used for signing.)

*Double-DS: 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 a further interval during which the old DNSKEY RRset expires from caches, the old DS record is removed.

*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.

In essence, "Double-Signature" means that the new KSK is introduced first and used to sign the DNSKEY RRset. The DS record is changed, and finally the old KSK removed. With "Double-DS" it is the other way around. Finally, Double-RRset does both updates more or less in parallel.
The strategies have different advantages and disadvantages:

*Double-Signature minimizes the number of interactions with the parent zone. However, for the period of the rollover the DNSKEY RRset is signed with two KSKs, so increasing the size of the response to a query for this data.

*Double-DS keeps the size of the DNSKEY RRset to a minimum, but does require the additional administrative overhead of two interactions with the parent to roll a KSK. (Although this can be mitigated if the parent has the ability for a child zone to schedule the withdrawal of the old key at the same time as the introduction of the new one.)

*Finally, Double-RRset allows the rollover to be done in roughly half the time of the other two methods; it also supports policies that require a period of running with old and new KSKs simultaneously. However, it does have the disadvantages of both the other two methods - it requires two signatures during the period of the rollover and two interactions with the parent.

2.3. Summary

TOC

The methods can be summarised as follows:

ZSK Method	KSK Method	Description
Pre-Publication	(not applicable)	Publish the DNSKEY before the RRSIG.
Double-Signature	Double-Signature	Publish the DNSKEY and RRSIG at same time. (For a KSK, this happens before the DS is published.)
Double-RRSIG	(not applicable)	Publish RRSIG before the DNSKEY.
(not applicable)	Double-DS	Publish DS before the DNSKEY.
(not applicable)	Double-RRset	Publish DNSKEY and DS in parallel.

Table 1

3. Key Rollover Timelines

TOC

3.1. Key States

TOC

During the rolling process, a key moves through different states. These states are:

Generated The key has been created, but has not yet been used for anything.

Published The DNSKEY record - or information associated with it - is published in the zone, but predecessors of the key (or associated information) may be held in resolver caches.

> The idea of "associated information" is used in rollover methods where RRSIG or DS records are published first and the DNSKEY is changed in an atomic operation. It allows the rollover still to be thought of as moving through a set of states. In the rest of this section, the term "key" should be taken to mean "key or associated information".
>
> **Ready** The key has been published for long enough to guarantee that all caches that might contain a copy of the key RRset have a copy that includes this key.
>
> **Active** The key is in the zone and has started to be used to sign RRsets or authenticate the DNSKEY RRset. Note that when this state is entered, it might not be possible for every validating resolver to use the key for validation: the zone signing may not have finished, or the data might not have reached the resolver because of propagation delays and/or caching issues. If this is the case, the resolver will have to rely on the key's predecessor instead.
>
> **Retired** The key is in the zone but a successor key has become active. As there may still be information in caches that that require use of the key, it is being retained until this information expires.
>
> **Dead** The key is published in the zone but there is no information anywhere that requires its presence.
>
> **Removed** The key has been removed from the zone.

There is one additional state, used where [RFC5011] (StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors," September 2007.) considerations are in effect (see Section 3.3.4 (Interaction with Configured Trust Anchors)):

> **Revoked** The key is published for a period with the "revoke" bit set as a way of notifying validating resolvers that have configured it as a trust anchor that it is about to be removed from the zone.

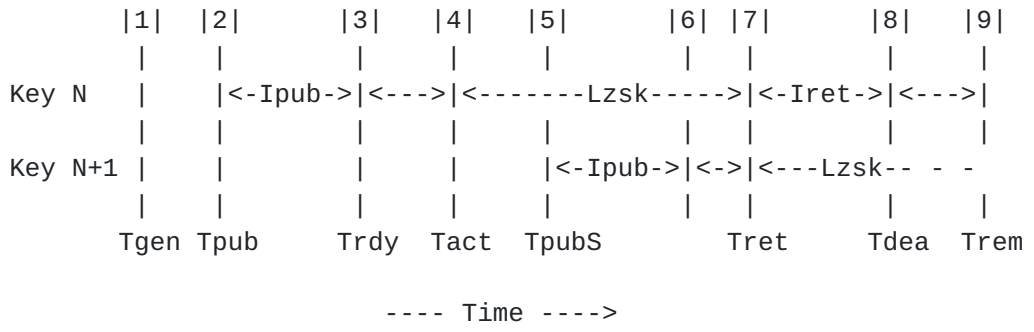
3.2. Zone-Signing Key Timelines

TOC

TOC

3.2.1. Pre-Publication Method

The following diagram shows the time line of a particular ZSK and its replacement by its successor using the Pre-Publication method. Time increases along the horizontal scale from left to right and the vertical lines indicate events in the life of the key. The events are numbered; significant times and time intervals are marked.



Event 1: key N is generated at the generate time (Tgen). Although there is no reason why the key cannot be generated immediately prior to publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.
Event 2: key N's DNSKEY record is put into the zone, i.e. it is added to the DNSKEY RRset which is then re-signed with the current key-signing key. The time at which this occurs is the key's publication time (Tpub), and the key is now said to be published. Note that the key is not yet used to sign records.
Event 3: before it can be used, the key must be published for long enough to guarantee that any resolver that has a copy of the DNSKEY RRset from the zone in its cache will have a copy of the RRset that includes this key: in other words, that any prior cached information about the DNSKEY RRset has expired.
This interval is the publication interval (Ipub) and, for the second or subsequent keys in the zone, is given by:

```
Ipub = Dprp + TTLkey
```

Here, Dprp is the propagation delay - the time taken for any change introduced at the master to replicate to all slave servers - which depends on the depth of the master-slave hierarchy. TTLkey is the time-to-live (TTL) for the DNSKEY records in the zone. The sum is therefore the time taken for existing DNSKEY records to expire from the caches of downstream validating resolvers, regardless of the nameserver from which they were retrieved.
In the case of the first key in the zone, Ipub is slightly different because it is not information about a DNSKEY RRset that may be cached, it is information about its absence. In this case:

```
Ipub = Dprp + Ingc
```

where Ingc is the negative cache interval from the zone's SOA record, calculated according to [RFC2308] (Andrews, M., “Negative Caching of DNS Queries (DNS NCACHE),” March 1998.) as the minimum of the TTL of the SOA record itself (TTLsoa), and the "minimum" field in the record's parameters (SOAmin), i.e.

```
Ingc = min(TTLsoa, SOAmin)
```

After a delay of Ipub, the key is said to be ready and could be used to sign records. The time at which this event occurs is the key's ready time (Trdy), which is given by:

```
Trdy = Tpub + Ipub
```

Event 4: at some later time, the key starts being used to sign RRsets. This point is the active time (Tact) and after this, the key is said to be active.
Event 5: while this key is active, thought must be given to its successor (key N+1). As with the introduction of the currently active key into the zone, the successor key will need to be published at least Ipub before it is used. Denoting the publication time of the successor key by TpubS, then:

```
TpubS <= Tact + Lzsk - Ipub
```

Here, Lzsk is the length of time for which a ZSK will be used (the ZSK lifetime). It should be noted that unlike the publication interval, Lzsk is not determined by timing logic, but by key management policy. Lzsk will be set by the operator according to their assessment of the risks posed by continuing to use a key and the risks associated with key rollover. However, operational considerations may mean a key is active for slightly more or less than Lzsk.
Event 6: while the key N is still active, its successor becomes ready. From this time onwards, it could be used to sign the zone.
Event 7: at some point the decision is made to 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, hence:

```
Tret = Tact + Lzsk
```

This point in time is the retire time (Tret) of key N; after this the key is said to be retired. (This time is also the point at which the successor key becomes active.)
Event 8: the retired key needs to be retained in the zone whilst any RRSIG records created using this key are still published in the zone or held in resolver caches. (It is possible that a resolver could have an unexpired RRSIG record and an expired DNSKEY RRset in the cache when it is asked to provide both to a client. In this case the DNSKEY RRset would need to be looked up again.) This means that once the key is no longer used to sign records, it should be retained in the zone for at least the retire interval (Iret) given by:

```
Iret = Dsgn + Dprp + TTLsig
```

Dsgn is the delay needed to ensure that all existing RRsets have been re-signed with the new key. Dprp is (as described above) the propagation delay, required to guarantee that the updated zone information has reached all slave servers, and TTLsig is the TTL of the RRSIG records.
(It should be noted that an upper limit on the retire interval is given by:

```
Iret = Lsig + Dskw
```

where Lsig is the lifetime of a signature (i.e. the interval between the time the signature was created and the signature end time), and Dskw is the clock skew - the maximum difference in time between the server and a validating resolver. The reasoning here is that whatever happens, a key only has to be available while any signatures created with it are valid. Wherever a signature record is held - published in the zone and/or held in a resolver cache - it won't be valid for longer than Lsig after it was created. The Dskw term is present to account for the fact that the signature end time is an absolute time rather than interval, and systems may not agree exactly about the time.)
The time at which all RRSIG records created with this key have expired from resolver caches is the dead time (Tdea), given by:

```
Tdea = Tret + Iret
```

...at which point the key is said to be dead.
Event 9: at any time after the key becomes dead, it can be removed from the zone and the DNSKEY RRset re-signed with the current key-signing key. This time is the removal time (Trem), given by:

```
Trem >= Tdea
```

...at which time the key is said to be removed.

3.2.2. Double-Signature Method

TOC

In the Double-Signature method, both the new key and signatures created using it are introduced at the same time. After a period during which this information propagates to validating resolver caches, the old key and signature are removed. The time line for the method is shown below:

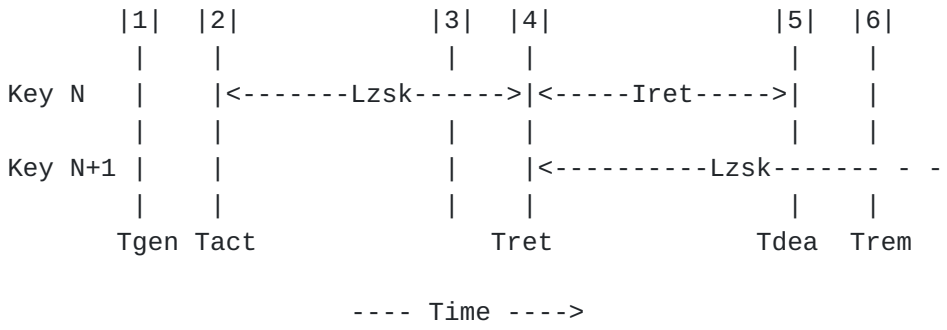


Figure 2: Timeline for a Double-Signature ZSK rollover.

Event 1: key N is generated at the generate time (Tgen). Although there is no reason why the key cannot be generated immediately prior to publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.
Event 2: key N is added to the DNSKEY RRset and is immediately used to sign the zone; existing signatures in the zone are not removed. This is the active time (Tact) and the key is said to be active.
Event 3: at some time later, the predecessor key (key N-1) and its signatures can be withdrawn from the zone. (The timing of key removal is discussed further in the description of event 5.)
Event 4: the successor key (key N+1) is introduced into the zone and starts being used to sign RRsets. The successor is key is now active and the current key (key N) is said to be retired. This time is the retire time of the key (Tret); it is also the active time of the successor key (TactS).

```
Tret = Tact + Lzsk
```

Event 5: before key N can be withdrawn from the zone, all RRsets that need to be signed must have been signed by the successor key (as a result, all these RRsets are now signed twice, once by key N and once by its successor) and the information must have reached all validating resolvers that may have RRsets from this zone cached.
This takes Iret, the retire interval, given by the expression:

```
Iret = Dsgn + Dprp + max(TTLkey, TTLsig)
```

As before, Dsgn is the time taken to sign the zone with the new key and Dprp is the propagation delay. The final term is the period to wait for old key and signature data to expire from caches. After the end of this interval, key N is said to be dead. This occurs at the dead time (Tdea) so:

```
Tdea = Tret + Iret
```

Event 6: at some later time key N and its signatures can be removed from the zone. This is the removal time Trem, given by:

```
Trem >= Tdea
```

3.2.3. Double-RRSIG Method

TOC

As noted above, "Double-Signature" is simultaneous rollover of both signature and key. The time line for a pure Double-Signature ZSK rollover (the Double-RRSIG method) - where new signatures are introduced, the key changed, and finally the old signatures removed - is shown below:

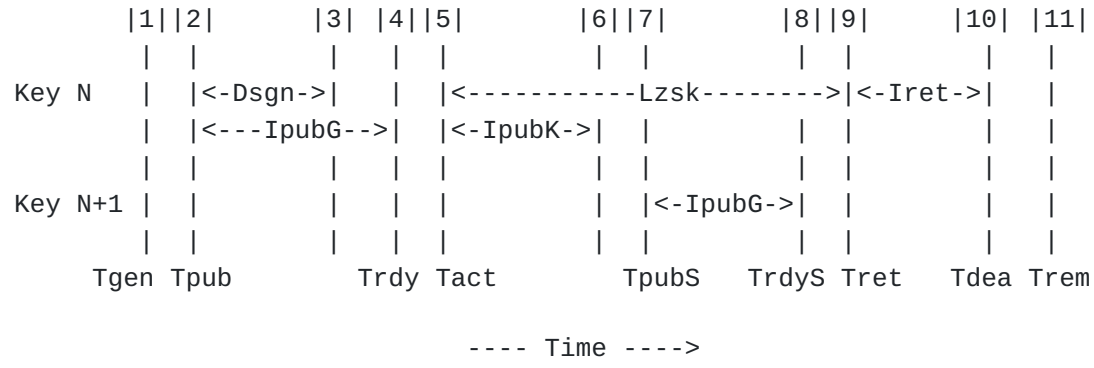


Figure 3: Timeline for a Double-Signature ZSK rollover.

Event 1: key N is generated at the generate time (Tgen). Although there is no reason why the key cannot be generated immediately prior to publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.
Event 2: key N is used to sign the zone but existing signatures are retained. Although the new ZSK is not published in the zone at this point, for analogy with the other ZSK rollover methods and because this is the first time that key information is visible (albeit indirectly through the created signatures) this time is called the publish time (Tpub).
Event 3: after the signing interval, Dsgn, all RRsets that need to be signed have been signed by the new key. (As a result, all these RRsets are now signed twice, once by the existing key and once by the (still-absent) key N.
Event 4: there is now a delay while the this information reaches all validating resolvers that may have RRsets from the zone cached. This interval is given by the expression:

```
Dprp + TTLsig
```

...comprising the delay for the information to propagate through the nameserver infrastructure plus the time taken for signature information to expire from caches.
Again in analogy with other key rollover methods, this is defined as key N's ready time (Trdy) and the key is said to be in the ready state. (Although the key is not in the zone, it is ready to be used.) The interval between the publication and ready times is the publication interval of the signature, IpubG, i.e.

```
Trdy = Tpub + IpubG
```

where

```
IpubG = Dsgn + Dprp + TTLsig
```

Event 5: at some later time the predecessor key is removed and the key N added to the DNSKEY RRset. As all the RRs have signatures created by the old and new keys, the records can still be authenticated. This time is the active time (Tact) and the key is now said to be active.
Event 6: After IpubK - the publication interval of the key - the newly added DNSKEY RRset has propagated through to all validating resolvers. At this point the old signatures can be removed from the zone. IpubK is given by:

```
IpubK = Dprp + TTLkey
```

Event 7: as before, at some later time thought must be given to rolling the key. The first step is to publish signatures created by the successor key (key N+1) early enough so that key N can be replaced after it has been active for its scheduled lifetime. This occurs at TpubS (the publication time of the successor), given by:

```
TpubS <= Tact + Lzsk - IpubG
```

Event 8: the signatures have propagated through the zone and the new key could be added to the zone. This time is the ready time of the successor (TrdyS).

```
TrdyS = TpubS + IpubG
```

... where IpubG is as defined above.
Event 9: at some later time key N is removed from the zone and the successor key added. This is the retire time of the key (Tret).
Event 10: The signatures must remain in the zone for long enough that the new DNSKEY RRset has had enough time to propagate to all caches. Once caches contain the new DNSKEY, the old signatures are no longer of use and can be considered to be dead. The time at which this occurs is the dead time (Tdea), given by:

```
Tdea = Tret + Iret
```

... where Iret is the retire interval, given by:

```
Iret = IpubK
```

Event 11: At some later time the signatures can be removed from the zone. Although the key has is not longer in the zone, this is information associated with it and so the time can be regarded as the key's remove time (Trem), given by:

```
Trem >= Tdea
```

3.3. Key-Signing Key Rollover Timelines

TOC

TOC

3.3.1. Double-Signature Method

The Double-Signature method (also knows as the double-DNSKEY method) involves introducing the new KSK to the zone and waiting until its presence has been registered by all validating resolvers. At this point, the DS record in the parent is changed. Once that change has propagated to all validating resolvers, the old KSK is removed.
The timing diagram for such a rollover is:

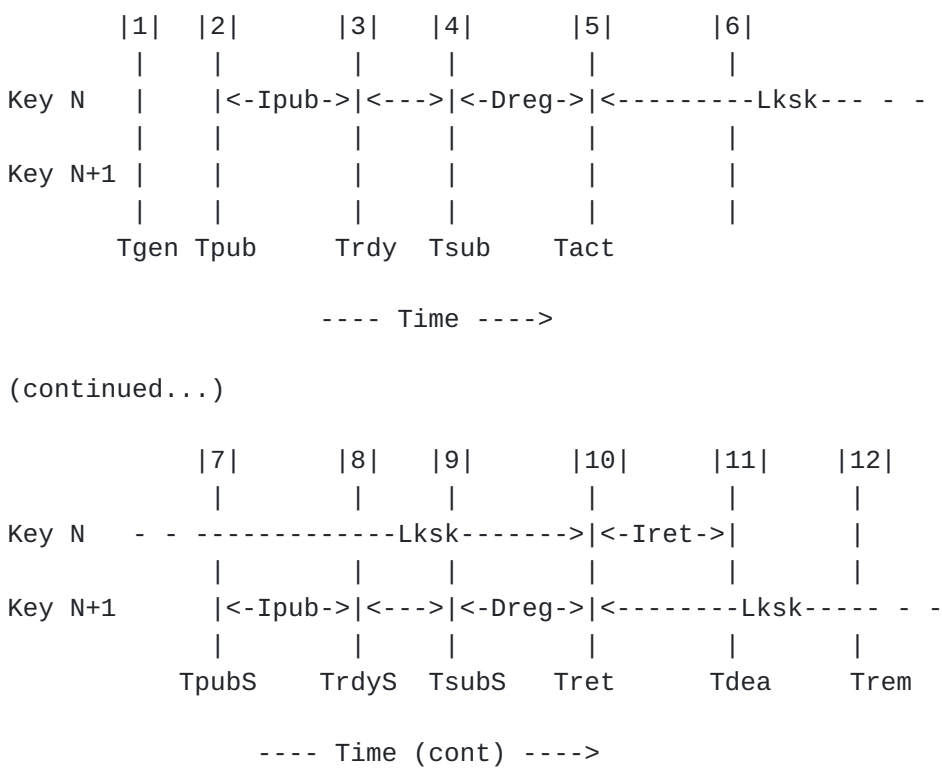


Figure 4: Timeline for a Double-Signature KSK rollover.

Event 1: key N is generated at time Tgen. As before, although there is no reason why the key cannot be generated immediately prior to publication, some implementations may find it convenient to create a central pool of keys and draw from it. For this reason, it is again shown as a separate event.
Event 2: key N is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by key N and all currently active KSKs. (So at this point, the DNSKEY RRset is signed by both key N and its predecessor KSK. If other KSKs were active, it is signed by these as

well.) This is the publication time (Tpub); after this the key is said to be published.
Event 3: before it can be used, the key must be published for long enough to guarantee that any validating resolver that has a copy of the DNSKEY RRset from the zone in its cache will have a copy of the RRset that includes this key: in other words, that any prior cached information about the DNSKEY RRset has expired.
The interval is the publication interval (Ipub) and, for the second or subsequent KSKs in the zone, is given by:

```
Ipub = Dprp + TTLkey
```

... where Dprp is the propagation delay for the zone and TTLkey the TTL for the DNSKEY RRset. The time at which this occurs is the key's ready time, Trdy, given by:

```
Trdy = Tpub + Ipub
```

Event 4: at some later time, the DS RR corresponding to the new KSK is submitted to the parent zone for publication. This time is the submission time, Tsub.
Event 5: the DS record is published in the parent zone. As this is the point at which all information for authentication - both DNSKEY and DS record - is available in the two zones, it is the active time of the key:

```
Tact = Tsub + Dreg
```

... where Dreg is the registration delay, the time taken after the DS record has been received by the parent zone manager for it to be placed in the zone. (Parent zones are often managed by different entities, and this term accounts of the organisational overhead of transferring a record.)
Event 6: at some time later, all validating resolvers that have the DS RRset cached will have a copy that includes the new DS record. For the second or subsequent DS records, this interval is given by the expression:

```
DprpP + TTLds
```

... where DprpP is the propagation delay in the parent zone and TTLds the TTL assigned to DS records in that zone.
In the case of the first DS record for the zone in question, the expression is slightly different because it is not information about a DS RRset that may be cached, it is information about its absence. In this case, the interval is:

```
DprpP + IngcP
```

where IngcP is the negative cache interval from the zone's SOA record, calculated according to [RFC2308] (Andrews, M., “Negative Caching of DNS Queries (DNS NCACHE),” March 1998.) as the minimum of the TTL of the parent SOA record itself (TTLsoaP), and the "minimum" field in the record's parameters (SOAminP), i.e.

```
IngcP = min(TTLsoaP, SOAminP)
```

Event 7: while key N is active, thought needs to be given to its successor (key N+1). At some time before the scheduled end of the KSK lifetime, the successor KSK is introduced into the zone and is used to sign the DNSKEY RRset. (As before, this means that the DNSKEY RRset is signed by both the current and successor KSK.) This is the publication time of the successor key, TpubS.
Event 8: after an interval Ipub, the successor key becomes ready (in that all validating resolvers that have a copy of the DNSKEY RRset have a copy of this key). This is the successor ready time, TrdyS.
Event 9: at the successor submission time (TsubS), the DS record corresponding to the successor key is submitted to the parent zone.
Event 10: the successor DS record is published in the parent zone and the current DS record withdrawn. The current key is said to be retired and the time at which this occurs is Tret, given by:
The relationships between these times are:

```
TpubS <= Tact + Lksk - Dreg - Ipub

Tret = Tact + Lksk
```

... where Lksk is the scheduled lifetime of the KSK.
Event 11: key N must remain in the zone until any validators that have the DS RRset cached have a copy of the DS RRset containing the new DS record. This interval is the retire interval, given by:

```
Iret = DprpP + TTLds
```

... where DprpP is the propagation delay in the parent zone and TTLds the TTL of a DS record.
As the key is no longer used for anything, it can also be said to be dead, in which case:

```
Tdea = Tret + Iret
```

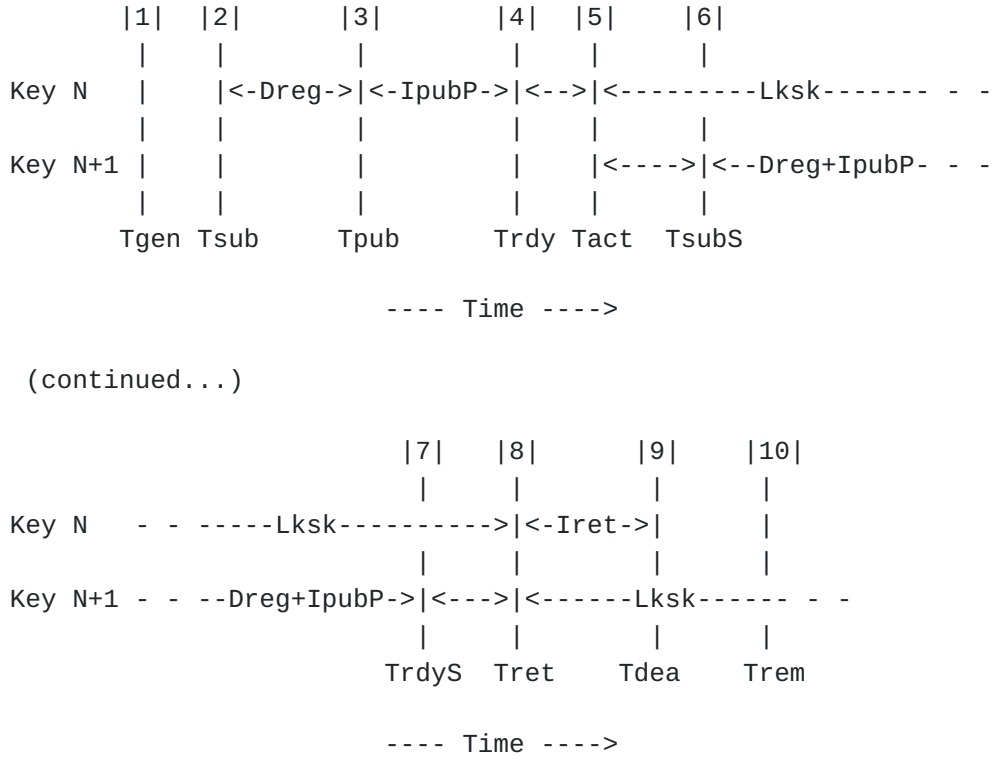
Event 12: at some later time, key N is removed from the zone (at the remove time Trem); the key is now said to be removed.

```
Trem >= Tdea
```

3.3.2. Double-DS Method

TOC

The Double-DS method is the reverse of the Double-Signature method is that it is the DS record that is pre-published (in the parent), and not the DNSKEY.
The timeline for the key rollover is shown below:



(continued...)

Figure 5: Timeline for a Double-DS KSK rollover.

Event 1: key N is generated at time Tgen. As before, although there is no reason why the key cannot be generated immediately prior to publication, some implementations may find it convenient to create a central pool of keys and draw from it. For this reason, it is again shown as a separate event.
Event 2: the DS record corresponding to key N is submitted for publication in the parent zone. This time is the submission time (Tsub).
Event 3: after the registration delay, Dreg, the DS record is published in the parent zone. This is the publication time Tpub, given by:

```
Tpub = Tsub + Dreg
```

Event 4: at some later time, any validating resolver that has copies of the DS RRset in its cache will have a copy of the DS record for key N. At this point, key N, if introduced into the DNSKEY RRset, could be used to validate the zone. For this reason, this time is known as the key's ready time, Trdy, and is given by:

```
Trdy = Tpub + IpubP
```

IpubP is the parent publication interval and is given by the expression:

```
IpubP = DprpP + TTLds
```

... where DprpP is the propagation delay in the parent zone and TTLds the TTL assigned to DS records in that zone.
Event 5: at some later time, the key rollover takes place. The predecessor key is withdrawn from the DNSKEY RRset and the new key (key N) introduced and used to sign the RRset.
As both DS records have been in the parent zone long enough to ensure that they are in the cache of any validating resolvers that have the DS RRset cached, the zone can be authenticated throughout the rollover - either the resolver has a copy of the DNSKEY RRset (and associated RRSIGs) authenticated by the predecessor key, or it has a copy of the updated RRset authenticated with the new key.
This time is the key's active time (Tact) and at this point the key is said to be active.
Event 6: at some point thought must be given to key replacement. The DS record for the successor key must be submitted to the parent zone at a time such that when the current key is withdrawn, any validating resolver that has DS records in its cache will have data about the DS record of the successor key. The time at which this occurs is the submission time of the successor, given by:

```
TsubS <= Tact + Lksk - IpubP - Dreg
```

... where Lksk is the lifetime of the KSK.
Event 7: the successor key (key N+1) enters the ready state i.e. its DS record is now in the caches of all validating resolvers that have the parent DS RRset cached. (This is the ready time of the successor, TrdyS.)
Event 8: when the current key has been active for its lifetime (Lksk), the current key is removed from the DNSKEY RRset and the successor key added; the RRset is then signed with the successor key. This point is the retire time of the key, Tret, given by:

```
Tret = Tact + Lksk
```

Event 9: at some later time, all copies of the old DNSKEY RRset have expired from caches and the old DS record is no longer needed. This is called the dead time, Tdea, and is given by:

```
Tdea = Tret + Iret
```

... where Iret is the retire interval, given by:

```
Iret = Dprp + TTLkey
```

As before, this term includes the time taken to propagate the RRset change through the master-slave hierarchy and the time take for the DNSKEY RRset to expire from caches.
Event 10: at some later time, the DS record is removed from the parent zone. This is the removal time (Trem), given by:

```
Trem >= Tdea
```

3.3.3. Double-RRset Method

TOC

In the Double-RRset method, both the DS and DNSKEY records are changed at the same time, so for a period the zone can be authenticated with either key. The advantage of this method is its applicability in cases where zone management policy requires overlap of authentication keys during a roll.
The timeline for this rollover is shown below:

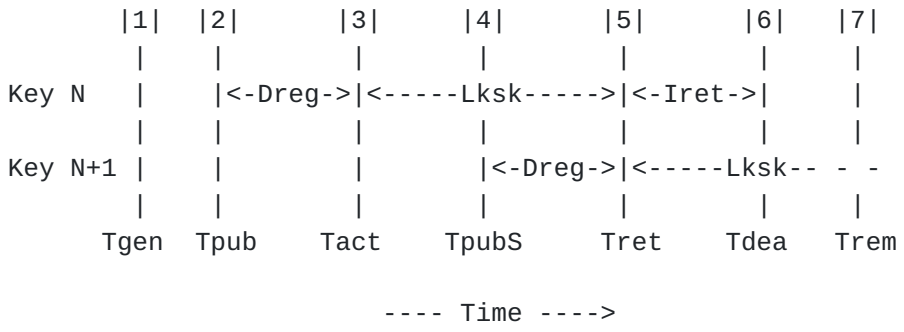


Figure 6: Timeline for a Double-RRset KSK rollover.

Event 1: key N is created at time Tgen and thereby immediately becomes generated. As before, although there is no reason why the key cannot be generated immediately prior to publication, some implementations may find it convenient to create a central pool of keys and draw from it. For this reason, it is again shown as a separate event.
Event 2: the key is added to and used for signing the DNSKEY RRset and is thereby published in the zone. At the same time the corresponding DS record is submitted to the parent zone for publication. This time is the publish time (Tpub) and the key is now said to be published.
Event 3: after Dreg, the registration delay, the DS record is published in the parent zone. At this point, the zones have all the information needed for a validating resolver to authenticate the zone, although the information may not yet have reached all validating resolver caches. This time is the active time (Tact) and the key is said to be active.

```
Tact = Tpub + Dreg
```

Event 4: at some point we need to give thought to key replacement. The successor key must be introduced into the zone (and its DS record submitted to the parent) at a time such that it becomes active when the current key has been active for its lifetime, Lksk. This time is TpubS, the publication time of the successor key, and is given by:

```
TpubS <= Tact + Lksk - Dreg
```

... where Lksk is the lifetime of the KSK.
Event 5: the successor key's DS record appears in the parent zone and the successor key becomes active. At this point, the current key becomes retired. This occurs at Tret, given by:

```
Tret = Tact + Lksk
```

Event 6: the current DNSKEY and DS record must be retained in the zones until any any validating resolver that has cached the DNSKEY and/or DS RRsets will have a copy of the data for the successor key. At this point the current key information is dead, as any validating resolver can perform authentication using the successor key. This is the dead time, Tdea, given by:

```
Tdea = Tret + Iret
```

... where Iret is the retire interval. This depends on how long both the successor DNSKEY and DS records take to propagate through the nameserver infrastructure and thence into validator caches. These delays are the publication intervals of the child and parent zones respectively, so a suitable expression for Iret is:

```
Iret = max(IpubP, IpubC)
```

IpubC is the publication interval of the DNSKEY in the child zone, IpubP that of the DS record in the parent.
The child term comprises two parts - the time taken for the introduction of the DNSKEY record to be propagated to the downstream secondary servers (= DprpC, the child propagation delay) and the time taken for information about the DNSKEY RRset to expire from the validating resolver cache, i.e.

```
IpubC = DprpC + TTLkey
```

TTLkey is the TTL for a DNSKEY record in the child zone. The parent term is similar:

```
IpubP = DprpP + TTLds
```

DprpP the propagation delay in the parent zone and TTLds the TTL for a DS record in the parent zone.
Event 7: at some later time, the DNSKEY record can be removed from the child zone and a request can be made to remove the DS record from the parent zone. This is the removal time, Trem and is given by:

```
Trem >= Tdea
```

3.3.4. Interaction with Configured Trust Anchors

TOC

Although the preceding sections have been concerned with rolling KSKs where the trust anchor is a DS record in the parent zone, zone managers may want to take account of the possibility that some validating resolvers may have configured trust anchors directly.
Rolling a configured trust anchor is dealt with in [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.). It requires introducing the KSK to be used as the trust anchor into the zone for a period of time before use, and retaining it (with the "revoke" bit set) for some time after use. The Double-Signature and Double-RRset methods can be adapted to include [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.) recommendations so that the rollover will also be signalled to validating resolvers with configured trust anchors. (The recommendations are not suitable for the Double-DS method. Introducing the new key early and retaining the old key after use effectively converts it into a form of Double-RRset.)

TOC

3.3.4.1. Addition of KSK

When the new key is introduced, the publication interval (Ipub) in the Double-Signature method should also be subject to the condition:

> Ipub >= max(30 days, TTLkey)

... where the right had side of the expression is the add hold-down time defined in section 2.4.1 of [RFC5011] (StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors," September 2007.).
In the Double-RRSIG method, the key should not be regarded as being active until the add hold-down time has passed. In other words, the following condition should be enforced:

> Tact >= Tpub + max(30 days, TTLkey)

(Effectively, this means extending the lifetime of the key by an appropriate amount.)

3.3.4.2. Removal of KSK

TOC

The timeline for the removal of the key in both methods is modified by introducing a new state, "revoked". When the key reaches the end of the retire period, instead of being declared "dead", it is revoked; the "revoke" bit is set on the DNSKEY RR and is published in (and used to sign) the DNSKEY RRset. The key is maintained in this state for the "revoke" interval, Irev, given by:

> Irev >= 30 days

... 30 days being the [RFC5011] (StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors," September 2007.) remove hold-down time. After this time, the key is dead and can be removed from the zone when convenient.

3.3.5. Introduction of First KSK

TOC

There is an additional consideration when introducing a KSK into a zone for the first time, and that is that no validating resolver should be in a position where it can access the trust anchor before the KSK appears in the zone. To do so will cause the validating resolver to declare the zone to be bogus.
This is important: in the case of a secure parent, it means ensuring that the DS record is not published in the parent zone until there is

no possibility that a validating resolver can obtain the record yet not be able to obtain the corresponding DNSKEY. In the case of an insecure parent, i.e. the initial creation of a new security apex, it is important to not configure trust anchors in validating resolvers until the DNSKEY RRset has had sufficient time to propagate. In both cases, this time is the trust anchor availability time (Ttaa) given by:

```
Ttaa >= Tpub + IpubC
```

where

```
IpubC = DprpC + TTLkeyC
```

or

```
IpubC = DprpC + IngcC
```

The first expression applies if there was previously a DNSKEY RRset in the child zone, the expression for IpubC including the TTLkeyC term to account for the time taken for that RRset to expire from caches. (It is possible that the zone was signed but that the trust anchor had not been submitted to the parent.)
If the introduction of the KSK caused the appearance of the first DNSKEY RRset in the child zone, the second expression applies in which the TTLkeyC term is replaced by Ingc to allow for the effect of negative caching.
As before, IngcC is the negative cache interval from the child zone's SOA record, calculated according to [RFC2308] (Andrews, M., “Negative Caching of DNS Queries (DNS NCACHE),” March 1998.) as the minimum of the TTL of the SOA record itself (TTLsoaC), and the "minimum" field in the record's parameters (SOAminC), i.e.

```
IngcC = min(TTLsoaC, SOAminC)
```

---

4. Standby Keys

TOC

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 in the ready state to sign the zone, having at least one additional key (a standby key) in this state at all times will minimise delay.
In the case of a ZSK, a standby key only really makes sense with the Pre-Publication method. A permanent standby DNSKEY RR should be included in zone or successor keys could be introduced as soon as

possible after a key becomes active. Either way results in an additional ZSK in the DNSKEY RRset that can immediately be used to sign the zone if the current key is compromised.
(Although in theory the mechanism could be used with both the Double-Signature and Double-RRSIG methods, it would require Pre-Publication of the signatures. Essentially, the standby key would be permanently active, as it would have to be periodically used to renew signatures. Zones would also permanently require two sets of signatures, something that could have a performance impact in large zones.)
A standby key can also be used with the Double-Signature and Double-DS methods of rolling a KSK. (The idea of a standby key in the Double-RRset effectively means having two active keys.) The Double-Signature method requires that the standby KSK be included in the DNSKEY RRset; rolling the key then requires just the introduction of the DS record in the parent. (Note that the DNSKEY should also be used to sign the DNSKEY RRset. As the RRset and its signatures travel together, merely adding the DNSKEY does not provide the desired time saving; to be used in a rollover requires that the DNSKEY RRset be signed with the standby key, and this introduces a delay whilst the RRset and its signatures propagate to the caches of validating resolvers. There is no time advantage over introducing a new DNSKEY and signing the RRset with it at the same time.)
In the Double-DS method of rolling a KSK, it is not a standby key that is present, it is a standby DS record in the parent zone. Whatever algorithm is used, the standby item of data can be introduced as a permanent standby, or be a successor introduced as early as possible.

---

5. Algorithm Considerations

TOC

The preceding sections have implicitly assumed that all keys and signatures are created using a single algorithm. However, [RFC4035] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “Protocol Modifications for the DNS Security Extensions,” March 2005.) (section 2.4) states that "There MUST be an RRSIG for each RRset using at least one DNSKEY of each algorithm in the zone apex DNSKEY RRset". Except in the case of an algorithm rollover - where the algorithms used to create the signatures are being changed - there is no relationship between the keys of different algorithms. This means that they can be rolled independently of one another. In other words, the key rollover logic described above should be run separately for each algorithm; the union of the results is included in the zone, which is signed using the active key for each algorithm.

TOC

6. Limitation of Scope

This document represents current thinking at the time of publication. However, the subject matter is evolving and it is more than likely that this document will need to be revised in the future.
Some of the techniques and ideas that DNSSEC operators are thinking about differ from this those described in this document. Of note are alternatives to the strict split into KSK and ZSK key roles and the consequences for rollover logic from partial signing (i.e. when the new key initially only signs a fraction of the zone while leaving other signatures generated by the old key in place).
Furthermore, as noted in section 5, this document covers only rolling keys of the same algorithm, it does not cover transition to/from/ addition/deletion of different algorithm. Algorithm rollovers will require a separate document.
The reader is therefore reminded that DNSSEC is as of publication in early stages of deployment, and best practices will likely change in the near term.

---

7. Summary

TOC

For ZSKs, "Pre-Publication" is generally considered to be the preferred way of rolling keys. As shown in this document, the time taken to roll is wholly dependent on parameters under the control of the zone manager.
In contrast, "Double-RRset" is the most efficient method for KSK rollover due to the ability to have new DS records and DNSKEY RRsets propagate in parallel. The time taken to roll KSKs may depend on factors related to the parent zone if the parent is signed. For zones that intend to comply with the recommendations of [RFC5011] (StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.), in virtually all cases the rollover time will be determined by the RFC5011 "add hold-down" and "remove hold-down" times. It should be emphasized that this delay is a policy choice and not a function of timing values and that it also requires changes to the rollover process due to the need to manage revocation of trust anchors. Finally, the treatment of emergency key rollover is significantly simplified by the introduction of stand-by keys as standard practice during all types of rollovers.

---

8. IANA Considerations

TOC

This memo includes no request to IANA.

9. Security Considerations

TOC

This document does not introduce any new security issues beyond those already discussed in [RFC4033] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements," March 2005.), [RFC4034] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions," March 2005.), [RFC4035] (Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions," March 2005.) and [RFC5011] (StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors," September 2007.).

10. Acknowledgements

TOC

The authors gratefully acknowledge help and contributions from Roy Arends and Wouter Wijngaards.

11. Change History

TOC

*draft-ietf-dnsop-dnssec-key-timing-00
 * Added section on limitation of scope.

*draft-morris-dnsop-dnssec-key-timing-02
 * General restructuring.
 * Added descriptions of more rollovers (IETF-76 meeting).
 * Improved description of key states and removed diagram.
 * Provided simpler description of standby keys.
 * Added section concerning first key in a zone.
 * Moved [RFC5011] (StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors," September 2007.) to a separate section.
 * Various nits fixed (Alfred Hoenes, Jeremy Reed, Scott Rose, Sion Lloyd, Tony Finch).

*draft-morris-dnsop-dnssec-key-timing-01
 * Use latest boilerplate for IPR text.
 * List different ways to roll a KSK (acknowledgements to Mark Andrews).
 * Restructure to concentrate on key timing, not management procedures.
 * Change symbol notation (Diane Davidowicz and others).
 * Added key state transition diagram (Diane Davidowicz).

* Corrected spelling, formatting, grammatical and style errors (Diane Davidowicz, Alfred Hoenes and Jinmei Tatuya).
* Added note that in the case of multiple algorithms, the signatures and rollovers for each algorithm can be considered as more or less independent (Alfred Hoenes).
* Take account of the fact that signing a zone is not atomic (Chris Thompson).
* Add section contrasting pre-publication rollover with double signature rollover (Matthijs Mekking).
* Retained distinction between first and subsequent keys in definition of initial publication interval (Matthijs Mekking).

*draft-morris-dnsop-dnssec-key-timing-00
Initial draft.

---

12. References

TOC

12.1. Normative References

TOC

[RFC2308]	Andrews, M., “Negative Caching of DNS Queries (DNS NCACHE),” RFC 2308, March 1998 (TXT, HTML, XML).
[RFC4033]	Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “DNS Security Introduction and Requirements,” RFC 4033, March 2005 (TXT).
[RFC4034]	Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “Resource Records for the DNS Security Extensions,” RFC 4034, March 2005 (TXT).
[RFC4035]	Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, “Protocol Modifications for the DNS Security Extensions,” RFC 4035, March 2005 (TXT).
[RFC5011]	StJohns, M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” RFC 5011, September 2007 (TXT).

12.2. Informative References

TOC

[RFC4641]	Kolkman, O. and R. Gieben, “DNSSEC Operational Practices,” RFC 4641, September 2006 (TXT).

TOC

Appendix A. List of Symbols

The document defines a number of symbols, all of which are listed here.
All are of the form:
All symbols used in the text are of the form:

```
<TYPE><id><INST>
```

where:
<TYPE> is an upper-case character indicating what type the symbol is.
Defined types are:

D delay: interval that is a feature of the process

I interval between two events

L lifetime: interval set by the zone manager

SOA parameter related to SOA RR

T a point in time

TTL TTL of a record

T and I are self-explanatory. D, and L are also time periods, but whereas I values are intervals between two events (even if the events are defined in terms of the interval, e.g. the dead time occurs "retire interval" after the retire time), D, and L are fixed intervals. An "L" interval (lifetime) is chosen by the zone manager and is a feature of policy. A "D" interval (delay) is a feature of the process, probably outside control of the zone manager. SOA and TTL are used just because they are common terms.
<id> is lower-case and defines what object or event the variable is related to, e.g.

act active

ngc negative cache

pub publication

Finally, <INST> is a capital letter that distinguishes between the same variable applying to different instances of an object and is one of:

C child

G signature

K key

P
parent

S successor

The list of variables used in the text is:

Dprp Propagation delay. The amount of time for a change made at a master nameserver to propagate to all the slave nameservers.

DprpC Propagation delay in the child zone.

DprpP Propagation delay in the parent zone.

Dreg Registration delay. As a parent zone is often managed by a different organisation to that managing the child zone, the delays associated with passing data between zones is captured by this term.

Dskw Clock skew. The maximum difference in time between the signing system and the resolver.

Dsgn Signing delay. After the introduction of a new ZSK, the amount of time taken for all the RRs in the zone to be signed with it.

Ingc Negative cache interval.

IngcP Negative cache interval of the child zone.

IngcP Negative cache interval of the parent zone.

Ipub Publication interval. The amount of time that must elapse after the publication of a key before it can be considered to have entered the ready state.

IpubC Publication interval in the child zone.

IpubG Publication interval for the signature.

IpubK Publication interval for the key.

IpubP Publication interval in the parent zone.

Iret Retire interval. The amount of time that must elapse after a key enters the retire state for any signatures created with it to be purged from validating resolver caches.

Irev Revoke interval. The amount of time that a KSK must remain published with the revoke bit set to satisfy [RFC5011] (StJohns,

M., “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” September 2007.) considerations.

Lksk Lifetime of a key-signing key. This is the intended amount of time for which this particular KSK is regarded as the active KSK. Depending on when the key is rolled-over, the actual lifetime may be longer or shorter than this.

Lzsk Lifetime of a zone-signing key. This is the intended amount of time for which the ZSK is used to sign the zone. Depending on when the key is rolled-over, the actual lifetime may be longer or shorter than this.

Lsig Lifetime of a signature: the difference in time between the signature's expiration time and the time at which the signature was created. (Note that this is not the difference between the signature's expiration and inception times: the latter is usually set a small amount of time before the signature is created to allow for clock skew between the signing system and the validating resolver.)

SOAmin Value of the "minimum" field from an SOA record.

SOAminC Value of the "minimum" field from an SOA record in the child zone.

SOAminP Value of the "minimum" field from an SOA record in the parent zone.

Tact Active time of the key; the time at which the key is regarded as the principal key for the zone.

TactS Active time of the successor key.

Tdea Dead time of a key. Applicable only to ZSKs, this is the time at which any record signatures held in validating resolver caches are guaranteed to be created with the successor key.

Tgen Generate time of a key. The time that a key is created.

Tpub Publish time of a key. The time that a key appears in a zone for the first time.

TpubS Publish time of the successor key.

Trem Removal time of a key. The time at which a key is removed from the zone.

Tret Retire time of a key. The time at which a successor key starts being used to sign the zone.

Trdy
: Ready time of a key. The time at which it can be guaranteed that validating resolvers that have key information from this zone cached have a copy of this key in their cache. (In the case of KSKs, should the validating resolvers also have DS information from the parent zone cached, the cache must include information about the DS record corresponding to the key.)

TrdyS Ready time of a successor key.

Tsub Submit time - the time at which the DS record of a KSK is submitted to the parent.

TsubS Submit time of the successor key.

TTLds Time to live of a DS record (in the parent zone).

TTLkey Time to live of a DNSKEY record.

TTLkeyC Time to live of a DNSKEY record in the child zone.

TTLsoa Time to live of a SOA record.

TTLsoaC Time to live of a SOA record in the child zone.

TTLsoaP Time to live of a SOA record in the parent zone.

TTLsig Time to live of an RRSIG record.

Ttaa Trust anchor availability time. The time at which a trust anchor record can be made available when a KSK is first introduced into a zone.

Authors' Addresses

TOC

	Stephen Morris
	Internet Systems Consortium
	950 Charter Street
	Redwood City, CA 94063
	USA
Phone:	+1 650 423 1300
Email:	stephen@isc.org
	Johan Ihren
	Netnod
	Franzengatan 5
	Stockholm, SE-112 51

	Sweden
Phone:	+46 8615 8573
Email:	johani@autonomica.se
	John Dickinson
	Sinodun Internet Technologies Ltd
	Stables 4 Suite 11, Howbery Park
	Wallingford, Oxfordshire OX10 8BA
	UK
Phone:	+44 1491 818120
Email:	jad@sinodun.com