Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 18, 2010

S. Morris
Nominet
J. Ihren
Autonomica
J. Dickinson
October 15, 2009

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

Status of this Memo

Copyright Notice

Abstract

This document describes the issues surrounding the timing of events

in the rolling of a key in a DNSSEC-secured zone. It presents timlines for the key rollover and explicitly identifies the relationships between the various parameters affecting the process.

Table of Contents

1. Introduction

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 reasons. In order to implement a key rollover, the keys need to introduced into and removed from the zone at the appropriate times. Considerations that must be taken into account are:

- Key and signature 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 validator may try to validate a signature retrieved from a cache with a key obtained separately.

- 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.

- 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.

Management policy, e.g. how long a key is used for, also needs to be taken into account. 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. In other words, although key management logic enforces policy, it may not enforce it strictly.

1.1. Terminology

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

A large number of symbols are used in this document to identify times, intervals, etc. All are listed in Appendix A.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Types of Key Rollover

As noted in the introduction, client resolvers may cache both key and signature RRsets. This means that when validating a signature record (or passing both RRsets to a client who has issued a query with the CD bit set), an RRSIG just read from an authoritative server may be paired with a cached DNSKEY or vice-versa. For the validation to be successful, the DNSKEY and RRSIG must be consistent.

Key rollover - the replacement of the key use to sign the zone with another - involves changing the contents of the DNSKEY RRset and re-signing the zone (so changing the RRSIG records). In order for a RR to be validated, at least one RRSIG in the associated signature RRset must be able to be validated by one of the keys in the DNSKEY RRset. To ensure uninterrupted security, the aim must be to ensure that this condition is true at all stages during the rollover process.

Two ways to achieve this goal are the pre-publication method and the double signature method.

2.1. Pre-Publication Method

In 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 resolvers contain both keys. At that point, the zone can be signed 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 will have a copy of the DNSKEY RRset 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, the old key can be removed from the DNSKEY RRset because there will be no signatures anywhere created using it.

2.2. Double-Signature Method

Double-signature, as the name implies, 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 and RRSIG RRsets allows at least one signature to be validated.

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

2.3. Comparison of Rollover Methods

Of the two methods, double-signature is the simplest conceptually - introduce the new key and new signatures, then (roughly) 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. This

affects both the overall size of the zone and the size of the responses.

Pre-publication is more complex - introduce the new key, one TTL later sign the records, and one TTL after that remove the old key. However, the amount of DNSSEC data is kept to a minimum, hence reducing the impact on performance.

2.4. Timing Considerations

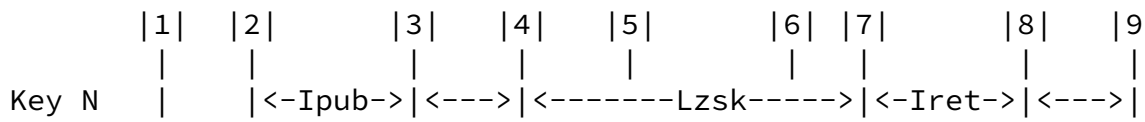
The rest of this paper describes the timing considerations related to the rolling of zone-signing keys (ZSKs) and key-signing keys (KSKs). Owing to the increase in the amount of DNSSEC data in the double-signature method, the pre-publication approach is preferred for rollover of ZSKs. However, in the case of KSK rollovers, the size increase is negligible and hence the complexity of pre-publication is not justified.

While this combination is the most common choice of rollover logic, there is nothing to preclude other combinations should the situation demand it. The rest of this document describes ZSK and KSK rollover timelines for the common case.

3. Zone-Signing Keys

3.1. Key Timeline

The following diagram shows the time line of a particular ZSK (zone-signing key) and its replacement by its successor. 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.



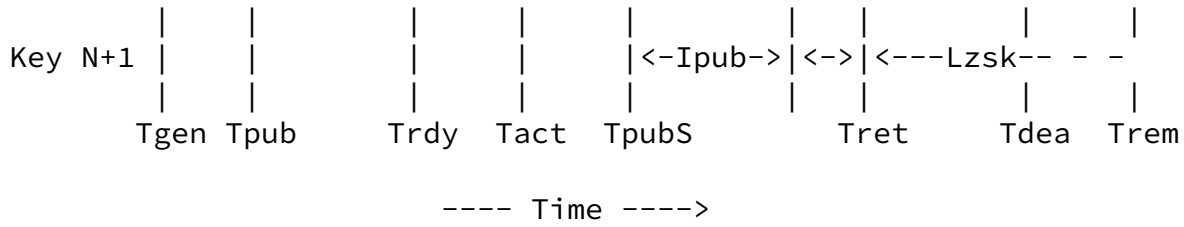


Figure 1: Timeline for a ZSK rollover.

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'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 key N 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.

The 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 take 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 validators, 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] 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 can 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 activation time (Tact) and after this, the key is said to be in the active state.

Event 5: while this key is active, thought must be given to its successor. 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 lives for slightly more or less than

Lzsk.

Event 6: while the current ZSK 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 validator. 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 expire from resolver caches is the dead time (Tdea), given by:

Tdea = Tret + Iret

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

...and the key is said to be in the removed state.

3.2. Key States

An alternative way of considering the key timeline is to regard the key as moving through a set of states, the state transitions being determined by time. The state transition diagram is linear and is shown in Figure 2:

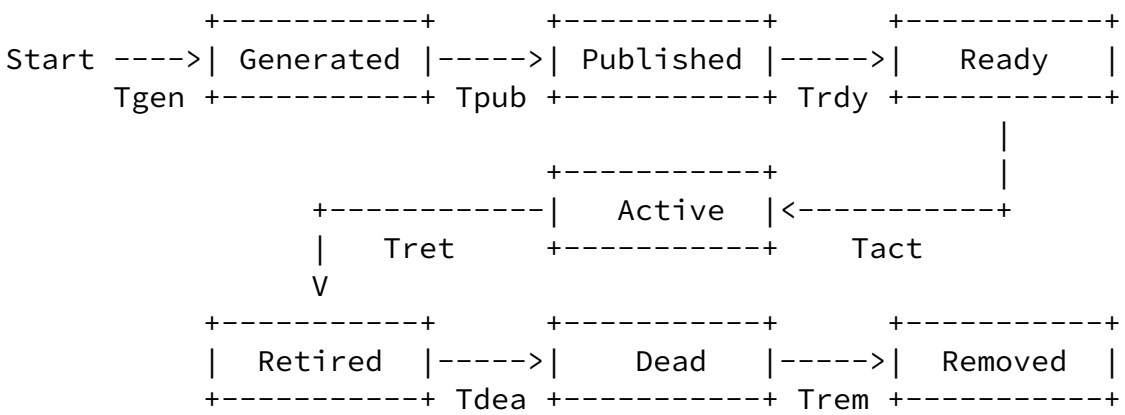


Figure 2: ZSK State Diagram.

The states are:

Generated The key has been created.

Published The DNSKEY record is published in the zone, but resolvers may have earlier versions of the DNSKEY RRset in their cache.

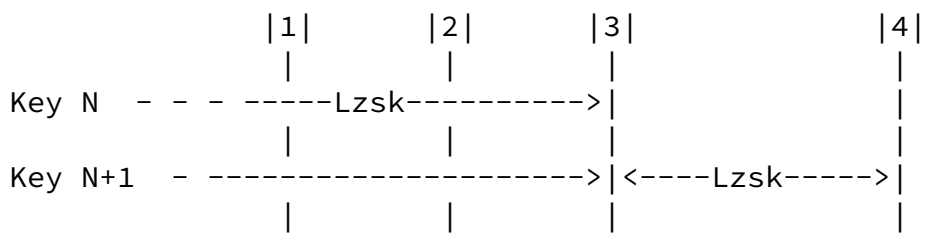
Ready	The key has been published for long enough to guarantee that all cached versions of the zone's DNSKEY RRset contain this key.
Active	The key is in the zone and is being used to sign RRsets.
Retired	The key is in the zone but is no longer being used to sign RRsets. However, there may still be RRSIG records in caches that were created with this key.
Dead	The key is published in the zone but there are no RRSIGs in existence created with this key.
Removed	The key has been removed from the zone.

3.3. Stand-By Zone-Signing Keys

Although ZSKs will usually be rolled according to some regular schedule, there may be occasions when an emergency ZSK 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 at least one stand-by ZSK in this state at all times will minimise delay.

3.3.1. Stand-By Key Scheduling

One way to achieve this is to regard successor keys as stand-by keys for emergency rollovers and to introduce them in the zone as early as possible. A modification of Figure 1 illustrates this:



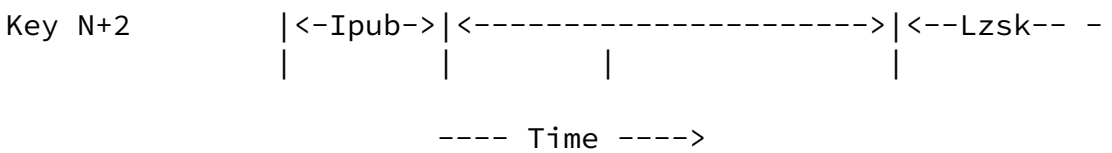


Figure 3: Timeline showing stand-by key replacement.

In this figure, it is assumed that key N is initially in the active

state and that key N+1 is in the ready state. Key N+1 is the successor to key N but is regarded as the stand-by key for an emergency re-signing until the time comes to use it to sign the zone.

Event 1: At least Ipub before key N's retire time, key N+2 is published in the zone.

Event 2: key N+2 moves into the ready state.

Event 3: key N is retired and key N+1 becomes active (as described in Section 3.1, events 7 - 9). Key N+2 is now regarded as the stand-by key.

Event 4: key N+1 is retired and key N+2 becomes the current key. By this time, key N+3 will have been published and be in the ready state.

The above illustrates one way of handling stand-by keys for emergency use. An equally valid alternative would be to have a permanent stand-by key. In this scheme, a key is published in the zone but, unless it needs to be used in an emergency, is never used to sign it. Instead, active keys are replaced by their successors as shown in Figure 1.

3.3.2. Number of Stand-By Keys

An emergency key rollover could be required at any time. Referring back to Figure 3, should an emergency rollover be required between events 2 and 3, the sequence would happen as previously described: there is a already key (key N+2) ready to take over as the stand-by key when the current stand-by key becomes active. In the worst case though, it may be required that the system run without an stand-by

key for a while. For example, if a key rollover was required between events 3 and 4 in Figure 3, the timeline would look like:

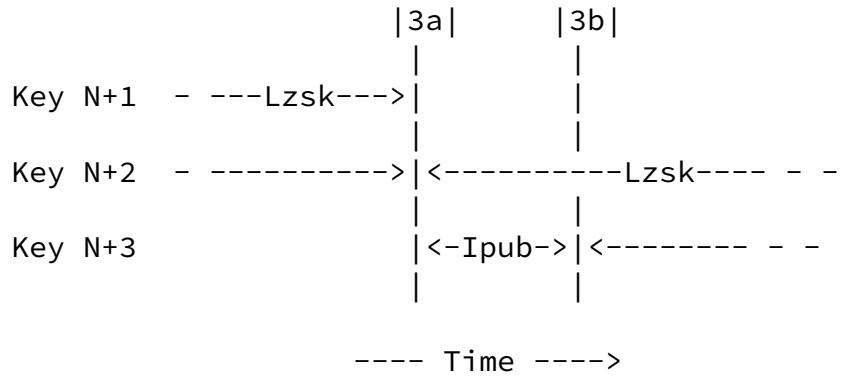


Figure 4: Timeline showing emergency key rollover.

(The interval shown above lies between events 3 and 4 in Figure 3, the events being labelled 3a and 3b to highlight this.)

Here it is assumed that key N+1 is initially in the active state and that the single stand-by key (N+2) is in the ready state. It is well before the active key's retire time, so there are only these two ZSKs in the zone. The events are:

Event 3a: an emergency ZSK rollover is required. Key N+1 is retired and key N+2 becomes active. At this time, key N+3 (which will ultimately become the new stand-by key) is published in the zone.

Event 3b: key N+3 moves into the ready state, after which it can be used to replace key N+1 should the need arise.

Between events 3a and 3b however, only the active key (key N+2) can be used to sign the zone. If a second emergency arises in this interval, the active key cannot be replaced: key rollover must wait until the new stand-by key (N+3) becomes ready. Of course, this can be mitigated by having a number of stand-by keys available, but how many is a matter of policy; there is a need to weigh the likelihood of a key compromise against the number of keys required.

4. Key-Signing Keys

4.1. Introduction

There are three significant differences between key-signing keys (KSKs) and ZSKs:

1. In the ZSK case the issue for the validator 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 and 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 a DS record in the parent (which is itsef trusted), or the KSK being explicitly configured as a trust anchor for the validator. Hence the additional two differences:

2. A KSK rollover algorithm may need to involve the parent zone in the addition and removal of DS records.

3. KSKs may be configured as so-called "trust anchors" in validating resolvers.

These differences have the following implications for KSK rollovers:

1. The rollover logic must ensure that validators are able to validate the DNSKEY RRset throughout the rollover process -

either through updates to the chain of trust from the parent zone or through updates to the trust anchor configuration.

2. Timings are not wholly within the control of the zone manager, in that the time taken to publish the DS records depends on the policies and procedures of the parent zone. A consequence of this is that the interdependence of the parent DS and child DNSKEY records means that when a new key is introduced, for a period downstream validators might have inconsistent data, i.e. the DS record without the DNSKEY record or vice-versa. Although this is valid state according to [RFC4035], the information cannot be used for validation until the validator has both components.

3. Securely removing such KSKs from the zone requires a mechanism for communicating this information to any validators that may have the KSK configured as a trust-anchor. The typical method would be by publishing the revoked key as described in [RFC5011].

There are some differences in the sequence of events between the cases of a zone where a KSK is authenticated via a DS record in the parent zone and one where it is authenticated by a trust anchor configured into a validator. These will be highlighted as appropriate.

4.2. Parent Zone Considerations

If (as is the usual case) the parent and child zones are managed by different entities, the timing of some of the steps in the KSK rollover operation may be subject to uncertainty.

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.

4.3. Rollover Strategies

When the parent zone is secured, there are several different ways to roll a KSK whilst ensuring that the zones do not go insecure or bogus in the process:

- Double KSK: 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 DNSKEY 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 validator caches, the old key is removed from the DNSKEY RRset.

- Double DS: the new DS record is published. After waiting for this change to propagate into the caches of all validators, 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 validator caches, the old DNSKEY and DS record are removed.

In essence, "Double KSK" means that the new KSK is introduced first, and then the new DS (for this KSK). With "Double DS" it is the other way around. Finally, Double RRset does both updates more or less in parallel.

Of the three methods, the double RRset method is preferred because:

- It allows the rollover to be done in the shortest time.

- It can support policies that require a period of running with old and new KSKs simultaneously.

4.4. Key Timeline

The timeline for the key rollover is shown below:

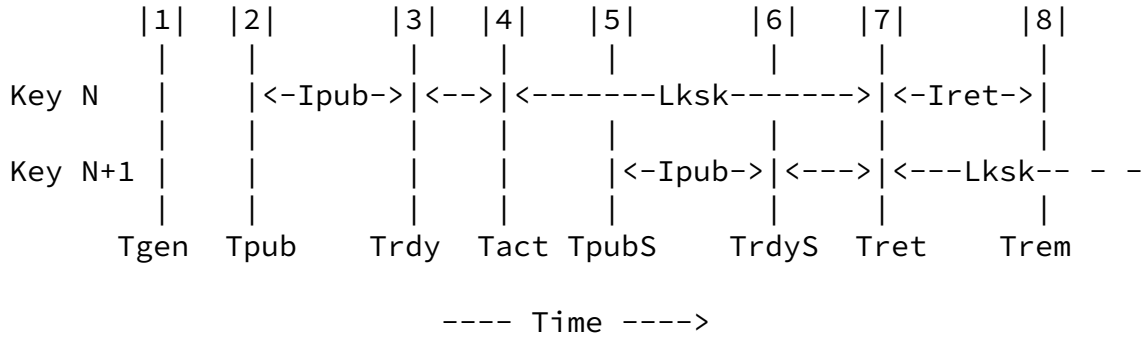


Figure 5: Timeline for a KSK rollover.

Event 1: key N is generated at time Tgen and enters the generate state. As before, although there is no reason why the key cannot be generated immediately prior to publication, some implementations may find its 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 this time the corresponding DS record is made available. If the parent zone is secure, this means submitting the DS record to the parent zone for publication; if not, it is distributed by some mechanism to allow validators to configure it as a trust anchor. This time is the publish time (Tpub) and the KSK is said to be in the published state.

Event 3: after some time (the publication interval, Ipub), any validator that has copies of the DNSKEY and/or DS RRsets in its cache will have a copy of the data for key N. This point is the ready time and is given by:

$$Trdy = Tpub + Ipub$$

Regarding the associated DS record, there are now two cases to consider, where the parent is signed and where the parent is not signed:

Event 3 (parent signed): In the case of the KSK, the publication interval depends on the publication interval of both the DNSKEY record and the DS record. These are independent, so a suitable expression for Ipub is:

Ipub = max(IpubC, IpubP)

IpubC is the publication interval in the child zone and IpubP that of the parent.

The child term comprises two parts - the time taken for the introduction of the DNSKEY record to be registered on the downstream secondary servers (= DprpC, the child propagation delay) and the time taken for information about the DNSKEY RRset to expire from the validator cache, i.e.

IpubC = DprpC + TTLkeyC

(TTLkeyC is the TTL for a DNSKEY record in the child zone.)

The parent term is similar, but also includes the time taken for the DS record to be included in the parent zone after the request has been made. In other words:

IpubP = Dreg + DprpP + TTLds

Dreg is the registration delay, which is the time taken between the submission of the DS record to the parent zone and its publication in the zone. DprpP the propagation delay in the parent zone and TTLds the TTL for a DS record.

Throughout the introduction of the two RRs, the zone can be validated by by the existing KSK and DS record. However, there are special considerations regarding the first KSK in a zone, and these are discussed below.

Event 3 (parent not signed): if the parent is not signed then there is no parent publication interval (theoretically the DS record could be configured in a validator immediately it is made available), in which case the minimumn value of the publication interval is given by:

Ipub = IpubC

Event 3 (common): In both cases, if the management policy is to support [RFC5011], there is also the additional condition that the new key needs to be published for at least as long as the RFC5011 add hold-down time, defined in that document as "30 days or the expiration time of the original TTL of the first trust point DNSKEY RRSet that contained the new key, whichever is greater".

This can be expressed as the condition:

Ipub >= max(30 days, TTLkey)

At Trdy, as the key has already been used to sign the DNSKEY RRset, the key is also active in that all other KSKs could be withdrawn from the zone at this point and the zone would still be valid. However, while a predecessor key is active, it is convenient to regard the successor key as merely being ready.

Event 4: at some later time, the predecessor key is withdrawn from the zone and, in the absence of any emergency keys, key N becomes the only KSK for the zone. The key is said to be active, and this time is the active time (Tact).

Event 5: as with the ZSK, at some point we need to give thought to key replacement. The successor key must be introduced into the zone at a time such that when the current key is withdrawn, any validator that has key information (DNSKEY and/or DS records) in its cache will have data about the successor key.

As before, this interval is the publication interval, Ipub. Denoting the publication time of the successor key as TpubS, we get:

TpubS <= Tact + Lksk - Ipub

... where Lksk is the lifetime of the KSK.

Event 6: the successor key (key N+1) enters the ready state. This occurs at TrdyS, given by:

TrdyS = TpubS + Ipub

Event 7: at some time after that a decision will be made to retire the current key (key N). This will be when the current key has been active for its lifetime (Lksk). At this point, the retire time, the successor key becomes active and the current key is said to be retired:

Tret = Tact + Lksk

(... with the obvious condition that Tret >= TrdyS.)

If the management policy is to support [RFC5011], the retired key should now have the revoke bit set and be included in the DNSKEY RRset. the revoked key should also be used to sign it.

Event 8: at some later time, the DNSKEY record can be removed from the child zone. If there is a secure parent, a request can be made to remove the DS record from the parent zone. This is the removal

time, Trem and is given by:

$$Trem = Tret + Iret$$

where, as before, Iret is the retire interval. This will be zero unless [RFC5011] is being followed, in which case Iret will be equal to the RFC5011 remove hold-down time value of 30 days.

Notes:

1. In the case of a ZSK, as pre-publication is the method of choice, only one key at a time is used to sign the zone. Therefore, when the active ZSK is retired, there may be copies of signatures created using it in the caches of downstream validators. For this reason, a copy of the ZSK has to be kept in the zone until all cached signatures have expired.

   With a KSK - where double RRset is the method of choice - both the active key and the successor key sign the DNSKEY RRset. By the time the successor becomes active, any validator with the DNSKEY RRset in its cache has a copy of the successor key. Therefore as soon as the active key is retired, it can be removed from the zone - there is no retire interval (unless [RFC5011] is being followed) or dead time (although for completeness, and in analogy with the ZSK, a dead time could be defined by Tdea = Trem).

2. There is an additional consideration when introducing a KSK into a zone for the first time, and that is that no validator can be in a position where it can access the trust anchor before the KSK appears in the zone. To do so will cause the validator to declare the zone to be bogus.

The second point 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 validator 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 validators 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. 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 one to allow for the effect of negative caching.

4.5. Key States

The key states for a KSK during the rollover are identical to those in Figure 2.

4.6. Stand-By Key-Signing Keys

In the same way that additional ZSKs are kept in a ready state in the zone to act as emergency keys, additional KSKs need to be available in the ready state for the same reason. The number of stand-by keys kept available is a matter of key management policy, and the logic for the introduction of stand-by keys into the zone follows the same

reasoning as that given in Section 3.3 on the introduction of stand-by ZSKs.

5. Algorithm Considerations

The preceding sections have implicitly assumed that all keys and signatures are created using a single algorithm. However, [RFC4035] (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 changes - there is no relationship between the keys of different algorithms. This means that they can be rolled independently of one another. (Indeed, the keys for each algorithm could, if desired, have different TTLs.) 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.

6. Summary

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], in virtually all cases the rollover time will be determined by the RFC5011 add and remove hold-down times. It should be emphasised 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 rollover is significantly

simplified by the introduction of stand-by keys as standard practice during all types of rollovers.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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

9. Acknowledgements

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

10. Change History

- draft-morris-dnsop-dnssec-key-timing-01
  * Use latest boilerplate for IPR text.
  * List different ways to roll a KSK (acknowledgements to Mark Andrews).
  * Restucture 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 Hones 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 Hones).
  * 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.

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

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
L	lifetime: calculated interval set by the zone manager
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 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

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.

DprpP Propagation delay in the parent zone.

Dreg Registration delay. As a parent zone are often managed by a different organisation to the one under consideration, 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.

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.

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 validator caches.

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 validator.)

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

Tact Active time of the key. For a ZSK, the time that they key is first used to sign the zone. For a KSK, the time at which this key is regarded as being the principal KSK for the zone.

Tdea Dead time of a key. Applicable only to ZSKs, this is the time at which any record signatures held in validator 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 a validators that have key information from this zone cached have a copy of this key in their cache. (In the case of KSKs, should the validators 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.

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.

TTLsig Time to live of an RRSIG record.

Ttsa 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

Stephen Morris
Nominet
Minerva House, Edmund Halley Road
Oxford, OX4 4DQ
UK

Phone: +44 1865 332211
Email: stephen@nominet.org.uk

Johan Ihren
Autonomica
Franzengatan 5
Stockholm, SE-112 51
Sweden

Phone: +46 8615 8573
Email: johani@autonomica.se

John Dickinson
Stables 4 Suite 10, Howbery Park

Wallingford, OX10 8BA
UK

Phone:
Email: jad@jadickinson.co.uk