dnsop
Internet-Draft
Updates: 7583 (if approved)
Intended status: Standards Track
Expires: June 22, 2018

W. Hardaker
USC/ISI
W. Kumari
Google

December 19, 2017

# Security Considerations for RFC5011 Publishers
# draft-ietf-dnsop-rfc5011-security-considerations-10

Abstract

This document extends the RFC5011 rollover strategy with timing advice that must be followed by the publisher in order to maintain security. Specifically, this document describes the math behind the minimum time-length that a DNS zone publisher must wait before signing exclusively with recently added DNSKEYs. This document also describes the minimum time-length that a DNS zone publisher must wait after publishing a revoked DNSKEY before assuming that all active RFC5011 resolvers should have seen the revocation-marked key and removed it from their list of trust anchors.

This document contains much math and complicated equations, but the summary is that the key rollover / revocation time is much longer than intuition would suggest. If you are not both publishing a DNSSEC DNSKEY, and using RFC5011 to advertise this DNSKEY as a new Secure Entry Point key for use as a trust anchor, you probably don't need to read this document.

Table of Contents

## 1. Introduction

[RFC5011] defines a mechanism by which DNSSEC validators can update their list of trust anchors when they've seen a new key published in a zone or revoke a properly marked key from a trust anchor list. However, RFC5011 [intentionally] provides no guidance to the publishers of DNSKEYs about how long they must wait before switching to exclusively using recently published keys for signing records, or how long they must wait before ceasing publication of a revoked key. Because of this lack of guidance, zone publishers may derive incorrect assumptions about safe usage of the RFC5011 DNSKEY advertising, rolling and revocation process. This document describes the minimum security requirements from a publisher's point of view and is intended to complement the guidance offered in RFC5011 (which is written to provide timing guidance solely to a Validating Resolver's point of view).

### 1.1. Document History and Motivation

To verify this lack of understanding is wide-spread, the authors reached out to 5 DNSSEC experts to ask them how long they thought they must wait before signing a zone exclusively with a new KSK [RFC4033] that was being introduced according to the 5011 process. All 5 experts answered with an insecure value, and we determined that this lack of mathematical understanding might cause security concerns in deployment. We hope that this companion document to RFC5011 will rectify this understanding and provide better guidance to zone publishers that wish to make use of the RFC5011 rollover process.

### 1.2. Safely Rolling the Root Zone's KSK in 2017/2018

One important note about ICANN's (currently in process) 2017/2018 KSK rollover plan for the root zone: the timing values chosen for rolling the KSK in the root zone appear completely safe, and are not affected by the timing concerns introduced by this draft

### 1.3. Requirements notation

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. Background

The RFC5011 process describes a process by which a RFC5011 Resolver may accept a newly published KSK as a trust anchor for validating future DNSSEC signed records. It also describes the process for publicly revoking a published KSK. This document augments that information with additional constraints, from the SEP publisher's points of view. Note that this document does not define any other operational guidance or recommendations about the RFC5011 process and restricts itself to solely the security and operational ramifications of switching to exclusively using recently added keys or removing revoked keys too soon.

Failure of a DNSKEY publisher to follow the minimum recommendations associated with this draft can result in potential denial-of-service attack opportunities against validating resolvers. Failure of a DNSKEY publisher to publish a revoked key for a long enough period of time may result in RFC5011 Resolvers leaving that key in their trust anchor storage beyond the key's expected lifetime.

## 3. Terminology

SEP Publisher The entity responsible for publishing a DNSKEY (with the Secure Entry Point (SEP) bit set) that can be used as a trust anchor.

Zone Signer The owner of a zone intending to publish a new Key-Signing-Key (KSK) that may become a trust anchor for validators following the RFC5011 process.

RFC5011 Resolver A DNSSEC Resolver that is using the RFC5011 processes to track and update trust anchors.

Attacker An entity intent on foiling the RFC5011 Resolver's ability to successfully adopt the Zone Signer's new DNSKEY as a new trust anchor or to prevent the RFC5011 Resolver from removing an old DNSKEY from its list of trust anchors.

sigExpirationTime The amount of time between the DNSKEY RRSIG's Signature Inception field and the Signature Expiration field.

Also see Section 2 of [RFC4033] and [RFC7719] for additional terminology.

## 4. Timing Associated with RFC5011 Processing

These sections define a high-level overview of [RFC5011] processing. These steps are not sufficient for proper RFC5011 implementation, but provide enough background for the reader to follow the discussion in this document. Readers need to fully understand [RFC5011] as well to fully comprehend the content and importance of this document.

### 4.1. Timing Associated with Publication

RFC5011's process of safely publishing a new DNSKEY and then assuming RFC5011 Resolvers have adopted it for trust falls into a number of high-level steps to be performed by the SEP Publisher. This document discusses the following scenario, which the principle way RFC5011 is currently being used (even though Section 6 of RFC5011 suggests having a stand-by key available):

1. Publish a new DNSKEY in a zone, but continue to sign the zone with the old one.

2. Wait a period of time.

3. Begin to exclusively use recently published DNSKEYs to sign the appropriate resource records.

This document discusses the time required to wait during step 2 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". Section 5 describes an attack based on this (common) erroneous belief, which can result in a denial of service attack against the zone.

### 4.2. Timing Associated with Revocation

RFC5011's process of advertising that an old key is to be revoked from RFC5011 Resolvers falls into a number of high-level steps:

1. Set the revoke bit on the DNSKEY to be revoked.

2. Sign the revoked DNSKEY with itself.

3. Wait a period of time.

4. Remove the revoked key from the zone.

This document discusses the time required to wait in step 3 of the above process. Some interpretations of RFC5011 have erroneously determined that the wait time is equal to RFC5011's "hold down time". This document describes an attack based on this (common) erroneous belief, which results in a revoked DNSKEY potentially remaining as a trust anchor in a RFC5011 Resolver long past its expected usage.

## 5. Denial of Service Attack Walkthrough

This section serves as an illustrative example of the problem being discussed in this document. Note that in order to keep the example simple enough to understand, some simplifications were made (such as by not creating a set of pre-signed RRSIGs and by not using values that result in the addHoldDownTime not being evenly divisible by the activeRefresh value); the mathematical formulas in Section 6 are, however, complete.

If an attacker is able to provide a RFC5011 Resolver with past responses, such as when it is in-path or able to perform any number of cache poisoning attacks, the attacker may be able to leave compliant RFC5011 Resolvers without an appropriate DNSKEY trust anchor. This scenario will remain until an administrator manually fixes the situation.

The time-line below illustrates an example of this situation.

### 5.1. Enumerated Attack Example

The following example settings are used in the example scenario within this section:

TTL (all records) 1 day

sigExpirationTime 10 days

Zone resigned every 1 day

Given these settings, the sequence of events in Section 5.1.1 depicts how a SEP Publisher that waits for only the RFC5011 hold time timer length of 30 days subjects its users to a potential Denial of Service attack. The timing schedule listed below is based on a SEP Publisher publishing a new Key Signing Key (KSK), with the intent that it will later be used as a trust anchor. We label this publication time as "T+0". All numbers in this sequence refer to days before and after this initial publication event. Thus, T-1 is the day before the introduction of the new key, and T+15 is the 15th day after the key was introduced into the fictitious zone being discussed.

In this dialog, we consider two keys within the example zone:

K_old: An older KSK and Trust Anchor being replaced.

K_new: A new KSK being transitioned into active use and expected to become a Trust Anchor via the RFC5011 automated trust anchor update process.

### 5.1.1. Attack Timing Breakdown

The steps shows an attack that foils the adoption of a new DNSKEY by a 5011 Resolver when the SEP Publisher that starts signing and publishing with the new DNSKEY too quickly.

T-1 The K_old based RRSIGs are being published by the Zone Signer. [It may also be signing ZSKs as well, but they are not relevant to this event so we will not talk further about them; we are only considering the RRSIGs that cover the DNSKEYs in this document.] The Attacker queries for, retrieves and caches this DNSKEY set and corresponding RRSIG signatures.

T+0 The Zone Signer adds K_new to their zone and signs the zone's key set with K_old. The RFC5011 Resolver (later to be under attack) retrieves this new key set and corresponding RRSIGs and notices the publication of K_new. The RFC5011 Resolver starts the (30-day) hold-down timer for K_new. [Note that in a more real-world scenario there will likely be a further delay between the point where the Zone Signer publishes a new RRSIG and the RFC5011 Resolver notices its publication; though not shown in this example, this delay is accounted for in the equation in Section 6 below]

T+5 The RFC5011 Resolver queries for the zone's keyset per the RFC5011 Active Refresh schedule, discussed in Section 2.3 of RFC5011. Instead of receiving the intended published keyset, the Attacker successfully replays the keyset and associated signatures recorded at T-1 to the victim RFC5011 Resolver. Because the signature lifetime is 10 days (in this example), the replayed signature and keyset is accepted as valid (being only 6 days old, which is less than sigExpirationTime) and the RFC5011 Resolver cancels the (30-day) hold-down timer for K_new, per the RFC5011 algorithm.

T+10 The RFC5011 Resolver queries for the zone's keyset and discovers a signed keyset that includes K_new (again), and is signed by K_old. Note: the attacker is unable to replay the records cached at T-1, because the signatures have now expired.

Thus at T+10, the RFC5011 Resolver starts (anew) the hold-timer for K_new.

T+11 through T+29 The RFC5011 Resolver continues checking the zone's key set at the prescribed regular intervals. During this period, the attacker can no longer replay traffic to their benefit.

T+30 The Zone Signer knows that this is the first time at which some validators might accept K_new as a new trust anchor, since the hold-down timer of a RFC5011 Resolver not under attack that had queried and retrieved K_new at T+0 would now have reached 30 days. However, the hold-down timer of our attacked RFC5011 Resolver is only at 20 days.

T+35 The Zone Signer (mistakenly) believes that all validators following the Active Refresh schedule (Section 2.3 of RFC5011) should have accepted K_new as a the new trust anchor (since the hold down time (30 days) + the query interval [which is just 1/2 the signature validity period in this example] would have passed). However, the hold-down timer of our attacked RFC5011 Resolver is only at 25 days (T+35 minus T+10); thus the RFC5011 Resolver won't consider it a valid trust anchor addition yet, as the required 30 days have not yet elapsed.

T+36 The Zone Signer, believing K_new is safe to use, switches their active signing KSK to K_new and publishes a new RRSIG, signed with (only) K_new, covering the DNSKEY set. Non-attacked RFC5011 validators, with a hold-down timer of at least 30 days, would have accepted K_new into their set of trusted keys. But, because our attacked RFC5011 Resolver now has a hold-down timer for K_new of only 26 days, it failed to ever accept K_new as a trust anchor. Since K_old is no longer being used to sign the zone's DNSKEYs, all the DNSKEY records from the zone will be treated as invalid. Subsequently, all of the records in the DNS tree below the zone's apex will be deemed invalid by DNSSEC.

## 6. Minimum RFC5011 Timing Requirements

This section defines the minimum timing requirements for making exclusive use of newly added DNSKEYs and timing requirements for ceasing the publication of DNSKEYs to be revoked. We break our timing solution requirements into two primary components: the mathematically-based security analysis of the RFC5011 publication process itself, and an extension of this that takes operational realities into account that further affect the recommended timings.

First, we define the term components used in all equations in Section 6.1.

### 6.1. Equation Components

#### 6.1.1. addHoldDownTime

The addHoldDownTime is defined in Section 2.4.1 of [RFC5011] as:

> The add hold-down time is 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 ensures that at least two validated DNSKEY RRSets that contain the new key MUST be seen by the resolver prior to the key's acceptance.

#### 6.1.2. lastSigExpirationTime

The latest value (i.e. the future most date and time) of any RRSig Signature Expiration field covering any DNSKEY RRSet containing only the old trust anchor(s) that are being superseded. Note that for organizations pre-creating signatures this time may be fairly far in the future unless they can be significantly assured that none of their pre-generated signatures can be replayed at a later date.

#### 6.1.3. sigExpirationTime

The amount of time between the DNSKEY RRSIG's Signature Inception field and the Signature Expiration field.

#### 6.1.4. sigExpirationTimeRemaining

sigExpirationTimeRemaining is defined in Section 3.

#### 6.1.5. activeRefresh

activeRefresh time is defined by RFC5011 by

> A resolver that has been configured for an automatic update of keys from a particular trust point MUST query that trust point (e.g., do a lookup for the DNSKEY RRSet and related RRSIG records) no less often than the lesser of 15 days, half the original TTL for the DNSKEY RRSet, or half the RRSIG expiration interval and no more often than once per hour.

This translates to:

```
activeRefresh = MAX(1 hour,
                    MIN(sigExpirationTime / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days)
                    )
```

#### 6.1.6. activeRefreshOffset

The activeRefreshOffset term must be added for situations where the activeRefresh value is not a factor of the addHoldDownTime. Specifically, activeRefreshOffset will be "addHoldDownTime % activeRefresh", where % is the mathematical mod operator (calculating the remainder in a division problem). This will frequently be zero, but could be nearly as large as activeRefresh itself.

Note that later (in Section 6.1.8), when real-world scenerios will trump this value that is useful only in theoretical worlds with no network delays and other operational considerations. We leave it here only as an important marker in the security analysis of the base RFC5011 protocol.

#### 6.1.7. driftSafetyMargin

Moving past the theoretical model parameters above, we not that clock drift, network delays and implementation differences will result in the RFC5011 Resolver query times to drift over time. Because of this, a driftSafetyMargin term must be introduce that accounts for these real world delays. We set this value to be the same as the activeRefresh value, which will ensure that any timing drift in RFC5011 Resolver queries will be accounted for.

Note: even a negative clock drift can actually cause RFC5011 Resolvers to require up to an extra activeRefresh period before it will accept a new DNSKEY as a trust anchor.

#### 6.1.8. timingSafetyMargin

Both of the activeRefreshOffset and driftSafetyMargin parameters deal with timing delays introduced by mathematical analysis of RFC5011 (activeRefreshOffset) and by real world considerations (driftSafetyMargin). To find a safe value to extend timing, we define a timingSafetyMargin that is the maximum of these two values. Since the driftSafetyMargin is set to activeRefresh, and activeRefreshOffset is always less than an activeRefresh, the final timingSafetyMargin value will be activeRefresh.

Explicitly expanding out the math:

```
timingSafetyMargin = min(activeRefreshOffset, driftSafetyMargin)

timingSafetyMargin = min(addHoldDownTime % activeRefresh,
                         activeRefresh)

timingSafetyMargin = activeRefresh
```

### 6.1.9. retrySafetyMargin

The retrySafetyMargin is an extra period of time to account for caching, network delays, dropped packets, and other operational concerns otherwise beyond the scope of this document. The value operators should chose is highly dependent on the deployment situation associated with their zone. Note that no value of a retrySafetyMargin can protect against resolvers that are "down". None the less, we do offer the following as one method considering reasonable values to select from.

The following list of variables need to be considered when selecting an appropriate retrySafetyMargin value:

successRate: A likely success rate for client queries and retries

numResolvers: The number of client RFC5011 Resolvers

Note that RFC5011 defines retryTime as:

> If the query fails, the resolver MUST repeat the query until satisfied no more often than once an hour and no less often than the lesser of 1 day, 10% of the original TTL, or 10% of the original expiration interval. That is, retryTime = MAX (1 hour, MIN (1 day, .1 * origTTL, .1 * expireInterval)).

With the successRate and numResolvers values selected and the definition of retryTime from RFC5011, one method for determining how many retryTime intervals to wait in order to reduce the set of uncompleted servers to 0 assuming normal probability is thus:

```
x = (1/(1 - successRate))

retryCountWait = Log_base_x(numResolvers)
```

To reduce the need for readers to pull out a scientific calculator, we offer the following lookup table based on successRate and numResolvers:

retryCountWait lookup table

| | | Number of client RFC5011 Resolvers (numResolvers) | | | | |
|---|---|---|---|---|---|---|
| | | 10,000 | 100,000 | 1,000,000 | 10,000,000 | 100,000,000 |
| Probability of Success Per Retry Interval (successRate) | 0.01 | 917 | 1146 | 1375 | 1604 | 1833 |
| | 0.05 | 180 | 225 | 270 | 315 | 360 |
| | 0.10 | 88 | 110 | 132 | 153 | 175 |
| | 0.15 | 57 | 71 | 86 | 100 | 114 |
| | 0.25 | 33 | 41 | 49 | 57 | 65 |
| | 0.50 | 14 | 17 | 20 | 24 | 27 |
| | 0.90 | 4 | 5 | 6 | 7 | 8 |
| | 0.95 | 4 | 4 | 5 | 6 | 7 |
| | 0.99 | 2 | 3 | 3 | 4 | 4 |
| | 0.999 | 2 | 2 | 2 | 3 | 3 |

Finally, a suggested value of retrySafetyMargin can then be this retryCountWait number multiplied by the retryTime from RFC5011:

retrySafetyMargin = retryCountWait * retryTime

## 6.2. Timing Requirements For Adding a New KSK

Section 6.2.1 defines a method for calculating the amount of time to wait until it is safe to start signing exclusively with a new DNSKEY (especially useful for writing code involving sleep based timers), and Section 6.2.2 defines a method for calculating a wall-clock value after which it is safe to start signing exclusively with a new DNSKEY (especially useful for writing code based on clock-based event triggers).

### 6.2.1. Wait Timer Based Calculation

Given the attack description in Section 5, the correct minimum length of time required for the Zone Signer to wait after publishing K_new but before exclusively using it and newer keys is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTimeRemaining
              + activeRefresh
              + timingSafetyMargin
              + retrySafetyMargin
```

#### 6.2.1.1. Fully expanded equation

Given the equation components defined in Section 6.1, the full expanded equation is:

```
addWaitTime = addHoldDownTime
              + sigExpirationTimeRemaining
              + MAX(1 hour,
                    MIN(sigExpirationTime / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days)
                    )
              + activeRefresh
              + retrySafetyMargin
```

### 6.2.2. Wall-Clock Based Calculation

The equations in Section 6.2.1 are defined based upon how long to wait from a particular moment in time. An alternative, but equivalent, method is to calculate the date and time before which it is unsafe to use a key for signing. This calculation thus becomes:

```
addWallClockTime = lastSigExpirationTime
                   + addHoldDownTime
                   + activeRefresh
                   + timingSafetyMargin
                   + retrySafetyMargin
```

where lastSigExpirationTime is the latest value of any sigExpirationTime for which RRSIGs were created that could potentially be replayed. Fully expanded, this becomes:

```
addWallClockTime = lastSigExpirationTime
                   + addHoldDownTime
                   + 2 * MAX(1 hour,
                             MIN(sigExpirationTime / 2,
                                 MAX(TTL of K_old DNSKEY RRSet) / 2,
                                 15 days)
                             )
                   + activeRefresh
                   + retrySafetyMargin
```

### 6.2.3. Timing Constraint Summary

The important timing constraint introduced by this memo relates to the last point at which a RFC5011 Resolver may have received a replayed original DNSKEY set, containing K_old and not K_new. The next query of the RFC5011 validator at which K_new will be seen

without the potential for a replay attack will occur after the old DNSKEY RRSIG's Signature Expriation Time. Thus, the latest time that a RFC5011 Validator may begin their hold down timer is an "Active Refresh" period after the last point that an attacker can replay the K_old DNSKEY set. The worst case scenario of this attack is if the attacker can replay K_old just seconds before the (DNSKEY RRSIG Signature Validity) field of the last K_old only RRSIG.

### 6.2.4. Additional Considerations for RFC7583

Note: our notion of addWaitTime is called "Itrp" in Section 3.3.4.1 of [RFC7583]. The equation for Itrp in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Itrp equation in RFC7583 also does not include the 2*TTL safety margin, though that is an operational consideration.

### 6.2.5. Example Scenario Calculations

For the parameters listed in Section 5.1, our resulting addWaitTime is:

```
  addWaitTime = 30
                + 10
                + 1 / 2
                + 1 / 2          (days)

  addWaitTime = 43               (days)
```

This addWaitTime of 42.5 days is 12.5 days longer than just the hold down timer, even with the needed retrySafetyMargin value being left out (which we exclude due to the lack of necessary operational parameters).

## 6.3. Timing Requirements For Revoking an Old KSK

This issue affects not just the publication of new DNSKEYs intended to be used as trust anchors, but also the length of time required to continuously publish a DNSKEY with the revoke bit set.

Section 6.2.1 defines a method for calculating the amount of time operators need to wait until it is safe to cease publishing a DNSKEY (especially useful for writing code involving sleep based timers), and Section 6.2.2 defines a method for calculating a minimal wall-clock value after which it is safe to cease publishing a DNSKEY (especially useful for writing code based on clock-based event triggers).

### 6.3.1. Wait Timer Based Calculation

Both of these publication timing requirements are affected by the attacks described in this document, but with revocation the key is revoked immediately and the addHoldDown timer does not apply. Thus the minimum amount of time that a SEP Publisher must wait before removing a revoked key from publication is:

```
remWaitTime = sigExpirationTimeRemaining
              + activeRefresh
              + timingSafetyMargin
              + retrySafetyMargin

remWaitTime = sigExpirationTimeRemaining
              + MAX(1 hour,
                    MIN((sigExpirationTime) / 2,
                        MAX(TTL of K_old DNSKEY RRSet) / 2,
                        15 days))
              + activeRefresh
              + retrySafetyMargin
```

Note also that adding retryTime intervals to the remWaitTime may be wise, just as it was for addWaitTime in Section 6.

### 6.3.2. Wall-Clock Based Calculation

Like before, the above equations are defined based upon how long to wait from a particular moment in time. An alternative, but equivalent, method is to calculate the date and time before which it is unsafe to cease publishing a revoked key. This calculation thus becomes:

```
   remWallClockTime = lastSigExpirationTime
                    + activeRefresh
                    + timingSafetyMargin
                    + retrySafetyMargin

   remWallClockTime = lastSigExpirationTime
                    + MAX(1 hour,
                          MIN((sigExpirationTime) / 2,
                              MAX(TTL of K_old DNSKEY RRSet) / 2,
                              15 days))
                    + timingSafetyMargin
                    + retrySafetyMargin
```

where lastSigExpirationTime is the latest value of any sigExpirationTime for which RRSIGs were created that could potentially be replayed. Fully expanded, this becomes:

### 6.3.3. Additional Considerations for RFC7583

Note that our notion of remWaitTime is called "Irev" in Section 3.3.4.2 of [RFC7583]. The equation for Irev in RFC7583 is insecure as it does not include the sigExpirationTime listed above. The Irev equation in RFC7583 also does not include a safety margin, though that is an operational consideration.

### 6.3.4. Example Scenario Calculations

For the parameters listed in Section 5.1, our example:

```
remwaitTime = 10
              + 1 / 2          (days)

remwaitTime = 10.5             (days)
```

Note that for the values in this example produce a length shorter than the recommended 30 days in RFC5011's section 6.6, step 3. Other values of sigExpirationTime and the original TTL of the K_old DNSKEY RRSet, however, can produce values longer than 30 days.

Note that because revocation happens immediately, an attacker has a much harder job tricking a RFC5011 Resolver into leaving a trust anchor in place, as the attacker must successfully replay the old data for every query a RFC5011 Resolver sends, not just one.

## 7. IANA Considerations

This document contains no IANA considerations.

## 8. Operational Considerations

A companion document to RFC5011 was expected to be published that describes the best operational practice considerations from the perspective of a zone publisher and SEP Publisher. However, this companion document has yet to be published. The authors of this document hope that it will at some point in the future, as RFC5011 timing can be tricky as we have shown, and a BCP is clearly warranted. This document is intended only to fill a single operational void which, when left misunderstood, can result in serious security ramifications. This document does not attempt to document any other missing operational guidance for zone publishers.

## 9. Security Considerations

This document, is solely about the security considerations with respect to the SEP Publisher's ability to advertise new DNSKEYs via the RFC5011 automated trust anchor update process. Thus the entire document is a discussion of Security Considerations when adding or removing DNSKEYs from trust anchor storage using the RFC5011 process.

For simplicity, this document assumes that the SEP Publisher will use a consistent RRSIG validity period. SEP Publishers that vary the length of RRSIG validity periods will need to adjust the sigExpirationTime value accordingly so that the equations in Section 6 and Section 6.3 use a value that coincides with the last time a replay of older RRSIGs will no longer succeed.

## 10. Acknowledgements

The authors would like to especially thank to Michael StJohns for his help and advice and the care and thought he put into RFC5011 itself and his continued reviews and suggestions for this document. He also designed the suggested math behind the suggested retrySafetyMargin values in Section 6.1.9.

We would also like to thank Bob Harold, Shane Kerr, Matthijs Mekking, Duane Wessels, Petr Petr Spacek, Ed Lewis, and the dnsop working group who have assisted with this document.

## 11. Normative References

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

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <http://www.rfc-editor.org/info/rfc5011>.

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

## Appendix A. Real World Example: The 2017 Root KSK Key Roll

In 2017 and 2018, ICANN expects to (or has, depending on when you're reading this) roll the key signing key (KSK) for the root zone. The relevant parameters associated with the root zone at the time of this writing is as follows:

```
addHoldDownTime:                    30 days
Old DNSKEY sigExpirationTime:       21 days
Old DNSKEY TTL:                      2 days
```

Thus, sticking this information into the equation in Section Section 6 yields (in days from publication time):

```
addWaitTime = 30
              + 21
              + MAX(1 hour,
                    MIN(21 / 2,     # activeRefresh
                        MAX(2) / 2,
                        15 days),
                    )
              + activeRefresh

addWaitTime = 30 + 21 + 1 + 1

addWaitTime = 53 days
```

Also note that we exclude the retrySafetyMargin value, which is calculated based on the expected client deployment size.

Thus, ICANN must wait a minimum of 52 days before switching to the newly published KSK (and 26 days before removing the old revoked key once it is published as revoked). ICANN's current plans involve waiting over 3 months before using the new KEY and 69 days before removing the old, revoked key. Thus, their current rollover plans are sufficiently secure from the attack discussed in this memo.

Authors' Addresses

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net