DNSOP
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

D. Migault
Ericsson
D. York
Internet Society
E. Lewis
ICANN

October 30, 2017

DNSSEC Validators Requirements
draft-mglt-dnsop-dnssec-validator-requirements-06

Abstract

DNSSEC provides data integrity and source authentication to a basic DNS RReet. Given a RRset, a public key and a signature, a DNSSEC validator checks the signature, time constraints, and other, local, policies. In case of mismatch the RRSet is considered illegitimate and is rejected.

Accuracy in DNSSEC validation, that is, avoiding false positives and catching true negatives, requires that both the signing process and validation process adhere to the protocol, which begins with external configuration parameters. This document describes requirements for a validator to be able to perform accurate validation.

Status of This Memo

Copyright Notice

Table of Contents

1. 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. Introduction

DNSSEC validation [RFC4033], [RFC4034] and [RFC4035] has two core concepts. One is the matching of a RRSIG resource record's contents to a RRset, making use of a DNSKEY resource record (named in the

RRSIG record). Two is the placing of trust in the DNSKEY resource record.

Evaluation based on a RRSIG record involves a few steps. Most visible is a cryptographic operation, matching the digital signature in the RRSIG with the specified public key in the named DNSKEY record and a properly prepared DNS RRset. This is meant to demonstrate that the RRset came from an entity with the private component of the key (source authenticity).

Not to be forgotten are the other matches to perform. To address the threat of reply attacks, wall-clock (absolute) time is checked. To address the authority of the source, the named DNSKEY record is checked for appropriateness (i.e., owned by the same zone is the default policy).

The RRSIG record also contains other information intended to help the validator perform its work, in some cases "sane value" checks are performed. For instance, the original TTL (needed to prepare the RR set for validation) ought to be equal to or higher than the received TTL.

Requirements related to validation exist in [RFC4033], [RFC4034] and [RFC4035]. However, the specification of the validation is not sufficient to enable a wide deployment of DNSSEC validators. In fact, there are a number of situations where the necessary condition are not met by the DNSSEC validator to perform DNSSEC validation. When such conditions are not met, the DNSSEC validation may qualify improperly a RRset as invalid. This document is focused on the necessary mechanisms that DNSSEC validators should implement in order to make DNSSEC validation output accurate. The mechanisms described in this document include, provisioning mechanisms as well as monitoring and management mechanisms that enables an administrator to validate the validity of the DNSSEC validation output.

3. Terminology

This document uses the following terminology:

DNSSEC validator: the entity that performs DNSSEC resolution and performs signature validation.

Accurate validation: validation that avoids false positives and catches true negatives. (not sure if this is needed, but seems appropriate)

Trust Anchor Data Store:

4. DNSSEC Validator Description

This is a conceptual block diagram of the elements involved with DNSSEC validation. This is not meant to be an architecture for code, this is meant to be a framework for discussion and explanation.

```
            +-------------+  +---------------+
            |             |  |               |
            | Time Source |  | Cryptographic |
            |             |  |   Libraries   |
            |             |  |               |
            +-------------+  +---------------+
                   |                 |
                   v                 v
            +--------------------------------+   +--------------+
            |                                |   |              |
            |                                |<--| Trust Anchor |
            |    DNSSEC Validation Engine    |   |   Manager &  |
            |                                |-->|   Storage    |
            |                                |   |              |
            +--------------------------------+   +--------------+
                  ^ |               ^                   |
                  | v               |                   |
            +-------------+  +---------------+          |
            |             |  |               |          |
            | DNS Caches  |  | DNS Messages  |<---------+
            |             |  |               |
            +-------------+  +---------------+
```

Figure 1: DNSSEC Validator Description

Time Source : Wall clock time Cryptograhic Libraries: Code performing mathematical functions.

DNS Message : Receiver of DNS responses DNS Caches: Positive and negative caches.

Trust Anchor Manager : database of trust anchors, manages trust DNSSEC Validation Engine: follows local policy to approve data.

a. Time Source -> DNSSEC Validation Engine Current time upon request, in appropriate time zone setting

b. Cryptographic Libraries-> DNSSEC Validation Engine Supplies code to perform math, the engine determines the DNSSEC Security Algorithms supported

c. DNS Caches <- DNSSEC Validation Engine Enter the results of a validation (positive data, negative failures)

d. DNS Caches -> DNSSEC Validation Engine Stored keys, etc., used in building a chain of trust

e. DNS Messages -> DNSSEC Validation Engine DNS Responses needed validation

f. Trust Anchor Management & Storage -> DNSSEC Validation Engine Supplies trust anchor information when needed.

g. Trust Anchor Management & Storage <- DNSSEC Validation Engine Information to update the trust anchor store, resulting from automated update requests.

h. Trust Anchor Management & Storage -> DNS Messages Requests made to manage trust anchors.

i. Not shown - Name Server Process Management interfaces to elements, handling of Checking Disabled request, responses, as well as all API requests made of the name server.

5. Time derivation and absence of Real Time Clock

With M2M communication some devices are not expecting to embed Real Time Clock (Raspberry Pi is one example of such devices). When these devices are re-plugged the initial time is set to January 1 1970. Other devices that have clocks that may suffer from time derivation. All these devices cannot rely on their time estimation to perform DNSSEC validation.

REQ1: A DNSSEC validator MUST be provided means to update the time without relying on DNSSEC.

Note that updating time in order to be able to perform DNSSEC validation may easily come with a chicken-and-egg problem when the NTP server is designated by its FQDN. The update mechanisms must consider the DNSSEC validator may not able to validate the DNSSEC queries. In other words, the mechanisms may have to update the time over an unsecure DNSSEC resolution.

6. Trust Anchor

6.1. Trust Anchor Bootstrapping

A validator needs to have trust anchors or it will never be able to construct a chain of trust. Trust anchors are defined by DNSSEC to be keys that are inherently trusted, configured by authorized parties, in the validator. The configuration can be via an automated process, such as Automated Updates of DNSSEC Trust Anchors [RFC5011], [I-D.ietf-dnsop-rfc5011-security-considerations], or via manual process.

An implementation of a validator needs to allow an operator to choose any automated process supported by the validator. (No requirements are stated about what processes to support, only one is standardized to date.) An implementation needs to also afford the operator the ability to override or manage via a purely manual process, the storage of managed keys. This includes adding, deleting, changing and inspecting.

Beyond the scope of these requirements are the decision processes of authorized parties in placing trust in keys.

REQ2: A DNSSEC validator MUST check the validity of its Trust Anchors. When a Trust Anchor cannot be verified, the DNSSEC validator MUST send a warning and SHOULD NOT start validating traffic without manual validation.

REQ3: A DNSSEC validator SHOULD be able to retrieve a Trust Anchor with bootstrapping mechanism. Such mechanism' security MUST NOT be based on DNSSEC, but could instead include downloading a XML file from a trusted URL, or a PKIX certificate.

Although some bootstrapping mechanisms to securely retrieve publish [RFC7958] and retrieve [UNBOUND-ANCHOR] the Root Zone Trust Anchor have been defined, it is believed these mechanisms should be extended to other KSKs or Trust Anchors. In fact it is not always possible to build a trusted delegation between the Root Zone and any sub zone. This may happen for example if one of the upper zones does not handle the secure delegation or improperly implement it. A DS RRset may not be properly filled or its associated signature cannot be validated. As the chain of trust between a zone and the root zone may not be validated, the DNSSEC validation for the zone requires a Trust Anchor. Such DNS(SEC) resolutions may be critical for infrastructure management. A company "Example" may, for example, address all its devices under the domain example.com and may not want disruption to happen if the .com delegation cannot be validated for any reason. Such companies may provision there DNSSEC validator with the Trust Anchor KSK for the zone example.com in addition to the regular DNSSEC delegation. Similarly some some domains may present different views

such as a "private" view and a "public view". These zones may have some different content, and may use a different KSK for each view.

6.2. Trust Anchor Data Store

When DNSSEC validator are running and a Trust Anchor KSK roll over is ongoing, a network administrator or any trust party may be willing to check whether the new published keys are being stored in a Trust Anchor Data Store with an appropriated status. Such inspection aims at detecting an non successful Trust Anchor roll over before traffic is being rejected. When a new Trust Anchor has not been considered by the DNSSEC validator, a trusted party may be able to provision the DNSSEC validator with the new Trust Anchor, and eventually may remove the revoked Trust Anchor.

While using a Trust Anchor that has been removed results in the DNSSEC validator rejecting multiple legitimate responses, the consequences associated to accepting a rogue Trust Anchor as a legitimate Trust Anchor are even worst. Such attacks would result in an attacker taking control of the entire naming space behind the Trust Anchor. In the case of the Root Zone KSK, for example, almost all name space would be under the control of the attacker. In addition, to the name space, once the rogue Trust Anchor is configured, there is little hope the DNSSEC validator be re-configured with the legitimate Trust Anchor without manual intervention. As a result, it is crucial to cautiously handle operations related to the Trust Anchor provisioning. Means must be provided so network administrator can clearly diagnose the reason a Trust Anchor is not valid to avoid accepting a rogue Trust Anchor inadvertently.

DNSSEC may also be used in some private environment. Corporate networks and home networks, for example, may want to take advantage of DNSSEC for a local scope network. Typically, a corporate network may use a local scope Trust Anchor to validate DNS RRsets provided by authoritative DNSSEC server in the corporate network. This use case is also known as the "split-view" use case. These RRsets within the corporate network may differ from those hosted on the public DNS infrastructure. Note that using different Trust Anchor for a given zone may expose a zone to signature invalidation. This is especially the case for DNSSEC validators that are expected to flip-flop between local and public scope. How validators have to handle the various provisioned Trust Anchors is out of scope of the document.

Home network may use DNSSEC with TLDs or associated domain names that are of local scope and not even registered in the public DNS infrastructure. This requires the ability to manage the Trust Anchor as well.

The necessity to interact with the Trust Anchors lead to the following requirements:

REQ4: A DNSSEC validator MUST store its Trust Anchors in a dedicated Trust Anchor Data Store. Such database MUST store informations associated to each Trust Anchor status as well as the time the status has been noticed by the DNSSEC validator. Such database MUST be resilient to DNSSEC validator reboot.

REQ5: Trust Anchor states SHOULD at least consider those described in [RFC5011] (Start, AddPend, Valid, Missing, Revoked, Removed). Additional states SHOULD also be able to indicate additional motivations for revoking the Trust Anchor such as a Trust Anchor known to be corrupted, a Trust anchor miss published, or part of a regular roll over procedure.

REQ6: A DNSSEC validator MUST provide access to the Trust Anchor Data Sase to authorized user only. Access control is expected to be based on a least privileged principles.

REQ7: A trusted party MUST be able to add, remove a Trust Anchor in the Trust Anchor Data Store.

6.3. Interactions with the cached RRsets

In addition when a Trust Anchor is revoked, the DNSSEC validator may behave differently if the revocation is motivated by a regular roll over operation or instead by revoking a Trust Anchor that is known as being corrupted. In the case the roll over procedure, is motivated by revoking a Trust Anchor that is known to be corrupted, the DNSSEC validator may be willing to flush all RRsets that depends on the Trust Anchor.

REQ8: A DNSSEC validator MUST be able to flush the cached RRsets that rely on a Trust Anchor.

7. ZSK / KSK

7.1. KSK/ZSK Data Store

A number of reasons may result in inconsistencies between the RRsets stored in the cache and those published by the authoritative server.

An emergency KSK / ZSK rollover may result in a new KSK / ZSK with associated new RRSIG published in the authoritative zone, while DNSSEC validator may still cache the old value of the ZSK / KSK. For a RRset not cached, the DNSSEC validator performs a DNSSEC query to the authoritative server that returns the RRset signed with the new

KSK / ZSK. The DNSSEC validator may not be able to retrieve the new KSK / ZSK while being unable to validate the signature with the old KSK / ZSK. This either result in a bogus resolution or in an invalid signature check. Note that by comparing the Key Tag Fields, the DNSSEC validator is able to notice the new KSK / ZSK used for signing differs from the one used to generate the received generated signature. However, the DNSSEC validator is not expectected to retrieve the new ZSK / KSK, as such behavior could be used by an attacker. Intsead, ZSK / ZSK key roll ove rprocedure are expected to avoid such inconsistencies.

Similarly, a KSK / ZSK roll over may be performed normally, that is as described in [RFC6781] and [RFC7583]. While the KSK / ZSK roll over is performed, there is no obligation to flush the RRsets in the cache that have been associated with the old key. In fact, these RRset may still be considered as trusted and be removed from the cache as their TTL timeout. With very long TTL, these RRset may remain in the cache while the ZSK / KSK with a shorter TTL is no longer published nor in the cache. In such situations, the purpose of the KSK / ZSK is to validate the data is considered trusted at the time it enters the cache, and such trust may remain after the KSK / ZSK is being rolled over. Note also that even though the data may not be associated to the KSK / ZSK that has been used to valiadte the data, the link between the KSK / ZSK and teh data is still stored in teh cache using the RRSIG. Note also that inconsistencies between the ZSK / KSK stored in the cache and those published on the authoritative server, may lead to inconsistencies to downstream DNSSEC validators that realy on multiple cache over time. Typically, a request for the KSK / ZSK may have been provided by a cache that is storing the new published value, while the data and associated sigature may be associated to the old KSK / ZSK.

KSK and ZSK are associated with configuration parameters, and as such are expected to be stored only in the cache. As a result, flushing their value from the cache could constitute a way forward to refresh them. On the other hand, their respective function is also to prevent illegitimate RRsets to be validated and so more understanding is need before taking any action associated to the KSK or ZSK. More specifically, the network administrator SHOULD be provided the appropriated information required to distinguish a misconfiguration from an attack.

The following requirements are thus considered for the KSK / ZSK.

REQ9: A DNSSEC validator MUST store its KSK/ZSK in a dedicated KSK/ ZSK Data Base. Such database MUST store informations associated to each KSK/ZSK status as well as the time the status has been noticed by the DNSSEC validator. Such

database MUST NOT be resilient to DNSSEC validator reboot, that is the information stored in the Data Base MUST NOT be used to populate the cache, while it MAY be used as second factor verification, or audit for example.

REQ10: KSK/ZSK status and informaton SHOULD be monitored continuously and associated with their respective state as well as verified time. These states and time SHOULD be resilient to reboot.

REQ11: KSK/ZSK states SHOULD at least consider those described in section 3.1 of [RFC7583] (Generated, Published, Ready, Active, Retired, Dead, Removed, Revoked). Additional states SHOULD also be able to indicate additional motivations for revoking the KSK/ZSK such as a KSK/ZSK known to be corrupted, a KSK/ZSK miss published, or part of a regular roll over procedure.

REQ12: A DNSSEC validator MUST provide access to the KSK/ZSK data base to authorized user only. Access control is expected to be based on a least privileged principles.

REQ13: A trusted party MUST be able to add, remove a Trust Anchor in the KSK/ZSK Database.

Similarly to its counter part the TA Data Store, the KSK/ZSK Data Store is expected to be resilient to reboot. However the motivation for having the KSK/ZSK Data Store resilient to reboot differs from those for making the TA Data Store resilient to reboot. TA Data Store needs to be resilient as the Trust Anchors are necessary to perform the DNSSEC validation. KSK/ZSK are not expected to be locally stored, but instead are expected to be resolved, validated by the TA and stored in the cache. The reason for making the KSK/ZSK Data Store resilient to reboot is mostly to enable audit of the DNSSEC validator.

7.2. KSK/ZSK Data Store and Trust Anchor Data Store

A zone may have been badly signed, which means that the KSK or ZSK cannot validate the RRSIG associated to the RRsets. This may not be due to a key roll over, but to an incompatibility between the keys (KSK or ZSK) and the signatures.

When such situation occurs, there is only a choice between not validating the RRsets or invalidating their signature. This is a policy design that needs to be taken by the network administrator. In other ways, flushing the RRset are not expected to address this issue. Such KSK/ZSK are known as Negative Trust Anchors [RFC7646].

With Negative Trust Anchor, the zone for a given time will be known as "known insecure". The DNSSEC Validator is not expected to perform signature validation for this zone. It is expected that this information is associated to a Time To Live (TTL).

Note that, this information may be used as an attack vector to impersonate a zone, and must be provided in a trusted way, by a trusted party.

If a zone has been badly signed, the administrator of the authoritative DNS server may resign the zone with the same keys or proceed to an emergency key rollover. If the signature is performed with the same keys, the DNSSEC Validator may notice by itself that RRSIG can be validated. On the other hand if a key rollover is performed, the newly received RRSIG will carry a new key id. Upon receiving a new key id in the RRSIG, the DNSSEC Validator is expected to retrieve the new ZSK/KSK. If the RRSIG can be validated, the DNSSEC Validator is expected to remove the "known insecure" flag.

However, if the KSK/ZSK are rolled over and RRSIG cannot be validated, it remains hard for the DNSSEC validator to determine whether the RRSIG cannot be validated or that RRSIG are invalid. As a result:

REQ14: A trusted party MUST be able to indicate a DNSSEC validator that a KSK or a ZSK as Negative Trust Anchor. Such Trust Anchors MUST NOT be used for RRSIG validation and MUST be moved to the Trust Anchor Data Store, so the information become resilient to reboot.

REQ15: A trusted party MUST be able to indicate a DNSSEC validator that a KSK/ZSK is known "back to secure".

7.3. **Interactions with cached RRsets**

The key roll over procedure intends to ensure that the published RRsets can be validated with the KSK / ZSK stored in the various cache of the DNSSEC validators. As a consequence, the key roll over enables trusted data to be cached. However, the key roll over does not necessarily prevents that cached be always validated with the currenlty published key. In fact, a cached data may have been validated by the former key and remain in the cache while the former key has been rolled out. Such inconsistencies may be acceptable and correspond to the following trust model: the KSK / ZSK validate the cached data can be trusted at time T. There is no specific information that leads to considers that trust at time T is subject to doubts at current time, so the cached data remain trusted.

While such inconsistencies may have little impact on end host DNSSEC validators, it may be different for large resolving platforms with downstream DNSSEC validators, and a DNSSEC validator may be willing to maintain its cached data consistent with the published KSK / ZSK. A trusted third party may willing to remove all cached RRsets that have been validated by the KSK/ZSK upon some specific states (revoked, or Removed for example), of after some time after the state is noticed. In this later case, only the RRset whose TTL has not expired yet would be flushed.

On the other hand, when a KSK / ZSK is known to be corrupted, this state may affect the trust that has been established at time T. In such case, the DNSSEC validator may be willling to flush all cached data that has been validated by the currently known corrupted KSK / ZSK, including the KSK / ZSK itslef.

As a result, the following requirements are expected:

REQ16: A DNSSEC validator MUST be able to flush the cached KSK/ZSK.

REQ17: A DNSSEC validator MUST be able to flush the cached RRsets associated to a KSK/ZSK.

8. DS

The DS RRset is stored in the parent zone to build a chain of trust with the child zone. This DS RRset can be invalid because its RDATA (KSK) is not anymore used in the child zone or because the DS is badly signed and cannot be validated by the DNSSEC Validator.

In both cases the child zone is considered as bogus and the valid child zone's KSK should become a Trust Anchor KSK. This requirements is fulfilled by the requirement to add a Trust Anchor in Section 6.

9. Cryptography Deprecation

As mentioned in [RFC8247] and [RFC8221] cryptography used one day is expected over the time to be replaced by new and more robust cryptographic mechanisms. In the case of DNSSEC signature protocols are likely to be updated over time. In order to anticipate the sunset of one of the signature scheme, a DNSSEC validator may willing to estimate the impact of deprecating one signature scheme.

Currently [RFC6975] provides the ability for a DNSSEC validator to announce an authoritative server the supported signature schemes. However, a DNSSEC validator is not able to determine other than by trying whether a signature scheme is supported by the authoritative server.

In order for a DNSSEC validator to safely deprecate one signature scheme the following requirement should be fulfilled.

REQ18: A DNSSEC validator SHOULD be able to request the signature scheme supported by an authoritative server.

10. Reporting

A DNSSEC validator receiving a DNS response cannot make the difference between receiving an non-secure response versus an attack. Dropping DNSSEC fields by a misconfigured middle boxes, such as DS, RRRSIG is considered as an attack.

A DNSSEC validator is expected to perform secure DNS resolution and as such protect its stub client. An invalid response may be the result of an attack or a misconfiguration, and the DNSSEC validator may play an important role in sharing this information.

REQ19: A DNSSEC validation SHOULD be able to report the unavailability of the DNSSEC service.

REQ20: A DNSSEC validator SHOULD be able to report a invalid DNSSEC validation.

11. IANA Considerations

There are no IANA consideration for this document.

12. Security Considerations

The requirements listed in this document aim at providing the DNSSEC validator appropriated information so DNSSEC validation can be performed. On the other hand, providing inappropriate information can lead to misconfiguring the DNSSEC validator, and thus disrupting the DNSSEC resolution service. As a result, enabling the setting of configuration parameters by a third party may open a wide surface of attacks.

As an appropriate time value is necessary to perform signature check (cf. Section 5), an attacker may provide rogue time value to prevent the DNSSEC validator to check signatures.

An attacker may also affect the resolution service by regularly asking the DNSSEC validator to flush the KSK/ZSK from its cache (cf. Section 7). All associated data will also be flushed. This generates additional DNSSEC resolution and additional validations, as RRSet that were cached require a DNSSEC resolution over the Internet.

This affects the resolution service by slowing down responses, and increases the load on the DNSSEC validator.

An attacker may ask the DNSSEC validator to consider a rogue KSK/ZSK (cf. Invalid DS in Section 8 or Private KSK in Section 6), thus hijacking the DNS zone. Similarly, (cf. Section 7) an attacker may inform the DNSSEC validator not to trust a given KSK in order to prevent DNSSEC validation to be performed.

An attacker (cf. Section 7) can advertise a "known insecure" KSK or ZSK is "back to secure" to prevent signature check to be performed correctly.

As a result, information considered by the DNSSEC validator should be from a trusted party. This trust party should have been authenticated, and the channel used to exchange the information should also be protected and authenticated.

13. Acknowledgment

The need to address DNSSEC issues on the resolver side started in the Home Networks mailing list and during the IETF87 in Berlin. Among others, people involved in the discussion were Ted Lemon, Ralph Weber, Normen Kowalewski, and Mikael Abrahamsson. People involved in the email discussion initiated by Jim Gettys were, with among others, Paul Wouters, Joe Abley and Michael Richardson.

The current document has been initiated after a discussion with Paul Wouter and Evan Hunt.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

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

[RFC6975] Crocker, S. and S. Rose, "Signaling Cryptographic Algorithm Understanding in DNS Security Extensions (DNSSEC)", RFC 6975, DOI 10.17487/RFC6975, July 2013, <https://www.rfc-editor.org/info/rfc6975>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/info/rfc8221>.

[RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <https://www.rfc-editor.org/info/rfc8247>.

14.2. Informational References

[I-D.ietf-dnsop-rfc5011-security-considerations] Hardaker, W. and W. Kumari, "Security Considerations for RFC5011 Publishers", draft-ietf-dnsop-rfc5011-security-considerations-07 (work in progress), October 2017.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <https://www.rfc-editor.org/info/rfc6781>.

[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>.

[RFC7646] Ebersman, P., Kumari, W., Griffiths, C., Livingood, J., and R. Weber, "Definition and Use of DNSSEC Negative Trust Anchors", RFC 7646, DOI 10.17487/RFC7646, September 2015, <https://www.rfc-editor.org/info/rfc7646>.

[RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <https://www.rfc-editor.org/info/rfc7958>.

[UNBOUND-ANCHOR] "unbound-anchor.c File Reference", <https://www.unbound.net/documentation/doxygen/unbound-anchor_8c.html#details>.

Authors' Addresses

Daniel Migault
Ericsson

Email: daniel.migault@ericsson.com

Dan York
Internet Society

Email: york@isoc.org

Edward Lewis
ICANN

Email: edward.lewis@icann.org