DNSOP
Internet-Draft
Intended status: Experimental
Expires: January 5, 2015

J. Lundstroem
.SE
W. Mekking
NLnet Labs

July 4, 2014

DNSSEC Key and Signing Policies
draft-mekking-dnsop-kasp-00

Abstract

This document describes how key policies should look like.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Table of Contents

1. Introduction 2
1.1. Key Words 2
1.2. Terminology 2
1.3. Data Modeling 3
1.3.1. Data Modeling Types 3
1.3.2. Data Modeling Arguments 3
2. KASP Contents 3
2.1. Preamble 3
2.1.1. Policies 4
2.1.1.1. Signatures 4
2.1.1.2. Authenticated Denial of Existence 6
2.1.1.3. Keys 8
2.1.1.4. Zone 11
2.1.1.5. Parent 13
3. References 14
3.1. Informative References 14
3.2. Normative References 14
Appendix A. Changelog 16

1. Introduction

A key and signing policy (KASP) defines a DNSSEC [RFC4033] [RFC4034] [RFC4035] policy for one or more zones.

1.1. Key Words

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].

1.2. Terminology

The reader is assumed to be familiar with DNSSEC described in [RFC4033] [RFC4034] [RFC4035], [RFC5155] and [RFC6781].

The following terminology is used throughout this document:

KASP: Key And Signing Policy, describes a DNSSEC policy that can be applied to one or more zones.

A key and signing policy can be expressed in any format. This document uses XML as example.

1.3. Data Modeling

1.3.1. Data Modeling Types

This document reuses the modeling types described in [RFC6020].

The following modeling types are used:

container: A container node does not have a value, but it has a list of child nodes in the data tree. The child nodes are defined in the container's substatements.

leaf: A leaf node has a value, but no child nodes in the data tree. A leaf node exists in zero or one instances in the data tree.

list: The "list" statement is used to define an interior data node in the schema tree. A list node may exist in multiple instances in the data tree. Each such instance is known as a list entry.

choice: The "choice" statement defines a set of alternatives, only one of which may exist at any one time. A choice node does not exist in the data tree.

A choice consists of a number of branches, defined with the "case" substatement. Each branch contains a number of child nodes. The nodes from at most one of the choice's branches exist at the same time.

1.3.2. Data Modeling Arguments

The following arguments are used:

string: A string.

duration: A duration, as specified in ISO 8601 [REF].

empty: An empty type.

integer: An integer.

2. KASP Contents

2.1. Preamble

All policies MUST be enclosed in a KASP container.

```
container KASP {
    list Policy { ... }
}
```

A KASP container MUST contain a sequence of policy entries and MUST NOT contain any other modeling types.

2.1.1. Policies

Each policy MUST have a "name" leaf which contains the name of the policy. The name is used to link a policy and the zones signed using it; each policy MUST have a unique name. A policy named "default" MAY be used to associate with all zones that do not have a policy explicitly configured. A policy SHOULD have a description associated with it. Furthermore, a policy MUST have the containers Signatures, Denial, Keys, Zone and Parent. These containers are described in the forthcoming sections.

```
list Policy {
    key "name";
    leaf name {
        mandatory true;
        type string;
    }
    leaf description {
        type string;
    }
    container Signatures { ... }
    container Denial { ... }
    container Keys { ... }
    container Zone { ... }
    container Parent { ... }
}
```

2.1.1.1. Signatures

A Signatures container defines the policy parameters for creating RRSIG records and MUST be included. It MUST contain the following leaf nodes: Resign, Refresh, InceptionOffset. It MAY have a leaf node Jitter. It MUST contain a Validity container that includes leaf nodes for the validity periods of certain type of resource record sets (RRsets). The Default leaf node sets the validity period for all RRsets that do not have a specific leaf node in this Validity container. The Denial leaf node sets the validity period for all NSEC and NSEC3 RRsets.

The Validity container MUST include leaf nodes Default and Denial and MAY include other leaf nodes to differentiate between even more types of RRsets.

```
container Signatures {
    leaf Resign {
        mandatory true;
        type duration;
    }
    leaf Refresh {
        mandatory true;
        type duration;
    }
    leaf Jitter {
        type duration;
    }
    leaf InceptionOffset {
        mandatory true;
        type duration;
    }
    container Validity {
        leaf Default {
            mandatory true;
            type duration;
        }
        leaf Denial {
            mandatory true;
            type duration;
        }
    }
}
```

Here:

1. Resign - the re-sign interval, which is the interval when the signer MUST re-sign the zone.

2. Refresh - the refresh interval, detailing when a signature MUST be refreshed. As signatures are typically valid for much longer than the interval between runs of the signer, there is no need to re-generate the signatures each time the signer runs. The signature MUST be refreshed when the time until the signature expiration is closer than the refresh interval or when the data has been changed. A value of zero (PT0S) MUST be interpreted as to refresh the signatures each re-sign interval.

3. Jitter - If present, the value added to the expiration time of signatures to ensure that not all signatures expire at the same

time. The actual value of Jitter to be added MUST be -j + (r % 2j), where j is the jitter value from the policy and r a random duration, uniformly ranging between -j and j, is added to signature validity period to get the signature expiration time.

4. InceptionOffset - a duration that MUST be subtracted from the time at which a record is signed to give the start time of the record. This is required to allow for clock skew between the signing system and the system on which the signature is checked. Without it, the possibility exists that the checking system could retrieve a signature whose start time is later than the current time.

5. Validity - groups two or more elements of information related to how long the signatures are valid for - Denial is the validity period for all NSEC and NSEC3 RRsets, Default is the validity period for all other RRset.

[MM: Add MaxZoneTTL]

The relationship between these elements is shown [RFC6781], Figure 11.

2.1.1.2. Authenticated Denial of Existence

Authenticated denial of existence information is included within a Denial container. It MUST contain either an empty leaf node NSEC or a container NSEC3. A NSEC3 container MUST include a leaf node for TTL and Resalt and MUST include a container Hash. Additionally, it MAY include an OptOut leaf node.

The Hash container MUST include three leaf nodes: Algorithm, Iterations and SaltLength.

```
container Denial {
    choice RRtype {
        case plain {
            leaf NSEC {
                mandatory true;
                type empty;
            }
        }
        case hashed {
            container NSEC3 {
                leaf TTL {
                    mandatory true;
                    type duration;
                }
                leaf OptOut {
                    type empty;
                }
                leaf Resalt {
                    mandatory true;
                    type duration;
                }
                container Hash {
                    leaf Algorithm {
                        mandatory true;
                        type integer;
                    }
                    leaf Iterations {
                        mandatory true;
                        type integer;
                    }
                    leaf SaltLength {
                        mandatory true;
                        type integer;
                    }
                }
            }
        }
    }
```

If NSEC is used, zones with this policy MUST include NSEC records when signing the zone. If NSEC3 is used, zones with this policy MUST include NSEC3 and NSEC3PARAM records with the appropriate policy values:

1. TTL - if present, the TTL for the NSEC3PARAM resource record MUST be set to this value. If not present, PT0S (0) SHOULD be used as TTL.

2. OptOut - if present, all included NSEC3 records SHOULD set the Opt-Out bit. The signer SHOULD NOT include NSEC3 records for insecure delegations.

3. Resalt - A new salt value MUST be generated each Resalt interval value.

4. Algorithm, Iterations, SaltLength MUST be used as the parameters to the hash algorithm.

The choice and case modeling types are not included in the actual data tree. In the case that NSEC is used, the XML example would be:

```
<Denial>
    <NSEC/>
</Denial>
```

2.1.1.3. Keys

Parameters relating to keys can be found in Keys container. This container MUST include leaf node for TTL, and MAY include leaf nodes for PublishSafety, RetireSafety, ShareKeys and Purge. Furthermore, it MAY include a KSK list, a ZSK list and/or a CSK list. The latter indicates that a Single-Type Signing Scheme is used. If no such lists are included in the policy, the zone remains unsigned.

```
container Keys {
    leaf TTL {
        mandatory true;
        type duration;
    }
    leaf PublishSafety {
        type duration;
    }
    leaf RetireSafety {
        type duration;
    }
    leaf ShareKeys {
        type empty;
    }
    leaf Purge {
        type duration;
    }
    list KSK { ... }
    list ZSK { ... }
    list CSK { ... }
}
```

The leaf nodes specify the following behavior:

1. TTL - This value MUST be used as the TTL of the DNSKEY resource records.

2. PublishSafety - If present, the publish safety margin is used to give some extra time to cover unforeseen events and MUST be added to the publish interval in key timing equations.

3. RetireSafety - If present and similar to PublishSafety, the retire safety margin MUST be added to the retire interval in key timing equations.

4. ShareKeys - if multiple zones are associated with the same policy, the presence of a ShareKeys node indicates that a physical key can be shared between zones.

5. Purge - if present, keys in the dead state (as defined in key-timing-bis) will be automatically purged from the database after this interval.

A CSK, KSK or ZSK list MUST include leaf nodes for Algorithm, Length, Lifetime and Repository. Additionally, they MAY include a RollType leaf, Standby leaf and ManualRollover leaf. A CSK or KSK list MAY also include a RFC5011 leaf.

```
list CSK {
    leaf Algorithm {
        mandatory true;
        type integer;
    }
    leaf Length {
        mandatory true;
        type integer;
    }
    leaf Lifetime {
        mandatory true;
        type duration;
    }
    leaf Repository {
        mandatory true;
        type string;
    }
    leaf RollType {
        type string;
    }
    leaf Standby {
        type empty;
```

```
        }
        leaf ManualRollover {
            type empty;
        }
        leaf RFC5011 {
            type empty;
        }
    }

    list KSK {
        leaf Algorithm {
            mandatory true;
            type integer;
        }
        leaf Length {
            mandatory true;
            type integer;
        }
        leaf Lifetime {
            mandatory true;
            type duration;
        }
        leaf Repository {
            mandatory true;
            type string;
        }
        leaf RollType {
            type string;
        }
        leaf Standby {
            type empty;
        }
        leaf ManualRollover {
            type empty;
        }
        leaf RFC5011 {
            type empty;
        }
    }

    list ZSK {
        leaf Algorithm {
            mandatory true;
            type integer;
        }
        leaf Length {
            mandatory true;
            type integer;
```

```
        }
        leaf Lifetime {
            mandatory true;
            type duration;
        }
        leaf Repository {
            mandatory true;
            type string;
        }
        leaf RollType {
            type string;
        }
        leaf Standby {
            type empty;
        }
        leaf ManualRollover {
            type empty;
        }
    }
```

where:

1. Algorithm - this algorithm MUST be used.

2. Length - this key length MUST be used.

3. Lifetime - determines how long the key SHOULD be used for before it is rolled.

4. Repository - determines the location of the physical keys. The corresponding type of keys MUST be stored in this repository.

5. Standby - if present, determines the number of standby keys MUST be held in the zone.

6. ManualRollover - if present, this indicates that the key rollover MUST NOT occur automatically, it may only be initiated by the operator.

7. RFC5011 - if present, this indicates that the key rollover MUST include additional time for key publication and retirement and MUST revoke the predecessor key accordingly.

2.1.1.4. Zone

The Zone container encloses general information concerning the zone. It MAY include a PropagationDelay leaf and MUST have a SOA container.

The SOA container MUST include a TTL leaf, Minimum leaf and Serial leaf.

```
container Zone {
    leaf PropagationDelay {
        type duration;
    }
    container SOA {
        leaf TTL {
            mandatory true;
            type duration;
        }
        leaf Minimum {
            mandatory true;
            type duration;
        }
        leaf Serial {
            mandatory true;
            type string;
        }
    }
}
```

The PropagationDelay leaf holds the amount of time needed for information changes at the master server for the zone to work its way through to all the secondary servers. The value MAY be used in equations related to key timings during a rollover. If PropagationDelay is not used, a signer MUST use different heuristics to make sure key timings during a rollover are correct, for example by querying the name servers for the required records to exist.

The SOA container gives values of parameters for the SOA record in the signed zone. These values will override values set for the SOA record in the unsigned zone file:

1. TTL - The TTL of the SOA record MUST be set to the value of the TTL leaf.

2. Minimum - The MINIMUM RDATA field of the SOA record MUST be set to the value of the Minimum leaf.

3. Serial - The format of the serial number in the signed zone. This is one of: "counter", datecounter, unixtime, keep.

When Serial is set to "counter", the SOA serial MUST be incremented by one every re-sign.

When Serial is set to "datecounter", the SOA serial MUST be set to YYYYMMDDCC, where YYYYMMDD represents the current date and CC the number of new re-signs that day. If there are more than 100 re-signs a day, the date MUST rollover to the day after and count is reset to 00.

When Serial is "unixtime", the SOA serial MUST be set to the seconds since the epoch (1970-01-01 UTC).

When Serial is "keep" the SOA serial MUST be set to the SERIAL RDATA field of the SOA record in the unsigned zone. If this does not increment the last used serial, a signer MUST NOT produce a signed zone.

2.1.1.5. Parent

If a DNSSEC zone is in a chain of trust, digest information about the KSKs used in the zone will be published in DS records in the parent zone. To properly roll keys, timing information about the parent zone must be configured in the Parent container. A Parent container MAY include a PropagationDelay leaf, a DS container and a SOA container. The DS container MAY include a TTL leaf and the SOA container MAY include TTL and Minimum leafs.

```
container Parent {
    leaf PropagationDelay {
        type duration;
    }
    container DS {
        leaf TTL {
            type duration;
        }
    }
    container SOA {
        leaf TTL {
            type duration;
        }
        leaf Minimum {
            type duration;
        }
    }
}
```

Here, the PropagationDelay leaf configures how long it takes to get a DS record published in the parent zone after submitting the DNSKEY or DS record to the parent zone manager. The value SHOULD be used in equations related to key timings during a rollover. If PropagationDelay is not used, a signer MUST use different heuristics

to make sure key timings during a rollover are correct, for example by querying the name servers for the required records to exist.

The DS and SOA containers give values of parameters for the DS record and SOA record in the parent zone. These values SHOULD be used in equations related to key timings during a rollover. If these values are not used, a signer MUST query the parent name servers in order to retrieve the correct values.

A before:

1. TTL - The TTL of the DS and SOA records MUST be set to the value of the corresponding TTL leaf.

2. Minimum - The MINIMUM RDATA field of the SOA record MUST be set to the value of the Minimum leaf.

3. References

3.1. Informative References

[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.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, December 2012.

3.2. Normative References

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Appendix A. Changelog

- Initial version

Authors' Addresses

Jerry Lundstroem
.SE
Ringvaegen 100
Box 7399
Stockholm 103 91
SE

EMail: jerry.lundstrom@iis.se
URI: http://www.iis.se/

W. (Matthijs) Mekking
NLnet Labs
Science Park 400
Amsterdam 1098 XH
NL

EMail: matthijs@nlnetlabs.nl
URI: http://www.nlnetlabs.nl/