

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 9, 2013

F. Ljunggren
Kirei AB
AM. Eklund Lowinder
.SE
T. Okubo
ICANN
November 05, 2012

A Framework for DNSSEC Policies and DNSSEC Practice Statements
draft-ietf-dnsop-dnssec-dps-framework-11

Abstract

This document presents a framework to assist writers of DNSSEC Policies and DNSSEC Practice Statements, such as Domain Managers and Zone Operators on both the top-level and secondary level, who are managing and operating a DNS zone with Security Extensions (DNSSEC) implemented.

In particular, the framework provides a comprehensive list of topics that should be considered for inclusion into a DNSSEC Policy definition and Practice Statement.

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 May 9, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Background	3
1.2. Purpose	3
1.3. Scope	4
2. Definitions	4
3. Concepts	6
3.1. DNSSEC Policy	6
3.2. DNSSEC Practice Statement	6
3.3. Relationship between DNSSEC Policy and Practice Statement	7
3.4. Set of Provisions	8
4. Contents of a Set of Provisions	10
4.1. Introduction	10
4.2. Publication and repositories	11
4.3. Operational requirements	12
4.4. Facility, management and operational controls	12
4.5. Technical security controls	17
4.6. Zone signing	21
4.7. Compliance audit	22
4.8. Legal matters	23
5. Outline of a Set of Provisions	23
6. IANA considerations	26
7. Security considerations	26
8. Acknowledgements	26
9. References	26
9.1. Normative references	26
9.2. Informative references	27
Authors' Addresses	27

1. Introduction

1.1. Background

The Domain Name System (DNS) was not originally designed with strong security mechanisms to provide integrity and authenticity of its data. Over the years, a number of vulnerabilities have been discovered that threaten the reliability and trustworthiness of the system.

The Domain Name System Security Extensions (DNSSEC, [RFC4033], [RFC4034], [RFC4035]) address these vulnerabilities by using public key cryptography to add data origin authentication, data integrity verification, and authenticated denial of existence capabilities to the DNS. In short, DNSSEC provides a way for software to verify the origin of DNS data and validate that it has not been modified in transit or by intermediaries.

To provide a means for stakeholders to evaluate the strength and security of the DNSSEC chain of trust, an entity operating a DNSSEC enabled zone may publish a DNSSEC Practice Statement (DPS), comprising statements describing critical security controls and procedures relevant for scrutinizing the trustworthiness of the system. The DPS may also identify any DNSSEC Policies (DP) it supports, explaining how it meets their requirements.

The DP and DPS are not primarily aimed at users who rely on signed responses from the DNS ("relying parties"); instead, their audience is other stakeholders of the DNS infrastructure, a group that may include bodies such as regulatory authorities.

Even though this document is heavily inspired by the Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework [RFC3647], with large parts being drawn from that document, the properties and structure of the DNSSEC trust model are fundamentally different from those of the X.509 PKI.

1.2. Purpose

The purpose of this document is twofold. Firstly, the document explains the concepts of a DNSSEC Policy (DP) and of a DNSSEC Practice Statement (DPS), and to describe the relationship between the two. Secondly, it presents a framework to encourage and assist writers of Policies and Practice Statements in creating consistent and comparable documents. In particular, the framework identifies the elements that should be considered in formulating a DP or a DPS. It does not, however, define a particular Policy or Practice Statement, nor does it seek to provide legal advice or

recommendations as to the contents.

1.3. Scope

The scope of this document is limited to discussion of the topics that can be covered in a DP or a DPS, but does not go into the specific details that could possibly be included in each one. In particular, this document describes the types of information that should be considered for inclusion in them.

The DNSSEC Policy and Practice Statement framework should be viewed and used as a checklist of factors that ought be taken in to consideration prior to deploying DNSSEC, and as an outline to create an operational practices disclosure document. As such, it focuses on the topics affected by the introduction of DNSSEC into a zone. Other aspects, such as the operations of name servers and registry systems, are considered out of scope. The framework is primarily aimed at TLD managers and organizations providing registry services, but may be used by high-value domain holders and so serve as a checklist for DNSSEC readiness at a high level.

This document assumes that the reader is familiar with the general concepts of DNS, DNSSEC and PKI.

2. Definitions

This document makes use of the following defined terms:

Audit logs - Control evidence information to prove the integrity of processes. This may be generated by DNS and DNSSEC-related systems, supplied by the surrounding facility, or obtained from manually generated, non-electronic documentation. Audit logs will be examined by the internal and/or external auditors.

Activation data - Data values, other than keys, required to operate the cryptographic modules used to protect the keys from unauthorized use.

Chain of Trust - A hierarchical structure of trust consisting of DNS keys, signatures, and delegation signer records that, when validated in a series, can provide proof of authenticity of the last element in the chain, providing that the first element is trusted. Usually, the first element is a trust anchor.

Compromise (Key Compromise) - Key Compromise is a situation where the private component of a Signing Key is lost, stolen, exposed, modified or used in an unauthorized manner. More strictly, even a suspicion

that one of these has occurred will be enough to be considered as key compromise.

DNS - The Domain Name System (DNS) is a hierarchical global naming catalog for computers, services, or any resource connected to the Internet.

DNS Zone - A portion of the global Domain Name System (DNS) namespace for which administrative responsibility has been delegated.

DNSSEC - DNS Security Extensions (DNSSEC) is a set of IETF specifications [RFC4033], [RFC4034], [RFC4035] that use public key cryptography to add data origin authentication, data integrity verification, and authenticated denial of existence capabilities to DNS.

DNSSEC Policy - A DNSSEC Policy (DP) sets forth the security requirements and standards to be implemented for a DNSSEC signed zone.

DNSSEC Practice Statement - A DNSSEC Practice Statement (DPS) is a practices disclosure document that may support and be a supplemental document to the DNSSEC Policy (if such exists), and states how the management of a given zone implements procedures and controls at a high level.

Key rollover - An operational process to change one of the DNSSEC keys used for signing a zone via distribution of public keys in a trusted manner.

Multi-person control - A security concept to distribute the authority of an operation over multiple persons, to mitigate threats caused by a single authorized individual. For example, a key recovery function may require some number of authorized individuals (m) out of the (n) to whom a portion of the recovery key was distributed, to combine their key fragments, before key recovery can occur.

PKI - Public Key Infrastructure (PKI) is a concept that makes use of asymmetric cryptography to provide a system with integrity, authentication, confidentiality and via distribution of public keys in a trusted manner.

Policy Authority - The body responsible for setting and administering a DNSSEC Policy, and for determining whether a DPS is suitable for that Policy.

Relying Party - An entity that relies on a signed response from the DNS.

Repository - A location on the Internet to store DP, DPS, Trust Anchors and other related information that should be kept public.

Security Posture - A Security Posture is an indicator of how secure an entity is and how secure the entity should be. It is the result of an adequate threat model and risk assessment.

Separation of Duties - A security concept that limits the influence of a single person by segregating roles and responsibilities.

Signing Key - Private component of an asymmetric key-pair that is used for signing of resource records within the zone. Note that the other component, called public key, is used for signature validation.

TLD - A Top-Level Domain (TLD) is one of the domains at the highest level below the root in the hierarchy of the DNS.

Trust Anchor - Public portion of a key-pair that is the authoritative entity used to authenticate the first element in a chain of trust.

3. Concepts

This section describes the concepts of a DNSSEC Policy and of a DNSSEC Practices Statement. Other related concepts are described as well.

3.1. DNSSEC Policy

A DNSSEC Policy (DP) sets forth requirements that are appropriate for a specified level of assurance. For example, a DP may encompass all topics of this framework, each with a certain set of security requirements, possibly grouped according to impact. The progression from medium to high levels of assurance would correspond to increasing security requirements and corresponding increasing levels of assurance.

A DP also constitutes a basis for an audit, accreditation, or another assessment of an entity. Each entity can be assessed against one or more DPs that it claims to implement.

3.2. DNSSEC Practice Statement

Most zone managers using DNSSEC will not have the need to create a thorough and detailed statement of practices. For example, a registrant may be the sole relying party of its own zone and would already be aware of the nature and trustworthiness of its services. In other cases, a zone manager may provide registration services with

only a very low level of assurances where the domain names being secured may pose only marginal risks if compromised. Publishing a DPS is most relevant for entities operating a zone that contains a significant number of delegations to other entities.

A DNSSEC Practice Statement (DPS) should contain information that is relevant to the stakeholders of the relevant zone(s). Since these generally include the Internet community, it should not contain such information that could be considered to be sensitive details of an entity's operations.

A DNSSEC Practice Statement may identify a supported DP, which may subsequently be used by a relying party to evaluate the trustworthiness of any digital signatures verified using the public key of that entity.

3.3. Relationship between DNSSEC Policy and Practice Statement

A DNSSEC Policy and a DNSSEC Practice Statement address the same set of topics of interest to the stakeholders in terms of the level of confidence ascribed to the security posture of a zone. The primary difference is in the focus of their provisions. A Policy sets forth the requirements and standards to be implemented for a DNSSEC signed zone, and may be used to communicate requirements that must be met by complying parties; as such it may also be used to determine or establish equivalency between policies associated with different zones. A Practice Statement, by contrast, describes how a zone operator (and possibly other participants in the management of a given zone) implements procedures and controls to meet the requirements of applicable Policies. In other words, the Policy says what needs to be done, the Practice Statement says what is being done.

An additional difference between a Policy and a Practice Statement relates to the scope of coverage of the two kinds of documents, in terms of its applicability. A Policy may apply to multiple organizations or multiple zones. By contrast, a Practice Statement would usually apply only to a single zone operator or a single organization, since it describes the actual controls in place that meet the requirements of applicable Policy.

For example, a TLD Manager or regulatory authority may define requirements in a Policy for the operation of one or more zones. The Policy will be a broad statement of the general requirements for managing the zone. A zone operator may be required to write its own Practice Statement to support the Policy, explaining how it meets the requirements of the Policy. Alternatively, a zone operator that is also the manager of that zone, and not governed by any external

Policy may still choose to disclose operational practices by publishing a DPS. The zone operator might do so to provide transparency and to gain community trust in its operations.

A Policy and a Practice Statement also differ in the level of detail each expresses: although there may be variations, a Practice Statement will provide a description of procedures and controls and so will usually be more detailed than a Policy, which provides general principles.

The main differences between a Policy and Practice Statement can be summarized as follows:

- (a) Operation of a DNS zone with DNSSEC may be governed by a Policy that establishes requirements stating what the entity operating that zone must do. An entity can use a Practice Statement to disclose how it meets the requirements of a Policy or how it has implemented critical processes and controls, absent a controlling Policy
- (b) A Policy may serve the purpose of establishing a common basis of trusted operation throughout a set of zones in the DNS hierarchy. By contrast, a Practice Statement is a statement of a single zone operator or organization.
- (c) A Practice Statement is generally more detailed than a Policy and specifies how the zone operator or organization implements critical processes and controls, and how the entity meets any requirements specified in the one or more Policies under which it operates DNSSEC.

3.4. Set of Provisions

A set of provisions is a collection of Policy requirements or Practice statements, which may employ the approach described in this framework by covering the topics appearing in Section 5 below. The topics are described in detail in section 4.

A Policy can be expressed as a single set of provisions.

A Practice Statement can also be expressed as a single set of provisions with each component addressing the requirements of one or more Policies. Alternatively, it could be a set of provisions that do not reference any particular policy but instead describe a set of self-imposed controls to the stakeholders. For example, a Practice Statement could be expressed as a combination of the following:

- (a) a list of Policies supported by the DPS;
- (b) for each Policy in (a), a set of provisions that contains statements addressing the requirements by filling in details not stipulated in that policy or expressly left to the discretion of the implementer. Such statements serve to show how this particular Practice Statement implements the requirements of the particular Policy; or
- (c) a set of provisions that contains statements regarding the DNSSEC operations practices, independent of any Policy.

The statements provided in (b) may augment or refine the stipulations of an applicable Policy, but generally they must not conflict with the stipulations. In certain cases however, a Policy Authority may permit exceptions because certain compensating controls of the entity disclosed in its Practices Statement allow it to provide a level of assurance equivalent to full compliance with the policy.

The framework outlines the contents of a set of provisions, in terms of eight primary components, as follows:

1. Introduction
2. Publication and Repositories
3. Operational Requirements
4. Facility, Management, and Operational Controls
5. Technical Security Controls
6. Zone Signing
7. Compliance Audit
8. Legal Matters

This framework can be used by Policy Authorities to write DNSSEC Policies and by zone operators to write a DNSSEC Practice Statements. Having a set of documents with the same structure facilitates comparisons with the corresponding documents of other zones.

4. Contents of a Set of Provisions

This section describes the contents of a set of provisions. Refer to Section 5 for the complete outline.

Drafters of DPSs conforming to this framework are permitted to add additional levels of subcomponents below those described here to meet specific needs. All components listed in Section 5 should be present, but drafters may leave components empty, only stating "no stipulation", if so required.

4.1. Introduction

This component identifies and introduces the set of provisions, and indicates the types of entities and applications for which the document (either Policy or Practice Statement) is targeted.

4.1.1. Overview

This subcomponent provides a general introduction to the document. It can also be used to provide a description of entities to which the Policy or Practice Statement applies.

4.1.2. Document name and identification

This subcomponent provides any applicable names or other identifiers of the document.

4.1.3. Community and applicability

This subcomponent identifies the stakeholders along with their expected roles and responsibilities. These include (but are not limited to) an entity signing the zone, entities relying on the signed zone, other entities that have operational dependency on the signed zone, and an entity that entrusted the zone signing.

4.1.4. Specification administration

This subcomponent contains the contact details of the organization responsible for managing the DP/DPS, as well as the specification change procedures. These procedures may include the description of the notification mechanisms used to provide advance notice of amendments which are deemed to materially affect the assurance provided by the entity, and how/when such amendments will be communicated to the stakeholders.

If a Policy Authority is responsible for determining whether a DPS is suitable for the Policy, this subcomponent may include the name and

contact information of the entity in charge of making such a determination. In this case, the subcomponent also includes the procedures by which this determination is made.

4.2. Publication and repositories

The component describes the requirements for an entity to publish information regarding its practices, public keys, the current status of such keys together with details relating to the repositories in which the information is held. This may include the responsibilities of publishing the DPS and of identifying documents that are not made publicly available owing to their sensitive nature, e.g. security controls, clearance procedures, or business information.

4.2.1. Repositories

This subcomponent describes the repository mechanisms used for making information available to the stakeholders, and may include:

- o The locations of the repositories and the means by which they may be accessed;
- o An identification of the entity or entities that operate repositories, such as a zone operator or a TLD Manager;
- o Access control on published information objects.
- o Any notification services which may be subscribed to by the stakeholders;

4.2.2. Publication of public keys

This subcomponent contains information relating to the publication of public keys:

- o Whether the public keys are included in a key hierarchy, published as Trust Anchors or both;
- o The data formats and methods available to validate the authenticity of public keys;
- o The frequency and timing of publishing new information (principally as advance notice for stakeholders relying on the public keys).

4.3. Operational requirements

This component describes the operational requirements when operating a DNSSEC signed zone.

4.3.1. Meaning of domain names

This subcomponent describes the overall policy of child zone naming, if any.

4.3.2. Identification and authentication of child zone manager

This subcomponent describes how the child zone manager has initially been identified, and how any subsequent change request is authenticated as originating from the manager or their authorized representative.

4.3.3. Registration of delegation signer (DS) resource records

This subcomponent describes the process of establishing the chain-of-trust to the child zone by incorporating delegation signer (DS) record(s) into the zone.

4.3.4. Method to prove possession of private key

This subcomponent describes whether, and if so under what circumstances, the child zone manager is required to provide proof of the possession of the private component of any current or subsequent child zone Signing Key corresponding to a DS record they wish to incorporate into the parent zone.

4.3.5. Removal of DS resource records

This subcomponent will explain how, when, and under what circumstances the DS records may be removed from the zone.

4.4. Facility, management and operational controls

This component describes non-technical security controls (i.e., physical, procedural, and personnel) in use by the entity to securely perform the DNSSEC related functions. Such controls include physical access, key management, disaster recovery, auditing, and archiving.

These non-technical security controls are critical for trusting the DNSSEC signatures, since lack of security may compromise DNSSEC operations. For example, it could result in the creation of signatures with erroneous information or in the compromise of the Signing Key.

Within each subcomponent, separate consideration will usually need to be given to each entity type.

4.4.1. Physical controls

In this subcomponent, the physical controls on the facility housing the entity systems are described. Topics addressed may include:

- o Site location and construction, such as requirements for multiple tiers of physical barriers, construction requirements for high-security areas etc. It may also describe the use of locked rooms, cages, safes, and cabinets etc.;
- o Physical access, i.e. mechanisms to control access from one area of the facility to another or additional controls for reaching into higher tiers, such as dual-access control and two-factor authentication;
- o Power and air conditioning;
- o Water exposures;
- o Fire prevention and protection;
- o Media storage, e.g. requiring the storage of backup media in a separate location that is physically secure and protected from fire, smoke, particle, and water damage;
- o Waste disposal; and
- o Off-site backup.

4.4.2. Procedural controls

In this subcomponent, requirements for recognizing trusted roles are described, together with a description of the responsibilities of each role. Examples of trusted roles include system administrators, security officers, crypto officers and system auditors.

For each task identified, the number of individuals required to perform the task (m of n rule, if applicable) should be stated for each role. Identification and authentication requirements for each role may also be defined.

This subcomponent also includes the separation of duties in terms of the roles that cannot be performed by the same individuals.

4.4.3. Personnel controls

This subcomponent addresses the following:

- o Qualifications, experience, and clearances that personnel must have as a condition of filling trusted roles or other important roles. Examples include credentials, job experiences, and official government clearances;
- o Background checks and clearance procedures that are required in connection with the hiring of personnel filling trusted roles or other important roles. Such roles may require a check of their criminal records, financial records, references, and any additional clearances required for the position in question;
- o Training requirements and training procedures for each role following the hiring of personnel;
- o Any retraining period and retraining procedures for each role after completion of initial training;
- o Frequency and sequence for job rotation among various roles;
- o Sanctions against personnel for unauthorized actions, such as unauthorized use of authority, or unauthorized use of the entity systems;
- o Controls on personnel that are contractors rather than employees of the entity; examples include:
 - * Bonding requirements on contract personnel;
 - * Contractual requirements including indemnification for damages due to the actions of the contractor personnel;
 - * Auditing and monitoring of contractor personnel; and
 - * Other controls on contracting personnel.
- o Documentation to be supplied to personnel during initial training, retraining, or otherwise.

4.4.4. Audit logging procedures

This subcomponent is used to describe event logging and audit systems, implemented for the purpose of maintaining an audit trail and to provide evidence of process integrity. Elements include the following:

- o Types of events recorded, such as records of key roll over and other key management operations, the personnel assigned to various roles, attempts to access the system and requests made to the system;
- o Frequency with which audit logs are processed or archived, for example, weekly, following an alarm or anomalous event, or whenever the audit log size reaches a particular size;
- o Period for which audit logs are kept;
- o Protection of audit logs:
 - * Who can view audit logs, for example only the audit administrator;
 - * Protection against modification of audit logs, for instance a requirement that no one may modify or delete the audit records or that only an audit administrator may delete an audit file as part of audit file rotation; and
 - * Protection against deletion of audit logs.
- o Audit log backup procedures;
- o Whether the audit log collection function is internal or external to the system;
- o Whether the subject who caused an audit event to occur is notified of the audit action; and
- o Vulnerability assessments, for example, where audit data is run through a tool that identifies potential attempts to breach the security of the system.

4.4.5. Compromise and disaster recovery

This subcomponent describes requirements relating to notification and recovery procedures in the event of compromise or disaster. Each of the following may need to be addressed separately:

- o Identification or listing of the applicable incident and compromise reporting and handling procedures, which may include the investigation of measures to prevent the event from reoccurring.
- o The recovery procedures used if computing resources, software, and/or data are corrupted or suspected to have been corrupted. These procedures describe how, and under what circumstances operations of the system are to be suspended, how and when normal operations are resumed, how the stakeholders are to be informed and how to assess the damage and carry out the root cause analysis.
- o The recovery procedures used if any keys are compromised. These procedures describe how a secure environment is re-established, how the keys are rolled over, how a new Trust Anchor is provided to the community (if applicable), and how new zone information is published.
- o The entity's capabilities to ensure business continuity following a natural or other disaster. Such capabilities may include the availability of a disaster recovery site at which operations may be recovered. They may also include procedures for securing its facility during the period of time following a natural or other disaster and before a secure environment is re-established, either at the original site or at a disaster recovery site. For example, procedures to protect against theft of sensitive materials from an earthquake-damaged site.

4.4.6. Entity termination

This subcomponent describes requirements relating to procedures for termination of a contract with an entity, termination notification and transition of responsibilities to another entity. The purpose may be to ensure that the transition process will be transparent to the stakeholders, and will not affect the services.

4.5. Technical security controls

This component is used to define the security measures taken to protect the cryptographic keys and activation data (e.g., PINs, passwords, or manually-held key shares) relevant to DNSSEC operations. Secure key management is critical to ensure that all secret and private keys and activation data are protected and used only by authorized personnel.

Also described here are other technical security controls used to perform the functions of key generation, authentication, registration, auditing, and archiving. Technical controls include life-cycle security controls, software development environment security, and operational security controls.

If applicable, other technical security controls on repositories, authoritative name servers, or other participants may also be documented here.

4.5.1. Key pair generation and installation

Key pair generation and installation need to be considered, which may involve answering the following questions:

1. Who generates the zone's public/private key pairs? How is the key generation performed? Is the key generation performed by hardware or software?
2. How is the private key installed in all parts of the key management system?
3. How are the zone's public keys provided securely to the parent zone and potential relying parties?
4. Who generates the public key parameters. Is the quality of the parameters checked during key generation?
5. For what purposes may the keys be used, and/or for what purposes should usage of the key be restricted?

4.5.2. Private key protection and cryptographic module engineering controls

Requirements for private key protection and cryptographic modules need to be considered for key generation and creation of signatures. The following questions may need to be answered:

1. What standards, if any, are required for the cryptographic module used to generate the keys? A cryptographic module can be composed of hardware, software, firmware, or any combination of them. For example, are the zone's signatures required to be generated using modules compliant with the US FIPS 140-2 standard? If so, what is the required FIPS 140-2 level of the module? Are there any other engineering or other controls relating to a cryptographic module, such as the identification of the cryptographic module boundary, input/output, roles and services, finite state machine, physical security, software security, operating system security, algorithm compliance, electromagnetic compatibility, and self tests.
2. Is the private key under m of n multi-person control? If yes, provide m and n (two person control is a special case of m of n , where $m = 2$ and $n \geq 2$).
3. Is the private key escrowed? If so, who is the escrow agent, in what form is the key escrowed (e.g. plaintext, encrypted, split key), and what are the security controls on the escrow system?
4. Is the private key backed up? If so, who is the backup agent, in what form is the key backed up (e.g. plaintext, encrypted, split key), and what are the security controls on the backup system?
5. Is the private key archived? If so, who is the archival agent, in what form is the key archived (e.g. plaintext, encrypted, split key), and what are the security controls on the archival system?
6. Under what circumstances, if any, can a private key be transferred into or from a cryptographic module? Who is permitted to perform such a transfer operation? In what form is the private key during the transfer (e.g., plaintext, encrypted, or split key)?
7. How is the private key stored in the module (e.g., plaintext, encrypted, or split key)?
8. Who can activate (use) the private key? What actions must be performed to activate the private key (e.g., login, power on, supply PIN, insert token/key, automatic, etc.)? Once the key is activated, is the key active for an indefinite period, active for one time, or active for a defined time period?

9. Who can deactivate the private key and how? Examples of methods of deactivating private keys include logging out, turning the power off, removing the token/key, automatic deactivation, and time expiration.
10. Who can destroy the private key and how? Examples of methods of destroying private keys include token surrender, token destruction, and zeroizing the key.

4.5.3. Other aspects of key pair management

Other aspects of key management need to be considered for the zone operator and other participants. For each of these types of entities, the following questions may need to be answered:

1. What are the life-cycle states for the management of any Signing Keys?
2. What is the operational period of these keys? What are the usage periods or active lifetimes for the pairs?

4.5.4. Activation data

Activation data refers to data values other than whole private keys that are required to operate private keys or cryptographic modules containing private keys, such as a PIN, passphrase, or portions of a private key used in a key-splitting scheme. Protection of activation data prevents unauthorized use of the private key and potentially needs to be considered for the zone operator and other participants. Such a consideration may need to address the entire life-cycle of the activation data from generation through archival and destruction. For each of the entity types, all of the questions listed in 4.5.1 through 4.5.3 potentially need to be answered with respect to activation data rather than with respect to keys.

4.5.5. Computer security controls

This subcomponent is used to describe computer security controls such as:

1. use of the trusted computing base concept or equivalent;

2. discretionary access control, labels, mandatory access controls;
3. object re-use;
4. auditing;
5. identification and authentication;
6. trusted path; and
7. security testing.

This subcomponent may also address requirements for product assurance, product evaluation analysis, testing, profiling, product certification, and/or product accreditation.

4.5.6. Network security controls

This subcomponent addresses network security related controls, including firewalls, routers and remote access.

4.5.7. Timestamping

This subcomponent addresses requirements or practices relating to the use of timestamps on various data. It may also discuss whether or not the time-stamping application must use a trusted time source.

4.5.8. Life cycle technical controls

This subcomponent addresses system development controls and security management controls.

System development controls include development environment security, development personnel security, configuration management security during product maintenance, software engineering practices, software development methodology, modularity, layering, use of failsafe design and implementation techniques (e.g. defensive programming) and development facility security.

Security management controls include execution of tools and procedures to ensure that the operational systems and networks adhere to configured security. These tools and procedures include checking the integrity of the security software, firmware, and hardware to ensure their correct operation.

4.6. Zone signing

This component covers all aspects of zone signing, including the cryptographic specification surrounding the Signing Keys, signing scheme, and methodology for key rollover and the actual zone signing. Child zones and other relying parties may depend on the information in this section to understand the expected data in the signed zone and determine their own behavior. In addition, this section will be used to state the compliance to the cryptographic and operational requirements pertaining to zone signing, if any.

4.6.1. Key lengths, key types and algorithms

This subcomponent describes the key generation algorithm, the key types used for signing the key set and zone data, and key lengths used to create the keys. It should also cover how changes to these key lengths, key types and algorithms may be performed.

4.6.2. Authenticated denial of existence

Authenticated denial of existence refers to the usage of NSEC [RFC4034], NSEC3 [RFC5155] or any other mechanism defined in the future that is used to authenticate the denial of existence of resource records. This subcomponent describes what mechanisms are used, any parameters associated with that mechanism, and how these mechanisms and parameters may be changed.

4.6.3. Signature format

This subcomponent is used to describe the signing method and algorithms used for the zone signing.

4.6.4. Key rollover

This subcomponent explains the key rollover scheme for each key type.

4.6.5. Signature life-time and re-signing frequency

This subcomponent describes the life-cycle of the Resource Record Signature (RRSIG) record.

4.6.6. Verification of resource records

This subsection addresses the controls around the verification of the resource records in order to validate and authenticate the data to be signed. This may include a separate key set verification process if using a split key signing scheme.

4.6.7. Resource records time-to-live

This subcomponent specifies the resource records' time-to-live (TTL) for all types relevant to DNSSEC, as well as any global parameters that affect the caching mechanisms of the resolvers.

4.7. Compliance audit

To prove the compliance with a Policy or the statements in the Practices Statement, a compliance audit can be conducted. This component describes how the audit is to be conducted at the zone operator and, possibly, at other involved entities.

4.7.1. Frequency of entity compliance audit

This subcomponent describes the frequency of the compliance audit.

4.7.2. Identity/qualifications of auditor

This subcomponent addresses what qualifications are required of the auditor. For instance it may be that an auditor must belong to a specific association or that they have certain certifications.

4.7.3. Auditor's relationship to audited party

This subcomponent is used to clarify the relationship between the auditor and the entity being audited. This becomes important if there are any requirements or guidelines for the selection of the auditor.

4.7.4. Topics covered by audit

Topics covered by audit depends on the scope of the audit. Since the DNSSEC Policy and Practices Statement is the document to be audited against, it is ideal to set the scope of the audit to the scope of the DP/DPS. However, the scope may be narrowed down or expanded as needed; for example, if there are not enough resources to conduct a full audit, or some portion is under development and not ready for the audit.

4.7.5. Actions taken as a result of deficiency

This subcomponent specifies the action taken in order to correct any discrepancy that has a security impact. This could be the remediation process for the audit findings or any other action to correct any discrepancy with the DNSSEC Policy or Practices Statement.

4.7.6. Communication of results

This subcomponent specifies how the results of the audit are communicated to the stakeholders.

4.8. Legal matters

The introduction of DNSSEC into a zone may have legal implications. Consequently, it may be appropriate to declare the legal status of the binding embodied in the DNSSEC digital signatures and to clarify on any limitations of liability asserted by the Registry Manager.

In most cases, the DPS is not a contract or part of a contract; instead, it is laid out so that its terms and conditions are applied to the parties by separate documents, such as registrar or registrant agreements. In other cases, its contents may form part of a legal contract between parties (either directly or via other agreements). In this case, legal expertise should be consulted when drawing up sections of the document that may have contractual implications.

At a minimum, the legal matters section should indicate under what jurisdiction the registry is operated, and provide references to any associated agreements that are in force. It may also be appropriate to inform of any identified implications on the protection of personally identifiable private information.

5. Outline of a Set of Provisions

This section contains a recommended outline for a set of provisions, intended to serve as a checklist or a standard template for use by DP or DPS writers. Such a common outline will facilitate:

- (a) Comparison of a DPS with a DP to ensure that the DPS faithfully implements the policy.
- (b) Comparison of two DPSs.

Section 4 of this document is structured so that it provides guidance for each corresponding component and sub-component of the outline.

1. INTRODUCTION

- 1.1. Overview
- 1.2. Document name and identification
- 1.3. Community and applicability
- 1.4. Specification administration

- 1.4.1. Specification administration organization
- 1.4.2. Contact information
- 1.4.3. Specification change procedures
- 2. PUBLICATION AND REPOSITORIES
 - 2.1. Repositories
 - 2.2. Publication of public keys
- 3. OPERATIONAL REQUIREMENTS
 - 3.1. Meaning of domain names
 - 3.2. Identification and authentication of child zone manager
 - 3.3. Registration of delegation signer (DS) resource records
 - 3.4. Method to prove possession of private key
 - 3.5. Removal of DS resource records
 - 3.5.1. Who can request removal
 - 3.5.2. Procedure for removal request
 - 3.5.3. Emergency removal request
- 4. FACILITY, MANAGEMENT AND OPERATIONAL CONTROLS
 - 4.1. Physical controls
 - 4.1.1. Site location and construction
 - 4.1.2. Physical access
 - 4.1.3. Power and air conditioning
 - 4.1.4. Water exposures
 - 4.1.5. Fire prevention and protection
 - 4.1.6. Media storage
 - 4.1.7. Waste disposal
 - 4.1.8. Off-site backup
 - 4.2. Procedural controls
 - 4.2.1. Trusted roles
 - 4.2.2. Number of persons required per task
 - 4.2.3. Identification and authentication for each role
 - 4.2.4. Tasks requiring separation of duties
 - 4.3. Personnel controls
 - 4.3.1. Qualifications, experience, and clearance requirements
 - 4.3.2. Background check procedures
 - 4.3.3. Training requirements
 - 4.3.4. Job rotation frequency and sequence
 - 4.3.5. Sanctions for unauthorized actions
 - 4.3.6. Contracting personnel requirements
 - 4.3.7. Documentation supplied to personnel
 - 4.4. Audit logging procedures
 - 4.4.1. Types of events recorded
 - 4.4.2. Frequency of processing log
 - 4.4.3. Retention period for audit log information
 - 4.4.4. Protection of audit log
 - 4.4.5. Audit log backup procedures
 - 4.4.6. Audit collection system
 - 4.4.7. Vulnerability assessments
 - 4.5. Compromise and disaster recovery

- 4.5.1. Incident and compromise handling procedures
- 4.5.2. Corrupted computing resources, software, and/or data
- 4.5.3. Entity private key compromise procedures
- 4.5.4. Business continuity and IT disaster recovery capabilities
- 4.6. Entity termination
- 5. TECHNICAL SECURITY CONTROLS
 - 5.1. Key pair generation and installation
 - 5.1.1. Key pair generation
 - 5.1.2. Public key delivery
 - 5.1.3. Public key parameters generation and quality checking
 - 5.1.4. Key usage purposes
 - 5.2. Private key protection and cryptographic module engineering controls
 - 5.2.1. Cryptographic module standards and controls
 - 5.2.2. Private key (m-of-n) multi-person control
 - 5.2.3. Private key escrow
 - 5.2.4. Private key backup
 - 5.2.5. Private key storage on cryptographic module
 - 5.2.6. Private key archival
 - 5.2.7. Private key transfer into or from a cryptographic module
 - 5.2.8. Method of activating private key
 - 5.2.9. Method of deactivating private key
 - 5.2.10. Method of destroying private key
 - 5.3. Other aspects of key pair management
 - 5.4. Activation data
 - 5.4.1. Activation data generation and installation
 - 5.4.2. Activation data protection
 - 5.4.3. Other aspects of activation data
 - 5.5. Computer security controls
 - 5.6. Network security controls
 - 5.7. Timestamping
 - 5.8. Life cycle technical controls
- 6. ZONE SIGNING
 - 6.1. Key lengths, key types and algorithms
 - 6.2. Authenticated denial of existence
 - 6.3. Signature format
 - 6.4. Key rollover
 - 6.5. Signature life-time and re-signing frequency
 - 6.6. Verification of resource records
 - 6.7. Resource records time-to-live
- 7. COMPLIANCE AUDIT
 - 7.1. Frequency of entity compliance audit
 - 7.2. Identity/qualifications of auditor
 - 7.3. Auditor's relationship to audited party

- 7.4. Topics covered by audit
- 7.5. Actions taken as a result of deficiency
- 7.6. Communication of results
- 8. LEGAL MATTERS

6. IANA considerations

No action required.

7. Security considerations

The sensitivity of the information protected by DNSSEC at different tiers in the DNS tree varies significantly. In addition, there are no restrictions as to what types of information (i.e., DNS records) that can be protected using DNSSEC. Each relying party must evaluate its own environment and the chain of trust originating from a Trust Anchor, the associated threats and vulnerabilities, to determine the level of risk it is willing to accept when relying on DNSSEC-protected objects.

8. Acknowledgements

This document is inspired by RFC 3647 and its predecessor (RFC 2527), and the authors acknowledge the work in the development of these documents.

In addition, the authors would like to acknowledge the contributions made by Richard Lamb, Jakob Schlyter and Stephen Morris.

9. References

9.1. Normative 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.

9.2. Informative references

- [RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, November 2003.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

Authors' Addresses

Fredrik Ljunggren
Kirei AB
P.O. Box 53204
Goteborg SE-400 16
Sweden

Email: fredrik@kirei.se

Anne-Marie Eklund Lowinder
.SE (The Internet Infrastructure Foundation)
P.O. Box 7399
Stockholm SE-103 91
Sweden

Email: amel@iis.se

Tomofumi Okubo
Internet Corporation For Assigned Names and Numbers
4676 Admiralty Way, Suite 330
Marina del Ray, CA 90292
USA

Email: tomofumi.okubo@icann.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

S. Morris
ISC
J. Ihren
Netnod
J. Dickinson
Sinodun
July 9, 2012

DNSSEC Key Timing Considerations
draft-ietf-dnsop-dnssec-key-timing-03.txt

Abstract

This document describes the issues surrounding the timing of events in the rolling of a key in a DNSSEC-secured zone. It presents timelines for the key rollover and explicitly identifies the relationships between the various parameters affecting the process.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Key Rolling Considerations	3
1.2. Types of Keys	4
1.3. Terminology	4
1.4. Requirements Language	4
2. Rollover Methods	5
2.1. ZSK Rollovers	5
2.2. KSK Rollovers	6
2.3. Summary	7
3. Key Rollover Timelines	8
3.1. Key States	8
3.2. Zone-Signing Key Timelines	9
3.2.1. Pre-Publication Method	9
3.2.2. Double-Signature Method	12
3.2.3. Double-RRSIG Method	13
3.3. Key-Signing Key Rollover Timelines	15
3.3.1. Double-Signature Method	16
3.3.2. Double-DS Method	18
3.3.3. Double-RRset Method	21
3.3.4. Interaction with Configured Trust Anchors	23
3.3.5. Introduction of First Keys	24
4. Standby Keys	25
5. Algorithm Considerations	26
6. Limitation of Scope	26
7. Summary	26
8. IANA Considerations	27
9. Security Considerations	27
10. Acknowledgements	27
11. Normative References	27
Appendix A. List of Symbols	28
Appendix B. Change History (To be removed on publication)	31
Authors' Addresses	33

1. Introduction

1.1. Key Rolling Considerations

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

- o DNSKEY records and associated information (such as the associated DS records or RRSIG records created with the key) are not only held at the authoritative nameserver, they are also cached by resolvers. The data on these systems can be interlinked, e.g., a validating resolver may try to validate a signature retrieved from a cache with a key obtained separately.
- o 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.
- o To allow for an emergency re-signing of the zone as soon as possible after a key compromise has been detected, standby keys (additional keys over and above those used to sign the zone) need to be present.
- o A query for the DNSKEY RRset returns all DNSKEY records in the zone. As there is limited space in the UDP packet (even with EDNS0 support), key records no longer needed must be periodically removed. (For the same reason, the number of standby keys in the zone should be restricted to the minimum required to support the key management policy.)

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

A high-level overview of key rollover can be found in [I-D.ietf-dnsop-rfc4641bis]. In contrast, this document focuses on the low-level timing detail of two classes of operations described there, the rollover of key-signing keys, and the rollover of zone signing keys.

1.2. Types of Keys

Although DNSSEC validation treats all keys equally, [RFC4033] recognises the broad classification of zone-signing keys (ZSK) and key-signing keys (KSK). A ZSK is used to authenticate information within the zone; a KSK is used to authenticate the zone's DNSKEY RRset. The main implication for this distinction concerns the consistency of information during a rollover.

During operation, a validating resolver must use separate pieces of information to perform an authentication. At the time of authentication, each piece of information may be in its cache or may need to be retrieved from the authoritative server. The rollover process needs to happen in such a way that at all times during the rollover the information is consistent. With a ZSK, the information is the RRSIG (plus associated RRset) and the DNSKEY. These are both obtained from the same zone. In the case of the KSK, the information is the DNSKEY and DS RRset with the latter being obtained from a different zone.

Although there are similarities in the algorithms to roll ZSKs and KSKs, there are a number of differences. For this reason, the two types of rollovers are described separately. It is also possible to use a single key as both the ZSK and KSK. However, the rolling of this type of key is not treated in this document.

1.3. Terminology

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

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

1.4. 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. Rollover Methods

2.1. ZSK Rollovers

A ZSK can be rolled in one of three ways:

- o Pre-Publication: described in [I-D.ietf-dnsop-rfc4641bis], the new key is introduced into the DNSKEY RRset which is then re-signed. This state of affairs remains in place for long enough to ensure that any cached DNSKEY RRsets contain both keys. At that point signatures created with the old key can be replaced by those created with the new key, and the old signatures removed. During the re-signing process (which may or may not be atomic depending on how the zone is managed), it doesn't matter which key an RRSIG record retrieved by a resolver was created with; cached copies of the DNSKEY RRset will contain 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 caches. After this, there will be no signatures anywhere that were created using the old key, and it can be removed from the DNSKEY RRset.

- o Double-Signature: also mentioned in [I-D.ietf-dnsop-rfc4641bis], this involves introducing the new key into the zone and using it to create additional RRSIG records; the old key and existing RRSIG records are retained. During the period in which the zone is being signed (again, the signing process may not be atomic), validating resolvers are always able to validate RRSIGs: any combination of old and new DNSKEY RRset and RRSIG allows at least one signature to be validated.

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

- o Double-RRSIG: strictly speaking, the use of the term "Double-Signature" above is a misnomer as the method is not only double signature, it is also double key as well. A true Double-Signature method (here called the Double-RRSIG method) involves introducing new signatures in the zone (while still retaining the old ones) but not introducing the new key.

Once the signing process is complete and enough time has elapsed to ensure that all caches that may contain an RR and associated RRSIG have a copy of both signatures, the key is changed. After a

further interval during which the old DNSKEY RRset expires from caches, the old signatures are removed from the zone.

Of three methods, Double-Signature is conceptually the simplest - introduce the new key and new signatures, then approximately one TTL later remove the old key and old signatures. Pre-Publication is more complex - introduce the new key, approximately one TTL later sign the records, and approximately one TTL after that remove the old key. Double-RRSIG is essentially the reverse of Pre-Publication - introduce the new signatures, approximately one TTL later change the key, and approximately one TTL after that remove the old signatures.

2.2. KSK Rollovers

For ZSKs, the issue for the validating resolver is to ensure that it has access to the ZSK that corresponds to a particular signature. In the KSK case, this can never be a problem as the KSK is only used for one signature (that over the DNSKEY RRset) and both the key 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 signed and validated DS record in the parent zone or an explicitly configured trust anchor. If the former, the rollover algorithm will need to involve the parent zone in the addition and removal of DS records, so timings are not wholly under the control of the zone manager. If the latter, [RFC5011] timings will be needed to roll the keys. (Even in the case where authentication is via a DS record, the zone manager may elect to include [RFC5011] timings in the key rolling process so as to cope with the possibility that the key has also been explicitly configured as a trust anchor.)

It is important to note that this does not preclude the development of key rollover logic; in accordance with the goal of the rollover logic being able to determine when a state change is "safe", the only effect of being dependent on the parent is that there may be a period of waiting for the parent to respond in addition to any delay the key rollover logic requires. Although this introduces additional delays, even with a parent that is less than ideally responsive the only effect will be a slowdown in the rollover state transitions. This may cause a policy violation, but will not cause any operational problems.

Like the ZSK case, there are three methods for rolling a KSK:

- o Double-Signature: also known as Double-DNSKEY, the new KSK is added to the DNSKEY RRset which is then signed with both the old and new key. After waiting for the old RRset to expire from

caches, the DS record in the parent zone is changed. After waiting a further interval for this change to be reflected in caches, the old key is removed from the RRset. (The name "Double-Signature" is used because, like the ZSK method of the same name, the new key is introduced and immediately used for signing.)

- o Double-DS: the new DS record is published. After waiting for this change to propagate into caches, the KSK is changed. After a further interval during which the old DNSKEY RRset expires from caches, the old DS record is removed.
- o 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 caches, the old DNSKEY and DS record are removed.

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

2.3. Summary

The methods can be summarised as follows:

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

Table 1

3. Key Rollover Timelines

3.1. Key States

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

Generated	The key has been created, but has not yet been used for anything.
Published	<p>The DNSKEY record - or information associated with it - is published in the zone, but predecessors of the key (or associated information) may be held in caches.</p> <p>The idea of "associated information" is used in rollover methods where RRSIG or DS records are published first and the DNSKEY is changed in an atomic operation. It allows the rollover still to be thought of as moving through a set of states. In the rest of this section, the term "key data" should be taken to mean "key or associated information".</p>
Ready	The new key data has been published for long enough to guarantee that any previous versions of the DNSKEY RRset have expired from caches.
Active	The key has started to be used to sign RRsets. Note that when this state is entered, it may not be possible for validating resolvers to use the key for validation in all cases: the zone signing may not have finished, or the data might not have reached the resolver because of propagation delays and/or caching issues. If this is the case, the resolver will have to rely on the key's predecessor instead.
Retired	A successor key has become active and this key is no longer being used to generate RRSIGs. However, as there may still be RRSIGs in caches that were generated using this key, it is being retained in the zone until they have expired.
Dead	The key is published in the zone but there is no longer information anywhere that requires its presence. Hence the key can be removed from the zone at any time.

Removed The key has been removed from the zone.

There is one additional state, used where [RFC5011] considerations are in effect (see Section 3.3.4):

Revoked The key is published for a period with the "revoke" bit set as a way of notifying validating resolvers that have configured it as an [RFC5011] trust anchor that it is about to be removed from the zone.

3.2. Zone-Signing Key Timelines

The following sections describe the rolling of a ZSK. They show the events in the lifetime of a key (referred to as "key N") and cover its replacement by its successor (key N+1).

3.2.1. Pre-Publication Method

The following diagram shows the timeline of a Pre-Publication rollover. Time increases along the horizontal scale from left to right and the vertical lines indicate events in the process. Significant times and time intervals are marked.

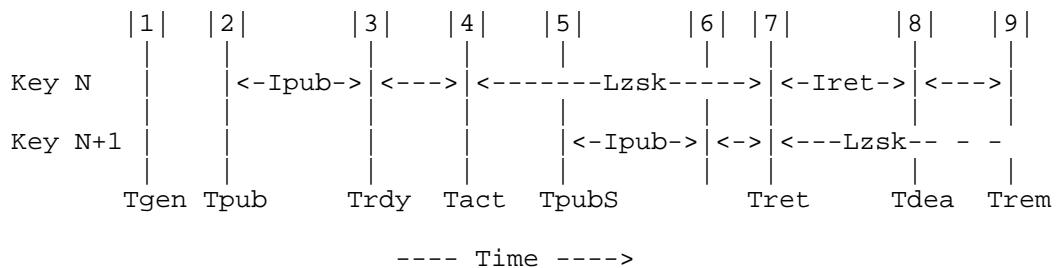


Figure 1: Timeline for a Pre-Publication ZSK rollover.

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 its 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 (T_{pub}), and the key is now said to be published. Note that the key is not yet used to sign records.

Event 3: Before it can be used, the key must be published for long enough to guarantee that any cached version of the zone's DNSKEY RRset includes this key.

This interval is the publication interval (I_{pub}) and, for the second or subsequent keys in the zone, is given by:

$$I_{pub} = D_{prp} + TTL_{key}$$

Here, D_{prp} is the propagation delay - the time taken in the worst-case situation for a change introduced at the master to replicate to all slave servers - which depends on the depth of the master-slave hierarchy. TTL_{key} is the time-to-live (TTL) for the DNSKEY records in the zone. The sum is therefore the maximum time taken for existing DNSKEY records to expire from caches, regardless of the nameserver from which they were retrieved.

(The case of introducing the first ZSK into the zone is discussed in Section 3.3.5.)

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

$$Trdy = T_{pub} + I_{pub}$$

Event 4: At some later time, the key starts being used to sign RRsets. This point is the activation time (T_{act}) and after this, the key is said to be active.

Event 5: At some point thought must be given to its successor (key $N+1$). As with the introduction of the currently active key into the zone, the successor key will need to be published at least I_{pub} before it is activated. Denoting the publication time of the successor key by T_{pubS} , then:

$$T_{pubS} \leq T_{act} + L_{zsk} - I_{pub}$$

Here, L_{zsk} is the length of time for which a ZSK will be used (the ZSK lifetime). It should be noted that unlike the publication interval, L_{zsk} is not determined by timing logic, but by key management policy. L_{zsk} will be set by the operator according to their assessment of the risks posed by continuing to use a key and the risks associated with key rollover. However, operational considerations may mean a key is active for slightly more or less

than $Lzsk$.

Event 6: While key N is still active, its successor becomes ready. From this time onwards, key $N+1$ could be used to sign the zone.

Event 7: When key N has been in use for an interval equal to the ZSK lifetime, it is retired (i.e., it will never again be used to generate new signatures) and key $N+1$ activated and used to sign the zone. This is the retire time of key N ($Tret$) and is given by:

$$Tret = Tact + Lzsk$$

It is also the activation time of the successor key ($TactS$). Note that operational considerations may cause key N to remain in use for longer than $Lzsk$; if so, the retirement actually occurs when the successor key is made 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 caches. (It is possible that a validating 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 maximum TTL of all the RRSIG records in the zone created with the ZSK.

The time at which all RRSIG records created with this key have expired from resolver caches is the dead time ($Tdea$), given by:

$$Tdea = Tret + Iret$$

... at which point the key is said to be dead.

Event 9: At any time after the key becomes dead, it can be removed from the zone's DNSKEY RRset, which must then be re-signed with the current key-signing key. This time is the removal time ($Trem$), given by:

$T_{rem} \geq T_{dea}$

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

3.2.2. Double-Signature Method

The timeline for a double-signature rollover is shown below. The diagram follows the convention described in Section 3.2.1

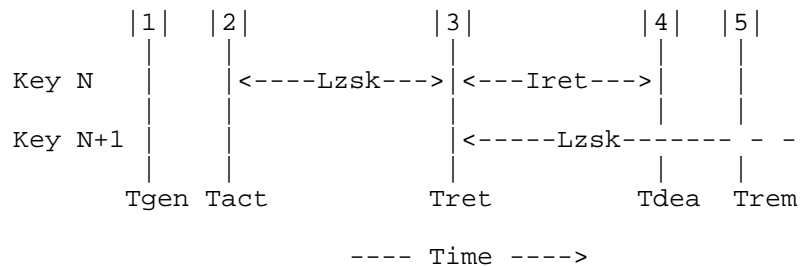


Figure 2: Timeline for a Double-Signature ZSK rollover.

Event 1: Key N is generated at the generate time (T_{gen}). Although there is no reason why the key cannot be generated immediately prior to its publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.

Event 2: Key N is added to the DNSKEY RRset and is then used to sign the zone; existing signatures in the zone are not removed. This is the activation time (T_{act}), after which the key is said to be active.

Event 3: After the current key (key N) has been in use for its intended lifetime ($Lzsk$), the successor key (key N+1) is introduced into the zone and starts being used to sign RRsets: neither the current key nor the signatures created with it are removed. The successor is key is now active and the current key is said to be retired. This time is the retire time of the key (T_{ret}); it is also the activation time of the successor key (T_{actS}).

$$T_{ret} = T_{act} + Lzsk$$

Event 4: Before key N can be withdrawn from the zone, all RRsets that need to be signed must have been signed by the successor key (key N+1) and any old RRsets that do not include the new key or new RRSIGs

must have expired from caches. Note that the signatures are not replaced - each RRset is signed by both the old and new key.

This takes I_{ret} , the retire interval, given by the expression:

$$I_{ret} = D_{sgn} + D_{prp} + \max(TTL_{key}, TTL_{sig})$$

As before, D_{sgn} is the delay needed to ensure that all existing RRsets have been signed with the new key and D_{prp} is the propagation delay. The final term (the maximum of TTL_{key} and TTL_{sig}) is the period to wait for key and signature data associated with key N to expire from caches. (TTL_{key} is the TTL of the DNSKEY RRset and TTL_{sig} is the maximum TTL of all the RRSIG records in the zone created with the ZSK. The two may be different: although the TTL of an RRSIG is equal to the TTL of the RRs in the associated RRset [RFC4034], the DNSKEY RRset only needs to be signed with the KSK.)

At the end of this interval, key N is said to be dead. This occurs at the dead time (T_{dea}) so:

$$T_{dea} = T_{ret} + I_{ret}$$

Event 5: At some later time key N and the signatures generated with it can be removed from the zone. This is the removal time T_{rem} , given by:

$$T_{rem} \geq T_{dea}$$

3.2.3. Double-RRSIG Method

The timeline for a double-signature rollover is shown below. The diagram follows the convention described in Section 3.2.1

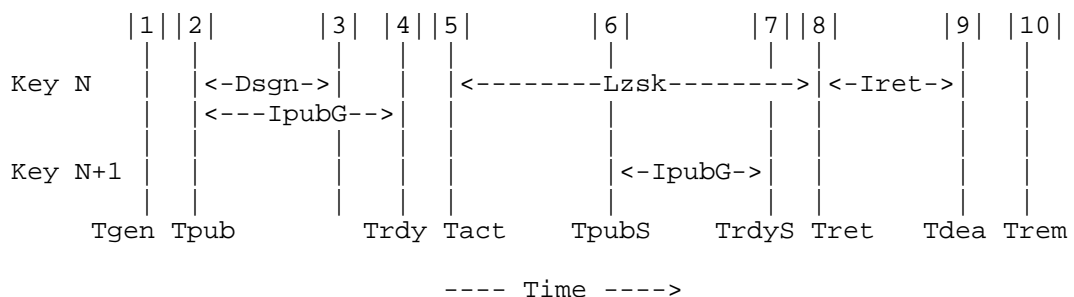


Figure 3: Timeline for a Double-Signature ZSK rollover.

Event 1: Key N is generated at the generate time (Tgen). Although there is no reason why the key cannot be generated immediately prior to its publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.

Event 2: Key N is used to sign the zone but existing signatures are retained. Although the new ZSK is not published in the zone at this point, for analogy with the other ZSK rollover methods and because this is the first time that key information is visible (albeit indirectly through the created signatures) this time is called the publication time (Tpub).

Event 3: After the signing interval, Dsgn, all RRsets that need to be signed have been signed by the new key. (As a result, all these RRsets are now signed twice, once by the (still-absent) key N and once by its predecessor.

Event 4: There is now a delay while the old signature information expires from caches. This interval is given by the expression:

$$Dprp + TTLsig$$

As before, Dprp is the propagation delay and TTLsig is the maximum TTL of all the RRSIG records in the zone created with the ZSK.

Again in analogy with other key rollover methods, this is defined as key N's ready time (Trdy) and the key is said to be in the ready state. (Although the key is not in the zone, it is ready to be used.) The interval between the publication and ready times is the publication interval of the signature, IpubG, i.e.,

$$Trdy = Tpub + IpubG$$

where

$$IpubG = Dsgn + Dprp + TTLsig$$

Event 5: At some later time the predecessor key is removed and the key N added to the DNSKEY RRset. As all the signed RRs have signatures created by the old and new keys, the records can still be authenticated. This time is the activation time (Tact) and the key is now said to be active.

Event 6: At some point thought must be given to rolling the key. The first step is to publish signatures created by the successor key (key

N+1) early enough for key N to be replaced after it has been active for its scheduled lifetime. This occurs at TpubS (the publication time of the successor), given by:

$$T_{pubS} \leq T_{act} + L_{zsk} - I_{pubG}$$

Event 7: The signatures have propagated and the new key could be added to the zone. This time is the ready time of the successor key (TrdyS).

$$T_{rdyS} = T_{pubS} + I_{pubG}$$

... where IpubG is as defined above.

Event 8: At some later time key N is removed from the zone's DNSKEY RRset and the successor key (key N+1) is added to it. This is the retire time of the key (Tret).

Event 9: The signatures must remain in the zone for long enough that the new DNSKEY RRset has had enough time to propagate to all caches. Once caches contain the new DNSKEY, the old signatures are no longer of use and can be considered to be dead as they cannot be validated by any key. In analogy with other rollover methods, the time at which this occurs is the dead time (Tdea), given by:

$$T_{dea} = T_{ret} + I_{ret}$$

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

$$I_{ret} = D_{prp} + TTL_{key}$$

Dprp is as defined earlier and TTLkey is the TTL of the DNSKEY RRset.

Event 10: At some later time the signatures can be removed from the zone. In analogy with other rollover methods, this time is called the remove time (Trem) and is given by:

$$T_{rem} \geq T_{dea}$$

3.3. Key-Signing Key Rollover Timelines

The following sections describe the rolling of a KSK. They show the events in the lifetime of a key (referred to as "key N") and cover its replacement by its successor (key N+1).

3.3.1. Double-Signature Method

The timeline for a double-signature rollover is shown below. The diagram follows the convention described in Section 3.2.1

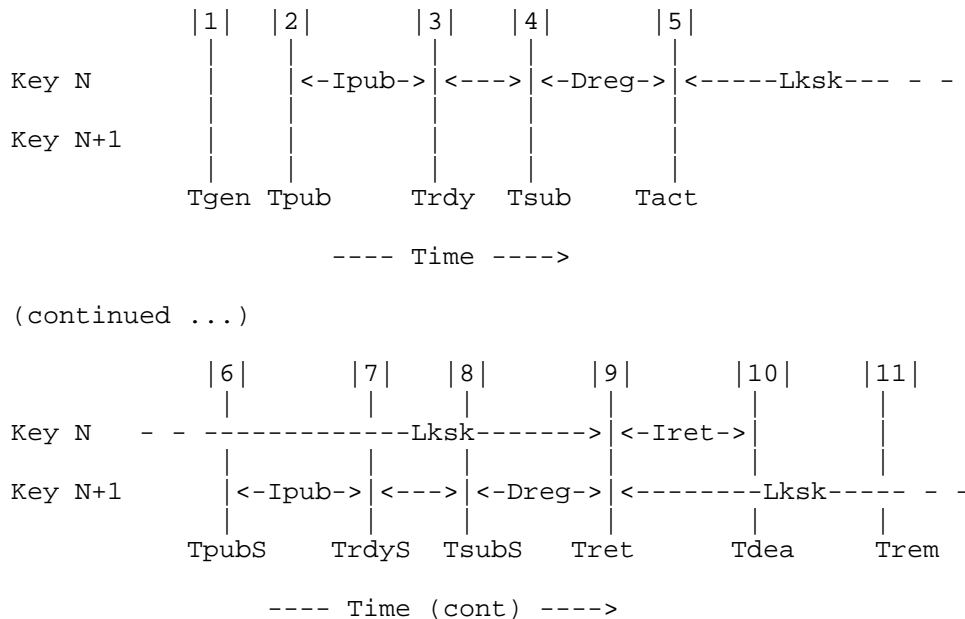


Figure 4: Timeline for a Double-Signature KSK rollover.

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 its publication in the zone (Event 2), some implementations may find it convenient to create a pool of keys in one operation and draw from that pool as required. For this reason, it is shown as a separate event. Keys that are available for use but not published are said to be generated.

Event 2: Key N is introduced into the zone; it is added to the DNSKEY RRset, which is then signed by key N and all currently active KSKs. (So at this point, the DNSKEY RRset is signed by both key N and its predecessor KSK. If other KSKs were active, it is signed by these as well.) This is the publication time (Tpub); after this the key is said to be published.

Event 3: Before it can be used, the key must be published for long enough to guarantee that any validating resolver that has a copy of

the DNSKEY RRset 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 (I_{pub}) and, for the second or subsequent KSKs in the zone, is given by:

$$I_{pub} = D_{prpC} + TTL_{key}$$

... where D_{prpC} is the propagation delay for the child zone (the zone containing the KSK being rolled) and TTL_{key} the TTL for the DNSKEY RRset. The time at which this occurs is the key's ready time, $Trdy$, given by:

$$Trdy = T_{pub} + I_{pub}$$

(The case of introducing the first KSK into the zone is discussed in Section 3.3.5.)

Event 4: At some later time, the DS record corresponding to the new KSK is submitted to the parent zone for publication. This time is the submission time, T_{sub} .

Event 5: The DS record is published in the parent zone. As this is the point at which all information for authentication - both DNSKEY and DS record - is available in the two zones, in analogy with other rollover methods, this is called the activation time of the key (T_{act}):

$$T_{act} = T_{sub} + D_{reg}$$

... where D_{reg} is the registration delay, the time taken after the DS record has been received by the parent zone manager for it to be placed in the zone. (Parent zones are often managed by different entities, and this term accounts for the organisational overhead of transferring a record.)

Event 6: While key N is active, thought needs to be given to its successor (key $N+1$). At some time before the scheduled end of the KSK lifetime, the successor KSK is published in the zone. (As before, this means that the DNSKEY RRset is signed by both the current and successor KSK.) This time is the publication time of the successor key, T_{pubS} , given by:

$$T_{pubS} \leq T_{act} + L_{sk} - D_{reg} - I_{pub}$$

... where L_{sk} is the scheduled lifetime of the KSK.

Event 7: After an interval I_{pub} , key $N+1$ becomes ready (in that all caches that have a copy of the DNSKEY RRset have a copy of this key). This time is the ready time of the successor ($TrdyS$).

Event 8: At the submission time of the successor (T_{subS}), the DS record corresponding to key $N+1$ is submitted to the parent zone.

Event 9: The successor DS record is published in the parent zone and the current DS record withdrawn. The current key is said to be retired and the time at which this occurs is T_{ret} , given by:

$$T_{ret} = T_{act} + L_{sk}$$

Event 10: Key N must remain in the zone until any caches that contain a copy of the DS RRset have a copy containing the new DS record. This interval is the retire interval, given by:

$$I_{ret} = D_{prpP} + TTL_{ds}$$

... where D_{prpP} is the propagation delay in the parent zone and TTL_{ds} the TTL of a DS record in the parent zone.

As the key is no longer used for anything, is said to be dead. This point is the dead time (T_{dea}), given by:

$$T_{dea} = T_{ret} + I_{ret}$$

Event 11: At some later time, key N is removed from the zone's DNSKEY RRset (at the remove time T_{rem}); the key is now said to be removed.

$$T_{rem} \geq T_{dea}$$

3.3.2. Double-DS Method

The timeline for a double-DS rollover is shown below. The diagram follows the convention described in Section 3.2.1

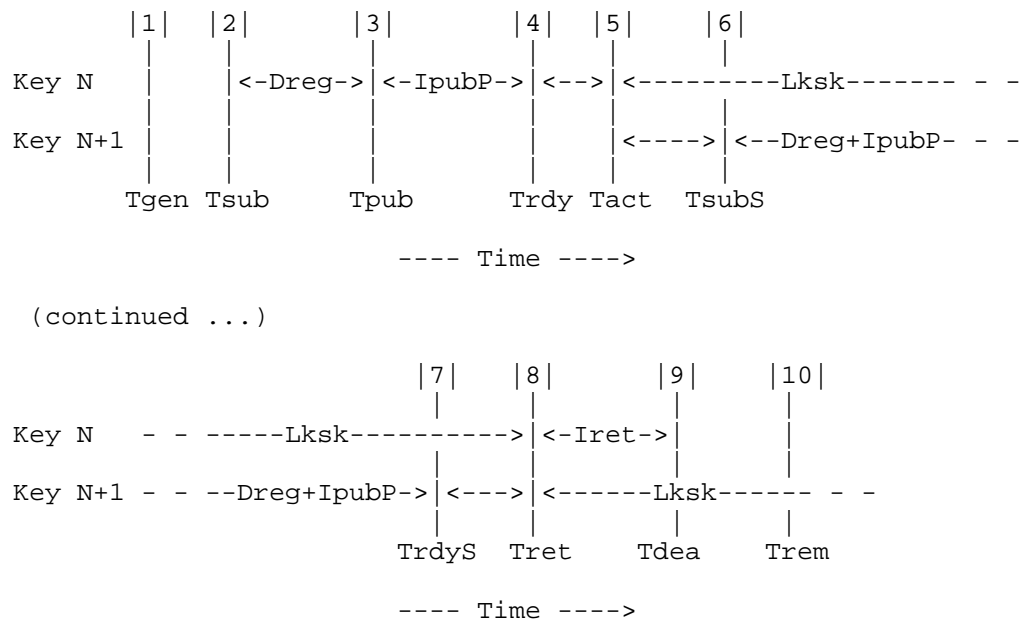


Figure 5: Timeline for a Double-DS KSK rollover.

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 its 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: The DS RR is submitted to the parent zone for publication. This time is the submission time, Tsub.

Event 3: After the registration delay, Dreg, the DS record is published in the parent zone. This is the publication time Tpub, given by:

$$T_{pub} = T_{sub} + D_{reg}$$

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

$$\text{Trdy} = \text{Tpub} + \text{IpubP}$$

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

$$\text{IpubP} = \text{DprpP} + \text{TTLds}$$

... where DprpP is the propagation delay for the parent zone and TTLds the TTL assigned to DS records in that zone.

Event 5: At some later time, the key rollover takes place and the new key (key N) is introduced into the DNSKEY RRset and used to sign that.

As both the old and new DS records have been in the parent zone long enough to ensure that they are in caches that contain the DS RRset, the zone can be authenticated throughout the rollover - the validating resolver either has a copy of the DNSKEY RRset authenticated by the predecessor key, or it has a copy of the updated RRset authenticated with the new key.

This time is key N's activation time (Tact) and at this point the key is said to be active.

Event 6: At some point thought must be given to key replacement. The DS record for the successor key must be submitted to the parent zone at a time such that when the current key is withdrawn, any cache that contains the zone's DS records has data about the DS record of the successor key. The time at which this occurs is the submission time of the successor, given by:

$$\text{TsubS} \leq \text{Tact} + \text{Lksk} - \text{IpubP} - \text{Dreg}$$

... where Lksk is the lifetime of key N according to policy.

Event 7: The successor key (key N+1) enters the ready state, i.e., its DS record is now in caches that contain the parent DS RRset. This is the ready time of the successor key, TrdyS.

(The interval between events 6 and 7 for the key N+1 correspond to the interval between events 2 and 4 for key N)

Event 8: When key N has been active for its lifetime (Lksk), it is replaced in the DNSKEY RRset by key N+1; the RRset is then signed with the new key. This is the retire time (Tret) of key N, given by:

$$T_{ret} = T_{act} + L_{sk}$$

Event 9: At some later time, all copies of the old DNSKEY RRset have expired from caches and the old DS record is no longer needed. In analogy with other rollover methods, this is called the dead time, T_{dea} , and is given by:

$$T_{dea} = T_{ret} + I_{ret}$$

... where I_{ret} is the retire interval, given by:

$$I_{ret} = D_{prpC} + TTL_{key}$$

As before, this term includes D_{prpC} , the time taken to propagate the RRset change through the master-slave hierarchy of the child zone and TTL_{key} , the time taken for the DNSKEY RRset to expire from caches.

Event 10: At some later time, the DS record is removed from the parent zone. In analogy with other rollover methods, this is the removal time (T_{rem}), given by:

$$T_{rem} \geq T_{dea}$$

3.3.3. Double-RRset Method

The timeline for a double-RRset rollover is shown below. The diagram follows the convention described in Section 3.2.1

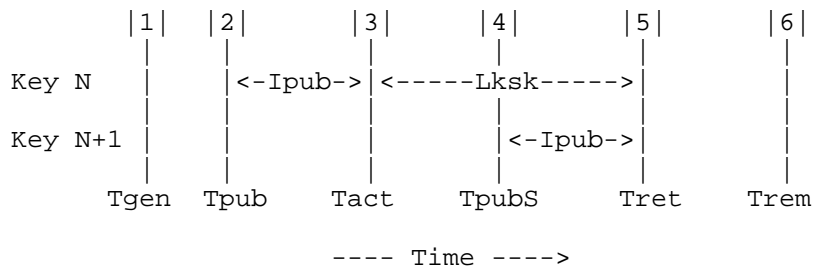


Figure 6: Timeline for a Double-RRset KSK rollover.

Event 1: Key N is generated at the generate time (T_{gen}). Although there is no reason why the key cannot be generated immediately prior to its 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: The key is added to and used for signing the DNSKEY RRset and is thereby published in the zone. At the same time the corresponding DS record is submitted to the parent zone for publication. This time is the publish time (T_{pub}) and the key is now said to be published.

Event 3: At some later time, the DS record is published in the parent zone and at some time after that, the updated information has reached all caches: any cache that holds a DNSKEY RRset from the child zone will have a copy that includes the new KSK, and any cache that has a copy of the parent DS RRset will have a copy that includes the new DS record.

The time at which this occurs is called the activation time of the new KSK (T_{act}), given by:

$$T_{act} = T_{pub} + I_{pub}$$

... where I_{pub} is the publication interval, given by:

$$I_{pub} = \max(I_{pubP}, I_{pubC}),$$

I_{pubP} being the publication interval in the parent zone and I_{pubC} the publication interval in the child zone. The parent zone's publication interval is given by:

$$I_{pubP} = D_{reg} + D_{prpP} + TTL_{ds}$$

where D_{reg} is the registration delay, the time taken for the DS record to be published in the parent zone. D_{prpP} is the parent zone's propagation delay and TTL_{ds} is the TTL of the DS record in that zone.

The child's publication interval is given by a similar equation:

$$I_{pubC} = D_{prpC} + TTL_{key}$$

... where D_{prpC} is the propagation delay in the child zone and TTL_{key} the TTL of a DNSKEY record.

Event 4: At some point we need to give thought to key replacement. The successor key (key $N+1$) must be introduced into the zone (and its DS record submitted to the parent) at a time such that it becomes active when the current key has been active for its lifetime, L_{ksk} . This time is T_{pubS} , the publication time of the successor key, and is given by:

$$T_{pubS} \leq Tact + L_{sk} - I_{pub}$$

... where L_{sk} is the lifetime of the KSK.

Event 5: Key N+1's DNSKEY and DS records are in any caches that contain the child zone DNSKEY and/or the parent zone DS RR, and so the zone can be validated with the new key. This is the activation time of the successor key ($TactS$) and by analogy with other rollover methods, it is also the retire time of the current key. Since at this time the zone can be validated by the successor key, there is no reason to keep the current key in the zone and the time can also be regarded as the current key's dead time. Thus:

$$T_{ret} = T_{dea} = TactS = Tact + L_{sk}$$

Event 6: At some later time, the key N's DS and DNSKEY records are removed from their respective zones. In analogy with other rollover methods, this is the removal time (T_{rem}), given by:

$$T_{rem} \geq T_{dea}$$

3.3.4. Interaction with Configured Trust Anchors

Although the preceding sections have been concerned with rolling KSKs where the trust anchor is a DS record in the parent zone, zone managers may want to take account of the possibility that some validating resolvers may have configured trust anchors directly.

Rolling a configured trust anchor is dealt with in [RFC5011]. It requires introducing the KSK to be used as the trust anchor into the zone for a period of time before use, and retaining it (with the "revoke" bit set) for some time after use.

3.3.4.1. Addition of KSK

When the new key is introduced, the publication interval (I_{pub}) in the Double-Signature and Double-RRset methods should also be subject to the condition:

$$I_{pub} \geq D_{prp} + \max(I_{trp}, TTL_{key})$$

... where the right hand side of the expression is the time taken for the change to propagate to all nameservers for the zone plus the "trust point" interval. This latter term is the interval required to guarantee that a resolver configured for the automatic update of keys from a particular trust point will see at least two validated DNSKEY RRsets containing the new key (a requirement from [RFC5011], section 2.4.1). It is defined by the expression:

$$\text{Itrp} \geq (2 * \text{queryInterval}) + (n * \text{retryTime})$$

... where queryInterval and retryTime are as defined in section 2.3 of [RFC5011]. "n" is the total number of retries needed by the resolver during the two attempts to get the DNSKEY RRset.

The first term of the expression $(2 * \text{queryInterval})$ represents the time to obtain two validated DNSKEY RRsets. The second term $(n * \text{retryTime})$ is a safety margin, with the value of "n" reflecting the degree of confidence in the communication between a resolver and the trust point.

In the Double-DS method, instead of swapping the KSK RRs in a single step, there must now be a period of overlap. In other words, the new KSK must be introduced into the zone at least:

$$\text{DprpC} + \max(\text{Itrp}, \text{TTLkey})$$

... before the switch is made.

3.3.4.2. Removal of KSK

The timeline for the removal of the key in all methods is modified by introducing a new state, "revoked". When the key reaches its dead time, instead of being declared "dead", it is revoked; the "revoke" bit is set in the published DNSKEY RR, and the DNSKEY RRset re-signed with the current and revoked keys. The key is maintained in this state for the "revoke" interval, Irev, given by:

$$\text{Irev} \geq 30 \text{ days}$$

... 30 days being the [RFC5011] remove hold-down time. After this time, the key is dead and can be removed from the zone.

3.3.5. Introduction of First Keys

There are no timing considerations associated with the introduction of the first keys into a zone other than they must be introduced and the zone validly signed before a chain of trust to the zone is created.

This is important: in the case of a secure parent, it means ensuring that the DS record is not published in the parent zone until there is no possibility that a validating resolver can obtain the record yet is not 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 not possible to guarantee this. It is up to the operator of the validating resolver to wait for the new KSK to appear at all

servers for the zone before configuring the trust anchor.

4. Standby Keys

Although keys will usually be rolled according to some regular schedule, there may be occasions when an emergency rollover is required, e.g., if the active key is suspected of being compromised. The aim of the emergency rollover is to allow the zone to be re-signed with a new key as soon as possible. As a key must be in the ready state to sign the zone, having at least one additional key (a standby key) in this state at all times will minimise delay.

In the case of a ZSK, a standby key only really makes sense with the Pre-Publication method. A permanent standby DNSKEY RR should be included in the zone or successor keys could be introduced as soon as possible after a key becomes active. Either way results in one or more additional ZSKs in the DNSKEY RRset that can immediately be used to sign the zone if the current key is compromised.

(Although in theory the mechanism could be used with both the Double-Signature and Double-RRSIG methods, it would require pre-publication of the signatures. Essentially, the standby key would be permanently active, as it would have to be periodically used to renew signatures. Zones would also permanently require two sets of signatures.)

It is also possible to have a standby KSK. The Double-Signature method requires that the standby KSK be included in the DNSKEY RRset; rolling the key then requires just the introduction of the DS record in the parent. Note that the standby KSK should also be used to sign the DNSKEY RRset. As the RRset and its signatures travel together, merely adding the KSK without using it to sign the DNSKEY RRset does not provide the desired time saving: for a KSK to be used in a rollover the DNSKEY RRset must be signed with it, and this would introduce a delay while the old RRset (not signed with the new key) expires from caches.

The idea of a standby KSK in the Double-RRset rollover method effectively means having two active keys (as the standby KSK and associated DS record would both be published at the same time in their respective zones).

Finally, in the Double-DS method of rolling a KSK, it is not a standby key that is present, it is a standby DS record in the parent zone.

Whatever algorithm is used, the standby item of data can be included in the zone on a permanent basis, or be a successor introduced as

early as possible.

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 changed - there is no relationship between the keys of different algorithms. This means that they can be rolled independently of one another. In other words, the key rollover logic described above should be run separately for each algorithm; the union of the results is included in the zone, which is signed using the active key for each algorithm.

6. Limitation of Scope

This document represents current thinking at the time of publication. However, the subject matter is evolving and it is more than likely that this document will need to be revised in the future.

Some of the techniques and ideas that DNSSEC operators are considering differ from those described in this document. Of particular interest are alternatives to the strict split into KSK and ZSK key roles and the consequences for rollover logic from partial signing (i.e., when the new key initially only signs a fraction of the zone while leaving other signatures generated by the old key in place).

Furthermore, as noted in section 5, this document covers only rolling keys of the same algorithm: it does not cover transitions between algorithms. The timing issues associated with algorithm rollovers will require a separate document.

The reader is therefore reminded that DNSSEC is, as of date of publication, in the early stages of deployment, and best practices may further develop over time.

7. 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 hold-down" and "remove hold-down" times. It should be emphasized that this delay is a policy choice and not a function of timing values and that it also requires changes to the rollover process due to the need to manage revocation of trust anchors.

Finally, the treatment of emergency key rollover is significantly simplified by the introduction of standby keys as standard practice during all types of rollovers.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

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

10. Acknowledgements

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

11. Normative References

- [I-D.ietf-dnsop-rfc4641bis]
Kolkman, O. and M. Mekking, "DNSSEC Operational Practices, Version 2", draft-ietf-dnsop-rfc4641bis-11 (work in progress), April 2012.
- [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, 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

I interval between two events

L lifetime: interval set by the zone manager

T a point in time

TTL TTL of a record

I and T and TTL are self-explanatory. Like I, both D and L are 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: a "D" interval (delay) is a feature of the process, probably outside control of the zone manager, whereas an "L" interval (lifetime) is chosen by the zone manager and is a feature of policy.

`<id>` is lower-case and defines what object or event the variable is related to, e.g.,

act	activation
pub	publication
ret	retire

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

C	child
G	signature
P	parent
S	successor

The list of variables used in the text is:

Dprp	Propagation delay. The amount of time for a change made at a master nameserver to propagate to all the slave nameservers.
DprpC	Propagation delay in the child zone.
DprpP	Propagation delay in the parent zone.
Dreg	Registration delay: the time taken for a DS record submitted to a parent zone to appear in it. As a parent zone is often managed by a different organisation to that managing the child zone, the delays associated with passing data between zones is captured by this term.
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.
Ipub	Publication interval. The amount of time that must elapse after the publication of a key before it can be assumed that any resolvers that have the DNSKEY RRset cached have a copy of this key.
IpubC	Publication interval in the child zone.

IpubG	Publication interval for the signature created by a ZSK: the amount of time that must elapse after the signature has been created before it can be assumed that any resolves that have the RRset and RRSIG cached have a copy of this signature.
IpubP	Publication interval in the parent zone.
Iret	Retire interval. The amount of time that must elapse after a key enters the retire state for any signatures created with it to be purged from validating resolver caches.
Irev	Revoke interval. The amount of time that a KSK must remain published with the revoke bit set to satisfy [RFC5011] considerations.
Itrp	Trust-point interval. The amount of time that a trust anchor must be published for to guarantee that a resolver configured for an automatic update of keys will see the new key at least twice.
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.
Tact	Activation time of the key; the time at which the key is regarded as the principal key for the zone.
TactS	Activation time of the successor key.
Tdea	Dead time of a key. Applicable only to ZSKs, this is the time at which any record signatures held in validating resolver caches are guaranteed to be created with the successor key.
Tgen	Generate time of a key. The time that a key is created.
Tpub	Publication time of a key. The time that a key appears in a zone for the first time.

TpubS	Publication time of the successor key.
Trem	Removal time of a key. The time at which a key is removed from the zone.
Tret	Retire time of a key. The time at which a successor key starts being used to sign the zone.
Trdy	Ready time of a key. The time at which it can be guaranteed that validating resolvers that have key information from this zone cached have a copy of this key in their cache. (In the case of KSKs, should the validating resolvers also have DS information from the parent zone cached, the cache must include information about the DS record corresponding to the key.)
TrdyS	Ready time of a successor key.
Tsub	Submission time - the time at which the DS record of a KSK is submitted to the parent.
TsubS	Submission time of the successor key.
TTLds	Time to live of a DS record (in the parent zone).
TTLkey	Time to live of a DNSKEY record. (By implication, this is also the time to live of the signatures on the DNSKEY RRset.)
TTLsig	The maximum time to live of all the RRSIG records in the zone that were created with the ZSK.

Appendix B. Change History (To be removed on publication)

- o draft-ietf-dnsop-dnssec-key-timing-03
 - * Clarifications of and corrections to wording (Marc Lampo, Alfred Hoenes).
 - * Updated timings related to trust anchor interaction (Matthijs Mekking).
 - * Updated RFC 4641 reference to 4641bis (Alfred Hoenes).
 - * Moved change history to end of document (Alfred Hoenes).
- o draft-ietf-dnsop-dnssec-key-timing-02
 - * Significant re-wording of some sections.
 - * Removal of events noting change of state of predecessor key from ZSK Double-RRSIG and Double-Signature methods.
 - * Change order of bullet points (and some wording) in section 1.1.

- * Remove discussion of advantages and disadvantages of key roll methods from section 2: draft is informative and does not give recommendations.
- * Removal of discussion of upper limit to retire time relationship to signature lifetime.
- * Remove timing details of first key in the zone and move discussion of first signing of a zone to later in the document). (Matthijs Mekking)
- * Removal of redundant symbols from Appendix A.
- o draft-ietf-dnsop-dnssec-key-timing-01
 - * Added section on limitation of scope.
- o draft-ietf-dnsop-dnssec-key-timing-00
 - * Change to author contact details.
- o draft-morris-dnsop-dnssec-key-timing-02
 - * General restructuring.
 - * Added descriptions of more rollovers (IETF-76 meeting).
 - * Improved description of key states and removed diagram.
 - * Provided simpler description of standby keys.
 - * Added section concerning first key in a zone.
 - * Moved [RFC5011] to a separate section.
 - * Various nits fixed (Alfred Hoenes, Jeremy Reed, Scott Rose, Sion Lloyd, Tony Finch).
- o draft-morris-dnsop-dnssec-key-timing-01
 - * Use latest boilerplate for IPR text.
 - * List different ways to roll a KSK (acknowledgements to Mark Andrews).
 - * Restructure to concentrate on key timing, not management procedures.
 - * Change symbol notation (Diane Davidowicz and others).
 - * Added key state transition diagram (Diane Davidowicz).
 - * Corrected spelling, formatting, grammatical and style errors (Diane Davidowicz, Alfred Hoenes and Jinmei Tatuya).
 - * Added note that in the case of multiple algorithms, the signatures and rollovers for each algorithm can be considered as more or less independent (Alfred Hoenes).
 - * Take account of the fact that signing a zone is not atomic (Chris Thompson).
 - * Add section contrasting pre-publication rollover with double signature rollover (Matthijs Mekking).
 - * Retained distinction between first and subsequent keys in definition of initial publication interval (Matthijs Mekking).
- o draft-morris-dnsop-dnssec-key-timing-00
 - Initial draft.

Authors' Addresses

Stephen Morris
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
USA

Phone: +1 650 423 1300
Email: stephen@isc.org

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

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

John Dickinson
Sinodun Internet Technologies Ltd
Stables 4 Suite 11, Howbery Park
Wallingford, Oxfordshire OX10 8BA
UK

Phone: +44 1491 818120
Email: jad@sinodun.com

DNSOP
Internet-Draft
Obsoletes: 4641 (if approved)
Intended status: Informational
Expires: March 15, 2013

O. Kolkman
W. Mekking
NLnet Labs
R. Gieben
SIDN Labs
September 11, 2012

DNSSEC Operational Practices, Version 2
draft-ietf-dnsop-rfc4641bis-13

Abstract

This document describes a set of practices for operating the DNS with security extensions (DNSSEC). The target audience is zone administrators deploying DNSSEC.

The document discusses operational aspects of using keys and signatures in the DNS. It discusses issues of key generation, key storage, signature generation, key rollover, and related policies.

This document obsoletes RFC 4641 as it covers more operational ground and gives more up-to-date requirements with respect to key sizes and the DNSSEC operations.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on March 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	6
1.1.	The Use of the Term 'key'	7
1.2.	Time Definitions	7
2.	Keeping the Chain of Trust Intact	9
3.	Key Generation and Storage	10
3.1.	Operational Motivation for Zone Signing Keys and Key Signing Keys	10
3.2.	Practical Consequences of KSK and ZSK Separation	12
3.2.1.	Rolling a KSK that is not a trust anchor	12
3.2.2.	Rolling a KSK that is a trust anchor	13
3.2.3.	The use of the SEP flag	14
3.3.	Key Effectivity Period	15
3.4.	Cryptographic Considerations	16
3.4.1.	Signature Algorithm	16
3.4.2.	Key Sizes	17
3.4.3.	Private Key Storage	18
3.4.4.	Key Generation	19
3.4.5.	Differentiation for 'High-Level' Zones?	19
4.	Signature Generation, Key Rollover, and Related Policies	21
4.1.	Key Rollovers	21
4.1.1.	Zone Signing Key Rollovers	21
4.1.1.1.	Pre-Publish Zone Signing Key Rollover	21
4.1.1.2.	Double Signature Zone Signing Key Rollover	24
4.1.1.3.	Pros and Cons of the Schemes	25
4.1.2.	Key Signing Key Rollovers	25
4.1.2.1.	Special Considerations for RFC 5011 KSK rollover	27
4.1.3.	Rollover for a Single Type Signing Scheme Key Rollover	28
4.1.4.	Algorithm rollovers	30
4.1.4.1.	Single Type Signing Scheme Algorithm Rollover	34
4.1.4.2.	Algorithm rollover, RFC 5011 style	34
4.1.4.3.	Single Signing Type Algorithm Rollover, RFC 5011 style	36
4.1.4.4.	NSEC to NSEC3 algorithm rollover	36
4.1.5.	Considerations for Automated Key Rollovers	37
4.2.	Planning for Emergency Key Rollover	37
4.2.1.	KSK Compromise	38
4.2.1.1.	Emergency Key Rollover Keeping the Chain of Trust Intact	38
4.2.1.2.	Emergency Key Rollover Breaking the Chain of Trust	39
4.2.2.	ZSK Compromise	40
4.2.3.	Compromises of Keys Anchored in Resolvers	40
4.2.4.	Stand-by Keys	40
4.3.	Parent Policies	41

4.3.1.	Initial Key Exchanges and Parental Policies Considerations	41
4.3.2.	Storing Keys or Hashes?	42
4.3.3.	Security Lameness	42
4.3.4.	DS Signature Validity Period	43
4.3.5.	Changing DNS Operators	44
4.3.5.1.	Cooperating DNS operators	44
4.3.5.2.	Non Cooperating DNS Operators	46
4.4.	Time in DNSSEC	48
4.4.1.	Time Considerations	48
4.4.2.	Signature Validity Periods	50
4.4.2.1.	Maximum Value	50
4.4.2.2.	Minimum Value	51
4.4.2.3.	Differentiation between RRsets	52
5.	Next Record type	54
5.1.	Differences between NSEC and NSEC3	54
5.2.	NSEC or NSEC3	55
5.3.	NSEC3 parameters	55
5.3.1.	NSEC3 Algorithm	55
5.3.2.	NSEC3 Iterations	56
5.3.3.	NSEC3 Salt	56
5.3.4.	Opt-Out	57
6.	Security Considerations	58
7.	IANA considerations	59
8.	Contributors and Acknowledgments	60
9.	References	61
9.1.	Normative References	61
9.2.	Informative References	61
Appendix A.	Terminology	64
Appendix B.	Typographic Conventions	66
Appendix C.	Transition Figures for Special Case Algorithm Rollovers	69
Appendix D.	Transition Figure for Changing DNS Operators	75
Appendix E.	Summary of Changes from RFC 4641	77
Appendix F.	Document Editing History	78
F.1.	draft-ietf-dnsop-rfc4641-00	78
F.2.	version 0->1	78
F.3.	version 1->2	79
F.4.	version 2->3	79
F.5.	version 3->4	80
F.6.	version 4->5	80
F.7.	version 5->6	80
F.8.	version 6->7	81
F.9.	version 7->8	81
F.10.	version 8->9	81
F.11.	version 9->10	82
F.12.	version 10->11	82
F.13.	version 11->12	82

F.14. version 12->13	82
F.15. Subversion information	82
Authors' Addresses	83

1. Introduction

This document describes how to run a DNS Security (DNSSEC)-enabled environment. It is intended for operators who have knowledge of the DNS (see RFC 1034 [RFC1034] and RFC 1035 [RFC1035]) and want to deploy DNSSEC (RFC 4033 [RFC4033], RFC 4034 [RFC4034], RFC 4035 [RFC4035] and RFC 5155 [RFC5155]). The focus of the document is on serving authoritative DNS information and is aimed at zone owners, name server operators, registries, registrars and registrants. It assumes that there is no direct relation between those entities and the operators of validating recursive name servers (validators).

During workshops and early operational deployment, operators and system administrators have gained experience about operating the DNS with security extensions (DNSSEC). This document translates these experiences into a set of practices for zone administrators. Although the DNS Root has been signed since July 15, 2010 and now more than 80 secure delegations are provisioned in the root, at the time of writing there still exists relatively little experience with DNSSEC in production environments below the top-level domain (TLD) level; this document should therefore explicitly not be seen as representing 'Best Current Practices'. Instead, it describes the decisions that should be made when deploying DNSSEC, gives the choices available for each one, and provides some operational guidelines. The document does not give strong recommendations. That may be subject for a future version of this document.

The procedures herein are focused on the maintenance of signed zones (i.e., signing and publishing zones on authoritative servers). It is intended that maintenance of zones such as re-signing or key rollovers be transparent to any verifying clients.

The structure of this document is as follows. In Section 2, we discuss the importance of keeping the "chain of trust" intact. Aspects of key generation and storage of keys are discussed in Section 3; the focus in this section is mainly on the security of the private part of the key(s). Section 4 describes considerations concerning the public part of the keys. Section 4.1 and Section 4.2 deal with the rollover, or replacement, of keys. Section 4.3 discusses considerations on how parents deal with their children's public keys in order to maintain chains of trust. Section 4.4 covers all kinds of timing issues around key publication. Section 5 covers the considerations regarding selecting and using NSEC or NSEC3 [RFC5155].

The typographic conventions used in this document are explained in Appendix B.

Since we describe operational suggestions and there are no protocol specifications, the RFC 2119 [RFC2119] language does not apply to this document, though we do use quotes from other documents that do include the RFC 2119 language.

This document obsoletes RFC 4641 [RFC4641].

1.1. The Use of the Term 'key'

It is assumed that the reader is familiar with the concept of asymmetric keys on which DNSSEC is based (public key cryptography RFC 4949 [RFC4949]). Therefore, this document will use the term 'key' rather loosely. Where it is written that 'a key is used to sign data' it is assumed that the reader understands that it is the private part of the key pair that is used for signing. It is also assumed that the reader understands that the public part of the key pair is published in the DNSKEY Resource Record and that it is the public part that is used in signature verification.

1.2. Time Definitions

In this document, we will be using a number of time-related terms. The following definitions apply:

Signature validity period: The period that a signature is valid. It starts at the (absolute) time specified in the signature inception field of the RRSIG RR and ends at the (absolute) time specified in the expiration field of the RRSIG RR. The document sometimes also uses the term "validity period", which means the same.

Signature publication period: The period that a signature is published. It starts at the time the signature is introduced in the zone for the first time and ends at the time when the signature is removed or replaced with a new signature. After one stops publishing an RRSIG in a zone, it may take a while before the RRSIG has expired from caches and has actually been removed from the DNS.

Key effectivity period: The period during which a key pair is expected to be effective. It is defined as the time between the earliest inception time stamp and the last expiration date of any signature made with this key, regardless of any discontinuity in the use of the key. The key effectivity period can span multiple signature validity periods.

Maximum/Minimum Zone Time to Live (TTL): The maximum or minimum value of the TTLs from the complete set of RRs in a zone, that are used by validators or resolvers. Note that the minimum TTL is not the same as the MINIMUM field in the SOA RR. See RFC 2308 [RFC2308] for more information.

2. Keeping the Chain of Trust Intact

Maintaining a valid chain of trust is important because broken chains of trust will result in data being marked as Bogus (as defined in RFC 4033 [RFC4033] Section 5), which may cause entire (sub)domains to become invisible to verifying clients. The administrators of secured zones need to realize that, to verifying clients, their zone is part of a chain of trust.

As mentioned in the introduction, the procedures herein are intended to ensure that maintenance of zones, such as re-signing or key rollovers, will be transparent to the verifying clients on the Internet.

Administrators of secured zones will need to keep in mind that data published on an authoritative primary server will not be immediately seen by verifying clients; it may take some time for the data to be transferred to other (secondary) authoritative name servers and clients may be fetching data from caching non-authoritative servers. In this light, note that the time until the data is available on the slave can be negligible when using NOTIFY [RFC1996] and incremental transfer (IXFR) [RFC1995]. It increases when full zone transfers (AXFR) are used in combination with NOTIFY. It increases even more if you rely on the full zone transfers being based only on the SOA timing parameters for refresh.

For the verifying clients, it is important that data from secured zones can be used to build chains of trust regardless of whether the data came directly from an authoritative server, a caching name server, or some middle box. Only by carefully using the available timing parameters can a zone administrator ensure that the data necessary for verification can be obtained.

The responsibility for maintaining the chain of trust is shared by administrators of secured zones in the chain of trust. This is most obvious in the case of a 'key compromise' when a trade-off must be made between maintaining a valid chain of trust and replacing the compromised keys as soon as possible. Then zone administrators will have to decide between keeping the chain of trust intact - thereby allowing for attacks with the compromised key - or deliberately breaking the chain of trust and making secured subdomains invisible to security-aware resolvers (also see Section 4.2).

3. Key Generation and Storage

This section describes a number of considerations with respect to the use of keys. For the design of an operational procedure for key generation and storage, a number of decisions need to be made:

- o Does one differentiate between Zone Signing Keys and Key Signing Keys or is the use of one type of key sufficient?
- o Are Key Signing Keys (likely to be) in use as trust anchors [RFC4033]?
- o What are the timing parameters that are allowed by the operational requirements?
- o What are the cryptographic parameters that fit the operational need?

The following section discusses the considerations that need to be taken into account when making those choices.

3.1. Operational Motivation for Zone Signing Keys and Key Signing Keys

The DNSSEC validation protocol does not distinguish between different types of DNSKEYs. The motivations to differentiate between keys are purely operational; validators will not make a distinction.

For operational reasons, described below, it is possible to designate one or more keys to have the role of Key Signing Keys (KSKs). These keys will only sign the apex DNSKEY RRset in a zone. Other keys can be used to sign all the other RRsets in a zone that require signatures. They are referred to as Zone Signing Keys (ZSKs). In cases where the differentiation between KSK and ZSK is not made, keys have both the role of KSK and ZSK, we talk about a Single Type Signing Scheme.

If the two functions are separated, then for almost any method of key management and zone signing, the KSK is used less frequently than the ZSK. Once a DNSKEY RRset is signed with the KSK, all the keys in the RRset can be used as ZSKs. If there has been an event that increases the risk that a ZSK is compromised, it can be simply replaced with a ZSK rollover. The new RRset is then re-signed with the KSK.

Changing a key that is a secure entry point (SEP) [RFC4034] for a zone can be relatively expensive as it involves interaction with third parties: when a key is only pointed to by a delegation signer (DS) [RFC4034] record in the parent zone, one needs to complete the interaction with the parent and wait for the updated DS record to

appear in the DNS. In the case where a key is configured as a trust anchor, one has to wait until one has sufficient confidence that all trust anchors have been replaced. In fact, it may be that one is not able to reach the complete user-base with information about the key rollover.

Given the assumption that for KSKs the SEP flag is set, the KSK can be distinguished from a ZSK by examining the flag field in the DNSKEY RR: If the flag field is an odd number it is a KSK; otherwise it is a ZSK.

There is also a risk that keys can be compromised through theft or loss. For keys that are installed on file-systems of name servers that are connected to the network (e.g. for dynamic updates), that risk is relatively high. Where keys are stored on Hardware Security Modules (HSMs) or stored off-line, such risk is relatively low. However, storing keys off-line or with more limitations on access control has a negative effect on the operational flexibility. By separating the KSK and ZSK functionality, these risks can be managed while making the tradeoff against the involved costs. For example, a KSK can be stored off-line or with more limitations on access control than ZSKs which need to be readily available for operational purposes such as the addition or deletion of zone data. A KSK stored on a smartcard, that is kept in a safe, combined with a ZSK stored on a file-system accessible by operators for daily routine may provide a better protection against key compromise without losing much operational flexibility. It must be said that some HSMs give the option to have your keys online, giving more protection and hardly affecting the operational flexibility. In those cases, a KSK-ZSK split is not more beneficial than the Single Type Signing Scheme.

It is worth mentioning that there's not much point obsessively protecting the key if you don't protect the zone files, which also live on the file-systems.

Finally there is a risk of cryptanalysis of the key material. The costs of such analysis are correlated to the length of the key. However, cryptanalysis arguments provide no strong motivation for a KSK/ZSK split. Suppose one differentiates between a KSK and a ZSK whereby the KSK effectivity period is X times the ZSK effectivity period. Then, in order for the resistance to cryptanalysis to be the same for the KSK and the ZSK, the KSK needs to be X times stronger than the ZSK. Since for all practical purposes, X will be somewhere of the order of 10 to 100, the associated key sizes will vary only about a byte in size for symmetric keys. When translated to asymmetric keys, the size difference is still too insignificant to warrant a key-split; it only marginally affects the packet size and signing speed.

The arguments for differentiation between the ZSK and KSK are weakest when:

- o the exposure to risk is low (e.g. when keys are stored on HSMs);
- o one can be certain that a key is not used as a trust anchor;
- o maintenance of the various keys cannot be performed through tools (is prone to human error); and
- o the interaction through the child-parent provisioning chain -- in particular the timely appearance of a new DS record in the parent zone in emergency situations -- is predictable.

If the above holds then the costs of the operational complexity of a KSK-ZSK split may outweigh the costs of operational flexibility and choosing a Single Type Signing Scheme is a reasonable option. In other cases we advise that the separation between KSKs and ZSKs is made.

3.2. Practical Consequences of KSK and ZSK Separation

A key that acts only as a Zone Signing Key is used to sign all the data except the DNSKEY RRset in a zone on a regular basis. When a ZSK is to be rolled, no interaction with the parent is needed. This allows for a relatively short key effectivity period.

A key with only the Key Signing Key role is to be used to sign the DNSKEY RRs in a zone. If a KSK is to be rolled, there may be interactions with other parties. These can include the administrators of the parent zone or administrators of verifying resolvers that have the particular key configured as secure entry points. In the latter case, everyone relying on the trust anchor needs to roll over to the new key, a process that may be subject to stability costs if automated trust anchor rollover mechanisms (e.g. RFC 5011 [RFC5011]) are not in place. Hence, the key effectivity period of these keys can and should be made much longer.

3.2.1. Rolling a KSK that is not a trust anchor

There are three schools of thought on rolling a KSK that is not a trust anchor:

1. It should be done frequently and regularly (possibly every few months) so that a key rollover remains an operational routine.
2. It should be done frequently but irregularly. Frequently meaning every few months, again based on the argument that a rollover is

a practiced and common operational routine, and irregular meaning with a large jitter, so that third parties do not start to rely on the key and will not be tempted to configure it as a trust anchor.

3. It should only be done when it is known or strongly suspected that the key can be or has been compromised, or in conjunction with operator change policies and procedures, like when a new algorithm or key storage is required.

There is no widespread agreement on which of these three schools of thought is better for different deployments of DNSSEC. There is a stability cost every time a non-anchor KSK is rolled over, but it is possibly low if the communication between the child and the parent is good. On the other hand, the only completely effective way to tell if the communication is good is to test it periodically. Thus, rolling a KSK with a parent is only done for two reasons: to test and verify the rolling system to prepare for an emergency, and in the case of (preventing) an actual emergency.

Finally, in most cases a zone administrator cannot be fully certain that the zone's KSK is not in use as a trust anchor somewhere. While the configuration of trust anchors is not the responsibility of the zone administrator, there may be stability costs for the validator administrator that (wrongfully) configured the trust anchor when the zone administrator rolls a KSK.

3.2.2. Rolling a KSK that is a trust anchor

The same operational concerns apply to the rollover of KSKs that are used as trust anchors: if a trust anchor replacement is done incorrectly, the entire domain that the trust anchor covers will become Bogus until the trust anchor is corrected.

In a large number of cases it will be safe to work from the assumption that one's keys are not in use as trust anchors. If a zone administrator publishes a "DNSSEC Signing Policy and/or a DNSSEC Practice Statement" [I-D.ietf-dnsop-dnssec-dps-framework] that should be explicit about the fact whether the existence of trust anchors will be taken into account. There may be cases where local policies enforce the configuration of trust anchors on zones which are mission critical (e.g. in enterprises where the trust anchor for the enterprise domain is configured in the enterprise's validator). It is expected that the zone administrators are aware of such circumstances.

One can argue that because of the difficulty of getting all users of a trust anchor to replace an old trust anchor with a new one, a KSK

that is a trust anchor should never be rolled unless it is known or strongly suspected that the key has been compromised. In other words the costs of a KSK rollover are prohibitively high because some users cannot be reached.

However, the "operational habit" argument also applies to trust anchor reconfiguration at the clients' validators. If a short key effectivity period is used and the trust anchor configuration has to be revisited on a regular basis, the odds that the configuration tends to be forgotten is smaller. In fact, the costs for those users can be minimized by automating the rollover with RFC 5011 [RFC5011] and by rolling the key regularly (and advertising such) so that the operators of validating resolvers will put the appropriate mechanism in place to deal with these stability costs: in other words, budget for these costs instead of incurring them unexpectedly.

It is therefore preferred to roll KSKs that are expected to be used as trust anchors on a regular basis if and only if those rollovers can be tracked using standardized (e.g. RFC 5011 [RFC5011]) mechanisms.

3.2.3. The use of the SEP flag

The so-called Secure Entry Point (SEP) [RFC4035] flag can be used to distinguish between keys that are intended to be used as the secure entry point into the zone when building chains of trust, i.e., they are (to be) pointed to by parental DS RRs or configured as a trust anchor.

While the SEP flag does not play any role in validation, it is used in practice for operational purposes such as for the rollover mechanism described in RFC 5011 [RFC5011]. The common convention is to set the SEP flag on any key that is used for key exchanges with the parent and/or potentially used for configuration as a trust anchor. Therefore, it is suggested that the SEP flag is set on keys that are used as KSKs and not on keys that are used as ZSKs, while in those cases where a distinction between KSK and ZSK is not made (i.e. for a Single Type Signing Scheme), it is suggested that the SEP flag is set on all keys.

Note: some signing tools may assume a KSK/ZSK split and use the (non) presence of the SEP flag to determine which key is to be used for signing zone data; these tools may get confused when a Single Type Signing Scheme is used.

3.3. Key Effectivity Period

In general the available key length sets an upper limit on the key effectivity period. For all practical purposes it is sufficient to define the key effectivity period based on purely operational requirements and match the key length to that value. Ignoring the operational perspective, a reasonable effectivity period for KSKs that have corresponding DS records in the parent zone is of the order of two decades or longer. That is, if one does not plan to test the rollover procedure, the key should be effective essentially forever, and only rolled over in case of emergency.

When one opts for a regular key-rollover, a reasonable key effectivity period for KSKs that have a parent zone is one year, meaning you have the intent to replace them after 12 months. The key effectivity period is merely a policy parameter, and should not be considered a constant value. For example, the real key effectivity period may be a little bit longer than 12 months, because not all actions needed to complete the rollover could be finished in time.

As argued above, this annual rollover gives operational practice of rollovers for both the zone and validator administrators. Besides, in most environments a year is a time-span that is easily planned and communicated.

Where keys are stored online and the exposure to various threats of compromise is fairly high, an intended key effectivity period of a month is reasonable for Zone Signing Keys.

Although very short key effectivity periods are theoretically possible, when replacing keys one has to take into account the rollover considerations from Section 4.1 and Section 4.4. Key replacement endures for a couple of Maximum Zone TTLs, depending on the rollover scenario. Therefore, a multiple of Maximum Zone TTL durations is a reasonable lower limit on the key effectivity period. Forcing a smaller key effectivity period will result in an unnecessary and inconveniently large DNSKEY RRset published in the zone.

The motivation for having the ZSK's effectivity period shorter than the KSK's effectivity period is rooted in the operational consideration that it is more likely that operators have more frequent read access to the ZSK than to the KSK. Thus, in cases where the ZSK cannot be afforded the same level of protection as the KSK (such as when zone keys are kept online), and where the risk of unauthorized disclosure of the ZSK's private key is not negligible (e.g. when HSMs are not in use), the ZSK's effectivity period should be kept shorter than the KSK's effectivity period.

In fact, if the risk of loss, theft, or other compromise is the same for a zone and key signing key, there is little reason to choose different effectivity periods for ZSKs and KSKs. And when the split between ZSKs and KSKs is not made, the argument is redundant.

There are certainly cases in which the use of a Single Type Signing Scheme with a long key effectivity period is a good choice, for example, where the costs and risks of compromise, and the costs and risks involved with having to perform an emergency roll are low.

3.4. Cryptographic Considerations

3.4.1. Signature Algorithm

At the time of writing, there are three types of signature algorithms that can be used in DNSSEC: RSA, DSA and GOST. Proposals for other algorithms are in the making. All three are fully specified in many freely-available documents, and are widely considered to be patent-free. The creation of signatures with RSA and DSA takes roughly the same time, but DSA is about ten times slower for signature verification. Also note that, in the context of DNSSEC, DSA is limited to a maximum of 1024 bit keys.

We suggest the use of RSA/SHA-256 as the preferred signature algorithms and RSA/SHA-1 as an alternative. Both have advantages and disadvantages. RSA/SHA-1 has been deployed for many years, while RSA/SHA-256 has only begun to be deployed. On the other hand, it is expected that if effective attacks on either algorithm appear, they will appear for RSA/SHA-1 first. RSA/MD5 should not be considered for use because RSA/MD5 will very likely be the first common-use signature algorithm to have an effective attack.

At the time of publication, it is known that the SHA-1 hash has cryptanalysis issues and work is in progress on addressing them. The use of public key algorithms based on hashes stronger than SHA-1 (e.g., SHA-256) is recommended, if these algorithms are available in implementations (see RFC 5702 [RFC5702] and RFC 4509 [RFC4509]).

Also at the time of publication, digital signature algorithms based on Elliptic Curve (EC) Cryptography with DNSSEC (GOST [RFC5933], ECDSA [RFC6605]) are being standardized and implemented. The use of EC has benefits in terms of size. On the other hand, one has to balance that against the amount of validating resolver implementations that will not recognize EC signatures and thus treat the zone as insecure. Beyond the observation of this trade-off, we will not discuss this further.

3.4.2. Key Sizes

This section assumes RSA keys, as suggested in the previous section.

DNSSEC signing keys should be large enough to avoid all known cryptographic attacks during the effectivity period of the key. To date, despite huge efforts, no one has broken a regular 1024-bit key; in fact, the best completed attack is estimated to be the equivalent of a 700-bit key. An attacker breaking a 1024-bit signing key would need to expend phenomenal amounts of networked computing power in a way that would not be detected in order to break a single key. Because of this, it is estimated that most zones can safely use 1024-bit keys for at least the next ten years (A 1024-bit asymmetric key has an approximate equivalent strength of a symmetric 80-bit key).

Depending on local policy (e.g. owners of keys that are used as extremely high value trust anchors, or non-anchor keys that may be difficult to roll over), it may be advisable to use lengths longer than 1024 bits. Typically, the next larger key size used is 2048 bits, which has the approximate equivalent strength of a symmetric 112-bit key (RFC 3766 [RFC3766]). Signing and verifying with a 2048-bit key takes longer than with a 1024-bit key. The increase depends on software and hardware implementations, but public operations (such as verification) are about four times slower, while private operations (such as signing) slow down about eight times.

Another way to decide on the size of key to use is to remember that the effort it takes for an attacker to break a 1024-bit key is the same regardless of how the key is used. If an attacker has the capability of breaking a 1024-bit DNSSEC key, he also has the capability of breaking one of the many 1024-bit TLS trust anchor keys that are currently installed in web browsers. If the value of a DNSSEC key is lower to the attacker than the value of a TLS trust anchor, the attacker will use the resources to attack the latter.

It is possible that there will be an unexpected improvement in the ability for attackers to break keys, and that such an attack would make it feasible to break 1024-bit keys but not 2048-bit keys. If such an improvement happens, it is likely that there will be a huge amount of publicity, particularly because of the large number of 1024-bit TLS trust anchors built into popular web browsers. At that time, all 1024-bit keys (both ones with parent zones and ones that are trust anchors) can be rolled over and replaced with larger keys.

Earlier documents (including the previous version of this document) urged the use of longer keys in situations where a particular key was "heavily used". That advice may have been true 15 years ago, but it is not true today when using RSA algorithms and keys of 1024 bits or

higher.

3.4.3. Private Key Storage

It is preferred that, where possible, zone private keys and the zone file master copy that is to be signed be kept and used in off-line, non-network-connected, physically secure machines only. Periodically, an application can be run to add authentication to a zone by adding RRSIG and NSEC/NSEC3 RRs. Then the augmented file can be transferred to the master.

When relying on dynamic update [RFC3007] or any other update mechanism that runs at a regular interval to manage a signed zone, be aware that at least one private key of the zone will have to reside on the master server, or reside on an HSM to which the server has access. This key is only as secure as the amount of exposure the server receives to unknown clients and on the level of security of the host. Although not mandatory, one could administer a zone using a "hidden master" scheme to minimize the risk. In this arrangement the master name server that processes the updates is unavailable to general hosts on the Internet; it is not listed in the NS RRset. The name servers in the NS RRset are able to receive zone updates through IXFR, AXFR, or an out-of-band distribution mechanism, possibly in combination with NOTIFY or another mechanism to trigger zone replication.

The ideal situation is to have a one-way information flow to the network to avoid the possibility of tampering from the network. Keeping the zone master on-line on the network and simply cycling it through an off-line signer does not do this. The on-line version could still be tampered with if the host it resides on is compromised. For maximum security, the master copy of the zone file should be off-net and should not be updated based on an unsecured network mediated communication.

The ideal situation may not be achievable because of economic tradeoffs between risks and costs. For instance, keeping a zone file off-line is not practical and will increase the costs of operating a DNS zone. So in practice the machines on which zone files are maintained will be connected to a network. Operators are advised to take security measures to shield unauthorized access to the master copy in order to prevent modification of DNS data before its signed.

Similarly the choice for storing a private key in an HSM will be influenced by a tradeoff between various concerns:

- o The risks that an unauthorized person has unnoticed read-access to the private key.

- o The remaining window of opportunity for the attacker.
- o The economic impact of the possible attacks (for a TLD that impact will typically be higher than for an individual users).
- o The costs of rolling the (compromised) keys. (The costs of rolling a ZSK is lowest and the costs of rolling a KSK that is in wide use as a trust anchor is highest.)
- o The costs of buying and maintaining an HSM.

For dynamically updated secured zones [RFC3007], both the master copy and the private key that is used to update signatures on updated RRs will need to be on-line.

3.4.4. Key Generation

Careful generation of all keys is a sometimes overlooked but absolutely essential element in any cryptographically secure system. The strongest algorithms used with the longest keys are still of no use if an adversary can guess enough to lower the size of the likely key space so that it can be exhaustively searched. Technical suggestions for the generation of random keys will be found in RFC 4086 [RFC4086] and NIST SP 800-90A [NIST-SP-800-90A]. In particular, one should carefully assess whether the random number generator used during key generation adheres to these suggestions. Typically, HSMs tend to provide a good facility for key generation.

Keys with a long effectivity period are particularly sensitive as they will represent a more valuable target and be subject to attack for a longer time than short-period keys. It is preferred that long-term key generation occur off-line in a manner isolated from the network via an air gap or, at a minimum, high-level secure hardware.

3.4.5. Differentiation for 'High-Level' Zones?

An earlier version of this document (RFC 4641 [RFC4641]) made a differentiation between key lengths for KSKs used for zones that are high in the DNS hierarchy and those for KSKs used low down.

This distinction is now considered not relevant. Longer key lengths for keys higher in the hierarchy are not useful because the cryptographic guidance is that everyone should use keys that no one can break. Also, it is impossible to judge which zones are more or less valuable to an attacker. An attack can only take place if the key compromise goes unnoticed and the attacker can act as a man-in-the-middle (MITM). For example, if example.com is compromised and the attacker forges answers for somebank.example.com. and sends them

out during an MITM, when the attack is discovered, it will be simple to prove that example.com has been compromised and the KSK will be rolled.

4. Signature Generation, Key Rollover, and Related Policies

4.1. Key Rollovers

Regardless of whether a zone uses periodic key rollovers, or only rolls keys in case of an irregular event, key rollovers are a fact of life when using DNSSEC. Zone administrators who are in the process of rolling their keys have to take into account that data published in previous versions of their zone still lives in caches. When deploying DNSSEC, this becomes an important consideration; ignoring data that may be in caches may lead to loss of service for clients.

The most pressing example of this occurs when zone material signed with an old key is being validated by a resolver that does not have the old zone key cached. If the old key is no longer present in the current zone, this validation fails, marking the data Bogus. Alternatively, an attempt could be made to validate data that is signed with a new key against an old key that lives in a local cache, also resulting in data being marked Bogus.

The typographic conventions used in the diagrams below are explained in Appendix B.

4.1.1. Zone Signing Key Rollovers

If the choice for splitting zone and key signing keys has been made, then those two types of keys can be rolled separately and zone signing keys can be rolled without taking into account DS records from the parent or the configuration of such a key as trust anchor.

For "Zone Signing Key rollovers", there are two ways to make sure that during the rollover data still cached can be verified with the new key sets or newly generated signatures can be verified with the keys still in caches. One scheme, described in Section 4.1.1.1, uses key pre-publication; the other uses double signatures, described in Section 4.1.1.2. The pros and cons are described in Section 4.1.1.3.

4.1.1.1. Pre-Publish Zone Signing Key Rollover

This section shows how to perform a ZSK rollover without the need to sign all the data in a zone twice -- the "Pre-Publish key rollover". This method has advantages in the case of a key compromise. If the old key is compromised, the new key has already been distributed in the DNS. The zone administrator is then able to quickly switch to the new key and remove the compromised key from the zone. Another major advantage is that the zone size does not double, as is the case with the Double Signature ZSK rollover.

Pre-Publish key rollover from DNSKEY_Z_10 to DNSKEY_Z_11 involves four stages as follows:

initial	new DNSKEY	new RRSIGs
SOA_0 RRSIG_Z_10(SOA)	SOA_1 RRSIG_Z_10(SOA)	SOA_2 RRSIG_Z_11(SOA)
DNSKEY_K_1 DNSKEY_Z_10 RRSIG_K_1(DNSKEY)	DNSKEY_K_1 DNSKEY_Z_10 DNSKEY_Z_11 RRSIG_K_1(DNSKEY)	DNSKEY_K_1 DNSKEY_Z_10 DNSKEY_Z_11 RRSIG_K_1(DNSKEY)

DNSKEY removal		

SOA_3 RRSIG_Z_11(SOA)		
DNSKEY_K_1 DNSKEY_Z_11		
RRSIG_K_1(DNSKEY)		

Figure 1: Pre-Publish Key Rollover

initial: Initial version of the zone: DNSKEY_K_1 is the Key Signing Key. DNSKEY_Z_10 is used to sign all the data of the zone, i.e., it is the Zone Signing Key.

new DNSKEY: DNSKEY_Z_11 is introduced into the key set (note that no signatures are generated with this key yet, but this does not secure against brute force attacks on its public key). The minimum duration of this pre-roll phase is the time it takes for the data to propagate to the authoritative servers, plus TTL value of the key set.

new RRSIGs: At the "new RRSIGs" stage , DNSKEY_Z_11 is used to sign the data in the zone exclusively (i.e., all the signatures from DNSKEY_Z_10 are removed from the zone). DNSKEY_Z_10 remains published in the key set. This way, data that was loaded into caches from the zone in the "new DNSKEY" step can still be verified with key sets fetched from this version of the zone. The minimum time that the key set including DNSKEY_Z_10 is to be published is the time that it takes for zone data from the

previous version of the zone to expire from old caches, i.e., the time it takes for this zone to propagate to all authoritative servers, plus the Maximum Zone TTL value of any of the data in the previous version of the zone.

DNSKEY removal: DNSKEY_Z_10 is removed from the zone. The key set, now only containing DNSKEY_K_1 and DNSKEY_Z_11, is re-signed with the DNSKEY_K_1.

The above scheme can be simplified by always publishing the "future" key immediately after the rollover. The scheme would look as follows (we show two rollovers); the future key is introduced in "new DNSKEY" as DNSKEY_Z_12 and again a newer one, numbered 13, in "new DNSKEY (II)":

initial	new RRSIGs	new DNSKEY
SOA_0 RRSIG_Z_10(SOA)	SOA_1 RRSIG_Z_11(SOA)	SOA_2 RRSIG_Z_11(SOA)
DNSKEY_K_1 DNSKEY_Z_10 DNSKEY_Z_11 RRSIG_K_1(DNSKEY)	DNSKEY_K_1 DNSKEY_Z_10 DNSKEY_Z_11 RRSIG_K_1(DNSKEY)	DNSKEY_K_1 DNSKEY_Z_11 DNSKEY_Z_12 RRSIG_K_1(DNSKEY)

new RRSIGs (II)		new DNSKEY (II)
SOA_3 RRSIG_Z_12(SOA)	SOA_4 RRSIG_Z_12(SOA)	
DNSKEY_K_1 DNSKEY_Z_11 DNSKEY_Z_12 RRSIG_K_1(DNSKEY)	DNSKEY_K_1 DNSKEY_Z_12 DNSKEY_Z_13 RRSIG_K_1(DNSKEY)	

Figure 2: Pre-Publish Zone Signing Key Rollover, Showing Two Rollovers

Note that the key introduced in the "new DNSKEY" phase is not used for production yet; the private key can thus be stored in a physically secure manner and does not need to be 'fetched' every time a zone needs to be signed.

4.1.1.2. Double Signature Zone Signing Key Rollover

This section shows how to perform a ZSK key rollover using the double zone data signature scheme, aptly named "Double Signature rollover".

During the "new DNSKEY" stage the new version of the zone file will need to propagate to all authoritative servers and the data that exists in (distant) caches will need to expire, requiring at least the propagation delay plus the Maximum Zone TTL of previous versions of the zone.

Double Signature ZSK rollover involves three stages as follows:

initial	new DNSKEY	DNSKEY removal
SOA_0	SOA_1	SOA_2
RRSIG_Z_10(SOA)	RRSIG_Z_10(SOA)	RRSIG_Z_11(SOA)
DNSKEY_K_1	DNSKEY_K_1	DNSKEY_K_1
DNSKEY_Z_10	DNSKEY_Z_10	DNSKEY_Z_11
RRSIG_K_1(DNSKEY)	RRSIG_K_1(DNSKEY)	RRSIG_K_1(DNSKEY)

Figure 3: Double Signature Zone Signing Key Rollover

initial: Initial Version of the zone: DNSKEY_K_1 is the Key Signing Key. DNSKEY_Z_10 is used to sign all the data of the zone, i.e., it is the Zone Signing Key.

new DNSKEY: At the "New DNSKEY" stage DNSKEY_Z_11 is introduced into the key set and all the data in the zone is signed with DNSKEY_Z_10 and DNSKEY_Z_11. The rollover period will need to continue until all data from version 0 (i.e. the version of the zone data containing SOA_0) of the zone has been replaced in all secondary servers and then has expired from remote caches. This will take at least the propagation delay plus the Maximum Zone TTL of version 0 of the zone.

DNSKEY removal: DNSKEY_Z_10 is removed from the zone as are all signatures created with it. The key set, now only containing DNSKEY_Z_11, is re-signed with DNSKEY_K_1 and DNSKEY_Z_11.

At every instance, RRSIGs from the previous version of the zone can be verified with the DNSKEY RRset from the current version and vice-versa. The duration of the "new DNSKEY" phase and the period between rollovers should be at least the propagation delay to secondary

servers plus the Maximum Zone TTL of the previous version of the zone.

Note that in this example we assumed for simplicity that the zone was not modified during the rollover. In fact new data can be introduced at anytime during this period, as long as it is signed with both keys.

4.1.1.3. Pros and Cons of the Schemes

Pre-Publish key rollover: This rollover does not involve signing the zone data twice. Instead, before the actual rollover, the new key is published in the key set and thus is available for cryptanalysis attacks. A small disadvantage is that this process requires four stages. Also the Pre-Publish scheme involves more parental work when used for KSK rollovers as explained in Section 4.1.2.

Double Signature ZSK rollover: The drawback of this approach is that during the rollover the number of signatures in your zone doubles; this may be prohibitive if you have very big zones. An advantage is that it only requires three stages.

4.1.2. Key Signing Key Rollovers

For the rollover of a Key Signing Key, the same considerations as for the rollover of a Zone Signing Key apply. However, we can use a Double Signature scheme to guarantee that old data (only the apex key set) in caches can be verified with a new key set and vice versa. Since only the key set is signed with a KSK, zone size considerations do not apply.

Note that KSK rollovers and ZSK rollovers are different in the sense that a KSK rollover requires interaction with the parent (and possibly replacing of trust anchors) and the ensuing delay while waiting for it.

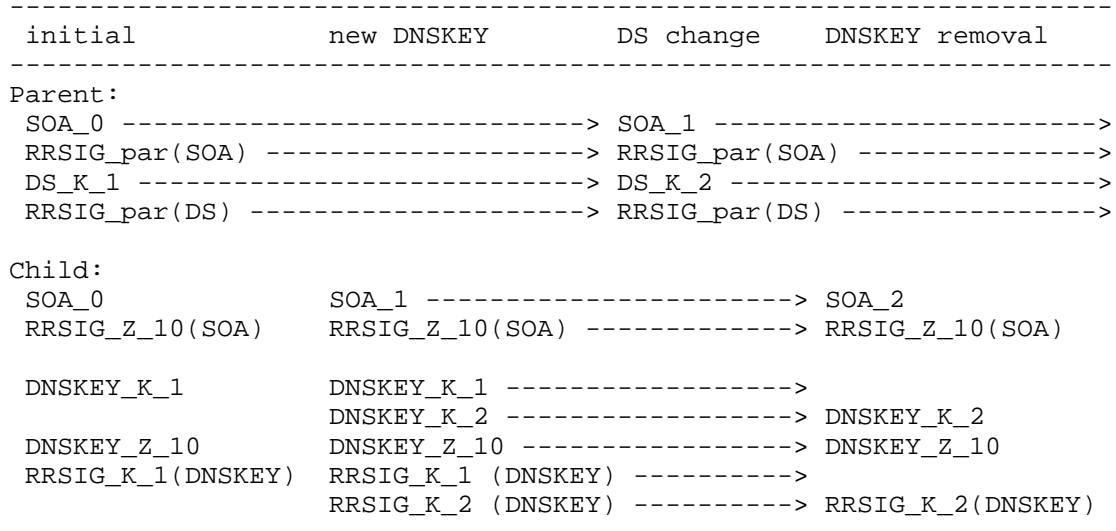


Figure 4: Stages of Deployment for a Double Signature Key Signing Key Rollover

initial: Initial version of the zone. The parental DS points to DNSKEY_K_1. Before the rollover starts, the child will have to verify what the TTL is of the DS RR that points to DNSKEY_K_1 -- it is needed during the rollover and we refer to the value as TTL_DS.

new DNSKEY: During the "new DNSKEY" phase, the zone administrator generates a second KSK, DNSKEY_K_2. The key is provided to the parent, and the child will have to wait until a new DS RR has been generated that points to DNSKEY_K_2. After that DS RR has been published on all servers authoritative for the parent's zone, the zone administrator has to wait at least TTL_DS to make sure that the old DS RR has expired from caches.

DS change: The parent replaces DS_K_1 with DS_K_2.

DNSKEY removal: DNSKEY_K_1 has been removed.

The scenario above puts the responsibility for maintaining a valid chain of trust with the child. It also is based on the premise that the parent only has one DS RR (per algorithm) per zone. An alternative mechanism has been considered. Using an established trust relation, the interaction can be performed in-band, and the removal of the keys by the child can possibly be signaled by the parent. In this mechanism, there are periods where there are two DS

RRs at the parent. This is known as a KSK Double-DS Rollover, and is shown in Figure 5. This method has some drawbacks for KSKs. We first describe the rollover scheme and then indicate these drawbacks.

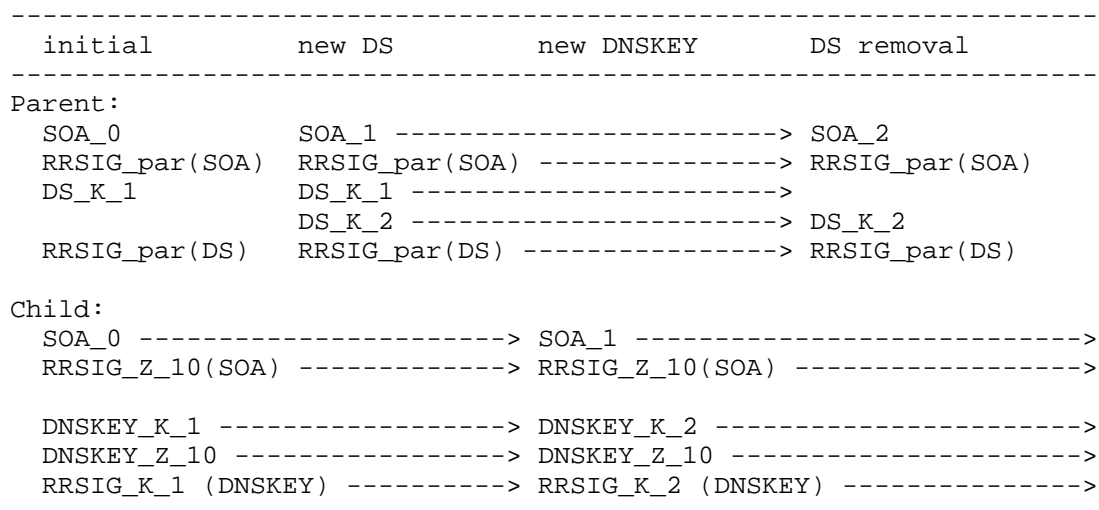


Figure 5: Stages of Deployment for a Double-DS Key Signing Key Rollover

When the child zone wants to roll, it notifies the parent during the "new DS" phase and submits the new key (or the corresponding DS) to the parent. The parent publishes DS_K_1 and DS_K_2, pointing to DNSKEY_K_1 and DNSKEY_K_2, respectively. During the rollover ("new DNSKEY" phase), which can take place as soon as the new DS set propagated through the DNS, the child replaces DNSKEY_K_1 with DNSKEY_K_2. If the old key has expired from caches, at the "DS removal" phase, the parent can be notified that the old DS record can be deleted.

The drawbacks of this scheme are that, during the "new DS" phase, the parent cannot verify the match between the DS_K_2 RR and DNSKEY_K_2 using the DNS -- as DNSKEY_K_2 is not yet published. Besides, we introduce a "security lame" key (see Section 4.3.3). Finally, the child-parent interaction consists of two steps. The "Double Signature" method only needs one interaction.

4.1.2.1. Special Considerations for RFC 5011 KSK rollover

The scenario sketched above assumes that the KSK is not in use as a trust anchor, but that validating name servers exclusively depend on

the parental DS record to establish the zone's security. If it is known that validating name servers have configured trust anchors, then that needs to be taken into account. Here we assume that zone administrators will deploy RFC 5011 [RFC5011] style rollovers.

RFC 5011 style rollovers increase the duration of key rollovers: the key to be removed must first be revoked. Thus, before the DNSKEY_K_1 removal phase, DNSKEY_K_1 must be published for one more Maximum Zone TTL with the REVOKE bit set. The revoked key must be self-signed, so in this phase the DNSKEY RRset must also be signed with DNSKEY_K_1.

4.1.3. Rollover for a Single Type Signing Scheme Key Rollover

The rollover of a key when a Single Type Signing Scheme is used is subject to the same requirement as the rollover of a KSK or ZSK: During any stage of the rollover, the chain of trust needs to continue to validate for any combination of data in the zone as well as data that may still live in distant caches.

There are two variants for this rollover. Since the choice for a Single Type Signing Scheme is motivated by operational simplicity we describe the most straightforward rollover scheme first.

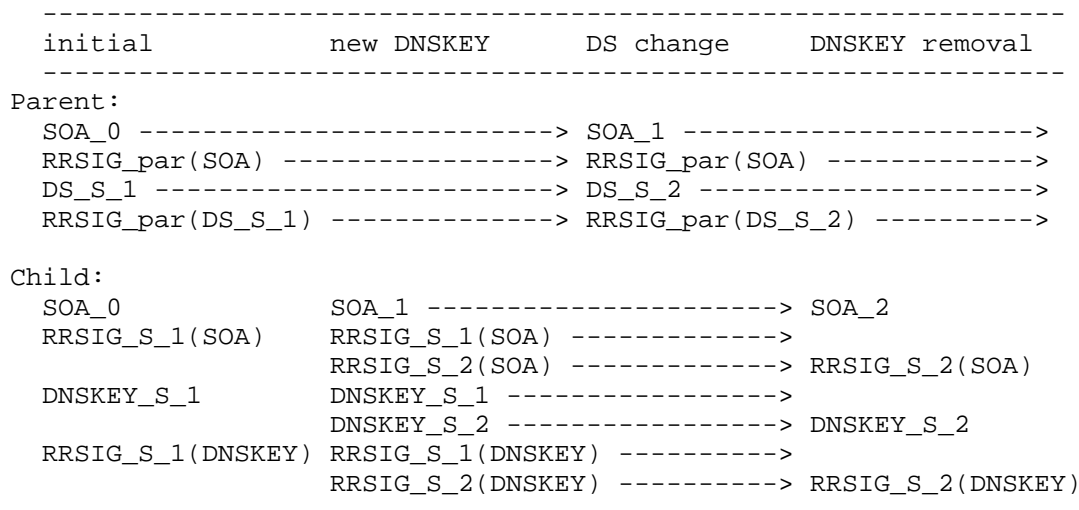


Figure 6: Stages of the Straightforward rollover in a Single Type Signing Scheme

initial: Parental DS points to DNSKEY_S_1. All RRsets in the zone are signed with DNSKEY_S_1.

new DNSKEY: A new key (DNSKEY_S_2) is introduced and all the RRsets are signed with both DNSKEY_S_1 and DNSKEY_S_2.

DS change: After the DNSKEY RRset with the two keys had time to propagate into distant caches (that is the key set exclusively containing DNSKEY_S_1 has been expired) the parental DS record can be changed.

DNSKEY removal: After the DS RRset containing DS_S_1 has expired from distant caches, DNSKEY_S_1 can be removed from the DNSKEY RRset.

In this first variant, the new signatures and new public key are added to the zone. Once they are propagated, the DS at the parent is switched. If the old DS has expired from the caches, the old signatures and old public key can be removed from the zone.

This rollover has the drawback that it introduces double signatures over all data of the zone. Taking these zone size considerations into account, it is possible to not introduce the signatures made with DNSKEY_S_2 at the "new DNSKEY" step. Instead, signatures of DNSKEY_S_1 are replaced with signatures of DNSKEY_S_2 in an additional stage between the "DS change" and "DNSKEY removal" step: After the DS RRset containing DS_S_1 has expired from distant caches, the signatures can be swapped. Only after the new signatures made with DNSKEY_S_2 have been propagated, the old public key DNSKEY_S_1 can be removed from the DNSKEY RRset.

The second variant of Single Type Signing Scheme Key rollover is the Double-DS rollover. In this variant, one introduces a new DNSKEY into the key set and submits the new DS to the parent. The new key is not yet used to sign RRsets. The signatures made with DNSKEY_S_1 are replaced with signatures made with DNSKEY_S_2 at the moment that DNSKEY_S_2 and DS_S_2 have been propagated.

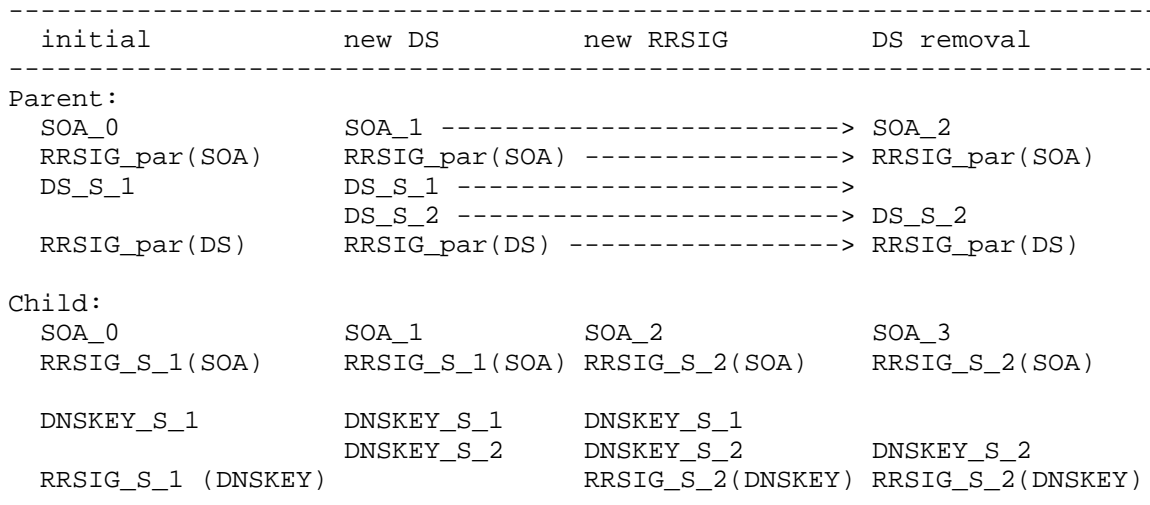


Figure 7: Stages of Deployment for a Double-DS Rollover in a Single Type Signing Scheme

4.1.4. Algorithm rollovers

A special class of key rollovers is the one needed for a change of signature algorithms (either adding a new algorithm, removing an old algorithm, or both). Additional steps are needed to retain integrity during this rollover. We first describe the generic case; special considerations for rollovers that involve trust anchors and single type keys are discussed later.

There exist both a conservative and a liberal approach for algorithm rollover. This has to do with section 2.2 in RFC 4035 [RFC4035]:

There MUST be an RRSIG for each RRset using at least one DNSKEY of each algorithm in the zone apex DNSKEY RRset. The apex DNSKEY RRset itself MUST be signed by each algorithm appearing in the DS RRset located at the delegating parent (if any).

The conservative approach interprets this section very strictly, meaning that it expects that every RRset has a valid signature for every algorithm signalled by the zone apex DNSKEY RRset - including RRsets in caches. The liberal approach uses a more loose interpretation of the section and limits the rule to RRsets in the zone at the authoritative name servers. There is a reasonable argument for saying that this is valid, because the specific section

is a subsection of section 2. in RFC 4035: Zone Signing.

When following the more liberal approach, algorithm rollover is just as easy as a regular Double-Signature KSK rollover (Section 4.1.2). Note that the Double-DS KSK rollover method cannot be used, since that would introduce a parental DS of which the apex DNSKEY RRset has not been signed with the introduced algorithm.

However, there are implementations of validators known that follow the more conservative approach. Performing a Double-Signature KSK algorithm rollover will temporarily make your zone appear as Bogus by such validators during the rollover. Therefore, the rollover described in this section will explain the stages of deployment assuming the conservative approach.

When adding a new algorithm, the signatures should be added first. After the TTL of RRSIGS has expired, and caches have dropped the old data covered by those signatures, the DNSKEY with the new algorithm can be added.

After the new algorithm has been added, the DS record can be exchanged using Double-Signature KSK Rollover.

When removing an old algorithm, the DS for the algorithm should be removed from the parent zone first, followed by the DNSKEY and the signatures (in the child zone).

Figure 8 describes the steps.

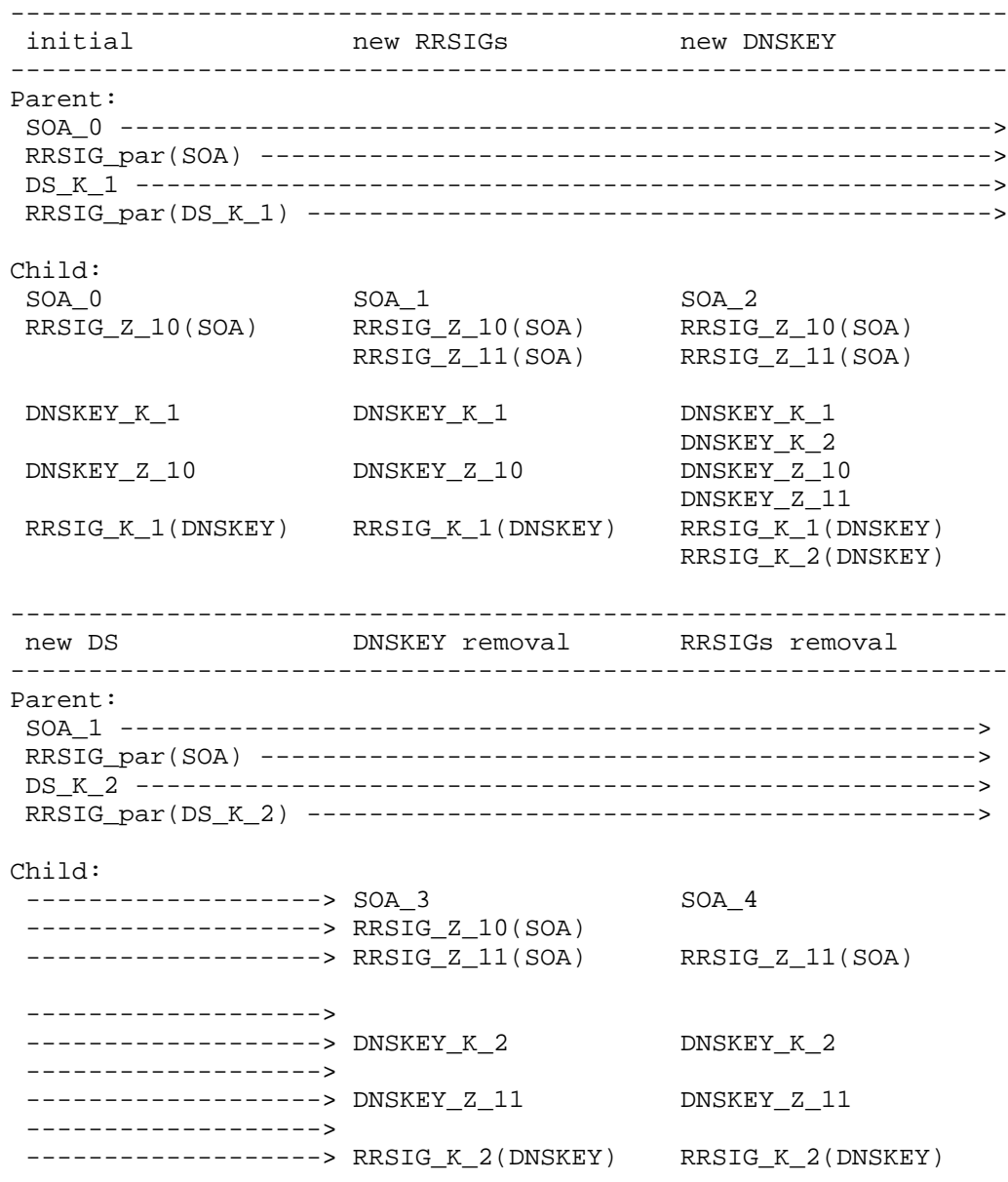


Figure 8: Stages of Deployment during an Algorithm Rollover

initial: Describes state of the zone before any transition is done.
The number of the keys may vary, but all keys (in DNSKEY records)
for the zone use the same algorithm.

new RRSIGs: The signatures made with the new key over all records in
the zone are added, but the key itself is not. This step is
needed to propagate the signatures created with the new algorithm
to the caches. If this is not done, it is possible for a resolver
to retrieve the new DNSKEY RRset (containing the new algorithm)
but to have RRsets in its cache with signatures created by the old
DNSKEY RRset (i.e. without the new algorithm).

The RRSIG for the DNSKEY RRset does not need to be pre-published
(since these records will travel together), and does not need
special processing in order to keep them synchronized.

new DNSKEY: After the old data has expired from caches, the new key
can be added to the zone.

new DS: After the cache data for the old DNSKEY RRset has expired,
the DS record for the new key can be added to the parent zone and
the DS record for the old key can be removed in the same step.

DNSKEY removal: After the cache data for the old DS RRset has
expired, the old algorithm can be removed. This time the old key
needs to be removed first, before removing the old signatures.

RRSIGs removal: After the cache data for the old DNSKEY RRset has
expired, the old signatures can also be removed during this step.

Below we deal with a few special cases of algorithm rollovers:

- 1: Single Type Signing Scheme Algorithm Rollover: when there is no
differentiation between Zone and Key signing keys
(Section 4.1.4.1).
- 2: RFC 5011 Algorithm Rollover: when trust anchors can track the
roll via RFC 5011 style rollover (Section 4.1.4.2).
- 3: 1 and 2 combined: when a Single Type Signing Scheme Algorithm
rollover is performed RFC 5011 style (Section 4.1.4.3).

In addition to the narrative below, these special cases are
represented in Figure 12, Figure 13 and Figure 14 in Appendix C.

4.1.4.1. Single Type Signing Scheme Algorithm Rollover

If one key is used that acts both as ZSK and KSK, the same scheme and figure as above (in Section 4.1.4, Figure 8) applies whereby all DNSKEY_Z_* records from the table are removed and all RRSIG_Z_* are replaced with RRSIG_S_*. All DNSKEY_K_* records are replaced with DNSKEY_S_* and all RRSIG_K_* records are replaced with RRSIG_S_*. The requirement to sign with both algorithms and make sure that old RRSIGS have the opportunity to expire from distant caches before introducing the new algorithm in the DNSKEY RRset is still valid.

This is shown in Figure 12 in Appendix C.

4.1.4.2. Algorithm rollover, RFC 5011 style

Trust anchor algorithm rollover is almost as simple as a regular RFC 5011 based rollover. However, the old trust anchor must be revoked before it is removed from the zone.

The timeline (see Figure 13 in Appendix C) is similar to that of Figure 8 above, but after the "new DS" step, an additional step is required where the DNSKEY is revoked. The details of this step ("revoke DNSKEY") are shown in figure Figure 9 below.


```

-----
  revoke DNSKEY
-----
Parent:
----->
----->
----->
----->

Child:
  SOA_3
  RRSIG_Z_10(SOA)
  RRSIG_Z_11(SOA)

  DNSKEY_K_1_REVOKED
  DNSKEY_K_2

  DNSKEY_Z_11
  RRSIG_K_1(DNSKEY)
  RRSIG_K_2(DNSKEY)
-----

```

Figure 9: The Revoke DNSKEY state that is added to an algorithm rollover when RFC 5011 is in use.

There is one exception to the requirement from RFC 4035 quoted in section 4.1.5 above: while all zone data must be signed with an unrevoked key, it is permissible to sign the key set with a revoked key. The somewhat esoteric argument is as follows:

Resolvers that do not understand the RFC 5011 Revoke flag will handle DNSKEY_K_1_REVOKED the same as if it was DNSKEY_K_1. In other words, they will handle the revoked key as a normal key, and thus RRsets signed with this key will validate. As a result, the signature matches the algorithm listed in the DNSKEY RRset.

Resolvers that do implement RFC 5011 will remove DNSKEY_K_1 from the set of trust anchors. That is okay, since they have already added DNSKEY_K_2 as the new trust anchor. Thus, algorithm 2 is the only signaled algorithm by now. That means, we only need RRSIG_K_2(DNSKEY) to authenticate the DNSKEY RRset, and we still are compliant with section 2.2 from RFC 4035: There must be an RRSIG for each RRset using at least one DNSKEY of each algorithm in the zone apex DNSKEY RRset.

4.1.4.3. Single Signing Type Algorithm Rollover, RFC 5011 style

If a decision is made to perform an RFC 5011 style rollover with a Single Signing Scheme key, it should be noted that section 2.1 of RFC 5011 states:

Once the resolver sees the REVOKE bit, it MUST NOT use this key as a trust anchor or for any other purpose except to validate the RRSIG it signed over the DNSKEY RRset specifically for the purpose of validating the revocation.

This means that if once DNSKEY_S_1 is revoked, it cannot be used to validate its signatures over non-DNSKEY RRsets. Thus, those RRsets should be signed with a shadow key, DNSKEY_Z_10, during the algorithm rollover. The shadow key can be removed at the same time the revoked DNSKEY_S_1 is removed from the zone. In other words, the zone must temporarily fall back to a KSK/ZSK split model during the rollover.

In other words, the rule that at every RRset there must be at least one signature for each algorithm used in the DNSKEY RRset still applies. This means that a different key with the same algorithm, other than the revoked key, must sign the entire zone. Thus, more operations are needed if the Single Type Signing Scheme is used. Before rolling the algorithm, a new key must be introduced with the same algorithm as the key that is candidate for revocation. That key can then temporarily act as ZSK during the algorithm rollover.

As with algorithm rollover RFC 5011 style, while all zone data must be signed with an unrevoked key, it is permissible to sign the key set with a revoked key using the same esoteric argument given in Section 4.1.4.2.

The lesson of all of this is that a Single Type Signing Scheme algorithm rollover using RFC 5011 is as complicated as the name of the rollover implies: reverting to a split key scheme for the duration of the rollover may be preferable.

4.1.4.4. NSEC to NSEC3 algorithm rollover

A special case is the rollover from an NSEC signed zone to an NSEC3 signed zone. In this case algorithm numbers are used to signal support for NSEC3 but they do not mandate the use of NSEC3. Therefore, NSEC records should remain in the zone until the rollover to a new algorithm has completed and the new DNSKEY RRset has populated distant caches, at the end of the "new DNSKEY" stage. At that point the validators that have not implemented NSEC3 will treat

the zone as unsecured as soon as they follow the chain of trust to the DS that points to a DNSKEY of the new algorithm, while validators that support NSEC3 will happily validate using NSEC. Turning on NSEC3 can then be done during the "new DS" step: increasing the serial number, introducing the NSEC3PARAM record to signal that NSEC3 authenticated denial of existence data should be served, and resigning the zone.

Summarizing, an NSEC to NSEC3 rollover is an ordinary algorithm rollover whereby NSEC is used all the time and only after that rollover finished NSEC3 needs to be deployed. The procedures are also listed in Sections 10.4 and 10.5 of RFC 5155 [RFC5155].

4.1.5. Considerations for Automated Key Rollovers

As keys must be renewed periodically, there is some motivation to automate the rollover process. Consider the following:

- o ZSK rollovers are easy to automate as only the child zone is involved.
- o A KSK rollover needs interaction between parent and child. Data exchange is needed to provide the new keys to the parent; consequently, this data must be authenticated and integrity must be guaranteed in order to avoid attacks on the rollover.

4.2. Planning for Emergency Key Rollover

This section deals with preparation for a possible key compromise. It is advised to have a documented procedure ready for when a key compromise is suspected or confirmed.

When the private material of one of a zone's keys is compromised it can be used by an attacker for as long as a valid trust chain exists. A trust chain remains intact for

- o as long as a signature over the compromised key in the trust chain is valid, and
- o as long as the DS RR in the parent zone points to the (compromised) key signing the DNSKEY RRset, and
- o as long as the (compromised) key is anchored in a resolver and is used as a starting point for validation (this is generally the hardest to update).

While a trust chain to a zone's compromised key exists, your namespace is vulnerable to abuse by anyone who has obtained

illegitimate possession of the key. Zone administrators have to make a decision as to whether the abuse of the compromised key is worse than having data in caches that cannot be validated. If the zone administrator chooses to break the trust chain to the compromised key, data in caches signed with this key cannot be validated. However, if the zone administrator chooses to take the path of a regular rollover, during the rollover the malicious key holder can continue to spoof data so that it appears to be valid.

4.2.1. KSK Compromise

A compromised KSK can be used to sign the key set of an attacker's version of the zone. That zone could be used to poison the DNS.

A zone containing a DNSKEY RRset with a compromised KSK is vulnerable as long as the compromised KSK is configured as trust anchor or a DS record in the parent zone points to it.

Therefore, when the KSK has been compromised, the trust anchor or the parent DS record should be replaced as soon as possible. It is local policy whether to break the trust chain during the emergency rollover. The trust chain would be broken when the compromised KSK is removed from the child's zone while the parent still has a DS record pointing to the compromised KSK. The assumption is that there is only one DS record at the parent. If there are multiple DS records this does not apply, although the chain of trust of this particular key is broken.

Note that an attacker's version of the zone still uses the compromised KSK and the presence of the corresponding DS record in the parent would cause the data in this zone to appear as valid. Removing the compromised key would cause the attacker's version of the zone to appear as valid and the original zone as Bogus. Therefore, we advise not to remove the KSK before the parent has a DS record for the new KSK in place.

4.2.1.1. Emergency Key Rollover Keeping the Chain of Trust Intact

If it is desired to perform an emergency key rollover in a manner that keeps the chain of trust intact, the timing of the replacement of the KSK is somewhat critical. The goal is to remove the compromised KSK as soon as the new DS RR is available at the parent. This means ensuring that the signature made with a new KSK over the key set that contains the compromised KSK expires just after the new DS appears at the parent. Expiration of that signature will cause expiration of that key set from the caches.

The procedure is as follows:

1. Introduce a new KSK into the key set, keep the compromised KSK in the key set. Lower the TTL for DNSKEYs so that the DNSKEY RRset will expire from caches sooner.
2. Sign the key set, with a short validity period. The validity period should expire shortly after the DS is expected to appear in the parent and the old DSes have expired from caches. This provides an upper limit on how long the compromised KSK can be used in a replay attack.
3. Upload the DS for this new key to the parent.
4. Follow the procedure of the regular KSK rollover: Wait for the DS to appear at the authoritative servers and then wait as long as the TTL of the old DS RRs. If necessary, re-sign the DNSKEY RRset and modify/extend the expiration time.
5. Remove the compromised DNSKEY RR from the zone and re-sign the key set using your "normal" TTL and signature validity period.

An additional danger of a key compromise is that the compromised key could be used to facilitate a legitimately looking DNSKEY/DS rollover and/or name server changes at the parent. When that happens, the domain may be in dispute. An authenticated out-of-band and secure notify mechanism to contact a parent is needed in this case.

Note that this is only a problem when the DNSKEY and or DS records are used to authenticate communication with the parent.

4.2.1.2. Emergency Key Rollover Breaking the Chain of Trust

There are two methods to perform an emergency key rollover in a manner that breaks the chain of trust. The first method causes the child zone to appear Bogus to validating resolvers. The other causes the child zone to appear Insecure. These are described below.

In the method that causes the child zone to appear Bogus to validating resolvers, the child zone replaces the current KSK with a new one and re-signs the key set. Next, it sends the DS of the new key to the parent. Only after the parent has placed the new DS in the zone is the child's chain of trust repaired. Note that until that time, the child zone is still vulnerable to spoofing: the attacker is still in possession of the compromised key that the DS points to.

An alternative method of breaking the chain of trust is by removing the DS RRs from the parent zone altogether. As a result, the child zone would become Insecure. After the DS has expired from distant

caches, the keys and signatures are removed from the child zone, new keys and signatures are introduced and finally, a new DS is submitted to the parent.

4.2.2. ZSK Compromise

Primarily because there is no interaction with the parent required when a ZSK is compromised, the situation is less severe than with a KSK compromise. The zone must still be re-signed with a new ZSK as soon as possible. As this is a local operation and requires no communication between the parent and child, this can be achieved fairly quickly. However, one has to take into account that -- just as with a normal rollover -- the immediate disappearance of the old compromised key may lead to verification problems. Also note that until the RRSIG over the compromised ZSK has expired, the zone may be still at risk.

4.2.3. Compromises of Keys Anchored in Resolvers

A key can also be pre-configured in resolvers as a trust anchor. If trust anchor keys are compromised, the administrators of resolvers using these keys should be notified of this fact. Zone administrators may consider setting up a mailing list to communicate the fact that a SEP key is about to be rolled over. This communication will of course need to be authenticated by some means, e.g. by using digital signatures.

End-users faced with the task of updating an anchored key should always verify the new key. New keys should be authenticated out-of-band, for example, through the use of an announcement website that is secured using Transport Layer Security (TLS) [RFC5246].

4.2.4. Stand-by Keys

Stand-by keys are keys that are published in your zone, but are not used to sign RRsets. There are two reasons why someone would want to use stand-by keys. One is to speed up the emergency key rollover. The other is to recover from a disaster that leaves your production private keys inaccessible.

The way to deal with stand-by keys differs for ZSKs and KSKs. To make a stand-by ZSK, you need to publish its DNSKEY RR. To make a stand-by KSK, you need to get its DS RR published at the parent.

Assuming you have your normal DNS operation, to prepare stand-by keys you need to:

- o Generate a stand-by ZSK and KSK. Store them safely in a different location to the place where the currently used ZSK and KSK are held.
- o Pre-publish the DNSKEY RR of the stand-by ZSK in the zone.
- o Pre-publish the DS of the stand-by KSK in the parent zone.

Now suppose a disaster occurs and disables access to the currently used keys. To recover from that situation, follow these procedures:

- o Set up your DNS operations and introduce the stand-by KSK into the zone.
- o Post-publish the disabled ZSK and sign the zone with the stand-by keys.
- o After some time, when the new signatures have been propagated, the old keys, old signatures and the old DS can be removed.
- o Generate a new stand-by key set at a different location and continue "normal" operation.

4.3. Parent Policies

4.3.1. Initial Key Exchanges and Parental Policies Considerations

The initial key exchange is always subject to the policies set by the parent. It is specifically important in a registry-registrar-registrant model where a registry maintains the parent zone, and the registrant (the user of the child-domain name) deals with the registry through an intermediary called a registrar (see [RFC3375] for a comprehensive definition). The key material is to be passed from the DNS operator to the parent via a registrar, where both DNS operator and registrar are selected by the registrant and might be different organisations. When designing a key exchange policy, one should take into account that the authentication and authorization mechanisms used during a key exchange should be as strong as the authentication and authorization mechanisms used for the exchange of delegation information between parent and child. That is, there is no implicit need in DNSSEC to make the authentication process stronger than it is for regular DNS.

Using the DNS itself as the source for the actual DNSKEY material has the benefit that it reduces the chances of user error. A DNSKEY query tool can make use of the SEP bit [RFC4035] to select the proper key(s) from a DNSSEC key set, thereby reducing the chance that the wrong DNSKEY is sent. It can validate the self-signature over a key;

thereby verifying the ownership of the private key material. Fetching the DNSKEY from the DNS ensures that the chain of trust remains intact once the parent publishes the DS RR indicating the child is secure.

Note: out-of-band verification is still needed when the key material is fetched for the first time, even via DNS. The parent can never be sure whether or not the DNSKEY RRs have been spoofed.

With some type of key rollovers, the DNSKEY is not pre-published and a DNSKEY query tool is not able to retrieve the successor key. In this case, the out-of-band method is required. This also allows the child to determine the digest algorithm of the DS record.

4.3.2. Storing Keys or Hashes?

When designing a registry system one should consider whether to store the DNSKEYs and/or the corresponding DSes. Since a child zone might wish to have a DS published using a message digest algorithm not yet understood by the registry, the registry can't count on being able to generate the DS record from a raw DNSKEY. Thus, we suggest that registry systems should be able to store DS RRs, even if they also store DNSKEYs (see also draft-ietf-dnsop-dnssec-trust-anchor [I-D.ietf-dnsop-dnssec-trust-anchor]).

The storage considerations also relate to the design of the customer interface and the method by which data is transferred between registrant and registry: will the child zone administrator be able to upload DS RRs with unknown hash algorithms or does the interface only allow DNSKEYs? When registries support the Extensible Provisioning Protocol (EPP) [RFC5910], that can be used for registrar-registry interactions since that protocol allows the transfer of both DS and, optionally, DNSKEY RRs. There is no standardized way for moving the data between the customer and the registrar. Different registrars have different mechanisms, ranging from simple web interfaces to various APIs. In some cases the use of the DNSSEC extensions to EPP may be applicable.

Having an out-of-band mechanism, such as a registry directory (e.g., Whois), to find out which keys are used to generate DS Resource Records for specific owners and/or zones may also help with troubleshooting.

4.3.3. Security Lameness

Security lameness is defined as the state whereby the parent has a DS RR pointing to a non-existent DNSKEY RR. Security lameness may occur temporarily during a Double-DS rollover scheme. However care should

be taken that not all DS RRs are pointing to a non-existing DNSKEY RR, which will cause the child's zone to be marked Bogus by verifying DNS clients.

As part of a comprehensive delegation check, the parent could, at key exchange time, verify that the child's key is actually configured in the DNS. However, if a parent does not understand the hashing algorithm used by the child, the parental checks are limited to only comparing the key id.

Child zones should be very careful in removing DNSKEY material, specifically SEP keys, for which a DS RR exists.

Once a zone is "security lame", a fix (e.g., removing a DS RR) will take time to propagate through the DNS.

4.3.4. DS Signature Validity Period

Since the DS can be replayed as long as it has a valid signature, a short signature validity period for the DS RRSIG minimizes the time a child is vulnerable in the case of a compromise of the child's KSK(s). A signature validity period that is too short introduces the possibility that a zone is marked Bogus in case of a configuration error in the signer. There may not be enough time to fix the problems before signatures expire (this is a generic argument; also see Section 4.4.2). Something as mundane as zone administrator unavailability during weekends shows the need for DS signature validity periods longer than two days. Just like any signature validity period, we suggest an absolute minimum for the DS signature validity period of a few days.

The maximum signature validity period of the DS record depends on how long child zones are willing to be vulnerable after a key compromise. On the other hand, shortening the DS signature validity period increases the operational risk for the parent. Therefore, the parent may have policy to use a signature validity period that is considerably longer than the child would hope for.

A compromise between the policy/operational constraints of the parent and minimizing damage for the child may result in a DS signature validity period somewhere between a week and several months.

In addition to the signature validity period, which sets a lower bound on the number of times the zone administrator will need to sign the zone data, and an upper bound to the time a child is vulnerable after key compromise, there is the TTL value on the DS RRs. Shortening the TTL reduces the damage of a successful replay attack. It does mean that the authoritative servers will see more queries.

But on the other hand, a short TTL lowers the persistence of DS RRsets in caches thereby increasing the speed with which updated DS RRsets propagate through the DNS.

4.3.5. Changing DNS Operators

The parent-child relation is often described in terms of a registry-registrar-registrant model, where a registry maintains the parent zone, and the registrant (the user of the child-domain name) deals with the registry through an intermediary called a registrar [RFC3375]. Registrants may out-source the maintenance of their DNS system, including the maintenance of DNSSEC key material, to the registrar or to another third party, referred to here as the DNS operator.

For various reasons, a registrant may want to move between DNS operators. How easy this move will be depends principally on the DNS operator from which the registrant is moving (the losing operator), as they have control over the DNS zone and its keys. The following sections describe the two cases: where the losing operator cooperates with the new operator (the gaining operator), and where the two do not cooperate.

4.3.5.1. Cooperating DNS operators

In this scenario, it is assumed that the losing operator will not pass any private key material to the gaining operator (that would constitute a trivial case) but is otherwise fully cooperative.

In this environment, the change could be made with a Pre-Publish ZSK rollover whereby the losing operator pre-publishes the ZSK of the gaining operator, combined with a Double Signature KSK rollover where the two registrars exchange public keys and independently generate a signature over those key sets that they combine and both publish in their copy of the zone. Once that is done they can use their own private keys to sign any of their zone content during the transfer.

initial		pre-publish	
Parent:			
NS_A		NS_A	
DS_A		DS_A	
Child at A:			
SOA_A0		SOA_A1	
RRSIG_Z_A(SOA)		RRSIG_Z_A(SOA)	
NS_A		NS_A	
RRSIG_Z_A(NS)		NS_B	
		RRSIG_Z_A(NS)	
DNSKEY_Z_A		DNSKEY_Z_A	
		DNSKEY_Z_B	
DNSKEY_K_A		DNSKEY_K_A	
		DNSKEY_K_B	
RRSIG_K_A(DNSKEY)		RRSIG_K_A(DNSKEY)	
		RRSIG_K_B(DNSKEY)	
Child at B:			
		SOA_B0	
		RRSIG_Z_B(SOA)	
		NS_B	
		RRSIG_Z_B(NS)	
		DNSKEY_Z_A	
		DNSKEY_Z_B	
		DNSKEY_K_A	
		DNSKEY_K_B	
		RRSIG_K_A(DNSKEY)	
		RRSIG_K_B(DNSKEY)	
relegation			
post migration			
Parent:			
NS_B		NS_B	
DS_B		DS_B	
Child at A:			
SOA_A1		SOA_B0	
RRSIG_Z_A(SOA)		RRSIG_Z_B(SOA)	
NS_A		NS_B	
NS_B		RRSIG_Z_B(NS)	
RRSIG_Z_A(NS)		RRSIG_Z_B(NS)	
DNSKEY_Z_A		DNSKEY_Z_A	
DNSKEY_Z_B		DNSKEY_Z_B	
DNSKEY_K_A		DNSKEY_Z_B	
DNSKEY_K_B		DNSKEY_K_A	
RRSIG_K_A(DNSKEY)		DNSKEY_K_B	
RRSIG_K_B(DNSKEY)		RRSIG_K_A(DNSKEY)	
		RRSIG_K_B(DNSKEY)	

Figure 10: Rollover for cooperating operators

In this figure A denotes the losing operator and B the gaining operator. RRSIG_Z is the RRSIG produced by a ZSK, RRSIG_K is produced with a KSK, the appended A or B indicates the producers of the key pair. "Child at A" is how the zone content is represented by the losing DNS operator and "Child at B" is how the zone content is represented by the gaining DNS operator.

The zone is initially delegated from the parent to the name servers of operator A. Operator A uses his own ZSK and KSK to sign the zone. The cooperating operator A will pre-publish the new NS record and the ZSK and KSK of operator B, including the RRSIG over the DNSKEY RRset generated by the KSK of B. Operator B needs to publish the same DNSKEY RRset. When that DNSKEY RRset has populated the caches, the re-delegation can be made, which involves adjusting the NS and DS records in the parent zone to point to operator B. And after all DNSSEC records related to A have expired from the caches, operator B can stop publishing the keys and signatures belonging to operator A and vice versa.

The requirement to exchange signatures has a couple of drawbacks. It requires more operational overhead, because not only the operators have to exchange public keys, they also have to exchange the signatures of the new DNSKEY RRset. This drawback does not exist if the Double Signature KSK rollover is replaced with a Double-DS KSK rollover. See Figure 15 in Appendix D for the diagram.

Thus, if the registry and registrars allow DS records to be published that do not point to a published DNSKEY in the child zone, the Double-DS KSK rollover is preferred (see Figure 5), in combination with the Pre-Publish ZSK rollover. This does not require sharing the KSK signatures between the operators, but both operators still have to publish each others ZSKs.

4.3.5.2. Non Cooperating DNS Operators

In the non-cooperative case matters are more complicated. The losing operator may not cooperate and leave the data in the DNS as is. In the extreme case the losing operator may become obstructive and publish a DNSKEY RR with a high TTL and corresponding signature validity period so that registrar A's DNSKEY could end up in caches for (in theory at least) tens of years.

The problem arises when a validator tries to validate with the losing operator's key and there is no signature material produced with the losing operator available in the delegation path after re-delegation from the losing operator to the gaining operator has taken place. One could imagine a rollover scenario where the gaining operator takes a copy of all RRSIGs created by the losing operator and

publishes those in conjunction with its own signatures, but that would not allow any changes in the zone content. Since a re-delegation took place, the NS RRset has - by definition - changed so such rollover scenario will not work. Besides if zone transfers are not allowed by the losing operator and NSEC3 is deployed in the losing operator's zone, then the gaining operator's zone will not have certainty that all of A's RRSIGs have been copied.

The only viable operation for the registrant is to have their zone go Insecure for the duration of the change. The registry should be asked to remove the DS RR pointing to the losing operator's DNSKEY and to change the NS RRset to point to the gaining operator. Once this has propagated through the DNS, the registry should be asked to insert the DS record pointing to the (newly signed) zone at operator B.

Note that some behavior of resolver implementations may aid in the process of changing DNS operators:

- o TTL sanity checking, as described in RFC 2308 [RFC2308], will limit the impact the actions of an obstructive, losing operator. Resolvers that implement TTL sanity checking will use an upper limit for TTLs on RRsets in responses.
- o If RRsets at the zone cut (are about to) expire, the resolver restarts its search above the zone cut. Otherwise, the resolver risks to keep using a name server that might be un-delegated by the parent.
- o Limiting the time DNSKEYs that seem to be unable to validate signatures are cached and/or trying to recover from cases where DNSKEYs do not seem to be able to validate data, also reduces the effects of the problem of non-cooperating registrars.

However, there is no operational methodology to work around this business issue, and proper contractual relationships between all involved parties seems to be the only solution to cope with these problems. It should be noted that in many cases, the problem with temporary broken delegations already exists when a zone changes from one DNS operator to another. Besides, it is often the case that when operators are changed, the services that that zone references also change operator, possibly involving some downtime.

In any case, to minimise such problems, the classic configuration is to have relative short TTL on all involved resource records. That will solve many of the problems regarding changes to a zone regardless of whether DNSSEC is used.

4.4. Time in DNSSEC

Without DNSSEC, all times in the DNS are relative. The SOA fields REFRESH, RETRY, and EXPIRATION are timers used to determine the time elapsed after a slave server synchronized with a master server. The Time to Live (TTL) value and the SOA RR minimum TTL parameter [RFC2308] are used to determine how long a forwarder should cache data (or negative responses) after it has been fetched from an authoritative server. By using a signature validity period, DNSSEC introduces the notion of an absolute time in the DNS. Signatures in DNSSEC have an expiration date after which the signature is marked as invalid and the signed data is to be considered Bogus.

The considerations in this section are all qualitative and focused on the operational and managerial issues. A more thorough quantitative analysis of rollover timing parameters can be found in draft-ietf-dnsop-dnssec-key-timing [I-D.ietf-dnsop-dnssec-key-timing].

4.4.1. Time Considerations

Because of the expiration of signatures, one should consider the following:

- o We suggest the Maximum Zone TTL value of your zone data to be smaller than your signature validity period.

If the TTL was of similar order as the signature validity period, then all RRsets fetched during the validity period would be cached until the signature expiration time. Section 8.1 of RFC 4033 [RFC4033] suggests that "the resolver may use the time remaining before expiration of the signature validity period of a signed RRset as an upper bound for the TTL". As a result, query load on authoritative servers would peak at signature expiration time, as this is also the time at which records simultaneously expire from caches.

Having a TTL that is at least a few times smaller than your signature validity period avoids query load peaks.

- o We suggest the signature publication period to end at least one Maximum Zone TTL duration (but preferably a minimum of a few days) before the end of the signature validity period.

Re-signing a zone shortly before the end of the signature validity period may cause simultaneous expiration of data from caches. This in turn may lead to peaks in the load on authoritative servers. To avoid this, schemes are deployed

whereby the zone is periodically visited for a re-signing operation and those signatures that are within a so called Refresh Period from signature expiration are recreated. Also see Section 4.4.2 below.

In case of an operational error, you would have one Maximum Zone TTL duration to resolve the problem. Re-signing a zone a few days before the end of the signature validity period ensures the signatures will survive at least a (long) weekend in case of such operational havoc. This is called the Refresh Period (see Section 4.4.2).

- o We suggest the Minimum Zone TTL to be long enough to both fetch and verify all the RRs in the trust chain. In workshop environments, it has been demonstrated [NIST-workshop] that a low TTL (under 5 to 10 minutes) caused disruptions because of the following two problems:
 - 1. During validation, some data may expire before the validation is complete. The validator should be able to keep all data until it is completed. This applies to all RRs needed to complete the chain of trust: DS, DNSKEY, RRSIG, and the final answers, i.e., the RRset that is returned for the initial query.
 - 2. Frequent verification causes load on recursive name servers. Data at delegation points, DS, DNSKEY, and RRSIG RRs benefit from caching. The TTL on those should be relatively long. Data at the leafs in the DNS tree has less impact on recursive name servers.
- o Slave servers will need to be able to fetch newly signed zones well before the RRSIGs in the zone served by the slave server pass their signature expiration time.

When a slave server is out of synchronization with its master and data in a zone is signed by expired signatures, it may be better for the slave server not to give out any answer.

Normally, a slave server that is not able to contact a master server for an extended period will expire a zone. When that happens, the server will respond differently to queries for that zone. Some servers issue SERVFAIL, whereas others turn off the 'AA' bit in the answers. The time of expiration is set in the SOA record and is relative to the last successful refresh between the master and the slave servers. There exists no coupling between the signature expiration of RRSIGs in the zone and the expire parameter in the SOA.

If the server serves a DNSSEC-secured zone, then it may happen that the signatures expire well before the SOA expiration timer counts down to zero. It is not possible to completely prevent this by modifying the SOA parameters.

However, the effects can be minimized where the SOA expiration time is equal to or shorter than the Refresh Period (see Section 4.4.2).

The consequence of an authoritative server not being able to update a zone for an extended period of time is that signatures may expire. In this case, non-secure resolvers will continue to be able to resolve data served by the particular slave servers while security-aware resolvers will experience problems because of answers being marked as Bogus.

We suggest the SOA expiration timer being approximately one third or a quarter of the signature validity period. It will allow problems with transfers from the master server to be noticed before signatures time out.

We also suggest that operators of name servers that supply secondary services develop systems to identify upcoming signature expirations in zones they slave and take appropriate action where such an event is detected.

When determining the value for the expiration parameter one has to take the following into account: what are the chances that all secondaries expire the zone? How quickly can the administrators of the secondary servers be reached to load a valid zone? These questions are not DNSSEC-specific but may influence the choice of your signature validity periods.

4.4.2. Signature Validity Periods

4.4.2.1. Maximum Value

The first consideration for choosing a maximum signature validity period is the risk of a replay attack. For low-value, long-term stable resources, the risks may be minimal and the signature validity period may be several months. Although signature validity periods of many years are allowed, the same operational habit arguments as given in Section 3.2.2 play a role: when a zone is re-signed with some regularity, then zone administrators remain conscious about the operational necessity of re-signing.

4.4.2.2. Minimum Value

The minimum value of the signature validity period is set for the time by which one would like to survive operational failure in provisioning: what is the time that a failure will be noticed, what is the time that action is expected to be taken? By answering these questions, availability of zone administrators during (long) weekends or time taken to access backup media can be taken into account. The result could easily suggest a minimum signature validity period of a few days.

Note however, the argument above is assuming that zone data has just been signed and published when the problem occurred. In practice it may be that a zone is signed according to a frequency set by the Re-Sign Period whereby the signer visits the zone content and only refreshes signatures that are within a given amount of time (the Refresh Period) of expiration. The Re-Sign Period must be smaller than the Refresh Period in order for zone data to be signed in a timely fashion.

If an operational problem occurs during re-signing, then the signatures in the zone to expire first are the ones that have been generated longest ago. In the worst case, these signatures are the Refresh Period minus the Re-Sign Period away from signature expiration.

To make matters slightly more complicated, some signers vary the signature validity period over a small range (the jitter interval) so that not all signatures expire at the same time.

In other words, the minimum signature validity period is set by first choosing the Refresh Period (usually a few days), then defining the Re-Sign Period in such a way that the Refresh Period minus the Re-Sign Period, minus the maximum jitter sets the time in which operational havoc can be resolved.

The relationship between signature times is illustrated in Figure 11.

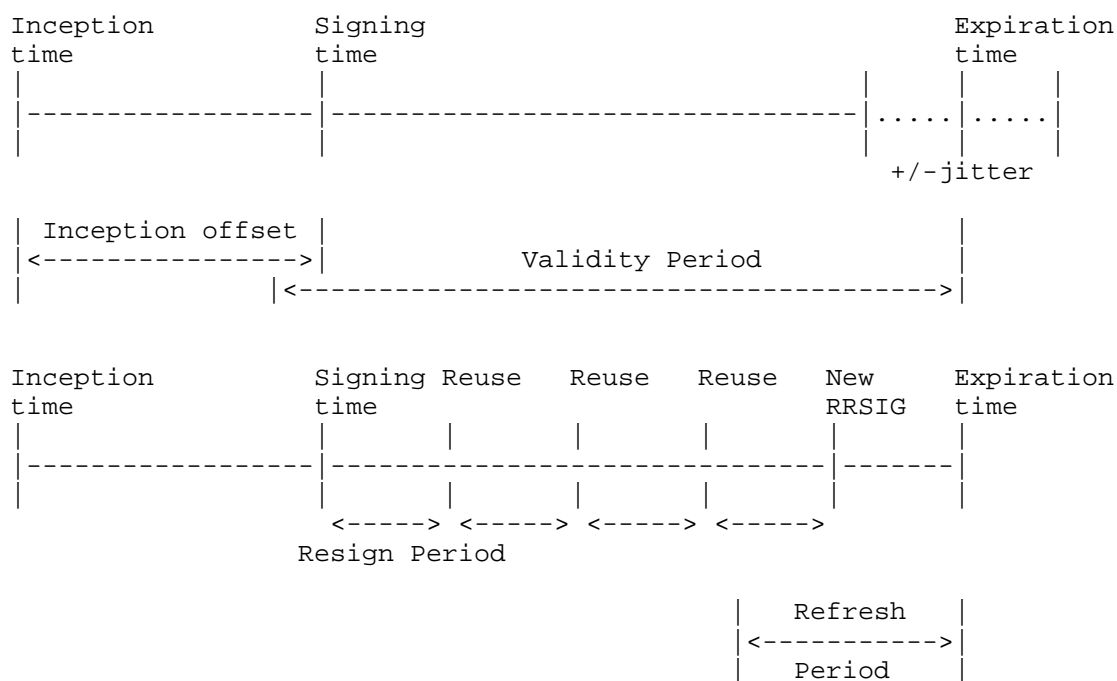


Figure 11: Signature Timing Parameters

Note that in the figure the validity of the signature starts shortly before the signing time. That is done to deal with validators that might have some clock skew. This is called the inception offset and it should be chosen so that false negatives are minimized to a reasonable level.

4.4.2.3. Differentiation between RRsets

It is possible to vary signature validity periods between signatures over different RRsets in the zone. In practice, this could be done when zones contain highly volatile data (which may be the case in Dynamic Update environments). Note however that the risk of replay (e.g., by stale secondary servers) is what should be leading in determining the signature validity period, since the TTLs on the data itself are still the primary parameter for cache expiry.

In some cases, the risk of replaying existing data might be different from the risk of replaying the denial of data. In those cases the signature validity period on NSEC or NSEC3 records may be tweaked accordingly.

When a zone contains secure delegations, then a relatively short

signature validity period protects the child against replay attacks in the case the child's key is compromised (see Section 4.3.4). Since there is a higher operational risk for the parent registry when choosing a short validity period and a higher operational risk for the child when choosing a long validity period, some (price) differentiation may occur for validity periods between individual DS RRs in a single zone.

There seem to be no other arguments for differentiation in validity periods.

5. Next Record type

One of the design tradeoffs made during the development of DNSSEC was to separate the signing and serving operations instead of performing cryptographic operations as DNS requests are being serviced. It is therefore necessary to create records that cover the very large number of non-existent names that lie between the names that do exist.

There are two mechanisms to provide authenticated proof of non-existence of domain names in DNSSEC: a clear-text one and an obfuscated-data one. Each mechanism:

- o includes a list of all the RRTYPEs present, which can be used to prove the non-existence of RRTYPEs at a certain name;
- o stores only the name for which the zone is authoritative (that is, glue in the zone is omitted); and
- o uses a specific RRTYPE to store information about the RRTYPEs present at the name: the clear-text mechanism uses NSEC, and the obfuscated-data mechanism uses NSEC3.

5.1. Differences between NSEC and NSEC3

The clear text mechanism (NSEC) is implemented using a sorted linked list of names in the zone. The obfuscated-data mechanism (NSEC3) is similar but first hashes the names using a one-way hash function, before creating a sorted linked list of the resulting (hashed) strings.

The NSEC record requires no cryptographic operations aside from the validation of its associated signature record. It is human readable and can be used in manual queries to determine correct operation. The disadvantage is that it allows for "zone walking", where one can request all the entries of a zone by following the linked list of NSEC RRs via the "Next Domain Name" field. Though all agree DNS data is accessible through query mechanisms, for some zone administrators this behavior is undesirable for policy, regulatory or other reasons.

Furthermore, NSEC requires a signature over every RR in the zone file, thereby ensuring that any denial of existence is cryptographically signed. However, in a large zone file containing many delegations, very few of which are to signed zones, this may produce unacceptable additional overhead, especially where insecure delegations are subject to frequent update (a typical example might be a TLD operator with few registrants using secure delegations). NSEC3 allows intervals between two secure delegations to "Opt-out" in

which case they may contain one or more insecure delegations, thus reducing the size and cryptographic complexity of the zone at the expense of the ability to cryptographically deny the existence of names in a specific span.

The NSEC3 record uses a hashing method of the requested name. To increase the workload required to guess entries in the zone, the number of hashing iterations can be specified in the NSEC3 record. Additionally, a salt can be specified that also modifies the hashes. Note that NSEC3 does not give full protection against information leakage from the zone (you can still derive the size of the zone, which RRtypes are in there, ...).

5.2. NSEC or NSEC3

The first motivation to deploy NSEC3, prevention of zone enumeration, only makes sense when zone content is not highly structured or trivially guessable. Highly structured zones such as `in-addr.arpa.`, `ip6.arpa.` and `el64.arpa.` can be trivially enumerated using ordinary DNS properties, while for small zones that only contain records in the APEX and a few common names such as "www" or "mail", guessing zone content and proving completeness is also trivial when using NSEC3. In these cases, the use of NSEC is preferred to ease the work required by signers and validating resolvers.

For large zones where there is an implication of "not readily available" names, such as those where one has to sign a non-disclosure agreement before obtaining it, NSEC3 is preferred. The second reason to consider NSEC3 is opt-out, which can reduce the number of NSEC3 records required. This is discussed further below (Section 5.3.4).

5.3. NSEC3 parameters

NSEC3 is controlled by a number of parameters, some of which can be varied: this section discusses the choice of those parameters.

5.3.1. NSEC3 Algorithm

The NSEC3 hashing algorithm is performed on the Fully Qualified Domain Name (FQDN) in its uncompressed form. This ensures brute force work done by an attacker for one FQDN cannot be re-used for another FQDN attack, as these entries are, by definition unique.

At the time of writing, there is only one NSEC3 hash algorithm defined. [RFC5155] specifically states: "When a new hash algorithm for use with NSEC3 is specified, a transition mechanism MUST also be defined." Therefore this document does not consider NSEC3 hash

algorithm transition.

5.3.2. NSEC3 Iterations

One of the concerns with NSEC3 is that a pre-calculated dictionary attack could be performed in order to assess if certain domain names exist within a zone or not. Two mechanisms are introduced in the NSEC3 specification to increase the costs of such dictionary attacks: iterations and salt.

Iterations define the number of additional times the hash function has been performed. A higher value results in greater resiliency against dictionary attacks, at a higher computational cost for both the server and resolver.

RFC 5155 Section 10.3 [RFC5155] considers the trade-offs between incurring cost during the signing process and imposing costs to the validating name server, while still providing a reasonable barrier against dictionary attacks. It provides useful limits of iterations for a given RSA key size. These are 150 iterations for 1024 bit keys, 500 iterations for 2048 bit keys and 2,500 iterations for 4096 bit keys. Choosing a value of 100 iterations is deemed to be a sufficiently costly yet not excessive value: In the worst case scenario, the performance of name servers would be halved, regardless of key size [nsec3-hash-performance].

5.3.3. NSEC3 Salt

While the NSEC3 iterations parameter increases the cost of hashing a dictionary word, the NSEC3 salt reduces the lifetime for which that calculated hash can be used. A change of the salt value by the zone administrator would cause an attacker to lose all pre-calculated work for that zone.

There must be a complete NSEC3 chain using the same salt value, that matches the salt value in the NSEC3PARAM record. NSEC3 salt changes do not need special rollover procedures. Since changing the salt requires all the NSEC3 records to be regenerated, and thus requires generating new RRSIG's over these NSEC3 records, it makes sense to align the change of the salt with a change of the Zone Signing Key, as that process in itself already usually requires all RRSIG's to be regenerated. If there is no critical dependency on incremental signing and the zone can be signed with little effort, there is no need for such alignment.

5.3.4. Opt-Out

The Opt-Out mechanism was introduced to allow for a gradual introduction of signed records in zones that contain mostly delegation records. The use of the Opt-Out flag changes the meaning of the NSEC3 span from authoritative denial of the existence of names within the span to a proof that DNSSEC is not available for the delegations within the span. This allows for the addition or removal of the delegations covered by the span without recalculating or re-signing RRs in the NSEC3 RR chain.

Opt-Out is specified to be used only over delegation points and will therefore only bring relief to zones with a large number of insecure delegations. This consideration typically holds for large top-level-domains and similar zones; in most other circumstances, Opt-Out should not be deployed. Further considerations can be found in Section 12.2 of RFC 5155 [RFC5155].

6. Security Considerations

DNSSEC adds data origin authentication and data integrity to the DNS, using digital signatures over resource record sets. DNSSEC does not protect against denial of service attacks, nor does it provide confidentiality. For more general security considerations related to DNSSEC, please see RFC 4033 [RFC4033], RFC 4034 [RFC4034] and RFC 4035 [RFC4035].

This document tries to assess the operational considerations to maintain a stable and secure DNSSEC service. When performing key rollovers, it is important to keep in mind that it takes time for the data to be propagated to the verifying clients. Also important to notice is that this data may be cached. Not taking into account the 'data propagation' properties in the DNS may cause validation failures, because cached data may mismatch with data fetched from the authoritative servers. This will make secured zones unavailable to security-aware resolvers.

7. IANA considerations

There are no IANA considerations with respect to this document

8. Contributors and Acknowledgments

Significant parts of the text of this document is copied from RFC 4641 [RFC4641]. That document was edited by Olaf Kolkman and Miek Gieben. Other people that contributed or where otherwise involved in that work were in random order: Rip Loomis, Olafur Gudmundsson, Wesley Griffin, Michael Richardson, Scott Rose, Rick van Rein, Tim McGinnis, Gilles Guette, Olivier Courtay, Sam Weiler, Jelte Jansen, Niall O'Reilly, Holger Zuleger, Ed Lewis, Hilarie Orman, Marcos Sanz, Peter Koch, Mike StJohns, Emma Bretherick, Adrian Bedford, Lindy Foster, and O. Courtay.

For this version of the document we would like to acknowledge a few people for significant contributions:

Paul Hoffman for his contribution on the choice of cryptographic parameters and addressing some of the trust anchor issues;

Jelte Jansen who provided the initial text in Section 4.1.4;

Paul Wouters who provided the initial text for Section 5 and Alex Bligh who improved it;

Erik Rescorla whose blogpost on "the Security of ZSK rollovers" inspired text in Section 3.1;

Stephen Morris who made a pass on English style and grammar;

Olafur Gudmundsson and Ondrej Sury who provided input on Section 4.1.4 based on actual operational experience.

Rickard Bellgrim reviewed the document extensively.

The figure in Section 4.4.2 was adapted from the OpenDNSSEC user documentation.

In addition valuable contributions in the form of text, comments, or review where provided by Mark Andrews, Patrik Faltstrom, Tony Finch, Alfred Hoenes, Bill Manning, Scott Rose, Wouter Wijngaards, Antoin Verschuren, Marc Lampo, George Barwood, Sebastian Castro and Suresh Krishnaswamy.

9. References

9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [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.
- [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, May 2006.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.
- [RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5702, October 2009.

9.2. Informative References

- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, August 1996.
- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.
- [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.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic

Update", RFC 3007, November 2000.

- [RFC3375] Hollenbeck, S., "Generic Registry-Registrar Protocol Requirements", RFC 3375, September 2002.
- [RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, September 2006.
- [RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC 5011, September 2007.
- [RFC5910] Gould, J. and S. Hollenbeck, "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 5910, May 2010.
- [RFC5933] Dolmatov, V., Chuprina, A., and I. Ustinov, "Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC", RFC 5933, July 2010.
- [RFC6605] Hoffman, P. and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, April 2012.
- [NIST-workshop] Rose, S., "NIST DNSSEC workshop notes", July 2001, <<http://www.ietf.org/mail-archive/web/dnsop/current/msg01020.html>>.
- [NIST-SP-800-90A] Barker, E. and J. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)", NIST Special Publication 800-90, March 2007.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [I-D.ietf-dnsop-dnssec-key-timing]

Morris, S., Ihren, J., and J. Dickinson, "DNSSEC Key Timing Considerations",
draft-ietf-dnsop-dnssec-key-timing-03 (work in progress),
July 2012.

[I-D.ietf-dnsop-dnssec-dps-framework]
Ljunggren, F., Eklund-Lowinder, A., and T. Okubo, "A Framework for DNSSEC Policies and DNSSEC Practice Statements", draft-ietf-dnsop-dnssec-dps-framework-08 (work in progress), June 2012.

[I-D.ietf-dnsop-dnssec-trust-anchor]
Larson, M. and O. Gudmundsson, "DNSSEC Trust Anchor Configuration and Maintenance",
draft-ietf-dnsop-dnssec-trust-anchor-04 (work in progress), October 2010.

[nsec3-hash-performance]
Schaeffer, Y., "NSEC3 Hash Performance", NLnet Labs document 2010-02, March 2010.

Appendix A. Terminology

In this document, there is some jargon used that is defined in other documents. In most cases, we have not copied the text from the documents defining the terms but have given a more elaborate explanation of the meaning. Note that these explanations should not be seen as authoritative.

Anchored key: A DNSKEY configured in resolvers around the globe. This key is hard to update, hence the term anchored.

Bogus: Also see Section 5 of RFC 4033 [RFC4033]. An RRset in DNSSEC is marked "Bogus" when a signature of an RRset does not validate against a DNSKEY.

Key rollover: A key rollover (also called key supercession in some environments) is the act of replacing one key pair with another at the end of a key effectivity period.

Key Signing Key or KSK: A Key Signing Key (KSK) is a key that is used exclusively for signing the apex key set. The fact that a key is a KSK is only relevant to the signing tool.

Key size: The term 'key size' can be substituted by 'modulus size' throughout the document for RSA keys. It is mathematically more correct to use modulus size for RSA keys, but as this is a document directed at operators we feel more at ease with the term key size.

Private and public keys: DNSSEC secures the DNS through the use of public key cryptography. Public key cryptography is based on the existence of two (mathematically related) keys, a public key and a private key. The public keys are published in the DNS by use of the DNSKEY Resource Record (DNSKEY RR). Private keys should remain private.

Refresh Period: The period before the expiration time of the signature, during which the signature is refreshed by the signer.

Re-Sign Period: This refers to the frequency with which a signing pass on the zone is performed. The Re-Sign Period defines when the zone is exposed to the signer. And on the signer - not all signatures in the zone have to be regenerated: that depends on the Refresh Period.

Secure Entry Point (SEP) key: A KSK that has a DS record in the parent zone pointing to it or is configured as a trust anchor. Although not required by the protocol, we suggest that the SEP flag [RFC4034] is set on these keys.

Self-signature: This only applies to signatures over DNSKEYs; a signature made with DNSKEY x over DNSKEY x is called a self-signature. Note: without further information, self-signatures convey no trust. They are useful to check the authenticity of the DNSKEY, i.e., they can be used as a hash.

Signing Jitter: A random variation in the signature validity period of RRSIGs in a zone to prevent all of them expiring at the same time.

Signer: The system that has access to the private key material and signs the Resource Record sets in a zone. A signer may be configured to sign only parts of the zone, e.g., only those RRsets for which existing signatures are about to expire.

Singing the zone file: The term used for the event where an administrator joyfully signs its zone file while producing melodic sound patterns.

Single Type Signing Scheme: A signing scheme whereby the distinction between Zone Signing Keys and Key Signing Keys is not made.

Zone administrator: The 'role' that is responsible for signing a zone and publishing it on the primary authoritative server.

Zone Signing Key (ZSK): A key that is used for signing all data in a zone (except, perhaps, the DNSKEY RRset). The fact that a key is a ZSK is only relevant to the signing tool.

Appendix B. Typographic Conventions

The following typographic conventions are used in this document:

Key notation: A key is denoted by `DNSKEY_x_y`, where `x` is an identifier for the type of key: `K` for Keys Signing Key, `Z` for Zone Signing Key and `S` when there is no distinction made between KSK and ZSKs but the key is used as a secure entry point. The '`y`' denotes a number or an identifier, `y` could be thought of as the key id.

RRsets ignored: If the signatures of non-DNSKEY RRsets have the same parameters as the SOA, then those are not mentioned; e.g., in the example below, the SOA is signed with the same parameters as the `foo.example.com` A RRset and the latter is therefore ignored in the abbreviated notation.

RRset notations: RRs are only denoted by the type. All other information -- owner, class, rdata, and TTL -- is left out. Thus: "`example.com 3600 IN A 192.0.2.1`" is reduced to "`A`". RRsets are a list of RRs. A example of this would be "`A1, A2`", specifying the RRset containing two "`A`" records. This could again be abbreviated to just "`A`".

Signature notation: Signatures are denoted as `RRSIG_x_y(type)`, which means that the RRset with the specific RRtype '`type`' is signed with `DNSKEY_x_y`. Signatures in the parent zone are denoted as `RRSIG_par(type)`.

SOA representation: SOAs are represented as `SOA_x`, where `x` is the serial number.

DS representation: DSs are represented as `DS_x_y`, where `x` and `y` are identifiers similar to the key notation: `x` is an an identifier for the type of key the DS record refers to, `y` is the 'key id' of the key it refers to.

Zone representation: Using the above notation we have simplified the representation of a signed zone by leaving out all unnecessary details such as the names and by representing all data by "`SOA_x`"

Using this notation the following signed zone:

```
example.com. 3600 IN SOA  ns1.example.com. olaf.example.net. (  
                        2005092303 ; serial  
                        450      ; refresh (7 minutes 30 seconds)  
                        600      ; retry (10 minutes)  
                        345600   ; expire (4 days)
```



```

                                300           ; minimum (5 minutes)
                                )
3600    RRSIG    SOA 5 2 3600 20120824013000 (
                                20100424013000 14 example.com.
                                NMaFnZmmZ8wevpCOI+/JxqWBzPxrnzPnSXfo
                                ...
                                OMY3rTMA2qorupQXjQ== )
3600    NS       ns1.example.com.
3600    NS       ns2.example.com.
3600    NS       ns3.example.com.
3600    RRSIG    NS 5 2 3600 20120824013000 (
                                20100424013000 14 example.com.
                                p0Cj3wzGoPFftFZjj3jeKGK6wGWLwY6mCBEz
                                ...
                                +SqZIoVHpVE7YBeH46wuyF8w4XknA40eimc4
                                zAgaJM/MeG08KpeHhg== )
3600    TXT      "Net::DNS domain"
3600    RRSIG    TXT 5 2 3600 20120824013000 (
                                20100424013000 14 example.com.
                                o7eP8LISK2TEutFQRvK/+U3wq7t4X+PQaQkP
                                ...
                                BcQlo99vwn+IS4+Jlg== )
300     NSEC      foo.example.com. NS SOA TXT RRSIG NSEC DNSKEY
300     RRSIG    NSEC 5 2 300 20120824013000 (
                                20100424013000 14 example.com.
                                JtHm8ta0diCWYGu/TdrE10lsYSHb1N2i/IX+
                                ...
                                PkXNI/Vgf4t3xZaIyw== )
3600    DNSKEY   256 3 5 (
                                AQPaoHW/nC0fj9HuCW3hACSGiP0AkPS3dQFX
                                ...
                                sAuryjQ/HFa5r4mrbhkJ
                                ) ; key id = 14
3600    DNSKEY   257 3 5 (
                                AQPUIszMMAi36agx/V+7Tw95l8PYmoVjHWvO
                                ...
                                oy88Nh+u2c9HF1tw0naH
                                ) ; key id = 15
3600    RRSIG    DNSKEY 5 2 3600 20120824013000 (
                                20100424013000 14 example.com.
                                HWj/VEr6p/FiUUil70QQWtk+NBILsJ9mdj5U
                                ...
                                QhhmMwV3tIxJk2eDRQ== )
3600    RRSIG    DNSKEY 5 2 3600 20120824013000 (
                                20100424013000 15 example.com.
                                P47CUy/xPV8qIEuaa4tMKG6ei3LQ8RYv3TwE
                                ...
                                JWl70YiUnUG3m9OL9w== )

```

```
foo.example.com. 3600 IN A 192.0.2.2
3600        RRSIG    A 5 3 3600 20120824013000 (
                 20100424013000 14 example.com.
                 xHr023P79YrSHHmtSL0alnlfUt4ywn/vWqsO
                 ...
                 JPV/SA4BkoFxiCPrDQ== )
300        NSEC       example.com. A RRSIG NSEC
300        RRSIG    NSEC 5 3 300 20120824013000 (
                 20100424013000 14 example.com.
                 Aaa4kgKhqY7Lzjq3rlPlFidymOeBEK1T6vUF
                 ...
                 Qe000JyzObxx27pY8A== )
```

is reduced to the following representation:

```
SOA_2005092303
RRSIG_Z_14(SOA_2005092303)
DNSKEY_K_14
DNSKEY_Z_15
RRSIG_K_14(DNSKEY)
RRSIG_Z_15(DNSKEY)
```

The rest of the zone data has the same signature as the SOA record, i.e., an RRSIG created with DNSKEY 14.

Appendix C. Transition Figures for Special Case Algorithm Rollovers

The figures in this Appendix complement and illustrate the special cases of algorithm rollovers as described in Section 4.1.4.

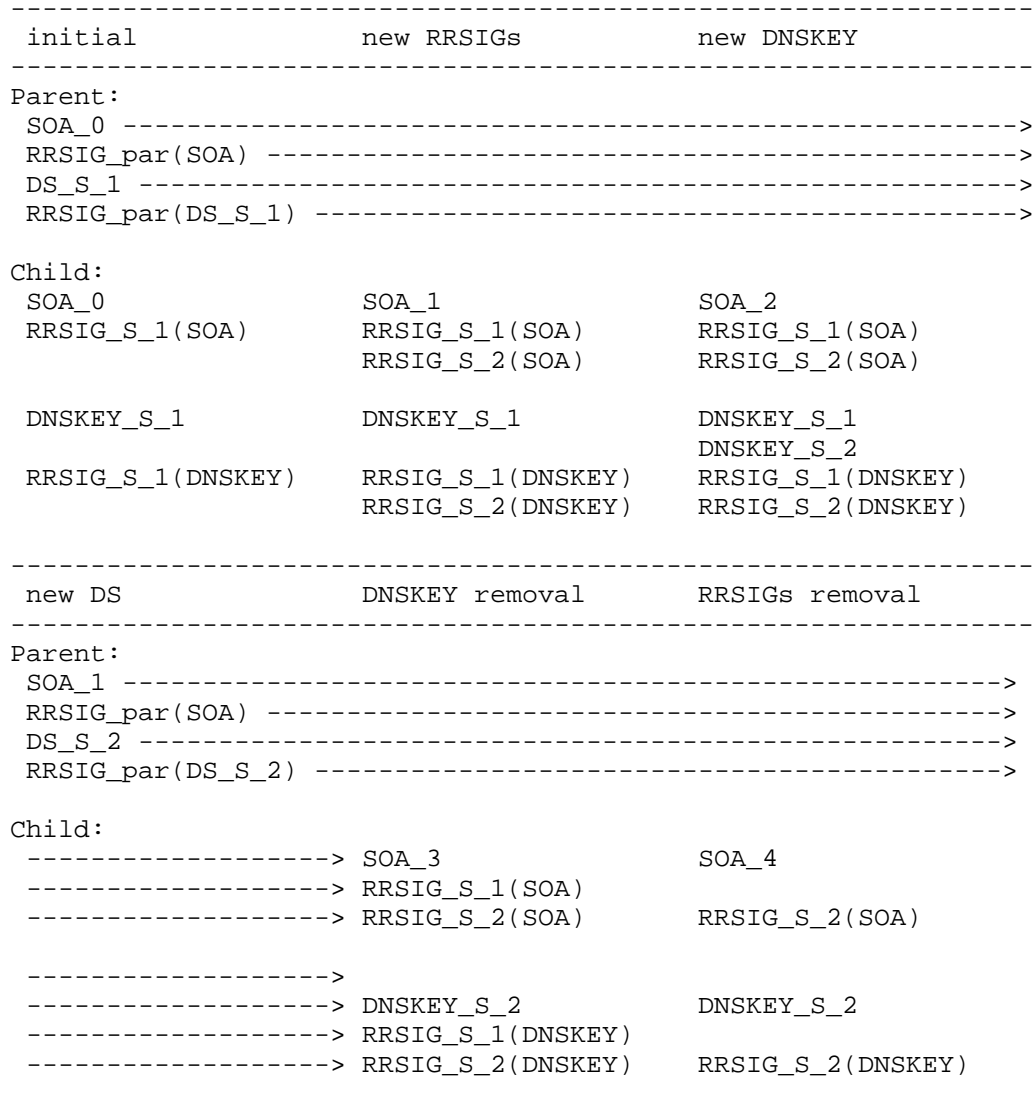


Figure 12: Single Type Signing Scheme Algorithm Roll

Also see Section 4.1.4.1.

```

-----
initial                new RRSIGs                new DNSKEY
-----

Parent:
SOA_0 ----->
RRSIG_par(SOA) ----->
DS_K_1 ----->
RRSIG_par(DS_K_1) ----->

Child:
SOA_0                SOA_1                SOA_2
RRSIG_Z_1(SOA)       RRSIG_Z_1(SOA)       RRSIG_Z_1(SOA)
                    RRSIG_Z_2(SOA)       RRSIG_Z_2(SOA)

DNSKEY_K_1           DNSKEY_K_1           DNSKEY_K_1
                    DNSKEY_K_2
DNSKEY_Z_1           DNSKEY_Z_1           DNSKEY_Z_1
                    DNSKEY_Z_2
RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)    RRSIG_K_1(DNSKEY)
                    RRSIG_K_2(DNSKEY)

-----

new DS                revoke DNSKEY                DNSKEY removal
-----

Parent:
SOA_1 ----->
RRSIG_par(SOA) ----->
DS_K_2 ----->
RRSIG_par(DS_K_2) ----->

Child:
-----> SOA_3                SOA_4
-----> RRSIG_Z_1(SOA)       RRSIG_Z_1(SOA)
-----> RRSIG_Z_2(SOA)       RRSIG_Z_2(SOA)

-----> DNSKEY_K_1_REVOKED
-----> DNSKEY_K_2                DNSKEY_K_2
----->
-----> DNSKEY_Z_2                DNSKEY_Z_2
-----> RRSIG_K_1(DNSKEY)
-----> RRSIG_K_2(DNSKEY)       RRSIG_K_2(DNSKEY)

-----

RRSIGs removal
-----

Parent:
----->
----->

```

```
----->
----->

Child:
  SOA_5
  RRSIG_Z_2(SOA)

  DNSKEY_K_2

  DNSKEY_Z_2

  RRSIG_K_2(DNSKEY)
-----
```

Figure 13: RFC 5011 Style algorithm roll

Also see Section 4.1.4.2.

```

-----
initial                new RRSIGs                new DNSKEY
-----
Parent:
SOA_0 ----->
RRSIG_par(SOA) ----->
DS_S_1 ----->
RRSIG_par(DS_S_1) ----->

Child:
SOA_0                SOA_1                SOA_2
RRSIG_S_1(SOA)
RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)      RRSIG_Z_10(SOA)
                     RRSIG_S_2(SOA)      RRSIG_S_2(SOA)

DNSKEY_S_1           DNSKEY_S_1           DNSKEY_S_1
DNSKEY_Z_10          DNSKEY_Z_10          DNSKEY_Z_10
                     DNSKEY_S_2
RRSIG_S_1(DNSKEY)    RRSIG_S_1(DNSKEY)    RRSIG_S_1(DNSKEY)
                     RRSIG_S_2(DNSKEY)    RRSIG_S_2(DNSKEY)

-----
new DS                revoke DNSKEY                DNSKEY removal
-----
Parent:
SOA_1 ----->
RRSIG_par(SOA) ----->
DS_S_2 ----->
RRSIG_par(DS_S_2) ----->

Child:
-----> SOA_3                SOA_4

-----> RRSIG_Z_10(SOA)
-----> RRSIG_S_2(SOA)      RRSIG_S_2(SOA)

-----> DNSKEY_S_1_REVOKED
-----> DNSKEY_Z_10
-----> DNSKEY_S_2                DNSKEY_S_2
-----> RRSIG_S_1(DNSKEY)      RRSIG_S_1(DNSKEY)
-----> RRSIG_S_2(DNSKEY)      RRSIG_S_2(DNSKEY)

-----
RRSIGs removal
-----
Parent:
----->
----->

```

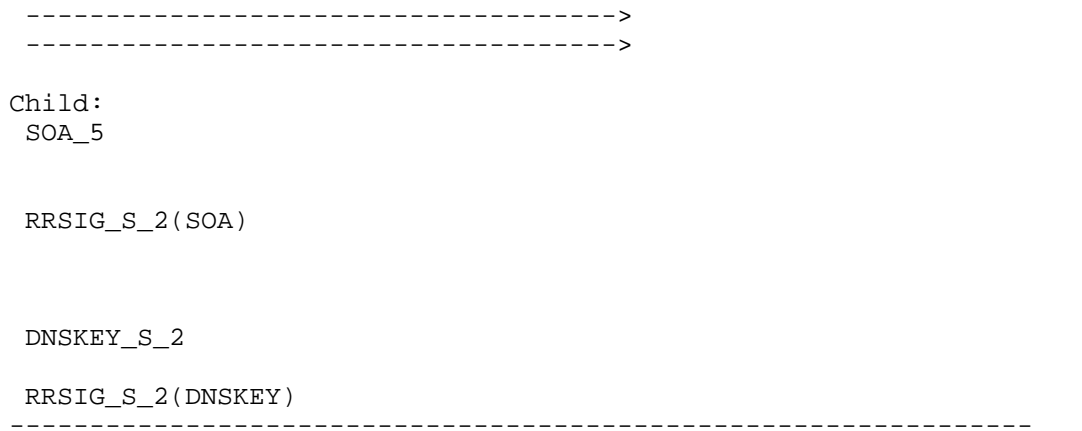


Figure 14: RFC 5011 algorithm roll in a Single Type Signing Scheme Environment

Also see Section 4.1.4.3.

Appendix D. Transition Figure for Changing DNS Operators

The figure in this Appendix complements and illustrates the special case of changing DNS operators as described in Section 4.3.5.1.

new DS		pre-publish

Parent:		
NS_A		NS_A
DS_A DS_B		DS_A DS_B

Child at A:	Child at A:	Child at B:
SOA_A0	SOA_A1	SOA_B0
RRSIG_Z_A(SOA)	RRSIG_Z_A(SOA)	RRSIG_Z_B(SOA)
NS_A	NS_A	NS_B
RRSIG_Z_A(NS)	NS_B	RRSIG_Z_B(NS)
	RRSIG_Z_A(NS)	
DNSKEY_Z_A	DNSKEY_Z_A	DNSKEY_Z_A
	DNSKEY_Z_B	DNSKEY_Z_B
DNSKEY_K_A	DNSKEY_K_A	DNSKEY_K_B
RRSIG_K_A(DNSKEY)	RRSIG_K_A(DNSKEY)	RRSIG_K_A(DNSKEY)
	RRSIG_K_B(DNSKEY)	RRSIG_K_B(DNSKEY)

redelegation		post migration

Parent:		
	NS_B	NS_B
	DS_A DS_B	DS_B

Child at A:	Child at B:	Child at B:
SOA_A1	SOA_B0	SOA_B1
RRSIG_Z_A(SOA)	RRSIG_Z_B(SOA)	RRSIG_Z_B(SOA)
NS_A	NS_B	NS_B
NS_B	RRSIG_Z_B(NS)	RRSIG_Z_B(NS)
RRSIG_Z_A(NS)		
DNSKEY_Z_A	DNSKEY_Z_A	
DNSKEY_Z_B	DNSKEY_Z_B	DNSKEY_Z_B
DNSKEY_K_A	DNSKEY_K_B	DNSKEY_K_B
RRSIG_K_A(DNSKEY)	RRSIG_K_B(DNSKEY)	RRSIG_K_B(DNSKEY)

Figure 15: An alternative rollover approach for cooperating operators

Appendix E. Summary of Changes from RFC 4641

This document differs from RFC 4641 [RFC4641] in the following ways:

- o Applied the errata from
 http://www.rfc-editor.org/errata_search.php?rfc=4641
- o RSA/SHA256 is being recommended in addition to RSA/SHA1.
- o Complete rewrite of Section 3.4.2 removing the table and suggesting a keysize of 1024 for keys in use for less than 8 years, issued up to at least 2015.
- o Removed the KSK for high level zones consideration.
- o Added text on Algorithm Rollover.
- o Added text on Changing (non cooperating) DNS Registrars.
- o Significant rewrite of Section 3 whereby the argument is made that the timescales for rollovers are made purely on operational arguments.
- o Added Section 5.
- o Introduced Single Type Signing Scheme terminology and made the arguments for the choice of a Single Type Signing Scheme more explicit.
- o Added a section about Stand-by keys

Appendix F. Document Editing History

[To be removed prior to publication as an RFC]

F.1. draft-ietf-dnsop-rfc4641-00

Version 0 was differs from RFC 4641 in the following ways.

- o Status of this memo appropriate for I-D
- o TOC formatting differs.
- o Whitespaces, linebreaks, and pagebreaks may be slightly different because of xml2rfc generation.
- o References slightly reordered.
- o Applied the errata from
http://www.rfc-editor.org/errata_search.php?rfc=4641
- o Inserted trivial "IANA considerations" section.

In other words it should not contain substantive changes in content as intended by the working group for the original RFC 4641.

F.2. version 0->1

Cryptography details rewritten. (See http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/cryptography_flawed)

- o Reference to NIST 800-90 added
- o RSA/SHA256 is being recommended in addition to RSA/SHA1.
- o Complete rewrite of Section 3.4.2 removing the table and suggesting a keysize of 1024 for keys in use for less than 8 years, issued up to at least 2015.
- o Replaced the reference to Schneiers' applied cryptography with a reference to RFC 4949.
- o Removed the KSK for high level zones consideration

Applied some differentiation with respect of the use of a KSK for parent or trust anchor relation http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/differentiation_trustanchor_parent

<http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/>

rollover_assumptions

Added Section 4.1.4 as suggested by Jelte Jansen in http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/Key_algorithm_roll

Added Section 4.3.5.2 Issue identified by Antoin Verschuren <http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/non-cooperative-registrars>

In Appendix A: ZSK does not necessarily sign the DNSKEY RRset.

F.3. version 1->2

- o Significant rewrite of Section 3 whereby the argument is made that the timescales for rollovers are made purely on operational arguments hopefully resolving http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/discussion_of_timescales
- o Added Section 5 based on <http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/NSEC-NSEC3>
- o Added a reference to draft-morris-dnsop-dnssec-key-timing [I-D.ietf-dnsop-dnssec-key-timing] for the quantitative analysis on keyrolls
- o Updated Section 4.3.5 to reflect that the problem occurs when changing DNS operators, and not DNS registrars, also added the table indicating the redelegation procedure. Added text about the fact that implementations will dismiss keys that fail to validate at some point.
- o Updated a number of references.

F.4. version 2->3

- o Added bulleted list to serve as an introduction on the decision tree in Section 3.
- o In section Section 3.1:
 - * tried to motivate that key length is not a strong motivation for KSK ZSK split (based on http://www.educatedguesswork.org/2009/10/on_the_security_of_zsk_rollove.html)
 - * Introduced Common Signing Key terminology and made the arguments for the choice of a Common Signing Key more explicit.

* Moved the SEP flag considerations to its own paragraph

- o In a few places in the document, but section Section 4 in particular the comments from Patrik Faltstrom (On Mar 24, 2010) on the clarity on the roles of the registrant, dns operator, registrar and registry was addressed.
- o Added some terms based on http://www.nlnetlabs.nl/svn/rfc4641bis/trunk/open-issues/timing_terminology
- o Added paragraph 2 and clarified the second but last paragraph of Section 3.2.2.
- o Clarified the table and some text in Section 4.1.4. Also added some text on what happens when the algorithm rollover also involves a roll from NSEC to NSEC3.
- o Added a paragraph about rolling KSKs that are also configured as trust anchors in Section 4.1.2
- o Added Section 4.1.3.
- o Added Section 4.4.2 to address issue "Signature_validity"

F.5. version 3->4

- o Stephen Morris submitted a large number of language, style and editorial nits.
- o Section 4.1.4 improved based on comments from Olafur Gudmundsson and Ondrej Sury.
- o Tried to improve consistency of notation in the various rollover figures

F.6. version 4->5

- o Improved consistency of notation
- o Matthijs Mekking provided substantive feedback on algorithm rollover and suggested the content of the subsections of Section 4.1.4 and the content of the figures in Appendix C

F.7. version 5->6

- o More improved consistency of notation and some other nits

- o Review of Rickard Bellgrim
- o Review of Sebastian Castro
- o Added a section about Stand-by keys
- o Algorithm rollover: Conservative or Liberal Approach
- o Added a reference to NSEC3 hash performance report
- o More clarifications on the topic of non cooperating operators

F.8. version 6->7

- o Fixed minor nits.
- o Clarified the Double DS Rollover in Changing DNS Operator sections.
- o Adjusted STSS Rollover Figures.
- o Remove the ZSK RRSIGs over DNSKEY RRset in Figures.
- o Added text: second variety on STSS Double DS Rollover.
- o Reviewed by Antoin Verschuren, Marc Lampo, George Barwood.

F.9. version 7->8

- o Signatures over DNSKEY RRset does not need to be propagated in the new RRSIGS step.

F.10. version 8->9

- o Peter Koch and Stephen Morris review
- o Editorial changes
- o Added Appendix D for clarifying the alternative approach on rollover for cooperating operators.
- o Added a paragraph to explain the rollover described in the figure in a bit more detail, in Section 4.3.5.

F.11. version 9->10

- o Final nits
- o Symbolic references

F.12. version 10->11

- o More review (Alfred Hoenes, Marc Lampo, Miek Gieben, Stephen Morris)
- o Style, language and layout changes to improve consistency and resolving ambiguity and removing duplicate sentences.
- o More clarifications
- o Small restructuring of paragraphs that fit better in other sections

F.13. version 11->12

- o WG chairs review

F.14. version 12->13

- o Fixed editors name (Miek Gieben)
- o Review Yuri Schaeffer, Ed Lewis
- o IESG, Gen-ART, secdir reviews

F.15. Subversion information

www.nlnetlabs.nl/svn/rfc4641bis/

\$Id: draft-ietf-dnsop-rfc4641bis.xml 127 2012-08-30 10:02:15Z matje \$

Authors' Addresses

Olaf M. Kolkman
NLnet Labs
Science Park 400
Amsterdam 1098 XH
The Netherlands

Email: olaf@nlnetlabs.nl
URI: <http://www.nlnetlabs.nl>

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

Email: matthijs@nlnetlabs.nl
URI: <http://www.nlnetlabs.nl>

R. (Miek) Gieben
SIDN Labs
Meander 501
Arnhem 6825 MD
The Netherlands

Email: miek.gieben@sidn.nl
URI: <http://www.sidn.nl>

Network Working Group
Internet-Draft
Updates: 6304 (if approved)
Intended status: BCP
Expires: December 1, 2012

W. Kumari
Google
W. Sotomayor
NRC-CNRC
J. Abley
ICANN
May 30, 2012

Omniscient AS112 Servers
draft-wkumari-dnsop-omniscient-as112-00

Abstract

The AS112 Project loosely coordinates Domain Name System (DNS) servers to which DNS zones corresponding to private use addresses are delegated. Queries for names within those zones have no useful responses in a global context. The purpose of the project is to reduce the load of such junk queries on the authoritative name servers that would otherwise receive them, directing the load instead to name servers operated within the AS112 project.

Adding and dropping zones from the AS112 servers is difficult, due to the loosely-coordinated nature of the project. This document proposes a mechanism by which AS112 name servers could answer authoritatively for all possible zones, reducing the add/drop problem to one of delegation within the DNS without operational impact on the servers themselves.

This document updates RFC 6304.

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 December 1, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Protocol Considerations	4
4. Operational Considerations	4
5. Addressing Considerations	5
6. Updates to RFC 6304	5
6.1. Changes to Section 3.4, Routing Software	5
6.2. Changes to Section 3.5, DNS Software	7
6.3. Changes to Section 3.6, Testing a Newly Installed Node	9
7. IANA Considerations	9
8. Security Considerations	10
9. Acknowledgements	10
10. References	10
10.1. Normative References	10
10.2. Informative References	10
Appendix A. Document Notes	11
A.1. Venue	11
A.2. Textual Substitutions	11
A.3. Open Questions	11
A.4. Change History	11
A.4.1. draft-wkumari-dnsop-omniscient-as112-00	11
A.4.2. draft-wkumari-omniscient-as112-00	12
Authors' Addresses	12

1. Introduction

The AS112 Project loosely coordinates Domain Name System (DNS) servers [RFC1034] to which DNS zones corresponding to private use addresses are delegated. Queries for names within those zones have no useful responses in a global context. The purpose of the project is to reduce the load of such junk queries on the authoritative name servers that would otherwise receive them, directing the load instead to name servers operated within the AS112 project.

To date, AS112 nameservers have been used exclusively for names corresponding to the reverse mapping for private-use IPv4 addresses. A description of current advice for AS112 operators, including motivations and guidance for technical deployment and operations can be found in [RFC6304].

Other DNS domains have analogously local significance. Examples corresponding to the reverse-mapping of special-use IPv4 and IPv6 addresses can be found in [RFC6303].

It is to be expected that new domains will be identified from time to time that fit the use pattern for which delegation to AS112 servers might be desirable. There is currently no mechanism by which particular zones can be reliably added to or dropped from AS112 servers, however. This is principally a consequence of the loosely-coordinated nature of the project, coupled with a desire to avoid lame delegations which might have unforeseen operational consequences.

This document proposes a mechanism by which AS112 servers could provide consistent, reliable negative responses for all DNS queries, eliminating the operational requirement to add or drop particular zones from all AS112 servers.

2. Terminology

An "Existing AS112 Server" is a DNS name server configured according to the guidance provided in [RFC6304] and listening on the IPv4 addresses 192.175.48.1 (PRISONER.IANA.ORG), 192.175.48.6 (BLACKHOLE-1.IANA.ORG) and 192.175.48.42 (BLACKHOLE-2.IANA.ORG).

An "Omniscient AS112 Server" is a DNS nameserver configured according to the guidance provided in [RFC6304], as extended by this document. Such servers listen on the same addresses as Existing AS112 Servers, but also additional addresses as described in Section 5.

Where discussions apply equally to Existing AS112 Servers and Omniscient AS112 Servers, the unqualified phrase "AS112 Server" is

used.

An "AS112 Zone" is a DNS zone which has been delegated to an AS112 Server.

An "Existing AS112 Zone" is an AS112 Zone which has been delegated to an existing AS112 Server.

3. Protocol Considerations

An AS112 Server responds with an authoritative (AA=1) name error (NXDOMAIN, RCODE=3) for any query request whose (QNAME, QCLASS) falls within an AS112 Zone [RFC1035].

AS112 Servers do not respond to zone transfer requests (QTYPE=252).

The name error (NXDOMAIN) response from an Omniscient AS112 Server differs from that sent by an Existing AS112 Server in that the closest enclosing SOA returned has a different owner name. Existing AS112 Servers return an authority-section SOA with an owner name corresponding to the apex of the AS112 Zone concerned; Omniscient AS112 Servers return an SOA with an owner name of ".". This difference has not been shown to cause any practical change in behaviour in commonly-deployed DNS resolver software.

4. Operational Considerations

Existing AS112 Servers address the protocol considerations described in Section 3 by serving each Existing AS112 Zone explicitly. In each case the zone contents are identical, containing only required apex SOA and NS records. Adding or dropping a delegation for an Existing AS112 Zone requires coordination amongst all deployed Existing AS112 Server operators in order to add or drop the zone.

There is no practical expectation that AS112 Server operators coordinate the configuration of their infrastructure or even make their existence known in any systematic way. Delegation of new zones to Existing AS112 Servers is hence problematic; there is an expectation that such delegations would be lame for a significant client population. Since the predictable behaviour of AS112 Servers from clients is desirable, and it is possible that significant variation would have operational consequences, no new zones should be delegated to existing AS112 Servers.

Omniscient AS112 Servers serve an unsigned root zone, containing only required apex SOA and NS records. Adding or dropping a delegation

for an AS112 Zone requires imposes no operational requirements on Omniscient AS112 Server operators.

Delegation of new AS112 Zones should only be made to Omniscient AS112 Servers. The desire to delegate new AS112 Zones therefore imposes a requirement on Omniscient AS112 Servers to listen on addresses which are different to those used by Existing AS112 Servers. Addressing is discussed in Section 5.

By ensuring that Omniscient AS112 Servers listen on Existing AS112 Servers' addresses as well as the new addresses specified in Section 5 a smooth migration is possible, allowing Existing AS112 Servers to be reconfigured as Omniscient AS112 Servers. Omniscient AS112 Servers are therefore a superset of AS112 Servers.

5. Addressing Considerations

Omniscient AS112 Servers listen on the following addresses:

- o IPv4-TBA1 (A.AS112.NET)
- o IPv6-TBA1 (A.AS112.NET)
- o IPv4-TBA2 (B.AS112.NET)
- o IPv6-TBA2 (B.AS112.NET)
- o IPv4-TBA3 (C.AS112.NET)
- o IPv6-TBA3 (C.AS112.NET)

Pv4-TBA1, IPv4-TBA2 and IPv4-TBA3 are covered by a single IPv4 prefix, IPv4-PREFIX-TBA. Similarly, IPv6-TBA1, IPv6-TBA2 and IPv6-TBA3 are covered by a single IPv6 prefix, IPv6-PREFIX-TBA.

The addresses specified for Omniscient AS112 Servers are deliberately different from those assigned to Existing AS112 Servers for reasons discussed in Section 4.

6. Updates to RFC 6304

6.1. Changes to Section 3.4, Routing Software

Omniscient AS112 Nodes with IPv4 connectivity should originate the IPv4 service prefix associated with Existing AS112 Nodes, 192.175.48.0/24, and also the IPv4 service prefix associated with Omniscient AS112 Nodes, IPv4-PREFIX.

Omniscient AS112 Nodes with IPv6 connectivity should originate the IPv6 service prefix IPv6-PREFIX-TBA.

Applying this direction to the "bgpd.conf" file included as an example in this section results in the configuration shown in Figure 1.

```
! bgpd.conf
!
hostname as112-bgpd
password <something>
enable password <supersomething>
!
! Note that all AS112 nodes use the local Autonomous System
! Number 112, and originate IPv4 and IPv6 prefixes (where IPv4
! and IPv6 connectivity is available) as follows:
!
!   IPv4:  192.175.48.0/24
!           IPv4-PREFIX-TBA
!
!   IPv6:  IPv6-PREFIX-TBA
!
! All other addresses shown below are illustrative, and
! actual numbers will depend on local circumstances.
!
router bgp 112
  bgp router-id 203.0.113.1
  !
  address-family ipv4
    network 192.175.48.0
    neighbor 192.0.2.1 remote-as 64496
    neighbor 192.0.2.1 next-hop-self
    neighbor 192.0.2.1 prefix-list AS112-v4 out
    neighbor 192.0.2.1 filter-list 1 out
    neighbor 192.0.2.2 remote-as 64497
    neighbor 192.0.2.2 next-hop-self
    neighbor 192.0.2.2 prefix-list AS112-v4 out
    neighbor 192.0.2.2 filter-list 1 out
    network 192.175.48.0/24
    network IPv4-PREFIX-TBA
  !
  address-family ipv6 unicast
    neighbor 2001:db8::1 remote-as 64496
    neighbor 2001:db8::1 next-hop-self
    neighbor 2001:db8::1 prefix-list AS112-v6 out
    neighbor 2001:db8::1 filter-list 1 out
    neighbor 2001:db8::2 remote-as 64497
    neighbor 2001:db8::2 next-hop-self
    neighbor 2001:db8::2 prefix-list AS112-v6 out
    neighbor 2001:db8::2 filter-list 1 out
    network IPv6-PREFIX-TBA
```



```
!  
ip prefix-list AS112-v4 permit 192.175.48.0/24  
ip prefix-list AS112-v4 permit IPv4-PREFIX-TBA  
!  
ipv6 prefix-list AS112-v6 permit IPv6-PREFIX-TBA  
!  
ip as-path access-list 1 permit ^$
```

Figure 1

6.2. Changes to Section 3.5, DNS Software

Omniscient AS112 Servers with IPv4 connectivity should include DNS software configured to listen on the addresses IPv4-TBA1, IPv4-TBA2 and IPv4-TBA3 in addition to the addresses used by Existing AS112 Servers.

Omniscient AS112 Servers with IPv6 connectivity should include DNS software configured to listen on the addresses IPv6-TBA1, IPv6-TBA2 and IPv6-TBA3.

Omniscient AS112 Servers serve an empty, unsigned root zone instead of explicitly serving the zones specified in [RFC6304].

Applying this direction to the "named.conf" file included as an example in this section results in the configuration fragment shown in Figure 2.

```
options {  
    // The following configuration stanza is for Omniscient AS112  
    // Servers with IPv4 connectivity  
  
    listen-on {  
        127.0.0.1;           // localhost  
  
        // The following address is node-dependent and should be set to  
        // something appropriate for the new AS112 node.  
  
        203.0.113.1;         // local address (globally unique, unicast)  
  
        // the following addresses correspond to Existing AS112 Server  
        // addresses  
  
        192.175.48.1;        // prisoner.iana.org (anycast)  
        192.175.48.6;        // blackhole-1.iana.org (anycast)  
        192.175.48.42;       // blackhole-2.iana.org (anycast)  
  
        // the following addresses are required by Omniscient AS112 Servers
```

```
    IPv4-TBA1;           // A.AS112.NET
    IPv4-TBA2;           // B.AS112.NET
    IPv4-TBA3;           // C.AS112.NET
};

// The following configuration stanza is for Omniscient AS112
// Servers with IPv6 connectivity

listen-on-v6 {
    ::1;                 // localhost

    IPv6-TBA1;           // A.AS112.NET
    IPv6-TBA2;           // B.AS112.NET
    IPv6-TBA3;           // C.AS112.NET
};

directory "/var/named";
recursion no;           // authoritative-only server
query-source address *;
};

// Log queries, so that when people call us about unexpected
// answers to queries they didn't realise they had sent, we
// have something to talk about. Note that activating this
// has the potential to create high CPU load and consume
// enormous amounts of disk space.

logging {
    channel "querylog" {
        file "/var/log/query.log" versions 2 size 500m;
        print-time yes;
    };
    category queries { querylog; };
};

// Substantially empty root zone (replaces explicit zone
// configuration specified in RFC 6304 for Existing AS112 Servers)

zone "." {
    type master;
    file "db.empty";
};

// Also answer authoritatively for the HOSTNAME.AS112.NET zone,
// which contains data of operational relevance.

zone "hostname.as112.net" {
    type master;
```

```
file "db.hostname.as112.net";

// No other zones should be hosted on this name server.
};
```

Figure 2

The "db.empty" file is updated to include references to nameservers used by Omniscient AS112 Servers, as shown in Figure 3.

```
; db.empty
;
; Empty zone for AS112 server.
;
$TTL      1W
@ IN SOA  A.AS112.NET. hostmaster.root-servers.org. (
                                1          ; serial number
                                1W         ; refresh
                                1M         ; retry
                                1W         ; expire
                                1W )       ; negative caching TTL
;
        NS      B.AS112.NET.
        NS      C.AS112.NET.
;
; There should be no other resource records included in this zone.
;
```

Figure 3

6.3. Changes to Section 3.6, Testing a Newly Installed Node

Testing should include all configured service addresses for an Omniscient AS112 Server (IPv4 or IPv6 or both, as appropriate). Note that the IPv4 service addresses include those described in [RFC6304] for Existing AS112 Servers.

7. IANA Considerations

This document describes infrastructure which could be used in the future to direct the IANA to delegate or redelegate infrastructure zones under its administrative control.

However, this document makes no request of the IANA.

8. Security Considerations

The contents of the Security Considerations section of [RFC6304] should be reviewed, since that discussion is pertinent to the operation of Omniscient AS112 Servers as well as Existing AS112 Servers.

The deployment of Omniscient AS112 Servers enables new delegations to AS112 Servers.

Queries received by an AS112 Server might reveal operational data for which there is an expectation of privacy. For example, leaked queries for an organisation's internal DNS names which are sent to an AS112 Server might reveal the existence of those names to the AS112 Server operator. The delegation of new zones to AS112 Servers has the potential to increase opportunities for such unintentional information leakage.

The delegation of new zones to AS112 Servers has the potential to increase the traffic received by those servers. AS112 Server operators are encouraged to monitor traffic levels, and to take appropriate steps if traffic levels threaten the stability of their networks.

9. Acknowledgements

The authors thank and acknowledge the contributions of Dr Paul Vixie, Shane Kerr and Bill Manning in the preparation of this document.

10. References

10.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC6304] Abley, J. and W. Maton, "AS112 Nameserver Operations", RFC 6304, July 2011.

10.2. Informative References

- [RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, July 2011.

Appendix A. Document Notes

This section (and sub-sections) contain information useful for development and review of this document, and should be removed prior to publication.

A.1. Venue

This document is an individual submission, and is not the product of an IETF working group. However, a suitable venue for discussion is the dnsop working group mailing list.

A.2. Textual Substitutions

The strings "IPv4-TBA1", "IPv4-TBA2" and "IPv4-TBA3" should be replaced in this document should be replaced with IPv4 addresses assigned for the purpose described. The covering IPv4 prefix for all three addresses should replace the string "IPv4-PREFIX-TBA".

Similarly, the strings "IPv6-TBA1", "IPv6-TBA2", "IPv6-TBA3" and "IPv6-PREFIX-TBA" should be substituted in the text with assigned production values.

A.3. Open Questions

1. Where to get IPv4 and IPv6 assignments from? There has already been an assignment to DNS-OARC by ARIN for v6 service for AS112 servers.

A.4. Change History

A.4.1. draft-wkumari-dnsop-omniscient-as112-00

Document title changed to include the dnsop keyword, so that IETF document automation can send courtesy notifications of document actions to the dnsop working group.

Abstract and introduction expanded.

RFC2119 requirements notation removed, since this is an informational document and any normative language would be toothless.

Discussion broken out into Protocol Considerations, Operational Considerations and Addressing Considerations.

Detailed updates to [RFC6304] added.

A.4.2. draft-wkumari-omniscient-as112-00

Initial draft, circulated privately, not submitted.

Authors' Addresses

Warren Kumari
Google
1600 Ampitheatre Parkway
Mountain View, CA 94043
USA

Email: warren@kumari.net

William F. Maton Sotomayor
National Research Council of Canada
1200 Montreal Road
Ottawa, ON K1A 0R6
Canada

Phone: +1 613 993 0880
Email: wfms@ryouko.imsb.nrc.ca

Joe Abley
ICANN
12025 Waterfront Drive, Suite 300
Los Angeles, CA 90094-2536
USA

Phone: +1 519 670 9327
Email: joe.abley@icann.org

DNSOP
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

P. Wouters
Red Hat
July 9, 2012

Secure parent-child DNS update use cases
draft-wouters-dnsop-secure-update-use-cases-00.txt

Abstract

DNS zone administrators occasionally need to update data published by a parent zone, such as NS and DS records. Traditionally these updates have happened out-of-band: through DNS registrar interfaces, EPP, websites, or manually by operators. Some updates could also be done using DNS Dynamic Update [RFC2136].

The IETF's DNSOP working group is considering proposing additional mechanisms for such updates, possibly leveraging DNSSEC for authentication.

This document presents some use cases to drive this design work.

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 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Terminology	4
3. DNS records with use cases for automated updates	4
3.1. The DS RRset	4
3.2. The NS RRset	5
3.3. Glue records	5
4. Use cases	5
4.1. DNSSEC use cases in the Registrant, Registrar, Registry model	5
4.1.1. Registrar has not adopted DNSSEC	5
4.1.2. Registrar supports DNSSEC tediously	5
4.1.3. sub-Registrar supports DNSSEC but Registrar does not	6
4.1.4. Registrant not setup to talk EPP to Registrar	6
4.2. DNSSEC use cases with direct parent-child DNS server communication	6
4.2.1. DNS management solution of different vendors cannot communicate	6
4.2.2. DNS management solution requires non-DNS traffic and new Authentication method	6
4.2.3. DNS Management GUI tools are lacking DNSSEC support	6
4.2.4. DNS management solution does not handle when being both child and parent	7
4.3. Non-DNSSEC related DNS record updates	7
4.3.1. NS record and glue updates for the parent	7
4.3.2. Parent changes its infrastructure	7
5. Relationships of zones and name servers	7
5.1. Hidden primary servers	8
5.2. Offline private keys	8
5.3. Parent infrastructure	8
5.4. Update capability indicator	8
5.5. Legalities	8
6. The in-band update process	8
6.1. No automatic updates	8
6.2. Automatic child to parent updates for the DS record only	9
6.3. Fully-automatic child to parent updates	9
6.4. Automatic parent to child updates	9

6.5. Fully-automatic child and parent synchronization	9
6.6. Semi-automatic update	9
7. Applicability of automated updates to DNS infrastructure records	9
7.1. Administrative Criteria	9
7.1.1. Contractual obligations	9
7.1.2. Company policy	10
7.1.3. Separation of roles	10
7.2. Content criteria	10
7.2.1. DS update changing a secure zone to become insecure	10
7.2.2. DS update changing a zone to become bogus	10
7.2.3. DS update changing a zone to become secure	11
7.2.4. NS update causing an outage	11
8. Security Considerations	11
9. IANA Considerations	11
10. Acknowledgements	11
11. References	12
11.1. Normative References	12
11.2. Informative References	12
Author's Address	12

1. Introduction

Existing mechanisms for child-to-parent communication in DNS have some constraints that limit their utility. In particular, they require an authentication, which typically requires an extra credential to be exchanged between parent and child. With the advent of DNSSEC, it might be possible to use DNSSEC to authenticate these updates.

Furthermore, current mechanisms such as dynamic update also require that the child zone be able to reach the master server for the parent zone. In environments with hidden masters, offline DNSSEC signers or other network architecture constraints, this is not always be feasible.

This document identifies the main targets and use cases for automated updates and the concerns related to such automation.

[Note: While the document describes the use-cases with the zone, not the name server, as actor, this should not be taken to mean the signaling must be within the zone]

2. Terminology

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 RFC 2119 [RFC2119].

3. DNS records with use cases for automated updates

This document limits the scope of use cases to those DNS records that relate to the parent-child relationship itself. Policies for the TTL could be dictated by the parent or the child, depending on the relationship.

3.1. The DS RRset

The DS record needs to be updated when the child zone performs a Key Signing Key rollover. The parent name server cannot necessarily confirm the updated information by looking into the child zone, for example when the child zone has a spare, unpublished, DNSKEY record. Some parents want to receive DNSKEYs and create the DS record based on the received record. Other parents do not want to be responsible for creating any data for the child, and want to receive ready-made DS records, optionally restrained by the parent's choices of valid algorithms.

3.2. The NS RRset

Both the child and the parent have a copy of the NS RRset. These RRsets are supposed to be identical. If they differ, it is referred as a "Lame Delegation". Keeping these sets synchronized would result in fewer lame delegations. Modifying the NS RRset is more complicated, as it could involve talking to name servers who do not yet know about the zone.

3.3. Glue records

Glue records are A or AAAA records that are needed to resolve an NS record that has a recursive relationship. For example, if the NS record for example.com points to ns.example.com, then a glue record is added to the parent zone (.com) for ns.example.com. Note that ns.example.com could be used in NS records for other zones as well.

4. Use cases

There are different kind of parent-child relationships. A very common relationship is the TLD registry using a Registry-Registrar-Registrant model. In this model, the child dictates the content to the parent. Another common parent-child relationship is the corporate relationship where the head office dictates some parent zone content to the child.

4.1. DNSSEC use cases in the Registrant, Registrar, Registry model

4.1.1. Registrar has not adopted DNSSEC

Registrant running the child zone needs to convey their DS record to the Registry running the parent zone. Registrant can only communicate to the Registry using a Registrar. This Registrar does not support the EPP option to convey the DS record from Registrant to Registry. By sending an update via DNS to the Registry, Registrant bypasses the limitations of the Registrar. This use case would require some kind of boot-strap.

4.1.2. Registrar supports DNSSEC tediously

Registrar supports sending a DS record to the Registry via EPP. Registrant needs to use a human-oriented website interface of Registrar, which is very hard to automate and would break every time Registrar modifies their website for Registrants. By sending an update via DNS to the Registry, Registrant bypasses the limitations of the Registrar.

4.1.3. sub-Registrar supports DNSSEC but Registrar does not

Registrant can send DNSSEC updates to their (sub)Registrar, but the Registrar does not support receiving updates from sub-Registrar and sub-Registrar cannot communicate to Registry directly. The Registrant or sub-Registrar could bypass the limitations of the Registrar by sending DNSSEC updates directly to the Registry.

4.1.4. Registrant not setup to talk EPP to Registrar

Registrant is a lightweight entity using an off-the-shelve DNSSEC management solution. They have no technical expertise to communicate using EPP to the Registrar or Registry. Their DNS software could automate sending DNSSEC updates to the Registrar or Registry.

4.2. DNSSEC use cases with direct parent-child DNS server communication

4.2.1. DNS management solution of different vendors cannot communicate

Two different vendors have implemented non-standard, vendor-specific methods for non-DNS parent-child interaction. The DNS administrator(s) have different devices that cannot communicate with each other. If a generic DNS method was standardized, devices could implement this method and inter-operate with each other.

4.2.2. DNS management solution requires non-DNS traffic and new Authentication method

A non-DNS method for updating DS records between parent and child has been implemented. This method requires a lot of overhead to deploy. A new authentication method between parent and child is needed, for which there is no standard, causing potential interoperability issues. Firewall zones for DNS servers need to be updated to allow non-DNS traffic. If a generic DNS method was standardized, devices could implement this method and inter-operate with each other.

4.2.3. DNS Management GUI tools are lacking DNSSEC support

The DNS administrator is both administrating parent and child zone using one or more DNS management solutions. These solutions are running known up to date name server software but the vendor has not yet adopted DNSSEC in their GUI. A standardized solution not requiring additional GUI components could support updates more readily.

4.2.4. DNS management solution does not handle when being both child and parent

The DNS administrator uses a vendor product that does not automate adding the DS in the parent zone, despite the child DNSKEY being available to it. The DNS administrator needs to manually calculate the DS record and add it to the parent. They can no longer run automated rollovers due to this required action that can only be performed manually. If a generic DNS method was standardized, the device could send updates irrespective of whether it also manages the parent zone without additional effort.

4.3. Non-DNSSEC related DNS record updates

4.3.1. NS record and glue updates for the parent

Registrant has a difficult time keeping parent glue and NS RRsets up to date due to using a manual process. After establishing an authenticated relationship between parent and child using the DNSKEY/DS records, the parent could update its glue records based on the child zone content, either by regular polling, or by receiving a notification of the child to update. The parent could distribute such a notification to its siblings.

4.3.2. Parent changes its infrastructure

Parent name servers are pulling zones from different hidden primaries run by different departments with hundreds of zones. The parent name server infrastructure changes, and it wants to all its hidden primaries to use a different NS RRset. The parent sends an update to the hidden primaries to update the NS RRset for their zones. This category would also cover dyndns solutions where clients send individual host record updates to a parent that might change its location.

5. Relationships of zones and name servers

While the relationship between child zone and parent zone are well defined, in practice the chain of DNS servers involved is more complicated. Often the authoritative servers for the child zone do not communicate directly with the authoritative servers of the parent zone. Any methods for signaling between the child and parent zone should attempt to accommodate the listed infrastructure.

5.1. Hidden primary servers

Zones could be updated with IXFR/AXFR using hidden primary servers. DNSSEC signers often work this way. These primary name servers are usually restricted via dedicated VPN links or firewalls, and may not be able to determine or communicate with the required parent server for sending or receiving updates.

5.2. Offline private keys

Some DNSSEC signing solutions keep the private key inside an HSM or otherwise keep the private keys offline. Updates would need to be able to be generated offline, transported to an internet connected machine, and then transmitted to the parent zone.

5.3. Parent infrastructure

Some parent zones will require receiving updates for child zones directly from the child name servers, facilitating their current use of firewalls to restrict communication within the network. Other parent zones, such as TLDs, will want to leave their current name server structure unchanged and prefer updates for the child to a special name server dedicated to receive these updates.

5.4. Update capability indicator

Servers or zones that do not support or allow secure updates should not be sent repetitive update requests.

5.5. Legalities

Some deployments need to take legal restrictions into account. One such example is the Registry, Registrar, Registrant model, where the Registrant and Registry have no formal relationship with each other or are prohibited from communicating directly with each other. In such situations, secure automated updates should not be attempted.

6. The in-band update process

Depending on the appropriate process and relationship between parent and child zone, there could be different requirements for the update process.

6.1. No automatic updates

Records must be added or modified by the administrator of the zone using an out-of-band method.

6.2. Automatic child to parent updates for the DS record only

The child can send updates of its DS record to the parent, but cannot request updates to the NS RRset or glue records. The parent must be able to reject DS records that do not comply to its allowed selection of valid DNSKEY algorithms.

6.3. Fully-automatic child to parent updates

The child can send updates of all its records hosted at the parent, including DS records, NS records and glue records. The parent must be able to reject certain updates based on local policy

6.4. Automatic parent to child updates

The parent can send updates to the child for the NS records and glue records.

6.5. Fully-automatic child and parent synchronization

Parent and child automatically synchronize with no interaction on the part of the operators. This could be uni-directional or bi-directional.

6.6. Semi-automatic update

Parent and child synchronize, but only on the request of the parent or child administrator.

7. Applicability of automated updates to DNS infrastructure records

Automation and direct communication might not be appropriate in all scenarios. Implementations should take note of the considerations in this section.

7.1. Administrative Criteria

There are many situations where automated updates would not be allowed, or in practice could not be deployed in certain jurisdictions or corporate structures. Automatic update solutions should allow for disabling any such updates to support these restricted deployments.

7.1.1. Contractual obligations

Some DNS deployments have contractual restrictions that prevents certain parties from directly communicating with each other. For

example, some TLDs using the RRR model do not allow the Registry to talk to Registrants directly.

7.1.2. Company policy

Corporations often separate duties to different individuals or departments, sometimes across different jurisdictions . For example, a DNS officer in one country might not have the authorization of the company to update a DNS zone run by a subsidiary in other country. However, the reverse policy could also be true, where a DNS officer in one country running the parent zone must be able to update any child zone record of a subsidiary in another country.

7.1.3. Separation of roles

A Registrant (or "owner") of a zone might use a subcontractor to run the infrastructure of its zone. It might not be appropriate for the subcontractor to make any changes in the infrastructure of the zone, despite being in possession of required private keys to send changes to the parent. Similarly, a DNS administrator might be using a DNSSEC signing service, but would not want to allow this signing service to make any changes to the zone content other than signing the zone.

7.2. Content criteria

[Note: With NS records, are there any cases where the NS and glue records in the parent zone should not be identical to those in the child zone? What if the child name servers report different NS RRsets?]

When a DNS update is requested by the child zone, the parent zone could check and see if such an update would cause (significant?) harm to the child zone, and potentially refuse such an update.

7.2.1. DS update changing a secure zone to become insecure

If a DS record deletion request would cause the last DS record in the parent for that zone to be deleted, DNSSEC validation for the child zone would change from secure to insecure. A parent zone might wish to refuse such an update or require an additional confirmation.

7.2.2. DS update changing a zone to become bogus

The parent zone has two DS records for a child zone. Only one of these matches a DNSKEY record in the child zone. If a DS record deletion request would cause the valid DS record in the parent zone to be deleted, DNSSEC validation for the child zone would change from

secure to bogus. Similarly, if a child zone is currently not signed, and the parent zone receives a DS record addition request, DNSSEC validation for the child zone would switch from insecure to bogus. A parent zone might wish to refuse such an update or require an additional confirmation.

7.2.3. DS update changing a zone to become secure

If a child zone becomes signed and automatically sends a DS addition request to the parent zone, the child zone would change from insecure to secure. This requires a sustained commitment by the child to maintain its DNSSEC status by regularly resigning its RRSIG records. The operators of the child zone might not be ready for such commitment, resulting in the zone becoming bogus at a later state. A parent zone might wish to refuse such an update or require an additional confirmation.

7.2.4. NS update causing an outage

If a child zone sends an NS update to the parent, the parent zone could check if the new NS records are properly configured to serve the child zone, guaranteeing that no service interruption would be caused by this update. A parent zone might wish to ignore such an update without an explicit override flag. This might be especially important to DNS operators that are unaware of these new DNS update mechanism and believe that changing zone content on the child would never cause any impacts to the parents.

8. Security Considerations

[Note: This currently overlaps with the section above]

An update of a DS record could change the authentication state of the parent-child relationship and should be handled with care. [Note: or require out-of-band signaling?]

9. IANA Considerations

This Internet Draft includes no request to IANA.

10. Acknowledgements

[Note: none yet]

11. References

11.1. Normative References

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

11.2. Informative References

- [RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
- [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.

Author's Address

Paul Wouters
Red Hat

Email: pwouters@redhat.com

