Root Zone KSK Operator Software Maintenance Procedure

Version 3.6

Root Zone KSK Operator Policy Management Authority

12 September 2024

Table of Contents

1 Introduction **3**

2 License **3**

3 Architectural Design and Functional Specification **4**

3.1 KSK Generator 4

3.1.1 Key Generation Handler 5

3.1.2 Key Parameters Validator 5

3.1.3 Trust Anchor Export Handler 5

3.1.4 Trust Anchor Assembler 5

3.2 KSR Signer 5

3.2.1 Key Signing Request (KSR) 6

3.2.2 KSR Processing Handler 6

3.2.3 RRSet Signer 6

3.2.4 Signed Key Response (SKR) 6

3.2.5 Signing Policy Configuration 6

3.3 Webservice KSR/SKR Processor 6

3.4 HSM Configuration 7

4 Software Maintenance Procedures **7**

4.1 Repositories 7

4.2 Branching 7

4.3 Source Code Documentation 7

4.4 Source Code Release 8

4.5 Ceremony Operating ENvironment (COEN) 8

Appendix A: Acronyms **9**

Appendix B: Change Log **10**

1 Introduction

Public Technical Identifiers (PTI) performs the Root Zone Key Signing Key (RZ KSK) Operator role pursuant to a contract from the Internet Corporation for Assigned Names and Numbers (ICANN).

The key management software described in this document includes the tools used by the RZ KSK Operator to create and maintain KSKs and to process Key Signing Requests (KSRs) submitted by the Zone Signing Key (ZSK) operator. The software also contains tools required to execute the Key Ceremonies.

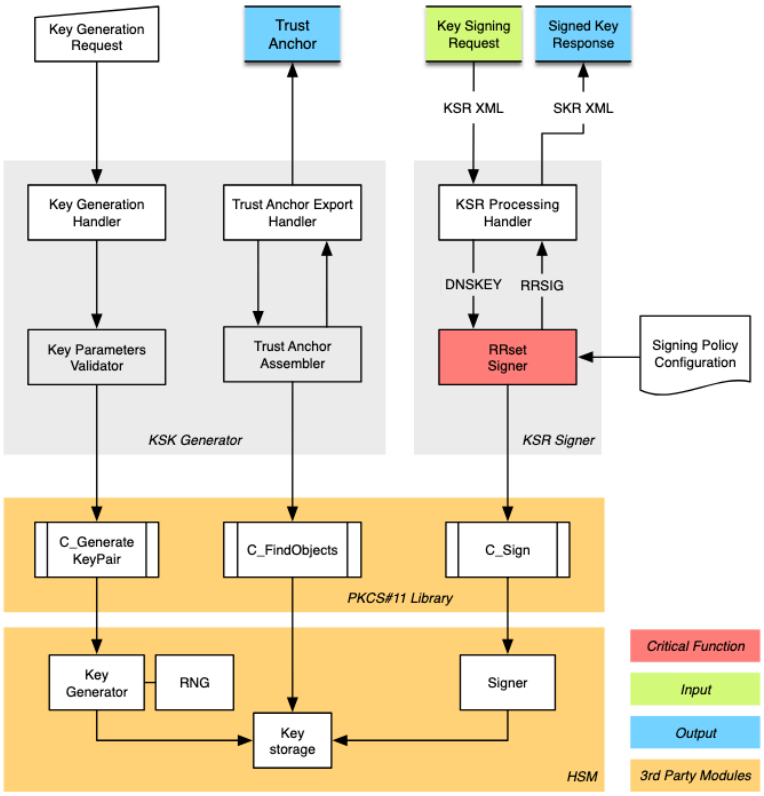
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2 License

The software described in this document is to be distributed under the following license:

3 Architectural Design and Functional Specification

The following figure outlines the high-level architectural design of the KSR Signer software, describing the security domains and functions maintained by the software.



The Key Management Tools consists of the following two critical core components:

- KSK Generator
- KSR Signer

All critical components interact with the user via the UNIX command line interface and uses vendor provided PKCS#11 libraries to communicate with any cryptographic modules.

There is a utility component named Webservice KSR/SKR Processor that is not critical and is not used during the KSK ceremonies.

3.1 KSK Generator

The KSK Generator generates a KSK inside an HSM. Fingerprint and keytag of the public key is returned to the user. Implemented by: kskm.keymaster.keygen

3.1.1 Key Generation Handler

The process of generating a new key involves the validation of the key parameters, successful processing of key generation requests in the HSM, the output of the fingerprint, and keytag of the new KSK.

The Key Generation Handler is responsible for receiving the Key Generation Request, coordinating those various functions with the Key Parameters Validator, the HSM via the PKCS#11 library, and returning appropriate success or failure indications to the user with other information about the generated KSK as described above.

3.1.2 Key Parameters Validator

The Key Parameters Validator ensures that the key generation is well-formed, internally-consistent, and is syntactically correct.

Key Parameters Validator ensures that when a new KSK is generated the key id (a.k.a. keytag) and keylabel are not equal to any previously generated key.

3.1.3 Trust Anchor Export Handler

The Trust Anchor Export Handler will export a public key as an RFC 7958 trust anchor. The output of the function is an XML file in the format defined by the RFC, suitable for the distribution methods in use by IANA.

3.1.4 Trust Anchor Assembler

For each key in the KSR Signer configuration file the Trust Anchor Assembler function will fetch the public key from HSM, check matching algorithm and key parameters, and create the keydigest using kskm.ta.data.KeyDigest.

This data is assembled into the XML format defined by RFC 7958 and returned to the Trust Anchor Export Handler.

3.2 KSR Signer

The KSR Signer will validate a Key Signing Request (KSR), generate signatures to be used in the root zone as RRSIGs over the apex DNSKEY RRSet as requested in the KSR, and output the result as a Signed Key Response (SKR).

The KSR Signer will generate signatures using one or more KSKs available on attached HSMs according to the Signing Policy Configuration. Implemented by: kskm.signer

3.2.1 Key Signing Request (KSR)

A KSR is supplied by the Root Zone Maintainer for processing as an XML document. The KSR specifies a set of time intervals together with the DNSKEY RDATA relating to the ZSKs that will be used during each of them. Every key bundle is signed using every included ZSK as proof of possession of the corresponding private key.

The SKR generated by processing the immediately preceding KSR (usually from the previous KSK Ceremony) is also supplied to the KSR Signer in order to construct a chain of trust to the current KSR, and for various consistency checks to be carried out.

3.2.2 KSR Processing Handler

The processing of a KSR involves successful confirmation that the KSR is well-formed, is syntactically-correct, internally-consistent and does not violate the constraints specified in the Signing Policy Configuration (for example, a KSR should not be allowed to request a signature with a validity period in the past, or too far in the future). It also verifies the KSR is consistent with the SKR produced during the previous KSK Ceremony for the previously-submitted KSR. The DNSKEY RRSets that include the relevant ZSKs and KSKs to be published during each time interval are then constructed. Finally, the signatures over those RRSets by the RRSet Signer are generated and assembled to construct the corresponding SKR.

3.2.3 RRSet Signer

The RRSet Signer generates signatures over a supplied DNSKEY RRSet using one or more KSKs specified in the Signing Policy Configuration. It uses the PKCS#11 interface to the local HSM to carry out the signing functions.

3.2.4 Signed Key Response (SKR)

The SKR is the product of the RRSet Signer. The KSR Processing Handler constructs an SKR from the KSR being processed and the signed RRSets generated by the RRSet Signer. The result is stored in a file as well-formed XML consistent with and validated against the SKR schema.

It is provided as a single file suitable for delivery to the Root Zone Maintainer.

3.2.5 Signing Policy Configuration

The Signing Policy Configuration provides the parameters necessary for KSR processing.

3.3 Webservice KSR/SKR Processor

The KSR/SKR Processor receives a KSR from an authorized client (operated by the Root Zone Maintainer) over a secure network channel (HTTP over TLS with client authentication), provides validation of the KSR (such as integrity, parameters, and validation of the previous SKR) to provide a

useful, real-time response to the client, and generates a notification when a KSR has successfully been submitted. Implemented by: kskm.wksr

3.4 HSM Configuration

HSM configurations for the Key Management Tools are read from hsmconfig files located in the current working directory. The configuration files include a set of environment variables that will be set before loading the vendor provided PKCS#11 library defined by the PKCS11_LIBRARY_PATH variable.

E.g., a configuration file for the AEP Keyper HSM would look something like this:

KEYPER_LIBRARY_PATH=/opt/dnssec

PKCS11_LIBRARY_PATH=/opt/Keyper/PKCS11Provider/pkcs11.linux_gcc_4_1_2_glibc_2_5_x86_64.so.5.02q

4 Software Maintenance Procedures

4.1 Repositories

The key management software is maintained in two separate repositories: one is the public repository for snapshots of the official editions of the key management tools source code which has gone through the proper commissioning process; the second is the private repository that stores the key management tool source code development.

Public repository: https://github.com/iana-org/dnssec-keytools

Internal repository: https://github.com/iana-internal/dnssec-keytools-dev/

4.2 Branching

Non-trivial changes and features SHOULD be developed in a separate branch/tags and merged into master later.

4.3 Source Code Documentation

- Code formatted using Black and isort. Use make reformat to tidy up source code before committing changes.
- Code documentation through the use of Doxygen.
- Documentation includes core method's description, arguments, and return values in line with the code.
- The code shall be PEP 8 compliant and adhere to docstring conventions PEP 257.

4.4 Source Code Release

After a pending release has been tested the following procedure is followed to create the release:

- Name the release with the string dnssec-keytools-VER-YYYYMMDD-COMMIT, where VER is the version, YYYYMMDD is the date, and COMMIT is the commit number.
- Create a compressed (gzip) TAR archive (.tar.gz).
- Add the TAR archive together with a SHA-256 hash to the dist subdirectory.

4.5 Ceremony Operating ENvironment (COEN)

COEN is a live operating system consisting of:

- A custom Debian GNU/Linux Live CD
- Key Management Tools
- PKCS#11 library
- Assorted utilities

Public repository: https://github.com/iana-org/coen

Internal repository: https://github.com/iana-internal/coen-dev

Appendix A: Acronyms

CA	Ceremony Administrator
ICANN	Internet Corporation for Assigned Names and Numbers
IW	Internal Witness
KSK	Key Signing Key
KSR	Key Signing Request
PMA	Root Zone KSK Operator Policy Management Authority
PTI	Public Technical Identifiers
RFC	Request for Comments
RKOS	RZ KSK Operations Security
RZ	Root Zone
ZSK	Zone Signing Key

Appendix B: Change Log

Revision 3 - 04 October 2018

- Converted the document to use the latest Word template.
- Made minor editorial, formatting, and style changes.
- Made all cross-references hyperlinks.
- Adopted the RFC “MUST”, “SHOULD”, etc. convention throughout each document. Added a paragraph to Section 1 that explains the RFC wording convention.
- Added an acronym list.
- Cover: Changed the version from 2.3 to 3.0.

Revision 3.1 - 28 October 2019

- Annual review: Update version information and dates.
- Made minor editorial, formatting, and style changes.
- Updated Appendix A to reflect only the acronyms present in the document.

Revision 3.2 - 04 November 2020

- Annual review: Update version information and dates.
- Section 4.1: Updated repository URLs.

Revision 3.3 - 22 September 2021

- Annual review: Update version information and dates.
- Section 1: Clarified use of key words as described in RFC 2119 and RFC 8174

Revision 3.4 - 19 October 2022

- Annual review: Update version information and dates.

Revision 3.5 - 12 October 2023

- Annual review: Update version information and dates.
- Section 3: Architectural Design and Functional Specification: Updated to reflect the new KSK signer software.
- Section 4: Software Maintenance Procedures: Updated to reflect the new KSK ceremony signer software.

Revision 3.6 - 12 September 2024

- Annual review: Update version information and dates.