Root Zone KSK Operator Software Maintenance Procedure

Version 3.3

Root Zone KSK Operator Policy Management Authority

22 September 2021

Table of Contents

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 Design

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

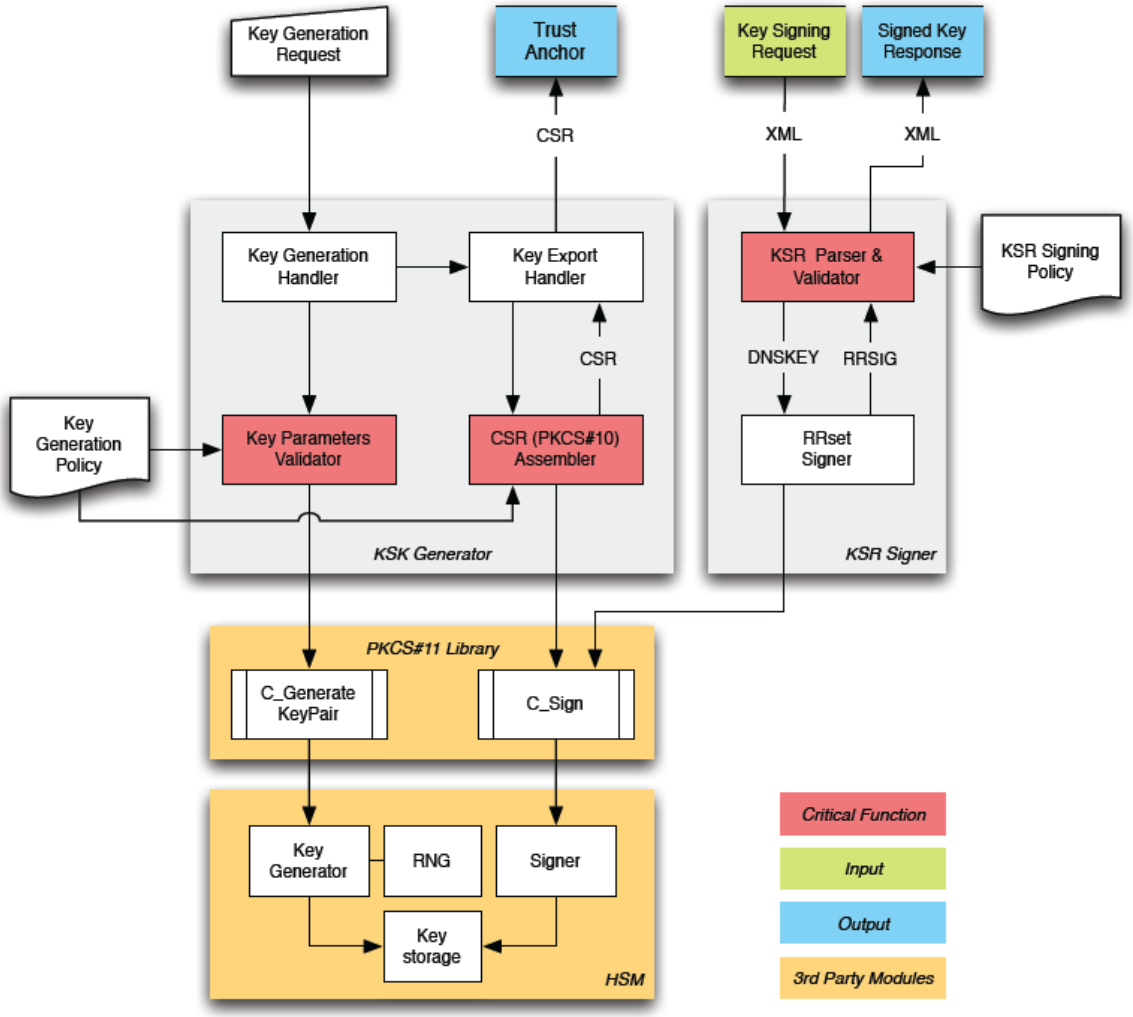
- KSK Generator (kskgen)
- KSR Signer (ksrsigner)

There are also a number of utility components:

- Change PIN Utility (changepin)
- Key Backup (keybackup)
- Webservice KSR Processor (wksr)

These components are not critical and are not described further in this document. All components interact with the user via the UNIX command line interface and uses vendor provided PKCS#11 libraries to communicate with any cryptographic modules.

3.1 Architectural Design



3.2 Specification of Security-Enforcing Functions

The following set of requirements MUST be met by the Key Management Tools:

Key Parameters Validator:

[R1] The security-enforcing function SHALL interpret the key generation policy and only accept the policy as input for the key generation request.

CSR Assembler:

[R2] The security-enforcing function SHALL construct a PKCS#10 Certificate Signing Request (CSR) using the Key Signing Key (KSK) generated using the PKCS#11 library. The CSR attributes SHALL be read from the key generation policy and the resulting CSR SHALL be formatted as specified in section 2.2 of "DNSSEC Trust Anchor Publication for the Root Zone" [RFC 7958]

KSR Parser & Validator:

Given a key signing request, KSR(n), and a previously signed key response, SKR(n-1), the security-enforcing function SHALL perform the following checks:

[R3] For each key bundle in KSR(n), verify the signature by each ZSK to confirm proof of possession of each ZSK. The inception and expiration times of the Resource Record Signatures (RRSIGs) in the KSR are ignored when checking proof of possession.

[R4] Check the integrity of SKR(n-1) by verifying the KSK signature over each key bundle. This verifies the integrity of all the ZSKs in SKR(n-1). When this check is performed as part of the KSR signing process, the KSK SHALL be verified using the private KSK used for signing the KSR.

[R5] Compare the pre-published ZSK from the last key bundle of SKR(n-1) with the ZSK published in the first key bundle of KSR(n). These keys MUST be identical.

[R6] Compare the post-published ZSK from the first key bundle of KSR(n) with ZSK published in the last key bundle of Signed Key Response (SKR)(n-1). These keys MUST be identical.

[R7] The security-enforcing function SHALL validate the parameters in the KSR against the KSR Signing Policy before requesting signing of any data.

[R8] Before the RRset Signer is invoked to sign a KSR, the security-enforcing function SHALL calculate a SHA-256 hash over the KSR XML file and present the hash to the operator and request confirmation. If the operator does not confirm the hash, the program MUST terminate.

[R9] After the RRset Signer has emitted a signed SKR, the security-enforcing function SHALL calculate a SHA-256 hash over the SKR XML file and present the hash to the operator.

3.3 Modular Design Description

3.3.1 KSK Generator

The KSK Generator (kskgen) generates a key signing key (KSK) inside a Hardware Software Module (HSM). The public key is signed by the private key and exported as a Certificate Signing Request (CSR) in PKCS#10 format.

The key generation configuration is statically compiled into the program and is defined in kskparams.h. Generating keys of an algorithm other than RSA is not supported.

- Key Generation Handler kskgen.c:main()
- Key Export Handler kskgen.c:main()
- Key Parameters Validator pkcs11_dnssec.c:pkcs11_genrsakey()
- CSR Assembler kskgen.c:create_csr()

3.3.2 KSR Signer

The KSR Signer (ksrsigner) will process a Key Signing Request (KSR) from the ZSK operator, validate and sign the request, finally emitting a Signed Key Response (SKR). The KSR signer policy is statically compiled into the program and is defined in ksrpolicy.h.

- KSR Parser ksrsigner.c:main() and ksrcommon.c:xmlparse()
- RRset Signer ksrsigner.c:rrsig()

3.3.3 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.GCC4.0.2.so.4.07

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 store 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

All source code is documented using Doxygen (http://www.doxygen.org/). To generate the HTML documentation (written into doxygen-doc/html/), use the following command:

make doxygen-run

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 icann-keytools-YYYYMMDD, where YYYYMMDD is the date.
- Run autogen.sh to rebuild the configure script and Makefiles.
- Configure the software using ./configure.
- Create a compressed (gzip) TAR archive (.tar.gz).
- Add the TAR archive together with a SHA-256 hash to the dist sub-directory.

4.5 Key Ceremony DVD

The Key Ceremony DVD is a binary distribution consisting of:

- A stripped UNIX/LINUX Live CD,
- The Key Management Tools (both source and binary), and
- The AEP Keyper PKCS#11 provider and assorted utilities.

DVD Release Procedures:

- Build the Key Ceremony DVD ISO file
- Create an attestation of the SHA-256 hash for DVD

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