Source Code Review Followup:

Domain Name Security Extensions Key Signing Key Management Tools

October 7, 2024

Reviewers:

Paul Wouters

Introduction

No Hats Corporation has performed a second Source Code Review of the Domain Name Security Extensions ("DNSSEC") Key Signing Key Management Tools software to certify that:

A. There is a documented architectural design describing the security domains and functions maintained by the signer.
B. The architectural design demonstrates that the signer system prevents bypass of the security-enforcing functionality.
C. There is a functional specification completely representing the signer system and all operations associated with it.
D. There is a modular design description and a one-to-one correspondence with the modular decomposition of the implementation.
E. The implementation representation completely and accurately implements the security-enforcing functions.

The software version ("Software") that No Hats has reviewed can be identified by the following repository values and last commit:

> https://github.com/iana-org/dnssec-keytools.git
> commit 872cbcdd30eb9a20e5dd96b8532d01dc15a16d0b04de2abda12c35
> (HEAD -> feature/2024, origin/feature/2024)
>
> Merge: 8017697 5fb736a
> Author: Jakob Schlyter <jakob@kirei.se>
> Date: Wed Aug 21 08:57:07 2024 +0200
>
> Merge pull request #81 from iana-internal/feature/eddsa
>
> Support for EdDSA.

No Hats and all of its personnel that performed work on this contract have not been, directly or indirectly, involved in the creation of the Software.

No Hats has reviewed all the source code, documentation and other files that came with the Software. The complete list of engineers who worked on this audit are:

> Paul Wouters paul@nohats.ca

This deliverable has been provided as a PDF file, along with a OpenPGP signed digital signature made by Paul Wouters, President of No Hats Corporation using a 4096 bit RSA key identified by key id 0x62D3582FE0FD94D2 with key fingerprint F384E78A1D3352D3EB928AB862D3582FE0FD94D2

Executive Summary

The difference between the previously tested version 0.0.1 and feature/2024 has been audited.

These differences cover 237 commits, of which 188 actual commits with changes. The remainder are merge commit artifacts. The total changes are 2962 additional lines and 2285 removed lines.

The software still conforms with its design goal. The software is written well, properly factored out and split into modules, and supported by test cases. It has been updated to comply with the recent python version 3.12. The new ecdsa/eddsa code that is not yet scheduled to be used is still missing test case coverage, but this does not affect the current required functionality of the software.

The unit tests work as expected and prove the assurances that the software can only operate within valid parameters set out by the DPS.

No errors were found within the software itself or with the testing.

Improved tooling has resulted in improved visibility and improved (visibility of) coverage. The update from `black`, `isort`, `green` and `voluptuous` to `poetry`, `ruff`, `pydantic`, `pytest` and `coverage` has been a valuable upgrade. pylint still shows a few minor items that might be worth addressing.

Any features not required or allowed by the DNSSEC Practice Statement (DPS) are still only available as configuration options or not at all. Proper configuration files ensure all requirements and limitations set by the DPS are accepted.

In total, 173 test cases pass (2 cannot be run by us). Testing coverage is 95% with 228 statements not covered which are either not yet in scope for the current DPS, testing code itself or documented as to why a test covering the code is not available or feasible.

We believe the software is suitable for production use.

The software in its current form is still not very easily installed on a regular OS such as RHEL/CentOS or Debian/Ubuntu. Running the test suite is not possible on the COEN OS itself, even though the software does seem to work as expected based on further manual testing. We still recommend that the software is extended a little to facilitate this use, as it will increase the number of people who will be able and willing to test the software, which will increase the public acceptance of the software. Adding more (python) software on the COEN OS image itself would also allow it run all the selftests before each production use of the software that would catch any accidental version change or bugs of various python modules between regular OS or docker testing and the COEN OS image itself.

Evaluation of Code

Structure / organization

The software is well structured into several python modules along functional boundaries.

Linting

Non functional code quality is governed by a lint ruleset that can be checked by running `make lint`. Running this command produces no warnings from the poetry and ruff python modules.
When running pylint manually on the src/ directory, it produces 589 warnings. After filtering these for non-important ones, 149 are left that are worth looking at. See Appendix A.

Having a noisy lint output is not useful for ongoing development and maintenance, as interpreting the output requires a lot of effort.

Code formatting

Consistency of formatting is ensured via `ruff` and `poetry` and no longer via `black` and `isort`. These tools can be run via `make reformat`. This has made the formatting more consistent and no longer produces warnings.

Type annotation

Large parts of the source code are annotated with types. In some places there is added code to help the type checker `mypy` infer types when it cannot do so on its own.
The `make typecheck` command runs mypy and all previous errors have been fixed.

CI environment

The CircleCI system has been updated for a more modern approach using GitHub Actions.

Evaluation of Test Suite

The test suite has been updated to use `poetry`, `coverage` and `pytest`, instead of `green`, and is invoked via `make test`. This is an excellent improvement.

Code Coverage

Code Coverage is validated using `coverage` and is invoked using `make coverage`.
There is a 95% Code Coverage, with 228 statements not covered.

The coverage tests show a number of items not covered by tests. These have been investigated and no issues were found. The reasons for not being covered are:

- Some code is the test code itself.
- Some code documents why it triggers an uncovered false positive.
- Some code is errors for not yet implemented functionality, or for abstracted functionally, such as src/kskm/common/public_key.py as a base class.
- Some code is error code that's unreachable right now, but might not be after a new error would be introduced, so these are defense-in-depth calls, causing a false positive.
- src/kskm/common/ecdsa_utils.py and src/kskm/common/eddsa_utils.py are not well covered but are also still in development and not currently required or allowed for production use as these algorithms fall outside the current DPS.
- Some code would require hsm to operate incorrectly to get triggered, which is hard to do with testing, and the non-covered code just raises an exception leading to program termination.

Some uncovered cases that could fairly easily be covered:

- class DNSRecords method format()
- src/kskm/signer/__init__.py function output_skr_xml(). although it is a thin layer on top of src/kskm/skr/output.py:def skr_to_xml() which is covered.
- src/kskm/common/integrity.py sha2wordlist is not tested directly, but is tested via its import via binary kskm-sha2wordlist

KSR Archive

2 tests are dependent on a “KSR Archive” and skipped if no such archive is provided and set via KSKM_KSR_ARCHIVE_PATH. There is no information provided where to obtain such an archive.

Test coverage

Script entry points not tested (unchanged from previous audit)

There is no test coverage for the command line entry points (src/kskm/tools). This is especially important for ksrsigner.py as it calls out to

- Load configuration, KSR and SKR
- Initialize the HSMs
- Check the KSR/SKR
- Create the new SKR

If ksrsigner.py were (in a future change) to skip the call to the ksr/skr checks, it could still continue to issue the SKR as no test coverage exists to validate.

As a result several assurance claims cannot be validated ia the test suite:
The assurance claim 1.1.2.3

> *“Test cases provide evidence that if a policy is rejected, 'ConfigurationError' is raised* ***and the load configuration procedure in main of 'keymaster.py' will return 'False'.***

The bold part #2 is not covered by the test suite.

The assurance claim 1.2.1.1

> *“Test cases provide evidence that (a) a SKR with signatures which can not be verified using the KSK is rejected, and (b) a SKR with valid signatures is accepted.”*

depends on check_skr_and_ksr being called and its return value obeyed, which is not tested as claimed.

Having test suite coverage is crucial for having confidence that future changes to the software do not regress critical functionality. We recommend moving some of the logic of ksrsigner.py into shared code so that the krsigner.py binary only parses arguments and calls library functions.

SKR xml file not fully validated

The SKR components are validated in memory, and then written into xml format to a file. The number of Request Bundles in the Request is verified although misconfiguration could state different numbers of received and written Request bundles in a Request. DNSKEY flags of the KSK are not verified ? DNSKEY flags of the ZSK are verified on input. There is no Assurance Claim that has ownership of the Request and Request Bundle verification of the SKR.

Assurance claims and DPS requirements

The software comes with a set of assurances specified in the file assurance-cases.md

A new version of the DPS (7th edition, 2024-03-15 was issued since the previous audit, but changes do not affect any code, as the only changes in this version were:

- Overall: Replaced smartcards and cards with credentials.
- Section 4.2.2: Added Domain Key.
- Section 5.2.1: Updated FIPS requirement.
- Appendix A.2: Defined Domain Key.

The assurances and their test cases as described below have not changed since the previous audit.

The Assurance claims should cover all the requirements from the DPS. At times, the software allows more flexibility than the DPS, in which case the software configuration should be able to produce a system that exactly matches all requirements of the DPS. The configuration file in config/ksrsigner.yaml provides a DPS compliant configuration, but has existing keys in it, so it cannot be used to test the software from a scratch configuration. It might be useful to provide a claim of Root Zone DPS compliance when the software detects a fully DPS compliant configuration file.

Missing Assurance claims

The DPS does not place any requirements on the SKR other than the RRSIG signature validation in Section 6.8. This is a bug in the DPS. There are further "requirements" on the SKR, such as proper DNSKEY flags, same amount of Request bundles in the SKR as there are in the KSR. While there is code and test case coverage for this, there is no Assurance Claim because there was no DPS requirement listed. The covered code and tests should probably be added to a new Assurance Claim.

Claims validated against test suite

The claims are still fully covering the DPS requirements. Please see the previous audit for full details.

Evaluation of DPS compliance

This chapter lists the related sections of the DNSSEC Practice Statement (DPS) for the Root Zone KSK Operator as published at https://www.iana.org/dnssec/icann-dps.txt and describes the compliance of the software to those DPS sections.

There has been no change in the latest DPS that requires changes to implementations.

DPS enumeration and Assurance Claims

The report showing full compliance is unchanged from the previous audit.

Compliance is based on a properly DPS compliant **request_policy** section in the used **ksrsigner.yaml** configuration file.

See previous audit for details.

Clashing Key Tags (modified from previous audit)

The DPS still does not state that it wishes to avoid a Key Tag clash between KSK and ZSK or previous KSKs. As such, no Assurance Claim is rejecting any received ZSK's or KSKs that have the same Key Tag value as the current or previous KSKs, with the exception of the current and next KSK Key Tag which is prevented. Note that a manual step in the ceremony script can still reject a new KSK based on Key Tag and for the key ceremony to rerun the generation of a new KSK resulting in a new Key Tag without such issues.

It is still recommended that an Assurance Claim and supporting code is added to prevent clashing Key Tags, despite that this is currently not a DPS requirement. This will also further require that the DPS for the ZSK takes the KSK Key Tags into account.

Testing of the script tools

Since the tools have not been changed or updated with new functionality, the detailed reports of the previous audit are valid against the current versions as well.

Suitability for third party users

Since the tools have not been changed or updated with new functionality, the detailed reports of the previous audit are valid against the current versions as well.

Software Installation

The software installation has not been changed or updated with new functionality, the detailed reports of the previous audit are valid against the current versions as well, with the exception that we were now able to run the docker environment, which was convenient for some of the performed testing.

Usability of the command line tools

Since the tools have not been changed or updated with new functionality, the detailed reports of the previous audit are valid against the current versions as well.

Minor complaints

Immutable python objects

Several validations rely on immutability/hashabilty of classes from kskm.common.data (Key class, for example, used to check if a key is in a bundle like `key in some_list`). The hashability is currently ensured by `@dataclass(frozen=True)` but no test case verifies that this is how it is done to prevent a less safe hashing from being implemented or that some fields are excluded by hash=None.

Git commit messages and merges

Additional time was spent due to poor git commit messages and wide branches and merges. For future use, we recommend to perform a clean-up rebase step to make the commits more logical, and have the git history contain less wide branch merges.

Key digests

The code contains:

```
keydigests={DIGEST_2010, DIGEST_2017},
```

There is now a 2024 new Key Digest available (key tag 38696). Should this be included ?

Wrong comment for function

```
def test_RSA_wrong_exponent(self) -> None:
"""Test validating a KSR with RSA-SHA1 (not supported)"""
```

It seems the comment was forgotten to be updated after a copy and paste action ?

Import module sorting

The following change was made:

```
-ignore = ["E501"]
+ignore = ["E501", "I001"]
```

Why not sort the imports to avoid needing to ignore I001?

DS hash length

The following code is present:

```
ds_sha256: str | None = Field(default=None, pattern=r"^[0-9a-fA-F]+$")
```

and:

```
HexDigestString = Annotated[str, StringConstraints(pattern=r"^[0-9a-fA-F]+$")]
```

Technically. this matches things that can be too short or too long. It should match on the exact length of the SHA256 digest.

Exponent use

Test_Sign_Formatting() uses an RSA exponent of 3. Since bad exponent testing is done elsewhere, we think this call should not be using an obsolete exponent of 3.

Format of the KSR and SKR files

It is understood that the KSR and SKR formats are currently in production use, and not planned to be updated for enrollment of this software. As thus that the software cannot be modified to use an improved format. See the previous audit report for details.

Required libraries for the dnssec-keytools software

An audit of the Operating System (OS) and required python libraries is out of scope for this audit. However, an inventory and brief evaluation of the python libraries used has been included.

The software is written in python, and requires a number of libraries. The specific requirements have been reduced compared to the previous audit that used `setup.py` install_requires, whereas the current `pyproject.toml` seems to have far less restrictions on versioning and thus encourages newer versions appropriately.

Previously, the no longer supported PyOpenSSL was used in src/kskm/wksr/peercert.py but this has now been properly updated to use the python `cryptography` module.

Suggested Changes

There are no required changes. Our recommended changes from the previous audit are still applicable.

Appendix A: pylint output

The following minor warnings and convention messages were filtered out:
(see https://pylint.readthedocs.io/en/latest/user_guide/messages/messages_overview.html)

C0103
C0114
C0115
C0116
C0209
C0301
R0801
R0801
W1201
W1203

The below messages are worth addressing, even if they currently do not seem to negatively affect the software.

src/kskm/common/config.py:46:4: W0105: String statement has no effect (pointless-string-statement)
src/kskm/common/config_wksr.py:20:8: W0511: TODO: could be a colon-separated string too before (fixme)
src/kskm/common/data.py:194:8: C0415: Import outside toplevel (kskm.common.ecdsa_utils.ecdsa_public_key_without_prefix, kskm.common.ecdsa_utils.expected_ecdsa_key_size, kskm.common.ecdsa_utils.get_ecdsa_pubkey_size, kskm.common.ecdsa_utils.is_algorithm_ecdsa) (import-outside-toplevel)
src/kskm/common/data.py:215:12: R1714: Consider merging these comparisons with 'in' by using 'flags in (FlagsDNSKEY.ZONE.value | FlagsDNSKEY.SEP.value, FlagsDNSKEY.ZONE.value | FlagsDNSKEY.SEP.value | FlagsDNSKEY.REVOKE.value, FlagsDNSKEY.ZONE.value)'. Use a set instead if elements are hashable. (consider-using-in)
src/kskm/common/data.py:213:40: W0613: Unused argument 'info' (unused-argument)
src/kskm/common/data.py:227:8: C0415: Import outside toplevel (kskm.common.dnssec.calculate_key_tag) (import-outside-toplevel)
src/kskm/common/eddsa_utils.py:65:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
src/kskm/common/parse_utils.py:125:13: R1714: Consider merging these comparisons with 'in' by using 'what in ('S', '')'. Use a set instead if elements are hashable. (consider-using-in)
src/kskm/common/parse_utils.py:92:0: R0912: Too many branches (13/12) (too-many-branches)
src/kskm/common/public_key.py:39:8: C0415: Import outside toplevel (kskm.common.ecdsa_utils.KSKM_PublicKey_ECDSA, kskm.common.ecdsa_utils.is_algorithm_ecdsa) (import-outside-toplevel)
src/kskm/common/public_key.py:40:8: C0415: Import outside toplevel (kskm.common.eddsa_utils.KSKM_PublicKey_EdDSA, kskm.common.eddsa_utils.is_algorithm_eddsa) (import-outside-toplevel)
src/kskm/common/public_key.py:41:8: C0415: Import outside toplevel (kskm.common.rsa_utils.KSKM_PublicKey_RSA, kskm.common.rsa_utils.is_algorithm_rsa) (import-outside-toplevel)
src/kskm/common/public_key.py:54:8: W0107: Unnecessary pass statement (unnecessary-pass)
src/kskm/common/public_key.py:59:8: W0107: Unnecessary pass statement (unnecessary-pass)
src/kskm/common/public_key.py:64:8: W0107: Unnecessary pass statement (unnecessary-pass)
src/kskm/common/public_key.py:70:8: W0107: Unnecessary pass statement (unnecessary-pass)
src/kskm/common/public_key.py:75:8: W0107: Unnecessary pass statement (unnecessary-pass)
src/kskm/common/rsa_utils.py:39:4: W0621: Redefining name 'rsa' from outer scope (line 8) (redefined-outer-name)
src/kskm/common/rsa_utils.py:59:4: E0213: Method '_check_algorithm' should have "self" as first argument (no-self-argument)
src/kskm/common/tests/test_config_schema.py:22:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/common/tests/test_config_schema.py:44:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

src/kskm/common/tests/test_config_schema.py:51:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/common/tests/test_config_schema.py:65:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/keymaster/inventory.py:31:55: R1735: Consider using '{}' instead of a call to 'dict'. (use-dict-literal)
src/kskm/keymaster/inventory.py:51:0: R0914: Too many local variables (17/15) (too-many-locals)
src/kskm/keymaster/inventory.py:51:0: R0912: Too many branches (13/12) (too-many-branches)
src/kskm/keymaster/tests/test_inventory.py:41:20: E1133: Non-iterable value self.p11modules is used in an iterating context (not-an-iterable)
src/kskm/ksr/parse_utils.py:60:8: W0622: Redefining built-in 'id' (redefined-builtin)
src/kskm/ksr/verify_bundles.py:74:0: R1711: Useless return at end of function or method (useless-return)
src/kskm/ksr/verify_bundles.py:74:39: W0613: Unused argument 'policy' (unused-argument)
src/kskm/ksr/verify_bundles.py:101:0: R0912: Too many branches (16/12) (too-many-branches)
src/kskm/ksr/verify_header.py:47:13: W0613: Unused argument 'request' (unused-argument)
src/kskm/ksr/verify_header.py:47:31: W0613: Unused argument 'policy' (unused-argument)
src/kskm/ksr/verify_policy.py:368:13: W0511: TODO: Is it perhaps only the _last_ interval in a cycle that should be permitted to be 9 or 11 days? (fixme)
src/kskm/ksr/verify_policy.py:85:4: C0200: Consider using enumerate instead of iterating with range and len (consider-using-enumerate)
src/kskm/ksr/verify_policy.py:195:11: R1716: Simplify chained comparison between the operands (chained-comparison)
src/kskm/ksr/verify_policy.py:293:4: C0200: Consider using enumerate instead of iterating with range and len (consider-using-enumerate)
src/kskm/ksr/verify_policy.py:344:4: C0200: Consider using enumerate instead of iterating with range and len (consider-using-enumerate)
src/kskm/ksr/tests/common.py:69:4: R0913: Too many arguments (12/5) (too-many-arguments)
src/kskm/ksr/tests/common.py:69:4: R0917: Too many positional arguments (12/5) (too-many-positional-arguments)
src/kskm/ksr/tests/common.py:144:4: W0237: Parameter 'request_bundle' has been renamed to 'bundle1' in overriding 'Test_Requests_With_Two_Bundles._make_request' method (arguments-renamed)
src/kskm/ksr/tests/common.py:252:4: R0913: Too many arguments (12/5) (too-many-arguments)
src/kskm/ksr/tests/common.py:252:4: R0917: Too many positional arguments (12/5) (too-many-positional-arguments)
src/kskm/ksr/tests/test_archive.py:116:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_basics.py:13:8: C0415: Import outside toplevel (kskm.ksr) (import-outside-toplevel)
src/kskm/ksr/tests/test_parse_ksr_root.py:19:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_parse_ksr_root.py:45:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_signature.py:18:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_validate_dnspython.py:41:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_validate_signatures.py:136:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/test_validate_signatures.py:146:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/ksr/tests/utils.py:17:4: C0415: Import outside toplevel (hashlib) (import-outside-toplevel)
src/kskm/ksr/tests/utils.py:19:4: C0415: Import outside toplevel (kskm.common.config.KSKMConfig) (import-outside-toplevel)
src/kskm/ksr/tests/utils.py:20:4: C0415: Import outside toplevel (kskm.misc.hsm.init_pkcs11_modules) (import-outside-toplevel)
src/kskm/ksr/tests/utils.py:25:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/misc/hsm.py:158:14: E1101: Instance of 'FieldInfo' has no 'sign' member (no-member)
src/kskm/misc/hsm.py:170:0: R0902: Too many instance attributes (9/7) (too-many-instance-attributes)
src/kskm/misc/hsm.py:173:4: R0912: Too many branches (15/12) (too-many-branches)
src/kskm/misc/hsm.py:433:4: R0914: Too many local variables (17/15) (too-many-locals)
src/kskm/misc/hsm.py:659:9: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/misc/tests/test_hsm_ops.py:40:20: E1133: Non-iterable value self.p11modules is used in an iterating context (not-an-iterable)
src/kskm/misc/tests/test_hsm_ops.py:46:17: E1136: Value 'self.p11modules' is unsubscriptable (unsubscriptable-object)
src/kskm/signer/policy.py:53:38: W0613: Unused argument 'policy' (unused-argument)
src/kskm/signer/policy.py:71:38: W0613: Unused argument 'policy' (unused-argument)
src/kskm/signer/sign.py:37:0: R0914: Too many local variables (19/15) (too-many-locals)
src/kskm/signer/sign.py:130:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
src/kskm/signer/sign.py:168:0: R0913: Too many arguments (6/5) (too-many-arguments)
src/kskm/signer/sign.py:168:0: R0917: Too many positional arguments (6/5) (too-many-positional-arguments)

src/kskm/signer/verify_chain.py:35:4: W0106: Expression "[logger.debug(x) for x in
format_bundles_for_humans(last_skr.bundles)]" is assigned to nothing (expression-not-assigned)
src/kskm/signer/verify_chain.py:37:4: W0106: Expression "[logger.debug(x) for x in format_bundles_for_humans(ksr.bundles)]"
is assigned to nothing (expression-not-assigned)
src/kskm/signer/tests/test_mockedhsm_sign.py:78:47: W0613: Unused argument 'hash_using_hsm' (unused-argument)
src/kskm/signer/tests/test_mockedhsm_sign.py:63:0: R0903: Too few public methods (1/2) (too-few-public-methods)
src/kskm/signer/tests/test_mockedhsm_sign.py:145:4: R0913: Too many arguments (7/5) (too-many-arguments)
src/kskm/signer/tests/test_mockedhsm_sign.py:145:4: R0917: Too many positional arguments (7/5)
(too-many-positional-arguments)
src/kskm/signer/tests/test_mockedhsm_sign.py:129:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_mockedhsm_sign.py:130:8: W0201: Attribute 'schema' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_mockedhsm_sign.py:143:8: W0201: Attribute 'request_zsk_policy' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_mockedhsm_sign.py:126:0: R0903: Too few public methods (1/2) (too-few-public-methods)
src/kskm/signer/tests/test_mockedhsm_sign.py:198:8: W0201: Attribute 'zsk_keys' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_signer_policy.py:22:0: R0902: Too many instance attributes (10/7) (too-many-instance-attributes)
src/kskm/signer/tests/test_signer_policy.py:27:13: W1514: Using open without explicitly specifying an encoding
(unspecified-encoding)
src/kskm/signer/tests/test_signer_policy.py:31:13: W1514: Using open without explicitly specifying an encoding
(unspecified-encoding)
src/kskm/signer/tests/test_signer_policy.py:35:13: W1514: Using open without explicitly specifying an encoding
(unspecified-encoding)
src/kskm/signer/tests/test_signer_policy.py:39:13: W1514: Using open without explicitly specifying an encoding
(unspecified-encoding)
src/kskm/signer/tests/test_softhsm_sign.py:72:9: W1514: Using open without explicitly specifying an encoding
(unspecified-encoding)
src/kskm/signer/tests/test_softhsm_sign.py:85:20: E1133: Non-iterable value self.p11modules is used in an iterating context
(not-an-iterable)
src/kskm/signer/tests/test_softhsm_sign.py:132:4: R0913: Too many arguments (7/5) (too-many-arguments)
src/kskm/signer/tests/test_softhsm_sign.py:132:4: R0917: Too many positional arguments (7/5)
(too-many-positional-arguments)
src/kskm/signer/tests/test_softhsm_sign.py:80:8: W0201: Attribute 'p11modules' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:91:8: W0201: Attribute 'zsk_key_label' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:92:8: W0201: Attribute 'ksk_key_label' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:93:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:94:8: W0201: Attribute 'schema' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:109:8: W0201: Attribute 'request_zsk_policy' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:388:4: R0914: Too many local variables (16/15) (too-many-locals)
src/kskm/signer/tests/test_softhsm_sign.py:238:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:318:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:355:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:403:8: W0201: Attribute 'config' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:239:8: W0201: Attribute 'schema' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:319:8: W0201: Attribute 'schema' defined outside __init__
(attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:356:8: W0201: Attribute 'schema' defined outside __init__
(attribute-defined-outside-init)

src/kskm/signer/tests/test_softhsm_sign.py:404:8: W0201: Attribute 'schema' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:242:8: W0201: Attribute 'zsk_key_label' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:243:8: W0201: Attribute 'ksk_key_label' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:491:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:552:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:599:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:632:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:660:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:688:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:714:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:753:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/signer/tests/test_softhsm_sign.py:757:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/signer/tests/test_softhsm_sign.py:748:8: W0201: Attribute 'config' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:751:8: W0201: Attribute 'data_dir' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:754:12: W0201: Attribute 'ksr_xml' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:755:12: W0201: Attribute 'ksr' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:758:12: W0201: Attribute 'last_skr_xml' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:759:12: W0201: Attribute 'last_skr' defined outside __init__ (attribute-defined-outside-init)
src/kskm/signer/tests/test_softhsm_sign.py:761:8: W0201: Attribute 'policy' defined outside __init__ (attribute-defined-outside-init)
src/kskm/skr/tests/test_output.py:35:8: C0200: Consider using enumerate instead of iterating with range and len (consider-using-enumerate)
src/kskm/skr/tests/test_parse_skr_root.py:18:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/tools/keymaster.py:114:4: W0613: Unused argument 'config' (unused-argument)
src/kskm/tools/keymaster.py:139:0: R0914: Too many local variables (16/15) (too-many-locals)
src/kskm/tools/keymaster.py:269:11: W0718: Catching too general exception Exception (broad-exception-caught)
src/kskm/tools/keymaster.py:139:0: R0915: Too many statements (52/50) (too-many-statements)
src/kskm/tools/ksrsigner.py:233:11: W0718: Catching too general exception Exception (broad-exception-caught)
src/kskm/tools/ksrsigner.py:176:0: R0912: Too many branches (15/12) (too-many-branches)
src/kskm/wksr/server.py:42:0: R0903: Too few public methods (1/2) (too-few-public-methods)
src/kskm/wksr/server.py:64:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
src/kskm/wksr/server.py:143:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.eddsa_utils -> kskm.common.public_key) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.data -> kskm.common.ecdsa_utils -> kskm.common.public_key -> kskm.common.eddsa_utils) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.ecdsa_utils -> kskm.common.public_key) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.data -> kskm.common.dnssec) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.data -> kskm.common.ecdsa_utils -> kskm.common.public_key) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.data -> kskm.common.ecdsa_utils) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.common.public_key -> kskm.common.rsa_utils) (cyclic-import)

src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.ksr -> kskm.ksr.load -> kskm.ksr.validate -> kskm.ksr.verify_policy) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.ksr -> kskm.ksr.load -> kskm.ksr.validate -> kskm.ksr.verify_bundles) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.ksr -> kskm.ksr.load -> kskm.ksr.validate -> kskm.ksr.verify_header) (cyclic-import)
src/kskm/wksr/server.py:1:0: R0401: Cyclic import (kskm.skr -> kskm.skr.output) (cyclic-import)