» rpkimancer

/ -pi -ke -a -mænsə /

“One who may be called upon to perform those secret rites and incantations necessary for the creation or interpretation of the mystical artifacts of the RPKI.”
rpkimancer started out as an attempt to solve two RPKI tooling problems:
Wanted a *simple* way to read RPKI signed objects for debugging and education purposes...

...without cracking out a **browser**

...without remembering **Byzantine openssl CLI options** and calculating byte-offsets
While working on RSC I-D module I wanted a way to check ASN.1 syntax in CI pipeline…

* Tried scripting the “Job Snjiders” method. **Attempt abandoned** within 30 mins.
problem #2

While working on RSC I-D module I wanted a way to check **ASN.1 syntax** in CI pipeline...

- Tried scripting the “Job Snjiders” method. **Attempt abandoned** within 30 mins.

- Tried re-purposing existing RP library code. Found that **no-one actually generates code from ASN.1** (*)
» **problem #2**

While working on RSC I-D module I wanted a way to *check ASN.1 syntax* in CI pipeline…

* Tried scripting the “Job Snjiders” method. **Attempt abandoned** within 30 mins.
* Tried re-purposing existing RP library code. Found that **no-one actually generates code from ASN.1** (*)
* Tried `asn1c1`. **Failed to compile** any of the PKIX/CMS dependencies.

(*)}
solution

* much searching...
solution

* much searching...

* found the *only* OSS ASN.1 compiler capable of dealing with the necessary syntax constructs: pycrate2...

...but needed quite a lot of wrapping to make it useful
solution

* much searching...

* found the *only* OSS ASN.1 compiler capable of dealing with the necessary syntax constructs: *pycrate*2...

...but needed quite a lot of wrapping to make it useful

* Began work on a Python library and CLI tool with the goal of being able to create and read arbitrary signed objects with only:
solution

- much searching...
- found the *only* OSS ASN.1 compiler capable of dealing with the necessary syntax constructs: `pycrate2`...
  ...but needed quite a lot of wrapping to make it useful
- Began work on a Python library and CLI tool with the goal of being able to create and read arbitrary signed objects with only:
  - `ASN.1 CONTENT-TYPE` definition
solution

- much searching...
- found the **only** OSS ASN.1 compiler capable of dealing with the necessary syntax constructs: pycrate2...
  
  ...but needed quite a lot of wrapping to make it useful

- Began work on a Python library and CLI tool with the goal of being able to create and read arbitrary signed objects with only:
  
  - ASN.1 CONTENT-TYPE definition
  - Python class with a simple constructor
status / features

* Runtime ASN.1 module compilation
status / features

- Runtime ASN.1 module compilation
- Import-time discovery of CONTENT-TYPE instance definitions

- Resource certificate implementations: TA (with TAL), CA and EE
- Standards-track signed objects in base package: MFT, ROA and GBR

- rpkincant CLI tool, demonstrates library usage:
  - rpkincant conjure: create a self-contained object tree
  - rpkincant perceive: decode and dump signed objects in various formats

- Plug-in architecture for adding signed object types, CLI extensions. Existing plugins for:
  - RSC rpkimancer-rsc
  - ASPA rpkimancer-aspa
» status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
* rpkincant CLI tool, demonstrates library usage:
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
* rpkincant CLI tool, demonstrates library usage:
  * rpkincant conjure: create a self-contained object tree
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
* rpkincant CLI tool, demonstrates library usage:
  * rpkincant conjure: create a self-contained object tree
  * rpkincant perceive: decode and dump signed objects in various formats
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
* rpkincant CLI tool, demonstrates library usage:
  * rpkincant conjure: create a self-contained object tree
  * rpkincant perceive: decode and dump signed objects in various formats
* Plug-in architecture for adding signed object types, CLI extensions. Existing plugins for:
status / features

* Runtime ASN.1 module compilation
* import-time discovery of CONTENT-TYPE instance definitions
* Resource certificate implementations: TA (with TAL), CA and EE
* Standards-track signed objects in base package: MFT, ROA and GBR
* rpkincant CLI tool, demonstrates library usage:
  * rpkincant conjure: create a self-contained object tree
  * rpkincant perceive: decode and dump signed objects in various formats
* Plug-in architecture for adding signed object types, CLI extensions. Existing plugins for:
  * RSC rpkimancer-rsc
status / features

* Runtime ASN.1 module compilation

* import-time discovery of CONTENT-TYPE instance definitions

* Resource certificate implementations: TA (with TAL), CA and EE

* Standards-track signed objects in base package: MFT, ROA and GBR

* rpkincant CLI tool, demonstrates library usage:
  * rpkincant conjure: create a self-contained object tree
  * rpkincant perceive: decode and dump signed objects in various formats

* Plug-in architecture for adding signed object types, CLI extensions. Existing plugins for:
  * RSC rpkimancer-rsc
  * ASPA rpkimancer-aspa
use cases

* Internet-draft module *validation* (CI)

At least two real bugs found so far:

- Manifest loop handling in FORT #55
- EE certificate commonName length in krill/rpki-rs #164
use cases

* Internet-draft module *validation* (CI)
* Work-in-progress object *prototyping* for interop testing

At least two real bugs found so far:

* Manifest loop handling in FORT #55
* EE certificate commonName length in krill/rpki-rs #164
» use cases

* Internet-draft module validation (CI)
* Work-in-progress object prototyping for interop testing
* RP/CA implementation bug search / confirmation

At least two real bugs found so far:
» use cases

* Internet-draft module *validation* (CI)
* Work-in-progress object *prototyping* for interop testing
* RP/CA implementation *bug search / confirmation*
* RP/CA *integration testing* and release qualification

At least two real bugs found so far:

1. Manifest loop handling in FORT #55
2. EE certificate commonName length in krill/rpki-rs #164
» use cases

* Internet-draft module validation (CI)
* Work-in-progress object prototyping for interop testing
* RP/CA implementation bug search / confirmation
* RP/CA integration testing and release qualification
* Ad-hoc object debugging

At least two real bugs found so far:

- Manifest loop handling in FORT #55
- EE certificate commonName length in krill/rpki-rs #164
» **use cases**

* Internet-draft module *validation* (CI)
* Work-in-progress object *prototyping* for interop testing
* RP/CA implementation *bug search / confirmation*
* RP/CA *integration testing* and release qualification
* Ad-hoc object *debugging*

At least two real bugs found so far:

* Manifest loop handling in FORT #55
» use cases

* Internet-draft module *validation* (CI)
* Work-in-progress object *prototyping* for interop testing
* RP/CA implementation *bug search* / confirmation
* RP/CA *integration testing* and release qualification
* Ad-hoc object *debugging*

At least two real bugs found so far:

* Manifest loop handling in FORT #55
* EE certificate *commonName* length in krill/rpki-rs #164
» **TODOs / ideas**

- BGPSec router certificates

help & suggestions (with/without PRs) welcome!
» **TODOs / ideas**

- BGPSec router certificates
- plugable directory layout definitions for use in local RP testing

help & suggestions (with/without PRs) welcome!
» TODOs / ideas

* BGPSec router certificates
* plugable directory layout definitions for use in local RP testing
* RRDP XML files generation

help & suggestions (with/without PRs) welcome!
» **TODOs / ideas**

* BGPSec router certificates
* plugable directory layout definitions for use in local RP testing
* RRDP XML files generation
* diff for signed objects

**help & suggestions (with/without PRs) welcome!**
» TODOs / ideas

* BGPSec router certificates
* plugable directory layout definitions for use in local RP testing
* RRDP XML files generation
* diff for signed objects
* plug-in template repo

help & suggestions (with/without PRs) welcome!
help wanted

* CA/RP implementors:
» help wanted

* CA/RP implementors:
  * preferred/recommended way to run against locally generated objects?

* Signed object I-D authors:
  * plug-ins for proposed object definitions
  * See rpkimancer-rsc for integration example with I-D git repo
help wanted

- CA/RP implementors:
  - preferred/recommended way to run against locally generated objects?
  - local cache directory layout plug-ins

Signed object I-D authors:
- plug-ins for proposed object definitions
See rpkimancer-rsc for integration example with I-D git repo
help wanted

- CA/RP implementors:
  - preferred/recommended way to run against locally generated objects?
  - local cache directory layout plug-ins
  - test harness, machine readable logs, etc

- Signed object I-D authors:
  - plug-ins for proposed object definitions
  - See rpkimancer-rsc for integration example with I-D git repo
help wanted

- CA/RP implementors:
  - preferred/recommended way to run against locally generated objects?
  - local cache directory layout plug-ins
  - test harness, machine readable logs, etc
- Signed object I-D authors:
help wanted

∗ CA/RP implementors:
  ∗ preferred/recommended way to run against locally generated objects?
  ∗ local cache directory layout plug-ins
  ∗ test harness, machine readable logs, etc

∗ Signed object I-D authors:
  ∗ plug-ins for proposed object definitions
help wanted

* CA/RP implementors:
  * preferred/recommended way to run against locally generated objects?
  * local cache directory layout plug-ins
  * test harness, machine readable logs, etc
* Signed object I-D authors:
  * plug-ins for proposed object definitions
  * See rpkimancer-rsc for integration example with I-D git repo
questions?