Dragon Research Lab rpki.net Summary

PARSONS, Inc.

16 August 2016

Contents

1 Introduction

The Dragon Research Lab rpki.net package is an open source implementation of both the Certification Authority (CA) and the Relying Party (RP) roles of the Resource Public Key Infrastructure [5] defined in the IETF SIDR working group. It includes also an implementation of the RPKI-RTR protocol [2].

2 Rpki.net

The rpki.net software package implements both the production (CA) and relying party (RP) functions of an RPKI environment.

Figure 1, below, shows the overview of the rpki.net architecture. See the rpki.net document "RPKI Tools Manual" [13] for more information.

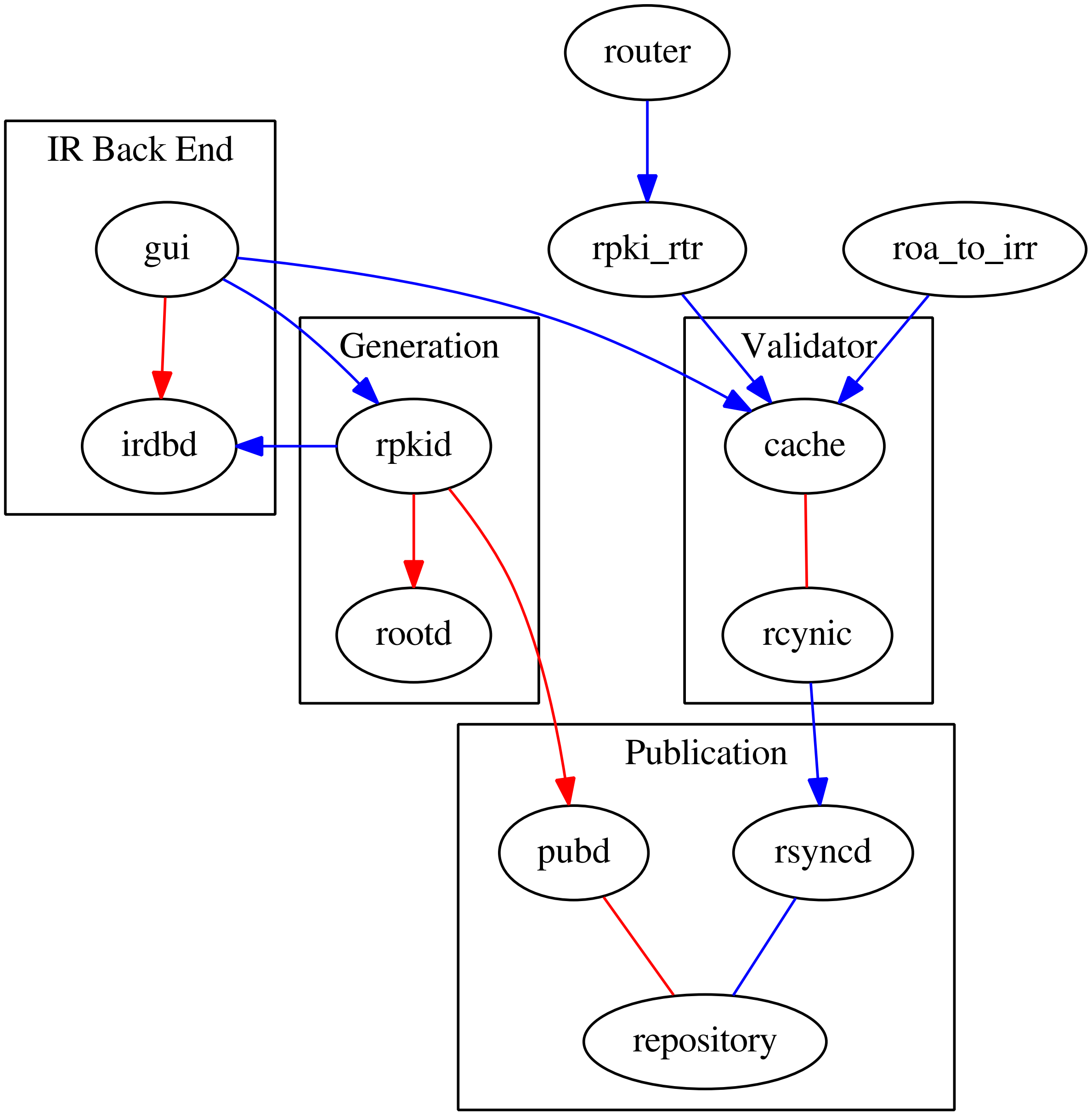

Figure 1: rpki.net

2.1 Relying Party Tools

Those who operate routers and want to use RPKI data to help secure them, will make use of the relying party tools. These tools implement the ”relying party” role of the RPKI system, that is, the entity which retrieves RPKI objects from repositories, validates them, and uses the result of that validation process as input to other processes, such as BGP security.

The RP main tools are rcynic and rtr-origin. rcynic is the primary validation tool. It does the actual work of RPKI validation: checking syntax, signatures, expiration times, and conformance to the profiles for RPKI objects. The other relying party programs take rcynic’s output as their input. rtr-origin is an implementation of the RPKI-RTR protocol, using rcynic’s output as its data source. rtr-origin includes the RPKI-RTR server, a test client, and a utility for examining the content of the database rtr-origin generates from the data supplied by rcynic.

2.2 Certificate Authority (CA) Tools

Those who control RPKI resources and need an engine that supports requesting certificates, issuing ROAs, or issuing certificates to other entities, will make use of the CA tools.

The RPKI CA engine is an implementation of the production-side tools for generating certificates, CRLs, ROAs, and other RPKI objects. The CA tools are implemented primarily in Python, with an extension module linked against an RFC-3779-enabled [6] version of the OpenSSL libraries to handle some of the low-level X.509 details.

(Since CAs are generally also relying parties (if only so that they can check the results of their own actions), network operators who operate a CA will likely use the relying party tools as well.)

The RPKI CA engine consists of a certificate issuance engine, a publication engine, and a registry backend. The separation of the certificate issuance engine from the publication engine allows for multiple certificates issuance engines to use the same publication engine. This also allows the publication to be publicly accessible, as is necessary, but the certificate issuance engine to be more protected.

The certificate issuance engine has a protocol (officially called the provisioning protocol [3], but known commonly as the up-down protocol) for making requests to its parent in the RPKI tree and responding to requests from its RPKI children. The publication engine has a publication protocol to speak to certificate issuance engines [14]. The certificate issuance engine and the registry backend communicate with a protocol called the “left-right” protocol. (Because it is expected that the CA backend processes and databases will be a local choice, there was no attempts to standardize the left-right protocol.)

The RPKI CA engine includes the following programs:

- **rpkid:** rpkid is the main RPKI certificate issuance engine daemon.
- **pubd:** pubd is the publication engine daemon.
- **IRBE:** irdbd, rpkic, and the Graphical User Interface (GUI) collectively make up the ”Internet registry back end” (IRBE) component of the system.

- **irdbd:** irdbd is a sample implementation of an Internet Registry Database (IRDB) daemon. rpkid calls into this to perform lookups via the left-right protocol.
- **rpkic:** rpkic is a command line interface to control rpkid and pubd.
- **GUI:** A web-based graphical interface to control rpkid and pubd(described in section 2.3).

The rpki.net package previously contained rootd as a separate daemon for handling the root of an RPKI certificate tree. Now the rootd daemon function is fully intergrated into the main daemon.

2.3 rpki.net GUI interface

In order to make the system easier to operate for end users, the need for a graphical interface to serve as a front end to the RPKI became apparent. A HTTP-based system was deemed the best way to achieve cross platform availability since every end user system has a graphical web browser available.

On the server side, the choice was made to make use of the Django [8] web framework, which allows for rapid prototyping of web applications. Django is open source, widely used, has excellent documentation, and has a very active community to draw on for help.

The initial design goal for the web interface was to provide a ”dashboard” that a user could view to get an overview of all RPKI resources under the user’s control. This dashboard presents information about: RPKI parents, children, repositories, ROAs. The dashboard view also attempts to draw the user’s attention to resources that are not covered by any ROA, in order to prevent inadvertent ”unknown” validation results.

The web interface also helps perform some validation when the user attempts to create a ROA, such as ensuring that the resources are actually under the user’s control.

Internally, the web interface is a front end to the IRDB and rpkid daemons. The web interface makes queries to rpkid to fetch the list of RPKI resources under a particular user’s control, and this information is stored in a database, from which the display in the web application is generated. As the user requests changes, such as creation of ROAs, the web interface makes changes to the IRDB, and notifies rpkid that it should contact irdbd to process those changes.

Since a resource holder is also interested in how other relying parties may view their origin authorizations, the web interface also allows the user to see the validity status of all routes that are covered by resources listed in their resource certificates. In order to perform this step, two additional sources of information need to used: the global RPKI for route authorizations, and RouteViews [12] for a snapshot of the current globally announced routes.

The snapshot of the global RPKI is generated by processing the output of the rcynic tool, described in section 2. This output consists of an Extensible Markup Language (XML) file, and a directory of resource certificates, ROAs, and Ghostbuster objects. For each object that was considered valid within the RPKI, the web interface software reads the repository objects and extracts information into a database that drives the web application display. Using this data, it is possible to perform the RPKI route origin validity checks that other relying parties will conduct, and give visual feedback to the user if there are route origins that are invalid or unknown.

The RouteViews project produces a dump of the global routing table every two hours. This dump is a merge of views from multiple sensors, so it gives a very good overview of the actual state of the global routing tables. A batch job runs in the background every two hours to fetch this data and import it into the database used by the web application. This allows the web interface to quickly find globally announced routes that are covered by RPKI resource certs and ROAs. Combined with the data from the global RPKI, the validity status of each route can be calculated and displayed for the user.

2.4 Software Dependencies

In keeping with the decision that the rpki.net software should be freely and openly available, the rpki.net software makes extensive use of several open software packages and libraries. This also allowed the implementers to focus on the features that are RPKI unique. The packages were chosen for their wide use, active user community for support and active implementation community for feature enhancement and bug fixes.

- **PostgreSQL** PostgreSQL is a widely used and popular open source database package. It was chosen for the default database support in the rpki.net package. Use of the Django ORM means other SQL databases can be substituted easily
- **Apache:** Apache is a commercial grade web server package used in the rpki.net user interface.
- **OpenSSL:** OpenSSL is a widely used implementation of many cryptographic features, including support for X.509 Public Key Infrastructure, XML, and many different cryptographic algorithms.
- **Python:** Python is a general purpose, high level programming language, recognized as being easy to learn, providing compact and efficiency in coding, and as a consequence providing ease of implementation and lower maintenance burden.
- **libxml:** libxml is a library providing an Application Programming Interface (API) for manipulating Extensible Markup Language (XML) data
- **YaML:** YAML Ain't Markup Language (YaML) is a human-readable data serialization format for all programming languages
- **Django:** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

3 Results: Rpki.net

The rpki.net code is the only fully compliant implementation of both the CA and relying party parts of the comprehensive solution, including the works in progress in the SIDR working group, such as router certificates.

The rpki.net package implements features discovered to provide performance benefits. The rpki.net package implements a hierarchical file structure, which proved to have such performance benefits for rsync synchronization that other implementations have adopted that approach. It also implements parallel download of RPKI data for syncrhonization improvements, a feature suggested by the RPSTIR project.

The rpki.net implementation of the CA and RP daemons (rpkid, pubd, etc) supports multiple entities, e.g., multiple CAs supported by one publication engine. The irdbd also supports multiple entities. That means that it is possible to extend the hosted model, such as the RIRs use to provide CA services for their members, to outsourcing the publication services for any CA. The rpki.net package also implements the publication protocol [14], which provides communication between the CA issuance engines and the publication service provider.

3.1 RPKI Supporting Protocols

The rpki.net package has served as the platform for investigating new protocols during design and specification stages. Consequently, rpki.net includes implementations of protocols that are still works in progress.

The “out of band” setup protocol [10] was created in early testing as an improvement in boostrapping the initial relationships between parent and child engine daemons and between publication protocol clients and servers. This protocol proved so useful to early adopters and in early testing that it was brought to the SIDR working group for standardization.

The rpki.net package contains implementations of both the server and the client roles of the RPKI-RTR protocol. The implementation now includes the enhancements to the rpki-rtr protocol that are works in progress in the SIDR working group. The enhancements provide for communication of router public keys to the router [11].

The rpki.net package provides an implementation of the publication protocol [14], which provides communication between the CA issuance engines and the publication service provider.

The rpki.net package supports generation and validation of certificates that use the RPKI Repository Delta Protocol (RRDP) retrieval protocol, an rsync alternative that is presently a work in progress in the SIDR working group [17].

3.2 Testing Scripts

The rpki.net implementation has the ability to represent an entire test network in a single easy to understand configuration file. It has the ability to write programs which generate those test configurations, which also proved to be useful, particularly for large scale testing.

The rpki.net implementation was the test subject for modeling BitTorrent as a potential RPKI transport protocol using the Starbed large scale emulation testbed [16]. A small number of very large scale configurations used there was a valuable output of that work[7], especially in terms of improving the core code. Initial testing was done with hand-constructed test objects. The large scale testing made possible by the test generation scripts helped identify implementation issues that would otherwise have been missed.

The rpki.net implementation also includes scripts for generating large numbers of router certificates, as would be needed in any testing of BGPSEC implementations. Another large test set generated a set of certificates and ROAs to represent all BGP announcements in a global full routing table, so as to test the RRDP implementation.

3.3 Balance Between Performance and Agile Coding

The rpki.net code uses both Python and C, with the choice made for performance reasons. Python is noted for its benefits in agile coding, and for its lower maintenance burden. This is particularly important when new protocol design choices are being explored - the faster development cycle is a help. Much of the rpki.net code originally written in C was converted to Python to adopt these advantages. However, there are places where C is needed for tasks that are performance sensitive. In rpki.net, C is used for cryptographic functions as well as the Abstract Syntax Notation One (ASN.1) and XML processing. C/Python API code was written to make the same functionalities available to the Python modules.

3.4 Flexible Database Support

The database use in the rpki.net GUI uses the Django Object-Relational Mapper (ORM). This makes it simpler to change the implementation to a different implementation of SQL, e.g., SQLite instead of PostgreSQL. This also permits use of Djang's "South" module to manage ugrades to the database schema, which minimizes impact on the users from such a change.

The rpki.net software is kept in a github repository, https://github.com/dragonresearch/rpki.net Binary packages are built nightly for a small set of popular unix operating systems.

3.5 Rpki.net GUI

The final design of the web interface was a result of feedback received from users over the course of the years. The numerous hands on workshops allowed network operators to become familiar with the software, and give valuable feedback on how the software could be improved to support day to day workflow.

An initial example of this feedback was that the original dashboard in the web interface was primarily focused around RPKI resources. This was a natural result of the developers being focused on the RPKI itself. However, operators are more interested in a route oriented view, since this is what they are dealing with. The result of this feedback was to alter the dashboard to give primary information about currently announced routes and their route origin validity status.

After RIPE started its RPKI pilot project which allowed resources holders to host their own private cryptograhic key material, and communicate via the up/down protocol, there were a number of users starting to use the software in a semi-production manner, and this provided additional useful feedback for development. One of the first results of this was a request to have the web interface support the initial setup process between RPKI parents and children. Previously, the user needed to use the command line tool to perform this step, but many users wanted the web interface to support this feature. The process involves uploading and downloading the XML files generated during the out of band setup procedure.

One ongoing issue that is not completely solved is the lag involved in updating the web interface display of what other relying party's view of the global RPKI might be after a user makes a change to their ROAs or children's resource certificates. The creation of the ROAs and resource certificates happens nearly instantaneously, but it takes an undetermined amount of time for the objects to be published to a repository, and then subsequently fetched by a relying party and proceed into origin validation results. This is the nature of a distributed, asynchronous system, but it can be confusing to the web interface user when the origin validation on their routes does not update as soon as the user submits the changes.

Another class of user that provided useful feedback was ISPs that were interested in providing RPKI "hosting" services, where customers would not hold their own private cryptographic key materials, but could use the software to manage their own resources. This is very similar to what RIRs such as RIPE provide for their members. The result of this request was that the software added support for separating the roles of resource holder and web interface user. Instead, the model should be that a particular web interface user may want to manage resources as several different resource holders, or that multiple users could manage a single resource holding entity. This more closely models the current situation with how the RIRs manage resources, where resources for an organization may be managed by several different departments.

As operators began to create ROAs in the RPKI system, they made errors typical of first use of a new technology. The production of one ROA will support the validity of one or more BGP routes. However, ROAs are strong statements of BGP route validity - not only are the BGP routes that match the ROA valid, but BGP routes that are not matched by any ROA can no longer be considered "not found". Because a statement has been made of what routes are authorized, other routes are considered unauthorized and "invalid"

In particular, if an ISP has subdelegated prefixes from its address space to a customer who is announcing it under their own AS, then a ROA that validates the ISP's announcement will make the customer's announcement appear invalid.

The web interface was expanded to retrieve a copy of RouteViews data and show the user what the impact would be for a ROA that was about to be created. This is a warning to the user to consider carefully what the impact of that ROA would be. Some but not all RIR implementations also provide this cautionary check.

4 Compliance with IETF SIDR Specifications

The SIDR published specifications of the RPKI are available at https://datatracker.ietf.org/wg/sidr/documents/. The rpki.net implementation is compliant with:

RFC 6480 An Infrastructure to Support Secure Internet Routing
RFC 6481 A Profile for Resource Certificate Repository Structure
RFC 6482 A Profile for Route Origin Authorizations (ROAs)
RFC 6483 Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)
RFC 6484 Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)
RFC 6485 The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)
RFC 6486 Manifests for the Resource Public Key Infrastructure (RPKI)
RFC 6487 A Profile for X.509 PKIX Resource Certificates
RFC 6488 Signed Object Template for the Resource Public Key Infrastructure (RPKI)
RFC 6490 Resource Public Key Infrastructure (RPKI) Trust Anchor Locator
RFC 6492 A Protocol for Provisioning Resource Certificates
RFC 6493 The Resource Public Key Infrastructure (RPKI) Ghostbusters Record
RFC 6810 The Resource Public Key Infrastructure (RPKI) to Router Protocol
RFC 6811 BGP Prefix Origin Validation
RFC 7730 Resource Public Key Infrastructure (RPKI) Trust Anchor Locator

The rpki.net software is also compliant with the following works in progress, SIDR work items that have not yet been published as final.

- draft-ietf-sidr-delta-protocol-03
  RPKI Repository Delta Protocol
  available at https://tools.ietf.org/id/draft-ietf-sidr-delta-protocol-03.txt
- draft-ietf-sidr-publication-08
  A Publication Protocol for the Resource Public Key Infrastructure (RPKI)
  available at https://tools.ietf.org/html/draft-ietf-sidr-publication-08.txt
- draft-ietf-sidr-rpki-oob-setup-04
  An Out-Of-Band Setup Protocol For RPKI Production Services
  available at https://tools.ietf.org/html/draft-ietf-sidr-rpki-oob-setup-04
- draft-ietf-sidr-rpki-rtr-rfc6810-bis-07
  The Resource Public Key Infrastructure (RPKI) to Router Protocol
  available at https://tools.ietf.org/html/draft-ietf-sidr-rpki-rtr-rfc6810-bis-07
- draft-ietf-sidr-rfc6485bis-05.txt
  The Profile for Algorithms and Key Sizes for use in the Resource Public Key Infrastructure
  available at https://tools.ietf.org/html/draft-ietf-sidr-rfc6485bis-05

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

API Application Programming Interface. 5, 10

AS Autonomous System
AS is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet. 10

ASN Autonomous System Number
ASN is the number associated to a Autonomous System (AS). 10

ASN.1 Abstract Syntax Notation One
ASN.1 is a language for describing structured information. 7, 10

BGP Border Gateway Protocol
This is the routing protocol used to transfer reachability and routing information between Autonomous Systems (AS's) on the internet. 10

BGPsec BGP Security
Additions to the BGP protocol to increase the security of the exchanged routing information. 10

BIRD The BIRD Internet Router Daemon is an open source routing software package [1]. 10

BPKI Business Public Key Infrastructure. 10

CA Certification Authority (x509)
An entity that issues digital certificates. The certificates are used to certify that the subject of the certificate owns the public key associated with the certificate. In the X.509 Public Key Infrastructure model, the CA is a trusted third party. That is, the owner of the certificate and the user, or relying party, of the certificate both trust the CA. 10

CMS Cryptographic Message Syntax
CMS is the IETF's standard for cryptographically protected messages. 10

ECDSA Elliptic Curve Digital Signature Algorithm (ECDSA) is a variant of the Digital Signature Algorithm (DSA) which operates on elliptic curve groups. The EC variant provides smaller key sizes for the same security level. This algorithm is used for signing and authenticating within the BGPSEC_path attribute in BGP UPDATE messages. 10

GUI Graphical User Interface. 3, 10

IANA Internet Assigned Numbers Authority
The IANA manages the DNS Root Zone, coordinates allocations from the global IP and AS number spaces, and serves as the central repository for protocol name and number registries used in many Internet protocols. 10

IETF Internet Engineering Task Force [4]
The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. 10

IRBE Internet Registry Back End. 3, 10

IRDB Internet Registry Data Base. 10

IRR Internet Routing Registry
The union of world-wide routing policy databases that use the Routing Policy Specification Language. 10

ISP Internet Service Provider. 10

ODBC Open Database Connectivity
ODBC is a standard programming language middleware API for accessing database management systems. 10

ORM Object-Relational Mapper
ORM is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. In Django, the data models are defined as Python classes. 10

RIR Regional Internet Registries
Regional Internet Registries (RIRs) manage, distribute, and register public Internet Number Resources within their respective regions. 10

ROA Route Origin Authorization
A ROA is a digitally signed object that provides a means of verifying that an IP address block holder has authorized an Autonomous System (AS) to originate routes to one or more prefixes within the address block. 10

RPKI Resource Public Key Infrastructure
A Public Key Infrastructure (PKI) used to support attestations about Internet Number Resource (INR) holdings. 10

RPSTIR Relying Party Security Technology for Internet Routing
Pronounced ”rip-stir”. Using the global Resource Public Key Infrastructure (RPKI), RPSTIR securely generates a list of authorized prefix-origin AS pairs. This list can be used by the RPKI-RTR protocol, enabling routers to detect false origin announcements due to errors by network operators. 10

RRDP RPKI Repository Delta Protocol. 6, 10

SIDR Secure Inter-Domain Routing (Working Group)
IETF Working Group that worked on creating BGPSEC [15]. 10

SQL Structured Query Language
SQL is a query language used for managing data held in a relational database system. 10

ssh Secure Shell
SSH is a protocol for secure remote login and other secure network services over an insecure networks. 10

SSL Secure Sockets Layer
SSL is a protocol that provides communication security over the Internet. 10

SVN Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and revision control system distributed as free software under the Apache License. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. 10

XML Extensible Markup Language
XML is a markup language defined by the W3C[18] that defines a set of rules for encoding documents in a format that is human and machine readable. 4, 5, 7, 10

YaML YAML Ain't Markup Language
YaML is a human-readable data serialization format. 5, 10

GLOSSARY OF TERMINOLOGY

Apache Apache is a commercial grade web server package. 5, 10

Django Django is a free and open source web application framework written in Python, that encourages rapid development and clean, pragmatic design. 5, 10

ghostbuster An RPKI signed object which contains contact information for a person responsible for the RPKI repository in which the object appears. 10

irdbd A sample implementation of an IR database daemon. 3, 10

left-right The left-right protocol is two separate client/server protocols over separate channels between the RPKI engine and the IR back end (IRBE). The IRBE is the client for one of the subprotocols, the RPKI engine is the client for the other. 10

MySQL a widely used and popular open source database package. 10

OpenSSL a widely used, open source implementation of many cryptographic features, including support for X.509 Public Key Infrastructure, XML, and many different cryptographic algorithms. 5, 10

PostgreSQL a powerful, open source object-relational database system, whose SQL implementation strongly conforms to the ANSI-SQL:2008 standard. 5, 10

prefix a contiguous block of Internet addresses, called a prefix because all the addresses share the same initial bit pattern. 10

pubd The publication engine daemon. 3, 10

rcynic The primary validation tool in the rpki.net package. 2, 10

Relying Party The entity which retrieves RPKI objects from repositories, validates them, and uses the result of that validation process as input to other processes, such as BGP security. 10

rootd A separate daemon for handling the root of an RPKI certificate tree. 3, 10

RouteViews Route Views is a project founded by Advanced Network Technology Center at the University of Oregon to allow Internet users to view global BGP routing information from the perspective of other locations around the internet. Originally created to help Internet Service Providers determine how their network prefixes were viewed by others in order to debug and optimise access to their network, Route Views is now used for a range of other purposes such as academic research. 10

rpki-rtr A protocol to deliver validated prefix origin data to routers. Described in RFC 6810 [2]. 10

rpki.net rpki.net is a project and website. The project provides a free, BSD License, open source, complete system for the Internet Registry or ISP. It includes separate components which may be combined to suit your needs. 10

rpkic A command line interface to control rpkid and pubd. 3, 10

rpkid The main RPKI certificate issuance daemon. 3, 10

rsync A file synchronization and file transfer program for Unix-like systems that minimizes network data transfer by using a form of delta encoding called the rsync algorithm. 10

rtr-origin rtr-origin is an implementation of the rpki-rtr protocol, including the rpki-rtr server, a test client and a utility for examining the content of the database rtr-origin generates. 2, 10

RTRlib RPKI RTR Client C Library
The RTRlib is an open-source C implementation of the RPKI/Router Protocol client [9]. 10

subversion Apache Subversion (often abbreviated SVN, after the command name svn) is a software versioning and revision control system distributed as free software under the Apache License. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. 10

up-down A RPKI certificate provisioning protocol which is expressed as a simple request/response interaction, where the client passes a request to the server, and the server generates a corresponding response. Described in RFC 6492[3]. 10

X.509 an ITU-T standard for public key infrastructure (PKI) certificates and certificate revocation lists. 10

References

[1] BIRD Internet Routing Daemon. http://bird.network.cz.

[2] R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810, January 2013.

[3] G. Huston, R. Loomans, B. Ellacott, and R. Austein. A Protocol for Provisioning Resource Certificates. RFC 6492, February 2012.

[4] IETF. The Internet Engineering Task Force. http://www.ietf.org.

[5] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, February 2012.

[6] C. Lynn, S. Kent, and K. Seo. X.509 Extensions for IP Addresses and AS Identifiers. RFC 3779, June 2004.

[7] Debbie Perouli, Olaf Maennel, Iain Phillips, Sonia Fahmy, Randy Bush, and Rob Austein. An Experimental Framework for BGP Security Evaluation. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 55(4):147–154, 2013.

[8] The Django Project. The django web framework. www.djangoproject.com.

[9] Joint project of the INET research group at the Hamburg University of Applied Sciences and the CST research group at Freie Universitt Berlin. Rtrlib - the rpki rtr client c library, June 2014. rpki.realmv6.org.

[10] R. Austein (editor). An Out-Of-Band Setup Protocol For RPKI Production Services. IETF SIDR Working Group Internet Draft, April 2016. `https://tools.ietf.org/html/draft-ietf-sidr-rpki-oob-setup`.

[11] R. Bush and R. Austein (editors). The Resource Public Key Infrastructure (RPKI) to Router Protocol. IETF SIDR Working Group Internet Draft, March 2016. `https://tools.ietf.org/html/draft-ietf-sidr-rpki-rtr-rfc6810-bis`.

[12] University of oregon route views archive project. `http://routeviews.org`.

[13] rpki.net. Rpki tools manual, October 2014. `http://subvert-rpki.hactrn.net/trunk/doc/manual.pdf`.

[14] S. Weiler, A. Sonalker, and R. Austein (editors). A Publication Protocol for the Resource Public Key Infrastructure (RPKI). IETF SIDR Working Group Internet Draft, March 2016. `https://tools.ietf.org/html/draft-ietf-sidr-publication`.

[15] SIDR. Secure Inter-Domain Routing, IETF Working Group. `https://datatracker.ietf.org/wg/sidr/`.

[16] Starbed large scale emulation testbed. `http://starbed.nict.go.jp/en/`.

[17] T. Bruijnzeels and O. Muravskiy and B. Weber and R. Austein. RPKI Repository Delta Protocol. IETF SIDR Working Group Internet Draft, July 2016. `https://tools.ietf.org/html/draft-ietf-sidr-delta-protocol`.

[18] W3C. World Wide Web Consortium (W3C). `http://www.w3.org/`.