Resource PKI (RPKI): Design and Operation of the Infrastructure

Project : GigaPort3
Project Year : 2011
Project Manager : Jac Kloots
Author(s) : Jac Kloots, François Kooman, and Benno Overeinder
Completion Date : 15-07-2011
Version : 1.0

Summary
This document describes the in-depth technologies and works of routing security. Focus is on global operations of RPKI and the SURFnet specific configuration as used in the Pilot infrastructure.
**Colophon**

Programme line: Gigaport3  
Part: FIP-2011  
Activity: FIP-11-02 Routing Security  
Deliverable: D02-1  
Access rights: public  
External party: NLNet Labs

This project was made possible by the support of SURF, the collaborative organisation for higher education institutes and research institutes aimed at breakthrough innovations in ICT. More information on SURF is available on the website www.surf.nl.
6 Matters one should know about Resource PKI: Design and Operation of the Infrastructure.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>In the BGP world all neighbour relations are based on good trust. Routes announced to each other are believed to be routes you are supposed to route. Until recently there was no way to prove the announced routes are allowed to originated from a particular network. With RPKI this changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is it?</td>
<td>RPKI is a mechanism that makes it possible to use digital certificates on network resources to proof a certain network prefix belongs to a certain network.</td>
</tr>
<tr>
<td>Whom is it for?</td>
<td>Networks who use BGP for their exchange of network prefixes</td>
</tr>
<tr>
<td>How does it work?</td>
<td>Using digital certificates in a Public Key Infrastructure to sign network resources a hierarchical trust tree is build. The digital certificate is used to sign network resources and allows network operators to check the origin of the announced prefixes. Routing policies than can be applied according to the result of the origin check</td>
</tr>
<tr>
<td>What can one do with it?</td>
<td>Based on the results of the validity of the announced prefixes network operators can apply routing policies. Valid routes can be used, invalid or unknown prefixes can be discarded or applied with a lower preference.</td>
</tr>
<tr>
<td>More information</td>
<td><a href="mailto:Jac.Kloots@SURFnet.nl">Jac.Kloots@SURFnet.nl</a>, Franç<a href="mailto:ois.Kooman@SURFnet.nl">ois.Kooman@SURFnet.nl</a>, <a href="mailto:Benno@nlnetlabs.nl">Benno@nlnetlabs.nl</a></td>
</tr>
</tbody>
</table>
## Index

<table>
<thead>
<tr>
<th>Resource PKI (RPKI): Design and Operation of the Infrastructure</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary.............................................................................</td>
<td>1</td>
</tr>
<tr>
<td>Colophon..........................................................................</td>
<td>2</td>
</tr>
<tr>
<td>Obligated to fill in unless there is approval from the Project Board to decay this fact sheet</td>
<td>3</td>
</tr>
<tr>
<td>6 Matters one should know about Resource PKI: Design and Operation of the Infrastructure</td>
<td>3</td>
</tr>
<tr>
<td>Introduction....................................................................</td>
<td>5</td>
</tr>
<tr>
<td>RPKI: Design and Its Protocols.....................................</td>
<td>5</td>
</tr>
<tr>
<td>Resource PKI: Resource Certification and PKI..................</td>
<td>5</td>
</tr>
<tr>
<td>Route Origin Authorizations..........................................</td>
<td>6</td>
</tr>
<tr>
<td>RPKI in Practice.........................................................</td>
<td>7</td>
</tr>
<tr>
<td>The RPKI rpki.net Implementation and Operation..................</td>
<td>7</td>
</tr>
<tr>
<td>Architecture of Software..............................................</td>
<td>7</td>
</tr>
<tr>
<td>RPKI Engine and Publication Daemon..................................</td>
<td>8</td>
</tr>
<tr>
<td>rcynic.............................................................................</td>
<td>8</td>
</tr>
<tr>
<td>RPKI in the RIPE Region...............................................</td>
<td>8</td>
</tr>
<tr>
<td>RPKI.net Installation and Configuration Instructions...........</td>
<td>9</td>
</tr>
<tr>
<td>Compile and Install......................................................</td>
<td>9</td>
</tr>
<tr>
<td>SURFnet Configuration...................................................</td>
<td>9</td>
</tr>
<tr>
<td>Packet filter / firewall ACL...........................................</td>
<td>10</td>
</tr>
<tr>
<td>Running rcynic.............................................................</td>
<td>10</td>
</tr>
<tr>
<td>Summary...........................................................................</td>
<td>10</td>
</tr>
<tr>
<td>References.......................................................................</td>
<td>11</td>
</tr>
<tr>
<td>Appendix A.......................................................................</td>
<td>12</td>
</tr>
</tbody>
</table>
Introduction

The Internet routing infrastructure was designed more than a decade ago. The "by default" routing protocol on the Internet is BGP-4 that was defined within the IETF in the period 1989–1995 (and a refinement of BGP-4 in 2006). At that time, security was not such an issue as it is today, and between the relative few participants/networks there was quite some level of trust. Understandably, the BGP protocol relied on this trust between the partners in the routing system, and no security considerations or requirements were formulated. And although BGP did not have security mechanisms, there are many ways for networks to secure their part of the routing system by mechanisms like prefix filtering [1,2] or network ingress filtering [3].

The various security mechanisms, which are complementary and provide partial protection against security threats, are used in different configurations. The result is a fairly secure, but not complete secure routing infrastructure. As the latter might be impossible to realize in all aspects in a distributed system like the Internet, there is certainly room for improvement. In particular high-profile incidents have raised the awareness of the community. Well-known address prefix hijacks are the YouTube-Pakistan incident (in 2008) and the fifteen minutes that China Telecom announced 40000 address prefixes of networks it did not own (in 2010). Other attacks like traffic deviation or man-in-the-middle attacks, have been demonstrated to be feasible on the real Internet routing fabric by Pilosov and Kapela at DEFCON 16.

One of the first major contributions to secure the routing system was the secure BGP (sBGP) proposal [4]. However, the required computational performance seemed to be beyond the capabilities of current routers used in the routing fabric. An alternative approach, soBGP [5], tried to find a balance between performance requirements and relaxed routing path validation. Other proposals to the routing security problem are presented in surveys by Huston et al. [1] and Butler et al. [2]. The IETF process approached routing security as two separate problems: route origin validation and route path validation. For route origin validation the Resource PKI (RPKI) infrastructure was defined. This work is in its final phase in the IETF SIDR working group. Recently, the SIDR working group also initiated new work on path validation, called BGPsec, which itself relies on the widespread deployment of RPKI.

The purpose of this document is to provide a technical overview of the RPKI architecture and its protocols. Next the rpki.net design and implementation of the RPKI architecture is presented, which is currently the single available implementation. For SURFnet, the interaction of rpki.net implementation with the RIPE certification and signing services are relevant, and its current state is described. The document concludes with some practical guidelines for the installation and configuration of the rpki.net software.

RPKI: Design and Its Protocols

For the validation of an address prefix route origin, an essential prerequisite is an authoritative data source of IP address block delegation from IANA, through the Regional Internet Registries (RIRs) and Local Internet Registries (LIRs), to the end-user ISPs. Besides this IP address block delegation that reflects the "right-of-use" of the address block by an entity (organization/ISP/...), an association of the address block with an originating Autonomous System (AS) must be reliably stated. Currently, such an authoritative data source is not available, even though much effort has been put in maintaining IRR databases, the content is incomplete and includes stale information.

The basic idea to construct an authoritative data source is to setup certificate authorities (CAs) to issue digital certificates for resource delegation and signed authority documents stating the AS is authorized to originate an address prefix. A resource public key infrastructure (RPKI) allows relying parties to validate assertions about IP address and AS numbers. In the next sections, an overview of the CAs and RPKI architecture is presented, but a more in-depth presentation is given by Huston et al. [6].

Resource PKI: Resource Certification and PKI

A resource certificate is an X.509 certificate that conforms the PKIX profile (RFC 5280) with a mandatory certificate extension that lists a collection of IP resources (IPv4 addresses, IPv6

1 http://www.rpki.net/
addresses, and AS numbers) (RFC 3779). The resource certificates reflect the resource allocation by the parent to the entity that is granted the "right-of-use". The certificate is with cryptographic singing operations testable for validity. In a hierarchy from, e.g., IANA, RIR, NIR, LIR, to ISP, each individual allocation can be tested by validating the resource certificates issued by the parent downwards. As an example, if a LIR receives an address block from a RIR, then only that RIR can issue a resource certificate for the LIR that includes its public key and the allocated resources. Anything the LIR signs using its private key can be verified via the RIR’s issued certificate. Of course, allocated resources by the RIR are reallocated in parts of possibly different sizes by the LIR to subsequently lower-level entities in the allocation hierarchy. The certificates issued by the LIR reflects this reallocation downward the hierarchy, see also Figure 1. Resource certificates are not necessarily long-lived: they must reflect changes in resource allocations.

Instead of constructing further allocations of resources, any holder of a resource itself might wish to use the resources. The holder of a resource can issue an End Entity (EE) certificate that is used to sign an attestation of a particular use of a resource. For example, if a resource holder wants to authorize an AS to origin a particular address block, it generates an EE certificate with the set of addresses that are intended to be originated by the AS. The use of the EE certificates will be explained in the next section.

The architecture of the infrastructure is described in the SIDR working group document draft-ietf-sidr-arch [7].

**Route Origin Authorizations**

The EE certificates are used to validate objects in the RPKI. The objects are signed with the EE certificate’s private key, and relying parties can validate the authority by checking whether the digital signature is correct. For route origin validations so-called route origin authorization (ROA) objects are provided in the RPKI [8]: a signed object that contains a list of address prefixes, an AS number, and the relevant EE certificate.

To show how EE certificates and ROAs are used, we continue the example where a resource holder wants to authorize an AS to originate an address prefix. Besides issuing an EE certificate for the resources, the address block holder also creates a ROA object that authorizes an AS to originate one or more particular address prefixes into the routing system. A relying party that wants to check if the origination AS in the routing system is legitimate, can validate the ROA:

- check digital signature in the ROA to make sure the object is not tampered with;
- resources in EE certificate contains the address prefixes in the ROA: one cannot place prefixes in the ROA that are not in the EE;
- check EE certificate is valid in the context of the RPKI, by verifying there is a chain of CA certificates that link a trust anchor to the EE certificate and the allocation of resources throughout the chain.
**RPKI in Practice**

Once RPKI is deployed, it is complementary to the BGP routing system. The BGP protocol is unchanged, only RPKI validation results can be used in the definition of policies. Different policies can define different actions according to the result of a validation step: strict, secure, preference, etc. A design trade-off is to which extent the RPKI functionality is integrated with the router hardware. Is the validation of an announcement performed inside the router, or is the cryptographic processing off-loaded to an external device and only the validation status has to be exchanged with the router.

In the future it is foreseen that IANA will act as the single trust anchor, in conformity with its role as root origin in the allocation of IP addresses and AS numbers. In the current deployment of RPKI, the five RIRs act as trust anchors. A testbed based on the rpki.net implementation is available for interested LIRs and ISPs to participate and experiment with RPKI in their organization. Cisco and Juniper also participate or work closely together with the rpki.net developers to test the protocols, the implementation, and the interoperability between the components in the infrastructure. Both Cisco and Juniper have software images for their (selection of) routers to interface with validated IP prefix caches and allow routing policy definitions based on the states of the validation step.

Given the dynamic nature of routing, EE certificates are typically not long-lived. To ensure that relying parties have up-to-date information in their validation caches, regular sweeps over the RPKI publication points are necessary. To relieve the burden to synchronize with potentially in the order of 10,000 participants, intermediaries aggregate and publish RPKI data. This allows relying parties to synchronize against a smaller set of RPKI publication points.

**The RPKI rpki.net Implementation and Operation**

This collection of Python modules implements a prototype of the RPKI protocol. The RPKI Engine is an implementation of the production-side tools for generating certificates, CRLs, and ROAs. The tool for the relying party to fetch and validate RPKI certificates is called rcynic: the cynical rsync.

**Architecture of Software**

The rpki.net software has three distinct components:

1. The modules rpkid, pubd, irdbd that take care of the signing (up/down protocol), publishing the signed resources, and LIR back end services (left/right protocol), see also Figure 2;
2. The modules rcynic, which creates a resource cache and validates the resources, and rtr-origin, which handles the communication between rcynic cache and router;
3. The router module that is integrated (natively) in the router to base routing decisions on the data retrieved from the cache.

![Figure 2. The rpki.net software architecture.](image-url)
**RPKI Engine and Publication Daemon**

The RPKI engine takes care of the up/down protocol. This involves quite a lot of different roles that need to be fulfilled. For instance:

1. create and operate a CA hierarchy. Either as root (operated currently by RIRs while optimally would be operated by IANA), as intermediate CA (have parents and children) or as leaf (have only parent(s));
2. be able to create EE certificates (sign resources);
3. publish those resources to a publication endpoint.

All of these components can also still be decentralized at different locations.

**rcynic**

Ideally, rcynic would point to the trust anchor of the system to retrieve all ROAs/resource certificates and store them in the local cache for validation and conveying them to (local) routers. Currently there are multiple trust anchors in use, every RIR has one, and some additional ones exist for testing. Figure 3 shows an rcynic example with one trust anchor at IANA.

![Figure 3. The rcynic cache gatherer.](image-url)

**RPKI in the RIPE Region**

In the RIPE LIR portal one can enable RPKI certification. RIPE will create a CA certificate for your resources and manage everything themselves. There is no way to delegate resource certification to child networks. The procedure is described on the RIPE website at [http://www.ripe.net/lir-services/resource-management/certification/ripe-ncc-certification-howto/ripe-ncc-certification-howto](http://www.ripe.net/lir-services/resource-management/certification/ripe-ncc-certification-howto/ripe-ncc-certification-howto). This is currently of limited use as the RIPE trust anchor certification cannot function with the RPKI software stack, due to modifications in the requirements for trust-anchor certificates, which were not implemented by RIPE. It is unknown when this issue will be resolved.
RPKI.net Installation and Configuration Instructions

The SURFnet setup currently operates in the rpki.net testbed as a leaf, while publishing as well. So the PKI infrastructure has only one leaf (surrogate RIPE operated by Rob Austein), signs some resources and has a publication point. All of this is operated on the domain rpki.surfnet.nl.

SURFnet signs two resources:
1. 195.169.0.0/16
2. 2001:610::/32

The software was compiled and is running on Debian Linux (testing).

Compile and Install

The compilation is done “in tree“ and also run from within the tree (without installing it). This is currently the recommended way [http://www.hactrn.net/rpki-dox/Installation.html].

The dependencies for the software are listed in the Installation instructions as well.

Downloading and compiling the software is straightforward:

$ svn co http://subvert-rpki.hactrn.net/
$ cd subvert-rpki.hactrn.net
$ ./configure && make

The rsync daemon is a required package:

# apt-get install rsync

SURFnet Configuration

The following commands will set up rpkid and pubd with the configuration we are running at SURFnet. The configuration is provided as a patch on the example config. See appendix A for the patch:

$ cp -r examples data.SURFnet
$ cd data.SURFnet
$ rm -rf .svn
$ patch -p1 < /path/to/rpkid_configuration.patch
$ mkdir publication
$ python ..../rpki-sql-setup.py
$ python ../myrpki.py initialize
$ cat entitydb/identity.xml
    ### send the identity.xml to the parent
# cp rsyncd.conf /etc/rsyncd.conf
# /etc/init.d/rsync restart

The daemons must be started:

$ python ../rpki-start-servers.py

Now wait for a response from the parent. Data from the responding parent is stored in the parent.xml file.

$ python ../myrpki.py configure_parent parent.xml
$ python ../myrpki.py configure_publication_client entitydb/repositories/ripe.xml
$ python ../myrpki.py configure_repository entitydb/pubclients/SURFnet.xml

The patch mentioned above (rpkid_configuration.patch) is in Appendix A. this makes clear what exactly was changed in relation to the in the rpki.net package included example files:

Special attention needs to be paid to the location of OpenSSL (in rpkid.conf) and the publication path in the rsync.conf file. The locations in Appendix A. are true for the SURFnet setup at rpki.surfnet.nl.
Packet filter / firewall ACL

The following TCP ports should be reachable from the outside world:

1. tcp/4402 for rpkid
2. tcp/4404 for pubd
3. tcp/873 for rsync

Running rcynic

The following section describes the configuration of rcynic. It configures rcynic both with the “official” RIPE trust anchor and Rob Austein’s testbed trust-anchor, which was used in the previous section of the document.

##
## This is the list of commands needs to get rpki validation cache populated
## and make a service available for a router to connect to. This is NOT
## production quality!
##
$ cd cd subvert-rpki.hactrn.net
$ cd rcynic/
$ cat << EOF >> rcynic.conf
[rcynic]
trust-anchor.0 = trust-anchors/ripe-ncc-ta.cer
trust-anchor.1 = trust-anchors/testbed-ripe.cer
EOF
$ mkdir trust-anchors
$ rsync rsync://rpki.ripe.net/ta/ripe-ncc-ta.cer trust-anchors/
$ rsync rsync://ripe.rpki.net/rpki/ripe/root.cer trust-anchors/testbed-ripe.cer

##
## MAKE SURE YOU VERIFY THE TRUST ANCHOR!!!
## http://tools.ietf.org/wg/sidr/draft-ietf-sidr-ta/
##
##
$ ./rcynic -c rcynic.conf -j 0          # "-j 0" sets start up delay jitter to 0

##
## This command should be placed in a cronjob, but then without the "-j 0"!
##
##
$ cd..
$ cd rtr-origin
$ python ./rtr-origin.py --cronjob ../rcynic  # also in a cronjob

$ echo "cd $PWD" > my_server.sh
$ echo "exec /usr/bin/python rtr-origin.py --server" >> my_server.sh
$ chmod +x my_server.sh

$ # "apt-get install netpipes" for the "faucet" tool
$ faucet 5001 --verbose --in --out ./my_server.sh

##
## You can add --daemon and remove --verbose if it should run in the background
##

Summary

The RPKI system has been designed to provide an authoritative data source for IP address block and AS number resources. Within the RPKI framework, a relying party can assess the validity of routing information. In particular, route origin validation to detect IP prefix hijacks, by mistake or on purpose, can be realized using the RPKI infrastructure. It is up to the network operator to define
local policies and which actions are taken on valid/invalid/unknown status of a route announcement.

In the past year, a testbed for RPKI has been constructed. The testbed is based on the rpki.net software package and allows network operators to install, configure, and experiment with RPKI. Besides the rpki.net software, two major router vendors, Cisco and Juniper, have made available software images for some of their routers such that validation can be used in routing policy definitions. SURFnet participates in this testbed and has an experimental setup with a server running the RPKI engine software as well as the rcynic gatherer that fills the validated cache. A Juniper router interoperates with the RPKI server to validate route announcements.

A follow-up document will describe the organizational aspects of deploying RPKI in a network. For example, the management of the resources and their associated certificates, which services are provided to customers, e.g., hosted certification services, or choosing intermediaries for publishing the certificates and ROAs.

References

Appendix A.

rpkid_configuration.patch

Only in examples: .svn
diff -ur examples/asns.csv data.SURFnet/asns.csv
--- examples/asns.csv   2011-07-12 15:53:33.000000000 +0200
+++ data.SURFnet/asns.csv       2011-07-12 15:53:29.000000000 +0200
@@ -2,4 +2,4 @@
  # Syntax: <child_handle> <asn>
  #
-Alice  64533
+Alice 64533

diff -ur examples/prefixes.csv data.SURFnet/prefixes.csv
--- examples/prefixes.csv       2011-07-12 15:53:33.000000000 +0200
+++ data.SURFnet/prefixes.csv   2011-07-12 15:53:29.000000000 +0200
@@ -3,6 +3,6 @@
  # Syntax: <child_handle> <prefix>/<length>
  #     or: <child_handle> <min>-<max>
  #
-Alice  192.0.2.0/27
-Bob    192.0.2.44-192.0.2.100
-Bob    10.0.0.0/8
+Alice 192.0.2.0/27
+#Alice 192.0.2.0/27
+#Bob   192.0.2.44-192.0.2.100
+#Bob   10.0.0.0/8

diff -ur examples/roas.csv data.SURFnet/roas.csv
--- examples/roas.csv   2011-07-12 15:53:33.000000000 +0200
+++ data.SURFnet/roas.csv       2011-07-12 15:53:29.000000000 +0200
@@ -2,4 +2,5 @@
  # Syntax: <prefix>/<length>-<maxlength> <asn> <group>
  #
-10.3.0.44/32   666     Mom
+-10.3.0.44/32   666     Mom
+195.169.0.0/16 1103 SURFnet

diff -ur examples/rpki.conf data.SURFnet/rpki.conf
--- examples/rpki.conf  2011-07-12 15:53:33.000000000 +0200
+++ data.SURFnet/rpki.conf      2011-07-12 15:53:29.000000000 +0200
@@ -18,7 +18,7 @@
  # digits, hyphen, underscore -- no whitespace, non-ASCII characters,
  # or other punctuation). You need to set this.
-
-handle                         = Me
+handle                         = SURFnet

  # Names of various files and directories. Don't change these without
  # a good reason.
@@ -41,7 +41,7 @@
  # unless you really know what you are doing. Port numbers can be any
  # legal TCP port number that you're not using for something else.
-
-rpki_server_host              = rpki.example.org
+rpki_server_host              = rpki.surfnet.nl
 rpki_server_port              = 4004
 irdbd_server_host             = localhost
 irdbd_server_port             = 4003
@@ -51,21 +51,21 @@
  # number of publication sites that relying parties need to check, so
  # don't enable this unless you have a good reason.
-
-run_pubd                       = false
+run_pubd                       = true

  # DNS hostname and server port number for pubd, if you're running it.
  # Hostname has to be a publicly reachable name to be useful, port can
  # be any legal TCP port number that you're not using for something
  # else.
-
-pubd_server_host               = pubd.example.org
+pubd_server_host               = rpki.surfnet.nl
 pubd_server_port               = 4002
# Contact information to include in offers of repository service.
# This only matters when we're running pubd. This should be a human
# readable string, perhaps containing an email address or URL.
-pubd_contact_info = repo-man@rpki.example.org
+pubd_contact_info = netmaster@surfnet.nl

# Whether you want to run your very own copy of rootd. Don't enable
# this unless you really know what you're doing.
@@ -130,7 +130,7 @@
 # no system copy installed, or if the system copy doesn't support CMS.
 # The copy of openssl built by this package should suffice.
-openssl = openssl
+openssl = /home/fkooman/subvert-rpki.hactrn.net/openssl/openssl-1.0.0d/apps/openssl

 # End of [myrpki] section

diff -ur examples/rsyncd.conf data.SURFnet/rsyncd.conf
--- examples/rsyncd.conf 2011-07-12 15:53:33.000000000 +0200
+++ data.SURFnet/rsyncd.conf 2011-07-12 15:53:29.000000000 +0200
@@ -35,11 +35,11 @@
    path = /some/where/publication
+    path = /home/fkooman/subvert-rpki.hactrn.net/rpkid/data.SURFnet/publication
    comment = RPKI Testbed

[rpki]
    use chroot = no
    read only = yes
-transfer logging = yes
+    transfer logging = yes
    path = /home/fkooman/subvert-rpki.hactrn.net/rpkid/data.SURFnet/publication
    comment = RPKI Testbed