BGP Origin Validation

ISP Workshops

Last updated 7th December 2020

Acknowledgements

- This material was built from contributions by Randy Bush, Mark Tinka, Tashi Phuntsho and others

- Use of these materials is encouraged as long as the source is fully acknowledged and this notice remains in place

- Bug fixes and improvements are welcomed
  - Please email *workshop (at) bgp4all.com*

Philip Smith

Validating BGP Route Announcements

- How do we know that an AS is permitted to originate the prefix it is originating?
- Implicit trust?
- Because the Internet Routing Registry says so?
  - The Internet Routing Registry (IRR) only documents routing policy
  - And has a large amount of outdated/incorrect information
- Is there something else?
  - Yes: Route Origin Authorisation

RPKI

- RPKI – Resource Public Key Infrastructure, the Certificate Infrastructure for origin and path validation
  - We need to be able to authoritatively prove who owns an IP prefix and which AS(s) may announce it
  - Prefix ownership follows the allocation hierarchy (IANA, RIRs, ISPs, etc)
  - Origin Validation
    - Using the RPKI to detect and prevent mis-originations of someone else’s prefixes (early 2012)
  - AS-Path Validation, in other words, BGPsec
    - Prevent Attacks on BGP (future work)

BGP – Why Origin Validation?

- Prevent YouTube accident & Far Worse
- Prevents most accidental announcements
- Does not prevent malicious path attacks
- That requires ‘Path Validation’ and locking the data plane to the control plane, the third step, BGPsec

What is RPKI?

- Resource Public Key Infrastructure (RPKI)
  - A security framework for verifying the association between resource holder and their Internet resources
  - Created to address the issues discussed in RFC 4593 “Generic Threats to Routing Protocols” (Oct 2006)
- Helps to secure Internet routing by validating routes
  - Proof that prefix announcements are coming from the legitimate holder of the resource
  - RFC 6480 – An Infrastructure to Support Secure Internet Routing (Feb 2012)
  - RFC 7115 – Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)

Benefits of RPKI for Routing

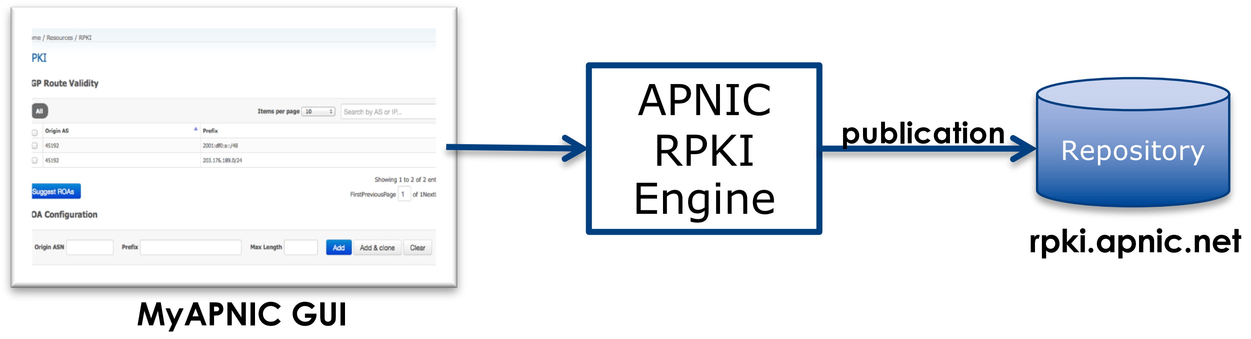
- Prevents route hijacking
  - A prefix originated by an AS without authorisation
  - Reason: malicious intent
- Prevents mis-origination
  - A prefix that is mistakenly originated by an AS which does not own it
  - Also route leakage
  - Reason: configuration mistake / fat finger

BGP Security (BGPsec)

- Extension to BGP that provides improved security for BGP routing
- Being worked on by the SIDR Working Group at IETF
- Implemented via a new optional non-transitive BGP attribute that contains a digital signature
- Two components:
  - BGP Prefix Origin Validation (using RPKI)
  - BGP Path Validation

Issuing Party

- Internet Registries (RIR, NIR, Large LIRs)
- Acts as a Certificate Authority and issues certificates for customers
- Provides a web interface to issue ROAs for customer prefixes
- Publishes the ROA records



me / Resources / RPKI

PKI

GP Route Validity

All — Items per page 10 — Search by AS or IP...

Origin AS	Prefix
45192	2001:df0:a::/48
45192	203.176.189.0/24

Showing 1 to 2 of 2 ent

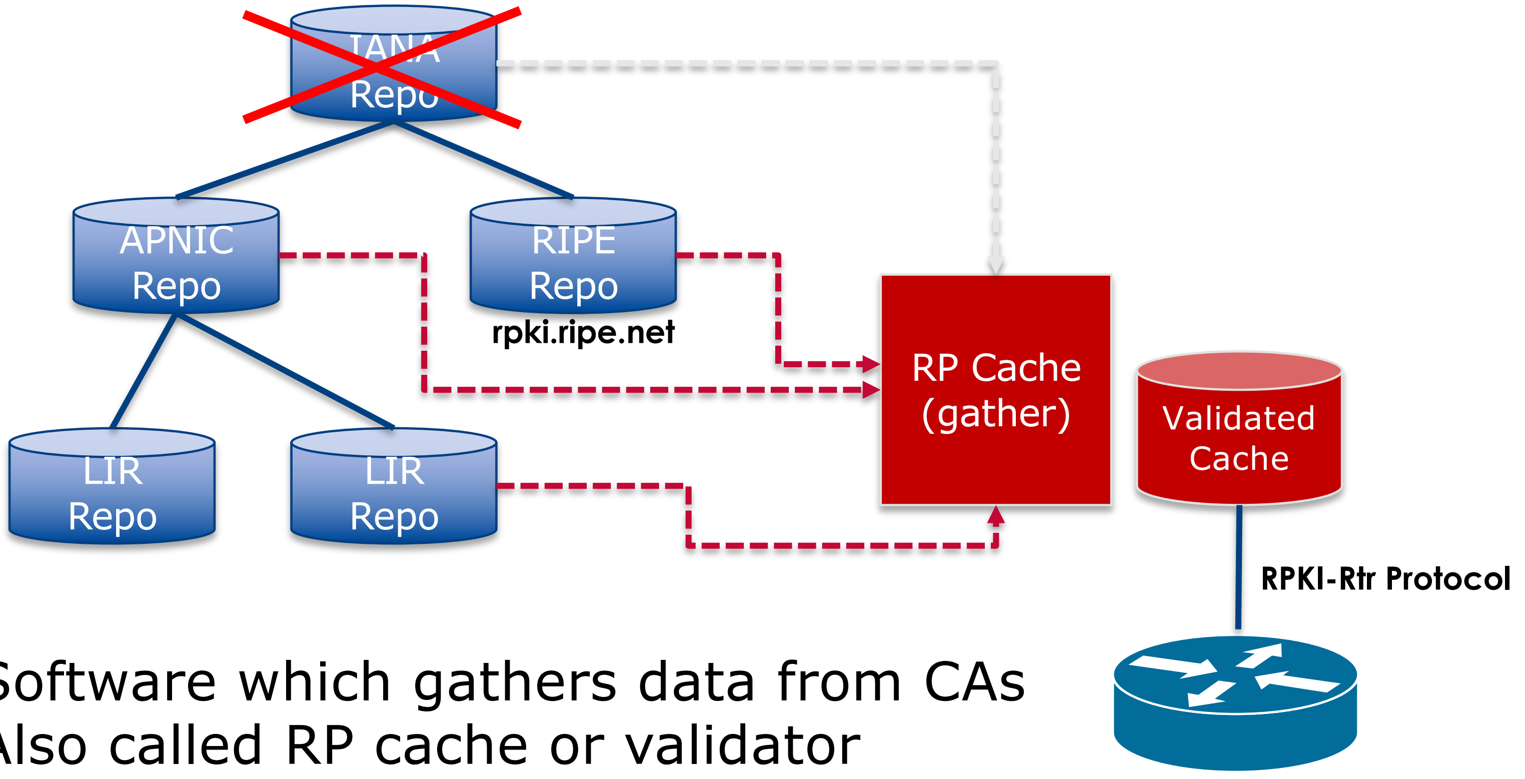
Suggest ROAs — FirstPreviousPage 1 of 1Next

OA Configuration

Origin ASN — Prefix — Max Length — Add — Add & clone — Clear

Courtesy of APNIC: https://apnic.net

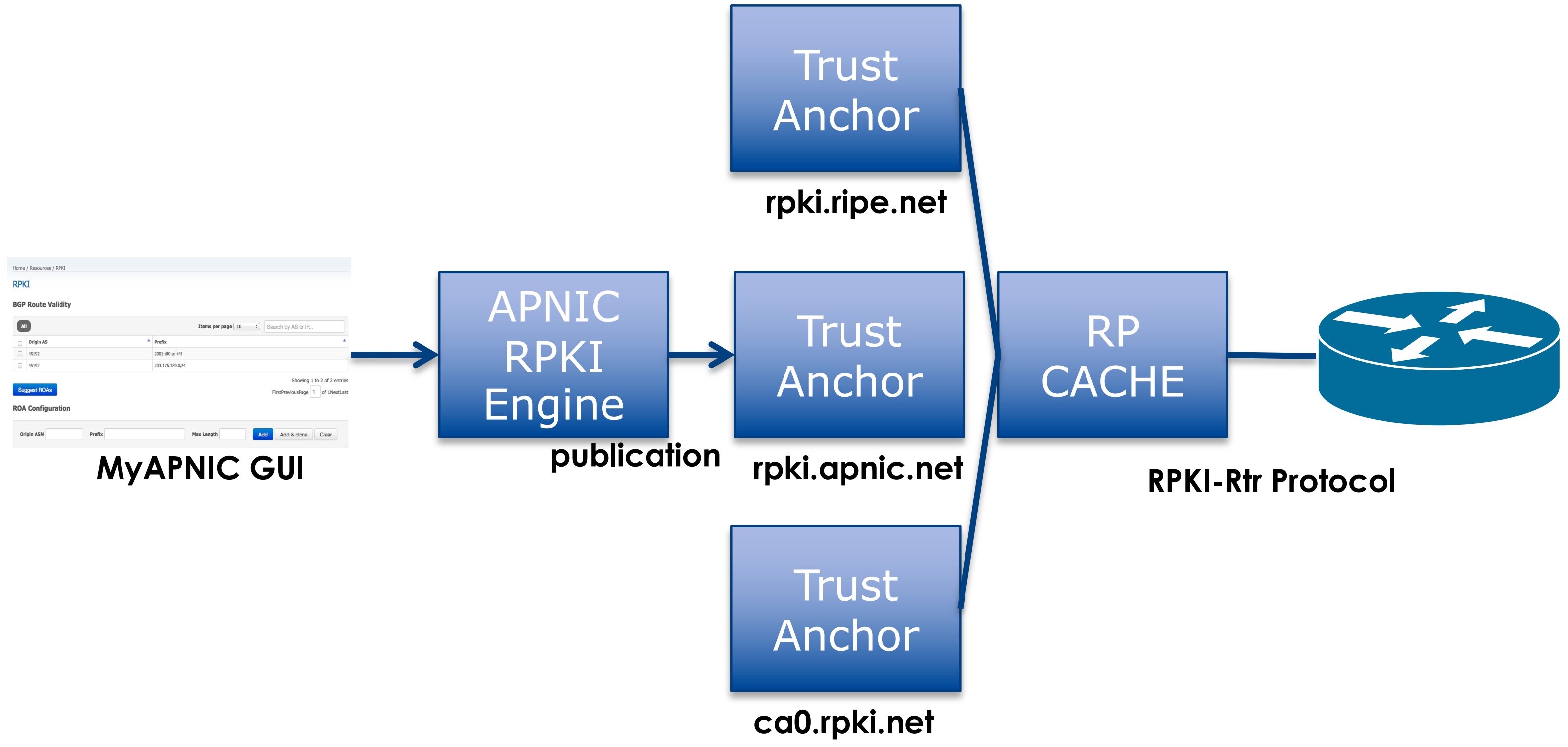
Relying Party (RP)



Software which gathers data from CAs
Also called RP cache or validator

Courtesy of APNIC: https://apnic.net

RPKI Components



Home / Resources / RPKI

RPKI

BGP Route Validity

All | Items per page 10 | Search by AS or IP...

Origin AS	Prefix
45192	2001:df0:a::/48
45192	203.176.189.0/24

Suggest ROAs

Showing 1 to 2 of 2 entries

FirstPreviousPage 1 of 1NextLast

ROA Configuration

Origin ASN | Prefix | Max Length | Add | Add & clone | Clear

Courtesy of APNIC: https://apnic.net

RPKI Service Models

- Hosted Model:
  - The RIR runs the CA on behalf of its members
    - Manage keys, repository, etc
    - Generate certificates for resource certifications
- Delegated Model:
  - Member becomes the CA, delegated from the parent CA (the RIR)
    - Operates the full RPKI system
    - Currently JPNIC, TWNIC and CNNIC operate CAs, delegated from APNIC
  - CA Software
    - NLnetLabs Krill: https://www.nlnetlabs.nl/projects/rpki/krill/

Route Origin Authorisation (ROA)

- A digital object that contains a list of address prefixes and one AS number
- It is an authority created by a prefix holder to authorise an AS Number to originate one or more specific route advertisements
- Publish a ROA using your RIR member portal
  - Consult your RIR for how to use their member portal to publish your ROAs

Route Origin Authorisation

- A typical ROA would look like this:

Prefix	10.10.0.0/16
Max-Length	/18
Origin-AS	AS65534

- There can be more than one ROA per address block
  - Allows the operator to originate prefixes from more than one AS
  - Caters for changes in routing policy or prefix origin

Creating ROAs

- Only create ROAs for the aggregate and the exact subnets expected in the routing table
- Examples:

Prefix	Max Length	Origin AS	Comments
10.10.0.0/16	/24	65534	ROA covers /16 through to /24 – any announced subnets to /24 will be Valid if from AS65534
10.10.0.0/16	/16	65534	ROA covers only /16 – any announced subnets will be Invalid
10.10.4.0/22	/24	65534	ROA covers this /22 through to /24
10.10.32.0/22	/24	64512	Valid ROA covers /22 through to /24 announcements from AS64512

Creating ROAs – Important Notes

- Always create ROAs for the aggregate and the individual subnets being routed in BGP
- Example:
  - If creating a ROA for 10.10.0.0/16 **and** "max prefix" length is set to /16
    - There will only be a valid ROA for 10.10.0.0/16
    - If a subnet of 10.10.0.0/16 is originated, it will be state Invalid

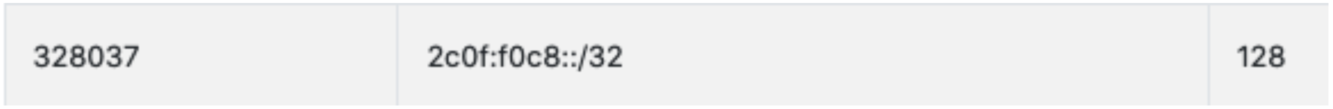
Creating ROAs – Important Notes

- Avoid creating ROAs for subnets of an aggregate unless they are actually being actively routed
  - If ROA exists, but subnet is not routed, it leaves an opportunity for someone else to mis-originate the subnet using the valid origin AS, resulting in a hijack

- https://datatracker.ietf.org/doc/draft-ietf-sidrops-rpkimaxlen/ has a good description of the care needed when creating ROAs
  - Recommendations:
    - Avoid using maxLength attribute unless in special cases
    - Use minimal ROAs wherever possible – only for prefixes that are actually being announced
  - Also a discussion about ROAs for facilitating DDoS Services
  - There is even a strong suggestion that "maxLength" should be deprecated

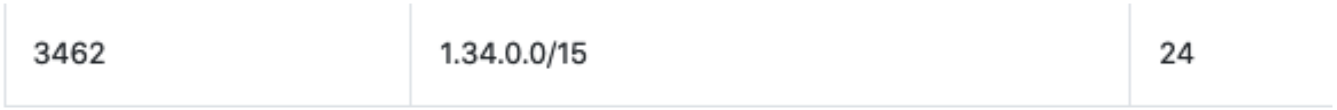
Creating ROAs – Important Notes

- Some current examples of problematic ROAs:



328037	2c0f:f0c8::/32	128

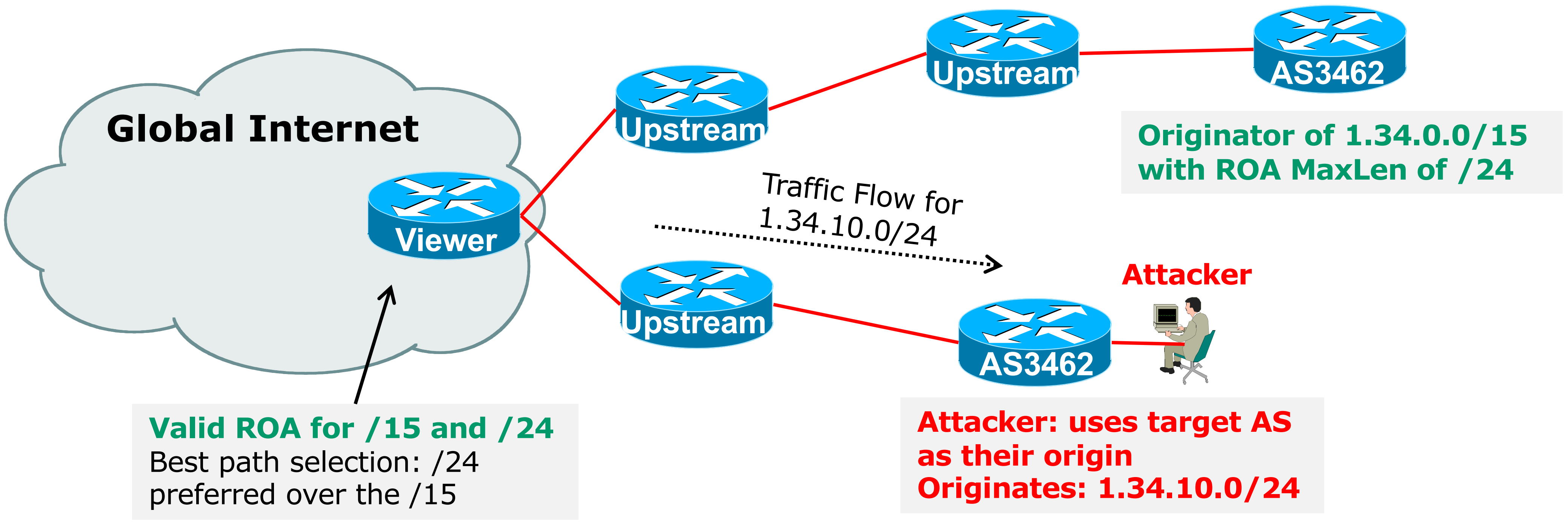
  - This means that any and every subnet of 2C0F:F0C8::/32 originated by AS328037 is valid
    - An attacker can use AS328037 as their origin AS to originate 2C0F:F0C8:A0:/48 to deny service to that address block
    - Known as a validated hijack!



3462	1.34.0.0/15	24

  - This means that any subnet of 1.34.0.0/15 down to a /24 as originated by AS3462 is valid
    - An attacker can use AS3462 as their origin AS to originate 1.34.10.0/24 to deny service to that address block

Creating ROAs: "Validated Hijack"


Global Internet
Viewer
Upstream
Upstream
Upstream
AS3462
AS3462
Originator of 1.34.0.0/15 with ROA MaxLen of /24
Traffic Flow for 1.34.10.0/24
Attacker
Valid ROA for /15 and /24
Best path selection: /24 preferred over the /15
Attacker: uses target AS as their origin
Originates: 1.34.10.0/24

- If the 1.34.10.0/24 prefix had had no ROA, route origin validation would have dropped the invalid announcement at the upstream AS

Creating ROAs: pre-RIR Address Space

- Some entities were assigned address space by InterNIC
  - This is prior to the existence of the RIRs
- How to sign ROAs for these resources?
- Some RIRs will support the signing of legacy address space ROAs
  - If there is documentation proving the holding
  - If there is some service agreement for resources allocated by the RIR
  - Or by some other arrangement
  - Example, APNIC:
    - https://www.apnic.net/wp-content/uploads/2018/02/APNIC-AR-2017.pdf
  - Example, RIPE NCC:
    - https://www.ripe.net/manage-ips-and-asns/resource-management/certification/resource-certification-rpki-for-provider-independent-end-users

Route Origin Validation

- Router must support RPKI
- Checks an RP cache / validator
  - Uses RtR protocol, described in RFC8210
- Validation returns 3 states:

State	Description
Valid	When authorisation is found for prefix X coming from ASN Y
Invalid	When authorisation is found for prefix X but **not** from ASN Y, or **not** allowable subnet size
Not Found	When no authorisation data is found for prefix X

Route Origin Validation – AS0

- RFC6483 also describes “Disavowal of Routing Origination”
  - AS 0 has been reserved for network operators and other entities to identify non-routed networks
  - Which means:
    - “A ROA with a subject of AS0 (AS0 ROA) is an attestation by the holder of a prefix that the prefix described in the ROA, and any more specific prefix, should not be used in a routing context”
- Any prefixes with ROA indicating AS 0 as the origin AS need to be dropped
  - If these prefixes appear with any other origin, their ROAs will be invalid, achieving this goal

Route Origin Validation – AS0

- Possible use cases of AS0:
  - Internal use of a prefix that should not appear in the global BGP table
  - Internet Exchange Point LAN must never appear in the global BGP table
  - Private Address space (IPv4) and non-Global Unicast space (IPv6)
  - Unassigned address space
    - This is under discussion within the various RIR policy fora
  - IPv4 and IPv6 address resources which should not appear in the global BGP table
    - For example, the special use address space described in RFC6890

Route Origin Validation

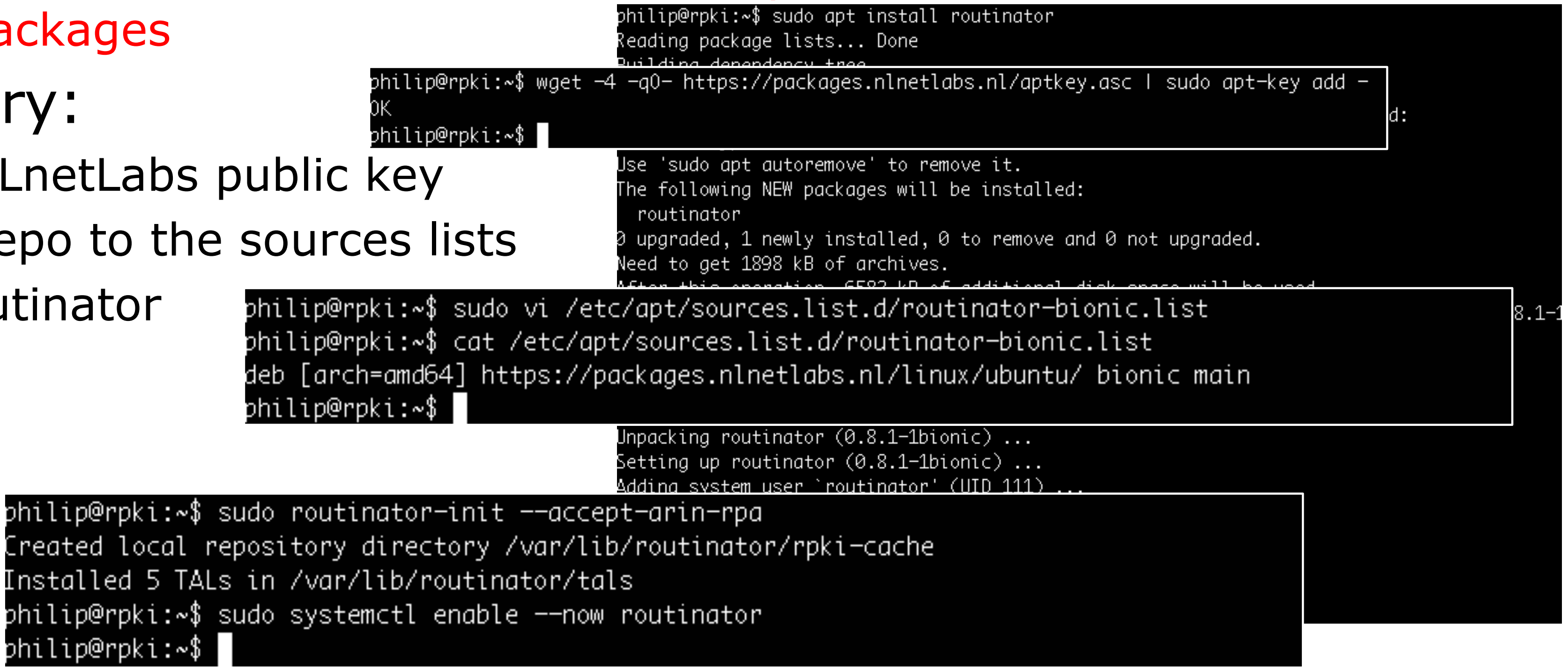
- Implementation support:
  - Cisco IOS – available from release 15.2
  - Cisco IOS/XR – available from release 4.3.2
  - Juniper – available from release 12.2
  - Nokia – available from release R12.0R4
  - Huawei – available from release V800R009C10
  - FRR – available from release 4.0
  - BIRD – available from release 1.6
  - OpenBGPD – available from OpenBSD release 6.4
  - GoBGP – available since 2018
  - VyOS – available from release 1.2.0-RC11

RPKI Validator Caches

- NLnet Labs Routinator
  - https://www.nlnetlabs.nl/projects/rpki/routinator/
  - https://github.com/NLnetLabs/routinator
- RIPE NCC validator
  - https://github.com/RIPE-NCC/rpki-validator-3/wiki
- LACNIC/NIC Mexico validator (FORT)
  - https://github.com/NICMx/FORT-validator
- Cloudflare validator (OctoRPKI)
  - https://github.com/cloudflare/cfrpki
  - https://blog.cloudflare.com/cloudflares-rpki-toolkit/

Installing a validator – NLnetLabs

- If using Ubuntu/Debian, then simply use the package manager, as described:
  - https://github.com/NLnetLabs/routinator#quick-start-with-debian-and-ubuntu-packages
- In summary:
  - Get the NLnetLabs public key
  - Add the repo to the sources lists
  - Install routinator
  - Initialise
  - Run



```
philip@rpki:~$ wget -4 -qO- https://packages.nlnetlabs.nl/aptkey.asc | sudo apt-key add -
OK
philip@rpki:~$
```

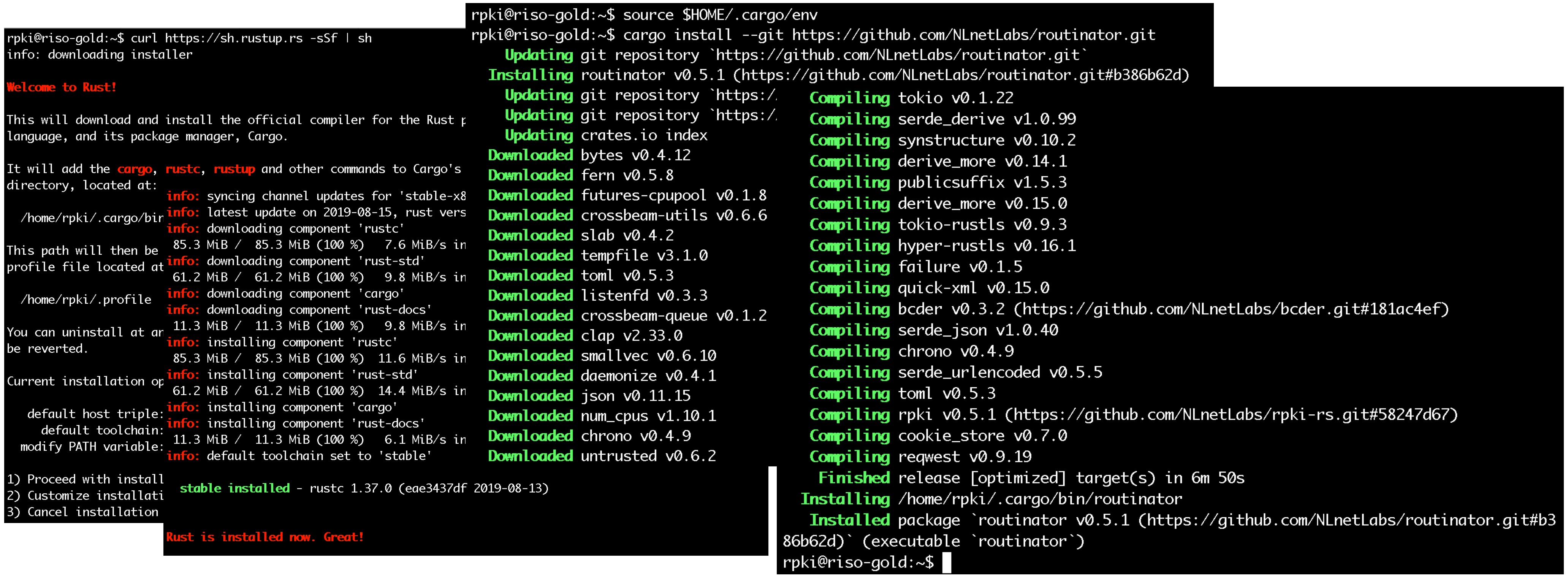
```
philip@rpki:~$ sudo vi /etc/apt/sources.list.d/routinator-bionic.list
philip@rpki:~$ cat /etc/apt/sources.list.d/routinator-bionic.list
deb [arch=amd64] https://packages.nlnetlabs.nl/linux/ubuntu/ bionic main
philip@rpki:~$
```

```
philip@rpki:~$ sudo apt install routinator
Reading package lists... Done
Building dependency tree
d:
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  routinator
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 1898 kB of archives.
After this operation, 6583 kB of additional disk space will be used.
8.1-1
Unpacking routinator (0.8.1-1bionic) ...
Setting up routinator (0.8.1-1bionic) ...
Adding system user `routinator' (UID 111) ...
```

```
philip@rpki:~$ sudo routinator-init --accept-arin-rpa
Created local repository directory /var/lib/routinator/rpki-cache
Installed 5 TALs in /var/lib/routinator/tals
philip@rpki:~$ sudo systemctl enable --now routinator
philip@rpki:~$
```

Installing a validator – NLnet Labs

- If building it from source, consult instructions at:
  - https://github.com/NLnetLabs/routinator



```
rpki@riso-gold:~$ curl https://sh.rustup.rs -sSf | sh
info: downloading installer

Welcome to Rust!

This will download and install the official compiler for the Rust p
language, and its package manager, Cargo.

It will add the cargo, rustc, rustup and other commands to Cargo's
directory, located at:

  /home/rpki/.cargo/bin

This path will then be
profile file located at

  /home/rpki/.profile

You can uninstall at an
be reverted.

Current installation op

   default host triple:
     default toolchain:
  modify PATH variable:

1) Proceed with install
2) Customize installati
3) Cancel installation
```

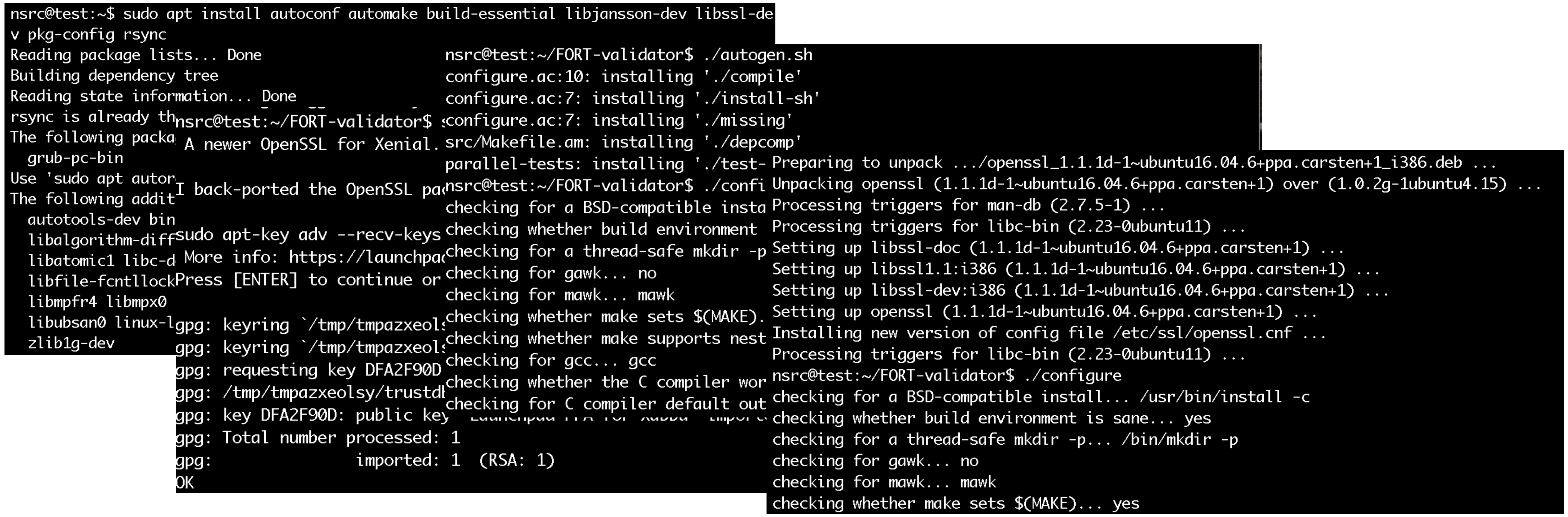
```
info: syncing channel updates for 'stable-x8
info: latest update on 2019-08-15, rust vers
info: downloading component 'rustc'
 85.3 MiB /  85.3 MiB (100 %)   7.6 MiB/s in
info: downloading component 'rust-std'
 61.2 MiB /  61.2 MiB (100 %)   9.8 MiB/s in
info: downloading component 'cargo'
info: downloading component 'rust-docs'
 11.3 MiB /  11.3 MiB (100 %)   9.8 MiB/s in
info: installing component 'rustc'
 85.3 MiB /  85.3 MiB (100 %)  11.6 MiB/s in
info: installing component 'rust-std'
 61.2 MiB /  61.2 MiB (100 %)  14.4 MiB/s in
info: installing component 'cargo'
info: installing component 'rust-docs'
 11.3 MiB /  11.3 MiB (100 %)   6.1 MiB/s in
info: default toolchain set to 'stable'

  stable installed - rustc 1.37.0 (eae3437df 2019-08-13)


Rust is installed now. Great!
```

```
rpki@riso-gold:~$ source $HOME/.cargo/env
rpki@riso-gold:~$ cargo install --git https://github.com/NLnetLabs/routinator.git
    Updating git repository `https://github.com/NLnetLabs/routinator.git`
  Installing routinator v0.5.1 (https://github.com/NLnetLabs/routinator.git#b386b62d)
    Updating git repository `https:/
    Updating git repository `https:/
    Updating crates.io index
  Downloaded bytes v0.4.12
  Downloaded fern v0.5.8
  Downloaded futures-cpupool v0.1.8
  Downloaded crossbeam-utils v0.6.6
  Downloaded slab v0.4.2
  Downloaded tempfile v3.1.0
  Downloaded toml v0.5.3
  Downloaded listenfd v0.3.3
  Downloaded crossbeam-queue v0.1.2
  Downloaded clap v2.33.0
  Downloaded smallvec v0.6.10
  Downloaded daemonize v0.4.1
  Downloaded json v0.11.15
  Downloaded num_cpus v1.10.1
  Downloaded chrono v0.4.9
  Downloaded untrusted v0.6.2
```

```
   Compiling tokio v0.1.22
   Compiling serde_derive v1.0.99
   Compiling synstructure v0.10.2
   Compiling derive_more v0.14.1
   Compiling publicsuffix v1.5.3
   Compiling derive_more v0.15.0
   Compiling tokio-rustls v0.9.3
   Compiling hyper-rustls v0.16.1
   Compiling failure v0.1.5
   Compiling quick-xml v0.15.0
   Compiling bcder v0.3.2 (https://github.com/NLnetLabs/bcder.git#181ac4ef)
   Compiling serde_json v1.0.40
   Compiling chrono v0.4.9
   Compiling serde_urlencoded v0.5.5
   Compiling toml v0.5.3
   Compiling rpki v0.5.1 (https://github.com/NLnetLabs/rpki-rs.git#58247d67)
   Compiling cookie_store v0.7.0
   Compiling reqwest v0.9.19
    Finished release [optimized] target(s) in 6m 50s
  Installing /home/rpki/.cargo/bin/routinator
   Installed package `routinator v0.5.1 (https://github.com/NLnetLabs/routinator.git#b3
86b62d)` (executable `routinator`)
rpki@riso-gold:~$
```

Installing a validator – FORT

- Consult instructions at:
  - https://nicmx.github.io/FORT-validator/installation.html
  - Note: Needs OpenSSL >=1.1



```
nsrc@test:~$ sudo apt install autoconf automake build-essential libjansson-dev libssl-de
v pkg-config rsync
Reading package lists... Done
Building dependency tree
Reading state information... Done
rsync is already th
The following packa
  grub-pc-bin
Use 'sudo apt autor
The following addit
  autotools-dev bin
  libalgorithm-diff
  libatomic1 libc-d
  libfile-fcntllock
  libmpfr4 libmpx0
  libubsan0 linux-l
  zlib1g-dev
```

```
nsrc@test:~/FORT-validator$ s
 A newer OpenSSL for Xenial.

I back-ported the OpenSSL pa

sudo apt-key adv --recv-keys
 More info: https://launchpa
Press [ENTER] to continue or
gpg: keyring `/tmp/tmpazxeol
gpg: keyring `/tmp/tmpazxeol
gpg: requesting key DFA2F90D
gpg: /tmp/tmpazxeolsy/trustd
gpg: key DFA2F90D: public key
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
```

```
nsrc@test:~/FORT-validator$ ./autogen.sh
configure.ac:10: installing './compile'
configure.ac:7: installing './install-sh'
configure.ac:7: installing './missing'
src/Makefile.am: installing './depcomp'
parallel-tests: installing './test-
nsrc@test:~/FORT-validator$ ./confi
checking for a BSD-compatible insta
checking whether build environment
checking for a thread-safe mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE).
checking whether make supports nest
checking for gcc... gcc
checking whether the C compiler wor
checking for C compiler default out
```

```
Preparing to unpack .../openssl_1.1.1d-1~ubuntu16.04.6+ppa.carsten+1_i386.deb ...
Unpacking openssl (1.1.1d-1~ubuntu16.04.6+ppa.carsten+1) over (1.0.2g-1ubuntu4.15) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libc-bin (2.23-0ubuntu11) ...
Setting up libssl-doc (1.1.1d-1~ubuntu16.04.6+ppa.carsten+1) ...
Setting up libssl1.1:i386 (1.1.1d-1~ubuntu16.04.6+ppa.carsten+1) ...
Setting up libssl-dev:i386 (1.1.1d-1~ubuntu16.04.6+ppa.carsten+1) ...
Setting up openssl (1.1.1d-1~ubuntu16.04.6+ppa.carsten+1) ...
Installing new version of config file /etc/ssl/openssl.cnf ...
Processing triggers for libc-bin (2.23-0ubuntu11) ...
nsrc@test:~/FORT-validator$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
```

RP Cache Deployment

- Network Operator design advice:
  - Deploy at least two Validator Caches
  - Geographically diverse
  - Perhaps two different implementations
    - For software independence
  - Implement on a Linux container so that the container can be moved between different server clusters as required
  - Configure validator to listen on both IPv4 and IPv6
    - Configure routers with both IPv4 and IPv6 validator connections
  - Securing the validator: Only permit routers running EBGP to have access to the validators

RP Cache Deployment: Open Questions

- Consider two different validator cache implementations
  - Gives software independence
  - What happens if the different cache implementations contain different VRPs?
  - Scenario 1:
    - Cache 1: route X is valid
    - Cache 2: route X is invalid
  - Scenario 2:
    - Cache 1: route X is valid
    - Cache 2: route X is NotFound
  - Answer: depends on router vendor implementation?!

Configure Router to Use Cache: Cisco IOS

- Point router to the local RPKI cache
  - Server listens on port 3323
  - Cache refreshed every 60 minutes (RFC8210 recommendation)
  - Example:

```
router bgp 64512
 bgp rpki server tcp 10.0.0.3 port 3323 refresh 3600
```

  - Once the router's RPKI table is populated, router indicates validation state in the BGP table

Cisco IOS status commands

- `show ip bgp rpki servers`
  - Displays the connection status to the RPKI servers
- `show ip bgp rpki table`
  - Shows the VRPs (validated ROA payloads)
- `show ip bgp`
  - Shows the BGP table with status indication next to the prefix
- `show ip bgp | i ^V`
  - Shows the status ”valid” prefixes in the BGP table

Configure Router to Use Cache: JunOS

1. Connect to validation cache:

```
routing-options {
  validation {
    group ISP {
      session 10.0.0.3;
      port 3323;
      refresh-time 600;
      hold-time 3600;
    }
  }
}
```

- (using same parameters as for the Cisco IOS example)

Configure Router to Use Cache: JunOS

2. Configure validation policies:

```
policy-options {
  policy-statement RPKI-validation {
    term VALID {
      from {
        protocol bgp;
        validation-database valid;
      }
      then {
        validation-state valid;
        next policy;
      }
    }
    term INVALID {
      from {
        protocol bgp;
        validation-database invalid;
      }
      then {
        validation-state invalid;
        next policy;
      }
    }
```

```
(continued)...

    term UNKNOWN {
      from {
        protocol bgp;
        validation-database unknown;
      }
      then {
        validation-state unknown;
        next policy;
      }
    }
  }
}
```

Configure Router to Use Cache: JunOS

3. Apply policy to eBGP session:

```
protocols {
  bgp {
    group EBGP {
      type external;
      local-address 10.0.1.1;
      neighbor 10.1.15.1 {
        description ”ISP Upstream";
        import [ RPKI-validation Upstream-in ];
        export LocalAS-out;
        peer-as 64511;
      }
    }
  }
}
```

- Note that policy options *Upstream-in* and *LocalAS-out* are the typical inbound and outbound filters needed for an eBGP session

JunOS status commands

- `show validation session detail`
  - Display the details of the connection to the RPKI servers
- `show validation replication database`
  - Shows the VRPs (validated ROA payloads)
- `show route protocol bgp`
  - Shows the BGP table with status indication next to the prefix

  `show route protocol bgp validation-state valid`
  - Shows the status ”valid” prefixes in the BGP table

Implementation notes

- Cisco IOS/IOS-XE
  - Prefixes originated locally into IBGP are automatically marked as Valid
    - There is no check against the cached validation table
    - Allows operator to originate non-signed address blocks or other entity address space inside their own IBGP
- JunOS
  - Complete separate between validation table and what happens in BGP
    - There has to be a specific policy statement for any action based on validation state

Implementation notes

- What happens when router cannot contact any validator cache?
  - Cisco IOS/IOS-XE – empties the VRP table within 5 minutes
  - Juniper & Nokia – keeps VRPs until their preconfigured expiry (default 60 minutes)
  - Other vendors – behaviour untested

- Design advice:
  - It is important to ensure that EBGP speaking routers can always remaining connected to a validator cache
    - **Minimum of two independent caches recommended!**

Check Server

```
lg-01-jnb.za>sh ip bgp rpki servers
BGP SOVC neighbor is 105.16.112.2/43779 connected to port 43779
Flags 64, Refresh time is 300, Serial number is 1463607299
InQ has 0 messages, OutQ has 0 messages, formatted msg 493
Session IO flags 3, Session flags 4008
 Neighbor Statistics:
  Prefixes 25880
  Connection attempts: 44691
  Connection failures: 351
  Errors sent: 35
  Errors received: 0

Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled
Mininum incoming TTL 0, Outgoing TTL 255
Local host: 105.22.32.2, Local port: 27575
Foreign host: 105.16.112.2, Foreign port: 43779
Connection tableid (VRF): 0
```

Courtesy of SEACOM: http://as37100.net

Check Server

```
philip@DREN-THIMPHU-BR> show validation session detail
Session 103.197.176.141, State: up, Session index: 2
  Group: DrukREN, Preference: 100
  Local IPv4 address: 103.197.176.5, Port: 3323
  Refresh time: 600s
  Hold time: 1800s
  Record Life time: 3600s
  Serial (Full Update): 0
  Serial (Incremental Update): 1
    Session flaps: 1
    Session uptime: 00:19:11
    Last PDU received: 00:00:34
    IPv4 prefix count: 94329
    IPv6 prefix count: 15992
```

Courtesy of DrukREN, Bhutan

RPKI Table (IPv4) – December 2020

```
160783 BGP sovc network entries using 25725280 bytes of memory
175947 BGP sovc record entries using 5630304 bytes of memory

Network                  Maxlen  Origin-AS  Source  Neighbor
1.0.0.0/24               24      13335      0       203.98.225.12/3323
1.0.4.0/24               24      38803      0       203.98.225.12/3323
1.0.4.0/22               22      38803      0       203.98.225.12/3323
1.0.5.0/24               24      38803      0       203.98.225.12/3323
1.0.6.0/24               24      38803      0       203.98.225.12/3323
1.0.7.0/24               24      38803      0       203.98.225.12/3323
1.1.1.0/24               24      13335      0       203.98.225.12/3323
1.1.4.0/22               22      4134       0       203.98.225.12/3323
1.1.16.0/20              20      4134       0       203.98.225.12/3323
1.2.9.0/24               24      4134       0       203.98.225.12/3323
1.2.10.0/24              24      4134       0       203.98.225.12/3323
1.2.11.0/24              24      4134       0       203.98.225.12/3323
1.2.12.0/22              22      4134       0       203.98.225.12/3323
1.3.0.0/16               16      4134       0       203.98.225.12/3323
1.6.0.0/22               24      9583       0       203.98.225.12/3323
1.6.4.0/22               24      9583       0       203.98.225.12/3323
```

RPKI Table (IPv6) – December 2020

```
27783 BGP sovc network entries using 5112072 bytes of memory
29915 BGP sovc record entries using 957280 bytes of memory

Network                    Maxlen  Origin-AS  Source  Neighbor
2001:200::/32              32      2500       0       203.98.225.12/3323
2001:200:136::/48          48      9367       0       203.98.225.12/3323
2001:200:1BA::/48          48      24047      0       203.98.225.12/3323
2001:200:900::/40          40      7660       0       203.98.225.12/3323
2001:200:8000::/35         35      4690       0       203.98.225.12/3323
2001:200:C000::/35         35      23634      0       203.98.225.12/3323
2001:200:E000::/35         35      7660       0       203.98.225.12/3323
2001:218:3002::/48         48      1613       0       203.98.225.12/3323
2001:260::/32              48      2518       0       203.98.225.12/3323
2001:288::/32              32      1659       0       203.98.225.12/3323
2001:2F0::/32              128     7514       0       203.98.225.12/3323
2001:300::/32              32      2497       0       203.98.225.12/3323
2001:360::/32              32      135887     0       203.98.225.12/3323
2001:360:12::/48           48      135887     0       203.98.225.12/3323
2001:360:13::/48           48      135887     0       203.98.225.12/3323
2001:360:14::/48           48      135887     0       203.98.225.12/3323
```

BGP Table (IPv4)

```
RPKI validation codes: V valid, I invalid, N Not found

Network              Metric LocPrf Path
N*>  1.0.4.0/24        0           37100 6939 4637 1221 38803 56203 i
N*>  1.0.5.0/24        0           37100 6939 4637 1221 38803 56203 i
...
V*>  1.9.0.0/16        0           37100 4788 i
N*>  1.10.8.0/24       0           37100 10026 18046 17408 58730 i
N*>  1.10.64.0/24      0           37100 6453 3491 133741 i
...
V*>  1.37.0.0/16       0           37100 4766 4775 i
N*>  1.38.0.0/23       0           37100 6453 1273 55410 38266 i
N*>  1.38.0.0/17       0           37100 6453 1273 55410 38266 {38266} i
...
I*   5.8.240.0/23      0           37100 44217 3178 i
I*   5.8.241.0/24      0           37100 44217 3178 i
I*   5.8.242.0/23      0           37100 44217 3178 i
I*   5.8.244.0/23      0           37100 44217 3178 i
...
```

Courtesy of SEACOM: http://as37100.net

BGP Table (IPv6)

```
RPKI validation codes: V valid, I invalid, N Not found

Network                    Metric LocPrf Path
N*>  2001::/32                  0        37100 6939 i
N*   2001:4:112::/48            0        37100 112 i
...
V*>  2001:240::/32              0         37100 2497 i
N*>  2001:250::/48              0         37100 6939 23911 45
N*>  2001:250::/32              0         37100 6939 23911 23910 i
...
V*>  2001:348::/32              0         37100 2497 7679 i
N*>  2001:350::/32              0         37100 2497 7671 i
N*>  2001:358::/32              0         37100 2497 4680 i
...
I*   2001:1218:101::/48         0         37100 6453 8151 278 i
I*   2001:1218:104::/48         0         37100 6453 8151 278 i
N*   2001:1221::/48             0         37100 2914 8151 28496 i
N*>  2001:1228::/32             0         37100 174 18592 i
...
```

Courtesy of SEACOM: http://as37100.net

RPKI BGP State: Valid

```
BGP routing table entry for 2001:240::/32, version 109576927
Paths: (2 available, best #2, table default)
  Not advertised to any peer
  Refresh Epoch 1
  37100 2497
    2C0F:FEB0:11:2::1 (FE80::2A8A:1C00:1560:5BC0) from
                                 2C0F:FEB0:11:2::1 (105.16.0.131)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 37100:2 37100:22000 37100:22004 37100:22060
      path 0828B828 RPKI State valid
      rx pathid: 0, tx pathid: 0x0
```

Courtesy of SEACOM: http://as37100.net

RPKI BGP State: Invalid

```
BGP routing table entry for 2001:1218:101::/48, version 149538323
Paths: (2 available, no best path)
  Not advertised to any peer
  Refresh Epoch 1
  37100 6453 8151 278
    2C0F:FEB0:B:3::1 (FE80::86B5:9C00:15F5:7C00) from
                                              2C0F:FEB0:B:3::1 (105.16.0.162)
      Origin IGP, metric 0, localpref 100, valid, external
      Community: 37100:1 37100:12
      path 0DA7D4FC RPKI State invalid
      rx pathid: 0, tx pathid: 0
```

Courtesy of SEACOM: http://as37100.net

RPKI BGP State: Not Found

```
BGP routing table entry for 2001:200::/32, version 124240929
Paths: (2 available, best #2, table default)
  Not advertised to any peer
  Refresh Epoch 1
  37100 2914 2500
    2C0F:FEB0:11:2::1 (FE80::2A8A:1C00:1560:5BC0) from
                               2C0F:FEB0:11:2::1 (105.16.0.131)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 37100:1 37100:13
      path 19D90E68 RPKI State not found
      rx pathid: 0, tx pathid: 0x0
```

Courtesy of SEACOM: http://as37100.net

Using RPKI

- Network operators can make decisions based on RPKI state:
  - Invalid – discard the prefix – several do this now!
  - NotFound – let it through (maybe low local preference)
  - Valid – let it through (high local preference)
- Some operators even considering making “Not Found” a discard event
  - But then Internet IPv4 BGP table would shrink to about 102000 prefixes and the IPv6 BGP table would shrink to about 17000 prefixes!

Deploying RPKI within an AS

- For fully supported Route Origin Validation across the network:
  - All EBGP speaking routers need talk with a validator
    - Supporting ROV means dropping **invalid**s as they arrive in the network
    - EBGP speaking routers are part of the operator IBGP mesh
  - IBGP speaking routers do not need to talk with a validator
    - Only **valid** and **NotFound** prefixes will be distributed from the EBGP speaking routers
    - The validation table is not distributed from router to router
- Note:
  - Cisco IOS/IOS-XE drops **invalid**s by default – to allow **invalid**s to be distributed by IBGP, use the per address-family command:

```
bgp bestpath prefix-validate allow-invalid
```

Propagating validation state

- RFC8097 describes the propagation of validation state between iBGP speakers
  - Defines an opaque extended BGP community

Extended Community	Meaning
0x4300:0:0	Valid
0x4300:0:1	NotFound
0x4300:0:2	Invalid

  - These extended communities can be used in IBGP to allow distribution of validation state along with the prefix if desired
  - On Cisco IOS/IOS-XE:

```
neighbor x.x.x.x announce rpki state
```

  - For JunOS, policy needs to be explicitly configured

Propagating validation state

- There are two important caveats when propagating validation state:
  - Interoperability – is the defined opaque extended community supported on all vendor equipment in a multi-vendor network?
    - Until recently JunOS would not allow the required opaque extended communities to be configured at the command line
  - Cisco IOS/IOS-XE behaviour:
    - Adds a step to the best path selection algorithm: checks validation state (***valid*** **preferred over** ***not found***) before checking local preference

JunOS: opaque extended community

- Supported only in most recent JunOS releases
  - Fixed from 17.4R3, 18.2R3, 18.4R2…

```
policy-options {
    community RPKI-VALID members 0x4300:0:0;
    community RPKI-UNKNOWN members 0x4300:0:1;
    community RPKI-INVALID members 0x4300:0:2;
}
```

JunOS: opaque extended community

- And we can now set policy to detect these communities being sent from Cisco IOS/IOS-XE routers
  - Under “policy-options”:

```
policy-statement PEER-in {
    term VALID {
        from community RPKI-VALID;
        then {
            validation-state valid;
            next policy;
        }
    }
    term INVALID {
        from community RPKI-INVALID;
        then {
            validation-state invalid;
            next policy;
        }
    }
    term UNKNOWN {
        from community RPKI-UNKNOWN;
        then {
            validation-state unknown;
            next policy;
        }
    }
}
```

Propagating validation state: Cisco IOS

- Cisco IOS/IOS-XE behaviour – example:
  - Prefix learned via two paths via two separate EBGP speaking routers
  - Prefix and validation state distributed by IBGP to core router (route reflector):

```
     Network          Next Hop      Metric LocPrf Weight Path
V*>i 61.45.249.0/24   100.68.1.1         0     50      0 121 20 135534 i
N* i                  100.68.1.3         0    200      0 20 135534 i
V*>i 61.45.250.0/24   100.68.1.1         0     50      0 121 30 135535 i
N* i                  100.68.1.3         0    150      0 30 135535 i
V*>i 61.45.251.0/24   100.68.1.1         0     50      0 121 122 40 135536 i
N* i                  100.68.1.3         0    150      0 40 135536 i
```

  - One EBGP speaking router talks with validator
  - The other EBGP speaking router does not (due to error or design)
  - Core router best path selection prefers *valid* path over *not found* even if the latter has higher local preference

Propagating validation state: Cisco IOS

- Looking at the path detail:

```
BGP routing table entry for 61.45.249.0/24, version 32
BGP Bestpath: deterministic-med
Paths: (2 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  121 20 135534, (Received from a RR-client)
    100.68.1.1 (metric 2) from 100.68.1.1 (100.68.1.1)
      Origin IGP, metric 0, localpref 50, valid, internal, best
      Extended Community: 0x4300:0:0
      path 67A585D0 RPKI State valid
  Refresh Epoch 1
  20 135534, (Received from a RR-client)
    100.68.1.3 (metric 2) from 100.68.1.3 (100.68.1.3)
      Origin IGP, metric 0, localpref 200, valid, internal
      Community: 10:1100
      Extended Community: 0x4300:0:1
      path 67A58918 RPKI State not found
```

Note best path

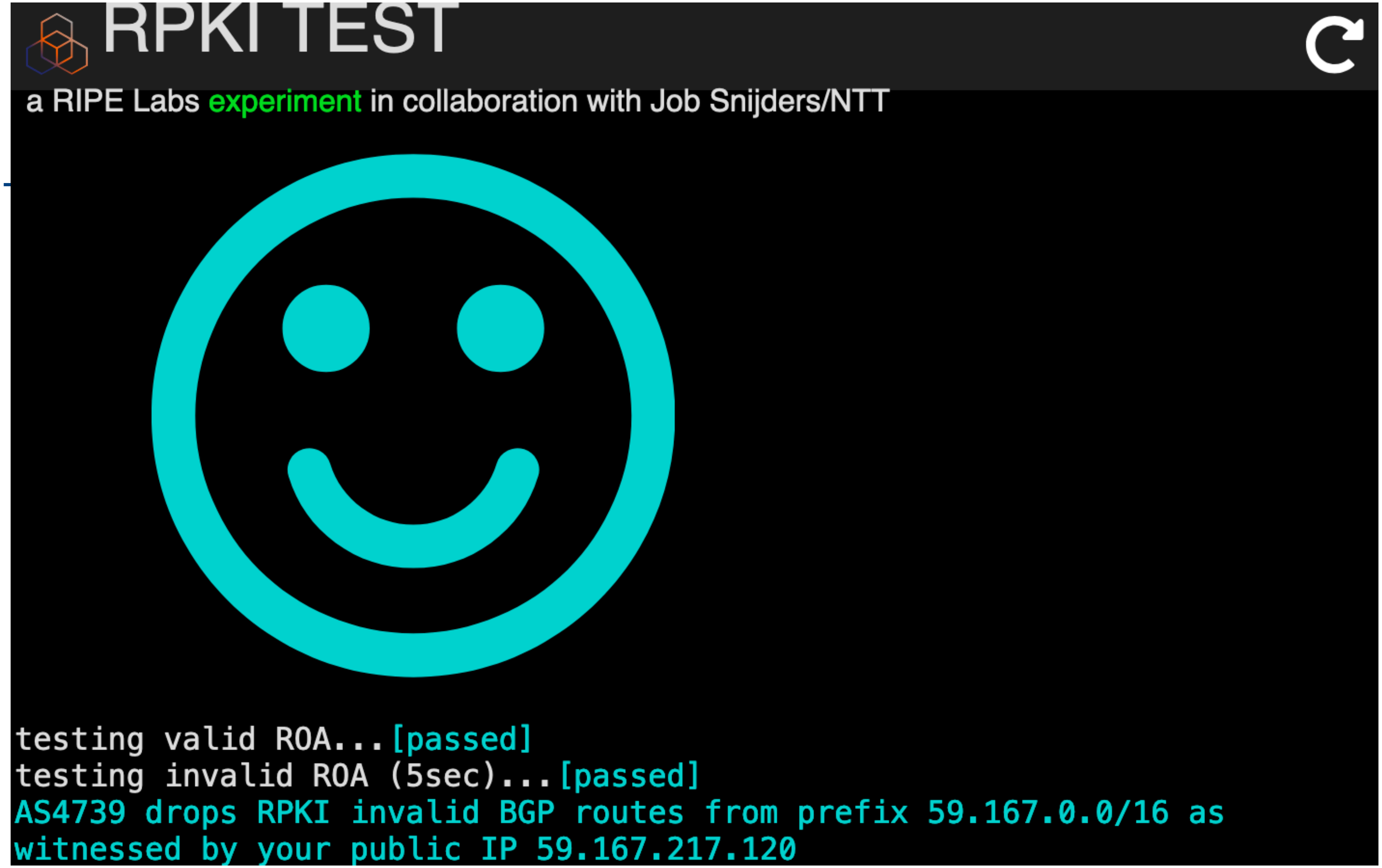
Propagating validation state

- Consider carefully if this is desired
- Current standard practice is to:
  - EBGP speaking routers have session with two diverse/redundant validators
  - Check validation state on EBGP speaking routers
  - Drop invalids on EBGP speaking routers
  - Distribute remaining prefixes by IBGP
  - Avoid propagating validation state (at least in Cisco IOS)

    -or-
  - Make sure that EBGP speaking routers never lose their connectivity to validators

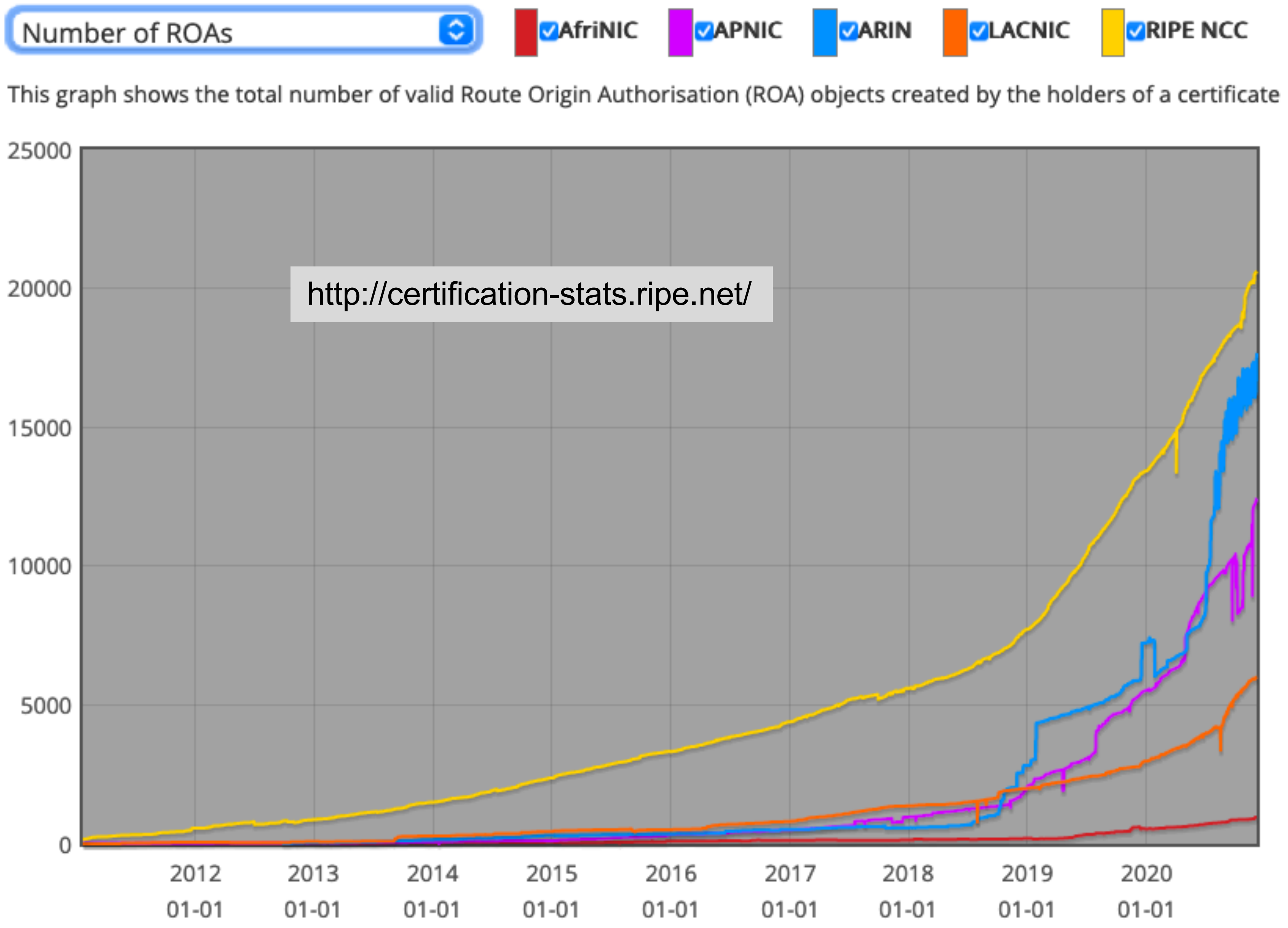
RPKI Summary

- All AS operators must consider deploying:
  - **Signing ROAs**
  - **Dropping Invalids** (ROV)
  - Test if you are doing both: http://www.ripe.net/s/rpki-test
- An important step to securing the routing system
- Doesn't secure the path, but that's the next important hurdle to cross
- With origin validation, the opportunities for malicious or accidental mis-origination disappear
- FAQ:
  - https://nlnetlabs.nl/projects/rpki/faq/


RPKI TEST

a RIPE Labs experiment in collaboration with Job Snijders/NTT

```
testing valid ROA...[passed]
testing invalid ROA (5sec)...[passed]
AS4739 drops RPKI invalid BGP routes from prefix 59.167.0.0/16 as
witnessed by your public IP 59.167.217.120
```

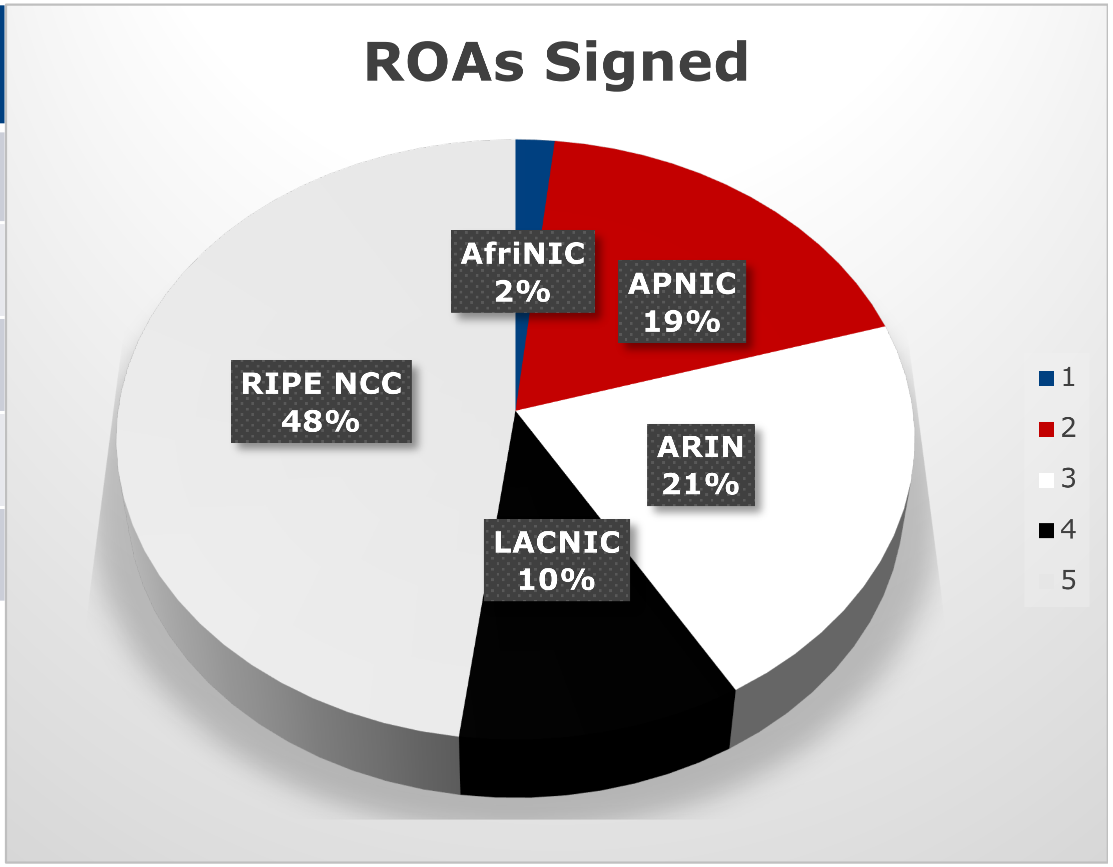


This graph shows the total number of valid Route Origin Authorisation (ROA) objects created by the holders of a certificate

Stats per RIR region

RIR	ROAs	IPv4 Prefixes	IPv6 Prefixes
AfriNIC	439	672	126
APNIC	4711	21302	3972
ARIN	5390	5125	1107
LACNIC	2676	5410	1160
RIPE NCC	12171	54932	8443

As on 10th October 2019



RPKI Deployment Status

- NIST keeps track of deployment status for research purposes:
  - https://rpki-monitor.antd.nist.gov/
- RIPE NCC statistics:
  - http://certification-stats.ripe.net/
- APNIC R&D ROA status:
  - RIPE NCC Validator running at APNIC
  - http://nong.rand.apnic.net:8080/roas

Major Operators deploying RPKI and ROV

- Telia

```
aut-num:        AS1299
org:            ORG-TCA23-RIPE
as-name:        TELIANET
descr:          Telia Carrier
<snip>
remarks:        AS1299 is matching RPKI validation state and reject
remarks:        invalid prefixes from peers, and are currently extending
remarks:        this to our customer connections.
remarks:
remarks:        Our looking-glass at https://lg.telia.net/ marks
remarks:        validation state for all prefixes.
remarks:
remarks:        Please review your registered ROAs to reduce number
remarks:        of invalid prefixes.
```

Major Operators deploying RPKI and ROV

- Telia
  - Dropping invalids from peers, extending to customers by early 2020
- AT&T
  - Dropping invalids from peers
- SEACOM
  - Dropping invalids from peers
- WorkOnLine Communications
  - Dropping invalids from peers
- Cloudflare

Routing Security

- Implement the recommendations in https://www.manrs.org/manrs
  1. Prevent propagation of incorrect routing information
     - Filter BGP peers, in & out!
  2. Prevent traffic with spoofed source addresses
     - BCP38 – Unicast Reverse Path Forwarding
  3. Facilitate communication between network operators
     - NOC to NOC Communication
     - Up-to-date details in Route and AS Objects, and PeeringDB
  4. Facilitate validation of routing information
     - Route Origin Authorisation using RPKI

MANRS

Summary

- Deploy RPKI
  - It is in the Internet’s best interest
- With wide deployment of RPKI it becomes possible to only allow validated prefix announcements into the Internet Routing System
  - Prevents mis-originations
  - Prevents prefix hijack
  - Makes the Internet infrastructure more reliable and more stable
  - Allows the next step: AS-PATH validation

BGP Origin Validation

ISP Workshops