The ISP Column
A column on various things Internet

May 2025
Geoff Huston

Resilience in the RPKI

I would like to look at the ways in which the operators of the number Resource Public Key Infrastructure (RPKI) have deployed this infrastructure in a way that maximises its available and performance and hardens it against potential service interruptions, or in other words, an examination of the resilience of the RPKI infrastructure.

For those who came in late, the RPKI is a hierarchically structured set of X.509 public key certificates that binds a public key value against an enumerated set of number resources (IPv4 address prefixes, IPv6 address prefixes and Autonomous System Numbers). It allows a key holder to demonstrate their control over a collection of number resources by demonstration through the use of a private key, and others to validate such an attestation of control through the verification of this via the matching public key certificate. It is being used in the area of routing security by adding verifiable digital signatures with the intent of providing some level of assurance of the authenticity of certain routing protocol transactions. Because of this application in the global routing system there are natural concerns relating to the resilience of the RPKI infrastructure, as there is no desire to add to the operational risk profile of the Internet's global routing system by adding a security framework which impairs the resilience of this system.

> Just to be pedantic here, the RPKI system is not a single hierarchy, but rather uses five such hierarchies, each rooted in a Trust Anchor that is published by each of the five Regional Internet Registries (RIRs).

When considering the topic of resilience of the RPKI infrastructure it is important to appreciate a number of distinctions between this infrastructure and other distributed data systems used in the infrastructure of the Internet, such as the Domain Name System (DNS).

> Aside: DNS Operation
>
> The DNS uses on-demand interrogation in providing its service of mapping names to associated data, such as IP addresses. Local clients are primed with the IP addresses of the name servers that serve the root zone of the DNS, and the task of resolution of a DNS name starts with a top-down discovery process to find the set of authoritative nameservers that serve the zone in which the name to be resolved is defined, followed by a query to one of these nameservers to retrieve the required data attribute.
>
> Notionally, every name resolution task commences with a query to a root zone server as the first query in the discovery process. This is a notional concept rather than a coded behaviour because name resolvers normally

cache the responses they receive for potential re-use during a specified cache lifetime.

Notwithstanding this caching behaviour, it is necessary for all DNS resolvers to have access to one of more root nameservers all of the time. If such access is prevented (such as when a local DNS resolver is isolated) it will continue to function for the duration of the remaining cache residence lifetimes in its local cache, but once these records have expired the resolver will be unable to resolve names.

For this reason, and for reasons of improved query performance, much effort has been placed in operating a global network of root zone nameservers, attempting to ensure that root service is available to all DNS resolver clients all of the time.

See https://www.potaroo.net/ispcol/2025-03/roots.html for more details.

RPKI Operation

The RPKI does not operate using the on-demand query/response mode as used by the DNS. The RPKI uses a mode of pre-provisioning, where local tools assemble and local validated credentials into RPKI-secured BGP speakers (routers) in the form of filter lists which are then applied to received (and sent) BGP updates. Each network operates one (or potentially more than one) local RPKI client instances (to be strictly correct in terminology I'm referring here to "local relying party service" instances) than maintains a local cache of all currently valid RPKI certificates and signed objects, which allows the client to then assemble a list of authorised address prefixes and associated origination AS. This list is converted into a router filter list, and the client service uses a dedicated protocol to pass the changes the changes to this filter list to a collection of managed routers. The routers can then apply this filter list to all incoming BGP announcements (and potentially to all outgoing BGP advertisements as well).

Each RPKI client periodically sweeps across all the RPKI publication points to ensure that its local copy of all RPKI objects is complete and current. The IETF standards do not specify how often an RPKI client should comb across the collection of published RPKI objects (certificates, manifests and signed objects) to detect changes. Many network operators use a client configuration that performs such a sweep every 10 minutes. Others use shorter intervals, and some use longer intervals.

The implication of this behaviour is that all RPKI publication points should be accessible by all RPKI clients all of the time, and certainly this is the ideal situation, but the system is more tolerant of operational interruption than this informal description might lead you to believe. An RPKI signed object is valid for the period of time listed in the date fields of the public key certificate whose private key signed this object. Once an object has been loaded and cached by a RPKI client, it can be considered as valid until the expiration time. This validity can be negated if the certificate is listed in a revocation list, or if the certificate is no longer listed in a publication point manifest. The implication for this in terms of system resiliency is that if a RPKI publication point is not accessible by a client, then the previously loaded valid objects will continue to be treated as current valid objects until they expire. This is roughly equivalent to the DNS cache lifetime directive in the DNS and makes the RPKI system somewhat tolerant of short-term interruptions in connectivity.

The DNS is effectively a two-state system, where the queried data exists, or it does not exist. The RPKI is a tri-state system, where a route object is either accepted as a "valid" prefix, or is a candidate for rejection ("invalid"") as existing validated RPKI objects contradict this route object, or third state of "unknown" where no valid RPKI objects are relevant to this route object. Conventional operational practice is to construct route filters that accept route advertisements for "valid" and "unknown" route objects, and discard advertisements for "invalid" routes. If a section of the RPKI is inaccessible to a local RPKI client for a protracted period where the RPKI timers expire, then the router's behaviour would not materially alter, on so far as previously "valid" objects would change to "unknown," but in general would not necessarily change to "invalid."

Resilience in RPKI Object Publication

Notwithstanding the tolerance of the RPKI system to interruptions in the availability of public RPKI objects, it is still desirable to ensure that the RPKI material, which consists of the Trust Anchor Locators (TALs), and the material published in each of the RPKI CA'S publications points is managed in a resilient a manner as possible.

Trust Anchor Locators

These objects are the notional equivalent of the root zone servers of the DNS, but at that point the similarity ends. These TAL objects are not used in the validation of RPKI objects, but are pointers to where the trust anchor objects (self-signed certificates) can be found.

These TAL objects are published by each RIR:

- APNIC: https://www.apnic.net/community/security/resource-certification/tal-archive/
- ARIN: https://www.arin.net/resources/manage/rpki/tal/
- RIPE NCC: https://tal.rpki.ripe.net/ripe-ncc-rfc8630.tal
- LACNIC: https://www.lacnic.net/4984/2/lacnic/rpki-trust-anchors
- AFRINIC: https://afrinic.net/resource-certification/tal

These URLs are all published using unicast servers, and do not use a Content Distribution Network (CDN). Why not go a step further any load these objects into a CDN so that the material is published in a replicated manner that is more resilient and faster to collect?

The reasons lie in the observation that this information is not retrieved by RPKI client systems when they are validating RPKI-signed objects. This is configuration information, loaded when the client system starts up. The information in these TAL objects is sufficiently static that a number of package distributors have taken to bundling up these TAL objects themselves, and distributing it as part of the RPKI packet suite (for example https://packages.debian.org/sid/rpki-trust-anchors) and there is no online query being performed in any case.

It would appear that there is little to be gained in terms of operational resilience by any alteration of these arrangements.

RPKI Publication Points

It's a slightly different story for the information published in the RPKI publication points by RPKI publishers.

However, this information, namely the certificates issued by this RPKI CA, all RPKI-signed products signed by this CA, and a manifest, is not part of any real time dependency used by routers in processing BGP updates or switching packets. The assembly of RPKI signed objects by client software, and the construction of filter lists to pass to RPKI-aware routers is a background task

that is offloaded to a RPKI engine, and the performance issue of time to retrieve the objects in an RPKI publication point is not necessarily a critical performance issue.

There are two access protocols used by RPKI client tools. The original protocol, RSYNC, is not readily amenable to publication platforms that use duplicated instances of the content and anycast transport. The alternative, namely the RPKI Repository Delta Protocol (RRDP) uses conventional HTTP URI syntax, and can be supported using CDNs and replicated publication that use either anycast or DNS steering of clients to the nearest server.

RRDP is designed to scale much better than RSYNC. In particular, RRDP is designed to allow use of an HTTPS caching infrastructure to reduce load on primary Repository Servers and increase resilience against denial-of-service attacks on the RPKI publication service.

RPKI Publishers, Clients and Operational Resilience

The RPKI environment is a somewhat different security environment that other applications of secure validation of the authenticity of information. Typically, clients perform an on-demand validation function on a single object, and the upper layer transaction is typically blocked until the validation function returns a result. This occurs in the validation of credentials in a TLS handshake, or in the validation of a signed DNS response in the case of DNSSEC.

The goal of the routing system is to flood all routing information to all parts of the network at all times. The aim of the RPKI framework is to provide an efficient means for all clients to validate this routing information using the credentials published by all RPKI publishers.

For a publisher, *reslience* implies that all RPKI clients need to be able to reliably access the information published by all RPKI publishers at all times. Similarly, all RPKI publishers need to publish their information in a way that all clients can efficiently access this published information.

In other realms, such as the DNS, or in web content, there has emerged in some realms a push for service self-sufficiency at a level of an economy or a region. For example, all DNS nameservers for the top level domain of ".xx" should also use names drawn from ".xx", be physically located in the geography defined by the domain ".xx". and preferably local copies of the DNS root zone would also be located within the same geography. That way were all forms of external connectivity to be disrupted, there would still be adequate local service to allow local DNS operations to continue to function, or so goes the thinking behind this.

Do the same considerations apply to the RPKI? Not really. In the extreme scenario of comprehensive failure of external connectivity, the inability to access various RPKI publication points would presumably cause local RPKI client caches to expire, and this, in turn, would cause RPKI validation to fail. However, as noted above, such a failure does not cause an RPKI validation function to deem routes "invalid". The failure mode of RPKI is a reversion to "unknown", which implies that local RPKI-aware routers would continue to accept whatever BGP routes were being announced under such circumstances. There is no ulterior goal of self sufficiency or enhanced operational resilience to be served in attempting to localise instances of RPKI credentials and trust anchors.

Conclusions

While the RPKI is an instance of a distributed data framework, the considerations relating to its resilience and efficiency of operation are somewhat unique.

It appears that best current approach to ensure the broad and continuous availability of published RPKI credential material is to use the RPKI Repository Delta Protocol in conjunction with content distribution networks and take advantage of their ability to replicate the material across a large scale distribution network, and use either anycast or DNS-based steering to steer the client's URL retrieval requests to the closest instance of the distribution network. This approach maximises the availability of the RPKI material for the diverse set of RPKI client instances.

The design of this system was mandated by a very strong position from vendors of BGP router implementations when this system was being designed, that it was to operate as an overlay to the existing BGP framework, and BGP functionality was not to be changed. That has implied that the RPKI credential system, where publishers need to flood their published information to all clients, cannot take advantage of one of the most efficient flooding protocols we know, namely BGP itself. Instead, we have had to revert to a less efficient system where clients need to periodically poll publishers to detect if the publisher has changed its published material. I suspect that the current design and operational practice in RPKI credential distribution has achieved pretty much all that it can achieve within the constraints of this overall approach. If we want to state aspirations of greater scalability, faster responses and greater operational resilience then I suspect that we will need to question this very basic stricture of being unable to use BGP itself to perform this security credential flooding function.

So far RRDP and the selective use of CDNs has reached a generally acceptable position with respect to these parameters of operational performance and resilience. However, if we change our expectations, or impose new roles on this RPKI framework, then doubtless we will need to come back and evaluate just how far we can push the current model and when it is worth embarking on a completely different approach.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net