

December 2025

RPKI Signed Checklists

Exploring the potential for stronger trust models

Table of Contents

Table of Contents	1
Acknowledgements	2
Executive Summary	3
Background: The Evolution Toward Resource-Linked Attestations	4
Letters of Authority (LOA): A Trust-Based Model	4
Internet Routing Registry (IRR): A Decentralised and Inconsistent System	4
Resource Public Key Infrastructure (RPKI): A Foundation for Secure Assertions	5
Resource Tagged Attestation (RTA)	6
RPKI Signed Checklist (RSC): Design Principles and Practical Implementation	7
Design Philosophy: Simplicity and Pragmatism	7
How RSC Works: Step-by-Step Process	8
Use Cases for RSC	12
Operational Adoption and Tooling	13
Legal	14
Trust, Intent, and Verification	14
Automation and Interoperability Reduce Burden	14
Business Reluctance and the Precedent of ROAs	15
Start with Cloud Providers, Not Telcos	15
Define and Mitigate Ecosystem Risks	16
Lower Operational and Legal Burden for RIRs	16
Scope of Responsibility and Disclaimers	16
No Exposure of Confidential Business Information	17
Way Forward: Advancing RSC Towards Operational Deployment by RIRs	17
Vision	17
Strategic Objectives	17
Action Plan	18
Acronyms	20

Acknowledgements

First and foremost, we would like to thank the American Registry for Internet Numbers (ARIN), which funded this work under the ARIN Community Grant Program 2024. This report would not have been possible without the contributions, insights, and support of several individuals and organisations who helped shape its direction and content.

Special thanks to:

- Aftab Siddiqui from APNIC foundation, as the principal author and lead investigator of this research study.
- Tom Harrison from APNIC, for his detailed work on RPKI Signed Checklists (RSC) and early implementation insights that informed much of the technical framing.
- Koen van Hove and Martin Hoffmann from NLnet Labs, particularly for early RTA tooling that inspired RSC evolution.
- Prof. Christopher S. Yoo from the University of Pennsylvania Carey Law School, for his legal perspectives on digital attestations and trust models.
- Taejoong (Tijay) Chung and Weitong Li from Virginia Tech, for their research and ongoing work on RPKI validation, deployment trends, and transparency
- Amreesh Phokeer from Internet Society, for his support and guidance throughout the development of this work.
- Colleagues and peers from the various Network Operator Group (NOG) and Regional Internet Registry (RIR) communities, who shared real-world use cases through targeted Surveys and helped validate assumptions from an operational standpoint.

Executive Summary

The Resource Public Key Infrastructure (RPKI) was developed to solve the foundational problem of verifying resource ownership and authorization in routing. RPKI allows Internet Number Resource (INR) holders, such as IP address or ASN assignees, to get cryptographically signed certificates from Regional Internet Registries (RIRs) that bind public keys to specific resources or objects.

RPKI Signed Checklists (RSCs) are one such object that provides a powerful, standards-based alternative to legacy authorization models, such as Letter of Authorizations (LOAs) and the Internet Routing Registry (IRR) system.

Results from an Internet Society survey indicate that RSCs are well-positioned to replace or simplify legacy LOA processes if implemented correctly, particularly in environments where automation and auditability are important. However, to support meaningful adoption at scale, several gaps must be addressed, including:

- Making tools, documentation, and best practices clearer and easier to follow for both technical and non-technical audiences.
- Reducing operational complexity, particularly around integration with existing RPKI systems.

Importantly, while 50% of respondents were open to testing RSCs in sandbox or pilot environments, the other half remain cautious, seeking more clarity around business value, compliance, and legal implications.

To unlock RSC's full potential, RIRs must move beyond certificate issuance and provide user-friendly services for RSC creation and validation. By engaging the community, refining tooling, and demonstrating real-world impact, RSCs can move from a promising concept to a widely adopted operational tool across the global Internet infrastructure.

Background: The Evolution Toward Resource-Linked Attestations

As the Internet scaled globally, routing relationships between networks became increasingly complex. These relationships, ranging from transit to peering and sub-allocation, required mechanisms to establish and verify which Autonomous Systems (ASes) were authorized to announce specific Internet Protocol (IP) prefixes. In the absence of a cryptographically verifiable global standard, network operators relied on a combination of informal documents and unverified registries, most notably Letters of Authority (LOA) and the Internet Routing Registry (IRR) system. Over time, the limitations of these legacy models became apparent, prompting the search for more secure alternatives.

Letters of Authority (LOA): A Trust-Based Model

For decades, LOAs have been the most common method of proving authorization to announce an IP prefix. A LOA is typically a hand-signed document, usually sent as a PDF file via email or just an email, issued by the legitimate resource holder to authorize another party, such as an upstream provider, to originate a route on their behalf.

While widely used, the LOA model is fundamentally trust-based and operationally fragile due to:

- There is no standardized format or delivery mechanism.
- Authentication of the sender is rarely enforced.
- Verification is manual, making it prone to oversight or social engineering.
- Revocation requires out-of-band coordination and is often overlooked.

The [2015 hijack of a /16 IPv4 block from IIJ \(Internet Initiative Japan\)](#) illustrates the failure of this model. A forged LOA was used to convince an ISP to announce the prefix, and the false route propagated globally for several days before being withdrawn. The incident was only uncovered due to operator vigilance and required multiple manual interventions.

Such examples reveal that, while operationally convenient, LOAs are easy to falsify and difficult to audit, especially in environments where multiple parties are involved in sub-allocation or leasing arrangements.

Internet Routing Registry (IRR): A Decentralised and Inconsistent System

To complement LOAs and support semi-automated filtering, many network operators were also using the IRR. The IRR system allows operators to publish structured route objects (e.g., route, route6, aut-num, aut-num6) that associate IP prefixes with authorized origin ASes. Tools like IRRToolSet and bgpq3/4 were commonly used to generate prefix lists from these databases.

However, the IRR system has critical limitations, including:

- **Fragmentation:** The IRR is a distributed system with no central authority. Databases such as RADb, ALTDB, NTTCOM, and RIR-maintained IRRs (e.g., RIPE, APNIC) operate independently and are often unsynchronized.
- **Unverified data:** Most IRRs permit any user to submit route objects without verifying that they control the IP space. As a result, bogus or conflicting entries are common.
- **Stale records:** Entries are rarely cleaned up. Even after IP space is transferred or decommissioned, outdated objects often remain in the system, leading to routing anomalies.
- **No cryptographic guarantees:** IRRs are based on plain-text, unauthenticated data. There is no digital signature, and no linkage to the legitimate resource holder's authority. If the IRR database is managed and operated by an RIR, then a level of trust is available because only resource holders can create objects; however, there are many IRR databases available.

These issues contribute to a growing lack of confidence in the IRR model. Operators are increasingly filtering IRR data through local policies or refusing to accept certain IRRs altogether. In some cases, IRR data directly contradicts valid RPKI data, highlighting the unreliability of the former.

Together, the LOA and IRR approaches fall short in providing automation, auditability, and security. Their limitations underscored the need for a mechanism that can deliver verifiable, machine-readable, and resource-bound assertions without relying on informal trust models.

Resource Public Key Infrastructure (RPKI): A Foundation for Secure Assertions

RPKI was developed to solve the foundational problem of verifying resource ownership and authorization in routing. It allows Internet Number Resource (INR) holders, such as IP address or Autonomous System Number (ASN) assignees, to get digital certificates from RIRs. These certificates bind public keys to specific resources, creating the basis for cryptographic validation.

The most widely used application of RPKI is the Route Origin Authorization (ROA). An ROA authorizes a specific ASN to originate a designated IP prefix. When validated and combined with RPKI-aware routers or validators, ROAs enable Route Origin Validation (ROV), providing protection against accidental misconfiguration and malicious prefix hijacking.

However, while ROAs address origin validation, they do not meet all operational needs. Specifically, RPKI lacks a mechanism for:

- Attesting to the delegation of resources in non-routing contexts.
- Signing arbitrary digital content (e.g., routing agreements, contracts).
- Supporting multi-party attestations across different holders of IP space.

These unmet needs inspired the development of a more flexible RPKI-based structure.

Resource Tagged Attestation (RTA)

To fill this gap, researchers and engineers, primarily from APNIC and NLnet Labs, proposed the Resource Tagged Attestation (RTA), described in draft-ietf-sidrops-rpki-rta-00.

The RTA specification aimed to enable:

- The signing of arbitrary digital artifacts (e.g., plain text, XML, configuration files).
- The attachment of resource tags (IP prefixes, ASNs) to those signed objects.
- The ability to include multiple signers, each attesting with their own RPKI certificates.
- A detached signature model, where the object and its attestation could be transmitted separately or bundled.

It was modelled as a Cryptographic Message Syntax (CMS) object and borrowed from existing structures in [RFC 6488], [RFC 5652], and detached signature practices in [RFC 5485] and [RFC 8358].

Limitations of the RTA Model

Despite its technical robustness, the RTA model faced several practical challenges that hindered its adoption, including:

- **Implementation complexity:** The requirement to handle multiple SignerInfo blocks, each with its own certificate and CRL, introduced significant overhead for implementers and validators.
- **Canonicalization burden:** RTA required canonicalization of input files (e.g., text, XML, HTML) to ensure consistent hashing across systems. This introduced risks of signature failure due to subtle formatting changes or platform differences.
- **Deviation from existing RPKI structures:** RTA did not fully align with the RPKI signed object profile in [RFC 6488], making integration with existing RPKI software more difficult.
- **Lack of compelling multi-signer use cases:** While the model allowed multi-party signatures, most real-world needs (e.g., BYOIP, sub-allocations, cloud deployments) required only single-party assertions.
- **No publication framework:** RTA was designed for out-of-band use (e.g., via email or HTTP), but lacked standardized mechanisms for publication, discovery, or validation within existing infrastructure.

Despite prototype implementations in Krill and Routinator, as well as early work by APNIC, RTA failed to gain widespread traction and was eventually deprioritised within the IETF SIDROPS working group.

Yet the core idea—binding signed digital artifacts to RPKI-validated resources—remained valuable. The operational community needed a solution that was simpler, deployable, and compatible with existing tooling.

This led to the emergence of the RPKI Signed Checklist (RSC) format.

RPKI Signed Checklist (RSC): Design Principles and Practical Implementation

The Internet Engineering Task Force (IETF) developed RSCs [RFC 9323] as a simplified yet powerful alternative to the Resource Tagged Attestation (RTA) model. It preserves the core objective of associating signed digital artefacts with specific INRs, while significantly reducing implementation complexity and improving usability in real-world scenarios.

Design Philosophy: Simplicity and Pragmatism

Where RTA aimed to be flexible and comprehensive, RSC was deliberately minimalist and practical. Its design was guided by several core principles, including:

- **Single-signer model:** Unlike RTA, RSC supports only one signer per object. This greatly simplifies CMS structure, validation logic, and operational workflows.
- **No canonicalization required:** RSC treats files as binary blobs; therefore, there is no need to canonicalize input. Hashing is performed on raw byte content, avoiding issues with encoding differences or line-ending variations.
- **Checklist format:** RSC allows a signer to assert a set of one or more file digests (i.e., checksums), optionally with file names. This allows batch attestation without coupling each file to its own signature.
- **External usage model:** Like RTA, RSC objects are not published in the global RPKI repository system. This gives operators flexibility in how they distribute and consume signed data, i.e., via APIs, web interfaces, email, or other channels.

RSCs are ideal for asserting authenticity and resource linkage of digital artifacts, using the same cryptographic trust infrastructure as ROAs—but applied to a broader context.

Technical Overview

An RSC object is a CMS signed object that includes:

- A digest algorithm (e.g., SHA-256).
- A list of filename-and-hash pairs (checklist).
- A resource block (ASN and/or IP prefixes) indicating the resources associated with the signer.
- A one-time-use End Entity (EE) certificate issued by the signer's RPKI Certification Authority (CA), binding the signature to specific resources.

Unlike other RPKI-signed objects (e.g., ROAs or manifests), RSCs omit the Subject Information Access (SIA) field from their certificates, as they are not intended to be published via RPKI repositories. This supports the model of private or context-specific distribution.

The checklist within an RSC can include:

- Named files, to verify the integrity of known files (e.g., "loa-acme-networks.pdf").
- Nameless digests to validate standalone content, such as payload hashes in APIs or JSON structures.

How RSC Works: Step-by-Step Process

The following steps are typically followed to create an RSC:

1. Prepare the Files

The resource holder identifies one or more digital files (e.g., routing authorization letters, configuration templates, JSON declarations) that they want to attest to.

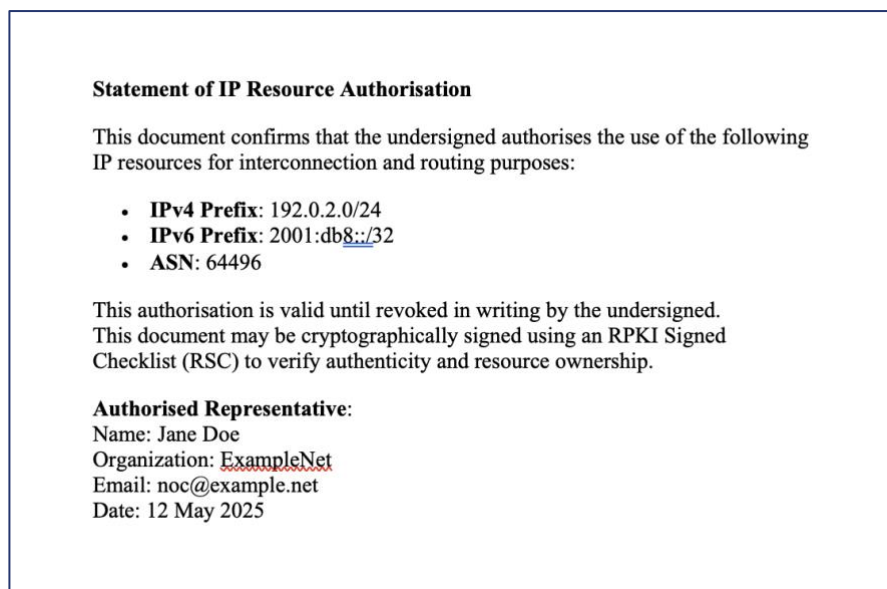
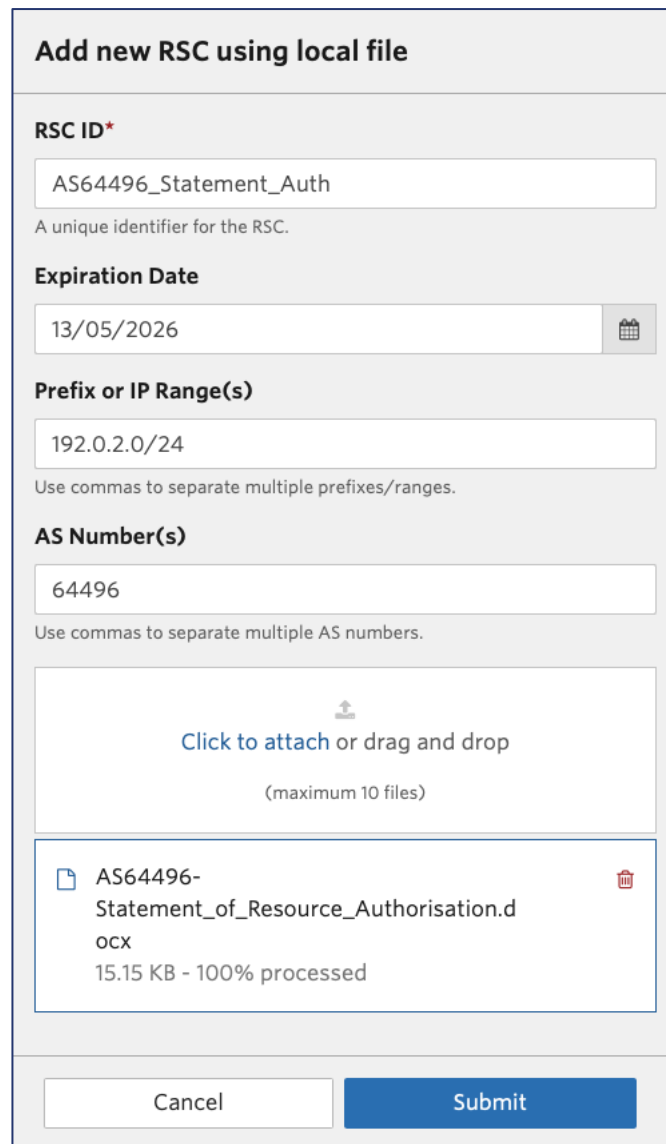


Figure 1: Sample File (.txt).

2. Compute File Hashes

A cryptographic hash (e.g., SHA-256) is calculated for each file. This ensures that any later modification of the file will cause validation to fail.



The screenshot shows a web form titled "Add new RSC using local file". It contains several input fields and a file upload section. The "RSC ID*" field has the value "AS64496_Statement_Auth" and a note "A unique identifier for the RSC.". The "Expiration Date" field has the value "13/05/2026" and a calendar icon. The "Prefix or IP Range(s)" field has the value "192.0.2.0/24" and a note "Use commas to separate multiple prefixes/ranges.". The "AS Number(s)" field has the value "64496" and a note "Use commas to separate multiple AS numbers.". Below these fields is a file upload area with a button "Click to attach or drag and drop" and the text "(maximum 10 files)". A file named "AS64496-Statement_of_Resource_Authorisation.docx" is shown as uploaded, with a size of "15.15 KB" and a status of "100% processed". At the bottom are "Cancel" and "Submit" buttons.

Add new RSC using local file

RSC ID*

AS64496_Statement_Auth

A unique identifier for the RSC.

Expiration Date

13/05/2026

Prefix or IP Range(s)

192.0.2.0/24

Use commas to separate multiple prefixes/ranges.

AS Number(s)

64496

Use commas to separate multiple AS numbers.

Click to attach or drag and drop
(maximum 10 files)

AS64496-Statement_of_Resource_Authorisation.docx
15.15 KB - 100% processed

Cancel Submit

Figure 2: APNIC Demo RSC Platform. Upload the file to create Hash.

3. Generate a One-Time EE Certificate

The resource holder's RPKI CA (typically via an RIR platform) issues a short-lived, one-time-use End-Entity (EE) certificate, which:

- Is bound to a specific ASN or IP prefix set,
- Is not reused,
- Has no Subject Information Access (SIA) extension, as the object is not published.

4. Create the Checklist

A checklist is created, consisting of one or more **fileName + hash** pairs. Each file may be named (e.g., **loa-203-0-113.pdf**) or represented by a hash only, depending on privacy preferences.

5. Sign the Checklist

Using the private key associated with the EE certificate, the checklist is signed in a CMS (Cryptographic Message Syntax) object. This creates the **.sig** file—the actual RSC.


 AS64496_Statement_Auth	AS64496-Statement_of_Resource_Authorisation.docx	pl36SYfbJz4ZiEfZaXygnVkckRZJfpSG6C5d2Ln+X4=	Download Revoke
Certificate Status Expires 2026-05-13			

Figure 3: .sig file created along with the Hash.

6. Distribute the RSC

The **.sig** file (RSC object) is shared out-of-band:

- Via email, API, or web interface,
- Alongside the original signed file.

7. Validate the RSC

A relying party (e.g., a cloud provider or upstream network) can:

- Extract the signature and checklist from the RSC,
- Use the included certificate to validate the signer's control over the specified resources,
- Re-compute file hashes and compare them to the signed values.

Step 1: Upload RSC

AS64496_Statement_Auth.sig Browse

Step 2: Specify digital objects

Specify digital objects by selecting the files or providing the filename (optional) and hash directly. Files selected here are not uploaded to APNIC's servers.

Files:

Choose files Browse

Filename and hash:

File name (optional) Remove

AS64496-Statement_of_Resource_Authorisation.docx

File hash*

pl36SYfbJz4ZiEfZaXygnVkckRZJfpSG6C5d2Ln+X4=

Add filename and hash

Verify

Figure 4: .sig file uploaded for verification along with the Hash or the original file.

✔ **Verification successful!**

Digital object verification succeeded.

Figure 5: Verification Status - Successful.

If all checks pass, this proves that:

- The file was signed by an entity that holds the claimed IP/ASN resources,
- The file has not been tampered with,
- The assertion (e.g., LOA, policy) was made by a valid resource holder.

Use Cases for RSC

RSC was designed to address operational scenarios where resource holders need to assert control, authorization, or verification over digital artefacts. Key use cases include:

Replacing LOAs in Routing Authorization

RSCs are ideal for replacing traditional LOAs by cryptographically signing a document that authorizes an upstream or peer to originate a route. Unlike static LOAs, an RSC:

- Can be verified automatically using standard RPKI tooling.
- Proves that the signer holds the IP prefix or ASN at the time of signing.
- Can be distributed as a.sig file alongside the original LOA or used to validate a hashed payload via a web interface.

Bring Your Own IP (BYOIP) in Cloud Environments

Cloud service providers require customers to prove ownership of IP prefixes before enabling BYOIP deployments. RSC allows customers to sign a set of files (e.g., JSON documents or declarations) to prove they control the relevant resources.

Cloud providers can automate the verification process by:

- Parsing the RSC.
- Validating its EE certificate and resource block.
- Verifying the hash of the customer-provided object.

This streamlines onboarding and eliminates the need for faxed LOAs or manual verification.

Verifying Objects in Third-Party Databases

Operators can use RSCs to prove the legitimacy of entries in:

- PeeringDB (e.g., verifying peering info based on ASN).
- Custom IRR or policy databases.
- Automation systems (e.g., provisioning pipelines or documentation repositories).

Rather than relying on usernames or email addresses, RSC enables proof via INR-holding cryptographic keys.

Geolocation Reporting to Third-Party Databases

Third-party IP geolocation providers such as Google, IPinfo, MaxMind, and others often allow resource holders to submit corrections or updates regarding the geolocation of their IP address blocks.

In parallel, RIRs now support geofeed as an IRR object, allowing resource holders to reference an external file (a geofeed) that contains location data (e.g., country, region, city) for their IP prefixes. This geofeed object indicates where resources are primarily used, helping geolocation providers align their databases accordingly.

However, IRR objects are not cryptographically validated, so there is no standardized way for geolocation providers to verify that the data truly originated from the legitimate resource holder.

Resource holders can sign the actual geolocation content, whether a geofeed CSV, JSON object, or other structured input, using an RSC. This signature proves:

- The data came from the current holder of the IP prefix, and
- The content has not been altered in transit.

By verifying the RSC, geolocation providers get a strong cryptographic assertion that the resource holder is legitimately submitting data for their own resources, closing a long-standing trust gap in geolocation workflows.

Internal Asset Management and Audit

Network operators managing multiple business units or customer allocations can sign documents and templates to assert delegation of resources or verify internal ownership structures. RSCs enable this to be done in a verifiable and easily detectable manner in the event of tampering, with optional public transparency.

Operational Adoption and Tooling

Internet Society conducted a survey in 2025 at various major regional NOG events to understand the awareness and understanding of RSCs. The results indicated:

- **Lack of awareness or understanding:** 60% of respondents indicated they were unfamiliar with how RSC works or how it fits into existing workflows.
- **Integration challenges:** 40% cited difficulty integrating RSC with their current RPKI infrastructure or automation pipelines.
- **Unclear business benefits:** 30% were unsure about the tangible operational or commercial value of implementing RSCs.
- **Regulatory and legal concerns:** Compliance and liability were raised as potential obstacles by a significant portion of respondents.
- **Willingness and need for testing:** Half of the respondents expressed openness to testing RSCs in a sandbox or pilot environment. The remaining half were more cautious, preferring to wait for further evaluation or clarity before engaging in testing or adoption.

Unlike ROAs, RSC objects are not publicly discoverable unless published intentionally. This makes them ideal for private business interactions, but also means that distribution responsibility falls on the signer, exactly as LOAs are traditionally shared.

Some tools and prototypes now support the creation and validation of RSCs, including:

- **rpki-client**: Open-source validator by OpenBSD with RSC validation support.
- **rpkimancer**: Python-based toolkit for crafting RPKI objects, including RSC.
- **APNIC's demo tools**: Web-based and CLI tools developed for proof-of-concept RSC signing and validation.

Legal and Administrative Considerations for RSC Deployment

As the Internet community moves toward more secure and automated validation of resource control, legal and administrative clarity is essential for widespread adoption. The RSC format, although new, builds upon established infrastructure and governance models already in place within the RPKI ecosystem.

The following considerations address concerns from legal, business, and operational standpoints, clarifying why RSC is not only feasible but also lower-risk compared to previous RPKI deployments.

Trust, Intent, and Verification

"There is no incentive for resource holders to provide misleading information via RSC, just as they wouldn't via an LOA (Letter of Authority). However, RSC offers stronger verification mechanisms that the information comes directly from the resource holder, irrespective of the content."

— Prof. Christopher S. Yoo, University of Pennsylvania Carey Law School

This framing is critical. RSC does not introduce a new trust model; rather, it strengthens existing workflows (such as LOAs) by anchoring them in cryptographic certainty. It ensures that a digital object, whether it's a routing authorization, declaration, or technical file, can be tied to a resource holder via their RPKI certificate, without making any assertions about the truth of the content itself.

This aligns with current legal principles, i.e., the responsibility for the accuracy of any claim lies with the signatory, not the platform that enables the signature.

Automation and Interoperability Reduce Burden

One major concern with any new validation mechanism is the operational overhead, particularly for business and legal teams who may be cautious about introducing friction into commercial processes.

RSC addresses this by supporting:

- Automation via standardised APIs,
- Consistency across RIR implementations, and
- Detached usage (i.e., no need for integration into global repositories).

By making RSC signing and verification machine-readable and automatable, the legal burden of processing and managing attestations decreases. It ensures businesses don't need to "reinvent" LOA workflows; instead, they can simply make them more verifiable.

Business Reluctance and the Precedent of ROAs

The initial hesitation to adopt RPKI ROAs and Route Origin Validation (ROV) was driven largely by a perception that "customers never asked for it." This narrative was disrupted once government procurement policies began requiring RPKI support, especially in cloud and telecom tenders.

RSC can follow the same path but with even less friction, since it:

- Doesn't alter routing policy or enforcement,
- Doesn't require validator or router integration, and
- Can be implemented entirely off-plane.

This makes it an attractive, low-risk option for incremental security enhancement that won't trigger business-side pushback.

Start with Cloud Providers, Not Telcos

Cloud providers have been among the earliest adopters of RPKI because of their:

- Extensive use of Bring Your Own IP (BYOIP),
- Focus on automation and scalability, and
- Regulatory exposure and audit requirements.

RSCs offer a streamlined approach for cloud providers to automate IP ownership verification during customer onboarding or upstream peering requests.

Rather than trying to convince large telecom operators, who are often slower-moving and policy-constrained, RSC adoption should begin with cloud platforms, which can then leverage their influence to pressure telcos toward acceptance.

Define and Mitigate Ecosystem Risks

To ensure RSC's reliability and defensibility, the ecosystem must be clearly mapped out. This includes answering:

- What tools exist to generate, publish, and verify RSCs?
- Who maintains those tools or libraries (e.g., RIRs, open-source communities)?
- What happens if a key service fails, such as key material expiry, API downtime, or validation mismatch?

RIRs and implementers should define:

- A reference validator,
- A simple support workflow, and
- A failure model outlining fallback or error-handling best practices.

This doesn't require the same high-availability concerns as live RPKI validation; RSCs are offline objects, but it still requires a support matrix and ownership model.

Lower Operational and Legal Burden for RIRs

Unlike ROAs or manifests, RSC objects don't require repository publication or affect global routing policy. This means RIRs are not liable for real-time availability or repository correctness. Instead, their role is limited to:

- Providing tooling to generate one-time-use EE certificates,
- Optionally offering a signing interface or API,
- Allowing validation via a common library or API endpoint.

As a result, the technical and legal risks for RIRs are lower than those introduced by live ROA/ROV enforcement. RSCs are simply an attestation format, not an enforcement mechanism.

Scope of Responsibility and Disclaimers

In any signing model, including RPKI, the content of the signed object is outside the scope of the infrastructure provider. The RIR's responsibility ends at:

- Issuing a valid certificate,
- Enabling signing operations, and
- Providing verification tooling.

To make this clear, RIRs should include disclaimers like those used in ROA/ROV deployments:

- “This RSC object has been signed using resources under your control. The content of the signed object has not been verified by [RIR].”
- “Verification confirms origin, not accuracy.”

This protects the RIR from disputes arising from misleading or invalid assertions made in RSC-covered documents.

No Exposure of Confidential Business Information

Some members may express concern that RSCs could expose sensitive peering or commercial information. This is unfounded.

RSC objects:

- Do not reveal contracts or pricing,
- Do not expose routing policies or filtering logic,
- Only confirm that “this entity that controls this prefix signed this file.”

In fact, compared to LOAs, which are often sent over email or shared with intermediaries. RSCs are more secure, more private (if not published), and less susceptible to tampering.

Way Forward: Advancing RSC Towards Operational Deployment by RIRs

Vision

Enable RIRs to offer RSC generation and validation services as part of their hosted RPKI platforms or as standalone tools, empowering members to replace insecure LOAs, streamline interconnect processes, and modernise routing authorization workflows.

Strategic Objectives

1. Standardize workflows for creating and validating RSCs.
2. Develop user-friendly tooling and API interfaces for RSC signing and validation.
3. Educate the membership on the limitations of LOAs/IRRs and the benefits of RSC.
4. Demonstrate real-world use cases.
5. Secure endorsement from technical communities (NOGs, IXP forums) and policy stakeholders.

Action Plan

Phase 1: Foundation Building and Advocacy

- Document Common Use Cases by creating brief, visually compelling case studies of how RSC can:
 - Replace LOAs (e.g., Acme ISP onboarding upstream).
 - Validate BYOIP for cloud services.
 - Prove prefix control in peering or leasing agreements..
- Build Alliances by partnering with:
 - RIR engineering and product teams (e.g., APNIC's early adoption is a reference point).
 - NOGs and IXPs to run pilot demonstrations.
 - Cloud providers and CDNs to highlight BYOIP use cases.
- Submit Policy Proposals or special interest group (SIG) presentations at RIR policy meetings or routing security SIGs to seed community awareness and create a record of public interest.

Phase 2: RIR Policy Framing

- **Frame the legal position clearly** by using authoritative messaging to assure RIRs and members that:
 - RSC does not expose RIRs to liability, and content responsibility lies with the signer.
 - RSCs are offline, optional, and low-risk. Unlike real-time enforcement tools like ROAs/ROVs.
 - Clear disclaimers and usage boundaries can shield RIRs from misuse of content.
- **Update Certification Practice Statements (CPS)** as Needed. This may require modifying language to accommodate one-time-use EE certificates for signing detached RSC objects without requiring repository publication or online availability guarantees.
- **Document roles and risk matrix** by identifying:
 - Which components (tools, certificates, validation logic) need to work?
 - What failure modes exist?
 - Who is responsible for the resolution (signer, validator, RIR)?
- **Highlight business and procurement alignment** by referencing the trajectory of RPKI ROA adoption:
 - Initially resisted by commercial teams.
 - Later adopted under government procurement mandates (especially in cloud and telco contracts).Position RSC similarly as a value-add with low integration overhead.

Phase 3: Prototype and Demonstration

- **Pilot program with volunteers** by recruiting early adopters to test RSC integration via:

- Web UI for uploading/creating artifacts and downloading.sig files
 - API endpoints for automated use (e.g., peering platforms, provisioning pipelines)
- **Open-source tooling and integration** by investing in:
 - Easy-to-use command-line tools (wrappers around rpkimancer, rpki-client).
 - Browser-based signing/verification interface.
 - Plugins for PeeringDB, IXP Manager, or IRR toolsets.
- **Demonstrate end-to-end workflows.** For example, "upload a routing contract and generate an RSC proving the signer controls 103.138.210.0/24."

Phase 4: Community Outreach and RIR Member Engagement

- **Prepare an RIR member engagement slide deck** that covers the:
 - Problem: LOAs are insecure, and IRR data is unreliable.
 - Solution: RSC - secure, verifiable, automated.
 - Benefits: Reduces operational friction, enables automation, and ensures trust.
- **Host tutorials and webinars** that showcase the creation, validation, and integration of RSC into daily operations. The target audiences for these should include:
 - LIRs and corporate members.
 - Managed service providers and cloud ops teams.
 - Routing and peering teams
- **Conduct a survey** to measure interest and readiness among RIR members to use an RSC-as-a-service platform.

Medium-Term Opportunities

- Create a universal RSC validator platform (hosted by an RIR or neutral org).
- Collaborate with PeeringDB to allow RSC verification of IP prefix ownership.
- Integrate RSC support into IXP route server configuration generators.
- Promote ASPA, RSC, and ROA as the modern replacements for LOA, IRR, and WHOIS.

Acronyms

APNIC (Asia Pacific Network Information Centre)	IJJ (Internet Initiative Japan)
API (Application Programming Interface)	INR (Internet Number Resource)
ARIN (American Registry for Internet Numbers)	IP (Internet Protocol; IPs, plural)
AS (Autonomous System; ASes, <i>plural</i>)	IRR (Internet Routing Registry)
ASN (Autonomous System Number, ASNs, plural)	ISP (Internet Service Provider)
ASPA (Autonomous System Provider Authorization)	IXP (Internet Exchange Point)
BGP (Border Gateway Protocol)	JSON (JavaScript Object Notation)
BYOIP (Bring Your Own IP)	LIRs (Local Internet Registries)
CA (Certification Authority)	LOA (Letter of Authority or Letter of Authorization)
CDN (Content Delivery Network)	NOG (Network Operator Group)
CLI (Command-Line Interface)	PDF (Portable Document Format)
CMS (Cryptographic Message Syntax)	RIR (Regional Internet Registry; RIRs, plural)
CPS (Certification Practice Statements)	ROA (Route Origin Authorization; ROAs, plural)
CRL (Certificate Revocation List)	ROV (Route Origin Validation)
CSV (Comma-Separated Values)	RPKI (Resource Public Key Infrastructure)
EE (End Entity)	RSC (RPKI Signed Checklist; RSCs, plural)
HTML (HyperText Markup Language - <i>implied in canonicalization context</i>)	RTA (Resource Tagged Attestation)
HTTP (Hypertext Transfer Protocol - <i>implied in RTA publication context</i>)	SIA (Subject Information Access)
IETF (Internet Engineering Task Force)	SIG (Special Interest Group)
	XML (eXtensible Markup Language)