Stream: Internet Engineering Task Force (IETF)
RFC: 9977
Category: Standards Track
Published: May 2026
ISSN: 2070-1721
Authors: O. Gasser, *IPinfo*; R. Bush, *IIJ Research & Arrcus*; M. Candela, *NTT*; R. Housley, *Vigil Security*

RFC 9977
Publishing End-Site Prefix Lengths

Abstract

This document specifies how to augment the Routing Policy Specification Language (RPSL) inetnum: class to refer specifically to prefixlen files, which are Comma-Separated Values (CSV) files used to specify end-site prefix lengths. This document also describes an optional mechanism that uses the Resource Public Key Infrastructure (RPKI) to authenticate the prefixlen files.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Internet Service Providers (ISPs) delegate IP addresses or entire IP prefixes to their users. Similarly, cloud providers assign customers who use their services, such as virtual machines (VMs), a prefix of a specific size. Therefore, there are many variations of end-site prefix lengths present in the Internet. Currently, there is no easy way for content providers to know the end-site prefix size of someone accessing their service. Knowing the correct end-site's prefix size has multiple implications such as:

Blocklisting/throttling: In IPv4, IP addresses can be blocked using variable prefix lengths for different prefixes, such as /22 for prefix A, /27 for prefix B, or /32 to block a single IPv4 address. Due to the large address space in IPv6, blocking at, e.g., the /48 or /56 level could lead to overblocking (throwing multiple VMs from different users into the same bucket), while blocking at more fine-granular levels, e.g., /64, /96, or even /128, to block a single IPv6 address would lead to filling up space in the blocklist pretty quickly. The use of temporary addresses in IPv6 [RFC8981] might lead to unwanted unblocking when addresses are blocked at a too-fine-granular level (e.g., /128). All these issues apply to throttling as well.

Rate limiting/CAPTCHAs: A similar issue arises on the Web, where neighboring prefixes might be thrown together (e.g., in the same /48 or /56 even though the ISP hands out /64s), which leads to people "jointly" running into rate limits and then either being blocked from a service or having to solve annoying CAPTCHAs.

Geolocation: Getting the right prefix size for geolocation is similarly difficult, especially for IPv6. If you aggregate too much, you throw together different clients in different locations; if you aggregate too little, you fill up the geolocation database with unnecessary entries.

This document specifies how to augment the Routing Policy Specification Language (RPSL) [RFC2725] inetnum: class to refer specifically to prefixlen files and how to use the files. In all places where inetnum: is used, inet6num: must also be assumed [RFC4012].

The reader may find [DBOBJECTS] informative, with certainly more verbose descriptions, on the inetnum: and inet6num database classes.

An optional means for authenticating prefixlen data is also defined in Section 6.

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. prefixlen Files

prefixlen files are CSV files [RFC4180] in text format with UTF-8 encoding [RFC3629], excluding problematic code points as described in [RFC9839]. Lines **MUST** be delimited by a line break (CRLF), and blank lines **MUST** be ignored. Text from a '#' character to the end of the current line **MUST** be treated as a comment only and is similarly ignored. The first field of each line that is not ignored specifies the prefix in question, the second the end-site prefix length within that prefix as an integer, and the third the number of end-sites within an end-site prefix length for networks using Carrier-Grade NAT (CGN) [RFC6598] or proxies. In all places Carrier-Grade NAT or CGN is used in this document, the specifications apply to proxies as well. Note that all three fields **MUST** be present. This means there **MUST** be exactly two commas in each non-commented line delimiting the three fields. The first field **MUST NOT** be empty on lines that are not comments, while the second and third field can be empty in certain scenarios. If both the second and third fields are empty, the publisher does not want to disclose any prefix length information.

3.1. End-Site Prefix Length Without CGN or Proxies

If an ISP delegates /56 IPv6 prefixes of the 2001:db8::/32 range and /32 IPv4 prefixes (i.e., a single IPv4 address) of the 192.0.2.0/24 range to its customers without the use of CGN [RFC6598] or proxy techniques, it would create a prefix length file containing the following example entries:

```
2001:db8::/32,56,1
192.0.2.0/24,32,1
```

Note the third field being set to '1', which signals the absence of CGN or proxies. This has the same meaning as the third field being left empty in this scenario.

3.2. End-Site Prefix Length with CGN or Proxies

prefixlen files can also be used to signal the presence of CGN [RFC6598] or proxies in networks. This is especially useful for cases where multiple end-sites behind a CGN or proxy service accessing a service at the same time might run into rate-limiting issues by service providers. If a prefixlen file signals the presence of a CGN, service providers can treat these prefixes in a way that rate limits are adjusted. To signal the presence of a CGN, the number of CGN end-sites is specified in the third field. For example, a CGN prefix 192.0.2.0/24 containing 4000 CGN end-sites would be specified as follows:

```
192.0.2.0/24,24,4000
```

Note the second field in the above example is set to '24', signaling that the 4000 CGN end-sites are present in the complete 192.0.2.0/24 prefix.

On the other hand, if these 4000 CGN end-sites are distributed 1000 each in the four /26 sub-prefixes within 192.0.2.0/24, this is specified as follows:

```
192.0.2.0/24,26,1000
```

It is important to note that the third field denoting the number of CGN end-sites is referring to the prefix length specified in the second field.

Note that this specification can be applied to IPv6 networks as well.

3.3. Longest Prefix Matching

Prefix length files can contain sub-prefix entries of a parent prefix; this needs to be taken into account when processing these files. For example, if a cloud provider assigns /120 IPv6 prefixes to each customer VM and a /64 prefix to premium customers, it would create a prefix length file containing the following example entries:

```
2001:db8::/32,120,
2001:db8:abcd::/48,64,
```

Note that the second entry in the above example is a subprefix of the first entry. Therefore, longest prefix matching has to be performed when parsing prefixlen files.

3.4. Not Specifying Any End-Site Prefix Length

If an ISP delegates /32 IPv4 prefixes (i.e., a single IPv4 address) of the 192.0.2.0/24 range to its customers without the use of CGN, and it has a special sub-prefix 192.0.2.0/28 where this policy does not apply, it can signal so with the following prefix length file:

```
192.0.2.0/24,32,
192.0.2.0/28,,
```

If both the second and third fields are empty, the publisher does not want to disclose any prefix length information. Any prefix length information from covering prefixes (192.0.2.0/24 in our example) **MUST** be discarded for sub-prefixes specified in prefixlen files (192.0.2.0/28 in our example).

3.5. Processing prefixlen Files

Multiple entries with exactly the same prefix **MUST** be considered an error, and consumer implementations **SHOULD** log the repeated entries for further administrative review. Publishers **MUST** take measures to ensure there is one and only one entry per prefix.

Upon encountering an erroneous entry in a prefixlen file, consumer implementations **MUST** skip that entry, log the error, and continue processing the remaining entries.

Content providers and other parties who wish to differentiate services based on end-site prefixes need to find the relevant prefixlen data. Section 4 specifies how to find the relevant prefixlen file given an IP address.

prefixlen data for large providers administrating a large number of networks and end-sites can contain millions of entries. The size of a file can be even larger if an unsigned prefixlen file combines data for many prefixes, if dual IPv4/IPv6 spaces are represented, etc.

This document also suggests an optional signature to strongly authenticate the data in the prefixlen files. The same approach to signatures is used in this document that was used in [RFC9632].

4. inetnum: Class

The original RPSL specifications ([RIPE81], [RIPE181], and a trail of subsequent documents) were written by the RIPE community. The IETF standardized RPSL in [RFC2622] and [RFC4012]. Since then, it has been modified and extensively enhanced in the Regional Internet Registry (RIR) community, mostly by RIPE [RIPE-DB]. At the time of publication, change control of RPSL effectively lies in the operator community.

The RPSL, and [RFC2725] and [RFC4012] used by the RIRs, specify the inetnum: database class. Each of these objects describes an IP address range and its attributes. The inetnum: objects form a hierarchy ordered on the address space.

Ideally, RPSL would be augmented to define a new RPSL prefixlen: attribute in the inetnum: class. Absent implementation of the prefixlen: attribute in a particular RIR database, this document defines the syntax of a prefixlen remarks: attribute, which contains an HTTPS URL of a prefixlen file. The format of the inetnum: prefixlen remarks: attribute **MUST** be as in this example, "remarks: Prefixlen ", where the token "Prefixlen" **MUST** be case-sensitive, followed by a URL that will vary but that **MUST** refer only to a single prefixlen file.

```
inetnum: 192.0.2.0/24 # example
remarks: Prefixlen https://example.com/prefixlen
```

While we leave global agreement of RPSL modification to the relevant parties, we specify that a proper prefixlen: attribute in the inetnum: class **MUST** be "prefixlen:" and **MUST** be followed by a single URL that will vary, but it **MUST** refer only to a single prefixlen file.

```
inetnum: 192.0.2.0/24 # example
prefixlen: https://example.com/prefixlen
```

The URL uses HTTPS, so the Web Public Key Infrastructure (WebPKI) provides authentication, integrity, and confidentiality for the fetched prefixlen file. However, the WebPKI cannot provide authentication of IP address space assignment. In contrast, the RPKI (see [RFC6481]) can be used to authenticate IP space assignment; see optional authentication in Section 6.

Until all producers of inetnum: objects, i.e., the RIRs, state that they have migrated to supporting the prefixlen: attribute, consumers looking at inetnum: objects to find prefixlen URLs **MUST** be able to consume the remarks: and prefixlen: forms.

The migration not only implies that the RIRs support the prefixlen: attribute, but that all registrants have migrated any inetnum: objects from remarks: to prefixlen:.

Any particular inetnum: object **SHOULD** have, at most, one prefixlen reference, whether a remarks: or prefixlen: attribute when it is implemented. As the remarks: form cannot be formally checked by the RIR, this cannot be formally enforced. A prefixlen: attribute is preferred, of course, if the RIR supports it. If there is more than one type of attribute in the inetnum: object, the prefixlen: attribute **MUST** be prioritized over the remarks: attribute.

For inetnum: instances covering the same address range, a signed prefixlen file **MUST** be preferred over an unsigned file. If none are signed, or more than one is signed, the (signed) inetnum: with the most recent last-modified: attribute **MUST** be preferred.

If a prefixlen file describes multiple disjoint ranges of IP address space, there are likely to be prefixlen references from multiple inetnum: objects. Files with prefixlen references from multiple inetnum: objects are not compatible with the signing procedure in Section 6.

An unsigned, and only an unsigned, prefixlen file **MAY** be referenced by multiple inetnum: instances and **MAY** contain prefixes from more than one registry.

When fetching, the most specific inetnum: object with a prefixlen reference **MUST** be used.

It is significant that prefixlen data may have finer granularity than the inetnum: that refers to them. For example, an inetnum: object for an address range P could refer to a prefixlen file in which P has been subdivided into one or more longer prefixes.

Backward-compatibility issues regarding the implementation of new RPSL attributes are covered by Section 10.2 of [RFC2622].

5. Fetching prefixlen Data

This document provides a guideline for how interested parties should fetch and read prefixlen files.

To minimize the load on RIRs' WHOIS [RFC3912] services, the RIR's bulk-download services **SHOULD** be used for large-scale access to gather inetnum: instances with prefixlen references. This uses efficient bulk access instead of fetching via brute-force search through the IP space. When using bulk-download services, they **MUST** be accessed using HTTPS [RFC9110]; FTP [RFC0959] **MUST NOT** be used.

On the other hand, RIRs are converging on RDAP support, which includes geofeed data; see [RFC9877]. It is hoped that this will be extended, or generalized, to support prefixlen data.

When reading data from a prefixlen file, one **MUST** ignore data outside the referring inetnum: object's address range. This is to avoid importing data about ranges not under the control of the operator. Note that signed files **MUST** only contain prefixes within the referring inetnum:'s range as mandated in Section 6.

If prefixlen files are fetched, other prefix length information from the inetnum: **MUST** be ignored.

Given an address range of interest, the most specific inetnum: object with a prefixlen reference **MUST** be used to fetch the prefixlen file. For example, if the fetching party finds the following inetnum: objects:

```
inetnum: 192.0.2.0/24 # example
remarks: Prefixlen https://example.com/prefixlen_1

inetnum: 192.0.2.0/26 # example
remarks: Prefixlen https://example.com/prefixlen_2
```

An application looking for prefixlen data for 192.0.2.0/29 **MUST** ignore data in prefixlen_1 because 192.0.2.0/29 is within the more specific 192.0.2.0/26 inetnum: covering that address range and that inetnum: does have a prefixlen reference.

6. Authenticating prefixlen Data (Optional)

The question arises whether a particular prefixlen data set is valid, i.e., is authorized by the "owner" of the IP address space and is authoritative in some sense. The inetnum: that points to the prefixlen file provides some assurance. Unfortunately, the RPSL in some repositories is weakly authenticated at best. An approach where RPSL was signed per the guidance in [RFC7909] would be good, except it would have to be deployed by all RPSL registries, and there is a fair number of them.

The remainder of this section specifies an optional authenticator for the prefixlen data set that follows the Signed Object Template for the Resource Public Key Infrastructure (RPKI) [RFC6488].

A single optional authenticator **MAY** be appended to a prefixlen file. It is a digest of the main body of the file signed by the private key of the relevant RPKI certificate for a covering address range. The following format bundles the relevant RPKI certificate with a signature over the prefixlen text.

The canonicalization procedure converts the data from their internal character representation to the UTF-8 character encoding (see [RFC3629]), and the <CRLF> sequence **MUST** be used to denote the end of each line of text. A blank line is represented solely by the <CRLF> sequence. For robustness, any non-printable characters **MUST NOT** be changed by canonicalization. Trailing blank lines **MUST NOT** appear at the end of the file. That is, the file must not end with multiple consecutive <CRLF> sequences. Any end-of-file marker used by an operating system is not considered to be part of the file content. When present, such end-of-file markers **MUST NOT** be covered by the digital signature.

If the authenticator is not in the canonical form described above, then the authenticator is invalid, which means that it is treated in the same manner as an unauthenticated prefixlen data.

Borrowing detached signatures from [RFC5485], after file canonicalization, the CMS (see [RFC5652]) is used to create a detached DER-encoded signature that is then Base64-encoded with padding (as defined in Section 4 of [RFC4648]) and line wrapped to 72 or fewer characters. The same digest algorithm **MUST** be used for calculating the message digest of the content being signed, which is the prefixlen file, and for calculating the message digest on the SignerInfo SignedAttributes (see [RFC8933]). The message digest algorithm identifier **MUST** appear in both the CMS SignedData DigestAlgorithmIdentifiers and the SignerInfo DigestAlgorithmIdentifier [RFC5652]. The RPKI certificate covering the prefixlen inetnum: object's address range is included in the CMS SignedData certificates field [RFC5652].

The address range of the signing certificate **MUST** cover all prefixes in the signed prefixlen file. If not, the authenticator is invalid.

The signing certificate **MUST NOT** include the Autonomous System Identifier Delegation certificate extension [RFC3779]. If it is present, the authenticator is invalid.

As with many other RPKI signed objects, the IP Address Delegation certificate extension **MUST NOT** use the "inherit" capability defined in Section 2.2.3.5 of [RFC3779]. If "inherit" is used, the authenticator is invalid.

An IP Address Delegation certificate extension using "inherit" would complicate processing. The implementation would have to build the certification path from the end-entity to the trust anchor and then validate the path from the trust anchor to the end-entity. Then, the parameter would have to be remembered when the validated public key was used to validate a signature on a CMS object. Having to remember things from certification-path validation for use with CMS object processing would be quite complex and error-prone. And, the certificates do not get that much bigger by repeating the information.

An address range A "covers" address range B if the range of B is identical to or a subset of A. "Address range" is used here because inetnum: objects and RPKI certificates need not align on Classless Inter-Domain Routing (CIDR) [RFC4632] prefix boundaries, while those of the lines in a prefixlen file do align.

The Certification Authority (CA) **MUST** generate a new End Entity (EE) certificate for each signing of a particular prefixlen file. The private key associated with the EE certificate **SHOULD** sign only one prefixlen file. That is, a new key pair **SHOULD** be generated for each new version of a particular prefixlen file. When the EE certificate is used in this fashion, it is termed a "one-time-use" EE certificate (see Section 3 of [RFC6487]).

On the other hand, verifying the signature has no similar complexity; the certificate, which is validated in the RPKI, contains the needed public key. The RPKI trust anchors for the RIRs are available to the party performing signature validation. Validation of the CMS signature over the prefixlen file involves:

1. Obtaining the signer's certificate from the CMS SignedData CertificateSet [RFC5652]. The certificate SubjectKeyIdentifier extension [RFC5280] **MUST** match the SubjectKeyIdentifier in the CMS SignerInfo SignerIdentifier [RFC5652]. If the key identifiers do not match, then validation **MUST** fail.
2. Validating the signer's certificate **MUST** ensure that it is part of the current manifest per [RFC9286] and that all resources are covered by the RPKI certificate.

3. Constructing and validating the certification path for the signer's certificate. All of the needed certificates are expected to be readily available in the RPKI repository. The certification path **MUST** be valid according to the validation algorithm in [RFC5280] and the additional checks specified in [RFC3779] associated with the IP Address Delegation certificate extension. If certification path validation is unsuccessful, then validation **MUST** fail.
4. Validating the CMS SignedData as specified in [RFC5652] using the public key from the validated signer's certificate. If the signature validation is unsuccessful, then validation **MUST** fail.
5. Confirming that the eContentType Object IDentifier (OID) is id-ct-prefixlenCSVwithCRLF (1.2.840.113549.1.9.16.1.57). This OID **MUST** appear within both the eContentType in the encapContentInfo object and the ContentType signed attribute in the signerInfo object (see [RFC6488]).
6. Verifying that the IP Address Delegation certificate extension [RFC3779] covers all of the address ranges of the prefixlen file. If all of the address ranges are not covered, then validation **MUST** fail.

All of the above steps **MUST** be successful to consider the prefixlen file signature to be valid.

The authenticator **MUST** be hidden as a series of "#" comments at the end of the prefixlen file. The following simple example is cryptographically incorrect:

```
# RPKI Signature: 192.0.2.0 - 192.0.2.255
# MIIGlwYJKoZIhvcNAQcCoIIGiDCCBoQCAQMxDTALBglghkgBZQMEAgEwDQYLKoZ
# IhvcNAQkQAS+gggSxMIIErTCCA5WgAwIBAgIUJ605QIPX8rW5m4Zwx3WyuW7hZu
...
# imwYkXpiMxw44EZqDjl36MiWsRDLdgoijBBcGbibwyAfGeR46k5raZCGvxG+4xa
# O8PDTxTfIYwAnBjRBKAqAZ7yX5xHfm58jUXsZJ7Ileq1S7G6Kk=
# End Signature: 192.0.2.0 - 192.0.2.255
```

A correct and full example is in Appendix A.

The CMS signature does not cover the signature lines.

The bracketing "# RPKI Signature:" and "# End Signature:" **MUST** be present as shown in the example. The RPKI Signature's IP address range **MUST** match that of the prefixlen URL in the inetnum: that points to the prefixlen file.

7. Operational Considerations

To create the needed inetnum: objects, an operator wishing to register the location of their prefixlen file needs to coordinate with their Regional Internet Registry (RIR) or National Internet Registry (NIR) and/or any provider Local Internet Registry (LIR) that has assigned address ranges to them. RIRs/NIRs provide means for assignees to create and maintain inetnum: objects. They also provide means of assigning or sub-assigning IP address resources and allowing the assignee to create WHOIS data, including inetnum: objects, thereby referring to prefixlen files.

The prefixlen files **MUST** be published via and fetched using HTTPS [RFC9110].

When using data from a prefixlen file, one **MUST** ignore data outside the referring inetnum: object's inetnum: attribute address range.

If and only if the prefixlen file is not signed per Section 6, then multiple inetnum: objects **MAY** refer to the same prefixlen file, and the consumer **MUST** use only lines in the prefixlen file where the prefix is covered by the address range of the inetnum: object's URL it has followed.

If the prefixlen file is signed, and the signer's certificate is replaced with another certificate, then the signature in the prefixlen file **MUST** be updated so that it can be properly validated with the new certificate.

It is good key hygiene to use a given key for only one purpose. To dedicate a signing private key for signing a prefixlen file, an RPKI Certification Authority (CA) may issue a subordinate certificate exclusively for the purpose shown in Appendix B.

Harvesting and publishing aggregated prefixlen data outside of the RPSL model **SHOULD** be avoided: it can have the effect that more specifics from one aggregatee could undesirably affect the less specifics of a different aggregatee. Moreover, publishing aggregated prefixlen data prevents the reader of the data to perform the checks described in Sections 5 and 6.

An anonymized version of bulk WHOIS data is openly available for all RIRs except ARIN, which requires an authorization. However, for users without such authorization, the same result can be achieved with extra RDAP effort. There is open-source code to pass over such data across all RIRs, collect all prefixlen references, and process them [PREFIXLEN-FINDER].

To prevent undue load on RPSL and prefixlen servers, entity-fetching prefixlen data using these mechanisms **MUST NOT** do frequent real-time lookups. prefixlen servers **SHOULD** send an HTTP Expires header [RFC9111] to signal when prefixlen data should be refetched. If an HTTP Expires or Cache-Control header is present, it **MUST** be honored by clients. As the data change very infrequently, in the absence of such an HTTP header signal, collectors **SHOULD NOT** fetch more frequently than weekly. It would be polite not to fetch at magic times such as midnight UTC, the first of the month, etc., because too many others are likely to do the same.

8. Implementation Status

As of November 2025, the prefixlen: attribute in inetnum objects has been implemented by the RIPE NCC database.

Registrants in databases that do not yet support the prefixlen: attribute are using the remarks:, or equivalent, attribute.

At the time of publication, the registry data published by ARIN are not the same RPSL as that of the other registries (see [RFC7485] for a survey of the WHOIS Tower of Babel); therefore, when fetching via bulk WHOIS over HTTPS [RFC9110], WHOIS [RFC3912], the Registration Data Access Protocol (RDAP) [RFC9083], etc., the "NetRange" or "ip network" attribute/key must be treated as "inetnum" and the "Comment" attribute must be treated as "remarks".

9. Security Considerations

The consumer of prefixlen data **SHOULD** fetch and process the data themselves. Importing datasets produced and/or processed by a third party places significant trust in the third party.

As mentioned in Section 6, some RPSL repositories have weak, if any, authentication. This allows spoofing of inetnum: objects pointing to malicious prefixlen files. Section 6 suggests an unfortunately complex method for stronger authentication based on the RPKI.

For example, if an inetnum: for a wide address range (e.g., a /16) points to an RPKI-signed prefixlen file, a customer or attacker could publish an unsigned equal or narrower (e.g., a /24) inetnum: in a WHOIS registry that has weak authorization, abusing the rule that the most-specific inetnum: object with a prefixlen reference **MUST** be used.

If signatures were mandatory, the above attack would be stymied, but, of course, that is not happening anytime soon.

The RPSL providers have had to throttle fetching from their servers due to too-frequent queries. Usually, they throttle by the querying IP address or block. Similar defenses will likely need to be deployed by prefixlen file servers.

As prefixlen files disclose which parts of a prefix belong to an end-site, attackers could better focus DDoS traffic towards a website hosted by a cloud provider by overwhelming only IP addresses from that specific end-site. Furthermore, information collected from prefixlen files could allow for more targeted IPv6 scanning/reconnaissance, where scanners (be it benevolent or malicious ones) can target specific sub-prefixes that they deem more interesting.

It is possible for publishers of prefixlen data to specify incorrect prefixlen data about their prefixes. This could be done either by mistake or on purpose. One example could be a malicious network operator trying to overflow the storage of databases that consume prefixlen data by setting a very specific prefix size (e.g., /128 for large blocks of IPv6 address space). In another example, a network operator might annotate their prefixes as using CGN to go around legitimate blocking or throttling. A third example would be a malicious provider publishing fake small allocations, so on receipt of complaints, they could plausibly respond by saying that they stopped the actions of a bad customer and move their malicious activities to a different prefix. As a fourth example, network operators could overwhelm consumers by publishing prefixlen files containing millions or even billions of entries (e.g., enumerating all possible /96 subprefixes of a /32 IPv6 prefix). Therefore, care should be taken when processing prefixlen data, as with any external third-party data.

10. IANA Considerations

IANA has registered an object identifier for one ASN.1 Module in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry as follows:

Description	OID	Reference
id-mod-prefixlen-2025	1.2.840.113549.1.9.16.0.87	RFC 9977

Table 1

IANA has registered an object identifier for one content type in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry as follows:

Description	OID	Reference
id-ct-prefixlenCSVwithCRLF	1.2.840.113549.1.9.16.1.57	RFC 9977

Table 2

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, DOI 10.17487/RFC2622, June 1999, <https://www.rfc-editor.org/info/rfc2622>.

[RFC2725] Villamizar, C., Alaettinoglu, C., Meyer, D., and S. Murphy, "Routing Policy System Security", RFC 2725, DOI 10.17487/RFC2725, December 1999, <https://www.rfc-editor.org/info/rfc2725>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <https://www.rfc-editor.org/info/rfc3779>.

[RFC4012] Blunk, L., Damas, J., Parent, F., and A. Robachevsky, "Routing Policy Specification Language next generation (RPSLng)", RFC 4012, DOI 10.17487/RFC4012, March 2005, <https://www.rfc-editor.org/info/rfc4012>.

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <https://www.rfc-editor.org/info/rfc4180>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC6268] Schaad, J. and S. Turner, "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)", RFC 6268, DOI 10.17487/RFC6268, July 2011, <https://www.rfc-editor.org/info/rfc6268>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <https://www.rfc-editor.org/info/rfc6487>.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <https://www.rfc-editor.org/info/rfc6488>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8933] Housley, R., "Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection", RFC 8933, DOI 10.17487/RFC8933, October 2020, <https://www.rfc-editor.org/info/rfc8933>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9286] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 9286, DOI 10.17487/RFC9286, June 2022, <https://www.rfc-editor.org/info/rfc9286>.

[RFC9839] Bray, T. and P. Hoffman, "Unicode Character Repertoire Subsets", RFC 9839, DOI 10.17487/RFC9839, August 2025, <https://www.rfc-editor.org/info/rfc9839>.

[X680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680>.

11.2. Informative References

[DBOBJECTS] RIPE NCC, "Descriptions of Primary Objects", <https://docs.db.ripe.net/RPSL-Object-Types/Descriptions-of-Primary-Objects>.

[PREFIXLEN-FINDER] "prefixlen-finder", commit 5f446617796bc17d7e9495513537438ec26ab8e6, May 2026, <https://github.com/massimocandela/prefixlen-finder>.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <https://www.rfc-editor.org/info/rfc959>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC5485] Housley, R., "Digital Signatures on Internet-Draft Documents", RFC 5485, DOI 10.17487/RFC5485, March 2009, <https://www.rfc-editor.org/info/rfc5485>.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, DOI 10.17487/RFC6598, April 2012, <https://www.rfc-editor.org/info/rfc6598>.

[RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin, "Inventory and Analysis of WHOIS Registration Objects", RFC 7485, DOI 10.17487/RFC7485, March 2015, <https://www.rfc-editor.org/info/rfc7485>.

[RFC7909] Kisteleki, R. and B. Haberman, "Securing Routing Policy Specification Language (RPSL) Objects with Resource Public Key Infrastructure (RPKI) Signatures", RFC 7909, DOI 10.17487/RFC7909, June 2016, <https://www.rfc-editor.org/info/rfc7909>.

[RFC8805] Kline, E., Duleba, K., Szamonek, Z., Moser, S., and W. Kumari, "A Format for Self-Published IP Geolocation Feeds", RFC 8805, DOI 10.17487/RFC8805, August 2020, <https://www.rfc-editor.org/info/rfc8805>.

[RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <https://www.rfc-editor.org/info/rfc8981>.

[RFC9083] Hollenbeck, S. and A. Newton, "JSON Responses for the Registration Data Access Protocol (RDAP)", STD 95, RFC 9083, DOI 10.17487/RFC9083, June 2021, <https://www.rfc-editor.org/info/rfc9083>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9632] Bush, R., Candela, M., Kumari, W., and R. Housley, "Finding and Using Geofeed Data", RFC 9632, DOI 10.17487/RFC9632, August 2024, <https://www.rfc-editor.org/info/rfc9632>.

[RFC9877] Singh, J. and T. Harrison, "Registration Data Access Protocol (RDAP) Extension for Geofeed Data", RFC 9877, DOI 10.17487/RFC9877, October 2025, <https://www.rfc-editor.org/info/rfc9877>.

[RIPE-DB] RIPE NCC, "RIPE Database Documentation", <https://www.ripe.net/manage-ips-and-asns/db/support/documentation/ripe-database-documentation>.

[RIPE181] Bates, T., Gerich, E., Joncheray, L., Jouanigot, J., Karrenberg, D., Terpstra, M., and J. Yu, "Representation Of IP Routing Policies In A Routing Registry", RIPE-181, October 1994, <https://www.ripe.net/publications/docs/ripe-181>.

[RIPE81] Bates, T., Jouanigot, J., Karrenberg, D., Lothberg, P., and M. Terpstra, "Representation Of IP Routing Policies In The RIPE Database", RIPE-081, February 1993, <https://www.ripe.net/publications/docs/ripe-081>.

Appendix A. ASN.1 Module

This appendix provides an ASN.1 Module [X680] for the CMS content type used for the prefixlen file.

CONTENT-TYPE is imported from the ASN.1 Module in [RFC6268].

```
<CODE BEGINS>
   PrefixLengthsModule-2025
     { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs9(9) smime(16) mod(0) 87 }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   -- EXPORTS ALL --

   IMPORTS
     CONTENT-TYPE
     FROM CryptographicMessageSyntax-2010 -- in [RFC6268]
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
       pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) } ;

   ContentSet CONTENT-TYPE ::= { ct-prefixlenCSVwithCRLF, ... }

   ct-prefixlenCSVwithCRLF CONTENT-TYPE ::=
     { TYPE UTF8String IDENTIFIED BY id-ct-prefixlenCSVwithCRLF }

   id-ct-prefixlenCSVwithCRLF OBJECT IDENTIFIER ::=
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
       pkcs-9(9) smime(16) ct(1) 57 }

   END
<CODE ENDS>
```

Appendix B. Example

This appendix provides an example, including a trust anchor, a Certificate Revocation List (CRL) signed by the trust anchor, a CA certificate subordinate to the trust anchor, a CRL signed by the CA, an end-entity certificate subordinate to the CA for signing the prefixlen file, and a detached signature.

The trust anchor is represented by a self-signed certificate. As usual in the RPKI, the trust anchor has authority over all IPv4 address blocks, all IPv6 address blocks, and all Autonomous System (AS) numbers.

```
-----BEGIN CERTIFICATE-----
MIIEQTCCAymgAwIBAgIUEggycNoFVRjAuN/Fw7URu0DEZNAwDQYJKoZIhvcNAQEL
BQAwFTETMBEGA1UEAxMKZXhhbXBsZS10YTAeFw0yMzA5MTkyMDMzMzlaFw0zMzA5
MTYyMDMzMzlaMBUxEzARBgNVBAMTCmV4YW1wbGUtdGEwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQDQprR+g/i4JyObVURTp1JpGM23vGPyE5fDKFPqV7rw
M1Amm7cnew66U02IzV0X5oiv5nSGfRX5UxsbR+vwPBMceQyDgS5lexFiv4fB/Vjf
DT2qX/UjsLL9QOeaSOh7ToJSLjmtpa0D9iz7ful3hdxRjpMMZiE/reX9/ymdpW/E
dg0F6+T9WGZE1miPeIjl5OZwnmLHCftkN/aaYk1iPNjNniHYIOjC1jSpABmoZyTj
sgrwLE2F1fIRkVkwASqToq/D5v9voXaYYaXUNJb4H/5wenRuvT5O/n6PXh70rMQy
F5yzLs96ytxqg5gGX9kabVnvxFU8nHfPa0rhlwfTJnljAgMBAAGjggGHMIIBgzAd
BgNVHQ4EFgQUwL1SXb7SeLIW7LOjQ5XSBguZCDIwHwYDVR0jBBgwFoAUwL1SXb7S
eLIW7LOjQ5XSBguZCDIwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYw
GAYDVR0gAQH/BA4wDDAKBggrBgEFBQcOAjCBuQYIKwYBBQUHAQsEgawwgakwPgYI
KwYBBQUHMAqGMnJzeW5jOi8vcnBraS5leGFtcGxlLm5ldC9yZXBvc2l0b3J5L2V4
YW1wbGUtdGEubWZ0MDUGCCsGAQUFBzANhilodHRwczovL3JyZHAuZXhhbXBsZS5u
ZXQvbm90aWZpY2F0aW9uLnhtbDAwBggrBgEFBQcwBYYkcnN5bmM6Ly9ycGtpLmV4
YW1wbGUubmV0L3JlcG9zaXRvcnkvMCcGCCsGAQUFBwEHAQH/BBgwFjAJBAIAATAD
AwEAMAkEAgACMAMDAQAwIQYIKwYBBQUHAQgBAf8EEjAQoA4wDDAKAgEAAgUA////
/zANBgkqhkiG9w0BAQsFAAOCAQEAa9eLY9QAmnlZOIyOzbpta5wqcOUQV/yR7o/0
1zkEZaSavKBt19lMK6AXZurx1T5jyjIwG7bEtZZThjtH2m80V5kc2tsFjSq/yp7N
JBclMHVd3tXse9If3nXYF4bxRIcir1lXlAbYN+Eo1U3i5qJO+fxouzt7Merk2Dih
nsenTeXKzN7tfmuCYZZHCC8viCoJWdH+o1uRM4TiQApZsUJ8sF4TABrrRJmA/Ed5
v0CTBbgqTx7yg0+VarFLPdnjYgtpoCJqwE2C1UpX15rZSaLVuGXtbwXd/cHEg5vF
W6QTsMeMQFEUa6hkicDGtxLTUdhckBgmCGoF2nlZii5f1BTWAg==
-----END CERTIFICATE-----
```


The CRL issued by the trust anchor.


```
-----BEGIN X509 CRL-----
MIIBjjB4AgEBMA0GCSqGSIb3DQEBCwUAMBUxEzARBgNVBAMTCmV4YW1wbGUtdGEX
DTI2MDUwNzIxMjI0OVoXDTI2MDYwNjIxMjI0OVqgLzAtMB8GA1UdIwQYMBaAFMC9
Ul2+0niyFuyzo0OV0gYLmQgyMAoGA1UdFAQDAgEMMA0GCSqGSIb3DQEBCwUAA4IB
AQCkHXyCcQHejmVdHOL5Diafa3ys4HTb2eRqeNaMzwfY6T1D26hX6XuUyu0C7LV2
OThlAL8JWiN2afgfs5juBAWdauwY5YSKAvQpXidFeCIXpSWLHmk545p7t9og6qpy
840l+N+J2WnP9iGNCqgKG06CiRAoPtZZQCqqLZVcrELtDAOFNmZF0Bf+cE2SmsZO
8N/ab/fw05Ptm/IBqN3j+ekaILELFRWUGPaAXMimWYn6sNmzYdihUn2fNff294PZ
Mygxfw8dpWlA01QQt8d9V+3NklyOKEB3X+X12eA4KYaVDCt4USWMlnlETNO3XwDe
Cg5BBjoh5EtXzsNWf2ipZTNb
-----END X509 CRL-----
```


The CA certificate is issued by the trust anchor. This certificate grants authority over one IPv4 address block (192.0.2.0/24), one IPv6 address block(2001:db8::/32), and two AS numbers (64496 and 64497).

```
-----BEGIN CERTIFICATE-----
MIIE+zCCA+OgAwIBAgIUcyCzS10hdfG65kbRq7toQAvRDMIwDQYJKoZIhvcNAQEL
BQAwFTETMBEGA1UEAxMKZXhhbXBsZS10YTAeFw0yNjA1MDcyMTIyNDlaFw0yNzA1
MDcyMTIyNDlaMDMxMTAvBgNVBAMTKDNBQ0UyQ0VGNEZCMjFCN0QxMUUzRTE4NEVG
QzFFMjk3QjM3Nzg2NDIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDc
zz1qwTxC2ocw5rqp8ktm2XyYkl8riBVuqlXwfefTxsR2YFpgz9vkYUd5Az9EVEG7
6wGIyZbtmhK63eEeaqbKz2GHub467498BXeVrYysO+YuIGgCEYKznNDZ4j5aaDbo
j5+4/z0Qvv6HEsxQd0f8br6lKJwgeRM6+fm7796HNPB0aqD7Zj9NRCLXjbB0DCgJ
liH6rXMKR86ofgll9V2mRjesvhdKYgkGbOif9rvxVpLJ/6zdru5CE9yeuJZ59l+n
YH/r6PzdJ4Q7yKrJX8qD6A60j4+biaU4MQ72KpsjhQNTTqF/HRwi0N54GDaknEwE
TnJQHgLJDYqww9yKWtjjAgMBAAGjggIjMIICHzAdBgNVHQ4EFgQUOs4s70+yG30R
4+GE78Hil7N3hkIwHwYDVR0jBBgwFoAUwL1SXb7SeLIW7LOjQ5XSBguZCDIwDwYD
VR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwGAYDVR0gAQH/BA4wDDAKBggr
BgEFBQcOAjBDBgNVHR8EPDA6MDigNqA0hjJyc3luYzovL3Jwa2kuZXhhbXBsZS5u
ZXQvcmVwb3NpdG9yeS9leGFtcGxlLXRhLmNybDBOBggrBgEFBQcBAQRCMEAwPgYI
KwYBBQUHMAKGMnJzeW5jOi8vcnBraS5leGFtcGxlLm5ldC9yZXBvc2l0b3J5L2V4
YW1wbGUtdGEuY2VyMIG5BggrBgEFBQcBCwSBrDCBqTA+BggrBgEFBQcwCoYycnN5
bmM6Ly9ycGtpLmV4YW1wbGUubmV0L3JlcG9zaXRvcnkvZXhhbXBsZS1jYS5tZnQw
NQYIKwYBBQUHMA2GKWh0dHBzOi8vcnJkcC5leGFtcGxlLm5ldC9ub3RpZmljYXRp
b24ueG1sMDAGCCsGAQUFBzAFhiRyc3luYzovL3Jwa2kuZXhhbXBsZS5uZXQvcmVw
b3NpdG9yeS8wLgYIKwYBBQUHAQcBAf8EHzAdMAwEAgABMAYDBADAAAIwDQQCAAIw
BwMFACABDbgwIQYIKwYBBQUHAQgBAf8EEjAQoA4wDDAKAgMA+/ACAwD78TANBgkq
hkiG9w0BAQsFAAOCAQEAipx10/ZrY11NYH+iVRtq32hAFGGaXysLWrjFVYd05+25
2nPtZYPmtLRf7TWMSwF27AkGPzvonjsRF2a7wdMAPDIW2nKctmDS1nFGWw+6vXyN
Di+jhwHm7+FyFWh3u2ilzop+o6ecUiCF8rkE22TWHRkBJforN0eqUjJi0R/o4oaB
q9sZs+Jr3vTmelRYjvP8Eej3AWRm+rilbP8yW3OOvV3sTvgJc4DmbFNJ2LBJ+cLx
1fjl+Wf/YHPo2kHw8f1TJsgXSI6kYBUradIyXIW1HGrWdiKiY+oXp+jVbf8cMvp/
KkLf1UqqCjgdu3GGQuukKjbNHeJPMuHmVw5Qa3iGzg==
-----END CERTIFICATE-----
```


The CRL issued by the CA.


```
-----BEGIN X509 CRL-----
MIIBrTCBlgIBATANBgkqhkiG9w0BAQsFADAzMTEwLwYDVQQDEygzQUNFMkNFRjRG
QjIxQjdEMTFFM0UxODRFRkMxRTI5N0IzNzc4NjQyFw0yNjA1MDcyMTIyNDlaFw0y
NjA2MDYyMTIyNDlaoC8wLTAfBgNVHSMEGDAWgBQ6zizvT7IbfRHj4YTvweKXs3eG
QjAKBgNVHRQEAwIBATANBgkqhkiG9w0BAQsFAAOCAQEAFEEWr/QvDz2efRDS9mep
GSpNS2QPbeV7Oz+rO5sZAIxrpuBZObe0NRlZaMamM0X+lSgKnEai2Ep5Pm4NzG6M
Z1dHSrp196l65o0CTiPK0r4IqEUfY1Q6tkzXzc/6c9kUxMerE1saY/OlN29yYJ4F
IDHrczvK5y1ddK8g3FB7fNjti4RCFAec8RsyizemDwS4JLd1R3y1+olJ5OH6Gvqq
uMTSAJHl4LL5DeAZm3WLzL49PJWcaKoNe0oAPDdEalW5GXlAMsbQw9W8mOvBKotP
5Q9k8VVXaILSFn2+AzPKX7fQXoA954KMVnDAgN0r8Fa743J7TlbFbk+l5+V/+88f
cA==
-----END X509 CRL-----
```


The end-entity certificate is issued by the CA. This certificate grants signature authority for one IPv4 address block (192.0.2.0/24). Signature authority for the IPv6 address block and the AS numbers is not needed for the prefixlen file that will be signed, so these items are not included in the end-entity certificate.

```
-----BEGIN CERTIFICATE-----
MIIEVjCCAz6gAwIBAgIUJ605QIPX8rW5m4Zwx3WyuW7hZvswDQYJKoZIhvcNAQEL
BQAwMzExMC8GA1UEAxMoM0FDRTJDRUY0RkIyMUI3RDExRTNFMTg0RUZDMUUyOTdC
Mzc3ODY0MjAeFw0yNjA1MDcyMTIyNDlaFw0yNzAzMDMyMTIyNDlaMDMxMTAvBgNV
BAMTKDkxNDY1MkEzQkQ1MUMxNDQyNjAxOTg4ODlGNUM0NUFCRjA1M0ExODcwggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCycTQrOb/qB2W3i3Ki8PhA/DEW
yii2TgGo9pgCwO9lsIRI6Zb/k+aSiWWP9kSczlcQgtPCVwr62hTQZCIowBN0BL0c
K0/5k1imJdi5qdM3nvKswM8CnoR11vB8pQFwruZmr5xphXRvE+mzuJVLgu2V1upm
BXuWloeymudh6WWJ+GDjwPXO3RiXBejBrOFNXhaFLe08y4DPfr/S/tXJOBm7QzQp
tmbPLYtGfprYu45liFFqqP94UeLpISfXd36AKGzqTFCcc3EW9l5UFE1MFLlnoEog
qtoLoKABt0IkOFGKeC/EgeaBdWLe469ddC9rQft5w6g6cmxG+aYDdIEB34zrAgMB
AAGjggFgMIIBXDAdBgNVHQ4EFgQUkUZSo71RwUQmAZiIn1xFq/BToYcwHwYDVR0j
BBgwFoAUOs4s70+yG30R4+GE78Hil7N3hkIwDgYDVR0PAQH/BAQDAgeAMBgGA1Ud
IAEB/wQOMAwwCgYIKwYBBQUHDgIwYQYDVR0fBFowWDBWoFSgUoZQcnN5bmM6Ly9y
cGtpLmV4YW1wbGUubmV0L3JlcG9zaXRvcnkvM0FDRTJDRUY0RkIyMUI3RDExRTNF
MTg0RUZDMUUyOTdCMzc3ODY0Mi5jcmwwbAYIKwYBBQUHAQEEYDBeMFwGCCsGAQUF
BzAChlByc3luYzovL3Jwa2kuZXhhbXBsZS5uZXQvcmVwb3NpdG9yeS8zQUNFMkNF
RjRGQjIxQjdEMTFFM0UxODRFRkMxRTI5N0IzNzc4NjQyLmNlcjAfBggrBgEFBQcB
BwEB/wQQMA4wDAQCAAEwBgMEAMAAAjANBgkqhkiG9w0BAQsFAAOCAQEAUIykBaqY
nR/U+AXYzCqRbMqdygFY9R11fiNQubpkf5kEYHFxTut0CZLz9dToxuHRDLbPhjJv
Ci3cDkb2ICy1Fdcit5oi9jFl1MD/sFa4l/FWGM07PhgKY+Isz3DXEw9furF7Al3I
gbB0and5HQrvQbO6AnqixSYDffANsnZssojMzlHJIA9OLHIuhGZ66t+yh2VclhwV
7JdS+0EdyA0npIrTGyp//pD5vrigF04y+J4Y61jFXfmbWZbNJF/bMzFeBxD2PKaE
uwixf65s3yI0JDjBbXjUtUhqyty0IZqV2HcuWU7MKH9Qc/wvrJDd4K4xTbkWWYgA
ql7bgmJTHpW2Gw==
-----END CERTIFICATE-----
```


The end-entity certificate is displayed below in detail. For brevity, the other two certificates are not.

```
   0 1110: SEQUENCE {
   4  830:  SEQUENCE {
   8    3:   [0] {
  10    1:    INTEGER 2
         :     }
  13   20:   INTEGER
         :    27 AD 39 40 83 D7 F2 B5 B9 9B 86 70 C7 75 B2 B9
         :    6E E1 66 FB
  35   13:   SEQUENCE {
  37    9:    OBJECT IDENTIFIER
         :     sha256WithRSAEncryption (1 2 840 113549 1 1 11)
  48    0:    NULL
         :     }
  50   51:   SEQUENCE {
  52   49:    SET {
  54   47:     SEQUENCE {
  56    3:      OBJECT IDENTIFIER commonName (2 5 4 3)
  61   40:      PrintableString
         :       '3ACE2CEF4FB21B7D11E3E184EFC1E297B3778642'
         :       }
         :      }
         :     }
 103   30:   SEQUENCE {
 105   13:    UTCTime 07/05/2026 21:22:49 GMT
 120   13:    UTCTime 03/03/2027 21:22:49 GMT
         :     }
 135   51:   SEQUENCE {
 137   49:    SET {
 139   47:     SEQUENCE {
 141    3:      OBJECT IDENTIFIER commonName (2 5 4 3)
 146   40:      PrintableString
         :       '914652A3BD51C144260198889F5C45ABF053A187'
         :       }
         :      }
         :     }
 188  290:   SEQUENCE {
 192   13:    SEQUENCE {
 194    9:     OBJECT IDENTIFIER
         :      rsaEncryption (1 2 840 113549 1 1 1)
 205    0:     NULL
         :      }
 207  271:    BIT STRING, encapsulates {
 212  266:     SEQUENCE {
 216  257:      INTEGER
         :       00 B2 71 34 2B 39 BF EA 07 65 B7 8B 72 A2 F0 F8
         :       40 FC 31 16 CA 28 B6 4E 01 A8 F6 98 02 C0 EF 65
         :       B0 84 48 E9 96 FF 93 E6 92 89 65 8F F6 44 9C CE
         :       57 10 82 D3 C2 57 0A FA DA 14 D0 64 22 28 C0 13
         :       74 04 BD 1C 2B 4F F9 93 58 A6 25 D8 B9 A9 D3 37
         :       9E F2 AC C0 CF 02 9E 84 75 D6 F0 7C A5 01 70 AE
         :       E6 66 AF 9C 69 85 74 6F 13 E9 B3 B8 95 4B 82 ED
         :       95 D6 EA 66 05 7B 96 96 87 B2 9A E7 61 E9 65 89
         :       F8 60 E3 C0 F5 CE DD 18 97 05 E8 C1 AC E1 4D 5E
         :       16 85 2D ED 3C CB 80 CF 7E BF D2 FE D5 C9 38 19
         :       BB 43 34 29 B6 66 CF 2D 8B 46 7E 9A D8 BB 8E 65
         :       88 51 6A A8 FF 78 51 E2 E9 21 27 D7 77 7E 80 28
         :       6C EA 4C 50 9C 73 71 16 F6 5E 54 14 4D 4C 14 B9
```

```
            :       67 A0 4A 20 AA DA 0B A0 A0 01 B7 42 24 38 51 8A
            :       78 2F C4 81 E6 81 75 62 DE E3 AF 5D 74 2F 6B 41
            :       FB 79 C3 A8 3A 72 6C 46 F9 A6 03 74 81 01 DF 8C
            :       EB
   477    3:       INTEGER 65537
            :        }
            :       }
            :      }
   482  352:    [3] {
   486  348:     SEQUENCE {
   490   29:      SEQUENCE {
   492    3:       OBJECT IDENTIFIER
            :        subjectKeyIdentifier (2 5 29 14)
   497   22:       OCTET STRING, encapsulates {
   499   20:        OCTET STRING
            :       91 46 52 A3 BD 51 C1 44 26 01 98 88 9F 5C 45 AB
            :       F0 53 A1 87
            :         }
            :        }
   521   31:      SEQUENCE {
   523    3:       OBJECT IDENTIFIER
            :        authorityKeyIdentifier (2 5 29 35)
   528   24:       OCTET STRING, encapsulates {
   530   22:        SEQUENCE {
   532   20:         [0]
            :       3A CE 2C EF 4F B2 1B 7D 11 E3 E1 84 EF C1 E2 97
            :       B3 77 86 42
            :          }
            :         }
            :        }
   554   14:      SEQUENCE {
   556    3:       OBJECT IDENTIFIER keyUsage (2 5 29 15)
   561    1:       BOOLEAN TRUE
   564    4:       OCTET STRING, encapsulates {
   566    2:        BIT STRING 7 unused bits
            :         '1'B (bit 0)
            :         }
            :        }
   570   24:      SEQUENCE {
   572    3:       OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
   577    1:       BOOLEAN TRUE
   580   14:       OCTET STRING, encapsulates {
   582   12:        SEQUENCE {
   584   10:         SEQUENCE {
   586    8:          OBJECT IDENTIFIER
            :           resourceCertificatePolicy (1 3 6 1 5 5 7 14 2)
            :           }
            :          }
            :         }
            :        }
   596   97:      SEQUENCE {
   598    3:       OBJECT IDENTIFIER
            :        cRLDistributionPoints (2 5 29 31)
   603   90:       OCTET STRING, encapsulates {
   605   88:        SEQUENCE {
   607   86:         SEQUENCE {
   609   84:          [0] {
   611   82:           [0] {
```

```
 613   80:           [6]
         :          'rsync://rpki.example.net/repository/3ACE'
         :          '2CEF4FB21B7D11E3E184EFC1E297B3778642.crl'
         :            }
         :           }
         :          }
         :         }
         :        }
         :       }
 695  108:     SEQUENCE {
 697    8:      OBJECT IDENTIFIER
         :       authorityInfoAccess (1 3 6 1 5 5 7 1 1)
 707   96:      OCTET STRING, encapsulates {
 709   94:       SEQUENCE {
 711   92:        SEQUENCE {
 713    8:         OBJECT IDENTIFIER
         :          caIssuers (1 3 6 1 5 5 7 48 2)
 723   80:         [6]
         :          'rsync://rpki.example.net/repository/3ACE'
         :          '2CEF4FB21B7D11E3E184EFC1E297B3778642.cer'
         :          }
         :         }
         :        }
         :       }
 805   31:     SEQUENCE {
 807    8:      OBJECT IDENTIFIER
         :       ipAddrBlocks (1 3 6 1 5 5 7 1 7)
 817    1:      BOOLEAN TRUE
 820   16:      OCTET STRING, encapsulates {
 822   14:       SEQUENCE {
 824   12:        SEQUENCE {
 826    2:         OCTET STRING 00 01
 830    6:         SEQUENCE {
 832    4:          BIT STRING
         :           '0100000000000000000000011'B
         :           }
         :          }
         :         }
         :        }
         :       }
         :      }
         :     }
         :    }
 838   13:  SEQUENCE {
 840    9:   OBJECT IDENTIFIER
         :    sha256WithRSAEncryption (1 2 840 113549 1 1 11)
 851    0:   NULL
         :    }
 853  257:  BIT STRING
         :   50 8C A4 05 AA 98 9D 1F D4 F8 05 D8 CC 2A 91 6C
         :   CA 9D CA 01 58 F5 1D 75 7E 23 50 B9 BA 64 7F 99
         :   04 60 71 71 4E EB 74 09 92 F3 F5 D4 E8 C6 E1 D1
         :   0C B6 CF 86 32 6F 0A 2D DC 0E 46 F6 20 2C B5 15
         :   D7 22 B7 9A 22 F6 31 65 D4 C0 FF B0 56 B8 97 F1
         :   56 18 CD 3B 3E 18 0A 63 E2 2C CF 70 D7 13 0F 5F
         :   BA B1 7B 02 5D C8 81 B0 74 6A 77 79 1D 0A EF 41
         :   B3 BA 02 7A A2 C5 26 03 7D F0 0D B2 76 6C B2 88
         :   CC CE 51 C9 20 0F 4E 2C 72 2E 84 66 7A EA DF B2
```

```
              :   87 65 5C 96 1C 15 EC 97 52 FB 41 1D C8 0D 27 A4
              :   8A D3 1B 2A 7F FE 90 F9 BE B8 A0 17 4E 32 F8 9E
              :   18 EB 58 C5 5D F9 9B 59 96 CD 24 5F DB 33 31 5E
              :   07 10 F6 3C A6 84 BB 08 B1 7F AE 6C DF 22 34 24
              :   38 C1 6D 78 D4 B5 48 6A CA DC B4 21 9A 95 D8 77
              :   2E 59 4E CC 28 7F 50 73 FC 2F AC 90 DD E0 AE 31
              :   4D B9 16 59 88 00 AA 5E DB 82 62 53 1E 95 B6 1B
              :   }
```

To allow reproduction of the signature results, the end-entity private key is provided. For brevity, the other two private keys are not.


```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAsnE0Kzm/6gdlt4tyovD4QPwxFsootk4BqPaYAsDvZbCESOmW
/5Pmkollj/ZEnM5XEILTwlcK+toU0GQiKMATdAS9HCtP+ZNYpiXYuanTN57yrMDP
Ap6EddbwfKUBcK7mZq+caYV0bxPps7iVS4LtldbqZgV7lpaHsprnYellifhg48D1
zt0YlwXowazhTV4WhS3tPMuAz36/0v7VyTgZu0M0KbZmzy2LRn6a2LuOZYhRaqj/
eFHi6SEn13d+gChs6kxQnHNxFvZeVBRNTBS5Z6BKIKraC6CgAbdCJDhRingvxIHm
gXVi3uOvXXQva0H7ecOoOnJsRvmmA3SBAd+M6wIDAQABAoIBAQCyB0FeMuKm8bRo
18aKjFGSPEoZi53srIz5bvUgIi92TBLez7ZnzL6Iym26oJ+5th+lCHGO/dqlhXio
pI50C5Yc9TFbblb/ECOsuCuuqKFjZ8CD3GVsHozXKJeMM+/o5YZXQrORj6UnwT0z
ol/JE5pIGUCIgsXX6tz9s5BP3lUAvVQHsv6+vEVKLxQ3wj/1vIL8O/CN036EV0GJ
mpkwmygPjfECT9wbWo0yn3jxJb36+M/QjjUP28oNIVn/IKoPZRXnqchEbuuCJ651
IsaFSqtiThm4WZtvCH/IDq+6/dcMucmTjIRcYwW7fdHfjplllVPve9c/OmpWEQvF
t3ArWUt5AoGBANs4764yHxo4mctLIE7G7l/tf9bP4KKUiYw4R4ByEocuqMC4yhmt
MPCfOFLOQet71OWCkjP2L/7EKUe9yx7G5KmxAHY6jOjvcRkvGsl6lWFOsQ8p126M
Y9hmGzMOjtsdhAiMmOWKzjvm4WqfMgghQe+PnjjSVkgTt+7BxpIuGBAvAoGBANBg
26FF5cDLpixOd3Za1YXsOgguwCaw3Plvi7vUZRpa/zBMELEtyOebfakkIRWNm07l
nE+lAZwxm+29PTD0nqCFE91teyzjnQaLO5kkAdJiFuVV3icLOGo399FrnJbKensm
FGSli+3KxQhCNIJJfgWzq4bE0ioAMjdGbYXzIYQFAoGBAM6tuDJ36KDU+hIS6wu6
O2TPSfZhF/zPo3pCWQ78/QDb+Zdw4IEiqoBA7F4NPVLg9Y/H8UTx9r/veqe7hPOo
Ok7NpIzSmKTHkc5XfZ60Zn9OLFoKbaQ40a1kXoJdWEu2YROaUlAe9F6/Rog6PHYz
vLE5qscRbu0XQhLkN+z7bg5bAoGBAKDsbDEb/dbqbyaAYpmwhH2sdRSkphg7Niwc
DNm9qWa1J6Zw1+M87I6Q8naRREuU1IAVqqWHVLr/ROBQ6NTJ1Uc5/qFeT2XXUgkf
taMKv61tuyjZK3sTmznMh0HfzUpWjEhWnCEuB+ZYVdmO52ZGw2A75RdrILL2+9Dc
PvDXVubRAoGAdqXeSWoLxuzZXzl8rsaKrQsTYaXnOWaZieU1SL5vVe8nK257UDqZ
E3ng2j5XPTUWli+aNGFEJGRoNtcQvO60O/sFZUhu52sqq9mWVYZNh1TB5aP8X+pV
iFcZOLUvQEcN6PA+YQK5FU11rAI1M0Gm5RDnVnUl0L2xfCYxb7FzV6Y=
-----END RSA PRIVATE KEY-----
```


Signing of "192.0.2.0/24,32,1" (terminated by CR and LF), yields the following detached CMS signature.

```
# RPKI Signature: 192.0.2.0 - 192.0.2.255
# MIIGQAYJKoZIhvcNAQcCoIIGMTCCBi0CAQMxDTALBglghkgBZQMEAgEwDQYLKoZ
# IhvcNAQkQATmgggRaMIIEVjCCAz6gAwIBAgIUJ605QIPX8rW5m4Zwx3WyuW7hZv
# swDQYJKoZIhvcNAQELBQAwMzExMC8GA1UEAxMoM0FDRTJDRUY0RkIyMUI3RDExR
# TNFMTg0RUZDMUUyOTdCMzc3ODY0MjAeFw0yNjA1MDcyMTIyNDlaFw0yNzAzMDMy
# MTIyNDlaMDMxMTAvBgNVBAMTKDkxNDY1MkEzQkQ1MUMxNDQyNjAxOTg4ODlGNUM
# 0NUFCRjA1M0ExODcwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCycT
# QrOb/qB2W3i3Ki8PhA/DEWyii2TgGo9pgCwO9lsIRI6Zb/k+aSiWWP9kSczlcQg
# tPCVwr62hTQZCIowBN0BL0cK0/5k1imJdi5qdM3nvKswM8CnoR11vB8pQFwruZm
# r5xphXRvE+mzuJVLgu2V1upmBXuWloeymudh6WWJ+GDjwPXO3RiXBejBrOFNXha
# FLe08y4DPfr/S/tXJOBm7QzQptmbPLYtGfprYu45liFFqqP94UeLpISfXd36AKG
# zqTFCcc3EW9l5UFE1MFLlnoEogqtoLoKABt0IkOFGKeC/EgeaBdWLe469ddC9rQ
# ft5w6g6cmxG+aYDdIEB34zrAgMBAAGjggFgMIIBXDAdBgNVHQ4EFgQUkUZSo71R
# wUQmAZiIn1xFq/BToYcwHwYDVR0jBBgwFoAUOs4s70+yG30R4+GE78Hil7N3hkI
# wDgYDVR0PAQH/BAQDAgeAMBgGA1UdIAEB/wQOMAwwCgYIKwYBBQUHDgIwYQYDVR
# 0fBFowWDBWoFSgUoZQcnN5bmM6Ly9ycGtpLmV4YW1wbGUubmV0L3JlcG9zaXRvc
# nkvM0FDRTJDRUY0RkIyMUI3RDExRTNFMTg0RUZDMUUyOTdCMzc3ODY0Mi5jcmww
# bAYIKwYBBQUHAQEEYDBeMFwGCCsGAQUFBzAChlByc3luYzovL3Jwa2kuZXhhbXB
# sZS5uZXQvcmVwb3NpdG9yeS8zQUNFMkNFRjRGQjIxQjdEMTFFM0UxODRFRkMxRT
# I5N0IzNzc4NjQyLmNlcjAfBggrBgEFBQcBBwEB/wQQMA4wDAQCAAEwBgMEAMAAA
# jANBgkqhkiG9w0BAQsFAAOCAQEAUIykBaqYnR/U+AXYzCqRbMqdygFY9R11fiNQ
# ubpkf5kEYHFxTut0CZLz9dToxuHRDLbPhjJvCi3cDkb2ICy1Fdcit5oi9jFl1MD
# /sFa4l/FWGM07PhgKY+Isz3DXEw9furF7Al3IgbB0and5HQrvQbO6AnqixSYDff
# ANsnZssojMzlHJIA9OLHIuhGZ66t+yh2VclhwV7JdS+0EdyA0npIrTGyp//pD5v
# rigF04y+J4Y61jFXfmbWZbNJF/bMzFeBxD2PKaEuwixf65s3yI0JDjBbXjUtUhq
# yty0IZqV2HcuWU7MKH9Qc/wvrJDd4K4xTbkWWYgAql7bgmJTHpW2GzGCAaowggG
# mAgEDgBSRRlKjvVHBRCYBmIifXEWr8FOhhzALBglghkgBZQMEAgGgazAaBgkqhk
# iG9w0BCQMxDQYLKoZIhvcNAQkQATkwHAYJKoZIhvcNAQkFMQ8XDTI2MDUwNzIxM
# jI0OVowLwYJKoZIhvcNAQkEMSIEIGMBdMKw5mjZYL9qP4ivwgMt8g2+qEO0+Dcn
# N5vQO1bNMA0GCSqGSIb3DQEBAQUABIIBAKzRicWBpSyN5nw39eDNfVai2H1mO0n
# APgZUmVF/vgSCWtR0da1iZots4qwn0XwvvIgu5eZ7edhn9axLXhjTAOQajT4cOw
# 9+raD7+SYdBIAUgZpuFy3Olnu4HykCd8Ub44lPfZVG1lF1LeN248+rWgozpE7xz
# Dv5G83OslbvVzGXaVShJM4fsDfpkpKoQ4LszlBeqguU2yTm3XWVjkxH7VJvTtIT
# SzO3jAqwqnCjfu3mnxCoz7LKES4DPZERsFoJv1zyDdHIXjPnfZuTBjjCOubjaQx
# rRwgZtQ8Ljz3gpz1VzL9mKAv0pUzcyxtQfakHwdYtxyO33z2InljtTFJCroI=
# End Signature: 192.0.2.0 - 192.0.2.255
```


Acknowledgments

Thanks to the authors of [RFC8805] and [RFC9632] and the folk they acknowledge from whom ideas and text have been liberally expropriated. Thanks to John R. Levine for providing useful feedback on the document.

Authors' Addresses

Oliver Gasser
IPinfo
Email: oliver@ipinfo.io

Randy Bush
IIJ Research & Arrcus
5147 Crystal Springs
Bainbridge Island, Washington 98110
United States of America
Email: randy@psg.com

Massimo Candela
NTT
Siriusdreef 70-72
2132 WT Hoofddorp
Netherlands
Email: massimo@ntt.net

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America
Email: housley@vigilsec.com