Internet Engineering Task Force (IETF)
Request for Comments: 9323
Category: Standards Track
ISSN: 2070-1721

J. Snijders
Fastly
T. Harrison
APNIC
B. Maddison
Workonline

November 2022

A Profile for RPKI Signed Checklists (RSCs)

Abstract

This document defines a Cryptographic Message Syntax (CMS) protected content type for use with the Resource Public Key Infrastructure (RPKI) to carry a general-purpose listing of checksums (a 'checklist'). The objective is to allow for the creation of an attestation, termed an "RPKI Signed Checklist (RSC)", which contains one or more checksums of arbitrary digital objects (files) that are signed with a specific set of Internet Number Resources. When validated, an RSC confirms that the respective Internet resource holder produced the RSC.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9323.

Table of Contents

1. Introduction
   1.1. Requirements Language
2. RSC Profile and Distribution
   2.1. RSC EE Certificates
3. The RSC eContentType
4. The RSC eContent
   4.1. Version
   4.2. Resources

4.2.1. ConstrainedASIdentifiers Type
4.2.2. ConstrainedIPAddrBlocks Type
4.3. digestAlgorithm
4.4. checkList
4.4.1. FileNameAndHash
5. RSC Validation
6. Verifying Files or Data Using RSC
7. Operational Considerations
8. Security Considerations
9. IANA Considerations
9.1. SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)
9.2. RPKI Signed Objects
9.3. RPKI Repository Name Schemes
9.4. SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)
9.5. Media Types
10. References
10.1. Normative References
10.2. Informative References
Acknowledgements
Authors' Addresses

1. Introduction

This document defines a Cryptographic Message Syntax (CMS) [RFC5652] [RFC6268] protected content type for a general-purpose listing of checksums (a 'checklist'), for use with the Resource Public Key Infrastructure (RPKI) [RFC6480]. The CMS protected content type is intended to provide for the creation and validation of an RPKI Signed Checklist (RSC), a checksum listing signed with a specific set of Internet Number Resources. The objective is to allow for the creation of an attestation that, when validated, provides a means to confirm a given Internet resource holder produced the RSC.

RPKI Signed Checklists are expected to facilitate inter-domain business use cases that depend on an ability to verify resource holdership. RPKI-based validation processes are expected to become the industry norm for automated Bring Your Own IP (BYOIP) on-boarding or establishment of physical interconnections between Autonomous Systems (ASes).

The RSC concept borrows heavily from Resource Tagged Attestation (RTA) [RPKI-RTA], Manifests [RFC9286], and OpenBSD's signify utility [signify]. The main difference between an RSC and RTA is that the RTA profile allows multiple signers to attest a single digital object through a checksum of its content, while the RSC profile allows a single signer to attest the content of multiple digital objects. A single signer profile is considered a simplification for both implementers and operators.

1.1. 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.

2. RSC Profile and Distribution

RSC follows the Signed Object Template for the RPKI [RFC6488] with one exception: because RSCs MUST NOT be distributed through the global RPKI repository system, the Subject Information Access (SIA) extension MUST be omitted from the RSC's X.509 End-Entity (EE) certificate.

What constitutes suitable transport for RSC files is deliberately unspecified. For example, it might be a USB stick, a web interface secured with HTTPS, an email signed with Pretty Good Privacy (PGP), a T-shirt printed with a QR code, or a carrier pigeon.

2.1. RSC EE Certificates

The Certification Authority (CA) MUST only sign one RSC with each EE certificate and MUST generate a new key pair for each new RSC. This type of EE certificate is termed a "one-time-use" EE certificate (see Section 3 of [RFC6487]).

3. The RSC eContentType

The eContentType for an RSC is defined as id-ct-signedChecklist, with Object Identifier (OID) 1.2.840.113549.1.9.16.1.48.

This OID MUST appear within both the eContentType in the encapContentInfo object and the ContentType signed attribute in the signerInfo object (see [RFC6488]).

4. The RSC eContent

The content of an RSC indicates that a checklist for arbitrary digital objects has been signed with a specific set of Internet Number Resources. An RSC is formally defined as follows:

```
RpkiSignedChecklist-2022
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs9(9) smime(16) mod(0)
    id-mod-rpkiSignedChecklist-2022(73) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  CONTENT-TYPE, Digest, DigestAlgorithmIdentifier
  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) }

  IPAddressOrRange, ASIdOrRange
  FROM IPAddrAndASCertExtn -- in [RFC3779]
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) mod(0)
      id-mod-ip-addr-and-as-ident(30) } ;

ct-rpkiSignedChecklist CONTENT-TYPE ::=
  { TYPE RpkiSignedChecklist
    IDENTIFIED BY id-ct-signedChecklist }

id-ct-signedChecklist OBJECT IDENTIFIER ::=
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-9(9) id-smime(16) id-ct(1) 48 }

RpkiSignedChecklist ::= SEQUENCE {
  version [0]           INTEGER DEFAULT 0,
  resources             ResourceBlock,
  digestAlgorithm       DigestAlgorithmIdentifier,
  checkList             SEQUENCE (SIZE(1..MAX)) OF FileNameAndHash }

FileNameAndHash ::= SEQUENCE {
  fileName              PortableFilename OPTIONAL,
  hash                  Digest }
```

```
PortableFilename ::=
  IA5String (FROM("a".."z" | "A".."Z" | "0".."9" | "." | "_" | "-"))

ResourceBlock ::= SEQUENCE {
  asID         [0]      ConstrainedASIdentifiers OPTIONAL,
  ipAddrBlocks [1]      ConstrainedIPAddrBlocks OPTIONAL }
  -- at least one of asID or ipAddrBlocks MUST be present
  ( WITH COMPONENTS { ..., asID PRESENT} |
    WITH COMPONENTS { ..., ipAddrBlocks PRESENT } )

ConstrainedIPAddrBlocks ::=
  SEQUENCE (SIZE(1..MAX)) OF ConstrainedIPAddressFamily

ConstrainedIPAddressFamily ::= SEQUENCE {
  addressFamily         OCTET STRING (SIZE(2)),
  addressesOrRanges     SEQUENCE (SIZE(1..MAX)) OF IPAddressOrRange }

ConstrainedASIdentifiers ::= SEQUENCE {
  asnum [0]             SEQUENCE (SIZE(1..MAX)) OF ASIdOrRange }

END
```

4.1. Version

The version number of the RpkiSignedChecklist MUST be 0.

4.2. Resources

The resources contained here are the resources used to mark the attestation and MUST be a subset of the set of resources listed by the EE certificate carried in the CMS certificates field.

If the asID field is present, it MUST contain an instance of ConstrainedASIdentifiers.

If the ipAddrBlocks field is present, it MUST contain an instance of ConstrainedIPAddrBlocks.

At least one of asID or ipAddrBlocks MUST be present.

ConstrainedASIdentifiers and ConstrainedIPAddrBlocks are specified such that the resulting DER-encoded data instances are binary compatible with ASIdentifiers and IPAddrBlocks (defined in [RFC3779]), respectively.

Implementations encountering decoding errors whilst attempting to read DER-encoded data using this specification should be aware of the possibility that the data may have been encoded using an implementation intended for use with [RFC3779]. Such data may contain elements prohibited by the current specification.

Attempting to decode the errored data using the more permissive specification contained in [RFC3779] may enable implementors to gather additional context for use when reporting errors to the user.

However, implementations MUST NOT ignore errors resulting from the more restrictive definitions contained herein; in particular, such errors MUST cause the validation procedure described in Section 5 to fail.

4.2.1. ConstrainedASIdentifiers Type

ConstrainedASIdentifiers is a SEQUENCE consisting of a single field, asnum, which in turn contains a SEQUENCE OF one or more ASIdOrRange instances as defined in [RFC3779].

ConstrainedASIdentifiers is defined such that the resulting DER-encoded data are binary compatible with ASIdentifiers defined in [RFC3779].

4.2.2. ConstrainedIPAddrBlocks Type

ConstrainedIPAddrBlocks is a SEQUENCE OF one or more instances of ConstrainedIPAddressFamily.

There MUST be only one instance of ConstrainedIPAddressFamily per unique Address Family Identifier (AFI).

The elements of ConstrainedIPAddressFamily MUST be ordered by ascending addressFamily values (treating the octets as unsigned numbers). Thus, when both IPv4 and IPv6 addresses are specified, the IPv4 addresses MUST precede the IPv6 addresses (since the IPv4 AFI of 0001 is less than the IPv6 AFI of 0002).

ConstrainedIPAddrBlocks is defined such that the resulting DER-encoded data are binary compatible with IPAddrBlocks defined in [RFC3779].

4.2.2.1. ConstrainedIPAddressFamily Type

4.2.2.1.1. addressFamily Field

The addressFamily field is an OCTET STRING containing a 2-octet AFI, in network byte order. Unlike IPAddrBlocks [RFC3779], a third octet containing a Subsequent Address Family Identifier (SAFI) MUST NOT be present. AFIs are specified in the "Address Family Numbers" registry [IANA.ADDRESS-FAMILY-NUMBERS] maintained by IANA.

4.2.2.1.2. addressesOrRanges Field

The addressesOrRanges element is a SEQUENCE OF one or more IPAddressOrRange values, as defined in [RFC3779]. The rules for canonicalization and encoding defined in Section 2.2.3.6 of [RFC3779] apply to the value of addressesOrRanges.

4.3. digestAlgorithm

The digest algorithm is used to create the message digest of the attested digital object(s). This algorithm MUST be a hashing algorithm defined in [RFC7935].

4.4. checkList

This field is a SEQUENCE OF one or more FileNameAndHash values. There is one FileNameAndHash entry for each digital object referenced on the RSC.

4.4.1. FileNameAndHash

Each FileNameAndHash is an ordered pair of the name of the directory entry containing the digital object and the message digest of the digital object.

The hash field is mandatory. The value of the hash field is the calculated message digest of the digital object. The hashing algorithm is specified in the digestAlgorithm field.

The fileName field is OPTIONAL. This is to allow RSCs to be used in a "stand-alone" fashion in which nameless digital objects are addressed directly through their respective message digest rather than through a file system abstraction.

If the fileName field is present, then its value:

* MUST contain only characters specified in the Portable Filename Character Set as defined in [POSIX].

* MUST be unique with respect to the other FileNameAndHash elements of checkList for which the fileName field is also present.

Conversely, if the fileName field is omitted, then the value of the hash field MUST be unique with respect to the other FileNameAndHash elements of checkList for which the fileName field is also omitted.

5. RSC Validation

Before a Relying Party (RP) can use an RSC to validate a set of digital objects, the RP MUST first validate the RSC. To validate an RSC, the RP MUST perform all the validation checks specified in [RFC6488], except for checking for the presence of an SIA extension, which MUST NOT be present in the EE certificate (see Section 2). In addition, the RP MUST perform the following RSC-specific validation steps:

1. The contents of the CMS eContent field MUST conform to all of the constraints described in Section 4, including the constraints described in Section 4.4.1.

2. If the asID field is present within the contents of the resources field, then the AS identifier delegation extension [RFC3779] MUST be present in the EE certificate contained in the CMS certificates field. The AS identifiers present in the eContent resources field MUST be a subset of those present in the certificate extension. The EE certificate’s AS identifier delegation extension MUST NOT contain "inherit" elements.

3. If the ipAddrBlocks field is present within the contents of the resources field, then the IP address delegation extension [RFC3779] MUST be present in the EE certificate contained in the CMS certificates field. The IP addresses present in the eContent resources field MUST be a subset of those present in the certificate extension. The EE certificate’s IP address delegation extension MUST NOT contain "inherit" elements.

6. Verifying Files or Data Using RSC

To verify a set of digital objects with an RSC:

* The RSC MUST be validated according to the procedure described in Section 5. If the RSC cannot be validated, verification MUST fail. This error SHOULD be reported to the user.

* For every digital object to be verified:

    1. If the verification procedure is provided with a filename for the object being verified (e.g., because the user has provided a file system path from which to read the object), then verification SHOULD proceed in "filename-aware" mode. Otherwise, verification SHOULD proceed in "filename-unaware" mode.

       Implementations MAY provide an option to override the verification mode, for example, to ignore the fact that the object is to be read from a file.

    2. The message digest MUST be computed from the file contents or data using the digest algorithm specified in the digestAlgorithm field of the RSC.

3. The digest computed in Step 2 MUST be compared to the value appearing in the hash field of all FileNameAndHash elements of the checkList field of the RSC.

   One or more FileNameAndHash elements MUST be found with a matching hash value; otherwise, verification MUST fail, and the error SHOULD be reported to the user.

4. If the mode selected in Step 1 is "filename-aware", then exactly one of the FileNameAndHash elements matched in Step 3 MUST contain a fileName field value exactly matching the filename of the object being verified.

   Alternatively, if the mode selected in Step 1 is "filename-unaware", then exactly one of the FileNameAndHash elements matched in Step 3 MUST have the fileName field omitted.

   Otherwise, verification MUST fail, and the error SHOULD be reported to the user.

Note that in the above procedure, not all elements of checkList necessarily need be used. That is, it is not an error if the length of checkList is greater than the size of the set of digital objects to be verified. However, in this situation, implementations SHOULD issue a warning to the user, allowing for corrective action to be taken if necessary.

7. Operational Considerations

When creating digital objects of a plain-text nature (such as ASCII, UTF-8, HTML, Javascript, and XML), converting such objects into a lossless compressed form is RECOMMENDED. Distributing plain-text objects within a compression envelope (such as GZIP [RFC1952]) might help avoid unexpected canonicalization at intermediate systems (which in turn would lead to checksum verification errors). Validator implementations are expected to treat a checksummed digital object as a string of arbitrary single octets.

If a fileName field is present, but no digital object within the set of to-be-verified digital objects has a filename that matches the content of that field, a validator implementation SHOULD compare the message digest of each digital object to the value from the hash field of the associated FileNameAndHash and report matches to the user for further consideration.

8. Security Considerations

RPs are hereby warned that the data in an RSC is self-asserted. When determining the meaning of any data contained in an RSC, RPs MUST NOT make any assumptions about the signer beyond the fact that it had sufficient control of the issuing CA to create the object. These data have not been verified by the Certificate Authority (CA) that issued the CA certificate to the entity that issued the EE certificate used to validate the RSC.

RPKI certificates are not bound to real-world identities; see [RFC9255] for an elaboration. RPs can only associate real-world entities to Internet Number Resources by additionally consulting an exogenous authority. RSCs are a tool to communicate assertions signed with Internet Number Resources and do not communicate any other aspect of the resource holder’s business operations, such as the identity of the resource holder itself.

RSC objects are not distributed through the RPKI repository system. From this, it follows that third parties who do not have a copy of a

given RSC may not be aware of the existence of that RSC. Since RSC objects use EE certificates but all other currently defined types of RPKI object profiles are published in public CA repositories, an observer may infer from discrepancies in the repository that RSC object(s) may exist. For example, if a CA does not use random serial numbers for certificates, an observer could detect gaps between the serial numbers of the published EE certificates. Similarly, if the CA includes a serial number on a Certificate Revocation List (CRL) that does not match any published object, an observer could postulate that an RSC EE certificate was revoked.

Conversely, a gap in serial numbers does not imply that an RSC exists. Nor does the presence in a CRL of a serial number unknown to the RP imply an RSC object exists: the implicitly referenced object might not be an RSC, it might have never been published, or it may have been revoked before it was visible to RPs. In general, it is not possible to confidently infer the existence or non-existence of RSCs from the repository state without access to a given RSC.

While a one-time-use EE certificate must only be used to generate and sign a single RSC object, CAs technically are not restricted from generating and signing multiple different RSC objects with a single key pair. Any RSC objects sharing the same EE certificate cannot be revoked individually.

9. IANA Considerations

9.1. SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)

IANA has allocated the following in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry:

Decimal	Description	References
48	id-ct-signedChecklist	RFC 9323

Table 1

9.2. RPKI Signed Objects

IANA has registered the OID for the RSC in the "RPKI Signed Objects" registry [RFC6488] as follows:

Name	OID	Reference
Signed Checklist	1.2.840.113549.1.9.16.1.48	RFC 9323

Table 2

9.3. RPKI Repository Name Schemes

IANA has added the Signed Checklist file extension to the "RPKI Repository Name Schemes" registry [RFC6481] as follows:

Filename Extension	RPKI Object	Reference
.sig	Signed Checklist	RFC 9323

Table 3

9.4. SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)

IANA has allocated the following in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

Decimal	Description	References
73	id-mod-rpkiSignedChecklist-2022	RFC 9323

Table 4

9.5. Media Types

IANA has registered the media type "application/rpki-checklist" in the "Media Types" registry as follows:

Type name: application

Subtype name: rpki-checklist

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: Carries an RPKI Signed Checklist. This media type contains no active content. See Section 5 of RFC 9323 for further information.

Interoperability considerations: N/A

Published specification: RFC 9323

Applications that use this media type: RPKI operators

Fragment identifier considerations: N/A

Additional information:

Content: This media type is a signed object, as defined in [RFC6488], which contains a payload of a list of checksums as defined in RFC 9323.
Magic number(s): N/A
File extension(s): .sig
Macintosh file type code(s): N/A

Person & email address to contact for further information: Job Snijders (job@fastly.com)

Intended usage: COMMON

Restrictions on usage: N/A

Author: Job Snijders (job@fastly.com)

Change controller: IETF

10. References

10.1. Normative References

[POSIX] IEEE and The Open Group, "Base Specifications", Issue 7,

DOI 10.1109/IEEESTD.2016.7582338, 2016,
<https://publications.opengroup.org/standards/unix/c165>.

[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>.

[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>.

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

[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>.

[RFC7935] Huston, G. and G. Michaelson, Ed., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure", RFC 7935, DOI 10.17487/RFC7935, August 2016, <https://www.rfc-editor.org/info/rfc7935>.

[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>.

[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>.

10.2. Informative References

[IANA.ADDRESS-FAMILY-NUMBERS]
IANA, "Address Family Numbers", <https://www.iana.org/assignments/address-family-numbers>.

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <https://www.rfc-editor.org/info/rfc1952>.

[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>.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <https://www.rfc-editor.org/info/rfc6480>.

[RFC9255] Bush, R. and R. Housley, "The 'I' in RPKI Does Not Stand for Identity", RFC 9255, DOI 10.17487/RFC9255, June 2022, <https://www.rfc-editor.org/info/rfc9255>.

[RPKI-RTA] Michaelson, G., Huston, G., Harrison, T., Bruijnzeels, T., and M. Hoffman, "A profile for Resource Tagged Attestations (RTAs)", Work in Progress, Internet-Draft, draft-ietf-sidrops-rpki-rta-00, 17 January 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-rpki-rta-00>.

[signify] Unangst, T. and M. Espie, "signify - cryptographically sign and verify files", <https://man.openbsd.org/signify>.

Acknowledgements

The authors wish to thank George Michaelson, Geoff Huston, Randy Bush, Stephen Kent, Matt Lepinski, Rob Austein, Ted Unangst, and Marc Espie for prior art. The authors thank Russ Housley for reviewing the ASN.1 notation and providing suggestions. The authors would like to thank Nimrod Levy, Tim Bruijnzeels, Alberto Leiva, Ties de Kock, Peter Peele, Claudio Jeker, Theo Buehler, Donald Eastlake 3rd, Erik Kline, Robert Wilton, Roman Danyliw, ric Vyncke, Lars Eggert, Paul Wouters, and Murray S. Kucherawy for document review and suggestions.

Authors' Addresses

Job Snijders
Fastly
Amsterdam
Netherlands
Email: job@fastly.com

Tom Harrison
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane QLD 4101
Australia
Email: tomh@apnic.net

Ben Maddison
Workonline Communications
Cape Town
South Africa
Email: benm@workonline.africa