Workgroup: SIDROPS
Internet-Draft:
draft-ietf-sidrops-cms-signing-time-07
Updates: 6488 (if approved)
Published: 16 April 2024
Intended Status: Standards Track
Expires: 18 October 2024
Authors: J. Snijders T. Harrison
Fastly APNIC

On the use of the CMS signing-time attribute in RPKI Signed Objects

Abstract

In the Resource Public Key Infrastructure (RPKI), Signed Objects are defined as Cryptographic Message Syntax (CMS) protected content types. Signed Objects contain a signing-time attribute, representing the purported time at which the object was signed by its issuer. RPKI repositories are accessible using the rsync and RPKI Repository Delta protocols, allowing Relying Parties (RPs) to synchronize a local copy of the RPKI repository used for validation with the remote repositories. This document describes how the CMS signing-time attribute can be used to avoid needless retransfers of data when switching between different synchronization protocols. This document updates RFC 6488 by mandating the presence of the CMS signing-time attribute and disallowing the use of the binary-signing-time attribute.

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.

Status of This Memo

This Internet-Draft will expire on 18 October 2024.

Table of Contents

1. Introduction

In the Resource Public Key Infrastructure (RPKI) [RFC6480], Signed Objects are defined as Cryptographic Message Syntax (CMS) [RFC5652] [RFC6268] protected content types by way of a standard template [RFC6488]. That template includes an optional CMS signing-time attribute, representing the time at which the object was signed by its issuer. At the time when the standard template was defined, rsync was the only distribution mechanism for RPKI repositories.

Since the publication of the standard template, a new, additional protocol for distribution of RPKI repositories has been developed: the RPKI Repository Delta Protocol (RRDP) [RFC8182]. While RPKI repository operators must provide rsync service, RRDP is typically

deployed alongside it as well, and preferred by default by most RP implementations. However, RP implementations also support fallback to rsync in the event of problems with the RRDP service. As deployment experience with RRDP has increased, the usefulness of optimizing switchovers by RPs from one mechanism to the other has become apparent.

This document describes how Repository Operators [RFC6481] and RPs can use the CMS signing-time attribute to minimize the burden of switching over from RRDP to rsync. Additionally, this document updates [RFC6488] by mandating the presence of the CMS signing-time attribute and disallowing the use of the binary-signing-time attribute.

2. Optimized switchover from RRDP to rsync

To avoid needless re-transfers of unchanged files in consecutive rsync synchronizations, [I-D.timbru-sidrops-publication-server-bcp] recommends the use of so-called 'deterministic' (normalized) timestamps for files. When the content of a file is unchanged, Repository Operators SHOULD ensure that the last modification timestamp of the file remains unchanged as well.

This document advances the aforementioned concept by describing a synchronization strategy through which needless transfers are also avoided upon first use of rsync, by leveraging data previously fetched via RRDP.

At the time of writing, all commonly used RP implementations will first attempt synchronization via RRDP, as described in [I-D.ietf-sidrops-prefer-rrdp]. If synchronization via RRDP fails for some reason (e.g. malformed XML, expired TLS certificate, HTTP connection timeout), the RP will attempt to synchronize via rsync instead.

In the rsync synchronization protocol, a file's last modification timestamp (from here on 'mod-time') and filesize are used to determine whether the general-purpose rsync synchronization algorithm needs to be executed for the file. This is the default mode for both the original rsync implementation [rsync] and the OpenBSD implementation [openrsync]. If the sender's copy of the file and the receiver's copy of the file both have the same mod-time and filesize, the files are assumed to contain the same content, and will be omitted from the list of files to be transferred. Ensuring consistency with respect to mod-time for both senders and receivers helps to reduce the burden of rsync synchronization in terms of network bandwidth, disk I/O operations, and CPU usage.

In order to reduce the burden of the rsync synchronization (following an RRDP failure), Repository Operators and RPs SHOULD adhere to the following guidelines.

2.1. Guidance for Repository Operators

When serializing RPKI Signed Objects to a filesystem hierarchy for publication via rsync, the mod-time of the file containing the Signed Object SHOULD be set to the value of the CMS signing-time attribute contained within the Signed Object.

2.2. Guidance for Relying Parties

When serializing RPKI Signed Objects retrieved via RRDP to a filesystem hierarchy, the mod-time of the file containing the Signed Object SHOULD be set to the value of the CMS signing-time attribute contained within the Signed Object.

If an RP uses RRDP to synthesize a filesystem hierarchy for the repository, then synchronizing to the corresponding directory directly is an option. Alternatively, the RP can synchronize to a new (empty) directory using the *--compare-dest=DIR* rsync feature, in order to avoid retrieving files that are already available by way of the synthesized filesystem hierarchy stemming from previous RRDP fetches. The *DIR* component is to be substituted with the name of the directory containing previously fetched and validated RPKI data (in its original DER-encoded form, to ensure the filesize parameter matches).

From the [rsync] man page for *--compare-dest=DIR*:

> This option instructs rsync to use *DIR* on the destination machine as an additional hierarchy to compare destination files against doing transfers (if the files are missing in the destination directory). If a file is found in *DIR* that is identical to the sender's file, the file will NOT be transferred to the destination directory. This is useful for creating a sparse backup of just files that have changed from an earlier backup.

From the [openrsync] man page for *--compare-dest=directory*:

> Use *directory* as an alternate base directory to compare files against on the destination machine. If file in *directory* is found and identical to the sender's file, the file will not be transferred.

3. Presence of the CMS signing-time attribute in public repositories

Analysing the [rpkiviews] archives containing millions of RPKI Signed Objects discovered via the five Regional Internet Registry

(RIR) Trust Anchors (TAs) from June 6th, 2022 until January 29th, 2024, each Signed Object contained a CMS signing-time attribute.

The above means that all of the commonly-used TAs and their subordinate Certification Authorities (CAs) produce Signed Objects that contain a CMS signing-time attribute. This means that making the CMS signing-time attribute mandatory would not cause any existing commonly-used TA or CA to become non-compliant.

As of January 29th, 2024, for 83.8% of Signed Objects the CMS signing-time timestamp matches the file's mod-time observed via rsync. This means that it is already the case that RPs would see a significant reduction in the amount of processing required in rsync if they adopted the strategy outlined in Section 2.2.

In the above-mentioned period of time, no Signed Objects were discovered with a CMS binary-signing-time [RFC6019] attribute in the specified repositories. Therefore, disallowing the use of the CMS binary-signing-time attribute would not cause any existing commonly-used TA or CA to become non-compliant.

4. Update to RFC 6488

This section updates [RFC6488] to make the CMS signing-time attribute mandatory and to disallow the presence of the CMS binary-signing-time attribute.

In section 2.1.6.4, the paragraph starting with "The signedAttrs element MUST be present and ..." and ending in "Other signed attributes MUST NOT be included." is replaced with the following text:

> The signedAttrs element MUST be present and MUST include the content-type, message-digest, and signing-time attributes [RFC5652]. Other signed attributes MUST NOT be included.

In section 2.1.6.4.3, the first sentence "The signing-time attribute MAY be present." is replaced with the following text:

> The signing-time attribute MUST be present.

In section 2.1.6.4.3, the sentence "Note that the presence or absence of the signing-time attribute MUST NOT affect the validity of the signed object (as specified in Section 3)." is removed.

Section 2.1.6.4.4 is removed in its entirety.

In section 3, the paragraph starting with "The signedAttrs field in the SignerInfo object is present ..." (1.f) is replaced with the following text:

> The signedAttrs field in the SignerInfo object is present and contains the content-type attribute (OID 1.2.840.113549.1.9.3), the message-digest attribute (OID 1.2.840.113549.1.9.4), and the signing-time attribute (1.2.840.113549.1.9.5).

In section 3, the paragraph starting with "The signedAttrs field in the SignerInfo object does not ..." (1.g) is replaced with the following text:

> The signedAttrs field in the SignerInfo object does not contain any attributes other than the following three: the content-type attribute (OID 1.2.840.113549.1.9.3), the message-digest attribute (OID 1.2.840.113549.1.9.4), and the signing-time attribute (OID 1.2.840.113549.1.9.5).

In section 9 ("Informative References"), [RFC6019] is removed from the list.

5. Security Considerations

No requirement is imposed concerning the correctness of the signing time attribute. It does not provide reliable information on the time the signature was produced and it bears no relevance for seamless switchover between RRDP and rsync.

While the Security Considerations in [RFC6019] mandate that the signing-time and binary-signing-time attributes, if both present, MUST provide the same date and time; a potential for ambiguity is removed by restricting the RPKI Signed Object profile to have only one field to store the purported signing time.

6. IANA Considerations

This document has no IANA actions.

7. Acknowledgements

The authors would like to thank Ties de Kock, Niels Bakker, Mikael Abrahamsson, Russ Housley, Zaheduzzaman Sarker, Éric Vyncke, Mahesh Jethanandani, and Roman Danyliw, for their helpful review of this document.

8. References

8.1. Normative References

[openrsync] Jeker, C., Obser, F., and K. Dzonsons, "openrsync", 2023, <https://www.openrsync.org/>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[rsync] Tridgell, A., Mackerras, P., and W. Davison, "rsync", 2022, <https://rsync.samba.org/>.

8.2. Informative References

[apnicrepository] APNIC, "APNIC Repository", 2023, <https://rpki.apnic.net/>.

[I-D.ietf-sidrops-prefer-rrdp]
Bruijnzeels, T., Bush, R., and G. G. Michaelson, "Resource Public Key Infrastructure (RPKI) Repository Requirements", Work in Progress, Internet-Draft, draft-ietf-sidrops-prefer-rrdp-02, 23 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-prefer-rrdp-02>.

[I-D.timbru-sidrops-publication-server-bcp] Bruijnzeels, T., de Kock, T., Hill, F., and T. Harrison, "RPKI Publication Server Best Current Practices", Work in Progress, Internet-Draft, draft-timbru-sidrops-publication-server-bcp-02, 18 January 2024, <https://datatracker.ietf.org/doc/html/draft-timbru-sidrops-publication-server-bcp-02>.

[krill-sync] NLNet Labs, "krill-sync - 0.3.0 development branch", December 2023, <https://github.com/NLnetLabs/krill-sync/commit/1df59eac3112384e11b44c2da3010f63925ec50e>.

[ls] IEEE and The Open Group, "ls - The Open Group Base Specifications Issue 7", 2018, <https://pubs.opengroup.org/onlinepubs/9699919799/utilities/ls.html>.

[PAM23] Fontugne, R., Phokeer, A., Pelsser, C., Vermeulen, K., and R. Bush, "RPKI Time-of-Flight: Tracking Delays in the Management, Control, and Data Planes", February 2023, <https://www.iijlab.net/en/members/romain/pdf/romain_pam23.pdf>.

[RFC6019] Housley, R., "BinaryTime: An Alternate Format for Representing Date and Time in ASN.1", RFC 6019, DOI 10.17487/RFC6019, September 2010, <https://www.rfc-editor.org/info/rfc6019>.

[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>.

[rpki-client] Jeker, C., Snijders, J., Dzonsons, K., and T. Buehler, "rpki-client", June 2023, <https://www.rpki-client.org/>.

[rpki-rrdp-tools-py] Kock, T. D., "rpki-rrdp-tools-py", November 2023, <https://github.com/ties/rpki-rrdp-tools-py/>.

[rpkitouch] Snijders, J., "rpkitouch", June 2023, <https://github.com/job/rpkitouch>.

[rpkiviews] Snijders, J., "rpkiviews", June 2023, <http://www.rpkiviews.org/>.

[rsyncit] RIPE NCC, "rsyncit", November 2023, <https://github.com/RIPE-NCC/rsyncit/>.

Appendix A. Considerations and Alternative Approaches

This section is to be removed before publishing as an RFC.

A slightly different approach that has been suggested is to normalize file mod-times based on the Signed Object's embedded End-Entity (EE) X.509 notBefore timestamp value. A downside of this approach is that objects from CAs not using one-time use EE certificates, per section 5.1.1 of [RFC9286] would result in multiple objects signed at different points in time with the same mod-times.

Additionally, CAs might backdate the notBefore timestamp to increase the validity window of the Signed Object, which in turn decreases insight for RPKI operators as to when exactly the Signed Object purportedly came into existence. Along similar lines, the notBefore timestamp may be set in the future for contractual reasons. Setting the mod-time of a file to a future date may be unintuitive for users, and some programs (e.g. GNU make) will warn on encountering files with such mod-times.

There is also an increased chance of two distinct objects published to the same path having the same mod-time and filesize under this approach, due to CAs setting the notBefore timestamp to some stable value for a given object and reissuance often not changing the file size (e.g. where a prefix or a max-length value is changed in a ROA). In such a situation, if the receiver has the first copy of a file, rsync retrieval will skip the second copy of the file, and the synchronization operation for the associated repository will result in a "failed fetch", per section 6.6 of [RFC9286], due to an inconsistency between the file's hash and the hash listed in the associated manifest. That in turn necessitates further retrieval

operations on the part of the receiver. The chance of two distinct objects being issued with the same mod-time and filesize when CMS signing-time is used to set the mod-time is much smaller, since it requires that those distinct objects be issued in very close succession.

Another downside of using notBefore is that Repository operators would need to deserialize both the CMS envelope and the X.509 EE certificate contained therein to extract a timestamp, instead of merely parsing the CMS envelope.

Ensuring the mod-time is set to the CMS signing-time gives RPKI operators a headstart when using tools like [ls], in the sense that the mod-time aligns with the purported time of object issuance.

The CMS signing-time attribute has proven useful in researching and tracking delays in various layers of the RPKI [PAM23]. Mandating the CMS signing-time to be present might aid future researchers studying the RPKI ecosystem.

The *--checksum* option to rsync disables the mod-time and filesize comparison check in favour of a check based on a whole-file checksum. This check is slower than the mod-time and filesize check, but (in instances where the file content has not changed) faster than the general-purpose rsync synchronization algorithm. Since ensuring consistency between the mod-time and filesize on both sides of the transaction is straightforward, there is no particular reason to pursue an approach based on rsync's *--checksum* feature.

Appendix B. Implementation status

This section is to be removed before publishing as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented

protocols more mature. It is up to the individual working groups to use this information as they see fit".

* [rpkitouch] - a timestamp setter utility for both rsync servers and RRDP clients by Job Snijders in C.

* [rpki-client] - a Relying Party implementation by OpenBSD in C, a client side implementation.

* [rsyncit] - a RRDP-to-rsync sync tool by RIPE NCC in Java, run on the server side.

* [apnicrepository] - the public APNIC RPKI repository - the APNIC rsync server normalizes timestamps.

* [rpki-rrdp-tools-py] - a number of client-side RRDP utilities by Ties de Kock in Python.

* [krill-sync] - a RRDP-to-rsync sync tool by NLNet Labs in Rust, run on the server side.

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