SIDR
Internet-Draft
Intended status: Standards Track
Expires: November 11, 2012

R. Kisteleki
RIPE NCC
B. Haberman
JHU APL

May 10, 2012

Securing RPSL Objects with RPKI Signatures

draft-ietf-sidr-rpsl-sig-05.txt

Abstract

This document describes a method to allow parties to electronically sign RPSL-like objects and validate such electronic signatures. This allows relying parties to detect accidental or malicious modifications on such objects. It also allows parties who run Internet Routing Registries or similar databases, but do not yet have RPSS-like authentication of the maintainers of certain objects, to verify that the additions or modifications of such database objects are done by the legitimate holder(s) of the Internet resources mentioned in those objects.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Objects stored in resource databases, like the RIPE DB, are generally protected by an authentication mechanism: anyone creating or modifying an object in the database has to have proper authorization to do so, and therefore has to go through an authentication procedure (provide a password, certificate, e-mail signature, etc.) However, for objects transferred between resource databases, the authentication is not guaranteed. This means when downloading an object stored in this database, one can reasonably safely claim that the object is authentic, but for an imported object one cannot. Also, once such an object is downloaded from the database, it becomes a simple (but still structured) text file with no integrity protection. More importantly, the authentication and integrity guarantees associated with these objects do not always ensure that the entity that generated them is authorized to make the assertions implied by the data contained in the objects.

A potential use for resource certificates [RFC6487] is to employ them to secure such (both imported and downloaded) database objects, by applying a digital signature over the object contents. A maintainer of such signed database objects MUST possess a relevant resource certificate, which shows him/her as the legitimate holder of the Internet number resources in question. This mechanism allows the users of such database objects to verify that the contents are in fact produced by the legitimate holder(s) of the Internet resources mentioned in those objects. It also allows the signatures to cover whole RPSL objects, or just selected attributes of them. In other words, a digital signature created using the private key associated with a resource certificate can offer object security in addition to the channel security already present in most of such databases. Object security in turn allows such objects to be hosted in different databases and still be independently verifiable.

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Stephen Kent 6/7/12 10:10 AM
Comment [1]: Please move the last sentence to the beginning, and re-word. It's odd to have a sentence that claims to contain the "more important" argument for using signatures be at the end ☺.

Stephen Kent 6/7/12 10:11 AM
Deleted: use

Stephen Kent 6/7/12 10:11 AM
Deleted: form of

Stephen Kent 6/7/12 10:12 AM
Deleted: an

Stephen Kent 6/7/12 2:18 PM
Comment [2]: Not reading ahead yet, but I argued previously that signing data that is not covered by the RPKI certificate creates a false sense of security. Hopefully this concern is addressed later in this document.

2. Signature Syntax and Semantics

When signing an RPSL object, the input for the signature process is transformed into a sequence of strings of (ASCII) data. The approach is similar to the one used in DKIM (Domain Key Identified Mail) [RFC4871]. In the case of RPSL, the object-to-be-signed closely resembles an SMTP header, so it seems reasonable to adapt DKIM's relevant features.

2.1. General Attributes, Meta Information

The digital signature associated with an RPSL object is itself a new attribute named "signature". It consists of mandatory and optional fields. These fields are structured in a sequence of name and value pairs, separated by a semicolon ";" and a white space. Collectively these fields make up the value for the new "signature" attribute. The "name" part of such a component is always a single ASCII character that serves as an identifier; the value is an ASCII string the contents of which depend on the field type. Mandatory fields must appear exactly once, whereas optional fields MUST appear at most once.

Mandatory fields of the "signature" attribute:

1. Version number of the signature (field "v"). This field MUST be set to "1".

2. Reference to the certificate corresponding to the private key used to sign this object (field "c"). This is a URL of type "rsync" or "http(s)" that points to a specific resource certificate in an RPKI repository. The value of this field MUST be an "rsync://..." or an "http[s]://..." URL. Any non URL-safe characters (including semicolon ";" and plus "+") must be URL encoded.

3. Signature method (field "m"): what hash and signature algorithms were used to create the signature. The allowed algorithms which can be used for the signature are specified in [RFC6485].

4. Time of signing (field "t"). The format of the value of this field is the number of seconds since Unix EPOCH (00:00:00 on January 1, 1970 in the UTC time zone). The value is expressed as the decimal representation of an unsigned integer.

5. The signed attributes (field "a"). This is a list of attribute names, each separated by an ASCII "+" character (if more than one attribute is enumerated). The list must include any attribute name at most once.

6. The signature itself (field "b"). This MUST be the last field in the list. The signature is the output of the signature algorithm using the appropriate private key and the calculated hash value of the object as inputs. The value of this field is the digital signature in base64 encoding [RFC4648].

Optional fields of the "signature" attribute:

Stephen Kent 6/7/12 2:34 PM
Comment [3]: "a" white space or just "white space" which allows more than one space, tab, etc?

Stephen Kent 6/7/12 2:36 PM
Comment [4]: rsync support is mandatory in the RPKI, while http is not. This distinction should be reflected in this attribute.

Stephen Kent 6/7/12 2:41 PM
Comment [5]: This does not tell me how to encode the signature method, unless you intend for the value of "m" to be an OID. If, in the future, we want to support DSA/ECDSA signatures, we may need to specific curves and points. Unless we assign an OID for each combination, this description of "m:" is not sufficient.

Stephen Kent 6/7/12 3:04 PM
Comment [6]: Since this field value is compared to the validity interval from certificates, it might make more sense to adopt the character string format used there. Also, that format has a time zone, and there is no mention of a TZ here, which leads to ambiguity re the comparison.

1. Signature expiration time (field "x"). The format of the value of this field is the number of seconds since Unix EPOCH (00:00:00 on January 1, 1970 in the UTC time zone). The value is expressed as the decimal representation of an unsigned integer.

2. Reference(s) to other party's certificate(s) (field "o"). If such certificates are mentioned (referred to) in any signature, then this signature should be considered valid only in case when there are other signatures over this current object, and these other signatures refer to, and can be verified with, the certificates mentioned in this field. This mechanism allows having multiple signatures over an object in such a way that all of these signatures have to be present and valid for the whole signature to be considered valid. This would allow interdependent multi-party signatures over an object. One application for such a mechanism is the case of a route[6] object, where both the prefix owner's and the AS owner's signature might be required (if they are different parties). The value of this field MUST be a list of "rsync://..." or "http[s]://..." URLs. If there are more such reference URLs, then they must be separated with a plus "+" sign. Any non URL-safe characters (including semicolon ";" and plus "+") must be URL encoded in all such URLs.

2.2. Signed Attributes

One can look at an RPSL object as an (ordered) set of attributes, each having a "key: value" syntax. Understanding this structure can help in developing more flexible methods for applying digital signatures.

Some of these attributes are automatically added by the database, some are database-dependent, and some do not carry operationally important information. This specification allows the maintainer of each object to specify which attributes are signed and which are not, from among all the attributes associated with an object; in other words, we define a way of including important attributes while excluding irrelevant ones. Allowing the maintainer an object to select the additional attributes covered by the digital signature, subject to the constraints noted below, achieves the goals established in Section 1.

The type of the object determines the minimum set of attributes that MUST be signed. The signer MAY choose to sign additional attributes, in order to provide integrity protection for those attributes too.

When verifying the signature of an object, the verifier has to check whether the signature itself is valid, and whether all the specified attributes are referenced in the signature. If not, the verifier

Stephen Kent 6/7/12 2:41 PM
Comment [7]: Same comment as above re signature time format.

Stephen Kent 6/7/12 3:48 PM
Comment [8]: This sentence is way too long to be understandable. Please rephrase. Also, this text does not explain how multiple signatures are applied, e.g., are they parallel or serial? Only later, in the details of signature generation, can on determine (reading between the lines) that multiple signatures are serialized, I think.

Stephen Kent 6/7/12 2:42 PM
Deleted: s

Stephen Kent 6/7/12 2:43 PM
Deleted: include

Stephen Kent 6/7/12 2:43 PM
Deleted: is

Stephen Kent 6/7/12 2:43 PM
Deleted: expected

Stephen Kent 6/7/12 2:43 PM
Comment [9]: Same comment as earlier re rsync vs. https support

Stephen Kent 6/7/12 2:44 PM
Comment [10]: Cite an RFC here.

Stephen Kent 6/7/12 2:44 PM
Comment [11]: More than what?

Stephen Kent 6/7/12 2:45 PM
Comment [12]: What does this mean?

Stephen Kent 6/7/12 2:45 PM
Deleted: yet

Stephen Kent 6/7/12 2:45 PM
Deleted: others

Stephen Kent 6/7/12 2:45 PM
Deleted: such

Stephen Kent 6/7/12 2:45 PM
Deleted: an

Stephen Kent 6/7/12 2:45 PM
Deleted: define

Stephen Kent 6/7/12 2:46 PM
Deleted: of the

Stephen Kent 6/7/12 2:51 PM
Deleted: that are

Stephen Kent 6/7/12 2:48 PM
Comment [13]: There is no clearly enumerated goal list in Section 1.

Stephen Kent 6/7/12 2:52 PM
Comment [14]: “referenced” is not well defined yet.

MUST reject the signature and treat the object as a non-signed RPSL object.

2.3. Storage of the Signature Data

The result of applying the signature mechanism once is exactly one new attribute for the object. As an illustration, the structure of a signed RPSL object is as follows:

```
attribute1:  value1
attribute2:  value2
attribute3:  value3
...
signature:   v=1; c=rsync://.....; m=sha256WithRSAEncryption;
             t=9999999999;
             a=attribute1+attribute2+attribute3+...;
             b=<base64 data>
```

2.4. Number Resource Coverage

Even if the signature(s) over the object are valid according to the signature validation rules, they may not be relevant to the object; they also need to cover the relevant Internet number resources mentioned in the object. The term “cover” means …

Therefore the Internet number resources present in [RFC3779] extensions of the certificate referred to in the "c" field of the signature (or in the union of such extensions in the "c" fields of the certificates, in case multiple signatures are present) MUST cover the resources in the primary key of the object (e.g., value of the "aut-num:" attribute of an aut-num object, value of the "inetnum:" attribute of an inetnum object, values of "route:" and "origin:" attributes of a route object, etc.).

2.5. Validity Time of the Signature

The validity time interval of a signature is the intersection of the validity time of the certificate used to verify the signature, the "not before" time specified by the "t" field of the signature, and the optional "not after" time specified by the "x" field of the signature.

When checking multiple signatures, these checks are applied to each signature, individually. Specifically, the signing time of the object must be contained with the validity interval of all of the certificates used to verify all of the signatures.

Stephen Kent 6/7/12 2:52 PM
Deleted: threat

Stephen Kent 6/7/12 2:52 PM
Deleted: regular,

Stephen Kent 6/7/12 2:57 PM
Comment [15]: Define this term, or cite an RFC that defines it in RPSL.

Stephen Kent 6/7/12 3:02 PM
Comment [16]: This example is too long a sentence/phrase to be easily understood. What is needed is a more explicit description of the relationship between the 3779 extension values and the relevant attributes, probably by a table. Also, the term “union” is ambiguous here.

Stephen Kent 6/7/12 3:10 PM
Comment [17]: I’m surprised that the requirement here does not require the signing time be bounded by the not before and not after times in each certificate, if that is the intent, it needs to be stated more clearly, as a series of checks to be performed.

3. Signature Creation and Validation Steps

3.1. Canonicalization

The notion of canonicalization is essential to digital signature generation and validation whenever data representations may change between a signer and one or more signature verifiers. Canonicalization defines how one transforms an a representation of data into a series of bits for signature generation and verification. The task of canonicalization is to make irrelevant any differences in representations of the same object that would otherwise cause signature verification to fail. Examples of this are:

o data transformations applied by the databases that host these objects (such as notational changes for IPv4/IPv6 prefixes, automatic addition/modification of "changed" attributes, etc.)

o the difference of line terminators used in different systems.

This means that a database might change the representation of some the submitted data after it was signed, which would cause signature verification to fail, absent canonicalization. This document specifies canonicalization rules to avoid this problem.

The following steps MUST be applied in order to achieve canonicalized representation of an object, before the signature generation and verification processes are is performed:

1. Comments (anything beginning with a "#") MUST be omitted.

2. Any trailing white space MUST be omitted.

3. A multi-line attribute MUST be converted into its single-line equivalent. This is accomplished by:

   * Converting all line endings to a single blank space.

   * Concatenating all lines into a single line.

   * Replacing the trailing blank space with a single new line ("\n").

4. Numerical fields must be converted to canonical representations. These include:

   * Date and time fields MUST be converted to 64-bit NTP Timestamp Format [RFC5905].

Stephen Kent 6/7/12 3:10 PM
Deleted: ,

Stephen Kent 6/7/12 3:10 PM
Deleted: which

Stephen Kent 6/7/12 3:11 PM
Deleted: could be

Stephen Kent 6/7/12 3:11 PM
Deleted: across

Stephen Kent 6/7/12 3:11 PM
Deleted: the

Stephen Kent 6/7/12 3:11 PM
Deleted: destination

Stephen Kent 6/7/12 3:12 PM
Deleted: parts of

Stephen Kent 6/7/12 3:12 PM
Deleted: strict

Stephen Kent 6/7/12 3:12 PM
Deleted: overcome

Stephen Kent 6/7/12 3:12 PM
Deleted: actual

Stephen Kent 6/7/12 3:13 PM
Deleted: (

Stephen Kent 6/7/12 3:13 PM
Deleted:)

Stephen Kent 6/7/12 3:12 PM
Deleted: can

Stephen Kent 6/7/12 3:13 PM
Deleted: begin

Stephen Kent 6/7/12 3:14 PM
Comment [18]: How about changing multiple white spaces to a single white space, analogous to step 3.2?

Stephen Kent 6/7/12 3:14 PM
Comment [19]: See my comments about time formats above.

* AS numbers MUST be converted to ASPLAIN syntax [RFC5396].

* IPv6 addresses must be canonicalized as defined in [RFC5952].

* IPv4 addresses MUST be converted to a 32-bit representation (e.g., Unix's inet_aton()).

* All IP prefixes (IPv4 and IPv6) MUST be represented in CIDR notation [RFC4632].

5. The name of each attribute MUST be converted into lower case, and MUST be kept as part of the attribute line.

6. A tab character ("\t") MUST be converted to a single space.

7. Multiple whitespaces MUST be collapsed into a single space (" ") character.

8. All line endings MUST be converted to a singe new line ("\n") character (thus avoiding CR vs. CRLF differences).

3.2. Signature Creation

Given an RPSL object, in order to create the digital signature, the following steps MUST be performed:

1. For each signature, a new key pair and certificate SHOULD be used. Therefore the signer SHOULD create a single-use key pair and end-entity resource certificate (see [RFC6487]) to be used for signing (and validating) this object.

2. Create a list of attribute names referring to the attributes that will be signed (contents of the "a" field). The minimum set of these attributes is determined by the object type (Section 4); the signer MAY select additional attributes to be signed.

3. Arrange the selected attributes according to the selection sequence specified in the "a" field as above, omitting all attributes that will not be signed.

4. Construct the new "signature" attribute, with all its fields, leaving the value of the "b" field empty.

5. Apply canonicalization rules to the result (including the "signature" attribute).

Stephen Kent 6/7/12 3:16 PM
Comment [20]: 32-bit ? This is binary object, while most other data values are character strings.

Stephen Kent 6/7/12 3:16 PM
Deleted: T

Stephen Kent 6/7/12 3:16 PM
Deleted: s

Stephen Kent 6/7/12 3:16 PM
Deleted: s

Stephen Kent 6/7/12 3:21 PM
Comment [21]: This described how to perform a single signature. It omits important details about performing multiple signatures on the same object.

Stephen Kent 6/7/12 3:18 PM
Deleted: this time

Stephen Kent 6/7/12 3:19 PM
Comment [22]: Do we specify a canonical order for the attributes that MUST be signed.

Stephen Kent 6/7/12 3:20 PM
Comment [23]: See comment above re attribute order.

6. Create the signature over the results of the canonicalization process (according to the signature and hash algorithms specified in the "m" field of the signature attribute).

7. Insert the base64 encoded value of the signature as the value of the "b" field.

8. Append the resulting "signature" attribute to the original object.

3.3. Signature Validation

In order to validate a signature of a signed an object, the following steps MUST be performed:

1. Verify the syntax of the "signature" attribute (i.e. whether it contains the mandatory and optional components and the syntax of these fields matches the specification as described in section 2.1.)

2. Fetch the certificate referred to in the "c" field of the "signature" attribute, and check its validity using the steps described in [RFC6487]. If the certificate(s) cannot be acquired, the signature fails. If any certificate fails the verification procedure in [RFC6487], the signature fails.

3. Extract the list of attributes that were signed using the signer from the "a" field of the "signature" attribute.

4. Verify that the list of signed attributes matches the minimum set of attributes for that object type. (If not all mandatory attributes are listed, the signature fails.)

5. Arrange the selected attributes according to the selection sequence provided in the value of the "a" field, omitting all non-signed attributes. If any attribute in the list is not present, the signature fails.

6. Replace the value of the signature field "b" of the "signature" attribute with an empty string.

7. Apply the canonicalization procedure (Section 3.1) to the selected attributes (including the "signature" attribute).

8. Check the validity of the signature using the signature algorithm specified in the "m" field of the signature attribute, the public key contained in the certificate mentioned in the "c" field of the signature, the signature value specified in the "b" field of the signature attribute, and the output of the canonicalization process.

Stephen Kent 6/7/12 3:22 PM
Deleted: over such

Stephen Kent 6/7/12 3:22 PM
Deleted: mathces

Stephen Kent 6/7/12 3:22 PM
Deleted: miminum

4. Signed Object Types, Set of Signed Attributes

This section describes a list of object types that MAY signed using this approach, and the set of attributes that MUST be signed for these object types.

This list generally excludes attributes that are used to maintain referential integrity in the databases that carry these objects. These attributes usually make sense only within the context of a database, whereas the scope of the signatures is a specific object. Since the attributes in a referred object (such as mnt-by, admin-c, tech-c, ...) can change without any modifications to the signed object itself, signing such attributes could lead to false sense of security (in terms of the contents of the signed data); therefore these attributes should be signed only in order to provide full integrity protection of the object itself.

The signature attribute is always included in the list, but is explicitly noted below for completeness. The attributes enumerated below for each object type are the ones that MUST be included the signed attribute list.

as-block:
* as-block
* org
* signature

aut-num:
* aut-num
* as-name
* member-of
* import
* mp-import
* export
* mp-export
* default
* mp-default
* signature

inet[6]num:
* inet[6]num
* netname
* country
* org
* status
* signature

route[6]:

Stephen Kent 6/7/12 3:58 PM
Comment [24]: The list below is confusing because it fails to state explicitly that these are these the MUST attributes. I like having a list of MAYs, as indicated here, instead of allowing a signer to choose ANY additional attributes to sign, but I don't see that list below.

Stephen Kent 6/7/12 3:49 PM
Deleted: ,

Stephen Kent 6/7/12 3:49 PM
Deleted: since t

Stephen Kent 6/7/12 3:50 PM
Deleted: such

Stephen Kent 6/7/12 3:50 PM
Deleted: only

Stephen Kent 6/7/12 3:50 PM
Deleted: one

Stephen Kent 6/7/12 3:52 PM
Deleted: the

Stephen Kent 6/7/12 3:53 PM
Deleted: only

Stephen Kent 6/7/12 3:53 PM
Deleted: done

Stephen Kent 6/7/12 3:55 PM
Comment [25]: By the end of this paragraph, I no longer know if these examples are precluded from signing, or if they are MAYs!

Stephen Kent 6/7/12 3:55 PM
Deleted: newly constructed

Stephen Kent 6/7/12 3:56 PM
Deleted: "

Stephen Kent 6/7/12 3:56 PM
Deleted: "

Stephen Kent 6/7/12 3:48 PM
Formatted: Underline

Stephen Kent 6/7/12 3:56 PM
Deleted: . ... [1]

* route[6]
* origin
* holes
* org
* member-of
* signature

For each signature, the RFC3779 extension appearing in the certificate used to verify the signature SHOULD include a resource entry that is equivalent to, or covers ("less specific" than) the following resources mentioned in the object the signature is attached to:

o For the as-block object type: the resource in the "as-block" attribute.

o For the aut-num object type: the resource in the "aut-num" attribute.

o For the inet[6]num object type: the resource in the "inet[6]num" attribute.

o For the route[6] object type: the resource in the "route[6]" or "origin" (or both) attributes.

5. Keys and Certificates used for Signature and Verification

The certificate that is referred to in the signature (in the "c" field):

o MUST be an end-entity (i.e. non-CA) certificate
o MUST conform to the X.509 PKIX Resource Certificate profile [RFC6487]
o MUST have an [RFC3779] extension that contains or covers at least one Internet number resource included in a signed attribute.
o SHOULD NOT be used to verify more than one signed object (i.e. should be a "single-use" EE certificate, as defined in [RFC6487]).

6. Security Considerations

RPSL objects stored in the IRR databases are public, and as such there is no need for confidentiality. Each signed RPSL object can have its integrity and authenticity verified using the supplied digital signature and the referenced certificate.

Since the RPSL signature approach leverages X.509 extensions, the security considerations in [RFC3779] apply here as well.

Stephen Kent 6/7/12 3:59 PM
Comment [26]: Here the certificate is singular. Make it clear where multiple signatures are needed/allowed, and where only one certificate is allowed.

Stephen Kent 6/7/12 3:57 PM
Comment [27]: Not MUST?

Stephen Kent 6/7/12 4:02 PM
Comment [28]: You need more text here to explain the verification procedure, and probably some examples too.

Stephen Kent 6/7/12 3:57 PM
Deleted: signatrure

Stephen Kent 6/7/12 4:00 PM
Comment [29]: "contains or covers" is imprecise.

Stephen Kent 6/7/12 4:01 PM
Comment [30]: Huh? I would think that, in the simple case, the certificate MUST contain one or more 3779 extensions that cover EVERY resource attribute.

7. IANA Considerations

[Note to IANA, to be removed prior to publication: there are no IANA considerations stated in this version of the document.]

8. Acknowledgements

The authors would like to acknowledge the valued contributions from Jos Boumans, Steve Kent, and Sean Turner in preparation of this document.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, June 2004.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, May 2007.

[RFC5396] Huston, G. and G. Michaelson, "Textual Representation of Autonomous System (AS) Numbers", RFC 5396, December 2008.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)", RFC 6485, February 2012.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487,

February 2012.

Authors' Addresses

Robert Kisteleki

Email: robert@ripe.net
URI: http://www.ripe.net

Brian Haberman
Johns Hopkins University Applied Physics Lab

Phone: +1 443 778 1319
Email: brian@innovationslab.net