Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 19, 2015, 2015

M. Lepinski, Ed.
BBN

January 19, 2015

BGPsec Protocol Specification
draft-ietf-sidr-bgpsec-protocol-11

Abstract

This document describes BGPsec, an extension to the Border Gateway Protocol (BGP) that provides security for the path of autonomous systems through which a BGP update message passes. BGPsec is implemented via a new optional non-transitive BGP path attribute that carries a digital signature produced by each autonomous system that propagates the update message.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document describes BGPsec, a mechanism for providing path

security for Border Gateway Protocol (BGP) [2] route advertisements. That is, a BGP speaker who receives a valid BGPsec update has cryptographic assurance that the advertised route has the following property: Every AS on the path of ASes listed in the update message has explicitly authorized the advertisement of the route to the subsequent AS in the path.

This document specifies a new optional (non-transitive) BGP path attribute, BGPsec_Path. It also describes how a BGPsec-compliant BGP speaker (referred to hereafter as a BGPsec speaker) can generate, propagate, and validate BGP update messages containing this attribute to obtain the above assurances.

BGPsec is intended to be used to supplement BGP Origin Validation [19] and when used in conjunction with origin validation, it is possible to prevent a wide variety of route hijacking attacks against BGP.

BGPsec relies on the Resource Public Key Infrastructure (RPKI) certificates that attest to the allocation of AS number and IP address resources. (For more information on the RPKI, see [7] and the documents referenced therein.) Any BGPsec speaker who wishes to send, to external (eBGP) peers, BGP update messages containing the BGPsec_Path needs to possess a private key associated with an RPKI router certificate [10] that corresponds to the BGPsec speaker's AS number. Note, however, that a BGPsec speaker does not need such a certificate in order to validate received update messages containing the BGPsec_Path attribute.

2. BGPsec Negotiation

This document defines a new BGP capability [6] that allows a BGP speaker to advertise to a neighbor the ability to send or to receive BGPsec update messages (i.e., update messages containing the BGPsec_Path attribute).

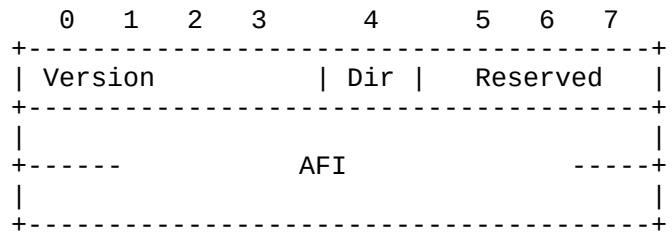
2.1. The BGPsec Capability

This capability has capability code : TBD

The capability length for this capability MUST be set to 3.

The three octets of the capability value are specified as follows.

BGPsec Send Capability Value:



The first four bits of the first octet indicate the version of BGPsec for which the BGP speaker is advertising support. This document defines only BGPsec version 0 (all four bits set to zero). Other versions of BGPsec may be defined in future documents. A BGPsec speaker MAY advertise support for multiple versions of BGPsec by including multiple versions of the BGPsec capability in its BGP OPEN message.

The fifth bit of the first octet is a direction bit which indicates whether the BGP speaker is advertising the capability to send BGPsec update messages or receive BGPsec update messages. The BGP speaker sets this bit to 0 to indicate the capability to receive BGPsec update messages. The BGP speaker sets this bit to 1 to indicate the capability to send BGPsec update messages.

The remaining three bits of the first octet are reserved for future use. These bits are set to zero by the sender of the capability and ignored by the receiver of the capability.

The second and third octets contain the 16-bit Address Family Identifier (AFI) which indicates the address family for which the BGPsec speaker is advertising support for BGPsec. This document only specifies BGPsec for use with two address families, IPv4 and IPv6, AFI values 1 and 2 respectively. BGPsec for use with other address families may be specified in future documents.

2.2. Negotiating BGPsec Support

In order to indicate that a BGP speaker is willing to send BGPsec update messages (for a particular address family), a BGP speaker sends the BGPsec Capability (see Section 2.1) with the Direction bit (the fifth bit of the first octet) set to 1. In order to indicate that the speaker is willing to receive BGP update messages containing the BGPsec_Path attribute (for a particular address family), a BGP speaker sends the BGPsec capability with the Direction bit set to 0. In order to advertise the capability to both send and receive BGPsec update messages, the BGP speaker sends two copies of the BGPsec capability (one with the direction bit set to 0 and one with the direction bit set to 1).

Similarly, if a BGP speaker wishes to use BGPsec with two different address families (i.e., IPv4 and IPv6) over the same BGP session, then the speaker includes two instances of this capability (one for each address family) in the BGP OPEN message. A BGP speaker SHOULD NOT advertise the capability of BGPsec support for a particular AFI unless it has also advertised the multiprotocol extension capability for the same AFI combination [3].

In a session where BGP session, a peer is permitted to send update messages containing the BGPsec_Path attribute if, and only if:

- The given peer sent the BGPsec capability for a particular version of BGPsec and a particular address family with the Direction bit set to 1; and

- The other peer sent the BGPsec capability for the same version of BGPsec and the same address family with the Direction bit set to 0.

In such a session, we say that the use of (the particular version of) BGPsec has been negotiated (for a particular address family). BGP update messages without the BGPsec_Path attribute MAY be sent within a session regardless of whether or not the use of BGPsec is successfully negotiated. However, if BGPsec is not successfully negotiated, then BGP update messages containing the BGPsec_Path attribute MUST NOT be sent.

This document defines the behavior of implementations in the case where BGPsec version zero is the only version that has been successfully negotiated. Any future document which specifies additional versions of BGPsec will need to specify behavior in the case that support for multiple versions is negotiated.

BGPsec cannot provide meaningful security guarantees without support for four-byte AS numbers. Therefore, any BGP speaker that announces the BGPsec capability, MUST also announce the capability for four-byte AS support [4]. If a BGP speaker sends the BGPsec capability but not the four-byte AS support capability then BGPsec has not been successfully negotiated, and update messages containing the BGPsec_Path attribute MUST NOT be sent within such a session.

Note that BGPsec update messages can be quite large, therefore any BGPsec speaker announcing the capability to receive BGPsec messages SHOULD also announce support for the capability to receive BGP extended messages [9].

3. The BGPsec_Path Attribute

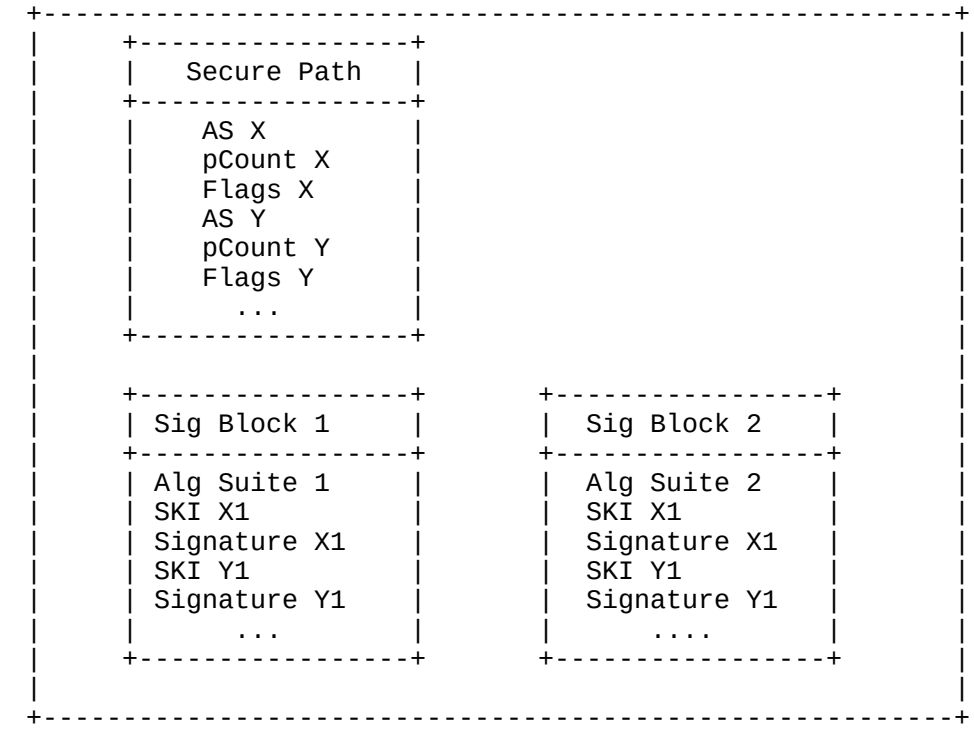
The BGPsec_Path attribute is a new optional non-transitive BGP path attribute.

This document registers a new attribute type code for this attribute : TBD

The BGPsec_Path attribute carries the secured information regarding the path of ASes through which an update message passes. This includes the digital signatures used to protect the path information. We refer to those update messages that contain the BGPsec_Path attribute as "BGPsec Update messages". The BGPsec_Path attribute replaces the AS_PATH attribute in a BGPsec update message. That is, update messages that contain the BGPsec_Path attribute MUST NOT contain the AS_PATH attribute, and vice versa.

The BGPsec_Path attribute is made up of several parts. The following high-level diagram provides an overview of the structure of the BGPsec_Path attribute:

High-Level Diagram of the BGPsec_Path Attribute



The following is the specification of the format for the BGPsec_Path attribute.

BGPsec_Path Attribute

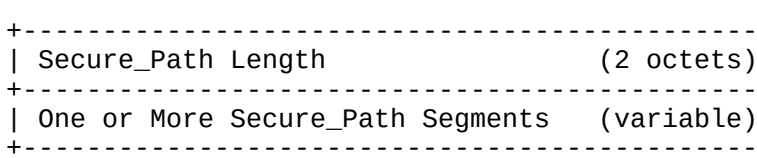
The Secure_Path contains AS path information for the BGPsec update message. This is logically equivalent to the information that is contained in a non-BGPsec AS_PATH attribute. The information in Secure_Path is used by BGPsec speakers in the same way that information from the AS_PATH is used by non-BGPsec speakers. The format of the Secure_Path is described below in Section 3.1.

The BGPsec_Path attribute will contain one or two Signature_Blocks, each of which corresponds to a different algorithm suite. Each of the Signature_Blocks will contain a signature segment for each AS number (i.e., Secure_Path segment) in the Secure_Path. In the most common case, the BGPsec_Path attribute will contain only a single Signature_Block. However, in order to enable a transition from an old algorithm suite to a new algorithm suite (without a flag day), it will be necessary to include two Signature_Blocks (one for the old algorithm suite and one for the new algorithm suite) during the transition period. (See Section 6.1 for more discussion of algorithm transitions.) The format of the Signature_Blocks is described below in Section 3.2.

3.1. Secure_Path

Here we provide a detailed description of the Secure_Path information in the BGPsec_Path attribute.

Secure_Path

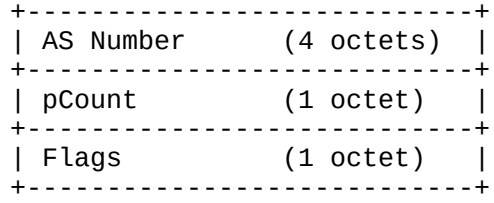


The Secure_Path Length contains the length (in octets) of the entire Secure_Path (including the two octets used to express this length field). As explained below, each Secure_Path segment is six octets long. Note that this means the Secure_Path Length is two greater than six times the number Secure_Path Segments (i.e., the number of

AS numbers in the path).

The Secure_Path contains one Secure_Path Segment for each (distinct) Autonomous System in the path to the originating AS of the NLRI specified in the update message.

Secure_Path Segment



The AS Number is the AS number of the BGP speaker that added this Secure_Path segment to the BGPsec_Path attribute. (See Section 4 for more information on populating this field.)

The pCount field contains the number of repetitions of the associated autonomous system number that the signature covers. This field enables a BGPsec speaker to mimic the semantics of prepending multiple copies of their AS to the AS_PATH without requiring the speaker to generate multiple signatures. The pCount field is also useful in managing route servers (see Section 4.2) and AS Number migrations, see [18] for details.

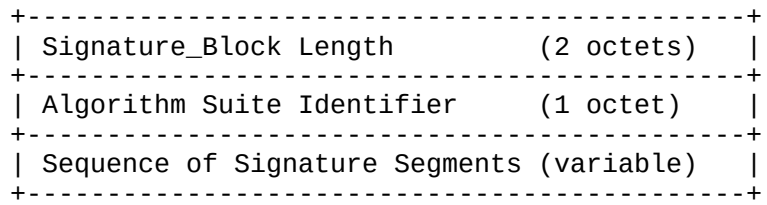
The first bit of the Flags field is the Confed_Segment flag. The Confed_Segment flag is set to one to indicate that the BGPsec speaker that constructed this Secure_Path segment is sending the update message to a peer AS within the same Autonomous System confederation [5]. (That is, the Confed_Segment flag is set in a BGPsec update message whenever, in a non-BGPsec update message, the BGP speaker’s AS would appear in a AS_PATH segment of type AS_CONFED_SEQUENCE.) In all other cases the Confed_Segment flag is set to zero.

The remaining seven bits of the Flags MUST be set to zero by the sender, and ignored by the receiver. Note, however, that the signature is computed over all eight bits of the flags field.

3.2. Signature_Block

Here we provide a detailed description of the Signature_Blocks in the BGPsec_Path attribute.

Signature_Block

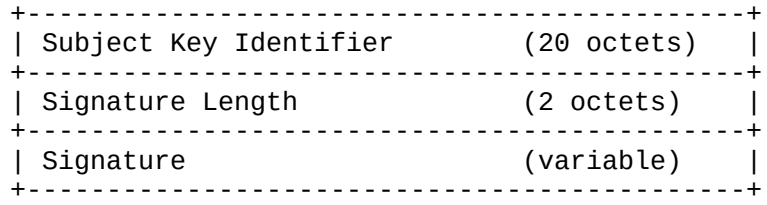


The Signature_Block Length is the total number of octets in the Signature_Block (including the two octets used to express this length field).

The Algorithm Suite Identifier is a one-octet identifier specifying the digest algorithm and digital signature algorithm used to produce the digital signature in each Signature Segment. An IANA registry of algorithm identifiers for use in BGPsec is specified in the BGPsec algorithms document [11].

A Signature_Block has exactly one Signature Segment for each Secure_Path Segment in the Secure_Path portion of the BGPsec_Path Attribute. (That is, one Signature Segment for each distinct AS on the path for the NLRI in the Update message.)

Signature Segments



The Subject Key Identifier contains the value in the Subject Key Identifier extension of the RPKI router certificate [10] that is used to verify the signature (see Section 5 for details on validity of BGPsec update messages).

The Signature Length field contains the size (in octets) of the value in the Signature field of the Signature Segment.

The Signature contains a digital signature that protects the NLRI and the BGPsec_Path attribute (see Sections 4 and 5 for details on signature generation and validation, respectively).

4. Generating a BGPsec Update

Sections 4.1 and 4.2 cover two cases in which a BGPsec speaker may generate an update message containing the BGPsec_Path attribute. The first case is that in which the BGPsec speaker originates a new route advertisement (Section 4.1). That is, the BGPsec speaker is constructing an update message in which the only AS to appear in the BGPsec_Path is the speaker's own AS. The second case is that in which the BGPsec speaker receives a route advertisement from a peer and then decides to propagate the route advertisement to an external (eBGP) peer (Section 4.2). That is, the BGPsec speaker has received a BGPsec update message and is constructing a new update message for the same NLRI in which the BGPsec_Path attribute will contain AS number(s) other than the speaker's own AS.

The remaining case is where the BGPsec speaker sends the update message to an internal (iBGP) peer. When originating a new route advertisement and sending it to an internal peer, the BGPsec speaker omits the BGPsec_Path attribute. When propagating a received route advertisement to an internal peer, the BGPsec speaker populates the BGPsec_Path attribute by copying the BGPsec_Path attribute from the received update message. That is, the BGPsec_Path attribute is copied verbatim. Note that in the case that a BGPsec speaker chooses to forward to an iBGP peer a BGPsec update message that has not been successfully validated (see Section 5), the BGPsec_Path attribute SHOULD NOT be removed. (See Section 7 for the security ramifications of removing BGPsec signatures.)

The information protected by the signature on a BGPsec update message includes the AS number of the peer to whom the update message is being sent. Therefore, if a BGPsec speaker wishes to send a BGPsec update to multiple BGP peers, it MUST generate a separate BGPsec update message for each unique peer AS to which the update message is sent.

A BGPsec update message MUST advertise a route to only a single NLRI. This is because a BGPsec speaker receiving an update message with multiple NLRI would be unable to construct a valid BGPsec update message (i.e., valid path signatures) containing a subset of the NLRI in the received update. If a BGPsec speaker wishes to advertise routes to multiple NLRI, then it MUST generate a separate BGPsec update message for each NLRI.

In order to create or add a new signature to a BGPsec update message with a given algorithm suite, the BGPsec speaker must possess a private key suitable for generating signatures for this algorithm suite. Additionally, this private key must correspond to the public key in a valid Resource PKI end-entity certificate whose AS number

resource extension includes the BGPsec speaker's AS number [10]. Note also that new signatures are only added to a BGPsec update message when a BGPsec speaker is generating an update message to send to an external peer (i.e., when the AS number of the peer is not equal to the BGPsec speaker's own AS number). Therefore, a BGPsec speaker who only sends BGPsec update messages to peers within its own AS, it does not need to possess any private signature keys.

Section 4.3 contains special processing instructions for members of an autonomous system confederation [5]. A BGPsec speaker that is not a member of such a confederation MUST set the Flags field of the Secure_Path Segment to zero in all BGPsec update messages it sends.

Section 4.4 contains instructions for reconstructing the AS_Path attribute in cases where a BGPsec speaker receives an update message with a BGPsec_Path attribute and wishes to propagate the update message to a peer who does not support BGPsec.

4.1. Originating a New BGPsec Update

In an update message that originates a new route advertisement (i.e., an update whose path will contain only a single AS number), when sending the route advertisement to an external, BGPsec-speaking peer, the BGPsec speaker creates a new BGPsec_Path attribute as follows.

First, the BGPsec speaker constructs the Secure_Path with a single Secure_Path Segment. The AS in this path is the BGPsec speaker's own AS number. In particular, this AS number MUST match an AS number in the AS number resource extension field of the Resource PKI router certificate(s) [10] that will be used to verify the digital signature(s) constructed by this BGPsec speaker.

The BGPsec_Path attribute and the AS_Path attribute are mutually exclusive. That is, any update message containing the BGPsec_Path attribute MUST NOT contain the AS_Path attribute. The information that would be contained in the AS_Path attribute is instead conveyed in the Secure_Path portion of the BGPsec_Path attribute.

The Resource PKI enables the legitimate holder of IP address prefix(es) to issue a signed object, called a Route Origination Authorization (ROA), that authorizes a given AS to originate routes to a given set of prefixes (see [8]). It is expected that most relying parties will utilize BGPsec in tandem with origin validation (see [19] and [20]). Therefore, it is RECOMMENDED that a BGPsec speaker only originate a BGPsec update advertising a route for a given prefix if there exists a valid ROA authorizing the BGPsec speaker's AS to originate routes to this prefix.

The pCount field of the Secure_Path Segment is typically set to the value 1. However, a BGPsec speaker may set the pCount field to a value greater than 1. Setting the pCount field to a value greater than one has the same semantics as repeating an AS number multiple times in the AS_PATH of a non-BGPsec update message (e.g., for traffic engineering purposes). Setting the pCount field to a value greater than one permits this repetition without requiring a separate digital signature for each repetition.

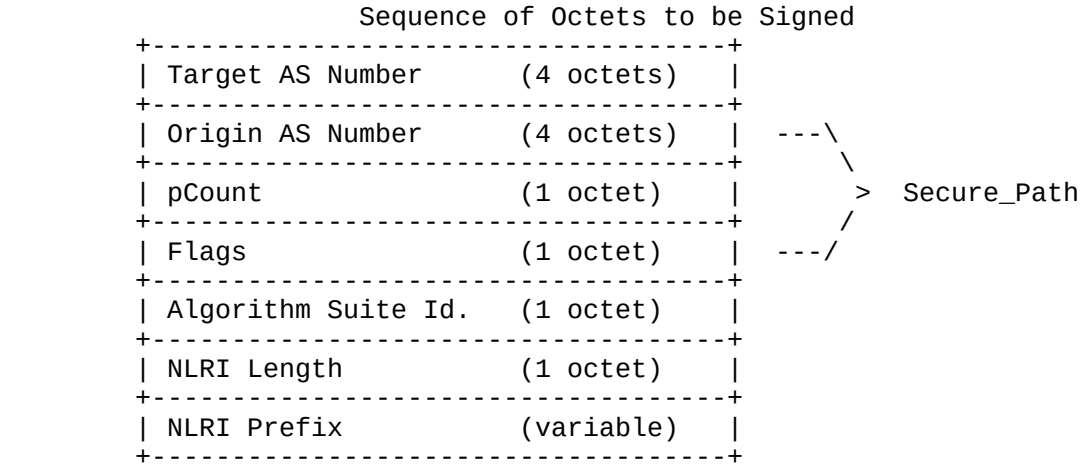
Typically, a BGPsec speaker will use only a single algorithm suite, and thus create only a single Signature_Block in the BGPsec_Path attribute. However, to ensure backwards compatibility during a period of transition from a 'current' algorithm suite to a 'new' algorithm suite, it will be necessary to originate update messages that contain a Signature_Block for both the 'current' and the 'new' algorithm suites (see Section 6.1).

When originating a new route advertisement, each Signature_Block MUST consist of a single Signature Segment. The following describes how the BGPsec speaker populates the fields of the Signature_Block.

The Subject Key Identifier field (see Section 3) is populated with the identifier contained in the Subject Key Identifier extension of the RPKI router certificate corresponding to the BGPsec speaker[10]. This Subject Key Identifier will be used by recipients of the route advertisement to identify the proper certificate to use in verifying the signature.

The Signature field contains a digital signature that binds the NLRI and BGPsec_Path attribute to the RPKI router certificate corresponding to the BGPsec speaker. The digital signature is computed as follows:

- Construct a sequence of octets by concatenating the Target AS Number, the Secure_Path (Origin AS, pCount, and Flags), Algorithm Suite Identifier, and NLRI. The Target AS Number is the AS to whom the BGPsec speaker intends to send the update message. (Note that the Target AS number is the AS number announced by the peer in the OPEN message of the BGP session within which the update is sent.)



- o Apply to this octet sequence the digest algorithm (for the algorithm suite of this Signature_Block) to obtain a digest value.

- o Apply to this digest value the signature algorithm, (for the algorithm suite of this Signature_Block) to obtain the digital signature. Then populate the Signature Field with this digital signature.

The Signature Length field is populated with the length (in octets) of the Signature field.

4.2. Propagating a Route Advertisement

When a BGPsec speaker receives a BGPsec update message containing a BGPsec_Path attribute (with one or more signatures) from an (internal or external) peer, it may choose to propagate the route advertisement by sending to its (internal or external) peers by creating a new BGPsec advertisement for the same prefix.

If a BGPsec router has received only a non-BGPsec update message (without the BGPsec_Path attribute), containing the AS_Path attribute, from a peer for a given prefix then it MUST NOT attach a BGPsec_Path attribute when it propagates the update message. (Note that a BGPsec router may also receive a non-BGPsec update message from an internal peer without the AS_Path attribute, i.e., with just the NLRI in it. In that case, the prefix is originating from that AS and hence the BGPsec speaker SHOULD sign and forward the update to its external peers, as specified in Section 4.1.)

Conversely, if a BGPsec router has received a BGPsec update message (with the BGPsec_Path attribute) from a peer for a given prefix and

it chooses to propagate that peer's route for the prefix, then it SHOULD propagate the route as a BGPsec update message containing the BGPsec_Path attribute.

Note that removing BGPsec signatures (i.e., propagating a route advertisement without the BGPsec_Path attribute) has significant security ramifications. (See Section 7 for discussion of the security ramifications of removing BGPsec signatures.) Therefore, when a route advertisement is received via a BGPsec update message, propagating the route advertisement without the BGPsec_Path attribute is NOT RECOMMENDED, unless the message is sent to a peer that did not advertise the capability to receive BGPsec update messages (see Section 4.4).

Furthermore, note that when a BGPsec speaker propagates a route advertisement with the BGPsec_Path attribute it is not attesting to the validation state of the update message it received. (See Section 7 for more discussion of the security semantics of BGPsec signatures.)

If the BGPsec speaker is producing an update message which would, in the absence of BGPsec, contain an AS_SET (e.g., the BGPsec speaker is performing proxy aggregation), then the BGPsec speaker MUST NOT include the BGPsec_Path attribute. In such a case, the BGPsec speaker must remove any existing BGPsec_Path in the received advertisement(s) for this prefix and produce a traditional (non-BGPsec) update message. It should be noted that BCP 172 [13] recommends against the use of AS_SET and AS_CONFED_SET in the AS_PATH of BGP updates.

To generate the BGPsec_Path attribute on the outgoing update message, the BGPsec speaker first prepends a new Secure_Path Segment (places in first position) to the Secure_Path. The AS number in this Secure_Path segment MUST match the AS number in the AS number resource extension field of the Resource PKI router certificate(s) that will be used to verify the digital signature(s) constructed by this BGPsec speaker[10].

The pCount is typically set to the value 1. A BGPsec speaker may set the pCount field to a value greater than 1. (See Section 4.1 for a discussion of setting pCount to a value greater than 1.)

A route server that participates in the BGP control path, but does not act as a transit AS in the data plane, may choose to set pCount to 0. This option enables the route server to participate in BGPsec and obtain the associated security guarantees without increasing the effective length of the AS path. (Note that BGPsec speakers compute the effective length of the AS path by summing the pCount values in

the BGPsec_Path attribute, see Section 5.) However, when a route server sets the pCount value to 0, it still inserts its AS number into the Secure_Path segment, as this information is needed to validate the signature added by the route server. (See [18] for a discussion of setting pCount to 0 to facilitate AS Number Migration.) BGPsec speakers SHOULD drop incoming update messages with pCount set to zero in cases where the BGPsec speaker does not expect its peer to set pCount to zero. (That is, pCount is only to be set to zero in cases such as route servers or AS Number Migration where the BGPsec speaker's peer expects pCount to be set to zero.)

If the received BGPsec update message contains two Signature_ Blocks and the BGPsec speaker supports both of the corresponding algorithms suites, then the new update message generated by the BGPsec speaker SHOULD include both of the Signature_Blocks. If the received BGPsec update message contains two Signature_Blocks and the BGPsec speaker only supports one of the two corresponding algorithm suites, then the BGPsec speaker MUST remove the Signature_Block corresponding to the algorithm suite that it does not understand. If the BGPsec speaker does not support the algorithm suites in any of the Signature_Blocks contained in the received update message, then the BGPsec speaker MUST NOT propagate the route advertisement with the BGPsec_Path attribute. (That is, if it chooses to propagate this route advertisement at all, it must do so as an unsigned BGP update message).

Note that in the case where the BGPsec_Path has two Signature_Blocks (corresponding to different algorithm suites), the validation algorithm (see Section 5.2) deems a BGPsec update message to be 'Valid' if there is at least one supported algorithm suite (and corresponding Signature_Block) that is deemed 'Valid'. This means that a 'Valid' BGPsec update message may contain a Signature_Block which is not deemed 'Valid' (e.g., contains signatures that the BGPsec does not successfully verify). Nonetheless, such Signature_Blocks MUST NOT be removed. (See Section 7 for a discussion of the security ramifications of this design choice.)

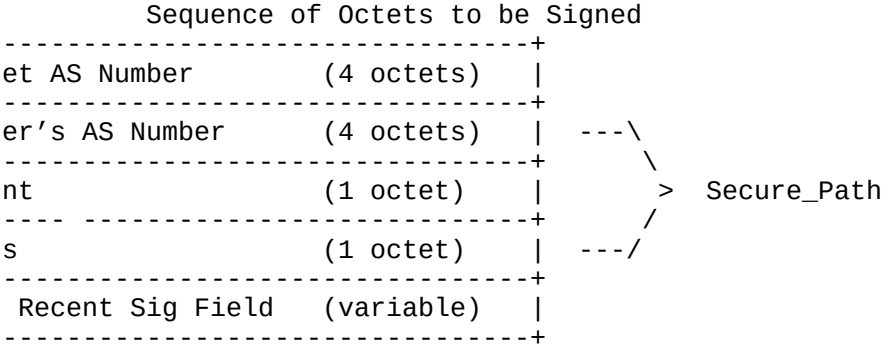
For each Signature_Block corresponding to an algorithm suite that the BGPsec speaker does support, the BGPsec speaker adds a new Signature Segment to the Signature_Block. This Signature Segment is prepended to the list of Signature Segments (placed in the first position) so that the list of Signature Segments appear in the same order as the corresponding Secure_Path segments. The BGPsec speaker populates the fields of this new signature segment as follows.

The Subject Key Identifier field in the new segment is populated with the identifier contained in the Subject Key Identifier extension of the RPKI router certificate corresponding to the BGPsec speaker [10].

This Subject Key Identifier will be used by recipients of the route advertisement to identify the proper certificate to use in verifying the signature.

The Signature field in the new segment contains a digital signature that binds the NLRI and BGPsec_Path attribute to the RPKI router certificate corresponding to the BGPsec speaker. The digital signature is computed as follows:

- Construct a sequence of octets by concatenating the Target AS number, the Secure_Path segment that is being added by the BGPsec speaker constructing the signature, and the signature field of the most recent Signature Segment (the one corresponding to AS from whom the BGPsec speaker's AS received the announcement). Note that the Target AS number is the AS number announced by the peer in the OPEN message of the BGP session within which the BGPsec update message is sent.



- Apply to this octet sequence the digest algorithm (for the algorithm suite of this Signature_Block) to obtain a digest value.

- Apply to this digest value the signature algorithm, (for the algorithm suite of this Signature_Block) to obtain the digital signature. Then populate the Signature Field with this digital signature.

The Signature Length field is populated with the length (in octets) of the Signature field.

4.3. Processing Instructions for Confederation Members

Members of autonomous system confederations [5] MUST additionally follow the instructions in this section for processing BGPsec update messages.

When a confederation member sends a BGPsec update message to a peer that is a member of the same confederation, the confederation member puts its (private) Member-AS Number (as opposed to the public AS Confederation Identifier) in the AS Number field of the Secure_Path Segment that it adds to the BGPsec update message. Furthermore, when a confederation member sends a BGPsec update message to a peer that is a member of the same confederation, the BGPsec speaker that generates the Secure_Path Segment sets the Confed_Segment flag to one. This means that in a BGPsec update message, an AS number appears in a Secure_Path Segment with the Confed_Segment flag set whenever, in a non-BGPsec update message, the AS number would appear in a segment of type AS_CONFED_SEQUENCE in a non-BGPsec update message.

Within a confederation, the verification of BGPsec signatures added by other members of the confederation is optional. If a confederation chooses not to have its members verify signatures added by other confederation members, then when sending a BGPsec update message to a peer that is a member of the same confederation, the confederation members MAY set the Signature field within the Signature_Segment that it generates to be zero (in lieu of calculating the correct digital signature as described in Sections 4.1 and 4.2). Note that if a confederation chooses not to verify digital signatures within the confederation, then BGPsec is able to provide no assurances about the integrity of the (private) Member-AS Numbers placed in Secure_Path segments where the Confed_Segment flag is set to one.

When a confederation member receives a BGPsec update message from a peer within the confederation and propagates it to a peer outside the confederation, it needs to remove all of the Secure_Path Segments added by confederation members as well as the corresponding Signature Segments. To do this, the confederation member propagating the route outside the confederation does the following:

- First, starting with the most recently added Secure_Path segment, remove all of the consecutive Secure_Path segments that have the Confed_Segment flag set to one. Stop this process once a Scure_Path segment is reached which has its Confed_Segment flag set to zero. Keep a count of the number of segments removed in this fashion.

- Second, starting with the most recently added Signature Segment, remove a number of Signature Segments equal to the number of Secure_Path Segments removed in the previous step. (That is, remove the K most recently added signature segments, where K is the number of Secure_Path Segments removed in the previous step.)

- Finally, add a Secure_Path Segment containing, in the AS field, the AS Confederation Identifier (the public AS number of the confederation) as well as a corresponding Signature Segment. Note that all fields other that the AS field are populated as per Sections 4.1 and 4.2.

When validating a received BGPsec update message, confederation members need to make the following adjustment to the algorithm presented in Section 5.2. When a confederation member processes (validates) a Signature Segment and its corresponding Secure_Path Segment, the confederation member must note the following. For a signature produced by a peer BGPsec speaker outside of a confederation, the Target AS will always be the AS Confederation Identifier (the public AS number of the confederation) as opposed to the Member-AS Number.

To handle this case, when a BGPsec speaker (that is a confederation member) processes a current Secure_Path Segment that has the Confed_Segment flag set to zero, if the next most recently added Secure_Path segment has the Confed_Segment flag set to one then, when computing the digest for the current Secure_Path segment, the BGPsec speaker takes the Target AS Number to be the AS Confederation Identifier of the validating BGPsec speaker’s own confederation. (Note that the algorithm in Section 5.2 processes Secure_Path Segments in order from most recently added to least recently added, therefore this special case will apply to the first Secure_Path segment that the algorithm encounters that has the Confed_Segment flag set to zero.)

Finally, as discussed above, an AS confederation may optionally decide that its members will not verify digital signatures added by members. In such a federation, when a confederation member runs the algorithm in Section 5.2, the confederation member, during processing of a Signature_Segment, first checks whether the Confed_Sequence flag in the corresponding Secure_Path segment is set to one. If the Confed_Sequence flag is set to one in the corresponding Secure_Path segment, the confederation member does not perform any further checks on the Signature_Segment and immediately moves on to the next Signature_Segment (and checks its corresponding Secure_Path segment). Note that as specified in Section 5.2, it is an error when a BGPsec speaker receives from a peer, who is not in the same AS confederation, a BGPsec update containing a Confed_Sequence flag set

to one. (As discussed in Section 5.2, any error in the BGPsec_Path attribute MUST be handled using the "treat-as-withdraw", approach as defined in RFC WXYZ [12].)

4.4. Reconstructing the AS_PATH Attribute

BGPsec update messages do not contain the AS_PATH attribute. However, the AS_PATH attribute can be reconstructed from the BGPsec_Path attribute. This is necessary in the case where a route advertisement is received via a BGPsec update message and then propagated to a peer via a non-BGPsec update message (e.g., because the latter peer does not support BGPsec). Note that there may be additional cases where an implementation finds it useful to perform this reconstruction.

The AS_PATH attribute can be constructed from the BGPsec_Path attribute as follows. Starting with an empty AS_PATH attribute, process the Secure_Path segments in order from least-recently added (corresponding to the origin) to most-recently added. For each Secure_Path segment perform the following steps:

1. If the Confed_Segment flag in the Secure_Path segment is set to one, then look at the most-recently added segment in the AS_PATH.

   * In the case where the AS_PATH is empty or in the case where the most-recently added segment is of type AS_SEQUENCE then add (prepend to the AS_PATH) a new AS_PATH segment of type AS_CONFED_SEQUENCE. This segment of type AS_CONFED_SEQUENCE shall contain a number of elements equal to the pCount field in the current Secure_Path segment. Each of these elements shall be the AS number contained in the current Secure_Path segment. (That is, if the pCount field is X, then the segment of type AS_CONFED_SEQUENCE contains X copies of the Secure_Path segment’s AS Number field.)

   * In the case where the most-recently added segment in the AS_PATH is of type AS_CONFED_SEQUENCE then add (prepend to the segment) a number of elements equal to the pCount field in the current Secure_Path segment. The value of each of these elements shall be the AS number contained in the current Secure_Path segment. (That is, if the pCount field is X, then add X copies of the Secure_Path segment’s AS Number field to the existing AS_CONFED_SEQUENCE.)

2. If the Confed_Segment flag in the Secure_Path segment is set to zero, then look at the most-recently added segment in the AS_PATH.

   * In the case where the AS_PATH is empty, and the pCount field in the Secure_Path segment is greater than zero, add (prepend to the AS_PATH) a new AS_PATH segment of type AS_SEQUENCE. This segment of type AS_SEQUENCE shall contain a number of elements equal to the pCount field in the current Secure_Path segment. Each of these elements shall be the AS number contained in the current Secure_Path segment. (That is, if the pCount field is X, then the segment of type AS_SEQUENCE contains X copies of the Secure_Path segment's AS Number field.)

   * In the case where the most recently added segment in the AS_PATH is of type AS_SEQUENCE then add (prepend to the segment) a number of elements equal to the pCount field in the current Secure_Path segment. The value of each of these elements shall be the AS number contained in the current Secure_Path segment. (That is, if the pCount field is X, then add X copies of the Secure_Path segment's AS Number field to the existing AS_SEQUENCE.)

5. Processing a Received BGPsec Update

Upon receiving a BGPsec update message from an external (eBGP) peer, a BGPsec speaker SHOULD validate the message to determine the authenticity of the path information contained in the BGPsec_Path attribute. Typically, a BGPsec speaker will also wish to perform origin validation (see [19] and [20]) on an incoming BGPsec update message, but such validation is independent of the validation described in this section.

Section 5.1 provides an overview of BGPsec validation and Section 5.2 provides a specific algorithm for performing such validation. (Note that an implementation need not follow the specific algorithm in Section 5.2 as long as the input/output behavior of the validation is identical to that of the algorithm in Section 5.2.) During exceptional conditions (e.g., the BGPsec speaker receives an incredibly large number of update messages at once) a BGPsec speaker MAY temporarily defer validation of incoming BGPsec update messages. The treatment of such BGPsec update messages, whose validation has been deferred, is a matter of local policy.

The validity of BGPsec update messages is a function of the current RPKI state. When a BGPsec speaker learns that RPKI state has changed

(e.g., from an RPKI validating cache via the RTR protocol), the BGPsec speaker MUST re-run validation on all affected update messages stored in its ADJ-RIB-IN. That is, when a given RPKI certificate ceases to be valid (e.g., it expires or is revoked), all update messages containing a signature whose SKI matches the SKI in the given certificate must be re-assessed to determine if they are still valid. If this reassessment determines that the validity state of an update has changed then, depending on local policy, it may be necessary to re-run best path selection.

BGPsec update messages do not contain an AS_PATH attribute. Therefore, a BGPsec speaker MUST utilize the AS path information in the BGPsec_Path attribute in all cases where it would otherwise use the AS path information in the AS_PATH attribute. The only exception to this rule is when AS path information must be updated in order to propagate a route to a peer (in which case the BGPsec speaker follows the instructions in Section 4). Section 4.4 provides an algorithm for constructing an AS_PATH attribute from a BGPsec_Path attribute. Whenever the use of AS path information is called for (e.g., loop detection, or use of AS path length in best path selection) the externally visible behavior of the implementation shall be the same as if the implementation had run the algorithm in Section 4.4 and used the resulting AS_PATH attribute as it would for a non-BGPsec update message.

Many signature algorithms are non-deterministic. That is, many signature algorithms will produce different signatures each time they are run (even when they are signing the same data with the same key). Therefore, if an implementation receives a BGPsec update from a peer and later receives a second BGPsec update message from the same peer, the implementation SHOULD treat the second message as a duplicate update message if it differs from the first update message only in the Signature fields (within the BGPsec_Path attribute). That is, if all the fields in the second update are identical to the fields in the first update message, except for the Signature fields, then the second update message should be treated as a duplicate of the first update message. Note that if other fields (e.g., the Subject Key Identifier field) within a Signature segment differ between two update messages then the two updates are not duplicates.

With regards to the processing of duplicate update messages, if the first update message is valid, then an implementation SHOULD NOT run the validation procedure on the second, duplicate update message (even if the bits of the signature field are different). If the first update message is not valid, then an implementation SHOULD run the validation procedure on the second duplicate update message (as the signatures in the second update may be valid even though the first contained a signature that was invalid).

5.1. Overview of BGPsec Validation

Validation of a BGPsec update messages makes use of data from RPKI certificates and signed Route Origination Authorizations (ROA). In particular, to validate update messages containing the BGPsec_Path attribute, it is necessary that the recipient have access to the following data obtained from valid RPKI certificates and ROAs:

- For each valid RPKI router certificate, the AS Number, Public Key and Subject Key Identifier are required,
- For each valid ROA, the AS Number and the list of IP address prefixes.

Note that the BGPsec speaker could perform the validation of RPKI certificates and ROAs on its own and extract the required data, or it could receive the same data from a trusted cache that performs RPKI validation on behalf of (some set of) BGPsec speakers. (For example, the trusted cache could deliver the necessary validity information to the BGPsec speaker using the router key PDU [16] for the RTR protocol [15].)

To validate a BGPsec update message containing the BGPsec_Path attribute, the recipient performs the validation steps specified in Section 5.2. The validation procedure results in one of two states: 'Valid' and 'Not Valid'.

It is expected that the output of the validation procedure will be used as an input to BGP route selection. However, BGP route selection, and thus the handling of the two validation states is a matter of local policy, and is handled using local policy mechanisms.

It is expected that BGP peers will generally prefer routes received via 'Valid' BGPsec update messages over both routes received via 'Not Valid' BGPsec update messages and routes received via update messages that do not contain the BGPsec_Path attribute. However, BGPsec specifies no changes to the BGP decision process. (See [17] for related operational considerations.)

BGPsec validation needs only be performed at the eBGP edge. The validation status of a BGP signed/unsigned update MAY be conveyed via iBGP from an ingress edge router to an egress edge router via some mechanism, according to local policy within an AS. As discussed in Section 4, when a BGPsec speaker chooses to forward a (syntactically correct) BGPsec update message, it SHOULD be forwarded with its BGPsec_Path attribute intact (regardless of the validation state of the update message). Based entirely on local policy, an egress router receiving a BGPsec update message from within its own AS MAY

choose to perform its own validation.

5.2. Validation Algorithm

This section specifies an algorithm for validation of BGPsec update messages. A conformant implementation MUST include a BGPsec update validation algorithm that is functionally equivalent to the externally visible behavior of this algorithm.

First, the recipient of a BGPsec update message performs a check to ensure that the message is properly formed. Specifically, the recipient performs the following checks:

1. Check to ensure that the entire BGPsec_Path attribute is syntactically correct (conforms to the specification in this document).

2. Check that each Signature_Block contains one Signature segment for each Secure_Path segment in the Secure_Path portion of the BGPsec_Path attribute. (Note that the entirety of each Signature_Block must be checked to ensure that it is well formed, even though the validation process may terminate before all signatures are cryptographically verified.)

3. Check that the update message does not contain an AS_PATH attribute.

4. If the update message was received from a peer that is not a member of the BGPsec speaker’s AS confederation, check to ensure that none of the Secure_Path segments contain a Flags field with the Confed_Sequence flag set to one.

5. If the update message was received from a peer that is not expected to set pCount equal to zero (see Section 4.2) then check to ensure that the pCount field in the most-recently added Secure_Path segment is not equal to zero.

If any of these checks fail, it is an error in the BGPsec_Path attribute. Any of these errors in the BGPsec_Path attribute are handled as per RFC WXYZ [12]. BGPsec speakers MUST handle these errors using the "treat-as-withdraw" approach as defined in RFC WXYZ [12].

Next, the BGPsec speaker examines the Signature_Blocks in the BGPsec_Path attribute. A Signature_Block corresponding to an algorithm suite that the BGPsec speaker does not support is not considered in validation. If there is no Signature_Block corresponding to an algorithm suite that the BGPsec speaker supports,

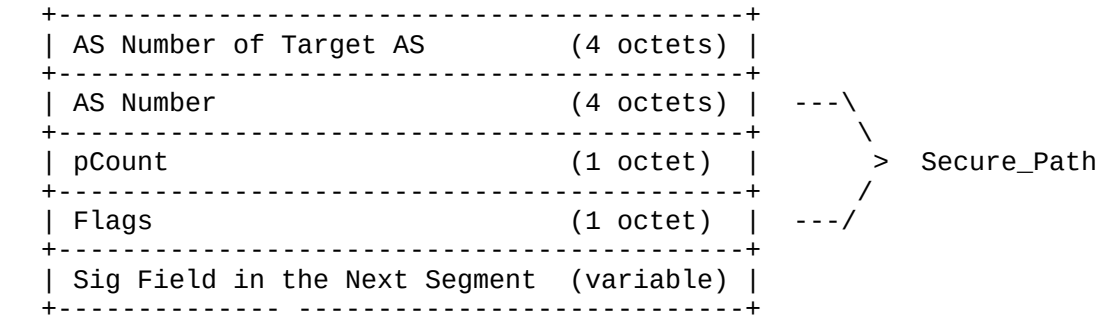
then the BGPsec speaker MUST treat the update message in the same manner that the BGPsec speaker would treat an (unsigned) update message that arrived without a BGPsec_Path attribute.

For each remaining Signature_Block (corresponding to an algorithm suite supported by the BGPsec speaker), the BGPsec speaker iterates through the Signature segments in the Signature_Block, starting with the most recently added segment (and concluding with the least recently added segment). Note that there is a one-to-one correspondence between Signature segments and Secure_Path segments within the BGPsec_Path attribute. The following steps make use of this correspondence.

- (Step I): Locate the public key needed to verify the signature (in the current Signature segment). To do this, consult the valid RPKI router certificate data and look up all valid (AS, SKI, Public Key) triples in which the AS matches the AS number in the corresponding Secure_Path segment. Of these triples that match the AS number, check whether there is an SKI that matches the value in the Subject Key Identifier field of the Signature segment. If this check finds no such matching SKI value, then mark the entire Signature_Block as 'Not Valid' and proceed to the next Signature_Block.

- (Step II): Compute the digest function (for the given algorithm suite) on the appropriate data. If the segment is not the (least recently added) segment corresponding to the origin AS, then the digest function should be computed on the following sequence of octets:

Sequence of Octets to be Hashed

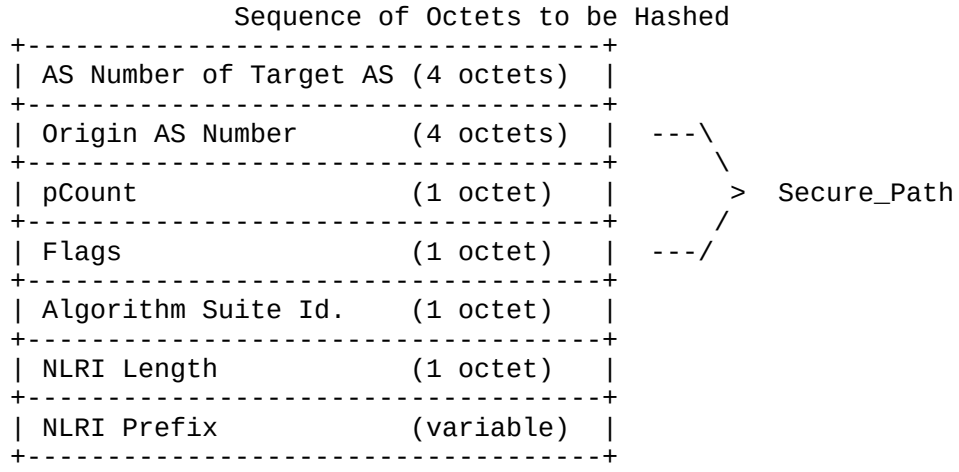


For the first segment to be processed (the most recently added segment), the 'AS Number of Target AS' is the AS number of the BGPsec speaker validating the update message. Note that if a BGPsec speaker uses multiple AS Numbers (e.g., the BGPsec speaker is a member of a confederation), the AS number used here MUST be the AS number announced in the OPEN message for the BGP session over which the BGPsec update was received.

For each other Signature Segment, the 'AS Number of Target AS' is the AS number in the Secure_Path segment that corresponds to the Signature Segment added immediately after the one being processed. (That is, in the Secure_Path segment that corresponds to the Signature segment that the validator just finished processing.)

The AS Number, pCount and Flags fields are taken from the Secure_Path segment that corresponds to the Signature segment currently being processed. The 'Signature Field in the Next Segment' is the Signature field found in the Signature segment that is next to be processed (that is, the next most recently added Signature Segment).

Alternatively, if the segment being processed corresponds to the origin AS (i.e., if it is the least recently added segment), then the digest function should be computed on the following sequence of octets:



The NLRI Length, NLRI Prefix, and Algorithm Suite Identifier are all obtained in a straight forward manner from the NLRI of the update message or the BGPsec_Path attribute being validated. The Origin AS Number, pCount, and Flags fields are taken from the Secure_Path segment corresponding to the Signature Segment currently being processed.

The 'AS Number of Target AS' is the AS Number from the Secure_Path segment that was added immediately after the Secure_Path segment containing the Origin AS Number. (That is, the Secure_Path segment corresponding to the Signature segment that the receiver just finished processing prior to the current Signature segment.)

- (Step III): Use the signature validation algorithm (for the given algorithm suite) to verify the signature in the current segment. That is, invoke the signature validation algorithm on the following three inputs: the value of the Signature field in the current segment; the digest value computed in Step II above; and the public key obtained from the valid RPKI data in Step I above. If the signature validation algorithm determines that the signature is invalid, then mark the entire Signature_Block as 'Not Valid' and proceed to the next Signature_Block. If the signature validation algorithm determines that the signature is valid, then continue processing Signature Segments (within the current Signature_Block).

If all Signature Segments within a Signature_Block pass validation (i.e., all segments are processed and the Signature_Block has not yet been marked 'Not Valid'), then the Signature_Block is marked as 'Valid'.

If at least one Signature_Block is marked as 'Valid', then the validation algorithm terminates and the BGPsec update message is deemed to be 'Valid'. (That is, if a BGPsec update message contains two Signature_Blocks then the update message is deemed 'Valid' if the first Signature_Block is marked 'Valid' OR the second Signature_Block is marked 'Valid'.)

6. Algorithms and Extensibility

6.1. Algorithm Suite Considerations

Note that there is currently no support for bilateral negotiation (using BGP capabilities) between BGPsec peers to use of a particular (digest and signature) algorithm suite. This is because the algorithm suite used by the sender of a BGPsec update message must be understood not only by the peer to whom he is directly sending the message, but also by all BGPsec speakers to whom the route advertisement is eventually propagated. Therefore, selection of an algorithm suite cannot be a local matter negotiated by BGP peers, but instead must be coordinated throughout the Internet.

To this end, a mandatory algorithm suites document will be created which specifies a mandatory-to-use 'current' algorithm suite for use by all BGPsec speakers [11].

It is anticipated that, in the future mandatory, the algorithm suites document will be updated to specify a transition from the 'current' algorithm suite to a 'new' algorithm suite. During the period of transition (likely a small number of years), all BGPsec update messages SHOULD simultaneously use both the 'current' algorithm suite and the 'new' algorithm suite. (Note that Sections 3 and 4 specify how the BGPsec_Path attribute can contain signatures, in parallel, for two algorithm suites.) Once the transition is complete, use of the old 'current' algorithm will be deprecated, use of the 'new' algorithm will be mandatory, and a subsequent 'even newer' algorithm suite may be specified as recommend to implement. Once the transition has successfully been completed in this manner, BGPsec speakers SHOULD include only a single Signature_Block (corresponding to the 'new' algorithm).

6.2. Extensibility Considerations

This section discusses potential changes to BGPsec that would require substantial changes to the processing of the BGPsec_Path and thus necessitate a new version of BGPsec. Examples of such changes include:

- o A new type of signature algorithm that produces signatures of variable length

- o A new type of signature algorithm for which the number of signatures in the Signature_Block is not equal to the number of ASes in the Secure_Path (e.g., aggregate signatures)

- o Changes to the data that is protected by the BGPsec signatures (e.g., attributes other than the AS path)

In the case that such a change to BGPsec were deemed desirable, it is expected that a subsequent version of BGPsec would be created and that this version of BGPsec would specify a new BGP path attribute, let's call it BGPsec_PATH_TWO, which is designed to accommodate the desired changes to BGPsec. In such a case, the mandatory algorithm suites document would be updated to specify algorithm suites appropriate for the new version of BGPsec.

At this point a transition would begin which is analogous to the algorithm transition discussed in Section 6.1. During the transition period all BGPsec speakers SHOULD simultaneously include both the BGPsec_Path attribute and the new BGPsec_PATH_TWO attribute. Once the transition is complete, the use of BGPsec_Path could then be deprecated, at which point BGPsec speakers SHOULD include only the new BGPsec_PATH_TWO attribute. Such a process could facilitate a transition to a new BGPsec semantics in a backwards compatible fashion.

7. Security Considerations

For a discussion of the BGPsec threat model and related security considerations, please see [14].

7.1 Security Guarantees

When used in conjunction with Origin Validation (see [19] and [20]), a BGPsec speaker who receives a valid BGPsec update message, containing a route advertisement for a given prefix, is provided with the following security guarantees:

- o The origin AS number corresponds to an autonomous system that has been authorized, in the RPKI, by the IP address space holder to originate route advertisements for the given prefix.

- o For each AS in the path, a BGPsec speaker authorized by the holder of the AS number intentionally chose (in accordance with local policy) to propagate the route advertisement to the subsequent AS

in the path.

That is, the recipient of a valid BGPsec Update message is assured that the Secure_Path portion of the BGPsec_Path attribute corresponds to a sequence of autonomous systems who have all agreed in principle to forward packets to the given prefix along the indicated path. (It should be noted that BGPsec does not offer any guarantee that the data packets would flow along the indicated path; it only guarantees that the BGP update conveying the path indeed propagated along the indicated path.) Furthermore, the recipient is assured that this path terminates in an autonomous system that has been authorized by the IP address space holder as a legitimate destination for traffic to the given prefix.

Note that although BGPsec provides a mechanism for an AS to validate that a received update message has certain security properties, the use of such a mechanism to influence route selection is completely a matter of local policy. Therefore, a BGPsec speaker can make no assumptions about the validity of a route received from an external BGPsec peer. That is, a compliant BGPsec peer may (depending on the local policy of the peer) send update messages that fail the validity test in Section 5. Thus, a BGPsec speaker MUST completely validate all BGPsec update messages received from external peers. (Validation of update messages received from internal peers is a matter of local policy, see Section 5).

7.2 On the Removal of BGPsec Signatures

There may be cases where a BGPsec speaker deems 'Valid' (as per the validation algorithm in Section 5.2) a BGPsec update message that contains both a 'Valid' and a 'Not Valid' Signature_Block. That is, the update message contains two sets of signatures corresponding to two algorithm suites, and one set of signatures verifies correctly and the other set of signatures fails to verify. In this case, the protocol specifies that a BGPsec speaker choosing to propagate the route advertisement in such an update message SHOULD add its signature to each of the Signature_Blocks. Thus the BGPsec speaker creates a signature using both algorithm suites and creates a new update message that contains both the 'Valid' and the 'Not Valid' set of signatures (from its own vantage point).

To understand the reason for such a design decision consider the case where the BGPsec speaker receives an update message with both a set of algorithm A signatures which are 'Valid' and a set of algorithm B signatures which are 'Not Valid'. In such a case it is possible (perhaps even likely, depending on the state of the algorithm transition) that some of the BGPsec speaker's peers (or other entities further 'downstream' in the BGP topology) do not support

algorithm A. Therefore, if the BGPsec speaker were to remove the 'Not Valid' set of signatures corresponding to algorithm B, such entities would treat the message as though it were unsigned. By including the 'Not Valid' set of signatures when propagating a route advertisement, the BGPsec speaker ensures that 'downstream' entities have as much information as possible to make an informed opinion about the validation status of a BGPsec update.

Note also that during a period of partial BGPsec deployment, a 'downstream' entity might reasonably treat unsigned messages differently from BGPsec updates that contain a single set of 'Not Valid' signatures. That is, by removing the set of 'Not Valid' signatures the BGPsec speaker might actually cause a downstream entity to 'upgrade' the status of a route advertisement from 'Not Valid' to unsigned. Finally, note that in the above scenario, the BGPsec speaker might have deemed algorithm A signatures 'Valid' only because of some issue with RPKI state local to his AS (for example, his AS might not yet have obtained a CRL indicating that a key used to verify an algorithm A signature belongs to a newly revoked certificate). In such a case, it is highly desirable for a downstream entity to treat the update as 'Not Valid' (due to the revocation) and not as 'unsigned' (which would happen if the 'Not Valid' Signature_Blocks were removed).

A similar argument applies to the case where a BGPsec speaker (for some reason such as lack of viable alternatives) selects as his best path (to a given prefix) a route obtained via a 'Not Valid' BGPsec update message. In such a case, the BGPsec speaker should propagate a signed BGPsec update message, adding his signature to the 'Not Valid' signatures that already exist. Again, this is to ensure that 'downstream' entities are able to make an informed decision and not erroneously treat the route as unsigned. It should also be noted that due to possible differences in RPKI data observed at different vantage points in the network, a BGPsec update deemed 'Not Valid' at an upstream BGPsec speaker may be deemed 'Valid' by another BGP speaker downstream.

Indeed, when a BGPsec speaker signs an outgoing update message, it is not attesting to a belief that all signatures prior to its are valid. Instead it is merely asserting that:

- The BGPsec speaker received the given route advertisement with the indicated NLRI and Secure_Path; and

- The BGPsec speaker chose to propagate an advertisement for this route to the peer (implicitly) indicated by the 'Target AS'

7.3 Mitigation of Denial of Service Attacks

The BGPsec update validation procedure is a potential target for denial of service attacks against a BGPsec speaker. Here we consider the mitigation only of denial of service attacks that are specific to BGPsec.

To mitigate the effectiveness of such denial of service attacks, BGPsec speakers should implement an update validation algorithm that performs expensive checks (e.g., signature verification) after performing less expensive checks (e.g., syntax checks). The validation algorithm specified in Section 5.2 was chosen so as to perform checks which are likely to be expensive after checks that are likely to be inexpensive. However, the relative cost of performing required validation steps may vary between implementations, and thus the algorithm specified in Section 5.2 may not provide the best denial of service protection for all implementations.

Additionally, sending update messages with very long AS paths (and hence a large number of signatures) is a potential mechanism to conduct denial of service attacks. For this reason, it is important that an implementation of the validation algorithm stops attempting to verify signatures as soon as an invalid signature is found. (This ensures that long sequences of invalid signatures cannot be used for denial of service attacks.) Furthermore, implementations can mitigate such attacks by only performing validation on update messages that, if valid, would be selected as the best path. That is, if an update message contains a route that would lose out in best path selection for other reasons (e.g., a very long AS path) then it is not necessary to determine the BGPsec-validity status of the route.

7.4 Additional Security Considerations

The mechanism of setting the pCount field to zero is included in this specification to enable route servers in the control path to participate in BGPsec without increasing the effective length of the AS-PATH. However, entities other than route servers could conceivably use this mechanism (set the pCount to zero) to attract traffic (by reducing the effective length of the AS-PATH) illegitimately. This risk is largely mitigated if every BGPsec speaker drops incoming update messages that set pCount to zero but come from a peer that is not a route server. However, note that a recipient of a BGPsec update message within which an upstream entity two or more hops away has set pCount to zero is unable to verify for themselves whether pCount was set to zero legitimately.

BGPsec does not provide protection against attacks at the transport layer. As with any BGP session, an adversary on the path between a

BGPsec speaker and its peer is able to perform attacks such as modifying valid BGPsec updates to cause them to fail validation, injecting (unsigned) BGP update messages without BGPsec_Path_Signature attributes, injecting BGPsec update messages with BGPsec_Path_Signature attributes that fail validation, or causing the peer to tear-down the BGP session. The use of BGPsec does nothing to increase the power of an on-path adversary -- in particular, even an on-path adversary cannot cause a BGPsec speaker to believe a BGPsec-invalid route is valid. However, as with any BGP session, BGPsec sessions SHOULD be protected by appropriate transport security mechanisms.

8. IANA Considerations

TBD: Need IANA to assign numbers for the two capabilities and the BGPsec_PATH attribute.

This document does not create any new IANA registries.

9. Contributors

9.1. Authors

Rob Austein
Dragon Research Labs
sra@hactrn.net

Steven Bellovin
Columbia University
smb@cs.columbia.edu

Randy Bush
Internet Initiative Japan
randy@psg.com

Russ Housley
Vigil Security
housley@vigilsec.com

Matt Lepinski
BBN Technologies
mlepinski.ietf@gmail.com

Stephen Kent
BBN Technologies
kent@bbn.com

Warren Kumari
Google
warren@kumari.net

Doug Montgomery
USA National Institute of Standards and Technology
dougm@nist.gov

Kotikalapudi Sriram
USA National Institute of Standards and Technology
kotikalapudi.sriram@nist.gov

Samuel Weiler
Sparta
weiler+ietf@watson.org

9.2. Acknowledgements

The authors would like to thank Michael Baer, Luke Berndt, Sharon Goldberg, Ed Kern, Chris Morrow, Doug Maughan, Pradosh Mohapatra,

Russ Mundy, Sandy Murphy, Keyur Patel, Mark Reynolds, Heather Schiller, Jason Schiller, John Scudder, Ruediger Volk and David Ward for their valuable input and review.

10. Normative References

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

[2] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4", RFC 4271, January 2006.

[3] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[4] Vohra, Q. and E. Chen, "BGP Support for Four-octet AS Number Space", RFC 4893, May 2007.

[5] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.

[6] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

[7] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

[8] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

[9] Patel, K., Ward, D., and R. Bush, "Extended Message support for BGP", draft-ietf-idr-bgp-extended-messages (work in progress), January 2015.

[10] Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles (work in progress), November 2014.

[11] Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", draft-ietf-sidr-bgpsec-algs (work in progress), July 2014.

[12] Scudder, J., Chen, E., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", draft-ietf-idr-error-handling (work in progress), December 2014.

11. Informative References

[13] Kumari, W. and K. Sriram, "Recommendation for Not Using AS_SET and AS_CONFED_SET in BGP", RFC 6472, December 2011.

[14] Kent, S. and A. Chi, "Threat Model for BGP Path Security", RFC 7132, February 2014.

[15] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, January 2013.

[16] Bush, R., Patel, K., and S. Turner, "Router Key PDU for RPKI-Router Protocol", draft-ymbk-rpki-rtr-keys (work in progress), April 2013.

[17] Bush, R., "BGPsec Operational Considerations", draft-ietf-sidr-bgpsec-ops (work in progress), May 2012.

[18] George, W. and S. Murphy, "BGPsec Considerations for AS Migration", draft-ietf-sidr-as-migration (work in progress), July 2014.

[19] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, February 2013.

[20] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, January 2013.

Author's Address

Matthew Lepinski (editor)
BBN Technologies
10 Moulton St
Cambridge, MA 55409
US

Phone: +1 617 873 5939
Email: mlepinski.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 29, 2017

M. Lepinski, Ed.
NCF
K. Sriram, Ed.
NIST

April 27, 2017

BGPsec Protocol Specification
draft-ietf-sidr-bgpsec-protocol-23

Abstract

This document describes BGPsec, an extension to the Border Gateway Protocol (BGP) that provides security for the path of autonomous systems (ASes) through which a BGP update message passes. BGPsec is implemented via an optional non-transitive BGP path attribute that carries digital signatures produced by each autonomous system that propagates the update message. The digital signatures provide confidence that every AS on the path of ASes listed in the update message has explicitly authorized the advertisement of the route.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 29, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes BGPsec, a mechanism for providing path security for Border Gateway Protocol (BGP) [RFC4271] route advertisements. That is, a BGP speaker who receives a valid BGPsec update has cryptographic assurance that the advertised route has the following property: Every AS on the path of ASes listed in the update message has explicitly authorized the advertisement of the route to the subsequent AS in the path.

This document specifies an optional (non-transitive) BGP path attribute, BGPsec_Path. It also describes how a BGPsec-compliant BGP speaker (referred to hereafter as a BGPsec speaker) can generate, propagate, and validate BGP update messages containing this attribute to obtain the above assurances.

BGPsec is intended to be used to supplement BGP Origin Validation [RFC6483][RFC6811] and when used in conjunction with origin validation, it is possible to prevent a wide variety of route hijacking attacks against BGP.

BGPsec relies on the Resource Public Key Infrastructure (RPKI) certificates that attest to the allocation of AS number and IP address resources. (For more information on the RPKI, see RFC 6480 [RFC6480] and the documents referenced therein.) Any BGPsec speaker who wishes to send, to external (eBGP) peers, BGP update messages containing the BGPsec_Path needs to possess a private key associated with an RPKI router certificate [I-D.ietf-sidr-bgpsec-pki-profiles] that corresponds to the BGPsec speaker's AS number. Note, however, that a BGPsec speaker does not need such a certificate in order to validate received update messages containing the BGPsec_Path attribute (see Section 5.2).

1.1. Requirements Language

The 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 RFC 2119 [RFC2119].

2. BGPsec Negotiation

This document defines a BGP capability [RFC5492] that allows a BGP speaker to advertise to a neighbor the ability to send or to receive BGPsec update messages (i.e., update messages containing the BGPsec_Path attribute).

2.1. The BGPsec Capability

This capability has capability code: TBD

The capability length for this capability MUST be set to 3.

The three octets of the capability format are specified in Figure 1.

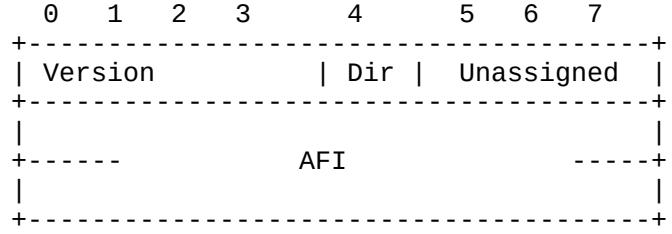


Figure 1: BGPsec Capability format.

The first four bits of the first octet indicate the version of BGPsec for which the BGP speaker is advertising support. This document defines only BGPsec version 0 (all four bits set to zero). Other versions of BGPsec may be defined in future documents. A BGPsec speaker MAY advertise support for multiple versions of BGPsec by including multiple versions of the BGPsec capability in its BGP OPEN message.

The fifth bit of the first octet is a direction bit which indicates whether the BGP speaker is advertising the capability to send BGPsec update messages or receive BGPsec update messages. The BGP speaker sets this bit to 0 to indicate the capability to receive BGPsec update messages. The BGP speaker sets this bit to 1 to indicate the capability to send BGPsec update messages.

The remaining three bits of the first octet are unassigned and for future use. These bits are set to zero by the sender of the capability and ignored by the receiver of the capability.

The second and third octets contain the 16-bit Address Family Identifier (AFI) which indicates the address family for which the BGPsec speaker is advertising support for BGPsec. This document only specifies BGPsec for use with two address families, IPv4 and IPv6, AFI values 1 and 2 respectively [IANA-AF]. BGPsec for use with other address families may be specified in future documents.

2.2. Negotiating BGPsec Support

In order to indicate that a BGP speaker is willing to send BGPsec update messages (for a particular address family), a BGP speaker sends the BGPsec Capability (see Section 2.1) with the Direction bit (the fifth bit of the first octet) set to 1. In order to indicate that the speaker is willing to receive BGP update messages containing the BGPsec_Path attribute (for a particular address family), a BGP speaker sends the BGPsec capability with the Direction bit set to 0. In order to advertise the capability to both send and receive BGPsec update messages, the BGP speaker sends two copies of the BGPsec capability (one with the direction bit set to 0 and one with the direction bit set to 1).

Similarly, if a BGP speaker wishes to use BGPsec with two different address families (i.e., IPv4 and IPv6) over the same BGP session, then the speaker includes two instances of this capability (one for each address family) in the BGP OPEN message. A BGP speaker MUST NOT announce BGPsec capability if it does not support the BGP multiprotocol extension [RFC4760]. Additionally, a BGP speaker MUST NOT advertise the capability of BGPsec support for a particular AFI unless it has also advertised the multiprotocol extension capability for the same AFI [RFC4760].

In a BGPsec peering session, a peer is permitted to send update messages containing the BGPsec_Path attribute if, and only if:

- The given peer sent the BGPsec capability for a particular version of BGPsec and a particular address family with the Direction bit set to 1; and

- The other (receiving) peer sent the BGPsec capability for the same version of BGPsec and the same address family with the Direction bit set to 0.

In such a session, it can be said that the use of the particular version of BGPsec has been negotiated for a particular address family. Traditional BGP update messages (i.e. unsigned, containing AS_PATH attribute) MAY be sent within a session regardless of whether or not the use of BGPsec is successfully negotiated. However, if BGPsec is not successfully negotiated, then BGP update messages containing the BGPsec_Path attribute MUST NOT be sent.

This document defines the behavior of implementations in the case where BGPsec version zero is the only version that has been successfully negotiated. Any future document which specifies additional versions of BGPsec will need to specify behavior in the case that support for multiple versions is negotiated.

BGPsec cannot provide meaningful security guarantees without support for four-byte AS numbers. Therefore, any BGP speaker that announces the BGPsec capability, MUST also announce the capability for four-byte AS support [RFC6793]. If a BGP speaker sends the BGPsec capability but not the four-byte AS support capability then BGPsec has not been successfully negotiated, and update messages containing the BGPsec_Path attribute MUST NOT be sent within such a session.

3. The BGPsec_Path Attribute

The BGPsec_Path attribute is an optional non-transitive BGP path attribute.

This document registers an attribute type code for this attribute: BGPsec_Path (see Section 9).

The BGPsec_Path attribute carries the secured information regarding the path of ASes through which an update message passes. This includes the digital signatures used to protect the path information. The update messages that contain the BGPsec_Path attribute are referred to as "BGPsec Update messages". The BGPsec_Path attribute replaces the AS_PATH attribute in a BGPsec update message. That is, update messages that contain the BGPsec_Path attribute MUST NOT contain the AS_PATH attribute, and vice versa.

The BGPsec_Path attribute is made up of several parts. The high-level diagram in Figure 2 provides an overview of the structure of the BGPsec_Path attribute.

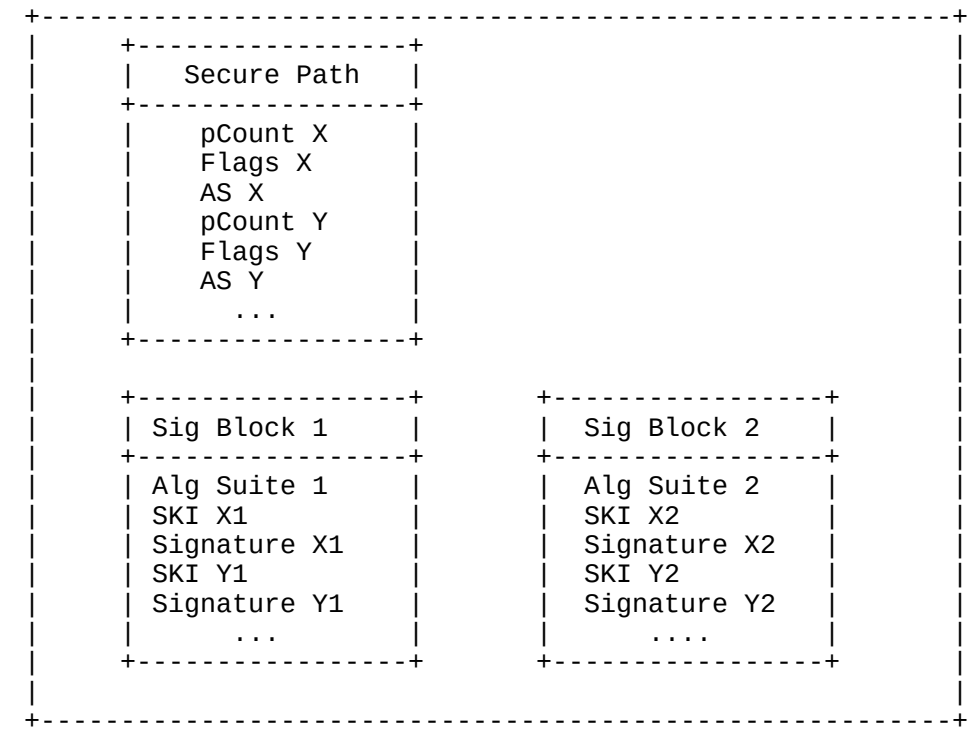


Figure 2: High-level diagram of the BGPsec_Path attribute.

Figure 3 provides the specification of the format for the BGPsec_Path attribute.

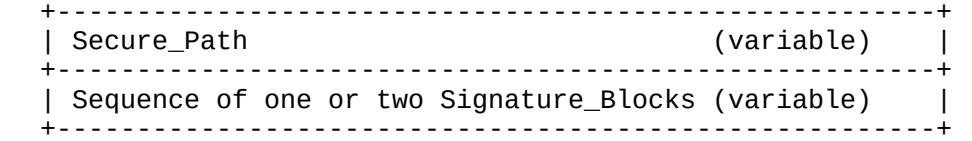


Figure 3: BGPsec_Path attribute format.

The Secure_Path contains AS path information for the BGPsec update message. This is logically equivalent to the information that is contained in a non-BGPsec AS_PATH attribute. The information in Secure_Path is used by BGPsec speakers in the same way that information from the AS_PATH is used by non-BGPsec speakers. The format of the Secure_Path is described below in Section 3.1.

The BGPsec_Path attribute will contain one or two Signature_Blocks, each of which corresponds to a different algorithm suite. Each of the Signature_Blocks will contain a Signature Segment for each AS number (i.e., Secure_Path Segment) in the Secure_Path. In the most common case, the BGPsec_Path attribute will contain only a single Signature_Block. However, in order to enable a transition from an old algorithm suite to a new algorithm suite (without a flag day), it will be necessary to include two Signature_Blocks (one for the old algorithm suite and one for the new algorithm suite) during the transition period. (See Section 6.1 for more discussion of algorithm transitions.) The format of the Signature_Blocks is described below in Section 3.2.

3.1. Secure_Path

A detailed description of the Secure_Path information in the BGPsec_Path attribute is provided here.

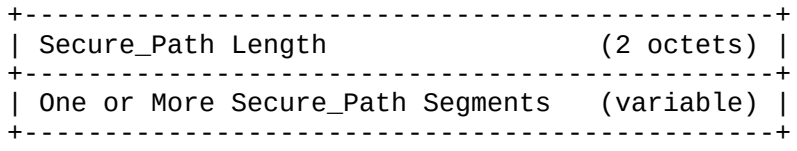


Figure 4: Secure_Path format.

The specification for the Secure_Path field is provided in Figure 4 and Figure 5. The Secure_Path Length contains the length (in octets) of the entire Secure_Path (including the two octets used to express this length field). As explained below, each Secure_Path Segment is six octets long. Note that this means the Secure_Path Length is two greater than six times the number Secure_Path Segments (i.e., the number of AS numbers in the path).

The Secure_Path contains one Secure_Path Segment (see Figure 5) for each Autonomous System in the path to the originating AS of the prefix specified in the update message. (Note: Repeated Autonomous Systems are compressed out using the pCount field as discussed below.)

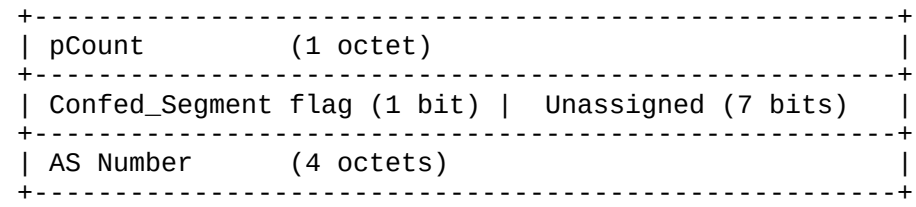


Figure 5: Secure_Path Segment format.

The AS Number (in Figure 5) is the AS number of the BGP speaker that added this Secure_Path Segment to the BGPsec_Path attribute. (See Section 4 for more information on populating this field.)

The pCount field contains the number of repetitions of the associated autonomous system number that the signature covers. This field enables a BGPsec speaker to mimic the semantics of prepending multiple copies of their AS to the AS_PATH without requiring the speaker to generate multiple signatures. Note that Section 9.1.2.2 ("Breaking Ties") in [RFC4271] mentions "number of AS numbers" in the AS_PATH attribute that is used in the route selection process. This metric (number of AS numbers) is the same as the AS path length obtained in BGPsec by summing the pCount values in the BGPsec_Path attribute. The pCount field is also useful in managing route servers (see Section 4.2), AS confederations (see Section 4.3), and AS Number migrations (see [I-D.ietf-sidr-as-migration] for details).

The left most (i.e. the most significant) bit of the Flags field in Figure 5 is the Confed_Segment flag. The Confed_Segment flag is set to one to indicate that the BGPsec speaker that constructed this Secure_Path Segment is sending the update message to a peer AS within the same Autonomous System confederation [RFC5065]. (That is, a sequence of consecutive Confed_Segment flags are set in a BGPsec update message whenever, in a non-BGPsec update message, an AS_PATH segment of type AS_CONFED_SEQUENCE occurs.) In all other cases the Confed_Segment flag is set to zero.

The remaining seven bits of the Flags are unassigned and MUST be set to zero by the sender, and ignored by the receiver. Note, however, that the signature is computed over all eight bits of the flags field.

As stated earlier in Section 2.2, BGPsec peering requires that the peering ASes MUST each support four-byte AS numbers. Currently-assigned two-byte AS numbers are converted into four-byte AS numbers by setting the two high-order octets of the four-octet field to zero [RFC6793].

3.2. Signature_Block

A detailed description of the Signature_Blocks in the BGPsec_Path attribute is provided here using Figure 6 and Figure 7.

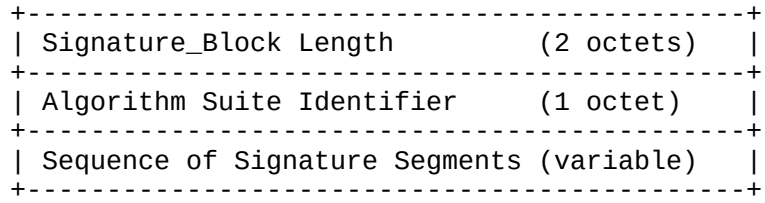


Figure 6: Signature_Block format.

The Signature_Block Length in Figure 6 is the total number of octets in the Signature_Block (including the two octets used to express this length field).

The Algorithm Suite Identifier is a one-octet identifier specifying the digest algorithm and digital signature algorithm used to produce the digital signature in each Signature Segment. An IANA registry of algorithm identifiers for use in BGPsec is specified in the BGPsec algorithms document [I-D.ietf-sidr-bgpsec-algs].

A Signature_Block in Figure 6 has exactly one Signature Segment (see Figure 7) for each Secure_Path Segment in the Secure_Path portion of the BGPsec_Path Attribute. (That is, one Signature Segment for each distinct AS on the path for the prefix in the Update message.)

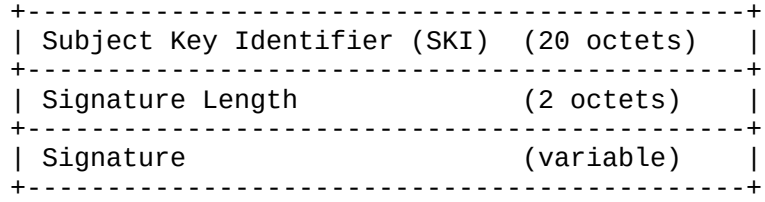


Figure 7: Signature Segment format.

The Subject Key Identifier (SKI) field in Figure 7 contains the value in the Subject Key Identifier extension of the RPKI router certificate [RFC6487] that is used to verify the signature (see Section 5 for details on validity of BGPsec update messages). The SKI field has a fixed 20 octets size. See Section 6.2 for considerations for the SKI size.

The Signature Length field contains the size (in octets) of the value in the Signature field of the Signature Segment.

The Signature in Figure 7 contains a digital signature that protects the prefix and the BGPsec_Path attribute (see Section 4 and Section 5 for details on signature generation and validation, respectively).

4. BGPsec Update Messages

Section 4.1 provides general guidance on the creation of BGPsec Update Messages -- that is, update messages containing the BGPsec_Path attribute.

Section 4.2 specifies how a BGPsec speaker generates the BGPsec_Path attribute to include in a BGPsec Update message.

Section 4.3 contains special processing instructions for members of an autonomous system confederation [RFC5065]. A BGPsec speaker that is not a member of such a confederation MUST NOT set the Confed_Segment flag in its Secure_Path Segment (i.e. leave the flag bit at default value zero) in all BGPsec update messages it sends.

Section 4.4 contains instructions for reconstructing the AS_PATH attribute in cases where a BGPsec speaker receives an update message with a BGPsec_Path attribute and wishes to propagate the update message to a peer who does not support BGPsec.

4.1. General Guidance

The information protected by the signature on a BGPsec update message includes the AS number of the peer to whom the update message is being sent. Therefore, if a BGPsec speaker wishes to send a BGPsec update to multiple BGP peers, it MUST generate a separate BGPsec update message for each unique peer AS to whom the update message is sent.

A BGPsec update message MUST advertise a route to only a single prefix. This is because a BGPsec speaker receiving an update message with multiple prefixes would be unable to construct a valid BGPsec update message (i.e., valid path signatures) containing a subset of the prefixes in the received update. If a BGPsec speaker wishes to advertise routes to multiple prefixes, then it MUST generate a separate BGPsec update message for each prefix. Additionally, a BGPsec update message MUST use the MP_REACH_NLRI [RFC4760] attribute to encode the prefix.

The BGPsec_Path attribute and the AS_PATH attribute are mutually exclusive. That is, any update message containing the BGPsec_Path

attribute MUST NOT contain the AS_PATH attribute. The information that would be contained in the AS_PATH attribute is instead conveyed in the Secure_Path portion of the BGPsec_Path attribute.

In order to create or add a new signature to a BGPsec update message with a given algorithm suite, the BGPsec speaker MUST possess a private key suitable for generating signatures for this algorithm suite. Additionally, this private key must correspond to the public key in a valid Resource PKI end-entity certificate whose AS number resource extension includes the BGPsec speaker's AS number [I-D.ietf-sidr-bgpsec-pki-profiles]. Note also that new signatures are only added to a BGPsec update message when a BGPsec speaker is generating an update message to send to an external peer (i.e., when the AS number of the peer is not equal to the BGPsec speaker's own AS number).

The Resource PKI enables the legitimate holder of IP address prefix(es) to issue a signed object, called a Route Origination Authorization (ROA), that authorizes a given AS to originate routes to a given set of prefixes (see RFC 6482 [RFC6482]). It is expected that most relying parties will utilize BGPsec in tandem with origin validation (see RFC 6483 [RFC6483] and RFC 6811 [RFC6811]). Therefore, it is RECOMMENDED that a BGPsec speaker only originate a BGPsec update advertising a route for a given prefix if there exists a valid ROA authorizing the BGPsec speaker's AS to originate routes to this prefix.

If a BGPsec router has received only a non-BGPsec update message containing the AS_PATH attribute (instead of the BGPsec_Path attribute) from a peer for a given prefix, then it MUST NOT attach a BGPsec_Path attribute when it propagates the update message. (Note that a BGPsec router may also receive a non-BGPsec update message from an internal peer without the AS_PATH attribute, i.e., with just the NLRI in it. In that case, the prefix is originating from that AS, and if it is selected for advertisement, the BGPsec speaker SHOULD attach a BGPsec_Path attribute and send a signed route (for that prefix) to its external BGPsec-speaking peers.)

Conversely, if a BGPsec router has received a BGPsec update message (with the BGPsec_Path attribute) from a peer for a given prefix and it chooses to propagate that peer's route for the prefix, then it SHOULD propagate the route as a BGPsec update message containing the BGPsec_Path attribute.

Note that removing BGPsec signatures (i.e., propagating a route advertisement without the BGPsec_Path attribute) has significant security ramifications. (See Section 8 for discussion of the security ramifications of removing BGPsec signatures.) Therefore,

when a route advertisement is received via a BGPsec update message, propagating the route advertisement without the BGPsec_Path attribute is NOT RECOMMENDED, unless the message is sent to a peer that did not advertise the capability to receive BGPsec update messages (see Section 4.4).

Furthermore, note that when a BGPsec speaker propagates a route advertisement with the BGPsec_Path attribute it is not attesting to the validation state of the update message it received. (See Section 8 for more discussion of the security semantics of BGPsec signatures.)

If the BGPsec speaker is producing an update message which would, in the absence of BGPsec, contain an AS_SET (e.g., the BGPsec speaker is performing proxy aggregation), then the BGPsec speaker MUST NOT include the BGPsec_Path attribute. In such a case, the BGPsec speaker MUST remove any existing BGPsec_Path in the received advertisement(s) for this prefix and produce a traditional (non-BGPsec) update message. It should be noted that BCP 172 [RFC6472] recommends against the use of AS_SET and AS_CONFED_SET in the AS_PATH of BGP updates.

The case where the BGPsec speaker sends a BGPsec update message to an iBGP peer is quite simple. When originating a new route advertisement and sending it to a BGPsec-capable iBGP peer, the BGPsec speaker omits the BGPsec_Path attribute. When originating a new route advertisement and sending it to a non-BGPsec iBGP peer, the BGPsec speaker includes an empty AS_PATH attribute in the update message. (An empty AS_PATH attribute is one whose length field contains the value zero [RFC4271].) When a BGPsec speaker chooses to forward a BGPsec update message to an iBGP peer, the BGPsec_Path attribute SHOULD NOT be removed, unless the peer doesn’t support BGPsec. In the case when an iBGP peer doesn’t support BGPsec, then a BGP update with AS_PATH is reconstructed from the BGPsec update and then forwarded (see Section 4.4). In particular, when forwarding to a BGPsec-capable iBGP (or eBGP) peer, the BGPsec_Path attribute SHOULD NOT be removed even in the case where the BGPsec update message has not been successfully validated. (See Section 5 for more information on validation, and Section 8 for the security ramifications of removing BGPsec signatures.)

All BGPsec update messages MUST conform to BGP’s maximum message size. If the resulting message exceeds the maximum message size, then the guidelines in Section 9.2 of RFC 4271 [RFC4271] MUST be followed.

4.2. Constructing the BGPsec_Path Attribute

When a BGPsec speaker receives a BGPsec update message containing a BGPsec_Path attribute (with one or more signatures) from an (internal or external) peer, it may choose to propagate the route advertisement by sending it to its other (internal or external) peers. When sending the route advertisement to an internal BGPsec-speaking peer, the BGPsec_Path attribute SHALL NOT be modified. When sending the route advertisement to an external BGPsec-speaking peer, the following procedures are used to form or update the BGPsec_Path attribute.

To generate the BGPsec_Path attribute on the outgoing update message, the BGPsec speaker first generates a new Secure_Path Segment. Note that if the BGPsec speaker is not the origin AS and there is an existing BGPsec_Path attribute, then the BGPsec speaker prepends its new Secure_Path Segment (places in first position) onto the existing Secure_Path.

The AS number in this Secure_Path Segment MUST match the AS number in the Subject field of the Resource PKI router certificate that will be used to verify the digital signature constructed by this BGPsec speaker (see Section 3.1.1 in [I-D.ietf-sidr-bgpsec-pki-profiles] and RFC 6487 [RFC6487]).

The pCount field of the Secure_Path Segment is typically set to the value 1. However, a BGPsec speaker may set the pCount field to a value greater than 1. Setting the pCount field to a value greater than one has the same semantics as repeating an AS number multiple times in the AS_PATH of a non-BGPsec update message (e.g., for traffic engineering purposes).

To prevent unnecessary processing load in the validation of BGPsec signatures, a BGPsec speaker SHOULD NOT produce multiple consecutive Secure_Path Segments with the same AS number. This means that to achieve the semantics of prepending the same AS number k times, a BGPsec speaker SHOULD produce a single Secure_Path Segment -- with pCount of k -- and a single corresponding Signature Segment.

A route server that participates in the BGP control plane, but does not act as a transit AS in the data plane, may choose to set pCount to 0. This option enables the route server to participate in BGPsec and obtain the associated security guarantees without increasing the length of the AS path. (Note that BGPsec speakers compute the length of the AS path by summing the pCount values in the BGPsec_Path attribute, see Section 5.) However, when a route server sets the pCount value to 0, it still inserts its AS number into the Secure_Path Segment, as this information is needed to validate the

signature added by the route server. See [I-D.ietf-sidr-as-migration] for a discussion of setting pCount to 0 to facilitate AS Number Migration. Also, see Section 4.3 for the use of pCount=0 in the context of an AS confederation. See Section 7.2 for operational guidance for configuring a BGPsec router for setting pCount=0 and/or accepting pCount=0 from a peer.

Next, the BGPsec speaker generates one or two Signature_Blocks. Typically, a BGPsec speaker will use only a single algorithm suite, and thus create only a single Signature_Block in the BGPsec_Path attribute. However, to ensure backwards compatibility during a period of transition from a 'current' algorithm suite to a 'new' algorithm suite, it will be necessary to originate update messages that contain a Signature_Block for both the 'current' and the 'new' algorithm suites (see Section 6.1).

If the received BGPsec update message contains two Signature_Blocks and the BGPsec speaker supports both of the corresponding algorithm suites, then the new update message generated by the BGPsec speaker MUST include both of the Signature_Blocks. If the received BGPsec update message contains two Signature_Blocks and the BGPsec speaker only supports one of the two corresponding algorithm suites, then the BGPsec speaker MUST remove the Signature_Block corresponding to the algorithm suite that it does not understand. If the BGPsec speaker does not support the algorithm suites in any of the Signature_Blocks contained in the received update message, then the BGPsec speaker MUST NOT propagate the route advertisement with the BGPsec_Path attribute. (That is, if it chooses to propagate this route advertisement at all, it MUST do so as an unsigned BGP update message. See Section 4.4 for more information on converting to an unsigned BGP message.)

Note that in the case where the BGPsec_Path has two Signature_Blocks (corresponding to different algorithm suites), the validation algorithm (see Section 5.2) deems a BGPsec update message to be 'Valid' if there is at least one supported algorithm suite (and corresponding Signature_Block) that is deemed 'Valid'. This means that a 'Valid' BGPsec update message may contain a Signature_Block which is not deemed 'Valid' (e.g., contains signatures that BGPsec does not successfully verify). Nonetheless, such Signature_Blocks MUST NOT be removed. (See Section 8 for a discussion of the security ramifications of this design choice.)

For each Signature_Block corresponding to an algorithm suite that the BGPsec speaker does support, the BGPsec speaker MUST add a new Signature Segment to the Signature_Block. This Signature Segment is prepended to the list of Signature Segments (placed in the first position) so that the list of Signature Segments appears in the same

order as the corresponding Secure_Path Segments. The BGPsec speaker populates the fields of this new Signature Segment as follows.

The Subject Key Identifier field in the new segment is populated with the identifier contained in the Subject Key Identifier extension of the RPKI router certificate corresponding to the BGPsec speaker [I-D.ietf-sidr-bgpsec-pki-profiles]. This Subject Key Identifier will be used by recipients of the route advertisement to identify the proper certificate to use in verifying the signature.

The Signature field in the new segment contains a digital signature that binds the prefix and BGPsec_Path attribute to the RPKI router certificate corresponding to the BGPsec speaker. The digital signature is computed as follows:

- For clarity, let us number the Secure_Path and corresponding Signature Segments from 1 to N as follows. Let Secure_Path Segment 1 and Signature Segment 1 be the segments produced by the origin AS. Let Secure_Path Segment 2 and Signature Segment 2 be the segments added by the next AS after the origin. Continue this method of numbering and ultimately let Secure_Path Segment N and Signature Segment N be those that are being added by the current AS. The current AS (Nth AS) is signing and forwarding the update to the next AS (i.e. (N+1)th AS) in the chain of ASes that form the AS path.

- In order to construct the digital signature for Signature Segment N (the Signature Segment being produced by the current AS), first construct the sequence of octets to be hashed as shown in Figure 8. This sequence of octets includes all the data that the Nth AS attests to by adding its digital signature in the update which is being forwarded to a BGPsec speaker in the (N+1)th AS. (For the design rationale for choosing the specific structure in Figure 8, please see [Borchert].)

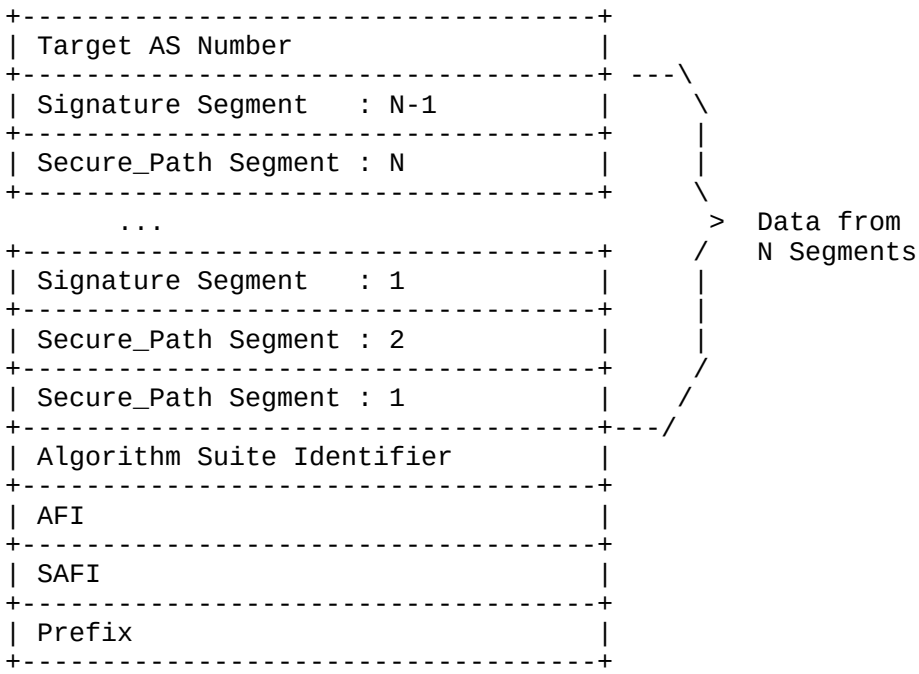


Figure 8: Sequence of octets to be hashed.

The elements in this sequence (Figure 8) MUST be ordered exactly as shown. The 'Target AS Number' is the AS to whom the BGPsec speaker intends to send the update message. (Note that the 'Target AS Number' is the AS number announced by the peer in the OPEN message of the BGP session within which the update is sent.) The Secure_Path and Signature Segments (1 through N-1) are obtained from the BGPsec_Path attribute. Finally, the Address Family Identifier (AFI), Subsequent Address Family Identifier (SAFI), and Prefix fields are obtained from the MP_REACH_NLRI attribute [RFC4760]. Additionally, in the Prefix field all of the trailing bits MUST be set to zero when constructing this sequence.

- Apply to this octet sequence (in Figure 8) the digest algorithm (for the algorithm suite of this Signature_Block) to obtain a digest value.

- Apply to this digest value the signature algorithm, (for the algorithm suite of this Signature_Block) to obtain the digital signature. Then populate the Signature Field (in Figure 7) with this digital signature.

The Signature Length field (in Figure 7) is populated with the length (in octets) of the value in the Signature field.

4.3. Processing Instructions for Confederation Members

Members of autonomous system confederations [RFC5065] MUST additionally follow the instructions in this section for processing BGPsec update messages.

When a BGPsec speaker in an AS confederation receives a BGPsec update from a peer that is external to the confederation and chooses to propagate the update within the confederation, then it first adds a signature signed to its own Member-AS (i.e. the Target AS number is the BGPsec speaker's Member-AS number). In this internally modified update, the newly added Secure_Path Segment contains the public AS number (i.e. Confederation Identifier), the Segment's pCount value is set to 0, and Confed_Segment flag is set to one. Setting pCount=0 in this case helps ensure that the AS path length is not unnecessarily incremented. The newly added signature is generated using a private key corresponding to the public AS number of the confederation. The BGPsec speaker propagates the modified update to its peers within the confederation.

Any BGPsec_Path modifications mentioned below in the context of propagation of the update within the confederation are in addition to the modification described above (i.e. with pCount=0).

When a BGPsec speaker sends a BGPsec update message to a peer that belongs within its own Member-AS, the confederation member SHALL NOT modify the BGPsec_Path attribute. When a BGPsec speaker sends a BGPsec update message to a peer that is within the same confederation but in a different Member-AS, the BGPsec speaker puts its Member-AS number in the AS Number field of the Secure_Path Segment that it adds to the BGPsec update message. Additionally, in this case, the Member-AS that generates the Secure_Path Segment sets the Confed_Segment flag to one. Further, the signature is generated with a private key corresponding to the BGPsec speaker's Member-AS Number. (Note: In this document, intra-Member-AS peering is regarded as iBGP and inter-Member-AS peering is regarded as eBGP. The latter is also known as confederation-eBGP.)

Within a confederation, the verification of BGPsec signatures added by other members of the confederation is optional. Note that if a confederation chooses not to verify digital signatures within the confederation, then BGPsec is able to provide no assurances about the integrity of the Member-AS Numbers placed in Secure_Path Segments where the Confed_Segment flag is set to one.

When a confederation member receives a BGPsec update message from a peer within the confederation and propagates it to a peer outside the confederation, it needs to remove all of the Secure_Path Segments added by confederation members as well as the corresponding Signature Segments. To do this, the confederation member propagating the route outside the confederation does the following:

- First, starting with the most recently added Secure_Path Segment, remove all of the consecutive Secure_Path Segments that have the Confed_Segment flag set to one. Stop this process once a Secure_Path Segment is reached which has its Confed_Segment flag set to zero. Keep a count of the number of segments removed in this fashion.

- Second, starting with the most recently added Signature Segment, remove a number of Signature Segments equal to the number of Secure_Path Segments removed in the previous step. (That is, remove the K most recently added Signature Segments, where K is the number of Secure_Path Segments removed in the previous step.)

- Finally, add a Secure_Path Segment containing, in the AS field, the AS Confederation Identifier (the public AS number of the confederation) as well as a corresponding Signature Segment. Note that all fields other than the AS field are populated as per Section 4.2.

Finally, as discussed above, an AS confederation MAY optionally decide that its members will not verify digital signatures added by members. In such a confederation, when a BGPsec speaker runs the algorithm in Section 5.2, the BGPsec speaker, during the process of Signature verifications, first checks whether the Confed_Segment flag in a Secure_Path Segment is set to one. If the flag is set to one, the BGPsec speaker skips the verification for the corresponding Signature, and immediately moves on to the next Secure_Path Segment. Note that as specified in Section 5.2, it is an error when a BGPsec speaker receives from a peer, who is not in the same AS confederation, a BGPsec update containing a Confed_Segment flag set to one.

4.4. Reconstructing the AS_PATH Attribute

BGPsec update messages do not contain the AS_PATH attribute. However, the AS_PATH attribute can be reconstructed from the BGPsec_Path attribute. This is necessary in the case where a route advertisement is received via a BGPsec update message and then propagated to a peer via a non-BGPsec update message (e.g., because the latter peer does not support BGPsec). Note that there may be additional cases where an implementation finds it useful to perform

this reconstruction. Before attempting to reconstruct an AS_PATH for the purpose of forwarding an unsigned (non-BGPsec) update to a peer, a BGPsec speaker MUST perform the basic integrity checks listed in Section 5.2 to ensure that the received BGPsec update is properly formed.

The AS_PATH attribute can be constructed from the BGPsec_Path attribute as follows. Starting with a blank AS_PATH attribute, process the Secure_Path Segments in order from least-recently added (corresponding to the origin) to most-recently added. For each Secure_Path Segment perform the following steps:

1. If the Secure_Path Segment has pCount=0, then do nothing (i.e. move on to process the next Secure_Path Segment).

2. If the Secure_Path Segment has pCount greater than 0 and the Confed_Segment flag is set to one, then look at the most-recently added segment in the AS_PATH.

   * In the case where the AS_PATH is blank or in the case where the most-recently added segment is of type AS_SEQUENCE, add (prepend to the AS_PATH) a new AS_PATH segment of type AS_CONFED_SEQUENCE. This segment of type AS_CONFED_SEQUENCE shall contain a number of elements equal to the pCount field in the current Secure_Path Segment. Each of these elements shall be the AS number contained in the current Secure_Path Segment. (That is, if the pCount field is X, then the segment of type AS_CONFED_SEQUENCE contains X copies of the Secure_Path Segment’s AS Number field.)

   * In the case where the most-recently added segment in the AS_PATH is of type AS_CONFED_SEQUENCE then add (prepend to the segment) a number of elements equal to the pCount field in the current Secure_Path Segment. The value of each of these elements shall be the AS number contained in the current Secure_Path Segment. (That is, if the pCount field is X, then add X copies of the Secure_Path Segment’s AS Number field to the existing AS_CONFED_SEQUENCE.)

3. If the Secure_Path Segment has pCount greater than 0 and the Confed_Segment flag is set to zero, then look at the most-recently added segment in the AS_PATH.

   * In the case where the AS_PATH is blank or in the case where the most-recently added segment is of type AS_CONFED_SEQUENCE, add (prepend to the AS_PATH) a new AS_PATH segment of type AS_SEQUENCE. This segment of type AS_SEQUENCE shall contain a number of elements equal to the pCount field in the current

Secure_Path Segment. Each of these elements shall be the AS number contained in the current Secure_Path Segment. (That is, if the pCount field is X, then the segment of type AS_SEQUENCE contains X copies of the Secure_Path Segment's AS Number field.)

* In the case where the most recently added segment in the AS_PATH is of type AS_SEQUENCE then add (prepend to the segment) a number of elements equal to the pCount field in the current Secure_Path Segment. The value of each of these elements shall be the AS number contained in the current Secure_Path Segment. (That is, if the pCount field is X, then add X copies of the Secure_Path Segment's AS Number field to the existing AS_SEQUENCE.)

As part of the above described procedure, the following additional actions are performed in order not to exceed the size limitations of AS_SEQUENCE and AS_CONFED_SEQUENCE. While adding the next Secure_Path Segment (with its prepends, if any) to the AS_PATH being assembled, if it would cause the AS_SEQUENCE (or AS_CONFED_SEQUENCE) at hand to exceed the limit of 255 AS numbers per segment [RFC4271] [RFC5065], then the BGPsec speaker would follow the recommendations in RFC 4271 [RFC4271] and RFC 5065 [RFC5065] of creating another segment of the same type (AS_SEQUENCE or AS_CONFED_SEQUENCE) and continue filling that.

Finally, one special case of reconstruction of AS_PATH is when the BGPsec_Path attribute is absent. As explained in Section 4.1, when a BGPsec speaker originates a prefix and sends it to a BGPsec-capable iBGP peer, the BGPsec_Path is not attached. So when received from a BGPsec-capable iBGP peer, no BGPsec_Path attribute in a BGPsec update is equivalent to an empty AS_PATH [RFC4271].

5. Processing a Received BGPsec Update

Upon receiving a BGPsec update message from an external (eBGP) peer, a BGPsec speaker SHOULD validate the message to determine the authenticity of the path information contained in the BGPsec_Path attribute. Typically, a BGPsec speaker will also wish to perform origin validation (see RFC 6483 [RFC6483] and RFC 6811 [RFC6811]) on an incoming BGPsec update message, but such validation is independent of the validation described in this section.

Section 5.1 provides an overview of BGPsec validation and Section 5.2 provides a specific algorithm for performing such validation. (Note that an implementation need not follow the specific algorithm in Section 5.2 as long as the input/output behavior of the validation is identical to that of the algorithm in Section 5.2.) During

exceptional conditions (e.g., the BGPsec speaker receives an incredibly large number of update messages at once) a BGPsec speaker MAY temporarily defer validation of incoming BGPsec update messages. The treatment of such BGPsec update messages, whose validation has been deferred, is a matter of local policy. However, an implementation SHOULD ensure that deferment of validation and status of deferred messages is visible to the operator.

The validity of BGPsec update messages is a function of the current RPKI state. When a BGPsec speaker learns that RPKI state has changed (e.g., from an RPKI validating cache via the RPKI-to-Router protocol [I-D.ietf-sidr-rpki-rtr-rfc6810-bis]), the BGPsec speaker MUST re-run validation on all affected update messages stored in its Adj-RIB-In [RFC4271]. For example, when a given RPKI router certificate ceases to be valid (e.g., it expires or is revoked), all update messages containing a signature whose SKI matches the SKI in the given certificate MUST be re-assessed to determine if they are still valid. If this reassessment determines that the validity state of an update has changed then, depending on local policy, it may be necessary to re-run best path selection.

BGPsec update messages do not contain an AS_PATH attribute. The Secure_Path contains AS path information for the BGPsec update message. Therefore, a BGPsec speaker MUST utilize the AS path information in the Secure_Path in all cases where it would otherwise use the AS path information in the AS_PATH attribute. The only exception to this rule is when AS path information must be updated in order to propagate a route to a peer (in which case the BGPsec speaker follows the instructions in Section 4). Section 4.4 provides an algorithm for constructing an AS_PATH attribute from a BGPsec_Path attribute. Whenever the use of AS path information is called for (e.g., loop detection, or use of AS path length in best path selection) the externally visible behavior of the implementation shall be the same as if the implementation had run the algorithm in Section 4.4 and used the resulting AS_PATH attribute as it would for a non-BGPsec update message.

5.1. Overview of BGPsec Validation

Validation of a BGPsec update message makes use of data from RPKI router certificates. In particular, it is necessary that the recipient have access to the following data obtained from valid RPKI router certificates: the AS Number, Public Key and Subject Key Identifier from each valid RPKI router certificate.

Note that the BGPsec speaker could perform the validation of RPKI router certificates on its own and extract the required data, or it could receive the same data from a trusted cache that performs RPKI

validation on behalf of (some set of) BGPsec speakers. (For example, the trusted cache could deliver the necessary validity information to the BGPsec speaker using the router key PDU for the RPKI-to-Router protocol [I-D.ietf-sidr-rpki-rtr-rfc6810-bis].)

To validate a BGPsec update message containing the BGPsec_Path attribute, the recipient performs the validation steps specified in Section 5.2. The validation procedure results in one of two states: 'Valid' and 'Not Valid'.

It is expected that the output of the validation procedure will be used as an input to BGP route selection. That said, BGP route selection, and thus the handling of the validation states is a matter of local policy, and is handled using local policy mechanisms. Implementations SHOULD enable operators to set such local policy on a per-session basis. (That is, it is expected that some operators will choose to treat BGPsec validation status differently for update messages received over different BGP sessions.)

BGPsec validation needs only be performed at the eBGP edge. The validation status of a BGP signed/unsigned update MAY be conveyed via iBGP from an ingress edge router to an egress edge router via some mechanism, according to local policy within an AS. As discussed in Section 4, when a BGPsec speaker chooses to forward a (syntactically correct) BGPsec update message, it SHOULD be forwarded with its BGPsec_Path attribute intact (regardless of the validation state of the update message). Based entirely on local policy, an egress router receiving a BGPsec update message from within its own AS MAY choose to perform its own validation.

5.2. Validation Algorithm

This section specifies an algorithm for validation of BGPsec update messages. A conformant implementation MUST include a BGPsec update validation algorithm that is functionally equivalent to the externally visible behavior of this algorithm.

First, the recipient of a BGPsec update message performs a check to ensure that the message is properly formed. Both syntactical and protocol violation errors are checked. BGPsec_Path attribute MUST be present when a BGPsec update is received from an external (eBGP) BGPsec peer and also when such an update is propagated to an internal (iBGP) BGPsec peer (see Section 4.2). The error checks specified in Section 6.3 of [RFC4271] are performed, except that for BGPsec updates the checks on the AS_PATH attribute do not apply and instead the following checks on BGPsec_Path attribute are performed:

1. Check to ensure that the entire BGPsec_Path attribute is syntactically correct (conforms to the specification in this document).

2. Check that AS number in the most recently added Secure_Path Segment (i.e. the one corresponding to the eBGP peer from which the update message was received) matches the AS number of that peer as specified in the BGP OPEN message. (Note: This check is performed only at an ingress BGPsec routers where the update is first received from a peer AS.)

3. Check that each Signature_Block contains one Signature Segment for each Secure_Path Segment in the Secure_Path portion of the BGPsec_Path attribute. (Note that the entirety of each Signature_Block MUST be checked to ensure that it is well formed, even though the validation process may terminate before all signatures are cryptographically verified.)

4. Check that the update message does not contain an AS_PATH attribute.

5. If the update message was received from an BGPsec peer that is not a member of the BGPsec speaker's AS confederation, check to ensure that none of the Secure_Path Segments contain a Flags field with the Confed_Segment flag set to one.

6. If the update message was received from a BGPsec peer that is a member of the BGPsec speaker's AS confederation, check to ensure that the Secure_Path Segment corresponding to that peer contains a Flags field with the Confed_Segment flag set to one.

7. If the update message was received from a peer that is not expected to set pCount=0 (see Section 4.2 and Section 4.3) then check to ensure that the pCount field in the most-recently added Secure_Path Segment is not equal to zero. (Note: See router configuration guidance related to this in Section 7.2.)

8. Using the equivalent of AS_PATH corresponding to the Secure_Path in the update (see Section 4.4), check that the local AS number is not present in the AS path (i.e. rule out AS loop).

If any of these checks fail, it is an error in the BGPsec_Path attribute. BGPsec speakers MUST handle any syntactical or protocol errors in the BGPsec_Path attribute using the "treat-as-withdraw" approach as defined in RFC 7606 [RFC7606]. (Note: Since the AS number of a transparent route server does appear in the Secure_Path with pCount=0, the route server MAY check if its local AS is listed

in the Secure_Path, and this check MAY be included in the loop detection check listed above.)

Next, the BGPsec speaker examines the Signature_Blocks in the BGPsec_Path attribute. A Signature_Block corresponding to an algorithm suite that the BGPsec speaker does not support is not considered in validation. If there is no Signature_Block corresponding to an algorithm suite that the BGPsec speaker supports, then in order to consider the update in the route selection process, the BGPsec speaker MUST strip the Signature_Block(s), reconstruct the AS_PATH from the Secure_Path (see Section 4.4), and treat the update as if it was received as an unsigned BGP update.

For each remaining Signature_Block (corresponding to an algorithm suite supported by the BGPsec speaker), the BGPsec speaker iterates through the Signature Segments in the Signature_Block, starting with the most recently added segment (and concluding with the least recently added segment). Note that there is a one-to-one correspondence between Signature Segments and Secure_Path Segments within the BGPsec_Path attribute. The following steps make use of this correspondence.

- (Step 1): Let there be K AS hops in a received BGPsec_Path attribute that is to be validated. Let AS(1), AS(2), ..., AS(K+1) denote the sequence of AS numbers from the origin AS to the validating AS. Let Secure_Path Segment N and Signature Segment N in the BGPsec_Path attribute refer to those corresponding to AS(N) (where N = 1, 2, ..., K). The BGPsec speaker that is processing and validating the BGPsec_Path attribute resides in AS(K+1). Let Signature Segment N be the Signature Segment that is currently being verified.

- (Step 2): Locate the public key needed to verify the signature (in the current Signature Segment). To do this, consult the valid RPKI router certificate data and look up all valid (AS, SKI, Public Key) triples in which the AS matches the AS number in the corresponding Secure_Path Segment. Of these triples that match the AS number, check whether there is an SKI that matches the value in the Subject Key Identifier field of the Signature Segment. If this check finds no such matching SKI value, then mark the entire Signature_Block as 'Not Valid' and proceed to the next Signature_Block.

- (Step 3): Compute the digest function (for the given algorithm suite) on the appropriate data.

  In order to verify the digital signature in Signature Segment N, construct the sequence of octets to be hashed as shown in Figure 9

(using the notations defined in Step 1). (Note that this sequence is the same sequence that was used by AS(N) that created the Signature Segment N (see Section 4.2 and Figure 8).)

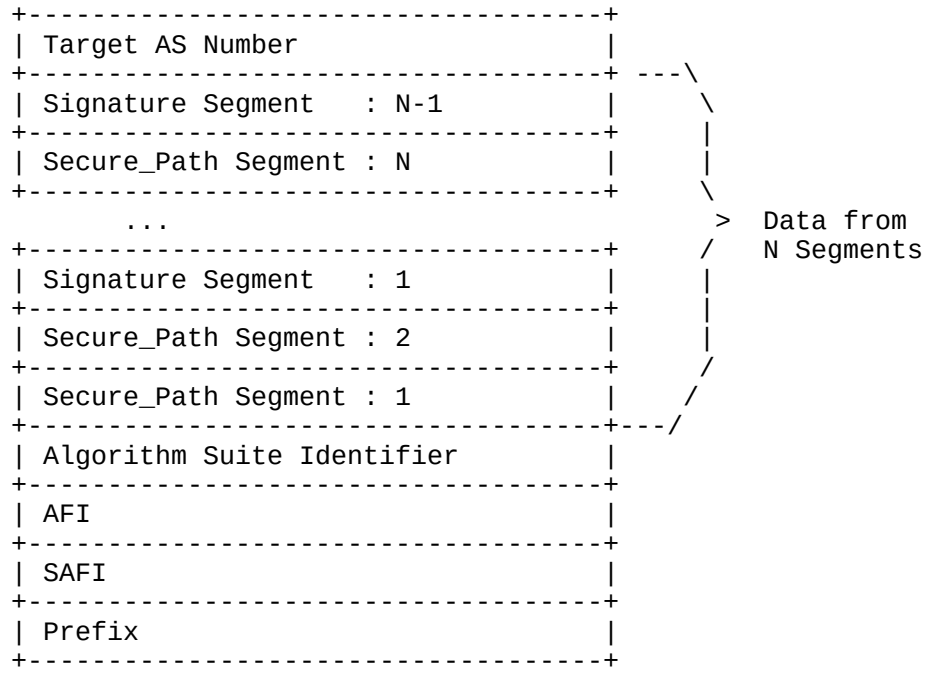


Figure 9: The Sequence of octets to be hashed for signature verification of Signature Segment N; N = 1,2, ..., K, where K is the number of AS hops in the BGPsec_Path attribute.

The elements in this sequence (Figure 9) MUST be ordered exactly as shown. For the first segment to be processed (the most recently added segment (i.e. N = K) given that there are K hops in the Secure_Path), the 'Target AS Number' is AS(K+1), the AS number of the BGPsec speaker validating the update message. Note that if a BGPsec speaker uses multiple AS Numbers (e.g., the BGPsec speaker is a member of a confederation), the AS number used here MUST be the AS number announced in the OPEN message for the BGP session over which the BGPsec update was received.

For each other Signature Segment (N smaller than K), the 'Target AS Number' is AS(N+1), the AS number in the Secure_Path Segment that corresponds to the Signature Segment added immediately after the one being processed. (That is, in the Secure_Path Segment

that corresponds to the Signature Segment that the validator just finished processing.)

The Secure_Path and Signature Segment are obtained from the BGPsec_Path attribute. The Address Family Identifier (AFI), Subsequent Address Family Identifier (SAFI), and Prefix fields are obtained from the MP_REACH_NLRI attribute [RFC4760]. Additionally, in the Prefix field all of the trailing bits MUST be set to zero when constructing this sequence.

- o (Step 4): Use the signature validation algorithm (for the given algorithm suite) to verify the signature in the current segment. That is, invoke the signature validation algorithm on the following three inputs: the value of the Signature field in the current segment; the digest value computed in Step 3 above; and the public key obtained from the valid RPKI data in Step 2 above. If the signature validation algorithm determines that the signature is invalid, then mark the entire Signature_Block as 'Not Valid' and proceed to the next Signature_Block. If the signature validation algorithm determines that the signature is valid, then continue processing Signature Segments (within the current Signature_Block).

If all Signature Segments within a Signature_Block pass validation (i.e., all segments are processed and the Signature_Block has not yet been marked 'Not Valid'), then the Signature_Block is marked as 'Valid'.

If at least one Signature_Block is marked as 'Valid', then the validation algorithm terminates and the BGPsec update message is deemed to be 'Valid'. (That is, if a BGPsec update message contains two Signature_Blocks then the update message is deemed 'Valid' if the first Signature_Block is marked 'Valid' OR the second Signature_Block is marked 'Valid'.)

6. Algorithms and Extensibility

6.1. Algorithm Suite Considerations

Note that there is currently no support for bilateral negotiation (using BGP capabilities) between BGPsec peers to use a particular (digest and signature) algorithm suite. This is because the algorithm suite used by the sender of a BGPsec update message MUST be understood not only by the peer to whom it is directly sending the message, but also by all BGPsec speakers to whom the route advertisement is eventually propagated. Therefore, selection of an algorithm suite cannot be a local matter negotiated by BGP peers, but instead must be coordinated throughout the Internet.

To this end, a mandatory algorithm suites document exists which specifies a mandatory-to-use 'current' algorithm suite for use by all BGPsec speakers [I-D.ietf-sidr-bgpsec-algs].

It is anticipated that, in the future, the mandatory algorithm suites document will be updated to specify a transition from the 'current' algorithm suite to a 'new' algorithm suite. During the period of transition, all BGPsec update messages SHOULD simultaneously use both the 'current' algorithm suite and the 'new' algorithm suite. (Note that Section 3 and Section 4 specify how the BGPsec_Path attribute can contain signatures, in parallel, for two algorithm suites.) Once the transition is complete, use of the old 'current' algorithm will be deprecated, use of the 'new' algorithm will be mandatory, and a subsequent 'even newer' algorithm suite may be specified as recommended to implement. Once the transition has successfully been completed in this manner, BGPsec speakers SHOULD include only a single Signature_Block (corresponding to the 'new' algorithm).

6.2. Considerations for the SKI Size

Depending on the method of generating key identifiers [RFC7093], the size of the SKI in a RPKI router certificate may vary. The SKI field in the BGPsec_Path attribute has a fixed 20 octets size (see Figure 7). If the SKI is longer than 20 octets, then use the leftmost 20 octets of the SKI (excluding the tag and length) [RFC7093]. If the SKI value is shorter than 20 octets, then pad the SKI (excluding the tag and length) to the right (least significant octets) with octets having zero values.

6.3. Extensibility Considerations

This section discusses potential changes to BGPsec that would require substantial changes to the processing of the BGPsec_Path and thus necessitate a new version of BGPsec. Examples of such changes include:

- A new type of signature algorithm that produces signatures of variable length
- A new type of signature algorithm for which the number of signatures in the Signature_Block is not equal to the number of ASes in the Secure_Path (e.g., aggregate signatures)
- Changes to the data that is protected by the BGPsec signatures (e.g., attributes other than the AS path)

In the case that such a change to BGPsec were deemed desirable, it is expected that a subsequent version of BGPsec would be created and

that this version of BGPsec would specify a new BGP path attribute, let's call it BGPsec_Path_Two, which is designed to accommodate the desired changes to BGPsec. In such a case, the mandatory algorithm suites document would be updated to specify algorithm suites appropriate for the new version of BGPsec.

At this point a transition would begin which is analogous to the algorithm transition discussed in Section 6.1. During the transition period all BGPsec speakers SHOULD simultaneously include both the BGPsec_Path attribute and the new BGPsec_Path_Two attribute. Once the transition is complete, the use of BGPsec_Path could then be deprecated, at which point BGPsec speakers should include only the new BGPsec_Path_Two attribute. Such a process could facilitate a transition to a new BGPsec semantics in a backwards compatible fashion.

7. Operations and Management Considerations

Some operations and management issues that are closely relevant to BGPsec protocol specification and its deployment are highlighted here. The Best Current Practices concerning operations and deployment of BGPsec are provided in [I-D.ietf-sidr-bgpsec-ops].

7.1. Capability Negotiation Failure

Section 2.2 describes the negotiation required to establish a BGPsec-capable peering session. Not only must the BGPsec capability be exchanged (and agreed on), but the BGP multiprotocol extension [RFC4760] for the same AFI and the four-byte AS capability [RFC6793] MUST also be exchanged. Failure to properly negotiate a BGPsec session, due to a missing capability, for example, may still result in the exchange of BGP (unsigned) updates. It is RECOMMENDED that an implementation log the failure to properly negotiate a BGPsec session. Also, an implementation MUST have the ability to prevent a BGP session from being established if configured for only BGPsec use.

7.2. Preventing Misuse of pCount=0

A peer that is an Internet Exchange Point (IXP) (i.e. Route Server) with a transparent AS is expected to set pCount=0 in its Secure_Path Segment while forwarding an update to a peer (see Section 4.2). Clearly, such an IXP MUST configure its BGPsec router to set pCount=0 in its Secure_Path Segment. This also means that a BGPsec speaker MUST be configured so that it permits pCount=0 from an IXP peer. Two other cases where pCount is set to zero are in the context AS confederation (see Section 4.3) and AS migration [I-D.ietf-sidr-as-migration]. In these two cases, pCount=0 is set and accepted within the same AS (albeit the AS has two different

identities). Note that if a BGPsec speaker does not expect a peer AS to set its pCount=0, and if an update received from that peer violates this, then the update MUST be considered to be in error (see the list of checks in Section 5.2). See Section 8.4 for a discussion of security considerations concerning pCount=0.

7.3. Early Termination of Signature Verification

During the validation of a BGPsec update, route processor performance speedup can be achieved by incorporating the following observations. An update is deemed 'Valid' if at least one of the Signature_Blocks is marked as 'Valid' (see Section 5.2). Therefore, if an update contains two Signature_Blocks and the first one verified is found 'Valid', then the second Signature_Block does not have to be verified. And if the update is chosen for best path, then the BGPsec speaker adds its signature (generated with the respective algorithm) to each of the two Signature_Blocks and forwards the update. Also, a BGPsec update is deemed 'Not Valid' if at least one signature in each of the Signature_Blocks is invalid. This principle can also be used for route processor workload savings, i.e. the verification for a Signature_Block terminates early when the first invalid signature is encountered.

7.4. Non-Deterministic Signature Algorithms

Many signature algorithms are non-deterministic. That is, many signature algorithms will produce different signatures each time they are run (even when they are signing the same data with the same key). Therefore, if a BGPsec router receives a BGPsec update from a peer and later receives a second BGPsec update message from the same peer for the same prefix with the same Secure_Path and SKIs, the second update MAY differ from the first update in the signature fields (for a non-deterministic signature algorithm). However, the two sets of signature fields will not differ if the sender caches and reuses the previous signature. For a deterministic signature algorithm, the signature fields MUST be identical between the two updates. On the basis of these observations, an implementation MAY incorporate optimizations in update validation processing.

7.5. Private AS Numbers

It is possible that a stub customer of an ISP employs a private AS number. Such a stub customer cannot publish a ROA in the global RPKI for the private AS number and the prefixes that they use. Also, the global RPKI cannot support private AS numbers (i.e. BGPsec speakers in private ASes cannot be issued router certificates in the global RPKI). For interactions between the stub customer (with private AS number) and the ISP, the following two scenarios are possible:

1. The stub customer sends an unsigned BGP update for a prefix to the ISP's AS. An edge BGPsec speaker in the ISP's AS may choose to propagate the prefix to its non-BGPsec and BGPsec peers. If so, the ISP's edge BGPsec speaker MUST strip the AS_PATH with the private AS number, and then (a) re-originate the prefix without any signatures towards its non-BGPsec peer and (b) re-originate the prefix including its own signature towards its BGPsec peer. In both cases (i.e. (a) and (b)), the prefix MUST have a ROA in the global RPKI authorizing the ISP's AS to originate it.

2. The ISP and the stub customer may use a local RPKI repository (using a mechanism such as described in [I-D.ietf-sidr-slurm]). Then there can be a ROA for the prefix originated by the stub AS, and the eBGP speaker in the stub AS can be a BGPsec speaker having a router certificate, albeit the ROA and router certificate are valid only locally. With this arrangement, the stub AS sends a signed update for the prefix to the ISP's AS. An edge BGPsec speaker in the ISP's AS validates the update using RPKI data based the local RPKI view. Further, it may choose to propagate the prefix to its non-BGPsec and BGPsec peers. If so, the ISP's edge BGPsec speaker MUST strip the Secure_Path and the Signature Segment received from the stub AS with the private AS number, and then (a) re-originate the prefix without any signatures towards its non-BGPsec peer and (b) re-originate the prefix including its own signature towards its BGPsec peer. In both cases (i.e. (a) and (b)), the prefix MUST have a ROA in the global RPKI authorizing the ISP's AS to originate it.

It is possible that private AS numbers are used in an AS confederation [RFC5065]. BGPsec protocol requires that when a BGPsec update propagates through a confederation, each Member-AS that forwards it to a peer Member-AS MUST sign the update (see Section 4.3). However, the global RPKI cannot support private AS numbers. In order for the BGPsec speakers in Member-ASes with private AS numbers to have digital certificates, there MUST be a mechanism in place in the confederation that allows establishment of a local, customized view of the RPKI, augmenting the global RPKI repository data as needed. Since this mechanism (for augmenting and maintaining a local image of RPKI data) operates locally within an AS or AS confederation, it need not be standard based. However, a standard-based mechanism can be used (see [I-D.ietf-sidr-slurm]). Recall that in order to prevent exposure of the internals of AS confederations, a BGPsec speaker exporting to a non-member removes all intra-confederation Secure_Path Segments and Signatures (see Section 4.3).

7.6. Robustness Considerations for Accessing RPKI Data

The deployment structure, technologies and best practices concerning global RPKI data to reach routers (via local RPKI caches) are described in [RFC6810] [I-D.ietf-sidr-rpki-rtr-rfc6810-bis] [I-D.ietf-sidr-publication] [RFC7115] [I-D.ietf-sidr-bgpsec-ops] [I-D.ietf-sidr-delta-protocol]. For example, serial-number based incremental update mechanisms are used for efficient transfer of just the data records that have changed since last update [RFC6810] [I-D.ietf-sidr-rpki-rtr-rfc6810-bis]. Update notification file is used by relying parties (RPs) to discover whether any changes exist between the state of the global RPKI repository and the RP's cache [I-D.ietf-sidr-delta-protocol]. The notification describes the location of the files containing the snapshot and incremental deltas which can be used by the RP to synchronize with the repository. Making use of these technologies and best practices results in enabling robustness, efficiency, and better security for the BGPsec routers and RPKI caches in terms of the flow of RPKI data from repositories to RPKI caches to routers. With these mechanisms, it is believed that an attacker wouldn't be able to meaningfully correlate RPKI data flows with BGPsec RP (or router) actions, thus avoiding attacks that may attempt to determine the set of ASes interacting with an RP via the interactions between the RP and RPKI servers.

7.7. Graceful Restart

During Graceful Restart (GR), restarting and receiving BGPsec speakers MUST follow the procedures specified in [RFC4724] for restarting and receiving BGP speakers, respectively. In particular, the behavior of retaining the forwarding state for the routes in the Loc-RIB [RFC4271] and marking them as stale as well as not differentiating between stale and other information during forwarding will be the same as specified in [RFC4724].

7.8. Robustness of Secret Random Number in ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) with curve P-256 is used for signing updates in BGPsec [I-D.ietf-sidr-bgpsec-algs]. For ECDSA, it is stated in Section 6.3 of [FIPS186-4] that a new secret random number "k" shall be generated prior to the generation of each digital signature. A high entropy random bit generator (RBG) must be used for generating "k", and any potential bias in the "k" generation algorithm must be mitigated (see methods described in [FIPS186-4] [SP800-90A]).

7.9. Incremental/Partial Deployment Considerations

How will migration from BGP to BGPsec look like? What are the benefits for the first adopters? Initially small groups of contiguous ASes would be doing BGPsec. There would be possibly one or more such groups in different geographic regions of the global Internet. Only the routes originated within each group and propagated within its borders would get the benefits of cryptographic AS path protection. As BGPsec adoption grows, each group grows in size and eventually they join together to form even larger BGPsec capable groups of contiguous ASes. The benefit for early adopters starts with AS path security within the contiguous-AS regions spanned by their respective groups. Over time they would see those contiguous-AS regions grow much larger.

During partial deployment, if an AS in the path doesn't support BGPsec, then BGP goes back to traditional mode, i.e. BGPsec updates are converted to unsigned updates before forwarding to that AS (see Section 4.4). At this point, the assurance that the update propagated via the sequence of ASes listed is lost. In other words, for the BGPsec routers residing in the ASes starting from the origin AS to the AS before the one not supporting BGPsec, the assurance can be still provided, but not beyond that (for the updates in consideration).

8. Security Considerations

For a discussion of the BGPsec threat model and related security considerations, please see RFC 7132 [RFC7132].

8.1. Security Guarantees

When used in conjunction with Origin Validation (see RFC 6483 [RFC6483] and RFC 6811 [RFC6811]), a BGPsec speaker who receives a valid BGPsec update message, containing a route advertisement for a given prefix, is provided with the following security guarantees:

- The origin AS number corresponds to an autonomous system that has been authorized, in the RPKI, by the IP address space holder to originate route advertisements for the given prefix.

- For each AS in the path, a BGPsec speaker authorized by the holder of the AS number intentionally chose (in accordance with local policy) to propagate the route advertisement to the subsequent AS in the path.

That is, the recipient of a valid BGPsec update message is assured that the update propagated via the sequence of ASes listed in the

Secure_Path portion of the BGPsec_Path attribute. (It should be noted that BGPsec does not offer any guarantee that the data packets would flow along the indicated path; it only guarantees that the BGP update conveying the path indeed propagated along the indicated path.) Furthermore, the recipient is assured that this path terminates in an autonomous system that has been authorized by the IP address space holder as a legitimate destination for traffic to the given prefix.

Note that although BGPsec provides a mechanism for an AS to validate that a received update message has certain security properties, the use of such a mechanism to influence route selection is completely a matter of local policy. Therefore, a BGPsec speaker can make no assumptions about the validity of a route received from an external (eBGP) BGPsec peer. That is, a compliant BGPsec peer may (depending on the local policy of the peer) send update messages that fail the validity test in Section 5. Thus, a BGPsec speaker MUST completely validate all BGPsec update messages received from external peers. (Validation of update messages received from internal peers is a matter of local policy, see Section 5.)

8.2. On the Removal of BGPsec Signatures

There may be cases where a BGPsec speaker deems 'Valid' (as per the validation algorithm in Section 5.2) a BGPsec update message that contains both a 'Valid' and a 'Not Valid' Signature_Block. That is, the update message contains two sets of signatures corresponding to two algorithm suites, and one set of signatures verifies correctly and the other set of signatures fails to verify. In this case, the protocol specifies that a BGPsec speaker choosing to propagate the route advertisement in such an update message MUST add its signature to each of the Signature_Blocks (see Section 4.2). Thus the BGPsec speaker creates a signature using both algorithm suites and creates a new update message that contains both the 'Valid' and the 'Not Valid' set of signatures (from its own vantage point).

To understand the reason for such a design decision, consider the case where the BGPsec speaker receives an update message with both a set of algorithm A signatures which are 'Valid' and a set of algorithm B signatures which are 'Not Valid'. In such a case it is possible (perhaps even likely, depending on the state of the algorithm transition) that some of the BGPsec speaker's peers (or other entities further 'downstream' in the BGP topology) do not support algorithm A. Therefore, if the BGPsec speaker were to remove the 'Not Valid' set of signatures corresponding to algorithm B, such entities would treat the message as though it were unsigned. By including the 'Not Valid' set of signatures when propagating a route advertisement, the BGPsec speaker ensures that 'downstream' entities

have as much information as possible to make an informed opinion about the validation status of a BGPsec update.

Note also that during a period of partial BGPsec deployment, a 'downstream' entity might reasonably treat unsigned messages differently from BGPsec updates that contain a single set of 'Not Valid' signatures. That is, by removing the set of 'Not Valid' signatures the BGPsec speaker might actually cause a downstream entity to 'upgrade' the status of a route advertisement from 'Not Valid' to unsigned. Finally, note that in the above scenario, the BGPsec speaker might have deemed algorithm A signatures 'Valid' only because of some issue with RPKI state local to its AS (for example, its AS might not yet have obtained a CRL indicating that a key used to verify an algorithm A signature belongs to a newly revoked certificate). In such a case, it is highly desirable for a downstream entity to treat the update as 'Not Valid' (due to the revocation) and not as 'unsigned' (which would happen if the 'Not Valid' Signature_Blocks were removed enroute).

A similar argument applies to the case where a BGPsec speaker (for some reason such as lack of viable alternatives) selects as its best path (to a given prefix) a route obtained via a 'Not Valid' BGPsec update message. In such a case, the BGPsec speaker should propagate a signed BGPsec update message, adding its signature to the 'Not Valid' signatures that already exist. Again, this is to ensure that 'downstream' entities are able to make an informed decision and not erroneously treat the route as unsigned. It should also be noted that due to possible differences in RPKI data observed at different vantage points in the network, a BGPsec update deemed 'Not Valid' at an upstream BGPsec speaker may be deemed 'Valid' by another BGP speaker downstream.

Indeed, when a BGPsec speaker signs an outgoing update message, it is not attesting to a belief that all signatures prior to its are valid. Instead it is merely asserting that:

- The BGPsec speaker received the given route advertisement with the indicated prefix, AFI, SAFI, and Secure_Path; and

- The BGPsec speaker chose to propagate an advertisement for this route to the peer (implicitly) indicated by the 'Target AS Number'.

8.3. Mitigation of Denial of Service Attacks

The BGPsec update validation procedure is a potential target for denial of service attacks against a BGPsec speaker. The mitigation

of denial of service attacks that are specific to the BGPsec protocol is considered here.

To mitigate the effectiveness of such denial of service attacks, BGPsec speakers should implement an update validation algorithm that performs expensive checks (e.g., signature verification) after performing less expensive checks (e.g., syntax checks). The validation algorithm specified in Section 5.2 was chosen so as to perform checks which are likely to be expensive after checks that are likely to be inexpensive. However, the relative cost of performing required validation steps may vary between implementations, and thus the algorithm specified in Section 5.2 may not provide the best denial of service protection for all implementations.

Additionally, sending update messages with very long AS paths (and hence a large number of signatures) is a potential mechanism to conduct denial of service attacks. For this reason, it is important that an implementation of the validation algorithm stops attempting to verify signatures as soon as an invalid signature is found. (This ensures that long sequences of invalid signatures cannot be used for denial of service attacks.) Furthermore, implementations can mitigate such attacks by only performing validation on update messages that, if valid, would be selected as the best path. That is, if an update message contains a route that would lose out in best path selection for other reasons (e.g., a very long AS path) then it is not necessary to determine the BGPsec-validity status of the route.

8.4. Additional Security Considerations

The mechanism of setting the pCount field to zero is included in this specification to enable route servers in the control path to participate in BGPsec without increasing the length of the AS path. Two other scenarios where pCount=0 is utilized are in the context AS confederation (see Section 4.3) and AS migration [I-D.ietf-sidr-as-migration]. In these two scenarios, pCount=0 is set and also accepted within the same AS (albeit the AS has two different identities). However, entities other than route servers, confederation ASes or migrating ASes could conceivably use this mechanism (set the pCount to zero) to attract traffic (by reducing the length of the AS path) illegitimately. This risk is largely mitigated if every BGPsec speaker follows the operational guidance in Section 7.2 for configuration for setting pCount=0 and/or accepting pCount=0 from a peer. However, note that a recipient of a BGPsec update message within which an upstream entity two or more hops away has set pCount to zero is unable to verify for themselves whether pCount was set to zero legitimately.

There is a possibility of passing a BGPsec update via tunneling between colluding ASes. For example, say, AS-X does not peer with AS-Y, but colludes with AS-Y, signs and sends a BGPsec update to AS-Y by tunneling. AS-Y can then further sign and propagate the BGPsec update to its peers. It is beyond the scope of the BGPsec protocol to detect this form of malicious behavior. BGPsec is designed to protect messages sent within BGP (i.e. within the control plane) - not when the control plane in bypassed.

A variant of the collusion by tunneling mentioned above can happen in the context of AS confederations. When a BGPsec router (outside of a confederation) is forwarding an update to a Member-AS in the confederation, it signs the update to the public AS number of the confederation and not to the member's AS number (see Section 4.3). The Member-AS can tunnel the signed update to another Member-AS as received (i.e. without adding a signature). The update can then be propagated using BGPsec to other confederation members or to BGPsec neighbors outside of the confederation. This kind of operation is possible, but no grave security or reachability compromise is feared for the following reasons: (1) The confederation members belong to one organization and strong internal trust is expected; and (2) Recall that the signatures that are internal to the confederation MUST be removed prior to forwarding the update to an outside BGPsec router (see Section 4.3).

BGPsec does not provide protection against attacks at the transport layer. As with any BGP session, an adversary on the path between a BGPsec speaker and its peer is able to perform attacks such as modifying valid BGPsec updates to cause them to fail validation, injecting (unsigned) BGP update messages without BGPsec_Path attributes, injecting BGPsec update messages with BGPsec_Path attributes that fail validation, or causing the peer to tear-down the BGP session. The use of BGPsec does nothing to increase the power of an on-path adversary -- in particular, even an on-path adversary cannot cause a BGPsec speaker to believe a BGPsec-invalid route is valid. However, as with any BGP session, BGPsec sessions SHOULD be protected by appropriate transport security mechanisms (see the Security Considerations section in [RFC4271]).

There is a possibility of replay attacks which are defined as follows. In the context of BGPsec, a replay attack occurs when a malicious BGPsec speaker in the AS path suppresses a prefix withdrawal (implicit or explicit). Further, a replay attack is said to occur also when a malicious BGPsec speaker replays a previously received BGPsec announcement for a prefix that has since been withdrawn. The mitigation strategy for replay attacks involves router certificate rollover; please see [I-D.ietf-sidrops-bgpsec-rollover] for details.

9. IANA Considerations

IANA is requested to register a new BGP capability from Section 2.1 in the BGP Capabilities Code registry's "IETF Review" range. The description for the new capability is "BGPsec Capability". The reference for the new capability is this document (i.e. the RFC that replaces draft-ietf-sidr-bgpsec-protocol).

IANA is also requested to register a new path attribute from Section 3 in the BGP Path Attributes registry. The code for this new attribute is "BGPsec_Path". The reference for the new attribute is this document (i.e. the RFC that replaces draft-ietf-sidr-bgpsec-protocol).

IANA is requested to define the "BGPsec Capability" registry in the Resource Public Key Infrastructure (RPKI) group. The registry is as shown in Figure 10 with values assigned from Section 2.1:

Bits	Field	Reference
0-3	Version Value = 0x0	[This RFC]
4	Direction (Both possible values 0 and 1 are fully specified by this RFC)	[This RFC]
5-7	Unassigned Value = 000 (in binary)	[This RFC]

Figure 10: IANA registry for BGPsec Capability.

The Direction bit (4th bit) has value either 0 or 1, and both values are fully specified by this document (i.e. the RFC that replaces draft-ietf-sidr-bgpsec-protocol). Future Version values and future values of the Unassigned bits are assigned using the "Standards Action" registration procedures defined in RFC 5226 [RFC5226].

IANA is requested to define the "BGPsec_Path Flags" registry in the RPKI group. The registry is as shown in Figure 11 with one value assigned from Section 3.1:

Flag	Description	Reference
0	Confed_Segment Bit value = 1 means Flag set (indicates Confed_Segment) Bit value = 0 is default	[This RFC]
1-7	Unassigned Value: All 7 bits set to zero	[This RFC]

Figure 11: IANA registry for BGPsec_Path Flags field.

Future values of the Unassigned bits are assigned using the "Standards Action" registration procedures defined in RFC 5226 [RFC5226].

10. Contributors

10.1. Authors

Rob Austein
Dragon Research Labs
sra@hactrn.net

Steven Bellovin
Columbia University
smb@cs.columbia.edu

Randy Bush
Internet Initiative Japan
randy@psg.com

Russ Housley
Vigil Security
housley@vigilsec.com

Matt Lepinski
New College of Florida
mlepinski@ncf.edu

Stephen Kent
BBN Technologies
kent@bbn.com

Warren Kumari

Google
warren@kumari.net

Doug Montgomery
USA National Institute of Standards and Technology
dougm@nist.gov

Kotikalapudi Sriram
USA National Institute of Standards and Technology
kotikalapudi.sriram@nist.gov

Samuel Weiler
W3C/MIT
weiler@csail.mit.edu

10.2. Acknowledgements

The authors would like to thank Michael Baer, Oliver Borchert, David Mandelberg, Mehmet Adalier, Sean Turner, John Scudder, Wes George, Jeff Haas, Keyur Patel, Alvaro Retana, Nevil Brownlee, Matthias Waehlisch, Sandy Murphy, Chris Morrow, Tim Polk, Russ Mundy, Wes Hardaker, Sharon Goldberg, Ed Kern, Doug Maughan, Pradosh Mohapatra, Mark Reynolds, Heather Schiller, Jason Schiller, Ruediger Volk, and David Ward for their review, comments, and suggestions during the course of this work. Thanks are also due to many IESG reviewers whose comments greatly helped improve the clarity, accuracy, and presentation in the document.

11. References

11.1. Normative References

[I-D.ietf-sidr-bgpsec-algs]
Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, & Signature Formats", draft-ietf-sidr-bgpsec-algs-18 (work in progress), April 2017.

[I-D.ietf-sidr-bgpsec-pki-profiles]
Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles-21 (work in progress), January 2017.

[IANA-AF] "Address Family Numbers", <http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, DOI 10.17487/RFC4724, January 2007, <http://www.rfc-editor.org/info/rfc4724>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <http://www.rfc-editor.org/info/rfc4760>.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <http://www.rfc-editor.org/info/rfc5065>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <http://www.rfc-editor.org/info/rfc5492>.

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <http://www.rfc-editor.org/info/rfc6482>.

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

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <http://www.rfc-editor.org/info/rfc6793>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <http://www.rfc-editor.org/info/rfc7606>.

11.2. Informative References

[Borchert]
Borchert, O. and M. Baer, "Modification request: draft-ietf-sidr-bgpsec-protocol-14", IETF SIDR WG Mailing List message , February 10, 2016, <https://mailarchive.ietf.org/arch/msg/sidr/8B_e4CNxQCUKeZ_AUzsdnn2f5Mu>.

[FIPS186-4]
"FIPS Standards Publication 186-4: Digital Signature Standard", July 2013, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[I-D.ietf-sidr-as-migration]
George, W. and S. Murphy, "BGPSec Considerations for AS Migration", draft-ietf-sidr-as-migration-06 (work in progress), December 2016.

[I-D.ietf-sidr-bgpsec-ops]
Bush, R., "BGPsec Operational Considerations", draft-ietf-sidr-bgpsec-ops-16 (work in progress), January 2017.

[I-D.ietf-sidr-delta-protocol]
Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "RPKI Repository Delta Protocol (RRDP)", draft-ietf-sidr-delta-protocol-08 (work in progress), March 2017.

[I-D.ietf-sidr-publication]
Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-12 (work in progress), March 2017.

[I-D.ietf-sidr-rpki-rtr-rfc6810-bis]
Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", draft-ietf-sidr-rpki-rtr-rfc6810-bis-09 (work in progress), February 2017.

[I-D.ietf-sidr-slurm]
Mandelberg, D., Ma, D., and T. Bruijnzeels, "Simplified Local internet nUmber Resource Management with the RPKI", draft-ietf-sidr-slurm-04 (work in progress), March 2017.

[I-D.ietf-sidrops-bgpsec-rollover]
Weis, B., Gagliano, R., and K. Patel, "BGPsec Router Certificate Rollover", draft-ietf-sidrops-bgpsec-rollover-00 (work in progress), March 2017.

[RFC6472] Kumari, W. and K. Sriram, "Recommendation for Not Using AS_SET and AS_CONFED_SET in BGP", BCP 172, RFC 6472, DOI 10.17487/RFC6472, December 2011, <http://www.rfc-editor.org/info/rfc6472>.

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

[RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, DOI 10.17487/RFC6483, February 2012, <http://www.rfc-editor.org/info/rfc6483>.

[RFC6810] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, DOI 10.17487/RFC6810, January 2013, <http://www.rfc-editor.org/info/rfc6810>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <http://www.rfc-editor.org/info/rfc6811>.

[RFC7093] Turner, S., Kent, S., and J. Manger, "Additional Methods for Generating Key Identifiers Values", RFC 7093, DOI 10.17487/RFC7093, December 2013, <http://www.rfc-editor.org/info/rfc7093>.

[RFC7115] Bush, R., "Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)", BCP 185, RFC 7115, DOI 10.17487/RFC7115, January 2014, <http://www.rfc-editor.org/info/rfc7115>.

[RFC7132] Kent, S. and A. Chi, "Threat Model for BGP Path Security", RFC 7132, DOI 10.17487/RFC7132, February 2014, <http://www.rfc-editor.org/info/rfc7132>.

[SP800-90A]
"NIST 800-90A: Deterministic Random Bit Generator Validation System", October 2015, <http://csrc.nist.gov/groups/STM/cavp/documents/drbg/DRBGVS.pdf>.

Authors' Addresses

Matthew Lepinski (editor)
NCF
5800 Bay Shore Road
Sarasota FL 34243
USA

Email: mlepinski@ncf.edu

Kotikalapudi Sriram (editor)
NIST
100 Bureau Drive
Gaithersburg MD 20899
USA

Email: kotikalapudi.sriram@nist.gov

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

R. Gagliano
K. Patel
B. Weis
Cisco Systems
March 6, 2015

BGPSEC Router Certificate Rollover
draft-ietf-sidr-bgpsec-rollover-03

Abstract

BGPSEC will need to address the impact from regular and emergency rollover processes for the BGPSEC End-Entity (EE) certificates that will be performed by Certificate Authorities (CAs) participating at the Resource Public Key Infrastructure (RPKI). Rollovers of BGPSEC EE certificates must be carefully managed in order to synchronize distribution of router public keys and the usage of those pubic keys by BGPSEC routers. This document provides general recommendations for that process, as well as describing reasons why the rollover of BGPSEC EE certificates might be necssary.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2015.

Table of Contents

1. Requirements notation

The 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].

2. Introduction

In BGPSEC, a key rollover (or re-keying) is the process of changing a router's key pair (or pairs), issuing the corresponding new End-Entity certificate and (if the old certificate is still valid) revoking the old certificate. This process will need to happen at regular intervals, normally due to local policies at each network. This document provides general recommendations for that process. Certificate Practice Statements (CPS) documents MAY reference these recommendations. This process is comceptually similar to the RPKI Key Rollover process defined in [RFC6489].

When a router receives or creates a new key pair (depending on the key provisioning mechanism to be selected), this key pair will be used to sign new BGPsec_Path attributes [I-D.ietf-sidr-bgpsec-protocol] that are originated or that transit through the BGP speaker. Additionally, the BGP speaker MUST refresh its outbound BGPsec Update messages to include a signature using the new key (replacing the replaced key). When the rollover process finishes, the old BGPSEC certificate (and its key) will not longer be valid and thus any BGPsec Update that includes a BGPsec_Path attribute with a signature performed by the old key will be invalid. Consequently, if the router does not refresh its outbound BGPsec Update messages, routing information may be lost after the rollover process is finished. It is therefore extremely important that the BGPSEC router key rollover be performed such that the probability of new router EE certificates have been distributed throughout the RPKI before the router begin signing BGPsec_Path attributes with a new private key.

It is also important for an AS to minimize the BGPSEC router key rollover interval (i.e., in between the time an AS distributes an EE certificate with a new public key and the time a BGPSEC router begins to use its new private key). This can be due to a need for a BGPSEC router to distribute BGPsec_Path attributes signed with a new private key in order to invalidate BGPsec_Path attributes signed with the old private key. In particular, if the AS suspects that a stale BGPsec_Path attribute is being distributed instead of the most recently signed attribute it can cause the stale BGPsec_Path attribute to be invalidated by completing a key rollover procedure. The BGPSEC rollover interval can be minimized when an automated certificate provisioning process such as Enrollment over Secure Transport (EST) [RFC7030]) is used.

The Security Requirements for BGP Path Validation [RFC7353] also describes the need for protection against a replay attack, necessitating controlling BGPSEC's window of exposure to replay attacks. The BGPsec rollver method in this document can be used to

achieve this goal.

In [I-D.ietf-sidr-rtr-keying], the "operator-driven" method is introduced and it enables that a key pair could be shared among different BGP Speakers. In this scenario, the roll-over of the correspondent BGPSEC certificate will impact all the BGP Speakers sharing the same private key.

3. Key rollover in BGPSEC

A BGPSEC EE certificate (as any X.509 certificate) will required a rollover process due to causes such as:

BGPSEC scheduled rollover: BGPSEC certificates have an expiration date (NotValidAfter) that requires a frequent rollover process. The validity period for these certificates is typically expressed at the CA's CPS document.

BGPSEC certificate fields changes: Information contained in a BGPSEC certificate (such as the ASN or the Subject) may need to be changed.

BGPSEC emergency rollover Some special circumstances (such as a compromised key) may require the replacement of a BGPSEC certificate.

BGPSEC signature anti-replay protection An AS may determine stale BGPsec_Path attributes continue to be propogated

In most of these cases (probably excepting when the key has been compromised), it is possible to generate a new certificate without changing the key pair. This practice simplifies the rollover process as the correspondent BGP speakers do not even need to be aware of the changes to its correspondent certificate. However, not replacing the certificate key for a long period of time increases the risk that the certificate key may be compromised.

3.1. A proposed process for BGPSEC key rollover

The BGPSEC key rollover process will be dependent on the key provisioning mechanisms that would be in place. The key provisioning mechanisms for BGPSEC are not yet fully documented (see [I-D.ietf-sidr-rtr-keying] as a work in progress document). We will assume that an automatic provisioning mechanism suchas EST will be in place. The use of EST will allow BGPSEC code to include automatic re-keying scripts with minimum development cost.

If we work under the assumption that an automatic mechanism will exist to rollover a BGPSEC certificate, a possible process could be as follows.

1. New Certificate Pre-Publication: The first step in the rollover mechanism is to pre-publish the new public key in a new certificate. In order to accomplish this goal, the new key pair and certificate will need to be generated and published at the appropriate RPKI repository publication point. The details of

this process will vary as they depend on whether the keys are assigned per-BGP speaker or shared, whether the keys are generated on each BGP speaker or in a central location and wether the RPKI repository is locally or externally hosted.

2. Staging Period: A staging period will be required from the time a new certificate is published in the RPKI global repository until the time it is fetched by RPKI caches around the globe. The exact minimum staging time is not clear and will require experimental results from RPKI operations. RPKI repository design documents mention a lower limit of 24 hours (NOTE: need reference only one I found is the ops document). If rollovers will be done frequently and we want to avoid the stage period, an administrator can always provision two certificate for every router. In this case when the rollover operation is needed, the relying parties around the globe would already have the new keys. A staging period may not be possible to implement during emergency key rollover, in which case routing information may be lost.

3. Twilight: At this moment, the BGP speaker that hold the private key that has been rolled-over will stop using the OLD key for signing and start using the NEW key. Also, the router will generate appropriate BGPsec_Path attributes just as in the typical operation of refreshing out-bound BGP polices. This operation may generate a great number of BGPsec_Path attributes (due to the need to refresh BGP outbound policies). In any given BGP SPEAKER, the Twilight moment may be different for every peer in order to distribute the system load (probably in the order of minutes to avoid reaching any expiration time).

4. Certificate Revocation: This is an optional step. As part of the rollover process, a CA MAY decide to revoke the OLD certificate by publishing its serial number on the CA's CRL. On the other side, the CA will just let the OLD certificate to expire and not revoke it. This chose will depend on the reasons that motivated the rollover process.

5. RPKI-Router Protocol Withdrawals: Either due to the revocation of the OLD certificate or to the expiration of the OLD certificate's validation, the RPKI relying parties around the globe will need to communicate to their RTR peers that the OLD certificate's public key is not longer valid (rtr withdrawal message). It is not documented yet what will be a router's reaction to a RTR withdrawal message but it should include the removal of any RIB entry that includes a BGPSEC attribute signed with that key and the generation of the correspondent BGP WITHDRAWALs (either implicit or explicit).

The proposed rollover mechanism will depend on the existence of an automatic provisioning process for BGPSEC certificates. It will require a staging mechanism based on the RPKI propagation time of around 24hours, and it will generate BGPsec_Path attributes for all prefixes in the router been re-keyed.

The first two steps (New Certificate Pre-Publication and Staging Period) could happen ahead of time from the rest of the process as each network operators could prepare itself to accelerate a future key roll-over.

When a new BGPSEC certificate is generated without changing its key, steps 3 (Twilight) and 5 (RPKI-Router Protocol Withdrawals) SHOULD NOT be executed.

4. BGPSEC key rollover as a measure against replays attacks in BGPSEC

There are two typical generic measures to mitigate replay attacks in any protocol: the addition of a timestamp or the addition of a serial number. However neither BGP nor BGPSEC provide either measure. This section discusses the use of BGPSEC Rollover as a measure to mitigate replay attacks.

4.1. BGPSEC Replay attack window requirement

In [RFC7353] Section 4.3, the need to limit the vulnerability to replay attacks is described. One important comment is that during a windows of exposure, a replay attack is effective only if there was a downstream topology change that makes the signed AS path not longer current. In other words, if there have been no topology changes, then no security threat comes from a replay of a BGPsec_Path attribute (the signed information is still valid).

The BGPSEC Ops document [I-D.ietf-sidr-bgpsec-ops] gives some ideas of requirements for the size of the BGPSEC windows of exposure to replay attacks. At that document, it is stated that for the vast majority of the prefixes, the requirement will be in the order of days or weeks. For a very small but critical fraction of the prefixes, the requirement may be in the order of hours.

4.2. BGPSEC key rollover as a mechanism to protect against replay attacks

Since the window requirement is in the order of days (as documented in [I-D.ietf-sidr-bgpsec-ops]) and the BGP speaker re-keying is the edge router of the origin AS, it is feasible for a BGPSEC Rollover to mitigate mitigate. In this case it is important to complete the full process (i.e. the OLD and NEW certificate do not share the same key). By re-keying an AS is letting the BGPSEC certificate validation time be a sort of "timestamp" against replay attacks. However, the use of frequent key rollovers comes with an additional administrative cost and risks if the process fails. As documented before, re-keying should be supported by automatic tools and for the great majority of the Internet it will be done with good lead time to correct any risk.

For a transit AS that also originates BGPsec_Path attributes for its own prefixes, the key rollover process may generate a large number of UPDATE messages (even the complete Default Free Zone or DFZ). For this reason, it is recommended that routers in this scenario been provisioned with two certificates: one to sign BGPsec_Path attributes in transit and a second one to sign an BGPsec_Path attribute for prefixes originated in its AS. Only the second certificate (for prefixes originated in its AS) should be rolled-over frequently as a

means of limiting replay attach windows. The transit BGPSEC certificate is expected to be longer living than the origin BGPSEC certificate.

Advantage of Re-keying as replay attack protection mechanism:

1. All expiration policies are maintained in RPKI

2. Most of the additional administrative cost is paid by the provider that wants to protect its infrastructure (RP load will increase as there is a need to validate more BGPSEC certificates)

3. Can be implemented in coordination with planned topology changes by either origin ASes or transit ASes (e.g., if an AS changes providers, it completes a BGP Rollover)

Disadvantage of Re-keying as replay attack protection mechanism:

1. More administrative load due to frequent rollover, although how frequent is still not clear. Some initial ideas in [I-D.ietf-sidr-bgpsec-ops]

2. Minimum window size bounded by RPKI propagation time to RPKI caches for new certificate and CRL (2x propagation time). If pre-provisioning done ahead of time the minimum windows size is reduced (to 1x propagation time for the CRL). However, more experimentation is needed when RPKI and RPs are more massively deployed.

3. Increases dynamics and size of RPKI repository.

4. More load on RPKI caches, but they are meant to do this work.

5. IANA Considerations

No IANA considerations

6. Security Considerations

Several possible reasons can cause routers participating in BGPSEC to replace rollover their signing keys and/or signatures containing their current signature verification key. Some reasons are due to the usual key management operations reasons (e.g.,key exposure, change of certificate attributes, due to policy). However BGPSEC routers also may need to change their signing keys and associated certificate as an anti-replay protection.

The BGPSEC Rollover method allows for an expedient rollover process when router certificates are distributed through the RPKI, but without causing routing failures due to a receiving router not being able to validate a BGPsec_Path attribute created by a router that is the subject of the rollover.

7. Acknowledgements

We would like to acknowledge Randy Bush, Sriram Kotikalapudi, Stephen Kent and Sandy Murphy.

8. References

8.1. Normative References

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

[RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, February 2012.

8.2. Informative References

[I-D.ietf-sidr-bgpsec-ops]
Bush, R., "BGPsec Operational Considerations", draft-ietf-sidr-bgpsec-ops-05 (work in progress), May 2012.

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol-11 (work in progress), January 2015.

[I-D.ietf-sidr-rtr-keying]
Patel, K. and R. Bush, "Router Keying for BGPsec", draft-ietf-sidr-rtr-keying-08 (work in progress), January 2015.

[RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

[RFC7353] Bellovin, S., Bush, R., and D. Ward, "Security Requirements for BGP Path Validation", RFC 7353, August 2014.

Authors' Addresses

Roque Gagliano
Cisco Systems
Avenue des Uttins 5
Rolle, VD 1180
Switzerland

Email: rogaglia@cisco.com

Keyur Patel
Cisco Systems
170 W. Tasman Driv
San Jose, CA 95134
CA

Email: keyupate@cisco.com

Brian Weis
Cisco Systems
170 W. Tasman Driv
San Jose, CA 95134
CA

Email: bew@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2017

R. Gagliano
B. Weis
Cisco Systems
K. Patel
Arrcus, Inc.

October 25, 2016

BGPsec Router Certificate Rollover
draft-ietf-sidr-bgpsec-rollover-06

Abstract

BGPsec will need to address the impact from regular and emergency rollover processes for the BGPsec End-Entity (EE) certificates that will be performed by Certificate Authorities (CAs) participating at the Resource Public Key Infrastructure (RPKI). Rollovers of BGPsec EE certificates must be carefully managed in order to synchronize distribution of router public keys and the usage of those pubic keys by BGPsec routers. This document provides general recommendations for that process, as well as describing reasons why the rollover of BGPsec EE certificates might be necessary.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 28, 2017.

Table of Contents

1. Requirements notation

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 [RFC2119].

2. Introduction

In BGPsec, a key rollover (or re-keying) is the process of changing a router's key pair (or pairs), issuing the corresponding new End-Entity certificate and (if the old certificate is still valid) revoking the old certificate. This process will need to happen at regular intervals, normally due to local policies at each network. This document provides general recommendations for that process. Certificate Practice Statements (CPS) documents MAY reference these recommendations. This memo only addresses changing of a router's key pair within the RPKI. Refer to [RFC6489] for a procedure to rollover RPKI Certificate Authority key pairs.

When a router receives or creates a new key pair (depending on the key provisioning mechanism to be selected), this key pair will be

used to sign new BGPsec_Path attributes [I-D.ietf-sidr-bgpsec-protocol] that are originated or that transit through the BGP speaker. Additionally, the BGP speaker MUST refresh its outbound BGPsec Update messages to include a signature using the new key (replacing the replaced key). When the rollover process finishes, the old BGPsec certificate (and its key) will not longer be valid and thus any BGPsec Update that includes a BGPsec_Path attribute with a signature performed by the old key will be invalid. Consequently, if the router does not refresh its outbound BGPsec Update messages, routing information may be treated as unauthenticated after the rollover process is finished. It is therefore extremely important that the BGPsec router key rollover be performed such that the probability of new router EE certificates have been distributed throughout the RPKI before the router begin signing BGPsec_Path attributes with a new private key.

It is also important for an AS to minimize the BGPsec router key rollover interval (i.e., in between the time an AS distributes an EE certificate with a new public key and the time a BGPsec router begins to use its new private key). This can be due to a need for a BGPsec router to distribute BGPsec_Path attributes signed with a new private key in order to invalidate BGPsec_Path attributes signed with the old private key. In particular, if the AS suspects that a stale BGPsec_Path attribute is being distributed instead of the most recently signed attribute it can cause the stale BGPsec_Path attribute to be invalidated by completing a key rollover procedure. The BGPsec rollover interval can be minimized when an automated certificate provisioning process such as Enrollment over Secure Transport (EST) [RFC7030]) is used.

The Security Requirements for BGP Path Validation [RFC7353] also describes the need for protecting against the replay of BGP UPDATE messages, such as controlling BGPsec's window of exposure to replay attacks. The BGPsec rollover method in this document can be used to achieve this goal.

In [I-D.ietf-sidr-rtr-keying], the "operator-driven" method is introduced, in which a key pair can be shared among different BGP Speakers. In this scenario, the roll-over of the correspondent BGPsec certificate will impact all the BGP Speakers sharing the same private key.

3. Key rollover in BGPsec

An BGPsec EE certificate SHOULD be replaced when the following events occur, and can be replaced for any other reason at the discretion of the AS responsible for the EE certificate.

BGPsec scheduled rollover: BGPsec certificates have an expiration date (NotValidAfter) that requires a frequent rollover process. The validity period for these certificates is typically expressed at the CA's CPS document.

BGPsec certificate fields changes: Information contained in a BGPsec certificate (such as the ASN or the Subject) may need to be changed.

BGPsec emergency rollover Some special circumstances (such as a compromised key) may require the replacement of a BGPsec certificate.

BGPsec signature anti-replay protection An AS may determine stale BGPsec_Path attributes signed by the AS are being propagated instead of the most recently signed BGPsec_Path attributes. Changing the BGPsec router signing key, distributing a new BGPsec EE certificate for the router,and revoking the old BGPsec EE certificate will invalidate the replayed BGPsec_Path attributes.

In some of these cases it is possible to generate a new certificate without changing the key pair. This practice simplifies the rollover process as the corresponding BGP speakers do not even need to be aware of the changes to its correspondent certificate. However, not replacing the certificate key for a long period of time increases the risk that the router private key may be compromised. Distributing the OLD public key in a new certificate is NOT RECOMMENDED when the rollover event is due to the key has been compromised or stale BGPsec_Path attribute signatures are being distributed.

3.1. A proposed process for BGPsec key rollover

The BGPsec key rollover process will be dependent on the key provisioning mechanisms that are adopted by an AS. The key provisioning mechanisms for BGPsec are not yet fully documented (see [I-D.ietf-sidr-rtr-keying] as a work in progress document). An automatic provisioning mechanism such as EST will allow BGPsec code to include automatic re-keying scripts with minimum development cost.

If we work under the assumption that an automatic mechanism will exist to rollover a BGPsec certificate, a RECOMMENDED process is as follows.

1. New Certificate Publication: The first step in the rollover mechanism is to publish the new public key in a new certificate. In order to accomplish this goal, the new key pair and certificate will need to be generated and published at the

appropriate RPKI repository publication point. The details of this process will vary as they depend on whether the keys are assigned per-BGP speaker or shared, whether the keys are generated on each BGP speaker or in a central location and whether the RPKI repository is locally or externally hosted.

2. Staging Period: A staging period will be required from the time a new certificate is published in the RPKI global repository until the time it is fetched by RPKI caches around the globe. The exact minimum staging time will be dictated by the conventional interval chosen between repository fetches. If rollovers will be done more frequently, an administrator can provision two certificates for every router concurrently with different valid start times. In this case when the rollover operation is needed, the relying parties around the globe would already have the new keys. A staging period may not be possible to implement during emergency key rollover, in which case routing information may be lost.

3. Twilight: At this moment, the BGP speaker that holds the private key that has been rolled-over will stop using the OLD key for signing and start using the NEW key. Also, the router will generate appropriate BGPsec_Path attributes just as in the typical operation of refreshing out-bound BGP polices. This operation may generate a great number of BGPsec_Path attributes (due to the need to refresh BGP outbound policies). In any given BGP SPEAKER, the Twilight moment may be different for every peer in order to distribute the system load (probably in the order of minutes to avoid reaching any expiration time).

4. Certificate Revocation: This is an optional step, but SHOULD be taken when the goal is to invalidate signatures used with the OLD key. Reasons to invalidate OLD signatures include when the AS has reason to believe that the router signing key has been compromised, and when the AS needs to invalidate BGPsec_Path attribute signatures used with this key. As part of the rollover process, a CA MAY decide to revoke the OLD certificate by publishing its serial number on the CA's CRL. On the other side, the CA will just let the OLD certificate to expire and not revoke it. This choice will depend on the reasons that motivated the rollover process.

5. RPKI-Router Protocol Withdrawals: At the expiration of the OLD certificate's validation, the RPKI relying parties around the globe will need to communicate to their router peers that the OLD certificate's public key is not longer valid (e.g., using the RPKI-Router Protocol described in [RFC6810]). It is not documented yet what will be a router's reaction to a message with

the withdrawal bit set to 1 in the RPKI-Router Protocol, but it should include the removal of any RIB entry that includes a BGPsec attribute signed with that key and the generation of the correspondent BGP WITHDRAWALs (either implicit or explicit).

The proposed rollover mechanism will depend on the existence of an automatic provisioning process for BGPsec certificates. It will require a staging mechanism based on the RPKI propagation time of around 24 hours, and it will generate BGPsec_Path attributes for all prefixes in the router been re-keyed.

The first two steps (New Certificate Publication and Staging Period) may happen in advance of the rest of the process. This will allow a network operator to accelerate its subsequent key roll-over.

When a new BGPsec certificate is generated without changing its key, steps 3 (Twilight) and 5 (RPKI-Router Protocol Withdrawals) SHOULD NOT be executed.

4. BGPsec key rollover as a measure against replays attacks in BGPsec

There are two typical generic measures to mitigate replay attacks in any protocol: the addition of a timestamp or the addition of a serial number. However neither BGP nor BGPsec provide either measure. This section discusses the use of BGPsec Rollover as a measure to mitigate replay attacks.

4.1. BGPsec Replay attack window requirement

In [RFC7353] Section 4.3, the need to limit the vulnerability to replay attacks is described. One important comment is that during a window of exposure, a replay attack is effective only in very specific circumstances: there is a downstream topology change that makes the signed AS path no longer current, and the topology change makes the replayed route preferable to the route associated with the new update. In particular, if there have been no topology change at all, then no security threat comes from a replay of a BGPsec_Path attribute because the signed information is still valid.

The BGPsec Ops document [I-D.ietf-sidr-bgpsec-ops] gives some ideas of requirements for the size of the BGPsec windows of exposure to replay attacks. At that document, it is stated that for the vast majority of the prefixes, the requirement will be in the order of days or weeks.

4.2. BGPsec key rollover as a mechanism to protect against replay attacks

Since the window requirement is in the order of a day (as documented in [I-D.ietf-sidr-bgpsec-ops]) and the BGP speaker re-keying is the edge router of the origin AS, it is feasible for a BGPsec Rollover to mitigate replays. In this case it is important to complete the full process (i.e. the OLD and NEW certificate do not share the same key). By re-keying an AS is letting the BGPsec certificate validation time be a sort of "timestamp" against replay attacks. However, the use of frequent key rollovers comes with an additional administrative cost and risks if the process fails. As documented before, re-keying should be supported by automatic tools and for the great majority of the Internet it will be done with good lead time to correct any risk.

For a transit AS that also originates BGPsec_Path attributes for its own prefixes, the key rollover process may generate a large number of UPDATE messages (even the complete Default Free Zone or DFZ). For this reason, it is recommended that routers in this scenario been provisioned with two certificates: one to sign BGPsec_Path attributes in transit and a second one to sign an BGPsec_Path attribute for prefixes originated in its AS. Only the second certificate (for prefixes originated in its AS) should be rolled-over frequently as a means of limiting replay attach windows. The transit BGPsec certificate is expected to be longer living than the origin BGPsec certificate.

Advantage of Re-keying as replay attack protection mechanism:

1. All expiration policies are maintained in RPKI

2. Much of the additional administrative cost is paid by the provider that wants to protect its infrastructure, as it bears the human cost of creating and initiating distribution of new router key pairs and router EE certificates. (It is true that the cost of relying parties will be affected by the new objects, but their responses should be completely automated or otherwise routine.)

3. Can be implemented in coordination with planned topology changes by either origin ASes or transit ASes (e.g., if an AS changes providers, it completes a BGP Rollover)

Disadvantage of Re-keying as replay attack protection mechanism:

1. More administrative load due to frequent rollover, although how frequent is still not clear. Some initial ideas in [I-D.ietf-sidr-bgpsec-ops]

2. Minimum window size bounded by RPKI propagation time to RPKI caches for new certificate and CRL (2x propagation time). If provisioning is done ahead of time the minimum window size is reduced (to 1x propagation time for the CRL). However, more experimentation is needed when RPKI and RPs are more massively deployed.

3. Increases dynamics and size of RPKI repository.

5. IANA Considerations

There are no IANA considerations. This section may be removed upon publication.

6. Security Considerations

Several possible reasons can cause routers participating in BGPsec to replace rollover their signing keys and/or signatures containing their current signature verification key. Some reasons are due to the usual key management operations reasons (e.g.,key exposure, change of certificate attributes, due to policy). However BGPsec routers also may need to change their signing keys and associated certificate as an anti-replay protection.

The BGPsec Rollover method allows for an expedient rollover process when router certificates are distributed through the RPKI, but without causing routing failures due to a receiving router not being able to validate a BGPsec_Path attribute created by a router that is the subject of the rollover.

7. Acknowledgements

We would like to acknowledge Randy Bush, Sriram Kotikalapudi, Stephen Kent and Sandy Murphy.

8. References

8.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[I-D.ietf-sidr-bgpsec-ops]
Bush, R., "BGPsec Operational Considerations", draft-ietf-sidr-bgpsec-ops-10 (work in progress), June 2016.

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M. and K. Sriram, "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol-18 (work in progress), August 2016.

[I-D.ietf-sidr-rtr-keying]
Bush, R., Turner, S., and K. Patel, "Router Keying for BGPsec", draft-ietf-sidr-rtr-keying-12 (work in progress), June 2016.

[RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, DOI 10.17487/RFC6489, February 2012, <http://www.rfc-editor.org/info/rfc6489>.

[RFC6810] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, DOI 10.17487/RFC6810, January 2013, <http://www.rfc-editor.org/info/rfc6810>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[RFC7353] Bellovin, S., Bush, R., and D. Ward, "Security Requirements for BGP Path Validation", RFC 7353, DOI 10.17487/RFC7353, August 2014, <http://www.rfc-editor.org/info/rfc7353>.

Authors' Addresses

Roque Gagliano
Cisco Systems
Avenue des Uttins 5
Rolle, VD 1180
Switzerland

Email: rogaglia@cisco.com

Brian Weis
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
CA

Email: bew@cisco.com

Keyur Patel
Arrcus, Inc.

Email: keyur@arrcus.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2015

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
D. Mandelberg
BBN Technologies
February 16, 2015

RPKI Repository Delta Protocol
draft-ietf-sidr-delta-protocol-00

Abstract

In the Resource Public Key Infrastructure (RPKI), certificate authorities publish certificates, including end entity certificates, and CRLs to repositories on publication servers. Relying Parties (RP) retrieve the published information from the repository and MAY store it in a cache. This document specifies a delta protocol which provides relying parties with a mechanism to query a repository for changes, thus enabling the RP to keep its state in sync with the repository.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 18, 2015.

Table of Contents

1. Requirements notation

The 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].

2. Introduction

In the Resource Public Key Infrastructure (RPKI), certification authorities (CAs) publish certificates [RFC6487], RPKI signed objects [RFC6488] , manifests [RFC6486] and CRLs to repositories. CAs may have an embedded mechanism to publish to these repositories, or they may use a separate publication server and communication protocol. RPKI repositories are currently accessible using rsync, allowing Relying Parties (RPs) to synchronise a local copy of the RPKI repository used for validation with the central repositories using the rsync protocol [RFC6481].

This document specifies an alternative repository access protocol based on notification, snapshot and delta files that an RP can retrieve over http(s). This allows RPs to perform a full (re-)synchronisation of their local copy of the repository using snapshot files. However, typically RPs will use delta files to keep their local repository updated after initial synchronisation.

This protocol is designed to be consistent with the publication protocol [I-D.ietf-sidr-publication] and treats publication events of one or more repository objects as immutable events that can be communicated to relying parties. This approach helps to minimize the amount of data that traverses the network and thus helps minimize the amount of time until repository convergence occurs. This protocol also provides a standards based way to obtain consistent, point in time views of a single repository eliminating a number of consistency related issues. Finally, this approach allows for caching infrastructure to be used to serve this immutable data, and thus helps to reduce the load on a publication server when a large a number of relying parties are querying it.

3. RPKI Repository Delta Protocol Implementation

3.1. Informal Overview

Certification Authorities (CA) in the RPKI use a publication server to publish their RPKI products, such as manifests, CRLs, signed certificates and RPKI signed objects. This publication server may be remote, or embedded in the CA engine itself. Certificates in the RPKI that use a publication server that supports this delta protocol include a special Subject Information Access (SIA) pointer referring to a notification file.

The notification file includes a globally unique session_id in the form of a version 4 UUID, and serial number that can be used by the Relying Party (RP) to determine if it and the repository are synchronised. Furthermore it includes a link to the most recent complete snapshot of current objects that are published by the publication servers, and a list of links to delta files, for each revision starting at a point determined by the publication server, up to the current revision of the repository.

This notification file is intended to be small so that is can easily be fetched over HTTP(S). The publication server may use HTTP caching infrastructure to reduce its load. The publication server should should avoid using a long caching interval, since the length of this interval determines when RPs will receive updated notification files, and thereby new products produced by Certification Authorities using this publication server. It is recommended that of no longer than five minutes is used for caching this file. If the caching infrastructure supports it another useful approach would be to expire the cache for the notification file URI as soon as a new notification file is known to be published.

An RP that first learns about a notification file location can download it, and then proceed to download the latest snapshot file, and thus create a local copy of the repository that is in sync with the publication server. The RP should remember the location of this notification file, the session_id and current serial number.

RPs are encouraged to re-fetch this notification file at regular intervals, but should not try to fetch the same file more frequently than once per minute. After re-fetching the notification file, the RP may find that there are one or more delta files available that allow it to synchronise with the current state.

If no contiguous chain of updates is available, or if the session_id has changed, the latest snapshot should be used instead. In this case the RP should then add the objects found in the latest snapshot to its local repository.

As soon as the RP fetches new content in this way it should start a validation process using its local repository. An example of a reason why an RP may not do this immediately is because it has learned of more than one notification location and it prefers to complete all its updates before validating.

The publication server may use http caching infrastructure to reduce its load. It should be noted that snapshots and deltas for any given session_id and serial number contain an immutable record of the state of the publication server at a certain point in time. For this reason these files can be cached indefinitely. To support this the publication server must use a globally unique URL for the location of

each of these snapshot and delta files. It is recommended that old versions of snapshot and delta files remain available for download for some time after they have last appeared on a notification file to provide some resiliency in case relying parties are slow to process.

3.2. Update Notification File

3.2.1. Purpose

The update notification file is used by RPs to discover whether any changes exist between the state of the publication server's repository and the RP's cache. It describes the location of the files containing the snapshot and incremental deltas which can be used by the RP to synchronize with the repository.

3.2.2. Cache Concerns

A repository server MAY use caching infrastructure to cache the notification file and reduce the load of http(s) requests to a central repository server. However, since this file is used by RPs to determine whether any updates are available it is strongly RECOMMENDED to use a short interval for caching, to avoid unnecessary delays. A maximum of delay of 5 minutes after a new notification file has been published seems like a reasonable compromise. This delay should not cause major problems for RPs and routing since a similar human time scale is expected to be involved in updating the contents of the RPKI on the one hand, i.e. creating and publishing new ROAs or router certificates, and updating actual BGP announcements in routers on the other. That said, real world measurements are needed on this subject, so this recommended maximum time may be subject to change in future.

There are various ways to ensure that the notification file is only cached for a certain time in caching infrastructure and different solutions, such as commercial Content Delivery Networks (CDNs), may provide different ways of achieving this. For example some CDNs have custom support to cache a file such as this notification file indefinetely, but allow a central server to notify the CDN through some protocol that an update is available and trigger the CDN to then refresh this file. In general a publication server may find certain HTTP headers to be useful, such as: Cache-Control: max-age=300

Finally it shoulf be noted that snapshot and delta files are intended to be cache-able for a much longer longer time. In support of this the URIs for each snapshot and delta file for a given session_id and serial number MUST be unique and the contents of those files MUST NOT change.

3.2.3. File Format and Validation

Example notification file:

```
   <notification xmlns="HTTP://www.ripe.net/rpki/rrdp" version="1" session_id="9
df4b597-af9e-4dca-bdda-719cce2c4e28" serial="2">
     <snapshot uri="HTTP://rpki.ripe.net/rpki-ca/rrdp/EEEA7F7AD96D85BBD1F7274FA7
DA0025984A2AF3D5A0538F77BEC732ECB1B068.xml" hash="EEEA7F7AD96D85BBD1F7274FA7DA00
25984A2AF3D5A0538F77BEC732ECB1B068"/>
     <delta serial="2" uri="HTTP://rpki.ripe.net/rpki-ca/rrdp/198BD94315E9372D7F
15688A5A61C7BA40D318210CDC799B6D3F9F24831CF21B.xml" hash="198BD94315E9372D7F1568
8A5A61C7BA40D318210CDC799B6D3F9F24831CF21B"/>
     <delta serial="1" uri="HTTP://rpki.ripe.net/rpki-ca/rrdp/8DE946FDA8C6A6E431
DFE3622E2A3E36B8F477B81FAFCC5E7552CC3350C609CC.xml" hash="8DE946FDA8C6A6E431DFE3
622E2A3E36B8F477B81FAFCC5E7552CC3350C609CC"/>
   </notification>
```

The following validation rules must be observed when creating or parsing notification files:

- A RP MUST NOT process any update notification file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.
- The XML namespace MUST be HTTP://www.ripe.net/rpki/rrdp
- The encoding MUST be us-ascii
- The version attribute in the notification root element MUST be 1
- The session_id attribute MUST be a random version 4 UUID unique to this session
- The serial attribute must be an unbounded, unsigned positive integer indicating the current version of the repository.
- The notification file MUST contain exactly one ’snapshot’ element for the current repository version.
- If delta elements are included they MUST form a contiguous sequence starting at a revision determined by the publication server, up to the current version of the repository.
- The hash attribute in snapshot and delta elements must be the hexadecimal encoding of the SHA-256 hash of the referenced file. The RP SHOULD verify this hash when the file is retrieved and reject it if it does not match.

3.2.4. Publication Server Initialisation

When the publication server (re-) initialises it MUST generate a new random version 4 UUID to be used as the session_id. Furthermore it MUST then generate a snapshot file for serial number ONE for this new session that includes all currently known published objects that the publication server is responsible for. This snapshot file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely. The format and caching concerns for snapshot files are explained in more detail below in Section 3.3. After the snapshot file has been published the publication server MUST publish a new notification file that contains the new session_id, has serial number ONE, has one reference to the snaphot file that was just published, and that contains no delta references.

3.2.5. Publishing Updates

Whenever the publication server receives updates from a CA it SHOULD generate an update as follows.

The new repository serial MUST be one greater than the current repository serial. A new delta file MUST be generated for this new serial, that contains all the updates, i.e. new, replaced and withdrawn objects, as a single change set. This delta file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely. The format and caching concerns for delta files are explained in more detail below in Section 3.4.

The publication server MUST also generate a new snaphost file for this new serial, that contains all current objects for this new serial. In other words it should include all publish elements found in this update, and it should exclude all previous publish elements for objects that have been withdrawn or updated. As above this new file MUST be made available at a URL that is unique to this session_id and new version before proceeding.

Finally an updated notification file MUST be created by the publication server. This new notification file MUST include a reference to the new snaphot file. The file SHOULD also include available delta files for this and previous updates. However, the server MUST not include more delta files than, when combined, exceed the size of the current snapshot.

The publication server MAY also choose to include fewer delta files if it is found that the efficiency gain in keeping notification files small outweighs the overhead of forcing a small number of relying parties to process full snapshot files. At the time of this writing it is not completely clear what would constitute reasonable parameters to determine this balance. Real world measurements are needed to help this discussion. Possible approaches are:

- The publication server may learn the retrieval distribution of old delta files by RPs over time, and decide to exclude deltas from the point where less than e.g. 0.1% of RPs would retrieve them.
- The server may decide to support deltas only for a limited time, e.g. 6 hours. So that RPs can recover easily from restart or reasonably short outage scenarios, and are only forced to do a full re-sync in case of prolonged outages.

If the publication server is not capable of performing the above for some reason, then it MUST perform a full re-initialisation, as explained above in Section 3.2.4.

3.3. Snapshot File

3.3.1. Purpose

A snapshot is intended to reflect the complete and current contents of the repository. There it MUST contain all objects from the repository current as of the time of the publication.

3.3.2. Cache Concerns

A repository server MAY use caching infrastructure to cache snapshot files and reduce the load of http(s) requests to a central repository server. To support this it is important that snaphot files for a specific session_id and serial have a unique URL. The files themselves reflect the content of the repository at a specific point in time, and for that reason they never change. Aside from space concerns this means that these files MAY therefore be cached indefinitely.

To support RPs that are slow to process old, possibly cached, notification files, the publication server SHOULD ensure that old snapshot files remain available for some time after have last appeared on a notification file. It is RECOMMENDED that these files are kept for at least two times as long as the notification file cache period, i.e. 10 minutes. However, space permitting, the publication server is welcome to keep these files available for longer.

3.3.3. File Format and Validation

Example snapshot file:

```
   <snapshot xmlns="HTTP://www.ripe.net/rpki/rrdp" version="1" session_id="9df4b
597-af9e-4dca-bdda-719cce2c4e28" serial="5932">
     <publish uri="rsync://bandito.ripe.net/repo/671570f06499fbd2d6ab76c4f22566f
e49d5de60.cer">
       MIIFNDCCBBygAwIBAgIBAjANBgkqhkiG9w0BAQsFADANMQswCQYDVQQDEwJUQTAeFw0xNDExM
TMw
       MzU4MjlaFw0xNTExMTMwMzU4MjlaMDMxMTAvBgNVBAMTKDY3MTU3MGYwNjQ5OWZiZDJkNmFiN
zZj
       NGYyMjU2NmZlNDlkNWRlNjAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDOlUYxD
Pwu
       hqVSG5VXcg96qTYt9aKOH8qV2lAU/jnY1rRl2W5Uoa8RrAIseou8ltLKonMcVulHyoyY+J9Gq
rzN
       45vRSgBaOuvLn6nTuoD0LQsD/m8c/wEmFjQllirxQykLGJLXn1eKdUs/OXGgrAUPzgvkciJds
g69
       6X44deHcbCU0ZQZSLxZBZEqjfgyoYgww9n/hK5Sfkb44LsBK1lESdBSRrTpFizrCxl22ptsH0
eW4
       ek80CV5YgCg4F4u9xlzS2DvB+1X3Nl1vvTZ6TJlpVjIVcvE+sKQ50ntUwWG1+lOJc+twRehhi
CAb
       yHhfaxID4B+7h5Rcpkh1Q1AUMG9JAgMBAAGjggJ3MIICczAdBgNVHQ4EFgQUZxVw8GSZ+9LWq
3bE
       8iVm/knV3mAwHwYDVR0jBBgwFoAUd4IboVLl+9bEbD6VrCsnqRClFNUwDwYDVR0TAQH/BAUwA
wEB
       /zAOBgNVHQ8BAf8EBAMCAQYwRQYIKwYBBQUHAQEEOTA3MDUGCCsGAQUFBzAChilodHRwOi8vY
mFu
       ZGl0by5yaXBlLm5ldC9ycGtpLWNhL3RhL3RhLmNlcjCCATAGCCsGAQUFBwELBIIBIjCCAR4wV
wYI
       KwYBBQUHMAWGS3JzeW5jOi8vYmFuZGl0by5yaXBlLm5ldC9yZXBvLzNhODdhNGIxLTZlMjItN
GE2
       My1hZDBmLTA2ZjgzYWQzY2ExNi9kZWZhdWx0LzCBgwYIKwYBBQUHMAqGd3JzeW5jOi8vYmFuZ
Gl0
       by5yaXBlLm5ldC9yZXBvLzNhODdhNGIxLTZlMjItNGE2My1hZDBmLTA2ZjgzYWQzY2ExNi9kZ
WZh
       dWx0LzY3MTU3MGYwNjQ5OWZiZDJkNmFiNzZjNGYyMjU2NmZlNDlkNWRlNjAubWZ0MD0GCCsGA
QUF
       BzANhjFodHRwOi8vYmFuZGl0by5yaXBlLm5ldC9ycGtpLWNhL25vdGlmeS9ub3RpZnkueG1sM
FsG
       A1UdHwRUMFIwUKBOoEyGSnJzeW5jOi8vYmFuZGl0by5yaXBlLm5ldC9yZXBvLzc3ODIxYmExN
TJl
       NWZiZDZjNDZjM2U5NWFjMmIyN2E5MTBhNTE0ZDUuY3JsMBgGA1UdIAEB/wQOMAwwCgYIKwYBB
QUH
       DgIwHgYIKwYBBQUHAQcBAf8EDzANMAsEAgABMAUDAwDAqDANBgkqhkiG9w0BAQsFAAOCAQEAk
Anl
       E+Fm1r3cmW8EEwhq4Wo37j7qC8ciU/E/zJqptROd8M8+2PDjCF8K7plf/SqYNUWjCk8zQv7Si
ala
       DP3JNI7oWkJ5K9zSU/qPGD8UbrfK5EF4g+++OAsxsOf/qeMVdZ6FlPIUV0wYj2s9w1zz/r16H
FV6
       QO785ajB50foqo/oQ74BSRbrlYkWrM8U45rdSiAMlyr0lHgv0OCqNK6AVR6y9Sp6bBUi7RotZ
5FN
       x0TgBRTA6xp4pjG5FimX1SanMaW1hgYqdc4X5aZ9gPiyqvBcOtFq91WnNTsm5Ox0cPNDCkMPL
AwW
       pHOiFA0PlD0vBPrvTR1hsgfKGd318Qzq+w==
     </publish>
     <publish uri="rsync://bandito.ripe.net/repo/77821ba152e5fbd6c46c3e95ac2b27a
910a514d5.mft">
       MIAGCSqGSIb3DQEHAqCAMIACAQMxDzANBglghkgBZQMEAgEFADCABgsqhkiG9w0BCRABGqCAJ
IAE
       gd0wgdoCAguWGA8yMDE0MTIwMzE4MDgzMloYDzIwMTQxMjA0MTgwODMyWgYJYIZIAWUDBAIBM
IGm
       MFEWLDY3MTU3MGYwNjQ5OWZiZDJkNmFiNzZjNGYyMjU2NmZlNDlkNWRlNjAuY2VyAyEAh0nT6
uSg
       nJQhGAnKgjxb9TDeGu9AEd8QK+GHXYop0U8wURYsNzc4MjFiYTE1MmU1ZmJkNmM0NmMzZTk1Y
WMy
       YjI3YTkxMGE1MTRkNS5jcmwDIQAZ658FmRCmFfxCpTfE8hZN00MnUEdohOiISZflCPbrUwAAA
```

```
AAA
        AKCAMIIEcjCCA1qgAwIBAgICC5gwDQYJKoZIhvcNAQELBQAwDTELMAkGA1UEAxMCVEEwHhcNM
TQx
        MjAzMTgwODMyWhcNMTQxMjEwMTgwODMyWjAzMTEwLwYDVQQDEyhlY2Y0NjhkMDY1MTMyNzFmN
Tkz
        MjZhNjQ2MGZmOTFhYTNiNGU2Njk4MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAk
iYz
        EpnsqHIPNEl/LvJmfZfOYzRlhv0Ewqg/RLi6XsE5dhWi0YAiFLbz0v/PfAjmJJFO6STsXkmc5
Cpp
        PoAl2+Ffx9Zujzy95hCNMqNgPSsqA92eAStLJALlvWrlYgTQEBV/hIjetDOEY/fL49gajyuKg
hOh
        +zgeEUCVhdIArEj/4j5E1vI7flwJjLP8SI36IWlKoz6cd88Gm8bLQRURAfe2lKW0quJk0RHOn
PZk
        babWuiiByoU24DCSy1+TBY4mEK6bi1R0iONqeYfaSUrxvWcDh8V6gNikiB+tfwRxrIOO1RTnK
nlI
        hIe2OC5mkP2gMY5ZUyynJZnS3Or+CY3IcQIDAQABo4IBtDCCAbAwHQYDVR0OBBYEFOz0aNBlE
ycf
        WTJqZGD/kao7TmaYMB8GA1UdIwQYMBaAFHeCG6FS5fvWxGw+lawrJ6kQpRTVMA4GA1UdDwEB/
wQE
        AwIHgDBFBggrBgEFBQcBAQQ5MDcwNQYIKwYBBQUHMAKGKWh0dHA6Ly9iYW5kaXRvLnJpcGUub
mV0
        L3Jwa2ktY2EvdGEvdGEuY2VyMGYGCCsGAQUFBwELBFowWDBWBggrBgEFBQcwC4ZKcnN5bmM6L
y9i
        YW5kaXRvLnJpcGUubmV0L3JlcG8vNzc4MjFiYTE1MmU1ZmJkNmM0NmMzZTk1YWMyYjI3YTkxM
GE1
        MTRkNS5tZnQwWwYDVR0fBFQwUjBQoE6gTIZKcnN5bmM6Ly9iYW5kaXRvLnJpcGUubmV0L3Jlc
G8v
        Nzc4MjFiYTE1MmU1ZmJkNmM0NmMzZTk1YWMyYjI3YTkxMGE1MTRkNS5jcmwwGAYDVR0gAQH/B
A4w
```

```
        DDAKBggrBgEFBQcOAjAhBggrBgEFBQcBBwEB/wQSMBAwBgQCAAEFADAGBAIAAgUAMBUGCCsGA
QUF
        BwEIAQH/BAYwBKACBQAwDQYJKoZIhvcNAQELBQADggEBAAjCpzNzjj7QGhmIG3Elt49cHUJe8
65w
        y2Uq3ZKW2aZgA5It29D07XlsHO8tM0EWVXTxsBbpdkiEnzQ4G8Zx/ZI09vLSJ8ZjzSh42QeMa
Nt6
        6zslilaw9rQcm/5jwxN18BRniwU/oavfRbn36AhfCmpegII/4DTZWjI63wucRrYHTHZm6Zajn
HKU
        DT1viomKZoZZDAUB4oQ7pN/Mw+t1K9F50VKz+9i3tnVhyt5wVaoEn/4sGRAL680A8Su0MKiyc
69t
        3DYqnvSgYtFNiBbHhNYooBpraylh5r7WngBxfm+VJYkSaPxU8T6sSz/Capt+1S2UWJGTcFaZl
251
        bm8nmXcAADGCAawwggGoAgEDgBTs9GjQZRMnH1kyamRg/5GqO05mmDANBglghkgBZQMEAgEFA
KBr
        MBoGCSqGSIb3DQEJAzENBgsqhkiG9w0BCRABGjAcBgkqhkiG9w0BCQUxDxcNMTQxMjAzMTgwO
DMy
        WjAvBgkqhkiG9w0BCQQxIgQgOUbuFjfSw4aMeIgLlDmT5xI7D05/mH6zVETECtMzWb0wDQYJK
oZI
        hvcNAQEBBQAEggEAbhfERg8rgzy0GAIPDKj5kNk+owpm7WnRDiUo+6Y30zfKKjFhh1L+N0Ei7
b6q
        r934eqEoac23wycF/A1e3+d4PoLzvFrm1n9rIia4BaD8GiUle6FEHd5njS7jOt5Kuej64yDFC
Htv
        ipt8tGFik4MpvEmP5EOhZ1cU/sErvlpdEsxQCaLsb6JUbIvoIHnWGXHE54QXkBvlucUSxypRo
qW3
        SnAX0vo0F1YNrSDe05So3pjjSmNHOuFFnxZMja+lIMMWTFylbKQJNpLIrb9a/uarfiL9BrGOD
WqE
        dzQh+k3QkTAUojq+YADL+ixO0eg2zpPm+eEU1F2+bGP2M5rbaUfqngAAAAAAAA==
      </publish>
      <publish uri="rsync://bandito.ripe.net/repo/77821ba152e5fbd6c46c3e95ac2b27a
910a514d5.crl">
        MIIBnzCBiAIBATANBgkqhkiG9w0BAQsFADANMQswCQYDVQQDEwJUQRcNMTQxMjAzMTgwODMyW
hcN
        MTQxMjA0MTgwODMyWjAVMBMCAguXFw0xNDEyMDMxODA4MzJaoDAwLjAfBgNVHSMEGDAWgBR3g
huh
        UuX71sRsPpWsKyepEKUU1TALBgNVHRQEBAICC5YwDQYJKoZIhvcNAQELBQADggEBAFQHr/2id
s7e
        hmfnX+PmyePSN2EM1fBMLwMud6dqyBF42iNa8N0H/jxMAkgm7SS98TUupZg1aIwqxLwGakFS6
VeD
        +zCnCGEeMUlXTpZaICDxMxJuJLBOVbqP2amPxWJ22g0+gTXM9KPAoWlNyAiMaNUP+nawjyfMz
Q4c
        WJjIi1kRHnIhiu9cZwEh9Ns/sC3adPJ8NV6LPpMkQDQvIynxV/fbTf/EwwwRfLy1szGLZSdml
4G0
        gHohkWaosr4R2A7sZOc/PZGtstqpBRTD8RwVJx0pseC6Zp/01WH/FjzNpXahFPgR1QXy3qBGE
HRh
        xA08g0+QiGSz+QX5PPQ2dBkTRIY=
      </publish>
   </snapshot>
```

The following validation rules must be observed when creating or parsing snapshot files:

- A RP MUST NOT process any snapshot file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.
- The XML namespace MUST be HTTP://www.ripe.net/rpki/rrdp.
- The encoding MUST be us-ascii.
- The version attribute in the notification root element MUST be 1
- The session_id attribute MUST match the expected session_id in the reference in the notification file.
- The serial attribute MUST match the expected serial in the reference in the notification file.
- The hexadecimal encoding of the SHA-256 hash of this snapshot file MUST match the hash attribute in the reference in the notification file.

3.4. Delta File

3.4.1. Purpose

An incremental delta file contains all changes for exactly one serial increment of the publication server. In other words a single delta will typically include all the new objects, updated objects and withdrawn objects that a Certification Authority sent to the publication server. In its simplest form the update could concern only a single object, but it is recommended that CAs send all changes for one of their key pairs: i.e. updated objects as well as a new manifest and CRL as one atomic update message.

3.4.2. Cache Concerns

A repository server MAY use caching infrastructure to cache delta files and reduce the load of http(s) requests to a central repository server. To support this it is important that delta files for a specific session_id and serial have a unique URL. The files themselves reflect the content of the repository at a specific point in time, and for that reason they never change. Aside from space concerns this means that these files MAY therefore be cached indefinitely.

To support RPs that are slow to process old, possibly cached, notification files, the publication server SHOULD ensure that old delta files remain available for some time after have last appeared on a notification file. It is RECOMMENDED that these files are kept for at least two times as long as the notification file cache period, i.e. 10 minutes. However, space permitting, the publication server is welcome to keep these files available for longer.

3.4.3. File Format and Validation

Example snapshot file:

```
   <delta xmlns="HTTP://www.ripe.net/rpki/rrdp" version="1" session_id="9df4b597
-af9e-4dca-bdda-719cce2c4e28" serial="5932">
     <publish uri="rsync://bandito.ripe.net/repo/3a87a4b1-6e22-4a63-ad0f-06f83ad
3ca16/default/671570f06499fbd2d6ab76c4f22566fe49d5de60.mft" hash="226AB8CD3C887A
6EBDDDF317F2FAFC9CF3EFC5D43A86347AC0FEFFE4DC0F607E">
       MIAGCSqGSIb3DQEHAqCAMIACAQMxDzANBglghkgBZQMEAgEFADCABgsqhkiG9w0BCRABGqCAJ
IAE
       gYkwgYYCAguWGA8yMDE0MTIwMzE4MDg0MFoYDzIwMTQxMjA0MTgwODQwWgYJYIZIAWUDBAIBM
FMw
       URYsNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyNTY2ZmU0OWQ1ZGU2MC5jcmwDIQD1h9mcw
KzN
       70He/gIMVxszGJlIXLh/TGzkaNTXxLixmwAAAAAAAKCAMIIFGjCCBAKgAwIBAgICC5YwDQYJK
oZI
       hvcNAQELBQAwMzExMC8GA1UEAxMoNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyNTY2ZmU0O
WQ1
       ZGU2MDAeFw0xNDEyMDMxODA4NDBaFw0xNDEyMTAxODA4NDBaMDMxMTAvBgNVBAMTKDMzMTI1Y
zA4
       NmEwOWEyOTJkYzQxMWI1MzkwODA2ZDA4NTMxM2E1MTQwggEiMA0GCSqGSIb3DQEBAQUAA4IBD
wAw
       ggEKAoIBAQCak1G7912fuezSWPpPqER3aZaP4HShGiIiRWeJLOYKpklAeSO8kRa9R+yDCa9CM
i1B
       lewW/C5Coomb9teVZRg31YkpZlXqqNGg8GVesNMJX3ryuizQ+WRUcgwJoakqWH7wPu5zdfFj4
Cpk
       BpgJF+6TBYwXTjAmxfFP0hm0QLWCLxEd1gBGEdBmOogHqfOZHU95GLjllzsPRmR13kyh7BbYM
ie+
       ENJAqqKBlQvW86xPEDMJKUc0uQDnTPCZQBqFwE1xrgUAuSCfJMUguAE8clsshOFe8ROF9t6NI
BxK
       oxA+PTYCpcthCBtdCFyWn/SLp1pb3gIA6xP9ESGlRHNumPL3AgMBAAGjggI2MIICMjAdBgNVH
Q4E
       FgQUMxJcCGoJopLcQRtTkIBtCFMTpRQwHwYDVR0jBBgwFoAUZxVw8GSZ+9LWq3bE8iVm/knV3
mAw
       DgYDVR0PAQH/BAQDAgeAMGcGCCsGAQUFBwEBBFswWTBXBggrBgEFBQcwAoZLcnN5bmM6Ly9iY
W5k
       aXRvLnJpcGUubmV0L3JlcG8vM2E4N2E0YjEtNmUyMi00YTYzLWFkMGYtMDZmODNhZDNjYTE2L
2Rl
       ZmF1bHQvMIGWBggrBgEFBQcBCwSBiTCBhjCBgwYIKwYBBQUHMAuGd3JzeW5jOi8vYmFuZGl0b
y5y
       aXBlLm5ldC9yZXBvLzNhODdhNGIxLTZlMjItNGE2My1hZDBmLTA2ZjgzYWQzY2ExNi9kZWZhd
Wx0
       LzY3MTU3MGYwNjQ5OWZiZDJkNmFiNzZjNGYyMjU2NmZlNDlkNWRlNjAubWZ0MIGJBgNVHR8Eg
YEw
       fzB9oHugeYZ3cnN5bmM6Ly9iYW5kaXRvLnJpcGUubmV0L3JlcG8vM2E4N2E0YjEtNmUyMi00Y
TYz
       LWFkMGYtMDZmODNhZDNjYTE2L2RlZmF1bHQvNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyN
TY2
       ZmU0OWQ1ZGU2MC5jcmwwGAYDVR0gAQH/BA4wDDAKBggrBgEFBQcOAjAhBggrBgEFBQcBBwEB/
wQS
       MBAwBgQCAAEFADAGBAIAAgUAMBUGCCsGAQUFBwEIAQH/BAYwBKACBQAwDQYJKoZIhvcNAQELB
QAD
       ggEBAEO1dSFDN4wZqtZ0fWo5G0YVN+mtk6tKhHPFwX7ydTofnHZkE2pO7C93XcgPcP4zLUBPt
5kS
       aH+0vcBxs9Vg//58cHRUEHhls9O/XcS8RXCVkNiga+9NB5s4oi0+i/gDU3eOUqE/jqSJAJAS+
Ehi
       tvNh0LuLrW92NrOfbYDk29how3uxK4JucIAQ05i63l7EAeQp3WeI8nVzB9Rfrkv+PSV+57mSX
XtJ
       /jWu3kyjvsxRjeUL3Im2Z1F48zfVF6pVaDT7ib4YbKOyAQTMpi4W6NZwgQskda9B8/0qV/d+2
JrC
       m3Ozm0t2laoH8xKP/OC33bBXLCxUvkVqvB/Y+TUXfAEAADGCAawwggGoAgEDgBQzElwIagmik
txB
       G1OQgG0IUxOlFDANBglghkgBZQMEAgEFAKBrMBoGCSqGSIb3DQEJAzENBgsqhkiG9w0BCRABG
jAc
       BgkqhkiG9w0BCQUxDxcNMTQxMjAzMTgwODQwWjAvBgkqhkiG9w0BCQQxIgQgdNPMbp9lJJHNM
mIz
```

```
       00ff73VkVFYWo2Uf6/b4zIzFZucwDQYJKoZIhvcNAQEBBQAEggEAXHNHm+DUD1s9IQMewvKso
NGi
       fXL2jG3yfuGys5x1aJji3bIKGiU+weHmnP9aoH9UFRLk6pW1wFOS0+6M87UD8cU17w9F10e02
58S
       9p7xHMgbrYqXrX9OucMqiN4M+ThDzyDXnfNAOgw5XNJu9KRndS9vyXS6lcvD7JTOhkyqKsrqH
XlM
       0pX+rYFtrF2RNjB54veooSkcKGojXReLttZbvVKWKwkVg2RJy4tt7MOGU0Q6qa/J5S7O6xvwP
jkY
       yCFvrHm+CgeXoR/3Hg/Rk/NdsK4K1u5dXhRh3KYv4P/hnGSD83aFE9t/DTicvl6SjaXFCtLtJ
lTX
       BqSW7wgZ6OoLxwAAAAAAAA==
     </publish>
     <publish uri="rsync://bandito.ripe.net/repo/3a87a4b1-6e22-4a63-ad0f-06f83ad
3ca16/default/671570f06499fbd2d6ab76c4f22566fe49d5de60.crl" hash="2B551A6C10CCA0
4C174B0CEB3B64652A5534D1385BEAA40A55A68CB06055E6BB">
       MIIBxTCBrgIBATANBgkqhkiG9w0BAQsFADAzMTEwLwYDVQQDEyg2NzE1NzBmMDY0OTlmYmQyZ
DZh
       Yjc2YzRmMjI1NjZmZTQ5ZDVkZTYwFw0xNDEyMDMxODA4NDBaFw0xNDEyMDQxODA4NDBaMBUwE
wIC
       C5UXDTE0MTIwMzE4MDg0MFqgMDAuMB8GA1UdIwQYMBaAFGcVcPBkmfvS1qt2xPIlZv5J1d5gM
AsG
       A1UdFAQEAgILljANBgkqhkiG9w0BAQsFAAOCAQEAIbL+8connmKLeypzs/P6FOHv8elmLp6dF
lId
       SDpZT7p6y9xLZkvuow39XOs6NB1AOA+92uao9hEV1XuEBGP98nsx0frL8HJtKcEn0q5LGqA4Y
eBG
       n28+Ldvlh4DetiKvFpsKW/VYqjRumHcgTdWpESY/f9hH3xW6JCggH5cFGFF/dCsCdGT1v+m53
zf4
       Dlz8KhRDEaok3UMycX9XUWMB5HSwf05Qrha2LIFf66uk6AQQEmV9ZiBq3IdbkdNd90TIVDMvn
SW/
       p9Xygdx8azaE2+hsOc9J7+E2kBuu4isLhvfZmChtFpxIUrljQRD4iUil8/xmB6MAIptoF1Esl
pAI
       aw==
     </publish>
     <withdraw uri="rsync://bandito.ripe.net/repo/3a87a4b1-6e22-4a63-ad0f-06f83a
d3ca16/default/example.roa" hash="2B551A6C10CCA04C174B0CEB3B64652A5534D1385BEAA4
0A55A68CB06055E6BB"/>
   </delta>
```

Note that a formal RELAX NG specification of this file format is included later in this document. A RP MUST NOT process any update notification file that is incomplete or not well formed.

The following validation rules must be observed when creating or parsing snapshot files:

- o A RP MUST NOT process any delta file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 of this document.
- o The XML namespace MUST be HTTP://www.ripe.net/rpki/rrdp.
- o The encoding MUST be us-ascii.
- o The version attribute in the notification root element MUST be 1
- o The session_id attribute MUST be a random version 4 UUID unique to this session
- o The session_id attribute MUST match the expected session_id in the reference in the notification file.
- o The serial attribute MUST match the expected serial in the reference in the notification file.
- o The hexadecimal encoding of the SHA-256 hash of this snapshot file MUST match the hash attribute in the reference in the notification file.
- o A publish element MUST include a hash attribute, if the object is intended to replace another object in the RPKI, and its value MUST be the hexadecimal encoding of the SHA-256 hash of the replaced object. If the published object does not replace another object the hash attribute MUST NOT be included. Note that this is an extension to the publication protocol that is not, yet, reflected in [I-D.ietf-sidr-publication].
- o Similarly a withdraw element MUST contain a hash attribute with the hexadecimal encoding of the SHA-256 hash of the withdrawn object. Including the hashes in this manner allows relying parties to identify specific objects by their hash rather than the URI where they are found.

3.5. SIA for CA certificates

Certificate Authorities that use this delta protocol MUST have an instance of an SIA AccessDescription in addition to the ones defined in [RFC6487],

```
AccessDescription ::= SEQUENCE {
  accessMethod OBJECT IDENTIFIER,
  accessLocation GeneralName }
```

This extension MUST use an accessMethod of id-ad-rpkiNotify, see: [IANA-AD-NUMBERS],

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
id-ad-rpkiNotify OBJECT IDENTIFIER ::= { id-ad 13 }
```

The accessLocation MUST be a URI [RFC3986], using the 'HTTP' or 'HTTPS' protocol, that will point to the update notification file for the publication server that publishes the products of this CA certificate.

Relying Parties that do not support this delta protocol MUST MUST NOT reject a CA certificate merely because it has an SIA extension containing this new kind of AccessDescription.

4. Relying Party Use

4.1. Full Synchronisation

When a Relying Party first encounters a notification file URI as an SIA of a certificate that it has validated it SHOULD retrieve the notification file and download the latest snapshot to get in sync with the current version of the publication server.

The RP SHOULD reject the snapshot file and raise an operator alert, if its hash does not match the hash listed in the notification file. However, if the RP does not have any prior state it may choose to process this snapshot file anyway. It should be noted that the RPKI objects are protected by object security, so problems or attacks on the publication server or transport can result in witholding or replaying old objects, but it cannot force the RP to accept invalid objects. Using the remaining or old objects for validation is probably better than rejecting everything, since the latter could be used as a denial of service vector on relying parties.

4.2. Processing Deltas

It is RECOMMENDED that the RP notes the URI, session_id and serial number when it first learns about a notification file. The RP MAY then poll the file to discover updates. How frequently the RP does this is largely up to local policy. The polling frequency determines in part what propogation time that a RP is willing to accept between the moment that a change is published in the RPKI, and the moment that those changes are processed and validated. As discussed in Section 3.2.2 there does not seem to be a need to have a propogation time that is below five minutes. Since the publication server infrastructure MAY cache the notification file for up to five minutes a slightly more frequent polling strategy may be useful, however the RP SHOULD NOT poll more frequently than once per minute. More frequent polling would only result in marginal gains in propagation, while causing unnecessary load on the caching infrastructure.

If the RP finds that the session_id has changed, or if it cannot find a contiguous chain of links to delta files from its current serial to publication server's current serial, then it MUST perform a full synchronisation instead of continuing to process deltas.

If the RP finds a contiguous chain of links to delta files from its current serial to the publication server's current serial, and the session_id has not changed, it should download all missing delta files. If any delta file cannot be downloaded, or its hash does not match the hash listed on the notification file, or if no such chain of deltas is availabe, or the session_id has changed, then the RP MUST perform a full synchronisation instead.

New objects found in delta files can be added to the RPs local copy of the repository. However, it is RECOMMENDED that the RP treats object updates and withdraws with some skepticism. A compromised publication server may not have access to the certification authorities' keys, but it can pretend valid objects have been withdrawn. Therefore it may be preferred to use a strategy where local copies of objects are only discarded when the RP is sure that they are no longer relevant, e.g. the CA has explicitly revoked them, removed the objects from a valid manifest that it issued, or they have expired.

5. XML Schema

The following is a RELAX NG compact form schema describing version 1 of this protocol.

```
#
# RelaxNG schema for RPKI Repository Delta Protocol (RRDP).
#

default namespace = "HTTP://www.ripe.net/rpki/rrdp"

version = xsd:positiveInteger   { maxInclusive="1" }
serial  = xsd:nonNegativeInteger
uri     = xsd:anyURI
uuid    = xsd:string            { pattern = "[\-0-9a-fA-F]+" }
hash    = xsd:string            { pattern = "[0-9a-fA-F]+" }
base64  = xsd:base64Binary

# Notification file: lists current snapshots and deltas

start |= element notification {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element snapshot {
    attribute uri  { uri },
    attribute hash { hash }
  },
  element delta {
    attribute serial { serial },
    attribute uri    { uri },
    attribute hash   { hash }
  }*
}

# Snapshot segment: think DNS AXFR.

start |= element snapshot {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element publish      {
    attribute uri { uri },
    base64
  }*
}

# Delta segment: think DNS IXFR.

start |= element delta {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  delta_element+
}

delta_element |= element publish  {
  attribute uri  { uri },
```

```
    attribute hash { hash }?,
    base64
  }

  delta_element |= element withdraw {
    attribute uri  { uri },
    attribute hash { hash }
  }

  # Local Variables:
  # indent-tabs-mode: nil
  # comment-start: "# "
  # comment-start-skip: "#[ \t]*"
  # End:
```

6. Security Considerations

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

TBD

9. References

[I-D.ietf-sidr-publication]
Weiler, S., Sonalker, A. and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", Internet-Draft draft-ietf-sidr-publication-05, February 2014.

[IANA-AD-NUMBERS]
"SMI Security for PKIX Access Descriptor", , <http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.48>.

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

[RFC3986] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC6481] Huston, G., Loomans, R. and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, February 2012.

[RFC6486] Austein, R., Huston, G., Kent, S. and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, February 2012.

[RFC6487] Huston, G., Michaelson, G. and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.

[RFC6488] Lepinski, M., Chi, A. and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, February 2012.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net

Bryan Weber
Cobenian

Email: bryan@cobenian.com

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net

David Mandelberg
BBN Technologies

Email: david@mandelberg.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs

March 13, 2017

RPKI Repository Delta Protocol (RRDP)
draft-ietf-sidr-delta-protocol-08

Abstract

In the Resource Public Key Infrastructure (RPKI), Certificate Authorities publish certificates, including end entity certificates, Certificate Revocation Lists (CRL), and RPKI signed objects to repositories. Relying Parties retrieve the published information from those repositories. This document specifies a new RPKI Repository Delta Protocol (RRDP) for this purpose. RRDP was specifically designed for scaling. It relies on a notification file which lists the current snapshot and delta files that can be retrieved using HTTP over TLS (HTTPS), and enables to use of CDNs or other caching infrastructure for the retrieval of these files.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Requirements notation

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

2. Introduction

In the Resource Public Key Infrastructure (RPKI), Certificate Authorities publish certificates [RFC6487], RPKI signed objects [RFC6488], manifests [RFC6486], and CRLs to repositories. CAs may have an embedded mechanism to publish to these repositories, or they may use a separate Repository Server and publication protocol. RPKI repositories are currently accessible using the [rsync] protocol, allowing Relying Parties to synchronise a local copy of the RPKI repository used for validation with the remote repositories [RFC6481].

[rsync] has proven valuable in the early deployment of RPKI, because it allowed operators to gain experience without the need to invent a custom protocol. However, operational experience has brought concerns to light that we wish to address here:

- [rsync] is designed to limit the amount of data that needs to be transferred between client and server. However the server needs to spend significant resources in terms of CPU and memory for every connection. This is a problem in an envisioned RPKI deployment where thousands of Relying Parties query a small number of central repositories, and it makes these repositories weak to denial of service attacks.

- A secondary concern is the lack of supported rsync server and client libraries. In practice all implementations have to make system calls to an rsync binary. This is inefficient, introduces fragility with regards to updates of this binary, makes it difficult to catch and report problems to operators, and it complicates software development and testing.

This document specifies an alternative repository access protocol based on notification, snapshot and delta files that a Relying Party can retrieve over the HTTPS protocol. This allows Relying Parties to perform either a full (re-)synchronisation of their local copy of the repository using snapshot files, or use delta files to keep their local repository updated after initial synchronisation. We call this the RPKI Repository Delta Protocol, or RRDP in short.

RRDP was designed to support scaling in RPKI’s asymmetric deployment. It is consistent (in terms of data structures) with the publication protocol [I-D.ietf-sidr-publication] and treats publication events of one or more repository objects as discrete events that can be communicated to Relying Parties. This approach helps to minimize the amount of data that traverses the network and thus helps minimize the amount of time until repository convergence occurs. RRDP also provides a standards based way to obtain consistent, point in time

views of a single repository, eliminating a number of consistency related issues. Finally, this approach allows these discrete events to be communicated as immutable files. This enables Repository Servers to pre-calculate these files only once for all clients - thus limiting the CPU and memory investments required, and enables the use of caching infrastructure to reduce the load on a repository server when a large number of Relying Parties are querying it.

This document allows the use of RRDP as an additional repository distribution mechanism for RPKI. In time RRDP may replace [rsync] as the only mandatory to implement repository distribution mechanism. However this transition is outside of the scope of this document.

3. RPKI Repository Delta Protocol Implementation

3.1. Informal Overview

Certification Authorities in the RPKI use a repository server to publish their RPKI products, such as manifests, CRLs, signed certificates and RPKI signed objects. This repository server may be remote, or embedded in the Certificate Authority engine itself. Certificates in the RPKI that use a repository server that supports RRDP include a special Subject Information Access (SIA) pointer referring to a notification file.

The notification file includes a globally unique session_id in the form of a version 4 UUID ([RFC4122]), and serial number that can be used by the Relying Party to determine if it and the repository are synchronised. Furthermore it includes a link to the most recent complete snapshot of current objects that are published by the repository server, and a list of links to delta files, for each revision starting at a point determined by the repository server, up to the current revision of the repository.

A Relying Party that learns about a notification file location for the first time can download it, and then proceed to download the latest snapshot file, and thus create a local copy of the repository that is in sync with the repository server. The Relying Party records the location of this notification file, the session_id and current serial number.

Relying Parties are encouraged to re-fetch this notification file at regular intervals, but not more often than once per minute. After re-fetching the notification file, the Relying Party may find that there are one or more delta files available that allow it to synchronise its local repository with the current state of the repository server. If no contiguous chain of deltas from the Relying Party’s serial to the latest repository serial is available, or if

the session_id has changed, the Relying Party performs a full resynchronisation instead.

As soon as the Relying Party fetches new content in this way it could start a validation process. An example of a reason why a Relying Party may not choose to do this immediately is because it has learned of more than one notification location and it prefers to complete all its updates before validating.

The repository server could use caching infrastructure to reduce its load, particularly because snapshots and deltas for any given session_id and serial number contain an immutable record of the state of the repository server at a certain point in time. For this reason these files can be cached indefinitely. Notification files are polled by Relying Parties to discover if updates exist, and for this reason notification files may not be cached for longer than one minute.

3.2. Certificate Authority Use

Certificate Authorities that use RRDP MUST include an instance of an SIA AccessDescription extension in resource certificates they produce, in addition to the ones defined in [RFC6487],

```
AccessDescription ::= SEQUENCE {
  accessMethod OBJECT IDENTIFIER,
  accessLocation GeneralName }
```

This extension MUST use an accessMethod of id-ad-rpkiNotify, see Section 6:

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
  dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-rpkiNotify OBJECT IDENTIFIER ::= { id-ad 13 }
```

The accessLocation MUST be an HTTPS URI as defined in [RFC7230], that will point to the update notification file for the repository server that publishes the products of this Certificate Authority certificate.

3.3. Repository Server Use

3.3.1. Initialisation

When the repository server initialises it performs the following actions:

- The server MUST generate a new random version 4 UUID (see section 4.1.3 of [RFC4122]) to be used as the session_id
- The server MUST then generate a snapshot file for serial number ONE for this new session that includes all currently known published objects that the repository server is responsible for. Note that this snapshot file may contain zero publish elements at this point if no objects have been submitted for publication yet.
- This snapshot file MUST be made available at a URL that is unique to this session_id and serial number, so that it can be cached indefinitely. The format and caching concerns for snapshot files are explained in more detail in Section 3.5.2.
- After the snapshot file has been published the repository server MUST publish a new notification file that contains the new session_id, has serial number ONE, has one reference to the snapshot file that was just published, and that contains no delta references. The format and caching concerns for update notification files are explained in more detail in Section 3.5.1.

3.3.2. Publishing Updates

Whenever the repository server receives updates from a Certificate Authority it MUST generate new snapshot and delta files within one minute. If a Repository Server services a large number of Certificate Authorities it MAY choose to combine updates from multiple CAs. If a Repository Server combines updates in this way, it MUST ensure that publication never postponed for longer than one minute for any of the CAs involved.

Updates are processed as follows:

- The new repository serial number MUST be one greater than the current repository serial number.
- A new delta file MUST be generated for this new serial. This delta file MUST include all new, replaced and withdrawn objects for multiple CAs if applicable, as a single change set.
- This delta file MUST be made available at a URL that is unique to the current session_id and serial number, so that it can be cached indefinitely.

- The format and caching concerns for delta files are explained in more detail in Section 3.5.3.

- The repository server MUST also generate a new snapshot file for this new serial. This file MUST contain all "publish" elements for all current objects.

- The snapshot file MUST be made available at a URL that is unique to this session and new serial, so that it can be cached indefinitely.

- The format and caching concerns for snapshot files are explained in more detail in Section 3.5.2.

- Any older delta files that, when combined with all more recent delta files, will result in total size of deltas exceeding the size of the snapshot, MUST be excluded to avoid that Relying Parties download more data than necessary.

- A new notification file MUST now be created by the repository server. This new notification file MUST include a reference to the new snapshot file, and all delta files selected in the previous steps.

- The format and caching concerns for update notification files are explained in more detail in Section 3.5.1.

If the repository server is not capable of performing the above for some reason, then it MUST perform a full re-initialisation, as explained above in Section 3.3.1.

3.4. Relying Party Use

3.4.1. Processing the Update Notification File

When a Relying Party performs RPKI validation and learns about a valid certificate with an SIA entry for the RRDP protocol, it SHOULD use this protocol as follows.

The Relying Party MUST download the update notification file, unless an update notification file was already downloaded and processed from the same location in this validation run, or because a polling strategy was used (see Section 3.4.4).

It is RECOMMENDED that Relying Party uses a "User-Agent" header explained in section 5.5.3. of [RFC7231] to identify the name and version of the Relying Party software used. It is useful to track

capabilities of Relying Parties in the event of changes to the RPKI standards.

When the Relying Party downloads an update notification file it MUST verify the file format and validation steps described in section Section 3.5.1.3. If this verification fails, the file MUST be rejected and RRDP cannot be used. See Section 3.4.5 for considerations.

The Relying Party MUST verify whether the session_id matches the last known session_id for this update notification file location. Note that even though the session_id is a random UUID value, it alone MUST NOT be used by a Relying Party as a unique identifier of a session, but always together with the location of the notification file. The reason for this is that a malicious server can use an existing session_id from another Repository Server.

If the session_id matches the last known session_id, then a Relying Party MAY download and process missing delta files as described in Section 3.4.2, provided that all delta files for serial numbers between the last processed serial number and the current serial number in the notification file can be processed this way.

If the session_id matches the last known session_id, but delta files were not used, then the Relying Party MUST download and process the snapshot file on the update notification file as described in Section 3.4.3.

If the session_id does not match the last known session_id, the Relying Party MUST update its last known session_id to the value specified in the downloaded notification file. The Relying Party MUST then download and process the snapshot file specified in the downloaded update notification file as described in Section 3.4.3.

3.4.2. Processing Delta Files

If an update notification file contains a contiguous chain of links to delta files from the last processed serial number to the current serial number, then Relying Parties MUST attempt to download and process all delta files in order of serial number as follows.

When the Relying Party downloads a delta file it MUST verify the file format and perform validation steps described in Section 3.5.3.3. If this verification fails, the file MUST be rejected.

Furthermore the Relying Party MUST verify that the hash of the contents of this file matches the hash on the update notification

file that referenced it. In case of a mismatch of this hash, the file MUST be rejected.

If a Relying Party retrieved a delta file that is valid according to the above criteria, it performs the following actions:

- The Relying Party MUST verify that the session_id matches the session_id of the notification file. If the session_id values do not match the file MUST be rejected.

- The Relying Party MUST verify that the serial number of this delta file is exactly one greater than the last processed serial number for this session_id, and if not this file MUST be rejected.

- The Relying Party SHOULD add all publish elements to a local storage and update its last processed serial number to the serial number of this delta file.

- When a Relying Party encounters a "withdraw" element, or a "publish" element where an object is replaced, in a delta that it retrieves from a Repository Server, it MUST verify that the object to be withdrawn or replaced was retrieved from this same Repository Server, before applying the appropriate action. Failing to do so will leave the Relying Party vulnerable to malicious Repository Servers instructing it to delete or change arbitrary objects.

If any delta file is rejected Relying Parties MUST process the current Snapshot File instead, as described in Section 3.4.3.

3.4.3. Processing a Snapshot File

Snapshot Files MUST only be used if Delta Files are unavailable, or were rejected. As is ensured, if the process described in Section 3.4.1 is followed.

When the Relying Party downloads a snapshot file it MUST verify the file format and validation steps described in Section 3.5.2.3. If this verification fails, the file MUST be rejected.

Furthermore the Relying Party MUST verify that the hash of the contents of this file matches the hash on the update notification file that referenced it. In case of a mismatch of this hash, the file MUST be rejected.

If a Relying Party retrieved a snapshot file that is valid according to the above criteria, it performs the following actions:

- The Relying Party MUST verify that the session_id matches the session_id of the notification file. If the session_id values do not match the file MUST be rejected.

- The Relying Party MUST verify that the serial number of this snapshot file is greater than the last processed serial number for this session_id. If this fails the file MUST be rejected.

- The Relying Party SHOULD then add all publish elements to a local storage and update its last processed serial number to the serial number of this snapshot file.

If a Snapshot File is rejected that means that RRDP cannot be used. See Section 3.4.5 for considerations.

3.4.4. Polling the Update Notification File

Once a Relying Party has learned about the location, session_id and last processed serial number of repository that uses the RRDP protocol, the Relying Party MAY start polling the repository server for updates. However the Relying Party MUST NOT poll for updates more often than once every 1 minute, and in order to reduce data usage Relying Parties MUST use the "If-Modified-Since" header explained in section 3.3 of [RFC7232] in requests.

If a Relying Party finds that updates are available it SHOULD download and process the file as described in Section 3.4.1, and initiate a new RPKI object validation process. However, a detailed description of the RPKI object validation process itself is out of scope of this document.

3.4.5. Considerations Regarding Operational Failures in RRDP

If a Relying Party experiences any issues with retrieving or processing any of the files used in this protocol, it will be unable to retrieve new RPKI data from the affected Repository Server.

Relying Parties could attempt to use alternative repository access mechanisms, if they are available, according to the accessMethod element value(s) specified in the SIA of the associated certificate (see Section 4.8.8 of [RFC6487]).

Furthermore Relying Parties may wish to employ re-try strategies while fetching RRDP files. Relying Parties are also advised to keep old objects in their local cache so that validation can be done using old objects.

It is also recommendable that re-validation and retrieval is performed pro-actively before manifests or CRLs go stale, or certificates expire, to ensure that problems on the side of the Relying Party can be identified and resolved before they cause major concerns.

3.5. File Definitions

3.5.1. Update Notification File

3.5.1.1. Purpose

The update notification file is used by Relying Parties to discover whether any changes exist between the state of the repository and the Relying Party's cache. It describes the location of the files containing the snapshot and incremental deltas which can be used by the Relying Party to synchronise with the repository.

3.5.1.2. Cache Concerns

A repository server MAY use caching infrastructure to cache the notification file and reduce the load of HTTPS requests. However, since this file is used by Relying Parties to determine whether any updates are available the repository server SHOULD ensure that this file is not cached for longer than 1 minute. An exception to this rule is that it is better to serve a stale notification file, than no notification file.

How this is achieved exactly depends on the caching infrastructure used. In general a repository server may find certain HTTP headers to be useful, such as: "Cache-Control: max-age=60" (see Section 5.2 of [RFC7234]). Another approach can be to have the repository server push out new versions of the notification file to the caching infrastructure when appropriate.

In case of a high load on a repository server or its distribution network, the Cache-Control HTTP header, or a similar mechanism, MAY be used to suggest an optimal (for the repository server) poll interval for Relying Parties. However, setting it to an interval longer than 1 hour is NOT RECOMMENDED. Relying parties SHOULD align the suggested interval with their operational practices and the expected update frequency of RPKI repository data, and MAY discard suggested value.

3.5.1.3. File Format and Validation

Example notification file:

```
<notification xmlns="http://www.ripe.net/rpki/rrdp"
      version="1"
      session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
      serial="3">
  <snapshot uri="https://host/9d-8/3/snapshot.xml" hash="AB"/>
  <delta serial="3" uri="https://host/9d-8/3/delta.xml" hash="CD"/>
  <delta serial="2" uri="https://host/9d-8/2/delta.xml" hash="EF"/>
</notification>
```

Note: URIs and hash values in this example are shortened because of formatting.

The following validation rules MUST be observed when creating or parsing notification files:

- A Relying Party MUST reject any update notification file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- The XML namespace MUST be http://www.ripe.net/rpki/rrdp
- The encoding MUST be US-ASCII
- The version attribute in the notification root element MUST be 1
- The session_id attribute MUST be a random version 4 UUID ([RFC4122]), unique to this session
- The serial attribute MUST be an unbounded, unsigned positive integer in decimal format indicating the current version of the repository.
- The notification file MUST contain exactly one ’snapshot’ element for the current repository version.
- If delta elements are included they MUST form a contiguous sequence of serial numbers starting at a revision determined by the repository server, up to the serial number mentioned in the notification element. Note that the elements may not be ordered.
- The hash attribute in snapshot and delta elements MUST be the hexadecimal encoding of the SHA-256 [SHS] hash of the referenced file. The Relying Party MUST verify this hash when the file is retrieved and reject the file if the hash does not match.

3.5.2. Snapshot File

3.5.2.1. Purpose

A snapshot is intended to reflect the complete and current contents of the repository for a specific session and version. Therefore it MUST contain all objects from the repository current as of the time of the publication.

3.5.2.2. Cache Concerns

A snapshot reflects the content of the repository at a specific point in time, and for that reason can be considered immutable data. Snapshot files MUST be published at a URL that is unique to the specific session and serial.

Because these files never change, they MAY be cached indefinitely. However, in order to prevent that these files use a lot of space in caching infrastructure it is RECOMMENDED that a limited interval is used in the order of hours or days.

To avoid race conditions where a Relying Party downloads a notification file moments before it’s updated, Repository Servers SHOULD retain old snapshot files for at least 5 minutes after a new notification file is published.

3.5.2.3. File Format and Validation

Example snapshot file:

```
<snapshot xmlns="http://www.ripe.net/rpki/rrdp"
       version="1"
       session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
       serial="2">
  <publish uri="rsync://rpki.ripe.net/Alice/Bob.cer">
    ZXhhbXBsZTE=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.mft">
    ZXhhbXBsZTI=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.crl">
    ZXhhbXBsZTM=
  </publish>
</snapshot>
```

The following rules MUST be observed when creating or parsing snapshot files:

- A Relying Party MUST reject any snapshot file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- The XML namespace MUST be http://www.ripe.net/rpki/rrdp.
- The encoding MUST be US-ASCII.
- The version attribute in the notification root element MUST be 1
- The session_id attribute MUST match the expected session_id in the reference in the notification file.
- The serial attribute MUST match the expected serial in the reference in the notification file.
- Note that the publish element is similar to the publish element defined in the publication protocol [I-D.ietf-sidr-publication]. However, the "tag" attribute is not used here because it is not relevant to Relying Parties. The "hash" attribute is not used here because this file represents a complete current state of the repository, and therefore it is not relevant to know which existing RPKI object (if any) is updated.

3.5.3. Delta File

3.5.3.1. Purpose

An incremental delta file contains all changes for exactly one serial increment of the repository server. In other words a single delta will typically include all the new objects, updated objects and withdrawn objects that a Certification Authority sent to the repository server. In its simplest form the update could concern only a single object, but it is RECOMMENDED that CAs send all changes for one of their key pairs (updated objects as well as a new manifest and CRL) as one atomic update message.

3.5.3.2. Cache Concerns

Deltas reflect the difference between two consecutive versions of a repository for a given session. For that reason deltas can be considered immutable data. Delta files MUST be published at a URL that is unique to the specific session and serial.

Because these files never change, they MAY be cached indefinitely. However, in order to prevent these files from using a lot of space in caching infrastructure it is RECOMMENDED that a limited interval is used in the order of hours or days.

To avoid race conditions where a Relying Party downloads a notification file moments before it's updated, Repository Servers SHOULD retain old delta files for at least 5 minutes after they are no longer included in the latest notification file.

3.5.3.3. File Format and Validation

Example delta file:

```
<delta xmlns="http://www.ripe.net/rpki/rrdp"
       version="1"
       session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
       serial="3">
  <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.mft"
           hash="50d8...545c">
    ZXhhbXBsZTQ=
  </publish>
  <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.crl"
           hash="5fb1...6a56">
    ZXhhbXBsZTU=
  </publish>
  <withdraw uri="rsync://rpki.ripe.net/repo/Alice/Bob.cer"
            hash="caeb...15c1"/>
</delta>
```

Note that a formal RELAX NG specification of this file format is included later in this document. A Relying Party MUST NOT process any delta file that is incomplete or not well-formed.

The following validation rules MUST be observed when creating or parsing delta files:

- A Relying Party MUST reject any delta file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- The XML namespace MUST be http://www.ripe.net/rpki/rrdp.
- The encoding MUST be US-ASCII.
- The version attribute in the delta root element MUST be 1
- The session_id attribute MUST be a random version 4 UUID unique to this session
- The session_id attribute MUST match the expected session_id in the reference in the notification file.

- The serial attribute MUST match the expected serial in the reference in the notification file.

- Note that the publish element is similar to the publish element defined in the publication protocol [I-D.ietf-sidr-publication]. However, the "tag" attribute is not used here because it is not relevant to Relying Parties.

3.5.4. XML Schema

The following is a RELAX NG compact form schema describing version 1 of this protocol.

```
#
# RelaxNG schema for RPKI Repository Delta Protocol (RRDP).
#

default namespace = "http://www.ripe.net/rpki/rrdp"

version = xsd:positiveInteger   { maxInclusive="1" }
serial  = xsd:positiveInteger
uri     = xsd:anyURI
uuid    = xsd:string            { pattern = "[\-0-9a-fA-F]+" }
hash    = xsd:string            { pattern = "[0-9a-fA-F]+" }
base64  = xsd:base64Binary

# Notification file: lists current snapshots and deltas

start |= element notification {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element snapshot {
    attribute uri  { uri },
    attribute hash { hash }
  },
  element delta {
    attribute serial { serial },
    attribute uri    { uri },
    attribute hash   { hash }
  }*
}

# Snapshot segment: think DNS AXFR.

start |= element snapshot {
  attribute version    { version },
  attribute session_id { uuid },
```

```
    attribute serial     { serial },
    element publish      {
      attribute uri { uri },
      base64
    }*
  }

  # Delta segment: think DNS IXFR.

  start |= element delta {
    attribute version    { version },
    attribute session_id { uuid },
    attribute serial     { serial },
    delta_element+
  }

  delta_element |= element publish  {
    attribute uri  { uri },
    attribute hash { hash }?,
    base64
  }

  delta_element |= element withdraw {
    attribute uri  { uri },
    attribute hash { hash }
  }

  # Local Variables:
  # indent-tabs-mode: nil
  # comment-start: "# "
  # comment-start-skip: "#[ \t]*"
  # End:
```

4. Operational Considerations

4.1. Compatibility with previous standards

This protocol has been designed to replace rsync as a distribution mechanism of an RPKI repository. However, it is also designed to co-exist with existing implementations based on rsync, to enable smooth transition from one distribution mechanism to another.

For every repository object listed in the snapshot and delta files both the hash of the object's content and the rsync URI [RFC5781] of its location in the repository are listed. This makes it possible to distribute the same RPKI repository, represented by a set of files on a filesystem, using both rsync and RRDP. It also enables Relying

Parties tools to query, combine, and consequently validate objects from repositories of different types.

4.2. Distribution considerations

One of the design goals of RRDP was to minimise load on a repository server while serving clients. To achieve this, neither the content, nor the URLs of the snapshot and delta files are modified after they have been published in the notification file. This allows their effective distribution, either by a single HTTP server, or using a Content Distribution Network (CDN).

The RECOMMENDED way for Relying Parties to keep up with the repository updates is to poll the Update Notification File for changes. The content of that file is updated with every new serial version of a repository (while its URL remains stable). To effectively implement distribution of the notification file, an "If-Modified-Since" HTTP request header is required to be present in all requests for notification file (see Section 3.4.4.) Therefore it is RECOMMENDED that Relying Party tools implement a mechanism to keep track of a previous successful fetch of a notification file.

Implementations of RRDP should also take care of not producing new versions of the repository (and subsequently, new Notification, Snapshot and Delta files) too often. Usually the maintenance of the RPKI repository includes regular updates of manifest and CRL objects, performed on a schedule. This often results in bursts of repository updates during a short period of time. Since the Relying Parties are required to poll for the Update Notification File not more often than once per minute (Section 3.4.4), it is not practical to generate new serial versions of the repository much more often than 1 per minute. It is allowed to combine multiple updates, possibly from different CAs, into a new serial repository version (Section 3.3.2). This will significantly shorten the size of the Update Notification File and total amount of data distributed to all Relying Parties.

4.3. HTTPS considerations

Note that a Man-in-the-Middle (MITM) cannot produce validly signed RPKI data, but can perform withhold or replay attacks targeting a Relying Party, and keep the Relying Party from learning about changes in the RPKI. Because of this Relying Parties SHOULD do TLS certificate and host name validation when they fetch from an RRDP Repository Server.

Relying Party tools SHOULD log any TLS certificate or host name validation issues found, so that an operator can investigate the cause. However, such validation issues are often due to

configuration errors, or a lack of a common TLS trust anchor. In these cases it is better if the Relying Party retrieves the signed RPKI data regardless, and performs validation on it. Therefore Relying Party MUST continue to retrieve the data in case of errors. The Relying Party MAY choose to log encountered issues only when fetching the notification update file, but not when it subsequently fetches snapshot or delta files from the same host. Furthermore the Relying Party MAY provide a way for operators to accept untrusted connections for a given host, after the cause has been identified.

It is RECOMMENDED that Relying Parties and Repository Servers follow the Best Current Practices outlined in [RFC7525] on the use of HTTP over TLS (HTTPS) [RFC7230]. Relying Parties SHOULD do TLS certificate and host name validation using subjectAltName dNSName identities as described in [RFC6125]. The rules and guidelines defined in [RFC6125] apply here, with the following considerations:

- Relying Parties and Repository Servers SHOULD support the DNS-ID identifier type. The DNS-ID identifier type SHOULD be present in Repository Server certificates.

- DNS names in Repository Server certificates SHOULD NOT contain the wildcard character "*".

- A CN field may be present in Repository Server certificates's subject name, but SHOULD NOT be used for authentication within the rules described in [RFC6125].

- This protocol does not require the use of SRV-IDs.

- This protocol does not require the use of URI-IDs.

Note however that this validation is done on a best effort basis, and serves to highlight potential issues, but RPKI object security does not depend on this. Therefore Relying Parties MAY deviate from the validation steps listed above.

5. Security Considerations

RRDP deals exclusively with transfer of RPKI objects from a repository server to a Relying Party. The trust relation between a Certificate Authority and its repository server is out of scope for this document. However, it should be noted that from a Relying Party point of view all RPKI objects (certificates, CRLs, and CMS-wrapped objects) are already covered by object security mechanisms including signed manifests. This allows validation of these objects even though the repository server itself is not trusted. This document makes no change to RPKI validation procedures per se.

The original RPKI transport protocol is rsync, which offers no channel security mechanism. RRDP replaces the use of rsync by HTTPS; while the channel security mechanism underlying RRDP (HTTPS) is not a cure-all, it does make some forms of denial of service attack more difficult for the attacker. HTTPS issues are discussed in more detail in Section 4.3.

Supporting both RRDP and rsync necessarily increases the number of opportunities for a malicious RPKI Certificate Authority to perform denial of service attacks on Relying Parties, by expanding the number of URIs which the Relying Party may need to contact in order to complete a validation run. However, other than the relative cost of HTTPS versus rsync, adding RRDP to the mix does not change this picture significantly: with either RRDP or rsync a malicious Certificate Authority can supply an effectively infinite series of URIs for the Relying Party to follow. The only real solution to this is for the Relying Party to apply some kind of bound to the amount of work it is willing to do. Note also that the attacker in this scenario must be an RPKI Certificate Authority, since otherwise the normal RPKI object security checks would reject the malicious URIs.

Processing costs for objects retrieved using RRDP may be somewhat different from the same objects retrieved using rsync: because RRDP treats an entire set of changes as a unit (one "delta"), it may not be practical to start processing any of the objects in the delta until the entire delta has been received. With rsync, by contrast, incremental processing may be easy, but the overall cost of transfer may be higher, as may be the number of corner cases in which the Relying Party retrieves some but not all of the updated objects. Overall, RRDP's behavior is closer to a proper transactional system, which (probably) leads to an overall reliability increase.

RRDP is designed to scale much better than rsync. In particular, RRDP is designed to allow use of HTTPS caching infrastructure to reduce load on primary Repository Servers and increase resilience against denial of service attacks on the RPKI publication service.

6. IANA Considerations

IANA is requested to update the reference for id-ad-rpkiNotify to this document in the PKIX Access Descriptor registry [IANA-AD-NUMBERS].

7. Acknowledgements

The authors would like to thank David Mandelberg for reviewing this document.

8. References

8.1. Normative References

[I-D.ietf-sidr-publication]
Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-12 (work in progress), March 2017.

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

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010, <http://www.rfc-editor.org/info/rfc5781>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <http://www.rfc-editor.org/info/rfc6125>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <http://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, <http://www.rfc-editor.org/info/rfc6487>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard", March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

8.2. Informative References

[IANA-AD-NUMBERS] "SMI Security for PKIX Access Descriptor", <http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.48>.

[RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <http://www.rfc-editor.org/info/rfc6486>.

[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, <http://www.rfc-editor.org/info/rfc6488>.

[rsync] "Rsync home page", <https://rsync.samba.org>.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net

Bryan Weber
Cobenian

Email: bryan@cobenian.com

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net

SIDR
Internet-Draft
Intended status: Standards Track
Expires: May 30, 2015

R. Kisteleki
RIPE NCC
B. Haberman
JHU APL

November 26, 2014

Securing RPSL Objects with RPKI Signatures
draft-ietf-sidr-rpsl-sig-06.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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 30, 2015.

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 use them to secure such (both imported and downloaded) database objects, by applying a form of 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 an Internet number resource. 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].

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, separated by an ASCII "+" character (if more than one attribute is enumerated). The list must include any attribute 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:

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 applications for such a mechanism include the case of a route[6] object, where both the prefix owner's and the AS owner's signature is expected (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, yet others do not carry operationally important information. This specification allows the maintainer of such an object to define which attributes are signed and which are not, from among all the attributes of the object; in other words, we define a way of including important attributes while excluding irrelevant ones. Allowing the maintainer an object to select the attributes that are covered by the digital signature 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 MUST reject the signature and threat the object as a regular, 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.

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.

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 differences in representations of the same object, which would otherwise cause signature verification to fail. Examples of this could be:

1. 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.)

2. the difference of line terminators across different systems.

This means that the destination database might change parts of the submitted data after it was signed, which would cause signature verification to fail. This document specifies strict canonicalization rules to overcome this problem.

The following steps MUST be applied in order to achieve canonicalized representation of an object, before the actual signature (verification) process can begin:

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].

   * 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 notaion [RFC4632].

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

6. Tab characters ("\t") MUST be converted to spaces.

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 this object this time.

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; the signer MAY select additional attributes.

3. Arrange the selected attributes according to the selection sequence specified in the "a" field as above, omiting 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).

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 over such an object, the following steps MUST be performed:

1. Verify the syntax of the "signature" attribute (ie. whether it contains the mandatory and optional components and the syntax of these fields mathces 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].

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 miminum set of attributes for that object type.

5. Arrange the selected attributes according to the selection sequence provided in the value of the "a" field, omitting all non-signed attributes.

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

7. Apply the canonicalization procedure 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.

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, since these usually make sense only within the context of such a database, whereas the scope of the signatures is only one specific object. Since the attributes in the referred object (such as mnt-by, admin-c, tech-c, ...) can change without any modifications to the signed object, signing such attributes could lead to false sense of security in terms of the contents of the signed data; therefore should only be done in order to provide full integrity protection of the object itself.

The newly constructed "signature" attribute is always included in the 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]:

* 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 signatrure is attached to:

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

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

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

- 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):

- MUST be an end-entity (ie. non-CA) certificate

- MUST conform to the X.509 PKIX Resource Certificate profile [RFC6487]

- MUST have an [RFC3779] extension that contains or covers at least one Internet number resource included in a signed attribute.

- SHOULD NOT be used to verify more than one signed object (ie. 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.

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

Email: brian@innovationslab.net

SIDR
Internet-Draft
Obsoletes: 6487 (if approved)
Intended status: Standards Track
Expires: September 10, 2015

G. Huston
G. Michaelson
APNIC
R. Loomans
Suncorp
A. Newton
ARIN
R. Hansen
BBN

March 9, 2015

A Profile for X.509 PKIX Resource Certificates
draft-rhansen-sidr-rfc6487bis-00

Abstract

This document defines a standard profile for X.509 certificates for the purpose of supporting validation of assertions of "right-of-use" of Internet Number Resources (INRs). The certificates issued under this profile are used to convey the issuer's authorization of the subject to be regarded as the current holder of a "right-of-use" of the INRs that are described in the certificate. This document contains the normative specification of Certificate and Certificate Revocation List (CRL) syntax in the Resource Public Key Infrastructure (RPKI). This document also specifies profiles for the format of certificate requests and specifies the Relying Party RPKI certificate path validation procedure.

This document obsoletes RFC 6487.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Table of Contents

1. Introduction

This document defines a standard profile for X.509 certificates [X.509] for use in the context of certification of Internet Number Resources (INRs), i.e., IP Addresses and Autonomous System (AS) numbers. Such certificates are termed "resource certificates". A resource certificate is a certificate that conforms to the PKIX profile [RFC5280], and that conforms to the constraints specified in this profile. A resource certificate attests that the issuer has granted the subject a "right-of-use" for a listed set of IP addresses and/or Autonomous System numbers.

This document is referenced by Section 7 of the "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)" [RFC6484]. It is an integral part of that policy and the normative specification for certificate and Certificate Revocation List (CRL) syntax used in the RPKI. The document also specifies profiles for the format of certificate requests, and the relying party (RP) RPKI certificate path validation procedure.

Resource certificates are to be used in a manner that is consistent with the RPKI Certificate Policy (CP) [RFC6484]. They are issued by entities that assign and/or allocate public INRs, and thus the RPKI is aligned with the public INR distribution function. When an INR is allocated or assigned by a number registry to an entity, this allocation can be described by an associated resource certificate. This certificate is issued by the number registry, and it binds the certificate subject's key to the INRs enumerated in the certificate.

One or two critical extensions, the IP Address Delegation or AS Identifier Delegation Extensions [RFC3779], enumerate the INRs that were allocated or assigned by the issuer to the subject.

Relying party (RP) validation of a resource certificate is performed in the manner specified in Section 7.1. This validation procedure differs from that described in Section 6 of [RFC5280], such that:

- o additional validation processing imposed by the INR extensions is required,
- o a confirmation of a public key match between the CRL issuer and the resource certificate issuer is required, and
- o the resource certificate is required to conform to this profile.

This profile defines those fields that are used in a resource certificate that MUST be present for the certificate to be valid. Any extensions not explicitly mentioned MUST be absent. The same applies to the CRLs used in the RPKI, that are also profiled in this document. A Certification Authority (CA) conforming to the RPKI CP MUST issue certificates and CRLs consistent with this profile.

1.1. Terminology

It is assumed that the reader is familiar with the terms and concepts described in "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" [RFC5280], and "X.509 Extensions for IP Addresses and AS Identifiers" [RFC3779].

The 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].

2. Describing Resources in Certificates

The framework for describing an association between the subject of a certificate and the INRs currently under the subject’s control is described in [RFC3779]. This profile further requires that:

- o Every resource certificate MUST contain either the IP Address Delegation or the Autonomous System Identifier Delegation extension, or both.
- o These extensions MUST be marked as critical.
- o The sorted canonical format describing INRs, with maximal spanning ranges and maximal spanning prefix masks, as defined in [RFC3779],

MUST be used for the resource extension field, except where the "inherit" construct is used instead.

When validating a resource certificate, an RP MUST verify that the INRs described in the issuer's resource certificate encompass the INRs of the resource certificate being validated. In this context, "encompass" allows for the issuer's INRs to be the same as, or a strict superset of, the subject's INRs.

3. End-Entity (EE) Certificates and Signing Functions in the RPKI

As noted in [RFC6480], the primary function of end-entity (EE) certificates in the RPKI is the verification of signed objects that relate to the usage of the INRs described in the certificate, e.g., Route Origin Authorizations (ROAs) and manifests.

The private key associated with an EE certificate is used to sign a single RPKI signed object, i.e., the EE certificate is used to validate only one object. The EE certificate is embedded in the object as part of a Cryptographic Message Syntax (CMS) signed-data structure [RFC6488]. Because of the one-to-one relationship between the EE certificate and the signed object, revocation of the certificate effectively revokes the corresponding signed object.

An EE certificate may be used to validate a sequence of signed objects, where each signed object in the sequence overwrites the previous instance of the signed object in the repository publication point, such that only one instance of the signed object is published at any point in time (e.g., an EE certificate MAY be used to sign a sequence of manifests [RFC6486]). Such EE certificates are termed "sequential use" EE certificates.

EE certificates used to validate only one instance of a signed object, and are not used thereafter or in any other validation context, are termed "one-time-use" EE certificates.

4. Resource Certificates

A resource certificate is a valid X.509 public key certificate, consistent with the PKIX profile [RFC5280], containing the fields listed in this section. Only the differences from [RFC5280] are noted below.

Unless specifically noted as being OPTIONAL, all the fields listed here MUST be present, and any other fields MUST NOT appear in a conforming resource certificate. Where a field value is specified here, this value MUST be used in conforming resource certificates.

4.1. Version

As resource certificates are X.509 version 3 certificates, the version MUST be 3 (i.e., the value of this field is 2).

RPs need not process version 1 or version 2 certificates (in contrast to [RFC5280]).

4.2. Serial Number

The serial number value is a positive integer that is unique for each certificate issued by a given CA.

4.3. Signature Algorithm

The algorithm used in this profile is specified in [RFC6485].

4.4. Issuer

The value of this field is a valid X.501 distinguished name.

An issuer name MUST contain one instance of the CommonName attribute and MAY contain one instance of the serialNumber attribute. If both attributes are present, it is RECOMMENDED that they appear as a set. The CommonName attribute MUST be encoded using the ASN.1 type PrintableString [X.680]. Issuer names are not intended to be descriptive of the identity of issuer.

The RPKI does not rely on issuer names being globally unique, for reasons of security. However, it is RECOMMENDED that issuer names be generated in a fashion that minimizes the likelihood of collisions. See Section 8 for (non-normative) suggested name-generation mechanisms that fulfill this recommendation.

4.5. Subject

The value of this field is a valid X.501 distinguished name [RFC4514], and is subject to the same constraints as the issuer name.

In the RPKI, the subject name is determined by the issuer, not proposed by the subject [RFC6481]. Each distinct subordinate CA and EE certified by the issuer MUST be identified using a subject name that is unique per issuer. In this context, "distinct" is defined as an entity and a given public key. An issuer SHOULD use a different subject name if the subject’s key pair has changed (i.e., when the CA issues a certificate as part of re-keying the subject.) Subject names are not intended to be descriptive of the identity of subject.

4.6. Validity

The certificate validity period is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter).

While a CA is typically advised against issuing a certificate with a validity period that spans a greater period of time than the validity period of the CA's certificate that will be used to validate the issued certificate, in the context of this profile, a CA MAY have valid grounds to issue a subordinate certificate with a validity period that exceeds the validity period of the CA's certificate.

4.6.1. notBefore

The "notBefore" time SHOULD be no earlier than the time of certificate generation.

In the RPKI, it is valid for a certificate to have a value for this field that pre-dates the same field value in any superior certificate. Relying Parties SHOULD NOT attempt to infer from this time information that a certificate was valid at a time in the past, or that it will be valid at a time in the future, as the scope of an RP's test of validity of a certificate refers specifically to validity at the current time.

4.6.2. notAfter

The "notAfter" time represents the anticipated lifetime of the current resource allocation or assignment arrangement between the issuer and the subject.

It is valid for a certificate to have a value for this field that post-dates the same field value in any superior certificate. The same caveats apply to RP's assumptions relating to the certificate's validity at any time other than the current time.

4.7. Subject Public Key Info

The algorithm used in this profile is specified in [RFC6485].

4.8. Resource Certificate Extensions

The following X.509 v3 extensions MUST be present in a conforming resource certificate, except where explicitly noted otherwise. Each extension in a resource certificate is designated as either critical or non-critical. A certificate-using system MUST reject the

certificate if it encounters an extension not explicitly mentioned in this document. This is in contrast to [RFC5280] which allows non-critical extensions to be ignored.

4.8.1. Basic Constraints

The Basic Constraints extension field is a critical extension in the resource certificate profile, and MUST be present when the subject is a CA, and MUST NOT be present otherwise.

The issuer determines whether the "cA" boolean is set.

The Path Length Constraint is not specified for RPKI certificates, and MUST NOT be present.

4.8.2. Subject Key Identifier

This extension MUST appear in all resource certificates. This extension is non-critical.

The Key Identifier used for resource certificates is the 160-bit SHA-1 hash of the value of the DER-encoded ASN.1 bit string of the Subject Public Key, as described in Section 4.2.1.2 of [RFC5280].

4.8.3. Authority Key Identifier

This extension MUST appear in all resource certificates, with the exception of a CA who issues a "self-signed" certificate. In a self-signed certificate, a CA MAY include this extension, and set it equal to the Subject Key Identifier. The authorityCertIssuer and authorityCertSerialNumber fields MUST NOT be present. This extension is non-critical.

The Key Identifier used for resource certificates is the 160-bit SHA-1 hash of the value of the DER-encoded ASN.1 bit string of the issuer’s public key, as described in Section 4.2.1.1 of [RFC5280].

4.8.4. Key Usage

This extension is a critical extension and MUST be present.

In certificates issued to certification authorities only, the keyCertSign and CRLSign bits are set to TRUE, and these MUST be the only bits set to TRUE.

In EE certificates, the digitalSignature bit MUST be set to TRUE and MUST be the only bit set to TRUE.

4.8.5. Extended Key Usage

The Extended Key Usage (EKU) extension MUST NOT appear in any CA certificate in the RPKI. This extension also MUST NOT appear in EE certificates used to verify RPKI objects (e.g., ROAs or manifests. The extension MUST NOT be marked critical.

The EKU extension MAY appear in EE certificates issued to routers or other devices. Permitted values for the EKU OIDs will be specified in Standards Track RFCs issued by other IETF working groups that adopt the RPKI profile and that identify application-specific requirements that motivate the use of such EKUs.

4.8.6. CRL Distribution Points

This extension MUST be present, except in "self-signed" certificates, and it is non-critical. In a self-signed certificate, this extension MUST be omitted.

In this profile, the scope of the CRL is specified to be all certificates issued by this CA issuer.

The CRL Distribution Points (CRLDP) extension identifies the location(s) of the CRL(s) associated with certificates issued by this issuer. The RPKI uses the URI [RFC3986] form of object identification. The preferred URI access mechanism is a single rsync URI ("rsync://") [RFC5781] that references a single inclusive CRL for each issuer.

In this profile, the certificate issuer is also the CRL issuer, implying that the CRLIssuer field MUST be omitted, and the distributionPoint field MUST be present. The Reasons field MUST be omitted.

The distributionPoint MUST contain the fullName field, and MUST NOT contain a nameRelativeToCRLIssuer. The form of the generalName MUST be of type URI.

The sequence of distributionPoint values MUST contain only a single DistributionPoint. The DistributionPoint MAY contain more than one URI value. An rsync URI [RFC5781] MUST be present in the DistributionPoint and MUST reference the most recent instance of this issuer’s CRL. Other access form URIs MAY be used in addition to the rsync URI, representing alternate access mechanisms for this CRL.

4.8.7. Authority Information Access

In the context of the RPKI, this extension identifies the publication point of the certificate of the issuer of the certificate in which the extension appears. In this profile, a single reference to the publication point of the immediate superior certificate MUST be present, except for a "self-signed" certificate, in which case the extension MUST be omitted. This extension is non-critical.

This profile uses a URI form of object identification. The preferred URI access mechanisms is "rsync", and an rsync URI [RFC5781] MUST be specified with an accessMethod value of id-ad-caIssuers. The URI MUST reference the point of publication of the certificate where this Issuer is the subject (the issuer's immediate superior certificate). Other accessMethod URIs referencing the same object MAY also be included in the value sequence of this extension.

A CA MUST use a persistent URL name scheme for CA certificates that it issues [RFC6481]. This implies that a reissued certificate overwrites a previously issued certificate (to the same subject) in the publication repository. In this way, certificates subordinate to the reissued (CA) certificate can maintain a constant Authority Information Access (AIA) extension pointer and thus need not be reissued when the parent certificate is reissued.

4.8.8. Subject Information Access

In the context of the RPKI, this Subject Information Access (SIA) extension identifies the publication point of products signed by the subject of the certificate.

4.8.8.1. SIA for CA Certificates

This extension MUST be present and MUST be marked non-critical.

This extension MUST have an instance of an accessMethod of id-ad-caRepository, with an accessLocation form of a URI that MUST specify an rsync URI [RFC5781]. This URI points to the directory containing all published material issued by this CA, i.e., all valid CA certificates, published EE certificates, the current CRL, manifest, and signed objects validated via EE certificates that have been issued by this CA [RFC6481]. Other accessDescription elements with an accessMethod of id-ad-caRepository MAY be present. In such cases, the accessLocation values describe alternate supported URI access mechanisms for the same directory. The ordering of URIs in this accessDescription sequence reflect the CA's relative preferences for access methods to be used by RPs, with the first element of the sequence being the most preferred by the CA.

This extension MUST have an instance of an AccessDescription with an accessMethod of id-ad-rpkiManifest,

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-rpkiManifest OBJECT IDENTIFIER ::= { id-ad 10 }
```

with an rsync URI [RFC5781] form of accessLocation. The URI points to the CA's manifest of published objects [RFC6486] as an object URL. Other accessDescription elements MAY exist for the id-ad-rpkiManifest accessMethod, where the accessLocation value indicates alternate access mechanisms for the same manifest object.

4.8.8.2. SIA for EE Certificates

This extension MUST be present and MUST be marked non-critical.

This extension MUST have an instance of an accessMethod of id-ad-signedObject,

```
id-ad-signedObject OBJECT IDENTIFIER ::= { id-ad 11 }
```

with an accessLocation form of a URI that MUST include an rsync URI [RFC5781]. This URI points to the signed object that is verified using this EE certificate [RFC6481]. Other accessDescription elements may exist for the id-ad-signedObject accessMethod, where the accessLocation value indicates alternate URI access mechanisms for the same object, ordered in terms of the EE's relative preference for supported access mechanisms.

Other AccessMethods MUST NOT be used for an EE certificates's SIA.

4.8.9. Certificate Policies

This extension MUST be present and MUST be marked critical. It MUST include exactly one policy, as specified in the RPKI Certificate Policy (CP) [RFC6484]. Exactly one policy qualifier MAY be included. If a policy qualifier is included, the policyQualifierId MUST be the Certification Practice Statement (CPS) pointer qualifier type (id-qt-cps).

As noted in [RFC5280], Section 4.2.1.4: "Optional qualifiers, which MAY be present, are not expected to change the definition of the policy." In this case, any optional policy qualifier that MAY be present in a resource certificate MUST NOT change the definition of the RPKI CP [RFC6484].

4.8.10. IP Resources

Either the IP Resources extension, or the AS Resources extension, or both, MUST be present in all RPKI certificates, and if present, MUST be marked critical.

This extension contains the list of IP address resources as per [RFC3779]. The value may specify the "inherit" element for a particular Address Family Identifier (AFI) value. In the context of resource certificates describing public number resources for use in the public Internet, the Subsequent AFI (SAFI) value MUST NOT be used.

This extension MUST either specify a non-empty set of IP address records, or use the "inherit" setting to indicate that the IP address resource set of this certificate is inherited from that of the certificate’s issuer.

4.8.11. AS Resources

Either the AS Resources extension, or the IP Resources extension, or both, MUST be present in all RPKI certificates, and if present, MUST be marked critical.

This extension contains the list of AS number resources as per [RFC3779], or it may specify the "inherit" element. Routing Domain Identifier (RDI) values are NOT supported in this profile and MUST NOT be used.

This extension MUST either specify a non-empty set of AS number records, or use the "inherit" setting to indicate that the AS number resource set of this certificate is inherited from that of the certificate’s issuer.

5. Resource Certificate Revocation Lists

Each CA MUST issue a version 2 CRL that is consistent with [RFC5280]. RPs are NOT required to process version 1 CRLs (in contrast to [RFC5280]). The CRL issuer is the CA. CRLs conforming to this profile MUST NOT include Indirect or Delta CRLs. The scope of each CRL MUST be all certificates issued by this CA.

The issuer name is as in Section 4.4 above.

Where two or more CRLs are issued by the same CA, the CRL with the highest value of the "CRL Number" field supersedes all other CRLs issued by this CA.

The algorithm used in CRLs issued under this profile is specified in [RFC6485].

The contents of the CRL are a list of all non-expired certificates that have been revoked by the CA.

An RPKI CA MUST include the two extensions, Authority Key Identifier and CRL Number, in every CRL that it issues. The Authority Key Identifier extension MUST follow the same restrictions as in Section 4.8.3 above. RPs MUST be prepared to process CRLs with these extensions. No other CRL extensions are allowed. The extensions mentioned above MUST NOT appear more than once each.

For each revoked resource certificate, only the two fields, Serial Number and Revocation Date, MUST be present, and all other fields MUST NOT be present. No CRL entry extensions are supported in this profile, and CRL entry extensions MUST NOT be present in a CRL.

6. Resource Certificate Requests

A resource certificate request MAY use either of PKCS#10 or Certificate Request Message Format (CRMF). A CA MUST support certificate issuance in PKCS#10 and a CA MAY support CRMF requests.

Note that there is no certificate response defined in this profile. For CA certificate requests, the CA places the resource certificate in the repository, as per [RFC6484]. No response is defined for EE certificate requests.

6.1. PCKS#10 Profile

This profile refines the specification in [RFC2986], as it relates to resource certificates. A Certificate Request Message object, formatted according to PKCS#10, is passed to a CA as the initial step in issuing a certificate.

With the exception of the SubjectPublicKeyinfo and the SIA extension request, the CA is permitted to alter any field in the request when issuing a certificate.

6.1.1. PKCS#10 Resource Certificate Request Template Fields

This profile applies the following additional requirements to fields that MAY appear in a CertificationRequestInfo:

Version

This field is mandatory and MUST have the value 0.

Subject
This field SHOULD be empty (i.e., NULL), in which case the CA MUST generate a subject name that is unique in the context of certificates issued by this CA. This field is allowed to be non-empty only for a re-key/reissuance request, and only if the CA has adopted a policy (in its Certificate Practice Statement (CPS)) that permits reuse of names in these circumstances.

SubjectPublicKeyInfo
This field specifies the subject's public key and the algorithm with which the key is used. The algorithm used in this profile is specified in [RFC6485].

Attributes
[RFC2986] defines the attributes field as key-value pairs where the key is an OID and the value's structure depends on the key.

The only attribute used in this profile is the extensionRequest attribute as defined in [RFC2985]. This attribute contains certificate extensions. The profile for extensions in certificate requests is specified in Section 6.3.

This profile applies the following additional constraint to fields that MAY appear in a CertificationRequest Object:

signatureAlgorithm
The signatureAlgorithm value is specified in [RFC6485].

6.2. CRMF Profile

This profile refines the Certificate Request Message Format (CRMF) specification in [RFC4211], as it relates to resource certificates. A Certificate Request Message object, formatted according to the CRMF, is passed to a CA as the initial step in certificate issuance.

With the exception of the SubjectPublicKeyinfo and the SIA extension request, the CA is permitted to alter any requested field when issuing the certificate.

6.2.1. CRMF Resource Certificate Request Template Fields

This profile applies the following additional requirements to fields that may appear in a Certificate Request Template:

version
This field SHOULD be omitted. If present, it MUST specify a request for a version 3 Certificate.

serialNumber
This field MUST be omitted.

signingAlgorithm
This field MUST be omitted.

issuer
This MUST be omitted in this profile.

Validity
This field MAY be omitted. If omitted, the CA will issue a Certificate with Validity dates as determined by the CA. If specified, then the CA MAY override the requested values with dates as determined by the CA.

Subject
This field MAY be omitted. If present, the value of this field SHOULD be empty (i.e., NULL), in which case the CA MUST generate a subject name that is unique in the context of certificates issued by this CA. This field is allowed to be non-empty only for a re-key/reissuance request, and only if the CA has adopted a policy (in its CPS) that permits the reuse of names in these circumstances.

PublicKey
This field MUST be present.

extensions
The profile for extensions in certificate requests is specified in Section 6.3.

6.2.2. Resource Certificate Request Control Fields

The following control fields are supported in this profile:

Authenticator Control
The intended model of authentication of the subject is a "long term" model, and the guidance offered in [RFC4211] is that the Authenticator Control field be used.

6.3. Certificate Extension Attributes in Certificate Requests

The following extensions MAY appear in a PKCS#10 or CRMF Certificate Request. Any other extensions MUST NOT appear in a Certificate Request. This profile places the following additional constraints on these extensions:

BasicConstraints
: If this is omitted, then the CA will issue an EE certificate (hence no BasicConstraints extension will be included).

  The pathLengthConstraint is not supported in this profile, and this field MUST be omitted.

  The CA MAY honor the cA boolean if set to TRUE (CA Certificate Request). If this bit is set, then it indicates that the subject is requesting a CA certificate.

  The CA MUST honor the cA bit if set to FALSE (EE Certificate Request), in which case the corresponding EE certificate will not contain a Basic Constraints extension.

KeyUsage
: The CA MAY honor KeyUsage extensions of keyCertSign and cRLSign if present, as long as this is consistent with the BasicConstraints SubjectType sub-field, when specified.

ExtendedKeyUsage
: The CA MAY honor ExtendedKeyUsage extensions in requests for EE certificates that are issued to routers or other devices, consistent with values specified in Standards Track RFCs that adopt this profile and that identify application-specific requirements that motivate the use of such EKUs.

SubjectInformationAccess
: This field MUST be present, and the field value SHOULD be honored by the CA if it conforms to the requirements set forth in Section 4.8.8. If the CA is unable to honor the requested value for this field, then the CA MUST reject the Certificate Request.

7. Resource Certificate Validation

This section describes the resource certificate validation procedure. This refines the generic procedure described in Section 6 of [RFC5280].

7.1. Resource Extension Validation

The IP Resources and AS Resources extensions [RFC3779] define critical extensions for INRs. These are ASN.1 encoded representations of the IPv4 and IPv6 address range and an AS number set.

Valid resource certificates MUST have a valid IP address and/or AS number resource extension. In order to validate a resource certificate, the resource extension MUST also be validated. This validation process relies on definitions of comparison of resource sets:

more specific
: Given two contiguous IP address ranges or two contiguous AS number ranges, A and B, A is "more specific" than B if range B includes all IP addresses or AS numbers described by range A, and if range B is larger than range A.

equal
: Given two contiguous IP address ranges or two contiguous AS number ranges, A and B, A is "equal" to B if range A describes precisely the same collection of IP addresses or AS numbers described by range B. The definition of "inheritance" in [RFC3779] is equivalent to this "equality" comparison.

encompass
: Given two IP address and AS number sets, X and Y, X "encompasses" Y if, for every contiguous range of IP addresses or AS numbers elements in set Y, the range element is either "more specific" than or "equal" to a contiguous range element within the set X.

Validation of a certificate's resource extension in the context of a certification path (see Section 7.2 entails that for every adjacent pair of certificates in the certification path (certificates 'x' and 'x + 1'), the number resources described in certificate 'x' "encompass" the number resources described in certificate 'x + 1', and the resources described in the trust anchor information "encompass" the resources described in the first certificate in the certification path.

7.2. Resource Certification Path Validation

Validation of signed resource data using a target resource certificate consists of verifying that the digital signature of the signed resource data is valid, using the public key of the target resource certificate, and also validating the resource certificate in the context of the RPKI, using the path validation process. This path validation process verifies, among other things, that a prospective certification path (a sequence of n certificates) satisfies the following conditions:

1. for all 'x' in {1, ..., n-1}, the subject of certificate 'x' is the issuer of certificate ('x' + 1);
2. certificate '1' is issued by a trust anchor;
3. certificate 'n' is the certificate to be validated; and
4. for all 'x' in {1, ..., n}, certificate 'x' is valid.

Certificate validation entails verifying that all of the following conditions hold, in addition to the certification path validation criteria specified in Section 6 of [RFC5280]:

1. The certificate can be verified using the issuer's public key and the signature algorithm
2. The current time lies within the certificate's Validity From and To values.
3. The certificate contains all fields that MUST be present, as defined by this specification, and contains values for selected fields that are defined as allowable values by this specification.
4. No field, or field value, that this specification defines as MUST NOT be present is used in the certificate.
5. The issuer has not revoked the certificate. A revoked certificate is identified by the certificate's serial number being listed on the issuer's current CRL, as identified by the CRLDP of the certificate, the CRL is itself valid, and the public key used to verify the signature on the CRL is the same public key used to verify the certificate itself.

6. The resource extension data is "encompassed" by the resource extension data contained in a valid certificate where this issuer is the subject (the previous certificate in the context of the ordered sequence defined by the certification path).

7. The certification path originates with a certificate issued by a trust anchor, and there exists a signing chain across the certification path where the subject of Certificate 'x' in the certification path matches the issuer in Certificate 'x + 1' in the certification path, and the public key in Certificate 'x' can verify the signature value in Certificate 'x+1'.

A certificate validation algorithm MAY perform these tests in any chosen order.

Certificates and CRLs used in this process MAY be found in a locally maintained cache, maintained by a regular synchronization across the distributed publication repository structure [RFC6481].

There exists the possibility of encountering certificate paths that are arbitrarily long, or attempting to generate paths with loops as means of creating a potential denial-of-service (DOS) attack on an RP. An RP executing this procedure MAY apply further heuristics to guide the certification path validation process to a halt in order to avoid some of the issues associated with attempts to validate such malformed certification path structures. Implementations of resource certificate validation MAY halt with a validation failure if the certification path length exceeds a locally defined configuration parameter.

8. Design Notes

The following notes provide some additional commentary on the considerations that lie behind some of the design choices that were made in the design of this certificate profile. These notes are non-normative, i.e., this section of the document does not constitute a formal part of the profile specification, and the interpretation of key words as defined in RFC 2119 are not applicable in this section of the document.

Certificate Extensions:

This profile does not permit the use of any other critical or non-critical extensions. The rationale for this restriction is that the resource certificate profile is intended for a specific defined use. In this context, having certificates with additional non-critical extensions that RPs may see as valid certificates without understanding the extensions is inappropriate, because if the RP were in a position to

understand the extensions, it would contradict or qualify this original judgment of validity in some way. This profile takes the position of minimalism over extensibility. The specific goal for the associated RPKI is to precisely match the INR allocation structure through an aligned certificate structure that describes the allocation and its context within the INR distribution hierarchy. The profile defines a resource certificate that is structured to meet these requirements.

Certification Authorities and Key Values:
: This profile uses a definition of an instance of a CA as a combination of a named entity and a key pair. Within this definition, a CA instance cannot rollover a key pair. However, the entity can generate a new instance of a CA with a new key pair and roll over all the signed subordinate products to the new CA [RFC6489].

  This has a number of implications in terms of subject name management, CRL Scope, and repository publication point management.

CRL Scope and Key Values:
: For CRL Scope, this profile specifies that a CA issues a single CRL at a time, and the scope of the CRL is all certificates issued by this CA. Because the CA instance is bound to a single key pair, this implies that the CA's public key, the key used to validate the CA's CRL, and the key used to validate the certificates revoked by that CRL are all the same key value.

Repository Publication Point:
: The definition of a CA affects the design of the repository publication system. In order to minimize the amount of forced re-certification on key rollover events, a repository publication regime that uses the same repository publication point for all CA instances that refers to the same entity, but with different key values, will minimize the extent of re-generation of certificates to only immediate subordinate certificates. This is described in [RFC6489].

Subject Name:
: This profile specifies that subject names must be unique per issuer, and does not specify that subject names must be globally unique (in terms of assured uniqueness). This is due to the nature of the RPKI as a distributed PKI, implying that there is no ready ability for certification authorities to coordinate a simple RPKI-wide unique name space without resorting to additional critical external dependencies. CAs

are advised to use subject name generation procedures that minimize the potential for name clashes.

One way to achieve this is for a CA to use a subject name practice that uses the CommonName component of the Distinguished Name as a constant value for any given entity that is the subject of CA-issued certificates, and set the serialNumber component of the Distinguished Name to a value that is derived from the hash of the subject public key value.

If the CA elects not to use the serialNumber component of the DistinguishedName, then it is considered beneficial that a CA generates CommonNames that have themselves a random component that includes significantly more than 40 bits of entropy in the name. Some non-normative recommendations to achieve this include:

1) Hash of the subject public key (encoded as ASCII HEX).
   example: cn="999d99d564de366a29cd8468c45ede1848e2cc14"

2) A Universally Unique IDentifier (UUID) [RFC4122]
   example: cn="6437d442-6fb5-49ba-bbdb-19c260652098"

3) A randomly generated ASCII HEX encoded string of length 20 or greater:
   example: cn="0f8fcc28e3be4869bc5f8fa114db05e1">
   (A string of 20 ASCII HEX digits would have 80-bits of entropy)

4) An internal database key or subscriber ID combined with one of the above
   example: cn="<DBkey1> (6437d442-6fb5-49ba-bbdb-19c2606520980)"
   (The issuing CA may wish to be able to extract the database key or subscriber ID from the commonName. Since only the issuing CA would need to be able to parse the commonName, the database key and the source of entropy (e.g., a UUID) could be separated in any way that the CA wants, as long as it conforms to the rules for PrintableString. The separator could be a space character, parenthesis, hyphen, slash, question mark, etc.

9. Operational Considerations for Profile Agility

This profile requires that relying parties reject certificates or CRLs that do not conform to the profile. (Through the remainder of this section, the term "certificate" is used to refer to both certificates and CRLs.) This includes certificates that contain

extensions that are prohibited, but that are otherwise valid as per [RFC5280]. This means that any change in the profile (e.g., extensions, permitted attributes or optional fields, or field encodings) for certificates used in the RPKI will not be backward compatible. In a general PKI context, this constraint probably would cause serious problems. In the RPKI, several factors minimize the difficulty of effecting changes of this sort.

Note that the RPKI is unique in that every relying party (RP) requires access to every certificate issued by the CAs in this system. An important update of the certificates used in the RPKI must be supported by all CAs and RPs in the system, lest views of the RPKI data differ across RPs. Thus, incremental changes require very careful coordination. It would not be appropriate to introduce a new extension, or authorize use of an extant, standard extension, for a security-relevant purpose on a piecemeal basis.

One might imagine that the "critical" flag in X.509 certificate extensions could be used to ameliorate this problem. However, this solution is not comprehensive and does not address the problem of adding a new, security-critical extension. (This is because such an extension needs to be supported universally, by all CAs and RPs.) Also, while some standard extensions can be marked either critical or non-critical, at the discretion of the issuer, not all have this property, i.e., some standard extensions are always non-critical. Moreover, there is no notion of criticality for attributes within a name or optional fields within a field or an extension. Thus, the critical flag is not a solution to this problem.

In typical PKI deployments, there are few CAs and many RPs. However, in the RPKI, essentially every CA in the RPKI is also an RP. Thus the set of entities that will need to change in order to issue certificates under a new format is the same set of entities that will need to change to accept these new certificates. To the extent that this is literally true, it says that CA/RP coordination for a change is tightly linked anyway. In reality, there is an important exception to this general observation. Small ISPs and holders of provider-independent allocations are expected to use managed CA services, offered by Regional Internet Registries (RIRs) and potentially by wholesale Internet Service Providers (ISPs). This reduces the number of distinct CA implementations that are needed and makes it easier to effect changes for certificate issuance. It seems very likely that these entities also will make use of RP software provided by their managed CA service provider, which reduces the number of distinct RP software implementations. Also note that many small ISPs (and holders of provider-independent allocations) employ default routes, and thus need not perform RP validation of RPKI data, eliminating these entities as RPs.

Widely available PKI RP software does not cache large numbers of certificates, an essential strategy for the RPKI. It does not process manifest or ROA data structures, essential elements of the RPKI repository system. Experience shows that such software deals poorly with revocation status data. Thus, extant RP software is not adequate for the RPKI, although some open source tools (e.g., OpenSSL and cryptlib) can be used as building blocks for an RPKI RP implementation. Thus, it is anticipated that RPs will make use of software that is designed specifically for the RPKI environment and is available from a limited number of open sources. Several RIRs and two companies are providing such software today. Thus it is feasible to coordinate change to this software among the small number of developers/maintainers.

If the resource certificate profile is changed in the future, e.g., by adding a new extension or changing the allowed set of name attributes or encoding of these attributes, the following procedure will be employed to effect deployment in the RPKI. The model is analogous to that described in [RPKI-ALG], but is simpler.

A new document will be issued as an update to this RFC. The CP for the RPKI [RFC6484] will be updated to reference the new certificate profile. The new CP will define a new policy OID for certificates issued under the new certificate profile. The updated CP also will define a timeline for transition to the new certificate (CRL) format. This timeline will define 3 phases and associated dates:

1. At the end of phase 1, all RPKI CAs MUST be capable of issuing certificates under the new profile, if requested by a subject. Any certificate issued under the new format will contain the new policy OID.

2. During phase 2, CAs MUST issue certificates under the new profile, and these certificates MUST coexist with certificates issued under the old format. (CAs will continue to issue certificates under the old OID/format as well.) The old and new certificates MUST be identical, except for the policy OID and any new extensions, encodings, etc. The new certificates, and associated signed objects, will coexist in the RPKI repository system during this phase, analogous to what is required by an algorithm transition for the RPKI [RPKI-ALG]. Relying parties MAY make use of the old or the new certificate formats when processing signed objects retrieved from the RPKI repository system. During this phase, a relying party that elects to process both formats will acquire the same values for all certificate fields that overlap between the old and

new formats. Thus if either certificate format is verifiable, the relying party accepts the data from that certificate. This allows CAs to issue certificates under the new format before all relying parties are prepared to process that format.

3. At the beginning of phase 3, all relying parties MUST be capable of processing certificates under the new format. During this phase, CAs will issue new certificates ONLY under the new format. Certificates issued under the old OID will be replaced with certificates containing the new policy OID. The repository system will no longer require matching old and new certificates under the different formats.

At the end of phase 3, all certificates under the old OID will have been replaced. The resource certificate profile RFC will be replaced to remove support for the old certificate format, and the CP will be replaced to remove reference to the old policy OID and to the old resource certificate profile RFC. The system will have returned to a new, steady state.

10. Security Considerations

The Security Considerations of [RFC5280] and [RFC3779] apply to resource certificates. The Security Considerations of [RFC2986] and [RFC4211] apply to resource certificate certification requests.

A resource certificate PKI cannot in and of itself resolve any forms of ambiguity relating to uniqueness of assertions of rights of use in the event that two or more valid certificates encompass the same resource. If the issuance of resource certificates is aligned to the status of resource allocations and assignments, then the information conveyed in a certificate is no better than the information in the allocation and assignment databases.

This profile requires that the key used to sign an issued certificate be the same key used to sign the CRL that can revoke the certificate, implying that the certification path used to validate the signature on a certificate is the same as that used to validate the signature of the CRL that can revoke the certificate. It is noted that this is a tighter constraint than required in X.509 PKIs, and there may be a risk in using a path validation implementation that is capable of using separate validation paths for a certificate and the corresponding CRL. If there are subject name collisions in the RPKI as a result of CAs not following the guidelines provided here relating to ensuring sufficient entropy in constructing subject names, and this is combined with the situation that an RP uses an implementation of validation path construction that is not in

conformance with this RPKI profile, then it is possible that the subject name collisions can cause an RP to conclude that an otherwise valid certificate has been revoked.

11. IANA Considerations

This document has no actions for IANA.

12. Acknowledgements

The authors would like to particularly acknowledge the valued contribution from Stephen Kent in reviewing this document and proposing numerous sections of text that have been incorporated into the document. The authors also acknowledge the contributions of Sandy Murphy, Robert Kisteleki, Randy Bush, Russ Housley, Ricardo Patara, and Rob Austein in the preparation and subsequent review of this document. The document also reflects review comments received from Roque Gagliano, Sean Turner, and David Cooper.

13. References

13.1. Normative References

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

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

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

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[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, May 2008.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, February 2010.

[RFC6484] Kent, S., Kong, D., Seo, K., and R. Watro, "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, February 2012.

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

[X.509] ITU-T, , "Recommendation X.509: The Directory - Authentication Framework", 2000.

[X.680] ITU-T, , "Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", 2002.

13.2. Informative References

[RFC2985] Nystrom, M. and B. Kaliski, "PKCS #9: Selected Object Classes and Attribute Types Version 2.0", RFC 2985, November 2000.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4514] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, February 2012.

[RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, February 2012.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, February 2012.

[RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, February 2012.

[RFC7318] Newton, A. and G. Huston, "Policy Qualifiers in Resource Public Key Infrastructure (RPKI) Certificates", RFC 7318, July 2014.

[RPKI-ALG] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for RPKI", Work in Progress, November 2011.

Appendix A. Example Resource Certificate

The following is an example resource certificate.

Certificate Name: 9JfgAEcq7Q-47IwMC5CJIJr6EJs.cer

```
Data:
  Version: 3 (0x2)
  Serial: 1500 (0x5dc)
  Signature Algorithm: SHA256WithRSAEncryption
  Issuer: CN=APNIC Production-CVPQSgUkLy7pOXdNeVWGvnFX_0s
  Validity
   Not Before: Oct 25 12:50:00 2008 GMT
    Not After : Jan 31 00:00:00 2010 GMT
  Subject: CN=A91872ED
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
    Modulus (2048 bit):
      00:bb:fb:4a:af:a4:b9:dc:d0:fa:6f:67:cc:27:39:
      34:d1:80:40:37:de:88:d1:64:a2:f1:b3:fa:c6:7f:
      bb:51:df:e1:c7:13:92:c3:c8:a2:aa:8c:d1:11:b3:
      aa:99:c0:ac:54:d3:65:83:c6:13:bf:0d:9f:33:2d:
      39:9f:ab:5f:cd:a3:e9:a1:fb:80:7d:1d:d0:2b:48:
      a5:55:e6:24:1f:06:41:35:1d:00:da:1f:99:85:13:
      26:39:24:c5:9a:81:15:98:fb:5f:f9:84:38:e5:d6:
      70:ce:5a:02:ca:dd:61:85:b3:43:2d:0b:35:d5:91:
      98:9d:da:1e:0f:c2:f6:97:b7:97:3e:e6:fc:c1:c4:
      3f:30:c4:81:03:25:99:09:4c:e2:4a:85:e7:46:4b:
      60:63:02:43:46:51:4d:ed:fd:a1:06:84:f1:4e:98:
      32:da:27:ee:80:82:d4:6b:cf:31:ea:21:af:6f:bd:
      70:34:e9:3f:d7:e4:24:cd:b8:e0:0f:8e:80:eb:11:
      1f:bc:c5:7e:05:8e:5c:7b:96:26:f8:2c:17:30:7d:
      08:9e:a4:72:66:f5:ca:23:2b:f2:ce:54:ec:4d:d9:
      d9:81:72:80:19:95:57:da:91:00:d9:b1:e8:8c:33:
      4a:9d:3c:4a:94:bf:74:4c:30:72:9b:1e:f5:8b:00:
      4d:e3
    Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Subject Key Identifier:
      F4:97:E0:00:47:2A:ED:0F:B8:EC:8C:0C:0B:90:89:
      20:9A:FA:10:9B

    X509v3 Authority Key Identifier:
      keyid:09:53:D0:4A:05:24:2F:2E:E9:39:77:4D:79:
      55:86:BE:71:57:FF:4B

    X509v3 Key Usage: critical
```

```
        Certificate Sign, CRL Sign

      X509v3 Basic Constraints: critical
        CA:TRUE

      X509v3 CRL Distribution Points:
        URI:rsync://rpki.apnic.net/repository/A3C38A24
            D60311DCAB08F31979BDBE39/CVPQSgUkLy7pOXdNe
            VWGvnFX_0s.crl

      Authority Information Access:
         CA Issuers - URI:rsync://rpki.apnic.net/repos
            itory/8BDFC7DED5FD11DCB14CF4B1A703F9B7/CVP
            QSgUkLy7pOXdNeVWGvnFX_0s.cer

      X509v3 Certificate Policies: critical
         Policy: 1.3.6.1.5.5.7.14.2

      Subject Information Access:
         CA Repository - URI:rsync://rpki.apnic.net/mem
             ber_repository/A91872ED/06A83982887911DD81
             3F432B2086D636/
         Manifest - URI:rsync://rpki.apnic.net/member_r
             epository/A91872ED/06A83982887911DD813F432
             B2086D636/9JfgAEcq7Q-47IwMC5CJIJr6EJs.mft

       sbgp-autonomousSysNum: critical
         Autonomous System Numbers:
           24021
           38610
           131072
           131074

       sbgp-ipAddrBlock: critical
         IPv4:
           203.133.248.0/22
           203.147.108.0/23

   Signature Algorithm: sha256WithRSAEncryption
       51:4c:77:e4:21:64:80:e9:35:30:20:9f:d8:4b:88:60:b8:1f:
       73:24:9d:b5:17:60:65:6a:28:cc:43:4b:68:97:ca:76:07:eb:
       dc:bd:a2:08:3c:8c:56:38:c6:0a:1e:a8:af:f5:b9:42:02:6b:
       77:e0:b1:1c:4a:88:e6:6f:b6:17:d3:59:41:d7:a0:62:86:59:
       29:79:26:76:34:d1:16:2d:75:05:cb:b2:99:bf:ca:c6:68:1b:
       b6:a9:b0:f4:43:2e:df:e3:7f:3c:b3:72:1a:99:fa:5d:94:a1:
       eb:57:9c:9a:2c:87:d6:40:32:c9:ff:a6:54:b8:91:87:fd:90:
       55:ef:12:3e:1e:2e:cf:c5:ea:c3:4c:09:62:4f:88:00:a0:7f:
       cd:67:83:bc:27:e1:74:2c:18:4e:3f:12:1d:ef:29:0f:e3:27:
```

```
00:ce:14:eb:f0:01:f0:36:25:a2:33:a8:c6:2f:31:18:22:30:
cf:ca:97:43:ed:84:75:53:ab:b7:6c:75:f7:2f:55:5c:2e:82:
0a:be:91:59:bf:c9:06:ef:bb:b4:a2:71:9e:03:b1:25:8e:29:
7a:30:88:66:b4:f2:16:6e:df:ad:78:ff:d3:b2:9c:29:48:e3:
be:87:5c:fc:20:2b:df:da:ca:30:58:c3:04:c9:63:72:48:8c:
0a:5f:97:71
```

Appendix B. Example Certificate Revocation List

The following is an example Certificate Revocation List.

```
   CRL Name: q66IrWSGuBE7jqx8PAUHAlHCqRw.crl
   Data:
     Version: 2
     Signature Algorithm:
       Hash: SHA256, Encryption: RSA
     Issuer: CN=Demo Production APNIC CA - Not for real use,
       E=ca@apnic.net
     This Update: Thu Jul 27 06:30:34 2006 GMT
     Next Update: Fri Jul 28 06:30:34 2006 GMT
     Authority Key Identifier: Key Identifier:
       ab:ae:88:ad:64:86:b8:11:3b:8e:ac:7c:3c:05:
       07:02:51:c2:a9:1c
     CRLNumber: 4
     Revoked Certificates: 1
       Serial Number: 1
       Revocation Date: Mon Jul 17 05:10:19 2006 GMT
       Serial Number: 2
       Revocation Date: Mon Jul 17 05:12:25 2006 GMT
       Serial Number: 4
       Revocation Date: Mon Jul 17 05:40:39 2006 GMT
     Signature:
       b2:5a:e8:7c:bd:a8:00:0f:03:1a:17:fd:40:2c:46:
       0e:d5:64:87:e7:e7:bc:10:7d:b6:3e:39:21:a9:12:
       f4:5a:d8:b8:d4:bd:57:1a:7d:2f:7c:0d:c6:4f:27:
       17:c8:0e:ae:8c:89:ff:00:f7:81:97:c3:a1:6a:0a:
       f7:d2:46:06:9a:d1:d5:4d:78:e1:b7:b0:58:4d:09:
       d6:7c:1e:a0:40:af:86:5d:8c:c9:48:f6:e6:20:2e:
       b9:b6:81:03:0b:51:ac:23:db:9f:c1:8e:d6:94:54:
       66:a5:68:52:ee:dd:0f:10:5d:21:b8:b8:19:ff:29:
       6f:51:2e:c8:74:5c:2a:d2:c5:fa:99:eb:c5:c2:a2:
       d0:96:fc:54:b3:ba:80:4b:92:7f:85:54:76:c9:12:
       cb:32:ea:1d:12:7b:f8:f9:a2:5c:a1:b1:06:8e:d8:
       c5:42:61:00:8c:f6:33:11:29:df:6e:b2:cc:c3:7c:
       d3:f3:0c:8d:5c:49:a5:fb:49:fd:e7:c4:73:68:0a:
       09:0e:6d:68:a9:06:52:3a:36:4f:19:47:83:59:da:
       02:5b:2a:d0:8a:7a:33:0a:d5:ce:be:b5:a2:7d:8d:
       59:a1:9d:ee:60:ce:77:3d:e1:86:9a:84:93:90:9f:
       34:a7:02:40:59:3a:a5:d1:18:fb:6f:fc:af:d4:02:
       d9
```

Appendix C. Differences from RFC 6487

The following changes were made since [RFC6487]:

- The changes from all verified errata have been incorporated.
- The changes from [RFC7318] have been incorporated.

- Section 4.8 was corrected to say that certificate extensions not mentioned in this document are rejected. The text now agrees with the text in Section 1 and Section 8.

- Section 5 now specifies the format of the Authority Key Identifier extension.

Authors' Addresses

Geoff Huston
APNIC

EMail: gih@apnic.net
URI: http://www.apnic.net

George Michaelson
APNIC

EMail: ggm@apnic.net
URI: http://www.apnic.net

Robert Loomans
Suncorp

EMail: Robert.Loomans@suncorp.com.au
URI: http://www.suncorpgroup.com.au/

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
USA

EMail: andy@arin.net
URI: http://www.arin.net

Richard Hansen
BBN Technologies
10 Moulton St
Cambridge, MA 02138-1119
USA

EMail: rhansen@bbn.com