

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2013

M. Lepinski, Ed.
BBN
February 25, 2013

BGPSEC Protocol Specification
draft-ietf-sidr-bgpsec-protocol-07

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 April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. BGPSEC Negotiation	3
2.1. BGPSEC Send Capability	3
2.2. BGPSEC Receive Capability	4
2.3. Negotiating BGPSEC Support	5
3. The BGPSEC_Path Attribute	6
3.1. Secure_Path	8
3.2. Signature_Block	9
4. Generating a BGPSEC Update	11
4.1. Originating a New BGPSEC Update	12
4.2. Propagating a Route Advertisement	14
4.3. Processing Instructions for Confederation Members	18
4.4. Reconstructing the AS_PATH Attribute	20
5. Processing a Received BGPSEC Update	21
5.1. Overview of BGPSEC Validation	23
5.2. Validation Algorithm	24
6. Algorithms and Extensibility	28
6.1. Algorithm Suite Considerations	28
6.2. Extensibility Considerations	28
7. Security Considerations	29
8. IANA Considerations	32
9. Contributors	32
9.1. Authors	32
9.2. Acknowledgements	34
10. Normative References	34
11. Informative References	35
Author's Address	35

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 two properties:

1. The route was originated by an AS that has been explicitly authorized by the holder of the IP address prefix to originate route advertisements for that prefix.
2. Every AS on the path of ASes through which the update message passes 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 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 BGP update messages to external peers (eBGP) containing the BGPSEC_Path needs to have the 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 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).

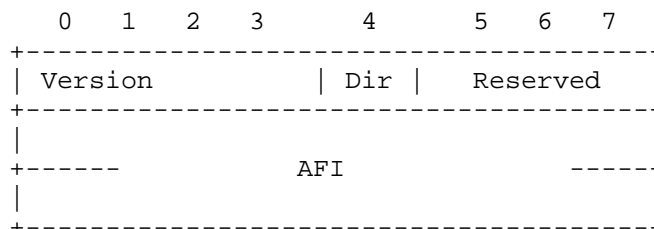
2.1. The BGPSEC Capability

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 message 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:

- o The given peer has sent the BGPSEC capability for a particular version of BGPSEC and a particular address family with the Direction bit set to 1; and
- o The other peer has 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. If there exist multiple versions have BGPSEC that are negotiated for a particular session, the behavior of the peers (e.g., which version of BGPSEC shall actually be used) will be specified in a future document.

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].

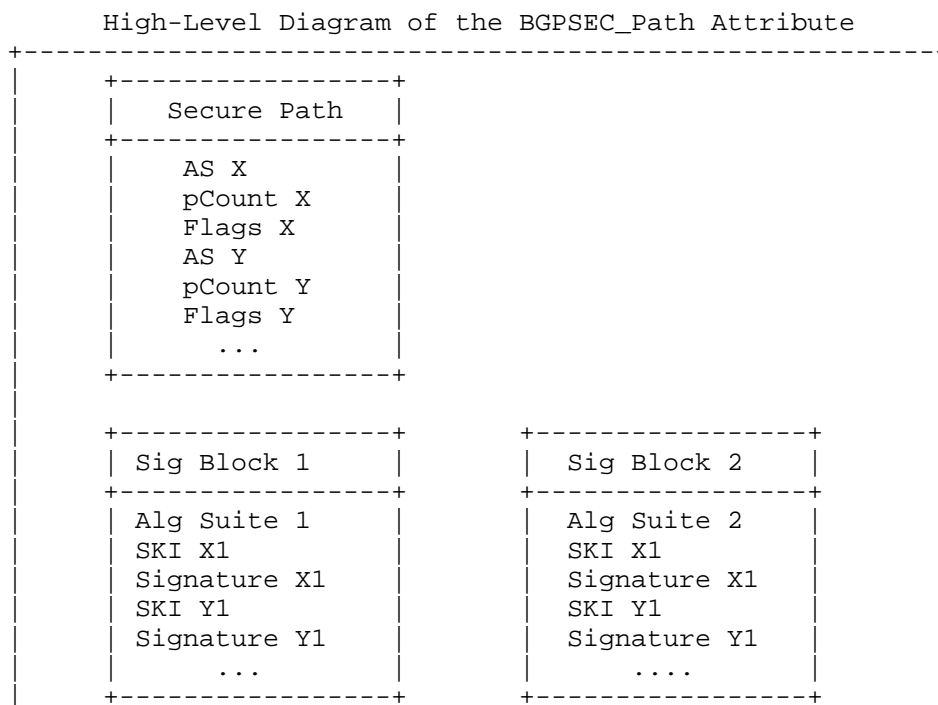
3. The BGPSEC_Path Attribute

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 this 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:



|
+-----+

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

BGPSEC_Path Attribute

Secure_Path	(variable)	
Sequence of one or two Signature_Blocks	(variable)	

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. A BGPSEC update message containing the BGPSEC_PATH attribute MUST NOT contain the AS_PATH attribute. The 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 one 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

Secure_Path Length	(2 octets)	
One or More Secure_Path Segments	(variable)	

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

AS Number	(4 octets)	
pCount	(1 octet)	
Flags	(1 octet)	

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 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

Signature_Block Length	(2 octets)
Algorithm Suite Identifier	(1 octet)
Sequence of Signature Segments	(variable)

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 created 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

Subject Key Identifier	(20 octets)
Signature Length	(2 octets)
Signature	(variable)

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 creates a new BGPSEC_Path attribute with zero Secure_Path segments and zero Signature Segments. 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.

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]). Note that validation of a BGPSEC update message will fail (i.e., the validation algorithm, specified in Section 5.2, returns 'Not Valid') unless there exists a valid ROA authorizing the first AS in the Secure_Path portion of the BGPSEC_Path attribute to originate routes to the prefix being advertised. Therefore, a BGPSEC speaker **SHOULD NOT** originate a BGPSEC update advertising a route for a given prefix unless 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.

If the BGPSEC speaker is not a member of an autonomous system confederation [5], then the Flags field of the Secure_Path Segment MUST be set to zero. (Members of a confederation should follow the special processing instructions for confederation members in Section 4.4.)

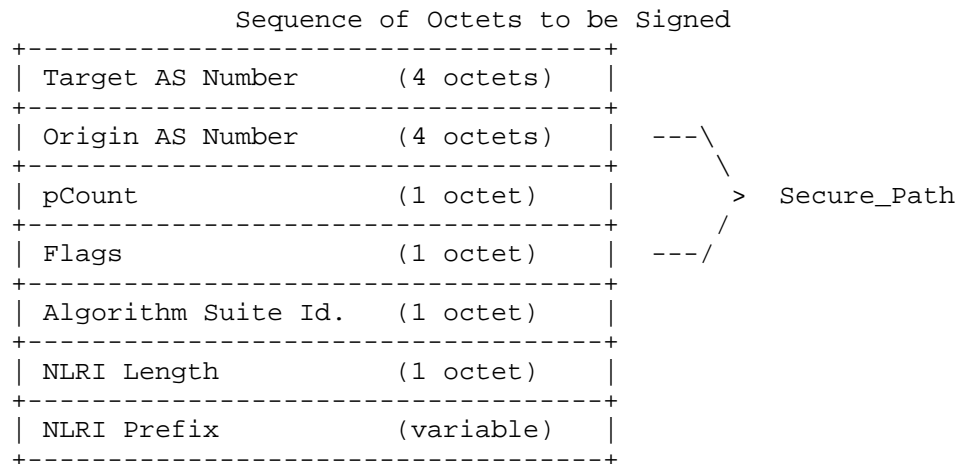
Typically, a BGPSEC speaker will use only a single algorithm suite, and thus create only a single Signature_Block in the BGPSEC_Path attribute. However, to ensure backwards compatibility during a period of transition from a 'current' algorithm suite to a 'new' algorithm suite, it will be necessary to 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 corresponding to the BGPSEC speaker. The digital signature is computed as follows:

- o 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 and if it chooses to propagate that peer's route for the prefix, then it MUST NOT attach any BGPSEC_Path attribute to the corresponding update being propagated. (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. However, the BGPSEC speaker MAY propagate the route as a (unsigned) BGP update message without 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 standard (non-BGPSEC) update message. It should be noted that BCP 172 [12] recommends against the use of AS_SET and AS_CONFED_SET in AS_PATH in 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. Note that the option of setting pCount to 0 is intended only for use by route servers that desire not to increase the effective AS-PATH length of routes they advertise. The pCount field SHOULD NOT be set to 0 in other circumstances. 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 (i.e., cases where the peer is not acting as a route server).

If the BGPSEC speaker is not a member of an autonomous system confederation [5], then the Confed_Segment bit of the Flags field of the Secure_Path Segment MUST be set to zero. (Members of a confederation should follow the special processing instructions for confederation members in Section 4.3.)

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 there are two Signature_Blocks (corresponding to different algorithm suites) that 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 then 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 appears in the same order as the corresponding Secure_Path segments in the Secure_Path portion of the BGPSEC_Path attribute. 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 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:

- o 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.

Sequence of Octets to be Signed

+-----+-----+-----+		
Target AS Number	(4 octets)	
+-----+-----+-----+		
Signer's AS Number	(4 octets)	
+-----+-----+-----+		
pCount	(1 octet)	
+-----+-----+-----+		
Flags	(1 octet)	
+-----+-----+-----+		
Most Recent Sig Field	(variable)	
+-----+-----+-----+		

---\

> Secure_Path

/

---/

- 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.

4.3. 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. Note that this means that in a BGPSEC update message, an AS number appears in a Secure_Path Segment with the Confed_Segment flag set to one, in precisely those circumstances where 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 to have its members not 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 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:

- o First, starting with the most recently added Secure_Path segments, 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.

- o 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.)
- o 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 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 that for a signature produced by a 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, when processing a Signature_Segment, the confederation member 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 for a BGPSEC speaker to receive a BGPSEC update messages containing a Secure_Path segment with the Confed_Sequence flag set to one from a peer who is not a member of the same AS confederation. (Such an error is treated in exactly the same way as receipt of a non-BGPSEC update message containing an AS_CONFED_SEQUENCE from a peer that is not a member of the same AS confederation.)

4.4. Reconstructing the AS_PATH Attribute

BGPSEC update messages do not contain the AS_PATH attribute. Note, however, that 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. 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. 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 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. Note that 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:

- o For each valid RPKI router certificate containing an AS Number extension, the AS Number, Public Key and Subject Key Identifier are required,
- o 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 [15] for the RTR protocol [14].)

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 shall be handled using local policy mechanisms. It is expected that BGP peers will generally prefer routes received via 'Valid' BGPSEC update messages over routes received via 'Not Valid' BGPSEC update messages as well as routes received via update messages that do not contain the BGPSEC_Path attribute. However, BGPSEC specifies no changes to the BGP decision process. (See [16] for related operational considerations.)

BGPSEC validation needs only be performed at 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 identify an error in the BGPSEC_Path attribute, then the implementation should notify the operator that an error has occurred and treat the update in a manner consistent with other BGP errors (i.e., following RFC 4271[2] or any future updates to that document).

Next, the BGPSEC speaker verifies that the origin AS is authorized to advertise the prefix in question. To do this, consult the valid ROA data to obtain a list of AS numbers that are associated with the given IP address prefix in the update message. Then locate the last (least recently added) AS number in the Secure_Path portion of the

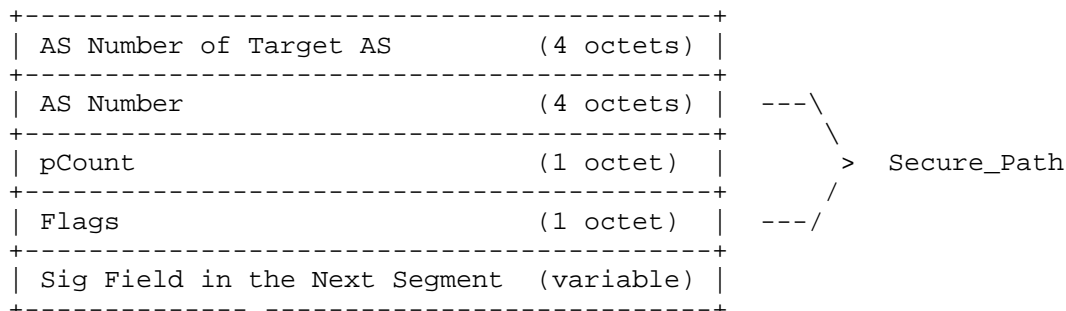
BGPSEC_Path attribute. If the origin AS in the Secure_Path is not in the set of AS numbers associated with the given prefix, then the BGPSEC update message is 'Not Valid' and the validation algorithm terminates.

Finally, 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 does not exist a 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.

- o (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.
- o (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

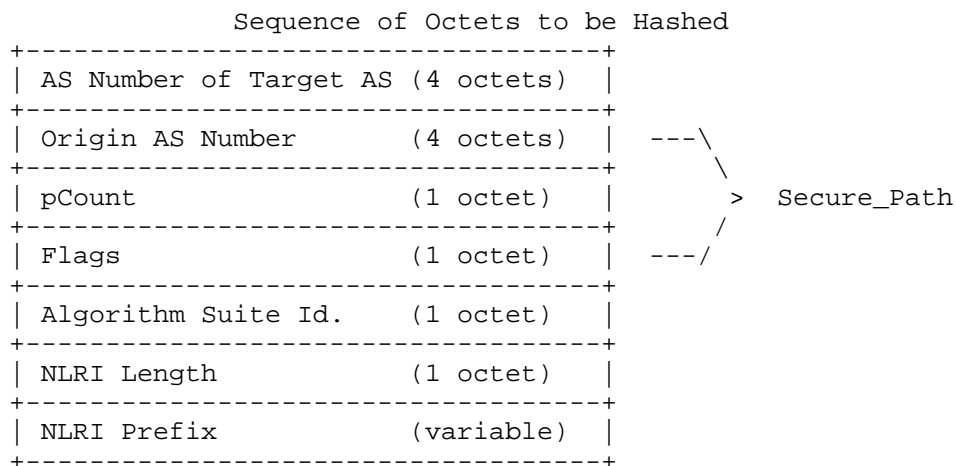


For the first segment to be processed (the most recently added segment), the 'AS Number of Target AS' is the AS number of the BGPSEC speaker validating the update message. Note that if a BGPSEC speaker uses multiple AS Numbers (e.g., the BGPSEC speaker is a member of a confederation), the AS number used 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.)

- o (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 between BGPSEC peers to use of a particular (digest and signature) algorithm suite using BGP capabilities. 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 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 discussion of the BGPSEC threat model and related security considerations, please see [13].

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 propagate 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).

Note that 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 if the BGPSEC speaker propagates the route advertisement received in such an update message then the BGPSEC speaker **SHOULD** add its signature to each of the Signature_Blocks using both the corresponding algorithm suite. 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 quite likely) 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 different 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 route to a given prefix a route obtained via a 'Not Valid' BGPSEC update message. (That is, a BGPSEC update containing only 'Not Valid' Signature_Blocks.) 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 may also be noted here that due to possible differences in RPKI data at different vantage points in the network, a BGPSEC update that was deemed 'Not Valid' at an upstream BGPSEC speaker may indeed be deemed 'Valid' at another BGP speaker downstream.

Therefore, it is important to note that 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:

- o The BGPSEC speaker received the given route advertisement with the indicated NLRI and Secure_Path; and
- o The BGPSEC speaker chose to propagate an advertisement for this route to the peer (implicitly) indicated by the 'Target AS'

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

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 in which an upstream entity that is two or more hops away set pCount to zero is unable to verify for themselves whether pCount was set to zero legitimately.

Finally, BGPSEC does not provide protection against attacks at the transport layer. 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, or injecting BGPSEC update messages with BGPSEC_Path_Signature attributes that fail validation, or causing the peer to tear-down the BGP session. Therefore, 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
lepinski@bbn.com

Stephen Kent
BBN Technologies
kent@bbn.com

Warren Kumari
Google
warren@kumari.net

Doug Montgomery
USA National Institute of Standards and Technology
dougmon@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 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", July 2012.
- [10] Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPSEC Router Certificates, Certificate Revocation Lists, and Certification Requests", April 2012.
- [11] Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", March 2012.

11. Informative References

- [12] Kumari, W. and K. Sriram, "Recommendation for Not Using AS_SET

and AS_CONFED_SET in BGP", RFC 6472, December 2011.

- [13] Kent, S., "Threat Model for BGP Path Security", February 2012.
- [14] Bush, R. and R. Austein, "The RPKI/Router Protocol", February 2012.
- [15] Bush, R., Patel, K., and S. Turner, "Router Key PDU for RPKI-Router Protocol", October 2012.
- [16] Bush, R., "BGPsec Operational Considerations", May 2012.

Author's Address

Matthew Lepinski (editor)
BBN
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

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. BGPsec Negotiation	3
2.1. The BGPsec Capability	4
2.2. Negotiating BGPsec Support	5
3. The BGPsec_Path Attribute	6
3.1. Secure_Path	8
3.2. Signature_Block	10
4. BGPsec Update Messages	11
4.1. General Guidance	11
4.2. Constructing the BGPsec_Path Attribute	14
4.3. Processing Instructions for Confederation Members	18
4.4. Reconstructing the AS_PATH Attribute	19
5. Processing a Received BGPsec Update	21
5.1. Overview of BGPsec Validation	22
5.2. Validation Algorithm	23
6. Algorithms and Extensibility	27
6.1. Algorithm Suite Considerations	27
6.2. Considerations for the SKI Size	28
6.3. Extensibility Considerations	28
7. Operations and Management Considerations	29
7.1. Capability Negotiation Failure	29
7.2. Preventing Misuse of pCount=0	29
7.3. Early Termination of Signature Verification	30
7.4. Non-Deterministic Signature Algorithms	30
7.5. Private AS Numbers	30
7.6. Robustness Considerations for Accessing RPKI Data	32
7.7. Graceful Restart	32
7.8. Robustness of Secret Random Number in ECDSA	32
7.9. Incremental/Partial Deployment Considerations	33
8. Security Considerations	33
8.1. Security Guarantees	33
8.2. On the Removal of BGPsec Signatures	34
8.3. Mitigation of Denial of Service Attacks	35
8.4. Additional Security Considerations	36
9. IANA Considerations	38
10. Contributors	39
10.1. Authors	39
10.2. Acknowledgements	40
11. References	40
11.1. Normative References	40

11.2. Informative References	42
Authors' Addresses	44

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.

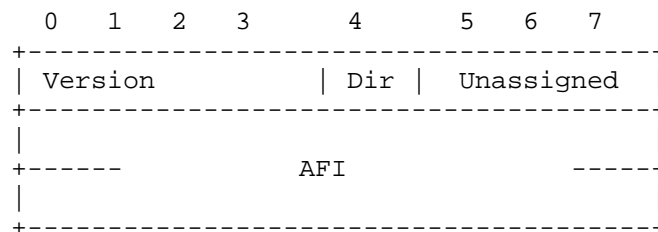


Figure 1: BGPsec Capability format.

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:

- o 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
- o 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.

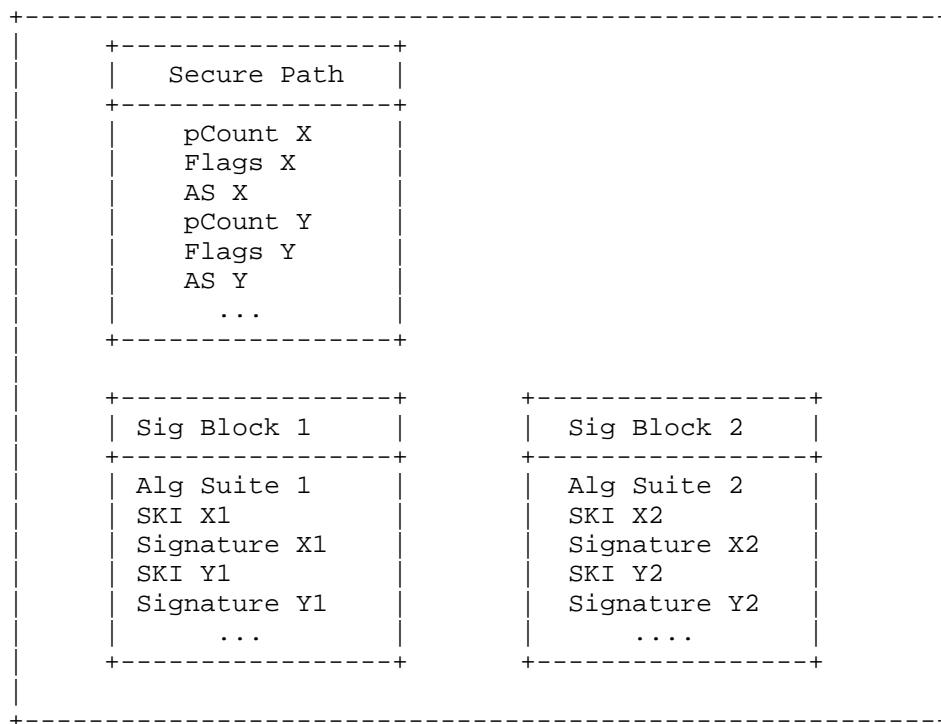


Figure 2: High-level diagram of the BGPsec_Path attribute.

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

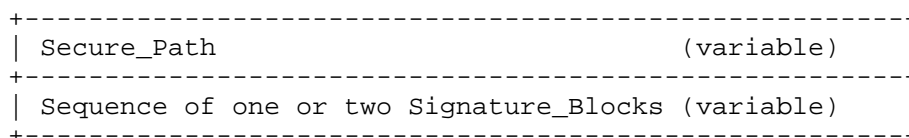


Figure 3: BGPsec_Path attribute format.

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.

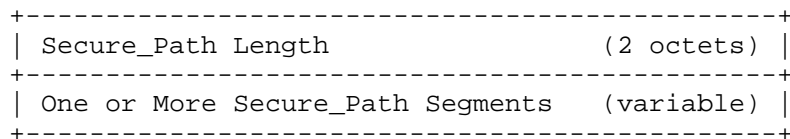


Figure 4: Secure_Path format.

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.)

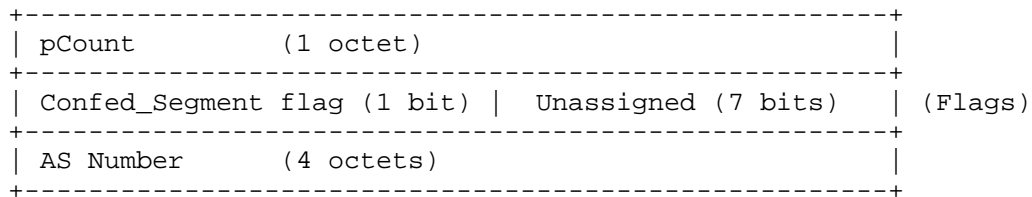


Figure 5: Secure_Path Segment format.

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.

	Signature_Block Length	(2 octets)	
	Algorithm Suite Identifier	(1 octet)	
	Sequence of Signature Segments	(variable)	

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.)

	Subject Key Identifier (SKI)	(20 octets)	
	Signature Length	(2 octets)	
	Signature	(variable)	

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:

- o 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.
- o 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].)

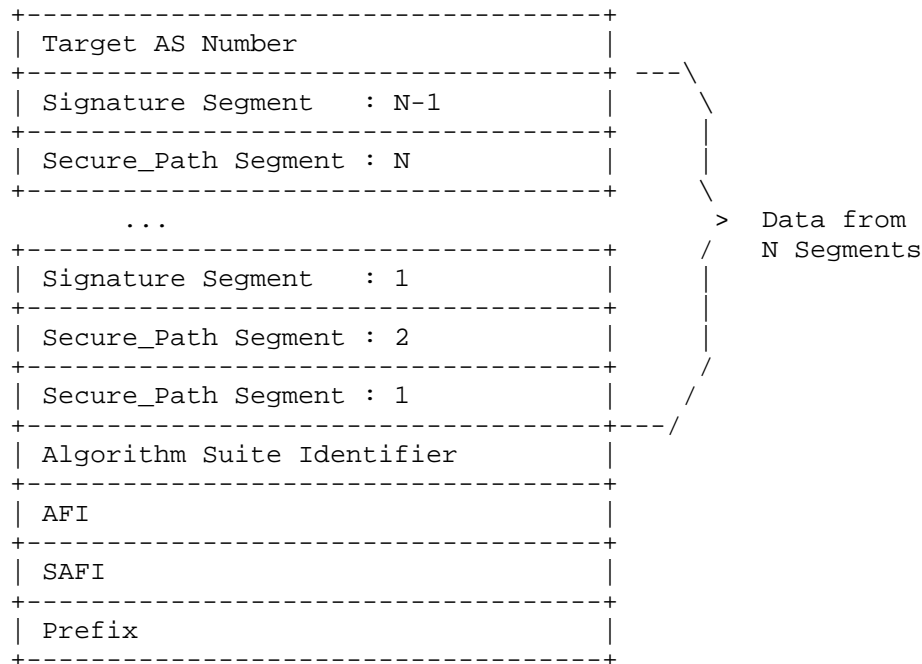


Figure 8: Sequence of octets to be hashed.

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.

- o Apply to this octet sequence (in Figure 8) 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 (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:

- o 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.
- o 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.)
- o 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.

- o (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.
- o (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.
- o (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).)

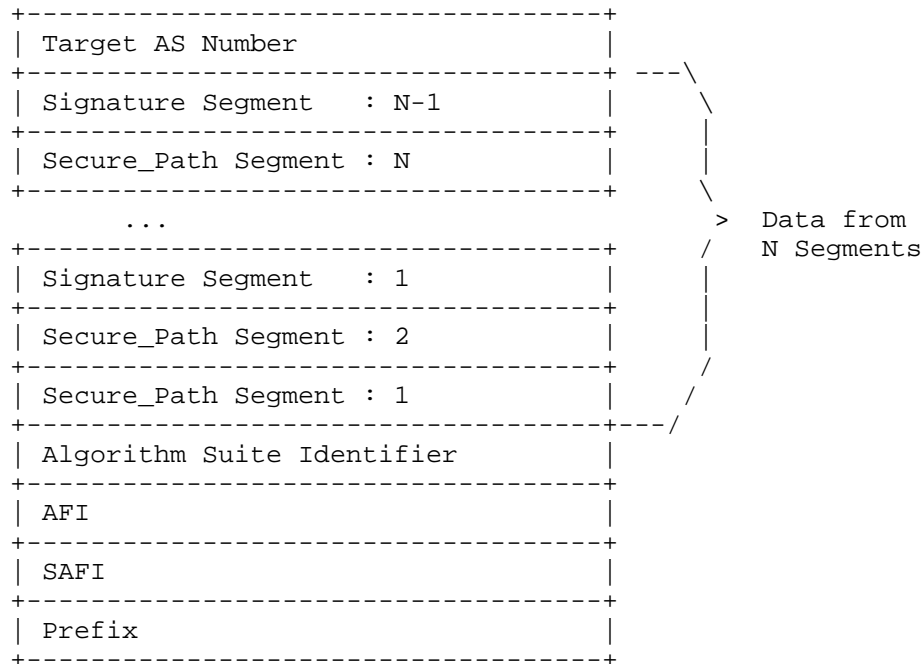


Figure 9: The Sequence of octets to be hashed for signature verification of Signature Segment N; $N = 1, 2, \dots, K$, where K is the number of AS hops in the BGPsec_Path attribute.

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:

- 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. 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:

- 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 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:

- o The BGPsec speaker received the given route advertisement with the indicated prefix, AFI, SAFI, and Secure_Path; and
- o 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 is 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@hacitrn.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
dougmn@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 Waehlich, 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](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: March 3, 2019

R. Bush
IIJ Lab / Dragon Research Lab
S. Turner
sn3rd
K. Patel
Arrcus, Inc.
August 30, 2018

Router Keying for BGPsec
draft-ietf-sidr-rtr-keying-16

Abstract

BGPsec-speaking routers are provisioned with private keys in order to sign BGPsec announcements. The corresponding public keys are published in the global Resource Public Key Infrastructure, enabling verification of BGPsec messages. This document describes two methods of generating the public-private key-pairs: router-driven and operator-driven.

Requirements Language

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

Status of This Memo

This Internet-Draft 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 16, 2017.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Management / Router Communication	3
3. Exchange Certificates	4
4. Set-Up	4
5. Generate PKCS#10	4
5.1. Router-Generated Keys	5
5.2. Operator-Generated Keys	5
5.2.1. Using PKCS#8 to Transfer Private Key	5
6. Send PKCS#10 and Receive PKCS#7	6
7. Install Certificate	6
8. Advanced Deployment Scenarios	7
9. Key Management	8
9.1. Key Validity	8
9.2. Key Roll-Over	9
9.3. Key Revocation	9
9.4. Router Replacement	10
10. Security Considerations	10
11. IANA Considerations	12
12. References	12
12.1. Normative References	12
12.1. Informative References	13
Appendix A. Management/Router Channel Security	15
Appendix B. The n00b Guide to BGPsec Key Management	15
Authors' Addresses	18

1. Introduction

BGPsec-speaking routers are provisioned with private keys, which allow them to digitally sign BGPsec announcements. To verify the signature, the public key, in the form of a certificate [RFC8209], is

published in the Resource Public Key Infrastructure (RPKI). This document describes provisioning of BGPsec-speaking routers with the appropriate public-private key-pairs. There are two sub-methods, router-driven and operator-driven.

These two sub-methods differ in where the keys are generated: on the router in the router-driven method, and elsewhere in the operator-driven method. Routers are required to support at least one of the methods in order to work in various deployment environments. Some routers may not allow the private key to be off-loaded while others may. While off-loading private keys would ease swapping of routing engines, exposure of private keys is a well known security risk.

In the operator-driven method, the operator generates the private/public key-pair and sends it to the router.

In the router-driven method, the router generates its own public/private key-pair.

The router-driven model mirrors the model used by traditional PKI subscribers; the private key never leaves trusted storage (e.g., Hardware Security Module). This is by design and supports classic PKI Certification Policies for (often human) subscribers which require the private key only ever be controlled by the subscriber to ensure that no one can impersonate the subscriber. For non-humans, this model does not always work. For example, when an operator wants to support hot-swappable routers the same private key needs to be installed in the soon-to-be online router that was used by the soon-to-be offline router. This motivated the operator-driven model.

The remainder of this document describes how operators can use the two methods to provision new and existing routers. The methods described involve the operator configuring the two end points (i.e., the management station and the router) and acting as the intermediary. Section 7 describes a method that requires more capable routers.

Useful References: [RFC8205] describes gritty details, [RFC8209] specifies the format for the PKCS#10 certification request, and [RFC8208] specifies the algorithms used to generate the PKCS#10's signature.

2. Management / Router Communication

Operators are free to use either the router-driven or operator-driven method as supported by the platform. Regardless of the method chosen, operators first establish a protected channel between the management system and the router. How this protected channel is

established is router-specific and is beyond scope of this document. Though other configuration mechanisms might be used, e.g. NetConf (see [RFC6470]); for simplicity, in this document, the protected channel between the management platform and the router is assumed to be an SSH-protected CLI. See Appendix A for security considerations for this protected channel.

3. Exchange Certificates

A number of options exist for the operator management station to exchange PKI-related information with routers and with the RPKI including:

- Use application/pkcs10 media type [RFC5967] to extract certificate requests and application/pkcs7-mime [I-D.lamps-rfc5751-bis] to return the issued certificate,
- Use FTP or HTTP per [RFC2585], and
- Use Enrollment over Secure Transport (EST) protocol per [RFC7030].

4. Set-Up

To start, the operator uses the protected channel to install the appropriate RPKI Trust Anchor's Certificate (TA Cert) in the router. This will later enable the router to validate the router certificate returned in the PKCS#7 certs-only message [I-D.lamps-rfc5751-bis].

The operator also configures the Autonomous System (AS) number to be used in the generated router certificate. This may be the sole AS configured on the router, or an operator choice if the router is configured with multiple ASs. A router with multiple ASs can be configured with multiple router certificates by following the process of this document for each desired certificate.

The operator configures or extracts from the router the BGP Identifier [RFC4271] to be used in the generated router certificate. In the case where the operator has chosen not to use unique per-router certificates, a BGP Identifier of 0 may be used.

5. Generate PKCS#10

The private key, and hence the PKCS#10 certification request, which is sometimes referred to as a Certificate Signing Request (CSR), may be generated by the router or by the operator.

The PKCS#10 request SHOULD be saved to enable verifying that the returned public key in the certificate corresponds to the private

used to generate the signature on the CSR.

NOTE: The PKCS#10 certification request does not include the AS number or the BGP Identifier for the router certificate. Therefore, the operator transmits the AS it has chosen or the router and the BGP Identifier as well when it sends the CSR to the CA.

5.1. Router-Generated Keys

In the router-generated method, once the protected channel is established and the initial Set-Up (Section 4) performed, the operator issues a command or commands for the router to generate the public/private key pair, to generate the PKCS#10 certification request, and to sign the PKCS#10 certification request with the private key. Once generated, the PKCS#10 certification request is returned to the operator over the protected channel.

The operator includes the chosen AS number and the BGP Identifier when it sends the CSR to the CA.

NOTE: If a router were to communicate directly with a CA to have the CA certify the PKCS#10 certification request, there would be no way for the CA to authenticate the router. As the operator knows the authenticity of the router, the operator mediates the communication with the CA.

5.2. Operator-Generated Keys

In the operator-generated method, the operator generates the public/private key pair on a management station and installs the private key into the router over the protected channel. Beware that experience has shown that copy and paste from a management station to a router can be unreliable for long texts.

The operator then creates and signs the PKCS#10 certification request with the private key; the operator includes the chosen AS number and the BGP Identifier when it sends the CSR to the CA.

Even if the operator cannot extract the private key from the router, this signature still provides a linkage between a private key and a router. That is the operator can verify the proof of possession (POP), as required by [RFC6484].

5.2.1. Using PKCS#8 to Transfer Private Key

A private key can be encapsulated in a PKCS#8 Asymmetric Key Package [RFC5958] and should be further encapsulated in Cryptographic Message Syntax (CMS) SignedData [RFC5652] and signed with the AS's End Entity

(EE) private key.

The router SHOULD verify the signature of the encapsulated PKCS#8 to ensure the returned private key did in fact come from the operator, but this requires that the operator also provision via the CLI or include in the SignedData the RPKI CA certificate and relevant AS's EE certificate(s). The router should inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope.

6. Send PKCS#10 and Receive PKCS#7

The operator uses RPKI management tools to communicate with the global RPKI system to have the appropriate CA validate the PKCS#10 certification request, sign the key in the PKCS#10 (i.e., certify it) and generate a PKCS#7 certs-only message, as well as publishing the certificate in the Global RPKI. External network connectivity may be needed if the certificate is to be published in the Global RPKI.

After the CA certifies the key, it does two things:

1. Publishes the certificate in the Global RPKI. The CA must have connectivity to the relevant publication point, which in turn must have external network connectivity as it is part of the Global RPKI.
2. Returns the certificate to the operator's management station, packaged in a PKCS#7 certs-only message, using the corresponding method by which it received the certificate request. It SHOULD include the certificate chain below the TA Certificate so that the router can validate the router certificate.

In the operator-generated method, the operator SHOULD extract the certificate from the PKCS#7 certs-only message, and verify that the private key it holds corresponds to the returned public key. If the operator saved the PKCS#10 it can check this correspondence by comparing the public key in the CSR to the public key in the returned certificate. If the operator has not saved the PKCS#10, it can check this correspondence by generating a signature on any data and then verifying the signature using the returned certificate.

In the operator-generated method, the operator has already installed the private key in the router (see Section 5.2).

7. Install Certificate

The operator provisions the PKCS#7 certs-only message into the router over the protected channel.

The router SHOULD extract the certificate from the PKCS#7 certs-only message and verify that the public key corresponds to the stored private key. If the router stored the PKCS#10, it can check this correspondence by comparing the public key in the CSR to the public key in the returned certificate. If the router did not store the PKCS#10, it can check this correspondence by generating a signature on any data and then verifying the signature using the returned certificate. The router SHOULD inform the operator whether it successfully received the certificate and whether or not the keys correspond; the mechanism is out of scope.

The router SHOULD also verify that the returned certificate validates back to the installed TA Certificate, i.e., the entire chain from the installed TA Certificate through subordinate CAs to the BGPsec certificate validate. To perform this verification the CA certificate chain needs to be returned along with the router's certificate in the PKCS#7 certs-only message. The router SHOULD inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope.

NOTE: The signature on the PKCS#8 and Certificate need not be made by the same entity. Signing the PKCS#8, permits more advanced configurations where the entity that generates the keys is not the direct CA.

8. Advanced Deployment Scenarios

More PKI-capable routers can take advantage of this increased functionality and lighten the operator's burden. Typically, these routers include either pre-installed manufacturer-generated certificates (e.g., IEEE 802.1 AR [802.1AR]) or pre-installed manufacturer-generated Pre-Shared Keys (PSK) as well as PKI-enrollment functionality and transport protocol, e.g., CMC's "Secure Transport" [RFC7030] or the original CMC transport protocol's [RFC5273]. When the operator first establishes a protected channel between the management system and the router, this pre-installed key material is used to authenticate the router.

The operator burden shifts here to include:

1. Securely communicating the router's authentication material to the CA prior to operator initiating the router's CSR. CAs use authentication material to determine whether the router is eligible to receive a certificate. Authentication material at a minimum includes the router's AS number and BGP Identifier as well as the router's key material, but can also include additional information. Authentication material can be communicated to the CA (i.e., CSRs signed by this key material

are issued certificates with this AS and BGP Identifier) or to the router (i.e., the operator uses the vendor-supplied management interface to include the AS number and BGP Identifier in the router-generated CSR).

2. Enabling the router to communicate with the CA. While the router-to-CA communications are operator-initiated, the operator's management interface need not be involved in the communications path. Enabling the router-to-CA connectivity MAY require connections to external networks (i.e., through firewalls, NATs, etc.).

Once configured, the operator can begin the process of enrolling the router. Because the router is communicating directly with the CA, there is no need for the operator to retrieve the PKCS#10 certification request from the router as in Section 5 or return the PKCS#7 certs-only message to the router as in Section 6. Note that the checks performed by the router in Section 7, namely extracting the certificate from the PKCS#7 certs-only message, verifying the public key corresponds to the private key, and that the returned certificate validated back to an installed trust anchor, SHOULD be performed. Likewise, the router SHOULD notify the operator if any of these fail, but this notification mechanism is out of scope.

When a router is so configured the communication with the CA SHOULD be automatically re-established by the router at future times to renew or rekey the certificate automatically when necessary (See Section 8). This further reduces the tasks required of the operator.

9. Key Management

Key management does not only include key generation, key provisioning, certificate issuance, and certificate distribution. It also includes assurance of key validity, key roll-over, and key preservation during router replacement. All of these responsibilities persist for as long as the operator wishes to operate the BGPsec-speaking router.

9.1. Key Validity

It is critical that a BGPsec speaking router is signing with a valid private key at all times. To this end, the operator needs to ensure the router always has a non-expired certificate. I.e. the key used to sign BGPsec announcements always has an associated certificate whose expiry time is after the current time.

Ensuring this is not terribly difficult but requires that either:

1. The router has a mechanism to notify the operator that the certificate has an impending expiration, and/or
2. The operator notes the expiry time of the certificate and uses a calendaring program to remind them of the expiry time, and/or
3. The RPKI CA warns the operator of pending expiration, and/or
4. The operator uses some other kind of automated process to search for and track the expiry times of router certificates.

It is advisable that expiration warnings happen well in advance of the actual expiry time.

Regardless of the technique used to track router certificate expiry times, it is advisable to notify additional operators in the same organization as the expiry time approaches thereby ensuring that the forgetfulness of one operator does not affect the entire organization.

Depending on inter-operator relationship, it may be helpful to notify a peer operator that one or more of their certificates are about to expire.

9.2. Key Roll-Over

Routers that support multiple private keys also greatly increase the chance that routers can continuously speak BGPsec because the new private key and certificate can be obtained and distributed prior to expiration of the operational key. Obviously, the router needs to know when to start using the new key. Once the new key is being used, having the already distributed certificate ensures continuous operation.

More information on how to proceed with a Key Roll-Over is described in [I-D.sidrops-bgpsec-rollover].

9.3. Key Revocation

Certain unfortunate circumstances may occur causing a need to revoke a router's BGPsec certificate. When this occurs, the operator needs to use the RPKI CA system to revoke the certificate by placing the router's BGPsec certificate on the Certificate Revocation List (CRL) as well as re-keying the router's certificate.

When an active router key is to be revoked, the process of requesting the CA to revoke, the process of the CA actually revoking the router's certificate, and then the process of re-keying/renewing the

router's certificate, (possibly distributing a new key and certificate to the router), and distributing the status takes time during which the operator must decide how they wish to maintain continuity of operations, with or without the compromised private key, or whether they wish to bring the router offline to address the compromise.

Keeping the router operational and BGPsec-speaking is the ideal goal, but if operational practices do not allow this then reconfiguring the router to disable BGPsec is likely preferred to bringing the router offline.

Routers which support more than one private key, where one is operational and other(s) are soon-to-be-operational, facilitate revocation events because the operator can configure the router to make a soon-to-be-operational key operational, request revocation of the compromised key, and then make a next generation soon-to-be-operational key, all hopefully without needing to take offline or reboot the router. For routers which support only one operational key, the operators should create or install the new private key, and then request revocation of the certificate corresponding to the compromised private key.

9.4. Router Replacement

Currently routers often generate private keys for uses such as SSH, and the private keys may not be seen or off-loaded from the router. While this is good security, it creates difficulties when a routing engine or whole router must be replaced in the field and all software which accesses the router must be updated with the new keys. Also, any network based initial contact with a new routing engine requires trust in the public key presented on first contact.

To allow operators to quickly replace routers without requiring update and distribution of the corresponding public keys in the RPKI, routers SHOULD allow the private BGPsec key to be inserted via a protected channel, e.g., SSH, NetConf (see [RFC6470]), SNMP. This lets the operator escrow the old private key via the mechanism used for operator-generated keys, see Section 5.2, such that it can be re-inserted into a replacement router. The router MAY allow the private key to be off-loaded via the protected channel, but this SHOULD be paired with functionality that sets the key into a permanent non-exportable state to ensure that it is not off-loaded at a future time by unauthorized operations.

10. Security Considerations

The router's manual will describe whether the router supports one,

the other, or both of the key generation options discussed in the earlier sections of this draft as well as other important security-related information (e.g., how to SSH to the router). After familiarizing one's self with the capabilities of the router, an operator is encouraged to ensure that the router is patched with the latest software updates available from the manufacturer.

This document defines no protocols so in some sense introduces no new security considerations. However, it relies on many others and the security considerations in the referenced documents should be consulted; notably, those document listed in Section 1 should be consulted first. PKI-relying protocols, of which BGPsec is one, have many issues to consider so many in fact entire books have been written to address them; so listing all PKI-related security considerations is neither useful nor helpful; regardless, some bootstrapping-related issues are listed here that are worth repeating:

Public-Private key pair generation: Mistakes here are for all practical purposes catastrophic because PKIs rely on the pairing of a difficult to generate public-private key pair with a signer; all key pairs MUST be generated from a good source of non-deterministic random input [RFC4086].

Private key protection at rest: Mistakes here are for all practical purposes catastrophic because disclosure of the private key allows another entity to masquerade as (i.e., impersonate) the signer; all private keys MUST be protected when at rest in a secure fashion. Obviously, how each router protects private keys is implementation specific. Likewise, the local storage format for the private key is just that, a local matter.

Private key protection in transit: Mistakes here are for all practical purposes catastrophic because disclosure of the private key allows another entity to masquerade as (i.e., impersonate) the signer; transport security is therefore strongly RECOMMENDED. The level of security provided by the transport layer's security mechanism SHOULD be commensurate with the strength of the BGPsec key; there's no point in spending time and energy to generate an excellent public-private key pair and then transmit the private key in the clear or with a known-to-be-broken algorithm, as it just undermines trust that the private key has been kept private. Additionally, operators SHOULD ensure the transport security mechanism is up to date, in order to addresses all known implementation bugs.

SSH key management is known, in some cases, to be lax [I-D.ylonen-sshkeybcp]; employees that no longer need access to a routers SHOULD be removed the router to ensure only those authorized

have access to a router.

Though the CA's certificate is installed on the router and used to verify that the returned certificate is in fact signed by the CA, the revocation status of the CA's certificate is rarely checked as the router may not have global connectivity or CRL-aware software. The operator MUST ensure that the installed CA certificate is valid.

11. IANA Considerations

This document has no IANA Considerations.

12. References

12.1. Normative References

- [I-D.sidrops-bgpsec-rollover]
Weis, B, R. Gagliano, and K. Patel, "BGPsec Router Certificate Rollover", draft-ietf-sidrops-bgpsec-rollover (work in progress), December 2017.
- [I-D.lamps-rfc5751-bis]
Schaad, J., Ramsdell, B, S. Turner,
"Secure/Multipurpose Internet Mail Extension (S/MIME)
Version 4.0", draft-ietf-lamps-rfc5751-
bis (work in progress), July 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
"Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <<https://www.rfc-editor.org/info/rfc4253>>.
- [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, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009,

<<https://www.rfc-editor.org/info/rfc5652>>.

- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <<https://www.rfc-editor.org/info/rfc5958>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8208] Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, and Signature Formats", RFC 8208, DOI 10.17487/RFC8208, September 2017, <<https://www.rfc-editor.org/info/rfc8208>>.
- [RFC8209] Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", RFC 8209, DOI 10.17487/RFC8209, September 2017, <<https://www.rfc-editor.org/info/rfc8209>>.
- [802.1AR] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

12.1. Informative References

- [I-D.ylonen-sshkeybc] Ylonen, T. and G. Kent, "Managing SSH Keys for Automated Access - Current Recommended Practice", draft-ylonen-sshkeybc (work in progress), April 2013.
- [RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, DOI 10.17487/RFC2585, May 1999, <<https://www.rfc-editor.org/info/rfc2585>>.
- [RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, DOI 10.17487/RFC3766, April 2004, <<https://www.rfc-editor.org/info/rfc3766>>.
- [RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, DOI

- 10.17487/RFC5273, June 2008, <<https://www.rfc-editor.org/info/rfc5273>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/info/rfc5480>>.
- [RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, DOI 10.17487/RFC5647, August 2009, <<https://www.rfc-editor.org/info/rfc5647>>.
- [RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <<https://www.rfc-editor.org/info/rfc5656>>.
- [RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, DOI 10.17487/RFC5967, August 2010, <<https://www.rfc-editor.org/info/rfc5967>>.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <<https://www.rfc-editor.org/info/rfc6187>>.
- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <<https://www.rfc-editor.org/info/rfc6470>>.
- [RFC6484] Kent, S., Kong, D., Seo, K., and R. Watro, "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, DOI 10.17487/RFC6484, February 2012, <<https://www.rfc-editor.org/info/rfc6484>>.
- [RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <<https://www.rfc-editor.org/info/rfc6668>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC8205] Lepinski, M., Ed., and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <<https://www.rfc-editor.org/info/rfc8205>>.

[SP800-57] National Institute of Standards and Technology (NIST),
Special Publication 800-57: Recommendation for Key
Management - Part 1 (Revised), March 2007.

Appendix A. Management/Router Channel Security

Encryption, integrity, authentication, and key exchange algorithms used by the protected channel SHOULD be of equal or greater strength than the BGPsec keys they protect, which for the algorithm specified in [RFC8208] is 128-bit; see [RFC5480] and by reference [SP800-57] for information about this strength claim as well as [RFC3766] for "how to determine the length of an asymmetric key as a function of a symmetric key strength requirement." In other words, for the encryption algorithm, do not use export grade crypto (40-56 bits of security), do not use Triple DES (112 bits of security). Suggested minimum algorithms would be AES-128: aes128-cbc [RFC4253] and AEAD_AES_128_GCM [RFC5647] for encryption, hmac-sha2-256 [RFC6668] or AESAD_AES_128_GCM [RFC5647] for integrity, ecdsa-sha2-nistp256 [RFC5656] for authentication, and ecdh-sha2-nistp256 [RFC5656] for key exchange.

Some routers support the use of public key certificates and SSH. The certificates used for the SSH session are different than the certificates used for BGPsec. The certificates used with SSH should also enable a level of security commensurate with BGPsec keys; x509v3-ecdsa-sha2-nistp256 [RFC6187] could be used for authentication.

The protected channel must provide confidentiality, authentication, and integrity and replay protection.

Appendix B. The n00b Guide to BGPsec Key Management

This appendix is informative. It attempts to explain all of the PKI technobabble in plainer language.

BGPsec speakers send signed BGPsec updates that are verified by other BGPsec speakers. In PKI parlance, the senders are referred to as signers and the receivers are referred to as relying parties. The signers with which we are concerned here are routers signing BGPsec updates. Signers use private keys to sign and relying parties use the corresponding public keys, in the form of X.509 public key certificates, to verify signatures. The third party involved is the entity that issues the X.509 public key certificate, the Certification Authority (CA). Key management is all about making these key pairs and the certificates, as well as ensuring that the relying parties trust that the certified public keys in fact correspond to the signers' private keys.

The specifics of key management greatly depend on the routers as well as management interfaces provided by the routers' vendor. Because of these differences, it is hard to write a definitive "how to," but this guide is intended to arm operators with enough information to ask the right questions. The other aspect that makes this guide informative is that the steps for the do-it-yourself (DIY) approach involve arcane commands while the GUI-based vendor-assisted management console approach will likely hide all of those commands behind some button clicks. Regardless, the operator will end up with a BGPsec-enabled router. Initially, we focus on the DIY approach and then follow up with some information about the GUI-based approach.

The first step in the DIY approach is to generate a private key; but in fact what you do is create a key pair; one part, the private key, is kept very private and the other part, the public key, is given out to verify whatever is signed. The two models for how to create the key pair are the subject of this document, but it boils down to either doing it on-router (router-driven) or off-router (operator-driven).

If you are generating keys on the router (router-driven), then you will need to access the router. Again, how you access the router is router-specific, but generally the DIY approach uses the CLI and accessing the router either directly via the router's craft port or over the network on an administrative interface. If accessing the router over the network be sure to do it securely (i.e., use SSHv2). Once logged into the router, issue a command or a series of commands that will generate the key pair for the algorithms referenced in the main body of this document; consult your router's documentation for the specific commands. The key generation process will yield multiple files: the private key and the public key; the file format varies depending on the arcane command you issued, but generally the files are DER or PEM-encoded.

The second step is to generate the certification request, which is often referred to as a certificate signing request (CSR) or PKCS#10 certification request, and to send it to the CA to be signed. To generate the CSR, you issue some more arcane commands while logged into the router; using the private key just generated to sign the certification request with the algorithms referenced in the main body of this document; the CSR is signed to prove to the CA that the router has possession of the private key (i.e., the signature is the proof-of-possession). The output of the command is the CSR file; the file format varies depending on the arcane command you issued, but generally the files are DER or PEM-encoded.

The third step is to retrieve the signed CSR from the router and send it to the CA. But before sending it, you need to also send the CA

the subject name and serial number for the router. The CA needs this information to issue the certificate. How you get the CSR to the CA, is beyond the scope of this document. While you are still connected to the router, install the Trust Anchor (TA) for the root of the PKI. At this point, you no longer need access to the router for BGPsec-related initiation purposes.

The fourth step is for the CA to issue the certificate based on the CSR you sent; the certificate will include the subject name, serial number, public key, and other fields as well as being signed by the CA. After the CA issues the certificate, the CA returns the certificate, and posts the certificate to the RPKI repository. Check that the certificate corresponds to the private key by verifying the signature on the CSR sent to the CA; this is just a check to make sure that the CA issued a certificate that includes a public key that is the pair of the private key (i.e., the math will work when verifying a signature generated by the private with the returned certificate).

If generating the keys off-router (operator-driven), then the same steps are used as the on-router key generation, (possibly with the same arcane commands as those used in the on-router approach), but no access to the router is needed the first three steps are done on an administrative workstation: o Step 1: Generate key pair; o Step 2: Create CSR and sign CSR with private key, and; o Step 3: Send CSR file with the subject name and serial number to CA.

After the CA has returned the certificate and you have checked the certificate, you need to put the private key and TA in the router. Assuming the DIY approach, you will be using the CLI and accessing the router either directly via the router's craft port or over the network on an admin interface; if accessing the router over the network make doubly sure it is done securely (i.e., use SSHv2) because the private key is being moved over the network. At this point, access to the router is no longer needed for BGPsec-related initiation purposes.

NOTE: Regardless of the approach taken, the first three steps could trivially be collapsed by a vendor-provided script to yield the private key and the signed CSR.

Given a GUI-based vendor-assisted management console, then all of these steps will likely be hidden behind pointing and clicking the way through BGPsec-enabling the router.

The scenarios described above require the operator to access each router, which does not scale well to large networks. An alternative

would be to create an image, perform the necessary steps to get the private key and trust anchor on the image, and then install the image via a management protocol.

One final word of advice; certificates include a notAfter field that unsurprisingly indicates when relying parties should no longer trust the certificate. To avoid having routers with expired certificates follow the recommendations in the Certification Policy (CP) [RFC6484] and make sure to renew the certificate at least one week prior to the notAfter date. Set a calendar reminder in order not to forget!

Authors' Addresses

Randy Bush
IIJ / Dragon Research Labs
5147 Crystal Springs
Bainbridge Island, Washington 98110
US

Email: randy@psg.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

Keyur Patel
Arrcus, Inc.

Email: keyur@arrcus.com

Network Working Group
Internet-Draft
Updates: 6487 (if approved)
Intended status: Standards Track
Expires: July 7, 2013

A. Newton
ARIN
G. Huston
APNIC
January 3, 2013

Policy Qualifiers in RPKI Certificates
draft-newton-sidr-policy-qualifiers-01

Abstract

This document updates RFC 6487 by clarifying the inclusion of policy qualifiers in the certificate policies extension of RPKI resource certificates.

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 July 7, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Update to RFC 6487	4
3. IANA Considerations	5
4. Security Considerations	6
5. Acknowledgements	7
6. Normative References	8
Authors' Addresses	9

1. Terminology

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. Update to RFC 6487

[RFC6487] profiles certificates, certificate revocation lists, and certificate signing requests specified in [RFC5280] for use in routing public key infrastructure.

[RFC5280] defines an extension to certificates for the listing of policy information (See section 4.2.1.4). [RFC6487] states in Section 4.8.9: "This extension MUST be present and MUST be marked critical. It MUST include exactly one policy, as specified in the RPKI CP [RFC6484]". This references the CertPolicyId of the sequence allowed in PolicyInformation as defined by [RFC5280].

[RFC5280] also specifies that PolicyInformation may optionally have a sequence of PolicyQualifierInfo objects. [RFC6487] does not specifically allow or disallow these PolicyQualifierInfo objects although it also states in section 4: "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."

This document updates [RFC6487], Section 4.8.9, as follows:

OLD:

This extension MUST be present and MUST be marked critical. It MUST include exactly one policy, as specified in the RPKI CP [RFC6484].

NEW:

This extension MUST be present and MUST be marked critical. It MUST include exactly one policy, as specified in the RPKI CP [RFC6484]. Exactly one policy qualifier MAY be included. If a policy qualifier is included, the policyQualifierId MUST be the 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 qualifiers that MAY be present in a resource certificate MUST NOT change the definition of the RPKI CP [RFC6484].

3. IANA Considerations

None.

4. Security Considerations

The Security Considerations of [RFC6487] apply to this document.

This document updates the RPKI certificate profile to specify that the certificate policies extension can include a policy qualifier, which is a URI. Checking of the URI might allow denial-of-service (DoS) attacks, where the target host may be subjected to bogus work resolving the URI. However, this specification, like [RFC5280], places no processing requirements on the URI included in the qualifier.

5. Acknowledgements

Frank Hill and Adam Guyot helped define the scope of this issue and identified and worked with RPKI validator implementers to clarify the use of policy qualifiers in resource certificates.

Sean Turner provided significant text to this document regarding the processing of the CPS URI and limiting the scope of the allowable content of the policy qualifier.

6. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [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.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.

Authors' Addresses

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
US

Email: andy@arin.net
URI: <http://www.arin.net>

Geoff Huston
Asia Pacific Network Information Center
6 Cordelia Street
South Brisbane QLD 4101
Australia

Email: gih@apnic.net
URI: <http://www.apnic.net>

Network Working Group
Internet-Draft
Updates: RFC 6490 (if approved)
Intended status: Standards Track
Expires: August 29, 2013

R. Gagliano
Cisco Systems
T. Manderson
ICANN
C. Martinez
LACNIC
February 25, 2013

Multiple Repository Publication Points support in the Resource Public
Key Infrastructure (RPKI)
draft-rogalia-sidr-multiple-publication-points-02

Abstract

The Resource Public Key Infrastructure (RPKI) depends on Relying Parties (RP) ability to access its Trust Anchors' certificate specified in the different "Trust Anchor Locator (TAL)" files and the Repository Objects located at the Certificate Authorities (CA) repositories hosted in its respective publication point. This document updates [RFC6490] by allowing multiple URI associated to a single public key in a TAL file and introduces the concept of multiple repository publication point operators for every CA in the RPKI. This document provides also recommendation for the RP behavior when analyzing signed objects that include multiple publications points.

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 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	3
2. Introduction	4
3. Multiple Operators support in TAL files	6
3.1. Update to RFC 6490 Section 2.1	6
3.2. Rules for Relying Parties (RP)	7
4. Multiple Operators support in Certificates	8
4.1. Rules for Relying Parties (RP)	8
5. IANA Considerations	9
6. Security Considerations	10
7. Acknowledgements	11
8. Normative References	12
Authors' Addresses	13

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

The RPKI repository system described in [RFC6481] requires scalability and diversity in order to address challenges such as Distributed Denial of Service (DDoS) attacks, to secure the availability of the system when performing maintenance activities and against possible security incidents in one particular implementation. Additionally, when a single operator manages a RPKI Repository Publication Point, it is more probable to introduce circular dependencies when the Route Origin Authorization (ROA) signed objects for the Repository Publication Point IP addresses are hosted in servers that uses those same addresses.

The current toolset for a CA to diversify its repository system is limited for both TA distribution and CA publication point management. In the case of trust anchors, [RFC6490] requires a unique URI per key on each TAL file. Conversely, in the case of the different publication points and although supported by [RFC6487], there is no current guidance on how RPs should support multiple publication points for the same object.

When using a single URI, the options for diversity and scalability are reduced to:

1. Give the content to a Content Delivery Network (CDN) to have the content distributed (as long as the CDN supports the CA's access method, which is not currently the case for rsync). The implementation will typically require the configuration of a CNAME resource record in the authoritative server pointing to a server farm inside the CDN who will handle load-balancing by using a set of internally defined metrics. If, for the sake of diversity, a CA administrator would like to use two different CDNs for the same URI it will need to modify the authoritative name server behavior to break RFC1034 standard behavior and allow multiple CNAME records for the same alias. This modification is not available by default on most of the more widely deployed DNS servers.
2. Copy the content to different Repository Publication Points around the globe (i.e. using [I-D.ietf-sidr-publication]) and load balance the content using different Domain Name System (DNS) techniques. The load balancing implementation will need to verify the availability of the target server before providing a DNS response to avoid blackholes caused by unavailable servers or clusters. This "feature" needs also be added to the authoritative name server or the full DNS resolution or outsourced to a third party (which would introduce another non-diversified element).

This document addresses this problem by enabling multiple operators for trust anchor material, and, while not making it mandatory, recommends the use of multiple publication points in signed objects.

The main idea is that the a CA will host its RPKI signed objects in different locations, using diverse routing paths and diverse DNS resolution. The RP will have more processing to perform to fetch the different objects when dealing with exceptions.

The first thing that is needed is to add multiple URIs support for each Trust Anchor. [RFC6490] requires that each TAL file includes a unique URI. This document removes this requirement by allowing one or more URI for each public key in a TAL file. In steady state, an RP should receive the same material from each of the different URI for the same root certificate. An exception could happen when the certificate is been updated or rolled-over, a process which should not have operational consequences.

For the root certificate trust anchor, this proposal has an additional consequence: it would create the idea of root-CA repository operators. This concept has worked well in the case of DNS, where one organization is responsible for creating the root zone material and a number of different organizations are responsible in running the root servers.

A CA can add support for multiple Repository Publication Points operators by adding more than one respective object for the Authority Information Access (AIA), the Subject Information Access (SIA) and the CRL Distribution Points (CRLDP) and which is supported by [RFC5280] and [RFC6487] . This document provides guidance on the RP expected behavior when analyzing signed objects with multiple Repository Publication Points in Section 4.

3. Multiple Operators support in TAL files

The idea of multiples operators support for a TA certificate expressed on its TAL file is similar to the support for several Root Server operators in a DNS hints file.

An example of such a TAL file with 3 operators would be:

```
rsync://rpki.operator1.org/rpki/hedgehog/root.cer
rsync://rpki.operator2.net/rpki/hedgehog/root.cer
rsync://rpki.operator3.biz/rpki/hedgehog/root.cer
```

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAovWQL2lh6knDx
GUG5hbtCXvvh4AOzjhDkSHlj22gn/loiM9IeDATIwP44vhQ6L/xvuk7W6
Kfa5ygmqQ+xOZOwTWPCrUbqaQyPNxokuivzyvqVZVDecOEqs78q58mSp9
nbtxmLRW7B67SJCBSzfa5XpVyXYEgYAJkk3fpmefU+AcctxvvHB5OVPIa
BfPcs80ICMgHQX+fphvute9XLxjfkKJWkhZqZ0v7pZm2uhkcPx1PMGcrG
ee0WSDC3fr3erLueagpiLsFjwwpX6F+Ms8vqz45H+DKmYKvPSstZjCCq9
aJ0qANT90tnfSDOS+aLRPjZryCNyvvBHxZXqj5YCGKtwIDAQAB
```

As we can see in this example, a RP would have different URI where to fetch the self-signed certificate for the trust anchor. In each location, the same result should be expected as all the URI share the same public key.

In order to increase diversity, It is RECOMMENDED that the different FQDN could be resolved to IP addresses included in ROA objects from different CAs and hosted in diverse repository publication points.

3.1. Update to RFC 6490 Section 2.1

The following text will replace the last paragraph on Section 2.1 of RFC 6490:

The TAL is an ordered sequence of:

- 1) One or more rsync URI [RFC5781],
- 2) A <CRLF> or <LF> line break after each URI,
- 3) A line containing a single <CRLF> or <LF> line break, and
- 4) A subjectPublicKeyInfo [RFC5280] in DER format [X.509], encoded in Base64 (see Section 4 of [RFC4648]).A

3.2. Rules for Relying Parties (RP)

A RP can use different rules to select the URI from where fetch the Trust Anchor certificate. Some examples are:

- o Using the order provided in the TAL file
- o Selecting the URI randomly from the available list
- o Creating a prioritized list of URIs based on RP specific parameters such as connection establishment delay

If the connection to the preferred URI fails or the fetched certificate public key does not match the TAL public key, the RP SHOULD fetch the TA certificate from the next URI of preference.

4. Multiple Operators support in Certificates

The support for multiple operators in the RPKI Certificate Authority (CA) and End Entity (EE) certificates is supported as the RFC 5082 allows multiple repository publication point operators as the SIA, AIA and CRLDP are implemented as sequences. Consequently, no changes are needed on the existing RPKI standard and this section could be considered informative.

In the case of the SIA extension, for each operator, the accessMethods for both the CA repository publication point and for the correspondent manifest needs to be added.

4.1. Rules for Relying Parties (RP)

A RP can use different rules to select the URI to fetch the different repository objects and when performing the validation.

When a RP needs to fetch one or more object from a list of possible URIs, it can chose the URI by adopting a locally defined rule that could be:

- o Using the order provided in the correspondent certificate
- o Selecting the URI randomly from the available list
- o Creating a prioritized list of URIs based on RP specific parameters such as connection establishment delay

If the connection to the preferred URI fails , the RP SHOULD fetch the repository objects from the next URI of preference.

5. IANA Considerations

No IANA requirements

6. Security Considerations

TBA

7. Acknowledgements

TBA.

8. Normative References

- [I-D.ietf-sidr-publication]
"A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", <<http://www.ietf.org/id/draft-ietf-sidr-publication-02.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, February 2012.
- [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.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.
- [RFC6490] Huston, G., Weiler, S., Michaelson, G., and S. Kent, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 6490, February 2012.
- [RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, February 2012.

Authors' Addresses

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

Email: rogaglia@cisco.com

Terry Manderson
ICANN

Email: terry.manderson@icann.org

Carlos Martinez
LACNIC

Email: carlos@lacnic.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 23, 2015

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
D. Mandelberg
BBN Technologies
December 22, 2014

RPKI Repository Delta Protocol
draft-tbruijnzeels-sidr-delta-protocol-03

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 June 23, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	2
2. Introduction	2
3. RPKI Repository Delta Protocol Implementation	3
3.1. Informal Overview	3
3.2. Update Notification File	5
3.2.1. Purpose	5
3.2.2. Cache Concerns	5
3.2.3. File Format and Validation	5
3.2.4. Publication Server Initialisation	6
3.2.5. Publishing Updates	6
3.3. Snapshot File	7
3.3.1. Purpose	7
3.3.2. Cache Concerns	7
3.3.3. File Format and Validation	8
3.4. Delta File	10
3.4.1. Purpose	10
3.4.2. Cache Concerns	11
3.4.3. File Format and Validation	11
3.5. SIA for CA certificates	13
4. Relying Party Use	14
4.1. Full Synchronisation	14
4.2. Processing Deltas	14
5. XML Schema	15
6. Security Considerations	17
7. IANA Considerations	17
8. Acknowledgements	17
9. References	17
Authors' Addresses	18

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 it can easily be fetched over HTTP(S). The publication server may use HTTP caching infrastructure to reduce its load. The publication server 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 indefinitely, 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 should 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/EEEE7F7AD96D85BBD1F7274FA7
DA0025984A2AF3D5A0538F77BEC732ECB1B068.xml" hash="EEEE7F7AD96D85BBD1F7274FA7DA00
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:

- o 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.
- 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 serial attribute must be an unbounded, unsigned positive integer indicating the current version of the repository.
- o The notification file MUST contain exactly one 'snapshot' element for the current repository version.
- o 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.
- o 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 snapshot 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 snapshot 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 snapshot 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:

- o 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.
- o 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 snapshot 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">
    MII FNDCCBBygAwIBAgIBAjANBgkqhkiG9w0BAQsFADANMQswCQYDVQQDEwJUQTAEFw0xNDExM
TMw
    MzU4MjlaFw0xNTEyMTMwMzU4MjlaMDMxMTAvBgNVBAMTKDY3MTU3MGYwNjQ5OWZiZDJKNmFiN
zZj
    NGYyMjU2NmZlNDlkNWRLNjAwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQD0lUYxD
Pwu
    hqVSG5VXcg96qTYt9aKOH8qV2lAU/jnYlRl2W5Uoa8RrAiseou8ltLKonMcVulHyoyY+J9Gq
rzN
    45vRSgBaOuvLn6nTuoD0LQsD/m8c/wEmFjQllirxQykLGJLXnleKdUs/OXGgrAUPzgvkciJds
g69
    6X44deHcbCU0ZQZSLxZBZEQjfgyoYgww9n/hK5Sfkb44LsBK1lESdBsRrTpFizrCx122ptsH0
eW4
    ek80CV5YgCg4F4u9xlzS2DvB+1X3N1lvvTZ6TJlpVjIVcve+sKQ50ntUwWG1+lOJc+twRehhi
CAB
    yHhfaxID4B+7h5Rcpkh1Q1AUMG9JAgMBAAGjggJ3MIICczAdBgNVHQ4EFggQUZxVw8GSZ+9LWq
3bE
    8iVm/knV3mAwHwYDVR0jBBgwFoAUd4IboVL1+9bEbD6VrCsnqRC1FNUwDwYDVR0TAQH/BAUwA
WEB
    /zAOBgNVHQ8BAf8EBAMCAQYwRQYIKwYBBQUHAQEEOATA3MDUGCCsGAQUFBzACHilodHRwOi8vY
mFu
    ZG10by5yaXB1Lm5ldC9ycGtpLWNhL3RhLmNlcjCCATAGCCsGAQUFBwELBIIBIjCCAR4wV
wYI
    KwYBBQUHMAWGS3JzeW5jOi8vYmFuZG10by5yaXB1Lm5ldC9yZXBvLzNhODdhNGIxLTZlMjItN
GE2
    MylhZDBmLTA2ZjgzYWQzY2ExNi9kZWZhdWx0LzCBgwYIKwYBBQUHMAqGd3JzeW5jOi8vYmFuZ
Gl0
    by5yaXB1Lm5ldC9yZXBvLzNhODdhNGIxLTZlMjItNGE2MylhZDBmLTA2ZjgzYWQzY2ExNi9kZ
WZh
    dWx0LzY3MTU3MGYwNjQ5OWZiZDJKNmFiNzZjNGYyMjU2NmZlNDlkNWRLNjAubWZ0MD0GCCsGA
QUF
    BzANhjFodHRwOi8vYmFuZG10by5yaXB1Lm5ldC9ycGtpLWNhL25vdGlmeS9ub3RpZnkueG1sM
FsG
    AlUdHwRUMFIwUKBOoEyGSnJzeW5jOi8vYmFuZG10by5yaXB1Lm5ldC9yZXBvLzZc3ODIxYmExN
TJl
    NWZiZDZjNDZjM2U5NWFiMmIyN2E5MTBhNTE0ZDUuY3JsMBGGA1UdIAEB/wQOMAwcGyYIKwYBB
QUH
    DgIwHgYIKwYBBQUHAQcBAf8EDzANMAseAgABMAUDAwDAqDANBgkqhkiG9w0BAQsFAAOCAQEAK
Anl
    E+Fmlr3cmW8EEwhq4Wo37j7qC8ciU/E/zJqptROd8M8+2PDjCF8K7plf/SqYNUWjCk8zQv7Si
ala
    DP3JNi7oWkJ5K9zSU/qPGD8UbrfK5EF4g+++OAsxsOf/qeMVdZ6FlPIUv0wYj2s9w1zz/r16H
FV6
    QO785ajB50foqo/oQ74BSRbrlyKwRm8U45rdSiAMlyr0lHgv0OCqNK6AVR6y9Sp6bBui7RotZ
5FN
    x0TgBRTA6xp4pjG5FimX1SanMaWlhgYqdc4X5aZ9gPiyqvBcOtFq91WnNTsm5Ox0cPNDCKMPL
AwW
    pHoifa0PlD0vBPrvTRlhsgfKGd318Qzq+w==
  </publish>
  <publish uri="rsync://bandito.ripe.net/repo/77821ba152e5fbd6c46c3e95ac2b27a
910a514d5.mft">
    MIAGCSqGSIb3DQEHAqCAMIACAQMDzANBgglghkgBZQMEAgEFADCABgsqhkiG9w0BCRABGqCAJ
IAE
    gd0wgdoCAguWGA8yMDE0MTIwMzE4MDgzMloYDzIwMTQxMjA0MTgwODMyWgYJYIZIAWUDBAIBM
IGm
    MFEWLDY3MTU3MGYwNjQ5OWZiZDJKNmFiNzZjNGYyMjU2NmZlNDlkNWRLNjAuY2VyAyEAh0nT6
uSg
    nJQhGAnKgjb9TDeGu9AEd8QK+GHXYop0U8wURYsNzc4MjFiYTElMmU1ZmJkNmM0NmMzZTk1Y
WMy
    Yji3YTkxMGE1MTRkNS5jcmwDIQAZ658FmRCmFfxCpTfE8hZN00MnUEdohOiISZflCPbrUwAAA
```

AAA AKCAMIIEcjCCA1qgAwIBAgICC5gdQYJKoZIhvcNAQELBQAwdTELMakGA1UEAxMCVEEwHhcNM
TQx MjAzMTgwODMyWhcNMTQxMjEwMTgwODMyWjAzMTEwLWYDVQQDEyh1Y2Y0NjhkMDY1MTMyNzFmN
Tkz MjZhNjQ2MGZmOTFhYTNIINGU2Njk4MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEak
iYz EpnsqHIPNEl/LvJmfZfOYzRlhv0Ewqg/RLi6XsE5dhWi0YAifLbz0v/PfAjmJJFO6STsXkmc5
Cpp PoAl2+Ffx9Zujzy95hCNMqNgPSSqA92eAstLJALlvWrlygtQEbv/hIjetDOEY/fL49gajyuKg
hOh +zgeEUCVhdiArEj/4j5ElvI7flwJjLP8SI36IwlKoz6cd88Gm8bLQRURafe2lKW0quJk0RHOn
PZk babWuiiByoU24DCSy1+TBY4mEK6bilR0iONqeYfaSurxvWcDh8V6gNikiB+tfwRxrIO0lRTnK
nli hIe2OC5mkP2gMY5ZUyynJZnS3Or+CY3IcQIDAQABo4IBtDCCAbAwHQYDVR0OBByEFoz0aNB1E
ycf WTJqZGD/kao7TmaYMB8GA1UdIwQYMBaAFHeCG6FS5fvWxGw+lawrJ6kQpRTVMA4GA1UdDwEB/
wQE AwIHgDBFBggrBgEFBQcBAQQ5MDcwNQYIKwYBBQUHMAKGKWh0dHA6Ly9iYW5kaXRvLnJpcGUub
mV0 L3Jwa2ktY2EvdGEvdGEuY2VyMGYGCCsGAQUFBwELBFowWDBWBggrBgEFBQcwC4ZKcnN5bmM6L
y9i YW5kaXRvLnJpcGUubmV0L3JlcG8vNzc4MjFiYTElMmU1ZmJkNmM0NmMzZTk1YWMyYjI3YTtxM
GE1 MTRkNS5tZnQwWwYDVR0fBFQwUjBQoE6gTIZKcnN5bmM6Ly9iYW5kaXRvLnJpcGUubmV0L3Jlc
G8v Nzc4MjFiYTElMmU1ZmJkNmM0NmMzZTk1YWMyYjI3YTtxMGE1MTRkNS5jcmwwGAYDVR0gAQH/B
A4w

```

DDAKBggrBgEFBQcOAjAhBggrBgEFBQcBBWEB/wQSMBAwBgQCAAEFADAGBAIAAgUAMBUGCCsGA
QUF
BwEIAQH/BAYwBKACBQAwdQYJKoZIhvcNAQELBQADggEBAAjCpzNzjj7QGhmIG3Elt49cHUJe8
65w
y2Uq3ZKW2aZgA5It29D07XlsHO8tM0EwVXTxsBbpdkiEnzQ4G8Zx/ZI09vLSJ8ZjzSh42QeMa
Nt6
6zslilaw9rQcm/5jwxN18BRniwU/oavfRbn36AhfCmpegiI/4DTZWji63wucRrYHThZm6Zajn
HKU
DTlviomKZoZZDAUB4oQ7pN/Mw+tlK9F50VKz+9i3tnVhyt5wVaoEn/4sGRAL680A8Su0MKiyc
69t
3DYqnvSgYtFNiBbHhNYooBpraylh5r7WngBxfm+VJYkSaPxU8T6sSz/Capt+1S2UWJGTcFaZl
251
bm8nmXcAADGCAawwggGoAgEDgBTs9GjQZRMnH1kyamRg/5Gq005mmDANBglghkgBZQMEAgEFA
KBr
MBoGCSqGSIB3DQEJAZENBgsqhkiG9w0BCRABGjAcBgkqhkiG9w0BCQUxDxcNMTQxMjAzMTgwO
DMY
WjAvBgkqhkiG9w0BCQQxIgQgOUbuFjfSw4aMeIgLLDmT5xI7D05/mH6zVETECTmZwb0wDQYJK
oZI
hvcNAQEBBQAEggEAbhfERg8rgzy0GAIPDKj5kNk+owpm7WnRDiUo+6Y30zfKKjFhh1L+N0Ei7
b6q
r934eqEoac23wycF/Ale3+d4PolzvFrmln9rIia4BaD8GiUle6FEHd5njS7jOt5Kuej64yDFC
Htv
ipt8tGFik4MpvEmP5EOhZlcU/sErvlpdEsxQCaLsb6JUbIvoIHnWGXHE54QXkBvlucUSxypRo
qW3
SnAX0vo0F1YNrSDe05So3pjJSmNHOUFFnxZMja+1IMMWTfYlbKQJNpLlrb9a/uarfIL9BrGOD
WqE
dzQh+k3QkTAUojq+YADL+ix00eg2zpPm+eEU1F2+bGP2M5rbaUfqngAAAAA==
</publish>
<publish uri="rsync://bandito.ripe.net/repo/77821ba152e5fbd6c46c3e95ac2b27a
910a514d5.crl">
MIIBnZCBiAIBATANBgkqhkiG9w0BAQsFADANMQswCQYDVQQDEwJUQRcNMTQxMjAzMTgwODMYW
hcN
MTQxMjAzMTgwODMYWjAVMBMCAguXFw0xNDEyMDMxODA4MzJaoDAwLjAFBgNVHSMEGDAWgBR3g
huh
UuX71sRsPpWsKyepEKUU1TALBgNVHRQEBAICC5YwDQYJKoZIhvcNAQELBQADggEBAFQHR/2id
s7e
hmfNX+PmyePSN2EM1fBMLwMud6dqyBF42iNa8N0H/jxMAkgm7SS98TUupZglaIwqxLwGakFS6
VeD
+zCnCGEeMULXTpZaICDxMxJuJLBOvbqP2amPxWJ22g0+gTXM9KPAoWlNyAiMaNUP+nawjyfMz
Q4c
WJjiilkrHnIhiu9cZwEh9Ns/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:

- o 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.
- 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 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.

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-06f83ad3ca16/default/671570f06499fbd2d6ab76c4f22566fe49d5de60.mft" hash="226AB8CD3C887A6EBDDDF317F2FAFC9CF3EFC5D43A86347AC0FEFFE4DC0F607E">
    MIAGCSqGSib3DQEHAqCAMIACAQMDzANBglghkgBZQMEAgEFADCABgsqhkig9w0BCRABGqCAJ
IAE
    gYkwgYYCaguWGA8yMDE0MTIwMzE4MDg0MFoYDzIwMTQxMjA0MTgwODQwWgYJYIZIAWUDBAIBM
FMw
    URYsNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyNTY2ZmU0OWQ1ZGU2MC5jcmwDIQD1h9mcw
KzN
    70He/gIMVxsZGJlIXLh/TGzkaNTXxLixmwAAAAAAKCAMIIFGjCCBAKgAwIBAgICC5YwDQYJK
oZI
    hvCNAQELBQAwMzExMC8GA1UEAxMoNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyNTY2ZmU0O
WQ1
    ZGU2MDAeFw0xNDEyMDMxODA4NDBaFw0xNDEyMTAxODA4NDBaMDMxMTAvBgNVBAMTKDMzMzI1Y
zA4
    NmEwOWEyOTJkYzQxMzkwODA2ZDA4NTMxM2E1MTQwggeiMA0GCSqGSib3DQEBAQUAA4IBD
wAw
    ggEKAoIBAQCaklG7912fuezSWPpPqER3aZaP4HShGiIiRWeJLOYKpklAeS08kRa9R+yDCa9CM
i1B
    lewW/C5Coomb9teVZRg31YkpZlXqqNGg8GVesNMJX3ryuizQ+WRUcgwJoakqWH7wPu5zdfFj4
Cpk
    BpgJF+6TBYwXTjAmxfFP0hm0QLWCLxEdlgBGEdBmOogHqfOZHU95GLjllzsPRmR13kyh7BbYM
ie+
    ENJAqqKBlQvW86xPEDMJKUC0uQDnTPCZQBqFwElxrgUAuSCfJMUguAE8clsshOfE8ROF9t6NI
BxK
    oxA+PTYCpctdCBtdCFyWn/SLp1pb3gIA6xP9ESGLRHNumPL3AgMBAAGjggI2MIICMjAdBgNVH
Q4E
    FgQUMxJcCGoJopLcQRtTkIBtCFMTpRQwHwYDVR0jBBgwFoAUZxVw8GSZ+9LWq3bE8iVm/knV3
mAw
    DgYDVR0PAQH/BAQDAgeAMGcGCCsGAQUFBwEBBFswWTBxBggrBgEFBQcwAoZLcnN5bmM6Ly9iY
W5k
    aXRvLnJpcGUubmV0L3JlcG8vM2E4N2E0YjE0YTYzLWFKMGYtMDZmODNhZDNjYTE2L
2Rl
    ZmF1bHQvMIGWBggrBgEFBQcBCwSBiTCBhjCBgwYIKwYBBQUHMAuGd3JzeW5jOi8vYmFuZG10b
y5y
    aXB1Lm5ldC9yZXBvLnNhODdhNGIxLTZlMjItNGE2My1hZDBmLTA2ZjgzYWQzY2ExNi9kZWZhd
Wx0
    LzY3MTU3MGYwNjQ5OWZiZDZkNmFiNzZjNGYyMjU2NmZlNDlkNWRLNjAubWZ0MIGJBG9VHR8Eg
YEw
    fzB9oHugeYZ3cnN5bmM6Ly9iYw5kaXRvLnJpcGUubmV0L3JlcG8vM2E4N2E0YjE0YTYzLWFK
TYz
    LWFkMGYtMDZmODNhZDNjYTE2L2RlZmF1bHQvNjcxNTcwZjA2NDk5ZmJkMmQ2YWI3NmM0ZjIyN
TY2
    ZmU0OWQ1ZGU2MC5jcmwwGAYDVR0gAQH/BA4wDDAKBggrBgEFBQcOAjAhBggrBgEFBQcBBwEB/
wQS
    MBAwBgQCAAEFADAGBAIAAgUAMBUGCCsGAQUFBwEIAQH/BAYwBKACBQAwDQYJKoZIhvcNAQELB
QAD
    ggEBAEOldSFDN4wZqtZ0fWo5G0YVN+mtk6tKhHPFwX7ydTofnHZkE2p07C93XcgPcP4zLUBPt
5kS
    aH+0vcBxs9Vg//58cHRUEHhls90/XcS8RXCvKniga+9NB5s4oi0+i/gDU3eOUqE/jqSJAJAS+
Ehi
    tvNh0LuLrW92NrOfbYDk29how3uxK4JucIAQ05i6317EAeQp3WeI8nVzB9Rfrkv+PSV+57mSX
XtJ
    /jWu3kyjvsxRjeUL3Im2Z1F48zfVF6pVaDT7ib4YbKOyAQTMpi4W6NZwgQskda9B8/0qV/d+2
JrC
    m3Ozm0t21aoH8xKP/OC33bBXLcXUvkVqvB/Y+TUXfAEAADGCAawwggGoAgEDgBQzElwIagmik
txB
    Gl0QgG0IUxOlFDANBglghkgBZQMEAgEFAKBrMBBoGCSqGSib3DQEJAZENBgsqhkig9w0BCRABG
jAc
    BgkqhkiG9w0BCQUxDxcNMTQxMjA0MTgwODQwWjAvBgkqhkiG9w0BCQQxIgQgdNPMbp9lJJHNM
mIz
```

00ff73VkVFYWo2Uf6/b4zIzFZucwDQYJKoZIhvcNAQEBBQAEggEAXHNHm+DUD1s9IQMewvKso
 NGi
 fXL2jG3yfuGys5x1aJji3bIKGiU+weHmnP9aoH9UFRLk6pW1wFOS0+6M87UD8cU17w9F10e02
 58S
 9p7xHMGbrYqXrX9OucMqiN4M+ThDzyDXnfNAOgw5XNJU9KRndS9vyXS6lcvD7JTOhkyqKsrqH
 XlM
 0pX+rYFtrF2RNjB54veooSkcKGojXReLttZbvVKWKwkVg2RJy4tt7MOGU0Q6qa/J5S7O6xvwP
 jkY
 yCFvrHm+CgeXoR/3Hg/Rk/NdsK4Klu5dXhRh3KYv4P/hnGSD83aFE9t/DTicvl6SjaXFCTLtJ
 lTX
 BqSW7wgZ6OoLxwAAAAAAAA==
 </publish>
 <publish uri="rsync://bandito.ripe.net/repo/3a87a4b1-6e22-4a63-ad0f-06f83ad
 3cal6/default/671570f06499fbd2d6ab76c4f22566fe49d5de60.crl" hash="2B551A6C10CCA0
 4C174B0CEB3B64652A5534D1385BEAA40A55A68CB06055E6BB">
 MIIBxTCBrgIBATANBgkqhkiG9w0BAQsFADAzMTEwLWYDVQQDEYg2NzE1NzBmMDY0OTlmYmQyZ
 DZh
 Yjc2YzRmMjI1NjZmZTQ5ZDVkZTYwFw0xNDEyMDMxODA4NDBaFw0xNDEyMDQxODA4NDBaMBUwE
 wIC
 C5UXDTE0MTIwMZE4MDg0MFqgMDAuMB8GA1UdIwQYMBaAFGcVcPBkmfvS1qt2xPILZv5J1d5gM
 AsG
 A1UdFAQEAgILl jANBgkqhkiG9w0BAQsFAAOCAQEAIbL+8connmKLeypzs/P6FOHv8elmLp6dF
 lId
 SDpZT7p6y9xLZkvuow39XOs6NB1AOA+92uao9hEV1XuEBGP98nsx0frL8HJtKcEn0q5LGqA4Y
 eBG
 n28+Ldvlh4DetiKvFpsKW/VYqjRumHcgTdWpESY/f9hH3xW6JCggh5cFGFF/dCsCdGT1v+m53
 zf4
 Dlz8KhRDEaok3UMycX9XUWMB5HSwf05Qrha2LIff66uk6AQQEmV9ZiBq3IdbkdNd90TIVDMvn
 SW/
 p9XygdX8azaE2+hsOc9J7+E2kBuu4isLhvfZmChtFpxIUrljQRD4iUil8/xmB6MAIptoF1Es1
 pAI
 aw==
 </publish>
 <withdraw uri="rsync://bandito.ripe.net/repo/3a87a4b1-6e22-4a63-ad0f-06f83a
 d3cal6/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 withholding 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 propagation 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 propagation 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 available, 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: Informational
Expires: August 15, 2013

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
February 11, 2013

RPKI Repository Analysis and Requirements
draft-tbruijnzeels-sidr-repo-analysis-00

Abstract

The current RPKI Resource Certificate Repository Structure (RFC6480 & RFC6481) uses rsync as both a delta and transfer protocol. Concerns have been raised about the scalability of this repository and the reliance on rsync. This document provides an analysis of these concerns and formulates requirements for future work.

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 15, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	2
2. Introduction	3
3. Concerns With current repository	3
3.1. Scalability of rsync(d) deltas	3
3.2. Update frequency and propagation times	3
3.2.1. Migrating to another ASN	4
3.2.2. Error in ROA	4
3.2.3. BGPsec	4
3.3. Lack of rsync standard and implementations	4
3.4. Inconsistent Responses	5
3.5. Single publication point per CA	6
3.6. Scalability through hierarchical fetching	6
4. Delta Protocol Requirements and Recommendations	7
4.1. Transport Agnostic	7
4.2. Support Publication Sets	7
4.3. Support non-hierarchical repository lay-out	7
4.4. Expected factors affecting repository load	7
4.4.1. Disclaimer	7
4.4.2. Size aspects of the global RPKI	7
4.4.3. Churn	10
4.4.4. Number of Relying Parties	10
4.4.5. Fetch frequency of Relying Parties	11
4.5. Expected RPKI Repository Requirements	11
4.5.1. Objects and Relying Parties	11
4.5.2. Update related throughput	12
4.5.3. Update related concurrency	12
4.5.4. Update related traffic volume	12
4.6. Reduce Load on Central Repositories	13
4.7. Update notifications	13
4.8. Reduce Churn	13
4.9. Signed Deltas	13
5. Security Considerations	14
6. Acknowledgements	14
7. Normative References	14
Authors' Addresses	14

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

The current RPKI Resource Certificate Repository Structure (RFC6480 & RFC6481) uses rsync as both a delta and transfer protocol and recommends that repositories be set up in a hierarchical way such that relying parties (validation tools) can fetch all updates in a single repository efficiently while performing top-down validation.

This structure has its benefits. In particular it has allowed for early deployment of RPKI without the need to re-invent a delta protocol, and this has allowed early adopters of RPKI to build up operational experience more quickly. The delta protocol also has benefits for relying party tools, allowing them to quickly retrieve what's new in a repository limiting fetch time and bandwidth usage.

Having said this, operational experience, as well as lab testing, have shown that there are concerns with regards to the current infrastructure that justify that the WG thinks about improvements in this space.

3. Concerns With current repository

3.1. Scalability of rsync(d) deltas

Rsync is a very efficient tool when used 1:1 between a client and server. The problem is that in a globally deployed RPKI we can expect in the order of 40k clients, roughly corresponding to the number of ASNs, to connect regularly to a repository server.

When the rsync built-in delta protocol is used (recursive fetching), the server is computationally involved in calculating the delta for each connected client. Performance measurements in a lab have shown that the maximum number of clients that can be processed per second (throughput), and the maximum number of concurrent clients are both linearly dependent on the repository size. The throughput is limited by server CPU. Concurrency is limited by server memory.

As a result a sufficiently large repository has to invest heavily in running multiple rsyncd instances to cope with the expected regular load of a large number of clients, and to counter the risks of DDoS attacks.

3.2. Update frequency and propagation times

The retrieval of signed objects is described in RFC6480 (section 6). There are no formal limits imposed by this informational RFC on the update frequency, but to prevent the overloading of repository servers as described above, the typical update interval of current tools is between 1-24 hours.

In previous discussions in the WG it was suggested that human scale propagation times (i.e. up to 24 hours) are good enough for the problem that we are trying to solve. There are however good reasons why much faster retrieval of newly signed objects is desirable.

3.2.1. Migrating to another ASN

In this scenario, a ROA exists for a prefix and ASN, but the prefix needs to be announced from another ASN.

In many cases the RP can foresee this and create an appropriate ROA well in advance, but there are also failure cases possible where this is not foreseeable.

3.2.2. Error in ROA

The CA operator made a mistake when it created a ROA. The ROA causes announcements that should be considered VALID to appear as INVALID, or vice versa. The CA would like to take appropriate action and revoke the ROA or issue additional ROAs. However, RPs that have received the mistaken ROA may not see these updates for some time.

3.2.3. BGPsec

The BGPsec protocol is still being discussed. However there are indications that it would be desirable if propagation times for new router certificates and CRLs could be reduced. In particular in the context of:

- o Planned router key roll-overs
- o Unplanned roll-out of new router hardware with new keys
- o Replay protection strategies that rely on having shorter cycles & propagation times for router certs and keys

3.3. Lack of rsync standard and implementations

There is only one known implementation of rsync and the standard is not described by any RFC. The implementation is non-modular, making it impossible to use the code as a library even when coding in the same language.

As a result all current implementations of relying party tooling have had no option but to use rsync as a pre-installed external process.

This has several major draw backs for the quality of implementations:

- o RP tools require that rsync is installed on a system, and they can not assume which version is installed.
- o Because there is only one implementation all RPKI repositories can be affected by a single bug or exploit in rsync.
- o Calling an external process is expensive limiting the benefits that can be gained from parallel processing.
- o Parsing downloaded objects is inefficient -- objects have to be downloaded to disk first before they can be read and parsed.
- o Dealing with errors is complicated -- exit codes are not always clear, stderr may have to be parsed. Exit codes and messages are not guaranteed to be the same across rsync versions.

3.4. Inconsistent Responses

An 'inconsistent' set of objects is a set of retrieved objects for a CA Certificate where there differences between the objects retrieved and the objects mentioned on the corresponding manifest. If any objects are missing, or if additional objects not mentioned on the manifest are found, or if any of the objects does not match the sha256 hash mentioned on the manifest, then the set as a whole is considered inconsistent. RFC6486 has text advising RPs on possible ways to treat each of these cases. However, there is a large degree of uncertainty as to how different RP tools, and operators, will deal with these corner cases because most decisions are left to local policy. This can lead to inconsistent and possibly surprising differences in the validation of RPKI data.

Missing ROA objects can be particularly problematic because other ROAs, that can be found and validated, may invalidate announcements that would have been marked as valid by these missing ROAS. Additional ROA objects are confusing because to the RP it's not clear whether this ROA was intentional and the MFT is out of date, or not.

The use of rsync as a delta protocol is problematic in this context, because rsync is non-transactional. As a result an RP may get partially updated CA repository objects if it happens to fetch while the objects on disk are being updated. This is confusing to the RP who can not tell the difference between this and a persistent error on the publisher side, or an attack involving partial replay or withholding of objects.

All of this adds up to a repository infrastructure and corresponding validation rules that leave a high degree of uncertainty in case of corner cases. The authors believe that it would be better to (1) improve the standards so that these corner cases are less likely to occur, and (2) formulate much stricter validation rules so that the uncertainty with regards to how RPs may deal with corner cases is further reduced.

3.5. Single publication point per CA

In the current design only publication point per CA is envisioned.

Even though such a publication point may employ various techniques to achieve high-availability, this leaves concerns with regards to:

- o Attacks on DNS for the publication point
- o A contractual tie-in between CA and publication server, with no way for planned migration (while staying up to date)
- o Failure of the publication server
- o (Legal) attacks on the publication server

3.6. Scalability through hierarchical fetching

The notion that child CAs can publish in a sub-directory of their parent CA publication point has been suggested as mitigation strategy for scalability of fetching RPKI data using rsync.

There are a number of reasons why this hierarchical model may not be advisable or even possible:

- o The parent CA may not wish to imply responsibility over objects issued by its child
- o Recursive rsync fetches on sufficiently large repositories are expensive. The parent CA may have no choice but to disallow recursive fetching to mitigate its DDoS vulnerabilities

- o CAs may wish to publish their own content, or they may wish to publish their content in a repository that is provided by an organisation other than their parent CA.

4. Delta Protocol Requirements and Recommendations

4.1. Transport Agnostic

A future delta protocol should be transport agnostic, allowing agility in future transport protocols between RPs and repositories, and sharing of deltas between RPs.

4.2. Support Publication Sets

A future delta protocol should enable CAs to publish new objects as a set so that errors in evaluating route origin validity as a result of incomplete information may be avoided as much as possible.

4.3. Support non-hierarchical repository lay-out

The scalability of a future delta protocol should not depend on a hierarchical repository lay-out. This is particularly important if one considers the possibility of third party publication servers and/or the possible use of mirror repositories. In both cases the CAs for which objects are published can most likely not be considered children of the publication server, or each other.

4.4. Expected factors affecting repository load

4.4.1. Disclaimer

The numbers cited below reflect our best current estimates based on relevant statistics currently at our disposal. They are intended to provide context for load testing proposed solutions.

We are of course open any suggestions and real world statistics that can improve these estimates.

4.4.2. Size aspects of the global RPKI

4.4.2.1. Mirroring

For the purpose of scalability it would be prudent to assume that mirroring should be supported in the RPKI to the point where one publication server can, in principle, mirror the complete global RPKI. In reality this may not happen to this extent, but any design that can support this should be adequate to support smaller numbers.

4.4.2.2. Number of CAs in the global RPKI

Assuming that only current RIR member organisations that are holding IPv4, IPv6 and/or ASN resources would act as Certificate Authorities (CA), the expected number of CAs in the global RPKI is expected to be around 50.000. However, if one also considers holders of Provider Independent (PI) resources this number may be larger. For reference: the RIPE NCC has roughly 25.000 PI prefixes registered, vs just roughly 9000 regular members. If these numbers are similar for all regions, and we assume the 'worst case' where all PI holders have their own CAs, then we are looking at number that is roughly four times larger: i.e. 200.000 CAs.

Note that each organisation will most likely find all their resources on one certificate, however in case an organisation holds resources from multiple parent sources more than certificate may be needed. For the moment we will assume that the number of CA certificates per organisation will be close to 1.

For each CA certificate 4 objects will be published in the global RPKI: the CA certificate itself (by the parent CA), one manifest, one CRL and one ghostbuster record.

4.4.2.3. Number of ROAs in the global RPKI

The number of ROAs in the global RPKI does not depend on the number of CAs. A small organisation may have only 1 ROA, while a large organisation will need many. Instead it is expected that the number of ROAs is related to the number of intended announcements that are seen in the global BGP. The current routing table has roughly 500.000 such announcements, but the size of the table has been growing steadily.

It should be noted that ROAs can be used to authorise more than one announcement, but there are restrictions:

- o The ASN must be the same.
- o The prefixes must all be held by the CA.
- o Furthermore the prefixes must occur on the same parent certificate. In other words: if an organisation has signed resources from more than one source they can not be aggregated on the same ROA.

Statistics for the RIPE region indicate that an aggregation factor of 3 announcements per ROA is reasonable. This would put the expected number of ROAs in the order of 200.000.

4.4.2.4. Number of router certificates in the global RPKI

The number of router certificates depends on the number of keys that will be used by BGPsec speaking routers.

At a specific ASN, different physical BGPsec speaking routers MAY use the same key, and therefore may require only one certificate for that key. On the other hand to support BGPsec roll-overs it may be advisable to publish not one, but two keys at the same time. Plus some operators may choose to use unique keys per physical router.

All in all it is not entirely clear to the authors how many certified keys may be, but on list numbers as high as 2.000.000 have been mentioned.

4.4.2.5. Total number of objects in the global RPKI

Using the number of objects cited in the previous sections, we can describe the total number of objects in the RPKI with the formula:

$$O_{total} = \#CA_{organisations} * \#Avg_CA_{cert_per_organisation} * 4 + \#ROAs + \#Router\ Certs$$

$$O_{total} = 200k * \sim 1 * 4 + 200k + 2M = 3M$$

4.4.2.6. Total size of objects in the global RPKI

Based on the current repositories deployed by the RIRs we find these average sizes for different object types:

type	size (bytes)	size in model
CA certificate	1416	1.5 kB
Manifest	1951	2 kB
CRL	692	0.7 kB
ROA	1846	2 kB
Ghostbuster record	unknown, expect similar to ROA	1.5 kB
Router certificate	unknown, expect similar to CA certificate	2 kB

We use rounded off decimal numbers for our calculations for simplicity, and because our predictions are intended to give an idea of the expected order of magnitude of the repository size only.

Using these numbers we can predict a global repository size with the formula:

$$\text{Stotal} = \# \text{CAorganisations} * (\text{CA_certificate_size} + \text{MFT_size} + \text{CRL_size} + \text{GB_size}) + \# \text{ROAs} * \text{ROA_size} + \# \text{Router Certs} * \text{Router_Cert_size}$$
$$\text{Stotal} = 200\text{k} * (1.5\text{k} + 2\text{k} + 0.7\text{k} + 2\text{k}) + 200\text{k} * 2\text{k} + 2\text{M} * 2\text{k}$$
$$\text{Stotal} = 4.6\text{G}$$

4.4.3. Churn

The daily churn in the RPKI, i.e. the amount of new objects we're expected to see, per 24 hours is another important factor to consider in the context of scalability

The current RIR managed RPKI services typically update MFT and CRLs for each CA every 8 hours, accounting for a churn of $200\text{k} * 2 * (24 \text{ hours} / 8 \text{ hours}) = 1.2 \text{ M objects per 24 hour}$ (833 per minute). The volume of this churn is expected to amount to $200\text{k} * (2\text{kB} + 0.7\text{kB}) * (24 \text{ hours} / 8 \text{ hours}) = 1.6 \text{ GB per 24 hour} = 1.1 \text{ MB per minute}$.

The expected churn in ROAs and router certificates are expected to depend on:

- o The amount of new planned announcements in BGP
- o The average number of routers requiring new keys being rolled-out daily.
- o BGP Sec key roll-overs

We have no clear idea about these numbers at this time, but we expect this number to be relatively small compared to the churn rate caused by republishing MFTs and CRLs.

4.4.4. Number of Relying Parties

It seems plausible that in a full deployment scenario each ASN will run at least two RP tool instance (one back-up).

There are currently around 40.000 ASNs in the global BGP, so this would suggest a number of 80.000 distinct client RPs accessing repositories.

Others have suggested that RPs can use (file) sharing techniques to reduce their dependency on central repository servers. If this

approach would be deployed this could reduce the number of RPs that central repository servers have to serve. We expect though that this sharing will be mostly used to ensure redundancy with an ASN, and much less between ASNs.

4.4.5. Fetch frequency of Relying Parties

The repository servers have little control of the fetch frequencies used by Relying Parties. As mentioned in section 3.2 Relying Parties have an interest in fetching new information much more frequently than they do currently. It's not clear right now what frequency will be most common in a full deployment scenario. We expect though that the desired update frequency will be in the order of every ten minutes. This seems to be in-line with operator time scale changes that would have to be made in BGP and the RPKI.

4.5. Expected RPKI Repository Requirements

4.5.1. Objects and Relying Parties

It should be noted that repository servers have no control over relying parties. RPs are responsible for their own infrastructure and keeping up to date. Well functioning RPs will try to stay up to date at all times, while avoiding to overload the server(s). Badly configured RPs may however fail to retrieve updates, or they may insist on checking for updates at a well-above average rate and cause additional server load.

Having said that we believe that we can stipulate some ball park parameters that large repositories should be prepared to deal with based on the estimates mentioned in the previous section.

The numbers below all assume averages of well behaved RPs. In reality repositories will have to deal with peak loads that may result from a number of different factors, like:

- o A large number of updates is available
- o The number of RP connections is not evenly distributed
- o There is an attack on the server

Defining these factors in formulas is however fairly complicated. The authors believe that it is a more pragmatic and useful strategy to take the naive estimates defined below as a starting point and require that new protocols are load tested to a degree where we can be confident that new implementations will be able to meet the normal load requirements easily, as well as peak load conditions that may exceed normal load by factors of 5-10.

4.5.2. Update related throughput

We define "throughput" as the total number of RP connections that the repository can server per minute. Note that this does not imply anything about the time each connection takes.

By definition the server has to be able to process a number of connections per time unit that is at least equal, and preferably comfortably bigger, than the number of new connections that are expected over that time unit. Failure to meet this number will inevitably lead to a build-up of client connections to the point where the server will no longer be able to accept new connections

Based on 80k RP tools fetching updates every 10 minutes we may assume that a throughput number of 8k connections / min. is the bare minimum that needs to be supported.

4.5.3. Update related concurrency

Another interesting load factor is given by the number of expected concurrent connections. A naive formula for this number is given by:

$$\# \text{Concurrent connections} = \# \text{New Connections (conn / minute)} * \# \text{Avg processing time (min / conn)}$$

E.g. if it would take 30 seconds to process the average connection, we would need to support:

$$8\text{k conn/min} * 0.5 \text{ min/conn} = 4\text{k concurrent RPs}$$

4.5.4. Update related traffic volume

Based un RPs getting deltas alone we expect that the volume of data that the repository server has to serve per minute can be determined by the formula:

$$\text{Vol_min} = \text{Churn_vol_min} * \# \text{RPs}$$
$$\text{Vol_min} = 1.1 \text{ M/min} * 80\text{k} = 88 \text{ GB/min}$$

4.6. Reduce Load on Central Repositories

In a full deployment scenario a large number of RPs are expected to approach a single repository server regularly. Estimates of how large this number of RPs is, and how regularly they will fetch updates, vary. However as a starting point one might expect one RP tool for each ASN, currently 40k, to fetch updates every five minutes. Whatever the real numbers may be it should be noted that the repository server has very little control over these numbers.

Because of this it makes sense to look into a delta protocol where the number of clients and frequency of fetching, has the least possible effect on the central repository server. E.g by enabling pre-computing of updates and offloading to caches or CDNs.

Although this may result in a protocol that causes the Relying Party to do more work, the trade-off of offloading CPU cycles to a large number of frequently polling RPs as opposed to spending CPU on the server is expected to scale much better.

4.7. Update notifications

Higher update frequencies and shorter propagation times are desired. On the other hand it would also be good if unnecessary synchronisation attempts were prevented to reduce load. For this reason a delta protocol would do well to support update notifications to RPs. Both push and poll based strategies may be used for this purpose.

4.8. Reduce Churn

A large part of the churn in the RPKI is caused by the regular republishing of Manifests and CRLs. If this frequency could be reduced without compromising security, such as an RP's sensitivity to replay attacks, then the total load on repository servers could be reduced significantly.

4.9. Signed Deltas

It should be noted that although RPs retrieve objects from untrusted sources, these objects are cryptographically validated. In other words a publisher, or monkey-in-the-middle, can not mislead the RP and generate valid objects without having access to the associated private keys. Having said that, this still leaves RPs vulnerable to attacks where information is withheld or replayed. RPs can notice such attacks if they rely on manifests to inform them about:

- o which objects should be expected

- o when the next update is expected at the latest

If deltas were signed it would be possible for RPs to detect attacks or transport errors sooner. However, signing deltas comes at a cost of complexity. In particular it will be difficult to communicate keys used for signing in a secure and dynamic (allow rolls) way. The benefit seems too limited to warrant this.

5. Security Considerations

TBD

6. Acknowledgements

TBD

7. Normative References

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

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