

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: March 28, 2013

W. George  
Time Warner Cable  
September 24, 2012

BGPsec Considerations for AS Migration  
draft-george-sidr-as-migration-00

Abstract

This draft discusses considerations for supporting and securing a common method for AS-Migration within the BGPsec protocol.

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 March 28, 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
1.1. Requirements Language . . . . .	3
2. General Scenario . . . . .	3
3. RPKI Considerations . . . . .	4
3.1. Origin Validation . . . . .	4
3.2. Path Validation . . . . .	5
3.2.1. Outbound announcements (PE-->CE) . . . . .	5
3.2.2. Inbound announcements (CE-->PE) . . . . .	6
4. Requirements . . . . .	6
5. Acknowledgements . . . . .	7
6. IANA Considerations . . . . .	7
7. Security Considerations . . . . .	7
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	8
Author's Address . . . . .	8

## 1. Introduction

There is a common method of managing an ASN migration using some BGP knobs that while commonly-used are not formally part of the BGP4 [RFC4271] protocol specification and may be vendor-specific in exact implementation. In order to ensure that this behavior is understood and considered for future modifications to the BGP4 protocol specification, especially as it concerns the handling of AS\_PATH attributes, the behavior and process has been defined in draft-ga-idr-as-migration [I-D.ga-idr-as-migration]. Accordingly, it is necessary to discuss this de facto standard to ensure that the process and features are properly supported in BGPsec [I-D.ietf-sidr-bgpsec-protocol], because it is explicitly designed to protect against changes in the BGP AS\_PATH, whether by choice, by misconfiguration, or by malicious intent. It is critical that the BGPsec protocol framework is able to support this operationally necessary tool without creating an unacceptable security risk or exploit in the process.

### 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. General Scenario

The use case being discussed in draft-ga-idr-as-migration [I-D.ga-idr-as-migration] is as follows: For whatever the reason, a provider is in the process of merging two or more ASNs, where eventually one subsumes the other(s). Confederations RFC 5065 [RFC5065] are *not* being implemented between the ASNs, but vendor-specific configuration knobs are being used to masquerade as the old ASN for the PE-CE eBGP session, or to manipulate the AS\_PATH, or both. While BGPsec [I-D.ietf-sidr-bgpsec-protocol] does have a case to handle standard confederation implementations, it may not be applicable in this exact case. The reason that this may drive a slightly different solution in BGPsec than a standard confederation is that unlike in a confederation, eBGP peers may not be peering with the "correct" external ASN, and the forward-signed updates are for a public ASN, rather than a private one, so there is no expectation that the BGP speaker should strip the updates before propagating the route to its eBGP neighbors.

In the following examples, AS200 is being subsumed by AS300, and both ASNs represent an SP network. AS100 and 400 represent end customer networks.

### 3. RPKI Considerations

Since the methods and implementation discussed in draft-ga-idr-as-migration [I-D.ga-idr-as-migration] are not technically a part of the BGP4 protocol implementation, but rather a vendor-specific optimization, BGPsec is not technically required to ensure that it continues functioning as it does today. However, this is widely used during network integrations resulting from mergers and acquisitions, as well as network redesigns, and therefore it is not feasible to simply eliminate this capability on any BGPsec-enabled routers/ASNs. What follows is a discussion of the potential issues to be considered regarding how ASN-migration and RPKI validation might interact.

Additionally, companies rarely stop with one merger/acquisition/divestiture, and end up accumulating several legacy ASNs over time. Since they are using methods to migrate that do not require coordination with customers, they do not have a great deal of control over the length of the transition period as they might with something completely under their administrative control like a key roll. This leaves many SPs with multiple legacy ASNs which don't go completely and cleanly away very quickly, if at all. As solutions are being proposed for RPKI implementations to solve this transition case, operational complexity and hardware scaling considerations associated with maintaining multiple legacy ASN keys on routers throughout the combined network have to be carefully considered. While part of the recommendation may be "SPs SHOULD NOT remain in this transition phase indefinitely because of the operational complexity and scaling considerations associated with maintaining multiple legacy ASN keys on routers throughout the combined network", this is of limited utility as a solution, and so every effort needs to be made to allow the transition period to be less onerous, on the assumption that it will likely be protracted.

#### 3.1. Origin Validation

In the scenario discussed, AS200 is being replaced by AS300. If there are any existing routes originated by AS200 on the router being moved into the new ASN, this is likely as simple as generating new ROAs for the routes with the new ASN and treating them as new routes to be added to AS300/removed from AS200. However, consider the situation where one or more PEs are still in AS200, and are originating one or more routes. When those routes arrive at PE1, which is now a part of AS300 and instructed to use replace-as to remove AS200 from the path, how does it handle routes originated from AS200? If the route now shows up as originating from AS300, any downstream peers' validation check will fail unless a ROA is \*also\* available for AS300 as the origin ASN, meaning that there will be

overlapping ROAs until all routers originating prefixes from AS200 are migrated to AS300.

[AUTHOR'S NOTE, remove before publishing: may need a citation to the specific text in the Origin validation spec RFC 6480 [RFC6480] or related documents that allows an overlap period on ROAs for transitions like this. also may need to rewrite this as a procedure in RFC2119 language, rather than a discussion. END NOTE]

### 3.2. Path Validation

BGPsec Path Validation requires that each router in the AS\_PATH cryptographically sign its update to assert that "Every AS listed in the AS\_PATH attribute of the update explicitly authorized the advertisement of the route to the subsequent AS in the AS\_PATH." Since this migration technique is explicitly modifying the AS\_PATH between two eBGP peers who are not coordinating with one another (are not in the same administrative domain), no level of trust can be assumed, and therefore it may be difficult to identify legitimate manipulation of the AS\_PATH for migration activities when compared to manipulation due to misconfiguration or malicious intent.

#### 3.2.1. Outbound announcements (PE-->CE)

When PE1 is moved from AS200 to AS300, it will be provisioned with the appropriate keys for AS300 so that it can begin forward-signing routes using AS300. However, there is currently no guidance in the BGPsec protocol specification on whether or not the forward-signed ASN value MUST match the configured "remote-as" to validate properly. That is, if CE1's BGP session is configured as "remote-as 200", the presence of "local-as 200" on PE1 will ensure that there is no ASN mismatch on the BGP session itself, but if CE1 receives updates from its remote neighbor (PE1) forward-signed from AS300, should the BGPsec validator on CE1 still consider those valid by default? If it does, is there any potential attack vector to consider?

If enabling strict validation that remote-AS and forward-signed-AS match is desirable, a possible alternative would be to retain the keys for AS200 on PE1, and forward-sign towards CE1 with AS200 and Pcount=0. However, this would mean passing a pcount=0 between two ASNs that are in different administrative and trust domains such that it could represent a significant attack vector to manipulate BGPsec-signed paths. The expectation for legitimate instances of Pcount=0 (to make a route-server that is not part of the transit path invisible) is that there is some sort of existing trust relationship between the operators of the route-server and the downstream peers such that the peers could be explicitly configured by policy to permit PCount=0 announcements only on the sessions where they are

expected, and otherwise reject them. For the same reason that things like local-as are used for ASN migration without end customer coordination, it is unrealistic to assume any sort of coordination between the SP and the administrators of CE1 to ensure that they will by policy accept PCount=0 signatures during the transition period, and therefore this may not be a workable solution.

### 3.2.2. Inbound announcements (CE-->PE)

Inbound is more complicated, because the CE doesn't know that PE1 has changed ASNs, so it is forward-signing all of its routes with AS200, not AS300. The BGPsec speaker cannot [MUST NOT??] manipulate previous signatures, and therefore cannot manipulate the previous AS\_Path without causing a mismatch that will invalidate the route. If the updates are simply left intact, the ISP would still need to publish and maintain valid and active public-keys for AS 200 if it is to appear in the BGPsec\_Path\_Signature in order that receivers can validate the BGPSEC\_Path\_Signature arrived intact/whole. However, if the updates are left intact, this will cause the AS\_PATH length to be increased, which as previously stated is undesirable. More discussion is needed to determine possible solutions that enable this transition without defeating the added security that BGPsec provides.

## 4. Requirements

PLACEHOLDER for a set of requirements, defining what SIDR Origin Validation/BGPSEC <RFC2119-word> do to enable this migration strategy to work securely. Need input from WG as to how to proceed.

Options:

1. Add implementation considerations and/or protocol changes to support AS-Migration into the BGPsec protocol doc or another existing BGPsec document
2. Use this draft to document how BGPsec and Origin Validation will need to handle the AS-migration procedure (move to PS status?), possibly as an update to the BGPsec protocol document
3. Explicitly prohibit/deprecate this process for AS-Migration (a la AS\_Sets) as something that cannot be properly secured within BGPsec
4. define a new/updated/improved standard method for asymmetric ASN migrations which is explicitly designed to operate within the bounds of BGPsec (instead of trying to make the existing implementations work with no changes) - probably in IDR

The author does not believe that option #3 is the correct course of action because this is in wide use among operators today, and no acceptable alternative exists to make the act of merging ASNs less onerous.

## 5. Acknowledgements

Thanks to Kotikalapudi Sriram and Shane Amante for their comments.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

This draft discusses a process by which one ASN is migrated into and subsumed by another. Because this involves manipulating the AS\_Path to make it deviate from the actual path that it took through the network, it is in some ways attempting to do exactly what BGPSEC is working to prevent. The BGPSEC implementation **MUST** be able to manage this legitimate use of AS\_Path manipulation without generating a vulnerability in the RPKI route security infrastructure that can be exploited by a malicious actor.

## 8. References

### 8.1. Normative References

[I-D.ga-idr-as-migration]

George, W. and S. Amante, "Autonomous System (AS) Migration Features and Their Effects on the BGP AS\_PATH Attribute", draft-ga-idr-as-migration-00 (work in progress), September 2012.

[I-D.ietf-sidr-bgpsec-protocol]

Lepinski, M., "BGPSEC Protocol Specification", draft-ietf-sidr-bgpsec-protocol-05 (work in progress), September 2012.

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

## 8.2. Informative References

- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

## Author's Address

Wesley George  
Time Warner Cable  
13820 Sunrise Valley Drive  
Herndon, VA 20171  
US

Phone: +1 703-561-2540  
Email: wesley.george@twcable.com





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

M. Lepinski, Ed.  
BBN  
October 22, 2012

BGPSEC Protocol Specification  
draft-ietf-sidr-bgpsec-protocol-06

## 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 two new BGP capabilities [6] that allow a BGP speaker to advertise to a neighbor the ability (respectively) to send or to receive BGPSEC update messages (i.e., update messages containing the BGPSEC\_Path attribute).

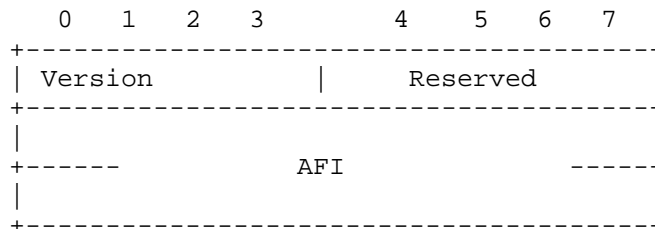
### 2.1. BGPSEC Send Capability

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 remaining four 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.

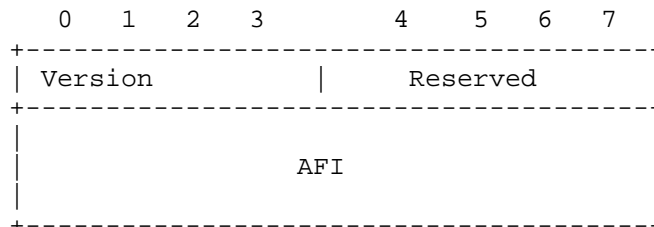
## 2.2. BGPSEC Receive Capability

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 Receive 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 remaining four 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 BGPSEC support. 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.

### 2.3. Negotiating BGPSEC Support

A BGP speaker sends the BGPSEC Send Capability (see Section 2.1) in order to indicate that the speaker is willing to send BGP update messages containing the BGPSEC\_Path attribute (for a particular address family). A BGP speaker sends the BGPSEC Receive Capability (see Section 2.2) in order to indicate that the speaker is willing to receive messages containing the BGPSEC\_Path attribute.

Note that if the BGPSEC 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 BGPSEC 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 Send Capability for a particular version of BGPSEC and a particular address family; and
- o The other peer has sent the BGPSEC Receive Capability for the same version of BGPSEC and the same address family.

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 capability to send BGPSEC messages, MUST also announce the capability for four-byte AS support [4]. If a BGP speaker sends the BGPSEC send 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

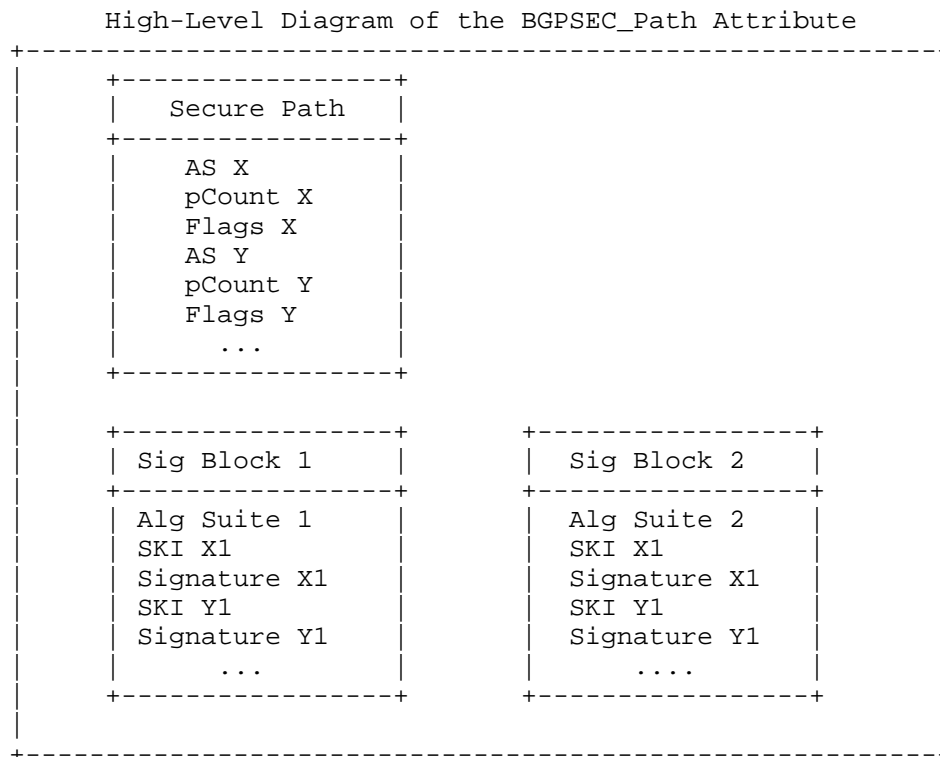
The BGPSEC\_Path attribute is a new optional non-transitive BGP path attribute.

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.)

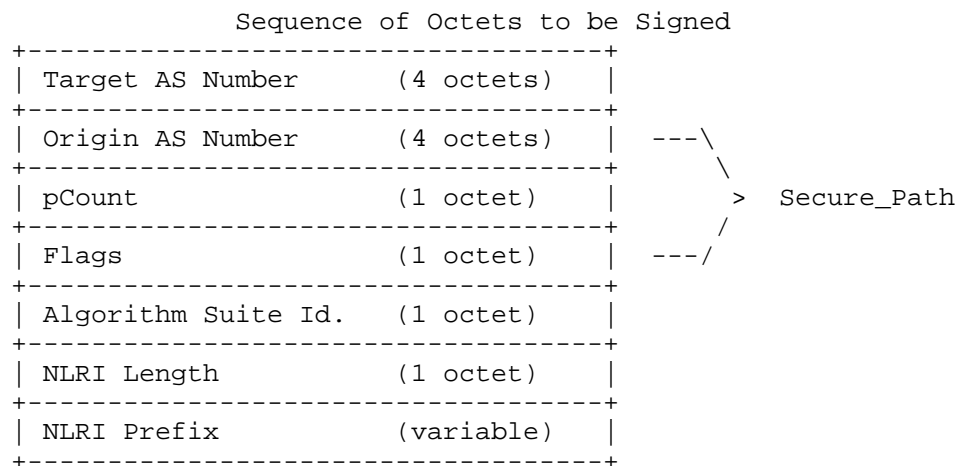
Typically, a BGPSEC speaker will use only a single algorithm suite, and thus create only a single Signature\_Block in the BGPSEC\_Path attribute. However, to ensure backwards compatibility during a period of transition from a 'current' algorithm suite to a 'new' algorithm suite, it will be necessary 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

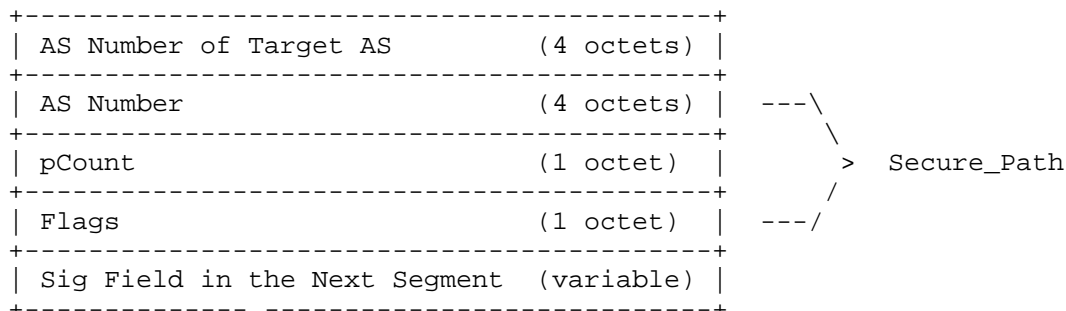
BGPSEC\_Path attribute. If the origin AS in the Secure\_Path is not in the set of AS numbers associated with the given prefix, then the BGPSEC update message is 'Not Valid' and the validation algorithm terminates.

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

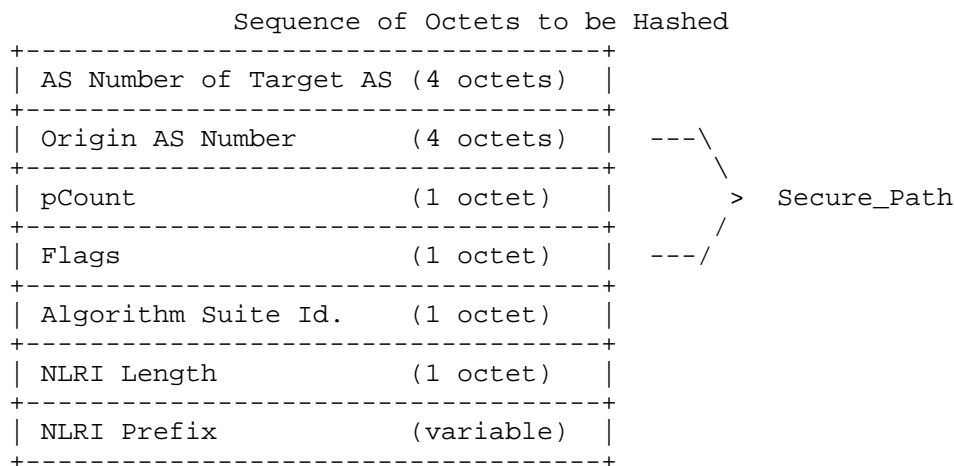


For the first segment to be processed (the most recently added segment), the 'AS Number of Target AS' is the AS number of the BGPSEC speaker validating the update message. Note that if a BGPSEC speaker uses multiple AS Numbers (e.g., the BGPSEC speaker is a member of a confederation), the AS number used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

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

R. Gagliano  
K. Patel  
B. Weis  
Cisco Systems  
October 22, 2012

BGPSEC router key rollover as an alternative to beaconing  
draft-ietf-sidr-bgpsec-rollover-01

## Abstract

BGPSEC will need to address the impact from regular and emergency rollover processes for the BGPSEC End-Entity (EE) certificates that will be performed by Certificate Authorities (CAs) participating at the Resource Public Key Infrastructure (RPKI). This document provides general recommendations for that process and specifies how this process is used to control BGPSEC's window of exposure to replay attacks.

## 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. Requirements notation . . . . .	3
2. Introduction . . . . .	4
3. Key rollover in BGPSEC . . . . .	5
3.1. A proposed process for BGPSEC key rollover . . . . .	5
4. BGPSEC key rollover as a measure against replays attacks in BGPSEC . . . . .	8
4.1. BGPSEC Replay attack window requirement . . . . .	8
4.2. BGPSEC key rollover as a mechanism to protect against replay attacks . . . . .	8
5. IANA Considerations . . . . .	10
6. Security Considerations . . . . .	11
7. Acknowledgements . . . . .	12
8. References . . . . .	13
8.1. Normative References . . . . .	13
8.2. Informative References . . . . .	13
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

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

When a router receives (or creates depending of the key provisioning mechanism to be selected) a new key pair, this key pair will be used to sign new BGP UPDATE messages that are originated or that transit through the BGP speaker. Additionally, the BGP speaker MUST refresh its outbound BGP UPDATE messages to update its respective BGPSEC attribute by including the correspondent signature performed with the new key. When the rollover process finishes, the old BGPSEC certificate (and its key) will not longer be valid and thus any BGP UPDATE that includes a BGPSEC attribute with a signature performed by the old key will be invalid. Consequently, if the router do not refresh its outbound BGP UPDATE messages, routing information may be lost after the rollover process is finished.

As a key rollover process invalidates BGP UPDATE messages signed with the old key, frequent key rollover processes could be used to control BGPSEC's window of exposure to replay attacks as required by [I-D.ietf-sidr-bgpsec-reqs]. This document explores the operational environment to achieve this goal.

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



### 3. Key rollover in BGPSEC

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

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

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

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

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

#### 3.1. A proposed process for BGPSEC key rollover

The BGPSEC key rollover process should be dependent of the key provisioning mechanisms that would be in place. The key provisioning mechanisms for BGPSEC are not yet fully documented (see [I-D.ietf-sidr-rtr-keying] as a work in progress document). We will assume that an automatic provisioning mechanism will be in place. (A possible provisioning mechanism is the Enrollment over Secure Transport (EST) [I-D.ietf-pkix-est]). That protocol will allow BGPSEC code to include automatic re-keying scripts with minimum development cost.

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

1. New Certificate Pre-Publication: The first step in the rollover mechanism is to pre-publish the new public key in a new certificate. In order to accomplish this goal, the new key pair and certificate will need to be generated and published at the appropriate RPKI repository publication point. The details of this process will vary as they depend on whether the keys are assigned per-BGP speaker or shared, whether the keys are

generated on each BGP speaker or in a central location and whether the RPKI repository is locally or externally hosted.

2. **Staging Period:** A staging period will be required from the time a new certificate is published in the RPKI global repository until the time it is fetched by RPKI caches around the globe. The exact minimum staging time is not clear and will require experimental results from RPKI operations. RPKI repository design documents mention a lower limit of 24 hours (NOTE: need reference only one I found is the ops document). If rollovers will be done frequently and we want to avoid the stage period, an administrator can always provision two certificates for every router. In this case when the rollover operation is needed, the relying parties around the globe would already have the new keys. A staging period may not be possible to implement during emergency key rollover, in which case routing information may be lost.
3. **Twilight:** At this moment, the BGP speaker that holds the private key that has been rolled-over will stop using the OLD key for signing and start using the NEW key. Also, the router will generate appropriate BGP UPDATES just as in the typical operation of refreshing out-bound BGP policies. This operation may generate a great number of BGP UPDATE messages (due to the need to refresh BGP outbound policies). In any given BGP SPEAKER, the Twilight moment may be different for every peer in order to distribute the system load (probably in the order of minutes to avoid reaching any expiration time).
4. **Certificate Revocation:** This is an optional step. As part of the rollover process, a CA MAY decide to revoke the OLD certificate by publishing its serial number on the CA's CRL. On the other side, the CA will just let the OLD certificate to expire and not revoke it. This choice will depend on the reasons that motivated the rollover process.
5. **RPKI-Router Protocol Withdrawals:** Either due to the revocation of the OLD certificate or to the expiration of the OLD certificate's validation, the RPKI relying parties around the globe will need to communicate to their RTR peers that the OLD certificate's public key is no longer valid (rtr withdrawal message). It is not documented yet what will be a router's reaction to a RTR withdrawal message but it should include the removal of any RIB entry that includes a BGPSEC attribute signed with that key and the generation of the correspondent BGP WITHDRAWALS (either implicit or explicit).

The proposed rollover mechanism will depend on the existence of an

automatic provisioning process for BGPSEC certificates. It will require a staging mechanism based on the RPKI propagation time of around 24hours, and it will generate BGP UPDATES for all prefixes in the router been re-keyed.

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

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

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

There are two typical generic measures to mitigate replay attacks in any protocol: the addition of a timestamp or the addition of a serial number. Currently BGPSEC offers a timestamp (expiration time) as a protection against re-play attacks of BGPSEC attributes. The process requires all BGP Speakers that originate a BGP UPDATE to re-advertise ("beacon") the message before it expires. This requirement changes a long standing BGP operational practice and the community has been searching for alternatives.

##### 4.1. BGPSEC Replay attack window requirement

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

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

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

The question we would like to ask is: can the key rollover process earlier described provide a similar protection against replay attacks without the need for beaconing?

The answer is that YES when the window requirement is in the order of days and the BGP speaker re-keying is the edge router of the origin AS and the full process is completed (i.e. the OLD and NEW certificate do not share the same key). By using re-keying, you are letting the BGPSEC certificate validation time as your timestamp against replay attacks. However, the use of frequent key rollovers comes with an additional administrative cost and risks if the process fails. As documented before, re-keying should be supported by automatic tools and for the great majority of the Internet it will be done with good lead time to correct any risk.

For a transit AS that also originates BGP UPDATES for its own prefixes, the key rollover process may generate a large number of

UPDATE messages (even the complete Default Free Zone or DFZ). For this reason, it is recommended that routers in this scenario be provisioned with two certificates: one to sign BGP UPDATES in transit and a second one to sign BGP UPDATE for prefixes originated in its AS. Only the second certificate (for prefixes originated in its AS) should be rolled-over frequently as a means of limiting replay attack windows. The transit BGPSEC certificate is expected to be longer living than the origin BGPSEC certificate.

Advantage of Re-keying as replay attack protection mechanism:

1. Does not require beaconing
2. All expiration policies are maintained in RPKI
3. Most of the additional administrative cost is paid by the provider that wants to protect its infrastructure (RP load will increase as there is a need to validate more BGPSEC certificates)
4. Can be implemented in coordination with planned topology changes by either origin ASes or transit ASes (if I am changing providers, I rollover)
5. Eliminates the discussion on who has the authority over the expiration time

Disadvantage of Re-keying as replay attack protection mechanism:

1. More administrative load due to frequent rollover, although how frequent is still not clear. Some initial ideas in [I-D.ietf-sidr-bgpsec-ops]
2. Minimum window size bounded by RPKI propagation time to RPKI caches for new certificate and CRL (2x propagation time). If pre-provisioning done ahead of time the minimum windows size is reduced (to 1x propagation time for the CRL). However, more experimentation is needed when RPKI and RPs are more massively deployed.
3. Increases dynamics and size of RPKI repository.
4. More load on RPKI caches, but they are meant to do this work.

## 5. IANA Considerations

No IANA considerations

## 6. Security Considerations

No security considerations.

## 7. Acknowledgements

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



## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6489] Huston, G., Michaelson, G., and S. Kent, "Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI)", BCP 174, RFC 6489, February 2012.

### 8.2. Informative References

- [I-D.ietf-pkix-cmc-serverkeygeneration]  
Schaad, J., Timmel, P., and S. Turner, "CMC Extensions: Server Key Generation",  
draft-ietf-pkix-cmc-serverkeygeneration-00 (work in progress), January 2012.
- [I-D.ietf-pkix-est]  
Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", draft-ietf-pkix-est-02 (work in progress), July 2012.
- [I-D.ietf-sidr-bgpsec-ops]  
Bush, R., "BGPsec Operational Considerations",  
draft-ietf-sidr-bgpsec-ops-05 (work in progress),  
May 2012.
- [I-D.ietf-sidr-bgpsec-reqs]  
Bellovin, S., Bush, R., and D. Ward, "Security Requirements for BGP Path Validation",  
draft-ietf-sidr-bgpsec-reqs-03 (work in progress),  
March 2012.
- [I-D.ietf-sidr-rtr-keying]  
Turner, S., Patel, K., and R. Bush, "Router Keying for BGPsec", draft-ietf-sidr-rtr-keying-00 (work in progress),  
May 2012.

Authors' Addresses

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

Email: rogaglia@cisco.com

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

Email: keyupate@cisco.com

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

Email: bew@cisco.com



Secure Inter-Domain Routing  
Internet-Draft  
Intended status: Standards Track  
Expires: April 14, 2013

M. Reynolds  
IPSw  
S. Kent  
BBN  
M. Lepinski  
BBN  
Oct 11, 2012

Local Trust Anchor Management for the Resource Public Key Infrastructure  
<draft-ietf-sidr-ltamgmt-07.txt>

#### Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 14, 2013.

#### Copyright and License 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.

## Abstract

This document describes a facility to enable a relying party (RP) to manage trust anchors (TAs) in the context of the Resource Public Key Infrastructure (RPKI). It is common in RP software (not just in the RPKI) to allow an RP to import TA material in the form of self-signed certificates. However, this approach to incorporating TAs is potentially dangerous. (These self-signed certificates rarely incorporate any extensions that impose constraints on the scope of the imported public keys, and the RP is not able to impose such constraints.) The facility described in this document allows an RP to impose constraints on such TAs. Because this mechanism is designed to operate in the RPKI context, the most important constraints are the Internet Number Resources (INRs) expressed via RFC 3779 extensions. These extensions bind address spaces and/or autonomous system (AS) numbers to entities. The primary motivation for the facility described in this document is to enable an RP to ensure that INR information that it has acquired via some trusted channel is not overridden by the information acquired from the RPKI repository system or by the putative TAs that the RP imports. Specifically, the mechanism allows an RP to specify a set of overriding bindings between public key identifiers and INR data. These bindings take precedence over any conflicting bindings acquired by the putative TAs and the certificates downloaded from the RPKI repository system. This mechanism is designed for local use by an RP, but any entity that is accorded administrative control over a set of RPs may use this mechanism to convey its view of the RPKI to RPs within its jurisdiction. The means by which this latter use case is effected is outside the scope of this document.

## Table of Contents

1	Introduction . . . . .	4
1.1	Terminology . . . . .	5
2	Overview of Certificate Processing . . . . .	5
2.1	Target Certificate Processing . . . . .	5
2.2	Perforation . . . . .	5
2.3	TA Re-parenting . . . . .	6
2.4	Paracertificates . . . . .	6
3	Format of the constraints file . . . . .	8
3.1	Relying party subsection . . . . .	8
3.2	Flags subsection . . . . .	8
3.3	Tags subsection . . . . .	9
3.3.1	Xvalidity_dates tag . . . . .	10
3.3.2	Xcrl_dp tag . . . . .	10
3.3.3	Xcp tag . . . . .	11
3.3.4	Xaia tag . . . . .	11
3.4	Blocks subsection . . . . .	12
4	Certificate Processing Algorithm . . . . .	13
4.1	Proofreading algorithm . . . . .	14
4.2	TA processing algorithm . . . . .	15
4.2.1	Preparatory processing (stage 0) . . . . .	16
4.2.2	Target processing (stage 1) . . . . .	17
4.2.3	Ancestor processing (stage 2) . . . . .	18
4.2.4	Tree processing (stage 3) . . . . .	19
4.2.5	TA re-parenting (stage 4) . . . . .	20
4.3	Discussion . . . . .	21
5	Implications for Path Discovery . . . . .	21
5.1	Two answers . . . . .	21
5.2	One answer . . . . .	22
5.3	No answer . . . . .	22
6	Implications for Revocation . . . . .	22
6.1	No state bits set . . . . .	23
6.2	ORIGINAL state bit set . . . . .	23
6.3	PARA state bit set . . . . .	23
6.4	Both ORIGINAL and PARA state bits set . . . . .	24
7	Security Considerations . . . . .	24
8	IANA Considerations . . . . .	24
9	Acknowledgements . . . . .	24
10	References . . . . .	24
10.1	Normative References . . . . .	24
10.2	Informative References . . . . .	25
	Authors' Addresses . . . . .	25
	Appendix A: Sample Constraints File . . . . .	26
	Appendix B: Optional Sorting Algorithm for Ancestor Processing . . . . .	27

## 1 Introduction

The Resource Public Key Infrastructure (RPKI) [RFC6480] is a PKI in which certificates are issued to facilitate management of Internet Resource Numbers (INRs). Such resources are expressed in the form of X.509v3 "resource" certificates with extensions defined by RFC 3779 [RFC6487]. Validation of a resource certificate is preceded by path discovery. In a PKI path discovery is effected by constructing a certificate path between a target certificate and a trust anchor (TA). No IETF standards define how to construct a certificate path; commonly such paths are based on a bottom-up search using Subject/Issuer name matching, but top-down and meet-in-the-middle approaches may also be employed [RFC4158]. In contrast, path validation is top-down, as defined by [RFC5280].

In the RPKI, certificates can be acquired in various ways, but the default is a top-down tree walk as described in [RFC6481], initialized via a Trust Anchor Locator [RFC6490]. Note that the process described there is not path discovery per se but the collecting of certificates to populate a local cache. Thus, the common, bottom-up path discovery approach is not inconsistent with these RFCs. Moreover, a bottom-up path discovery approach is more general, accommodating certificates that might be acquired by other means, i.e., not from an RPKI repository. There are circumstances under which an RP may wish to override the INR specifications obtained through the RPKI distributed repository system [RFC6481]. This document describes a mechanism by which an RP may override any conflicting information expressed via putative TAs and the certificates downloaded from the RPKI repository system. Thus the algorithms described in this document adopt a bottom-up path discovery approach.

To effect this local control, this document calls for a relying party to specify a set of bindings between public key identifiers and INRs through a text file known as a constraints file. The constraints expressed in this file then take precedence over any competing claims expressed by resource certificates acquired from the distributed repository system. (The means by which a relying party acquires the key identifier and the RFC 3779 extension data used to populate the constraints file is outside the scope of this document.) The relying party also may use a local publication point (the root of a local directory tree that is made available as if it were a remote repository) as a source of certificates and CRLs (and other RPKI signed objects, e.g., ROAs and manifests) that do not appear in the RPKI repository system.

In order to allow reuse of existing, standard path validation mechanisms, the RP-imposed constraints are realized by having the RP itself represented as the only TA known in the local certificate validation context. To ensure that all RPKI certificates can be validated relative to this TA, this RP TA certificate must contain all-encompassing resource allocations, i.e. 0/0 for IPv4, 0::/0 for IPv6 and 0-4294967295 for AS numbers. Thus, a conforming implementation of this mechanism must be able to cause a self-signed certification authority (CA) certificate to be created with a locally generated key pair. It also must be able to issue CA certificates subordinate to this TA. Finally, a conforming implementation of this





mechanism must process the constraints file and modify certificates as needed in order to enforce the constraints asserted in the file.

The remainder of this document describes in detail the types of certificate modification that may occur, the syntax and semantics of the constraints file, and the implications of certificate modification on path discovery and revocation.

## 1.1 Terminology

It is assumed that the reader is familiar with the terms and concepts described in "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" [RFC5280] and "X.509 Extensions for IP Addresses and AS Identifiers" [RFC3779].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

## 2 Overview of Certificate Processing

The fundamental aspect of the facility described in this document is one of certificate modification. The constraints file, described in more detail in the next section, contains assertions about INRs that are to be specially processed. As a result of this processing, certificates in the local copy of the RPKI repository are transformed into new certificates satisfying the INR constraints so specified. This enables the RP to override conflicting assertions about resource holdings as acquired from the RPKI repository system. Three forms of certificate modification can occur. (Every certificate is digitally signed and thus cannot be modified without "breaking" its signature. In the context of this document we assume that certificates that are modified have been validated previously. Thus the content can be modified, locally, without the need to preserve the integrity of the signature. These modified certificates are referred to as paracertificates (see section 2.4 below).)

### 2.1 Target Certificate Processing

If a certificate is acquired from the RPKI repository system and its Subject key identifier (SKI) is listed in the constraints file, it will be reissued directly under the RP TA certificate, with (possibly) modified RFC 3779 extensions. (The SKI is used as a compact reference to the public key in a target certificate.) The modified extensions will include any RFC 3779 data expressed in the constraints file. Other certificate fields may also be modified to maintain consistency. (These fields are enumerated in Table 1, and discussed in Section 3.3.) In Section 4.2, target certificate processing corresponds to stage one of the algorithm. (When a target certificate is re-parented, all subordinate signed products will still be valid, unless the set of INRs in the targeted certificate is reduced.)

### 2.2 Perforation

When a target certificate is re-issued directly under the RP's TA, its INRs MUST be removed from all of its parent (CA) certificates. (If these INRs were not removed, then conflicting assertions about INRs could arise and undermine the authority of the RP TA.) Thus, every

certificate acquired from the RPKI repository MUST be examined to determine if it contains an RFC 3779 extension that intersects the resource data in the constraints file. If there is an intersection the certificate will be reissued directly under the RP TA, with modified RFC 3779 extensions. We refer to the process of modifying the RFC 3779 extension in an affected certificate as "perforation" (because the process will create "holes" in these extensions). The

modified extensions will exclude any RFC 3779 data expressed in the constraints file. In the certificate processing algorithm described in Section 4.2, perforation corresponds to stage two of the algorithm ("ancestor processing") and also to stage three of the algorithm ("tree processing").

### 2.3 TA Re-parenting

All valid, self-signed certificates offered as TAs in the public RPKI certificate hierarchy, e.g., self-signed certificates issued by IANA or RIRs, will be re-issued under the RP TA certificate. This processing is done even though all but one of these certificates might not intersect any resources specified in the constraints file. We refer to this reissuance as "re-parenting" since the issuer (parent) of the certificate has been changed. The issuer name is changed from that of the certificate subject (this is a self-signed certificate) to that of the RP TA. In the certificate processing algorithm described in Section 4.2, TA re-parenting corresponds to stage four of the algorithm. (In a more generic PKI context, re-parenting enables an RP to insert extensions in these certificates to impose constraints on path processing in a fashion consistent with RFC 5280. In this fashion an RP can impose name constraints, policy constraints, etc.)

### 2.4 Paracertificates

If a certificate is subject to any of the three forms of processing just described, that certificate will be referred to as an "original" certificate and the processed (output) certificate will be referred to as a paracertificate. When an original certificate is transformed into a paracertificate all the fields and extensions from the original certificate will be retained, except as indicated in Table 1, below.

Original Certificate Field	Action
Version	unchanged
Serial number	created per note A
Signature	replaced if needed with RP's signing alg
Issuer	replaced with RP's name
Validity dates	replaced per note B
Subject	unchanged
Subject public key info	unchanged
Extensions	
Subject key identifier	unchanged
Key usage	unchanged
Basic constraints	unchanged
CRL distribution points	replaced per note B
Certificate policy	replaced per note B
Authority info access	replaced per note B
Authority key ident	replaced with RP's
IP address block	modified as described
AS number block	modified as described
Subject info access	unchanged
All other extensions	unchanged
Signature Algorithm	same as above
Signature value	new

Table 1 Certificate Field Modifications

Note A. The serial number will be created by concatenating the current time (the number of seconds since Jan 1, 1970) with a count of the certificates created in the current run. Because all paracertificates are issued directly below the RP TA, this algorithm ensures serial number uniqueness.

Note B. These fields are derived (as described in Section 3.3 below) from parameters in the constraints file (if present); otherwise, they take on values from the certificates from which the paracertificates are derived.

### 3 Format of the constraints file

This section describes the syntax of the constraints file. (The syntax has been defined to enable creation and distribution of constraint files to a set of RPs, by an authorized third party.) The model described below is nominal; implementations need not match all details of this model as presented, but the external behavior of implementations **MUST** correspond to the externally observable characteristics of this model in order to be compliant. It is **RECOMMENDED** that the syntax described herein be supported, to facilitate interoperability between creators and consumers of constraints files.

The constraints file consists of four logical subsections: the relying party subsection, the flags subsection, the tags subsection and the blocks subsection. The relying party subsection and the blocks subsection are **REQUIRED** and **MUST** be present; the flags and tags subsections are **OPTIONAL**. Each subsection is described in more detail below. Note that the semicolon (;) character acts as the comment character, to enable annotating constraints files. All characters from a semicolon to the end of that line are ignored. In addition, lines consisting only of whitespace are ignored. The subsections **MUST** occur in the order indicated. An example constraints file is given in Appendix A.

#### 3.1 Relying party subsection

The relying party subsection is a **REQUIRED** subsection of the constraints file. It **MUST** be the first subsection of the constraints file, and it **MUST** consist of two lines of the form:  
(**RECOMMENDED**)

```
PRIVATEKEYMETHOD    value [ ... value ]
TACERTIFICATE        value
```

The first line provides a pointer (including an access method) to the RP's private key. This line consists of the string literal **PRIVATEKEYMETHOD**, followed by one or more whitespace delimited string values. These values are passed to the certificate processing algorithm as described below. Note that this entry, as for all entries in the constraints file, is case sensitive.

The second line of this subsection consists of the string literal **TACERTIFICATE**, followed by exactly one string value. This value is the name of a file containing the relying party's TA certificate. The file name is passed to the certificate processing algorithm as described below.

#### 3.2 Flags subsection

The flags subsection of the constraints file is an **OPTIONAL** subsection. If present it **MUST** immediately follow the relying party

subsection. The flags subsection consists of one or more lines of the form

```
CONTROL  flagname  booleanvalue
```

Each such line is referred to as a control line. Each control line MUST contain exactly three whitespace delimited strings. The first string MUST be the literal CONTROL. The second string MUST be one of the following three literals:

```
resource_nounion
intersection_always
treegrowth
```

The third string denotes a Boolean value, and MUST be one of the literals TRUE or FALSE. Control flags influence the global operation of the certificate processing algorithm; the semantics of the flags is described in Section 4.2. Note that each flag has a default value, so that if the corresponding CONTROL line does not appear in the constraints file, the algorithm flag is considered to take the corresponding default value. The default value for each flag is FALSE. Thus, if any flag is not named in a control line it takes the value FALSE. If the flags subsection is absent, all three flags assume the default value FALSE.

### 3.3 Tags subsection

The tags subsection is an OPTIONAL subsection in the constraints file. If present it MUST immediately follow the relying party subsection (if the flags subsection is absent) or the flags subsection (if it is present). The tags subsection consists of one or more lines of the form

```
TAG  tagname  tagvalue [ ... tagvalue ]
```

Each such line is referred to as a tag line. Each tag line MUST consist of at least three whitespace delimited string values, the first of which must be the literal TAG. The second string value gives the name of the tag, and subsequent string(s) give the value(s) of the tag. The tag name MUST be one of the following four string literals:

```
Xvalidity_dates
Xcrl_dp
Xcp
Xaia
```

The purpose of the tag lines is to provide an indication of the means

by which paracertificate fields, specifically those indicated above under "Note B", of Table 1 are constructed. Each tag has a default, so that if the corresponding tag line is not present in the constraints file, the default behavior is used when constructing the paracertificates. The syntax and semantics of each tag line is described next.

Note that the tag lines are considered to be global; the action of each tag line (or the default action, if that tag line is not present) applies to all paracertificates that are created as part of the certificate processing algorithm.

### 3.3.1 Xvalidity\_dates tag

This tag line is used to control the value of the notBefore and notAfter fields in paracertificates. If this tag line is specified and there is a single tagvalue which is the literal string C, the paracertificate validity interval is copied from the original certificate validity interval from which it is derived. If this tag is specified and there is a single tagvalue which is the literal string R, the paracertificate validity interval is copied from the validity interval of the RP's TA certificate. If this tag is specified and the tagvalue is neither of these literals, then exactly two tagvalues MUST be specified. Each must be a Generalized Time string of the form YYYYMMDDHHMMSSZ. The first tagvalue is assigned to the notBefore field and the second tagvalue is assigned to the notAfter field. It MUST be the case that the tagvalues can be parsed as valid Generalized Time strings such that notBefore is less than notAfter, and also such that notAfter represents a time in the future (i.e., the paracertificate has not already expired).

If this tag line is not present in the constraints file the default behavior is to copy the validity interval from the original certificate to the corresponding paracertificate.

### 3.3.2 Xcrl dp tag

This tag line is used to control the value of the CRL distribution point extension in paracertificates. If this tag line is specified and there is a single tagvalue that is the string literal C, the CRLDP of the paracertificate is copied from the CRLDP of the original certificate from which it is derived. If this tag line is specified and there is a single tagvalue that is the string literal R, the CRLDP of the paracertificate is copied from the CRLDP of the RP's TA certificate. If this tag line is specified and there is a single tagvalue that is not one of these two reserved literals, or if there is more than one tagvalue, then each tagvalue is interpreted as a URI that will be placed in the CRLDP sequence in the

paracertificate.

If this tag line is not present in the constraints file the default behavior is to copy the CRLDP from the original certificate into the corresponding paracertificate.

### 3.3.3 Xcp tag

This tag line is used to control the value of the policyQualifierId field in paracertificates. If this tag line is specified there MUST be exactly one tagvalue. If the tagvalue is the string literal C, the paracertificate value is copied from the value in the corresponding original certificate. If the tagvalue is the string literal R, the paracertificate value is copied from the value in the RP's top level TA certificate. If the tagvalue is the string literal D, the paracertificate value is set to the default OID. If the tagvalue is not one of these reserved string literals, then the tagvalue MUST be an OID specified using the standard dotted notation. The value in the paracertificate's policyQualifierId field is set to this OID. Note the RFC 5280 specifies that only a single policy may be specified in a certificate, so only a single tagvalue is permitted in this tag line, even though the CertificatePolicy field is an ASN.1 sequence.

If this tag line is not specified the default behavior is to use the default OID in creating the paracertificate.

This option permits the RP to convert a value of the policyQualifierId field in a certificate (that would not be in conformance with the RPKI CP) to a conforming value in the paracertificate. This conversion enables use of RPKI validation software that checks the policy field against that specified in the RPKI CP [RFC6484].

### 3.3.4 Xaia tag

This tag line is used to control the value of the Authority Information Access (AIA) extension in the paracertificate. If this tag line is present then it MUST have exactly one tagvalue. If this tagvalue is the string literal C, then the AIA field in the paracertificate is copied from the AIA field in the original certificate from which it is derived. If this tag line is present and the tagvalue is not the reserved string literal, then the tagvalue MUST be a URI. This URI is set as the AIA extension of the paracertificates that are created.

If this tag line is not specified the default behavior is to use copy the AIA field from the original certificate to the AIA field of the paracertificate.



### 3.4 Blocks subsection

The blocks subsection is a REQUIRED subsection of the constraints file. If the tags subsection is present, the blocks subsection MUST appear immediately after it. This MUST be the last subsection in the constraints file. The blocks subsection consists of one or more blocks, known as target blocks. A target block is used to specify an association between a certificate (identified by an SKI) and a set of resource assertions. Each target block contains four regions, an SKI region, an IPv4 region, an IPv6 region and an AS number region. All regions MUST be present in a target block.

The SKI region contains a single line beginning with the string literal SKI and followed by forty hexadecimal characters giving the subject key identifier of a certificate, known as the target certificate. The hex character string MAY contain embedded whitespace or colon characters (included to improve readability), which are ignored. The IPv4 region consists of a line containing only the string literal IPv4. This line is followed by zero or more lines containing IPv4 prefixes in the format described in RFC 3779. The IPv6 region consists of a line containing only the string literal IPv6, followed by zero or more lines containing IPv6 prefixes using the format described in RFC 3513. (The presence of the IPv4 and IPv6 literals is to simplify parsing of the constraints file.) Finally, the AS number region consists of a line containing only the string literal AS#, followed by zero or more lines containing AS numbers (one per line). The AS numbers are specified in decimal notation as recommended in RFC 5396. A target block is terminated by either the end of the constraints file, or by the beginning of the next target block, as signaled by its opening SKI region line. An example target block is shown below. (The indentation used below is employed to improve readability and is not required.) See also the complete constraints file example in Appendix A. Note that whitespace, as always, is ignored.

```
SKI 00:12:33:44:00:BA:BA:DE:EB:EE:00:99:88:77:66:55:44:33:22:11
IPv4
  10.2.3/24
  10.8/16
IPv6
  1:2:3:4:5:6/112
AS#
  123
  567
```

The blocks subsection MUST contain at least one target block. Note that it is OPTIONAL that the SKI refer to a certificate that is known

or resolvable within the context of the local RPKI repository. Also, there is no REQUIRED or implied ordering of target blocks within the block subsection. Since blocks may occur in any order, the outcome of processing a constraints file may depend on the order in which target blocks occur within the constraints file. The next section of this document contains a detailed description of the certificate processing algorithm.

#### 4 Certificate Processing Algorithm

The section describes the certificate processing algorithm by which paracertificates are created from original certificates in the local RPKI repository. For the purposes of describing this algorithm, it will be assumed that certificates are persistently associated with state (or metadata) information. This state information is nominally represented by an array of named bits associated with each certificate. No specific implementation of this functionality is mandated by this document. Any implementation that provides the indicated functionality is acceptable, and need not actually consist of a bit field associated with each certificate.

The following state bits used in certificate processing are

- NOCHAIN
- ORIGINAL
- PARA
- TARGET

If the NOCHAIN bit is set, this indicates that a full path between the given certificate and a TA has not yet been discovered. If the ORIGINAL bit is set, this indicates that the certificate in question has been processed by some part of the processing algorithm described in Section 4.2. If it was processed as part of stage one processing, as described in section 4.2.2, the TARGET bit also will be set. Finally, every paracertificate will have the PARA bit set.

At the beginning of algorithm processing each certificate in the local RPKI repository has the ORIGINAL, PARA and TARGET bits clear. If a certificate has a complete, validated path to a TA, or is itself a TA, then that certificate will have the NOCHAIN bit clear, otherwise it will have the NOCHAIN bit set. As the certificate processing algorithm proceeds, the metadata state of original certificates may change. In addition, since the certificate processing algorithm may also be creating paracertificates, it is responsible for actively setting or clearing the state of these four bits on those paracertificates.

The certificate processing algorithm consists of two sub-algorithms:

"proofreading" and "TA processing". Conceptually, the proofreading algorithm performs syntactic checks on the constraints file, while the TA processing algorithm performs the actual certificate transformation processing. If the proofreading algorithm does not succeed in parsing the constraints file, the TA processing-algorithm is not executed. Note also that if the constraints file is not present, neither algorithm is executed and the local RPKI repository is not modified. Each of the constituent algorithms will now be described in detail.

#### 4.1 Proofreading algorithm

The proofreading algorithm checks the constraints file for syntactic errors, e.g., missing REQUIRED subsections, or malformed addresses. Implementation of this algorithm is OPTIONAL. If it is implemented, the following text defines correct operation for the algorithm. The proofreading algorithms performs a set of heuristic checks, such as checking for prefixes that are too large (e.g., larger than /8). The proofreading algorithm also SHOULD examine resource regions (IPv4, IPv6 and AS# regions) within the blocks subsection, and reorder such resources within a region in ascending numeric order. On encountering any error the proofreading algorithm SHOULD provide an error message indicating the line on which the error occurred as well as informative text that is sufficiently descriptive as to allow the user to identify and correct the error. An implementation of the proofreading algorithm MUST NOT assume that it has access to the local RPKI repository (even read-only access). An implementation of the proofreading algorithm MUST NOT alter the local RPKI repository in any way; it also MUST NOT change any of the metadata associated with certificates in that repository. (Recall that the processing described here is creating a copy of that local repository.) For simplicity the remainder of this document assumes that the proofreading algorithm produces a transformed output file. This file contains the same syntactic information as the text version of the constraints file.

The proofreading algorithm performs the following syntactic checks on the constraints file:

- verifies the presence of the REQUIRED relying party subsection and the REQUIRED blocks subsection.
- verifies the order of the two, three or four subsections as stated above.
- verifies that the relying party subsection conforms to the specification given in Section 3.1 above.
- verifies that, if present, the tags and flags subsections conform to the specifications in Sections 3.2 and 3.3 above.

After these checks have been performed, the proofreading algorithm then checks the blocks subsection:

- splits the blocks subsection into constituent target blocks, as delimited by the SKI region line(s)
- verifies that at least one target block is present
- verifies that each SKI region line contains exactly forty hexadecimal digits and contains no additional characters other than whitespace or colon characters.

For each target the proofreading algorithm:



- verifies the presence of the IPv4, IPv6 and AS# regions, and verifies that at least one such resource is present.
- verifies that, for each IPv4 prefix, IPv6 prefix and autonomous system number given, that the indicated resource is syntactically valid according to the appropriate RFC definition, as described in Section 3.4.
- verifies that no IPv4 resource has a prefix larger than /8.
- optionally performing reordering within each of the three resource regions so that stated resources occur in ascending numerical order.

(If the proofreading algorithm has performed any reordering of information it MAY overwrite the constraints file. If it does so, however, it MUST preserve all information contained within the file, including information that is not parsed (such as comments). If the proofreading algorithm has performed any reordering of information but has not overwritten the constraints file, it MAY produce a transformed output file, as described above. If the proofreading algorithm has performed any reordering of information, but has neither overwritten the constraints file nor produced a transformed output file, it MUST provide an error message to the user indicating what reordering was performed.)

#### 4.2 TA processing algorithm

The TA processing algorithm acts on the constraints file (as processed by the proofreading algorithm) and the contents of the local RPKI repository to produce paracertificates for the purpose of enforcing the resource allocations as expressed in the constraints file. The TA processing algorithm operates in five stages, a preparatory stage (stage 0), target processing (stage 1), ancestor processing (stage 2), tree processing (stage 3) and TA re-parenting (stage 4). Conceptually, during the preparatory stage the proofreader output file is read and a set of internal RP, tag and flag variables are set based on the contents of that file. (If the constraint file has not specified one or more of the tags and/or flags, those tags and flags are set to default values.) During target processing all certificates specified by a target block are processed, and the resources for those certificates are (potentially) expanded; for each target found a new paracertificate is manufactured with its various fields set, as shown in Table 1, using the values of the internal variables set in the preparatory stage and also, of course, the fields of the original certificate (and, potentially, fields of the RP's TA certificate). In stage 2 (ancestor) processing, all ancestors of the each target certificate are found, and the claimed resources are then removed (perforated). A new paracertificate with these diminished resources is crafted, with its fields generated based on internal variable settings, original certificate field values, and, potentially, the fields of the RP's TA certificate. In tree processing (stage 3), the

entire local RPKI repository is searching for any other certificates that have resources that intersect a target resource, and that were not otherwise processed during a preceding stage. Perforation is again performed for any such intersecting certificates, and paracertificates created as in stage 2. In the fourth (last) stage, TA re-parenting, any TA certificates in the local RPKI repository that have not already been processed are now re-parented under the RP's TA certificate. This transformation creates paracertificates; however, these paracertificates may have RFC 3779 resources that were not altered during algorithm processing. The final output of algorithm processing will be threefold:

- the metadata information on some (original) certificates in the repository MAY be altered.
- paracertificates will be created, with the appropriate metadata, and entered into the repository.
- the TA processing algorithm SHOULD produce a human readable log of its actions, indicating which paracertificates were created and why. The remainder of this section describes the processing stages of the algorithm in detail.

#### 4.2.1 Preparatory processing (stage 0)

During preparatory processing, the output of the proofreader algorithm, is read. Internal variables are set corresponding to each tag and flag, if present, or to their defaults, if absent. Internal variables are set corresponding to the PRIVATEKEYMETHOD value string(s) and the TACERTIFICATE string. The TA processing algorithm is queried to determine if it supports the indicated private key access methodology. This query is performed in an implementation-specific manner. In particular, an implementation is free to vacuously return success to this query. The TA processing algorithm next uses the value string for the TACERTIFICATE to locate this certificate, again in an implementation-specific manner. The certificate in question may already be present in the local RPKI repository, or it may be located elsewhere. The implementation is free to create the top level certificate at this time, and then assign to this newly-created certificate the name indicated. It is necessary only that, at the conclusion of this processing, a valid trust anchor certificate for the relying party has been created or otherwise obtained.

Some form of access to the RP's private key and top level certificate are required for subsequent correct operation of the algorithm. Therefore, stage 0 processing MUST terminate if one or both conditions are not satisfied. In the error case, the implementation SHOULD provide an error message of sufficient detail that the user can correct the error(s). If stage 0 processing does not succeed, no further stages of TA processing are executed.

#### 4.2.2 Target processing (stage 1)

During target processing, the TA processing algorithm reads all target blocks in the proofreader output file. It then processes each target block in the order specified in the file. In the description that follows, except where noted, the operation of the algorithm on a single target block will be described. Note, however, that all stage 1 processing is executed before any processing in subsequent stages is performed.

The algorithm first obtains the SKI region of the target block. It then locates (in an implementation-dependent manner) the certificate identified by the SKI. Note that this search is performed only against (original) certificates, not against paracertificates. If more than one original certificate is found matching this SKI, there are two possible scenarios. If a resource holder has two certificates issued by the same CA, with overlapping validity intervals and the same key, but distinct subject names (typically, by virtue of the SerialNumber parts being different), then these two certificates are both considered to be (distinct) targets, and are both processed. If, however, a resource holder has certificates issued by two different CAs, containing different resources, but using the same key, there is no unambiguous method to decide which of the certificates is intended as the target. In this latter case the algorithm MUST issue a warning to that effect, mark the target block in question as unavailable for processing by subsequent stages and proceed to the next target block. If no certificate is found then the algorithm SHOULD issue a warning to that effect and proceed to process the next target block.

If a single (original) certificate is found matching the indicated SKI, then the algorithm takes the following actions. First, it sets the ORIGINAL state bit for the certificate found. Second, it sets the TARGET state bit for the certificate found. Third, it extracts the INRs from the certificate. If the global resource\_nounion flag is TRUE, the algorithm compares the extracted certificate INRs with the INRs specified in the constraints file. If the two resource sets are different, the algorithm SHOULD issue a warning noting the difference. An output resource set is then formed that is identical to the resource set extracted from the certificate. If, however, the resource\_nounion flag is FALSE, then the output resource set is calculated by forming the union of the resources extracted from the certificate and the resources specified for this target block in the constraints file. A paracertificate is then constructed according to Table 1, using fields from the original certificate, the tags that had been set during

stage 0, and, if necessary, fields from the RP's TA certificate. The INR resources of the paracertificate are equated to the derived output resource set. The PARA state bit is set for the newly created paracertificate.

#### 4.2.3 Ancestor processing (stage 2)

The goal of ancestor processing is to discover all ancestors of a target certificate and remove from those ancestors the resources specified in the target blocks corresponding to the targets being processed. Note that it is possible that, for a given chain from a target certificate to a trust anchor, another target might be encountered. This is handled by removing all the target resources of all descendants. The set of all targets that are descendants of the given certificate is formed. The union of all the target resources of the corresponding target blocks is computed, and this union is then removed from the shared ancestor.

In detail, the algorithm is as follows. First, all (original) target certificates processed during stage 1 processing are collected. Second, any collected certificates that have the NOCHAIN state bit set are eliminated from the collection. (Note that, as a result of eliminating such certificates, the resulting collection may be empty, in which case this stage of algorithm processing terminates, and processing advances to stage 3.) Next, an implementation MAY sort the collection. The optional sorting algorithm is described in Appendix B. Note that all stage 2 processing is completed before any stage 3 processing.

Two levels of nested iteration are performed. The outer iteration is effected over all certificates in the collection; the inner iteration is over all ancestors of the designated certificate being processed. The first certificate in the collection is chosen, and a resource set R is initialized based on the resources of the target block for that certificate (since the certificate is in the collection, it must be a target certificate, and thus correspond to a target block). The parent of the certificate is then located using ordinary path discovery over original certificates only. The ancestor's certificate resources A are then extracted. These resources are then perforated with respect to R. That is, an output set of resources is created by forming the intersection I of A and R, and then taking the set difference  $A - I$  as the output resources. A paracertificate is then created containing resources that are these output resources, and containing other fields and extensions from the original certificate (and possibly the RP's TA certificate) according to the procedure given in Table 1. The PARA state bit is set on this paracertificate and the ORIGINAL state bit is set on A. If A is also a target certificate, as indicated by its TARGET state bit being set, then



there will already have been a paracertificate created for it. This previous paracertificate is destroyed in favor of the newly created paracertificate. In this case also, the set R is augmented by adding into it the set of resources of the target block for A. The algorithm then proceeds to process the parent of A. This inner iteration continues until the self-signed certificate at the root of the path is encountered and processed. The outer iteration then continues by clearing R and proceeding to the next certificate in the target collection.

Note that ancestor processing has the potential for order dependency, as mentioned earlier in this document. If sorting is not implemented, or if the sorting algorithm fails to completely process the collection of target certificates because the allotted maximum number of iterations has been realized, it may be the case that an ancestor of a certificate logically occurs before that certificate in the collection. Whenever an existing paracertificate is replaced by a newly created paracertificate during ancestor processing, the algorithm SHOULD alert the user, and SHOULD log sufficient detail such that the user is able to determine which resources were perforated from the original certificate in order to create the (new) paracertificate.

In addition, implementations MUST provide for conflict detection and notification during ancestor processing. During ancestor processing a certificate may be encountered two or more times and the modifications dictated by the ancestor processing algorithm may be in conflict. If this situation arises the algorithm MUST refrain from processing that certificate. Further, the implementation MUST present the user with an error message that contains enough detail so that the user can locate those directives in the constraints file that are creating the conflict. For example, during one stage of the processing algorithm it may be directed that resources R1 be added to a certificate C, while during a different stage of the processing algorithm it may be directed that resources R2 be removed from certificate C. If the resource sets R1 and R2 have a non-empty intersection, that is a conflict.

#### 4.2.4 Tree processing (stage 3)

The goal of tree processing is to locate other certificates containing INRs that conflict with the resources allocated to a target, by virtue of the INRs specified in the constraints file. The certificates processed are not ancestors of any target. The algorithm used is described below.

First, all target certificates are collected. Second, all target certificates that have the NOCHAIN state bit set are eliminated from this collection. Third, if the intersection\_always

global flag is set, target blocks that occur in the constraints file, but that did not correspond to a certificate in the local repository, are added to the collection. In tree processing, unlike ancestor processing, this collection is not sorted. An iteration is now performed over each certificate (or set of target block resources) in the collection. Note that the collection may be empty, in which case this stage of algorithm processing terminates, and processing advances to stage 4. Note also that all stage 3 processing is performed before any stage 4 processing.

Given a certificate or target resource block, each top level original TA certificate is examined. If that TA certificate has an intersection with the target block resources, then the certificate is perforated with respect to those resources. A paracertificate is created based on the contents of the original certificate (and possibly the RP's TA certificate, as indicated in Table 1) using the perforated resources. The ORIGINAL state bit is set on the original certificate processed in this manner, and the PARA state bit is set on the paracertificate just created. An inner iteration then begins on the descendants of the original certificate just processed. There are two ways in which this iteration may proceed. If the treegrowth global flag is clear, then examination of the children proceeds until all children are exhausted, or until one child is found with intersecting resources. If the treegrowth global flag is set, all children are examined. If a transfer of resources is in process, more than one child may possess intersecting resources. In this case, it is RECOMMENDED that the treegrowth flag be set. The inner iteration proceeds until all descendants have been examined and no further intersecting resources are found. The outer iteration then continues with the next certificate or target resource block in the collection. Note that unlike ancestor processing, there is no concept of a potentially cumulating resource collection R; only the resources in the target block are used for perforation.

#### 4.2.5 TA re-parenting (stage 4)

In the final stage of TA algorithm processing, all TA certificates (other than the RP's TA certificate) that have not already been processed are now processed. At this stage all unprocessed TA certificates have no intersection with any target resource blocks. As such, in creating the corresponding paracertificates, the output resource set is identical to the input resource set. Other transformations as described in Table 1 are performed. The original TA certificates have the ORIGINAL state bit set; the newly created paracertificates have the PARA state bit set. Note that once stage four processing is completely, only a single TA certificate will remain in an unprocessed state, namely the relying party's own TA certificate.

#### 4.3 Discussion

The algorithm described in this document effectively creates two coexisting certificate hierarchies: the original certificate hierarchy and the paracertificate hierarchy. Original certificates are not removed during any of the processing described in the previous section. Some original certificates may move from having no state bits set (or only the NOCHAIN state bit set) to having one or both of the ORIGINAL and TARGET state bits set. In addition, the NOCHAIN state bit will still be set if it was set before any processing. The paracertificate hierarchy, however, is intended to supersede the original hierarchy for ROA validation. The presence of two hierarchies has implications for path discovery, and for revocation.

If one thinks of a certificate as being "named" by its SKI, then there can now be two certificates with the same name, an original certificate and a paracertificate. The next two sections discuss the implications of this duality in detail. Before proceeding, it is worth noting that even without the existence of the paracertificate hierarchy, cases may exist in which two or more original certificates have the same SKI. As noted earlier, in Section 4.2.2, these cases may be subdivided into the case in which such certificates are distinguishable by virtue of having different subject names, but identical issuers and resource sets, versus all other cases. In the distinguishable case, the path discovery algorithm treats the original certificates as separate certificates, and processes them separately. In all other cases, the original certificates should be treated as indistinguishable, and path validation should fail.

#### 5 Implications for Path Discovery

Path discovery proceeds from a child certificate C by asking for a parent certificate P such that the AKI of C is equal to the SKI of P. With one hierarchy this question would produce at most one answer. With two hierarchies, the original certificate hierarchy and the paracertificate hierarchy, the question may produce two answers, one answer, or no answer. Each of these cases is considered in turn.

##### 5.1 Two answers

If two paths are discovered, it SHOULD be the case that one of the matches is a certificate with the ORIGINAL state bit set and the PARA state bit clear, while the other match inversely has the ORIGINAL state bit clear and the PARA state bit set. If any other combination of ORIGINAL and PARA state bits obtains, the path discovery algorithm MUST alert the user. In addition, the path discovery algorithm SHOULD refrain from attempting to make a

choice as to which of the two certificates is the putative parent. In the no-error case, with the state bits as indicated, the certificate with the PARA state bit set is chosen as the parent P. Note this means, in effect, that all children of the original certificate have been re-parented under the paracertificate.

## 5.2 One answer

If the matching certificate has neither the ORIGINAL state bit set nor the PARA state bit set, this certificate is the parent. If the matching certificate has the PARA state bit set but the ORIGINAL state bit not set, this certificate is the parent. (This situation would arise, for example, if the original certificate had been revoked by its issuer but the paracertificate had not been revoked by the RP.) If the matching certificate has the ORIGINAL state bit set but the PARA state bit not set, this is not an error but it is a situation in which path discovery MUST be forced to fail. The parent P MUST be set to NULL, and the NOCHAIN state bit must be set on C and all its descendants; the user SHOULD be warned. Even if the RP has revoked the paracertificate, the original certificate MAY persist. Forcing path discovery to unsuccessfully terminate is a reflection of the RP's preference for path discovery to fail as opposed to using the original hierarchy. Finally, if the matching certificate has both the ORIGINAL and PARA state bits set, this is an error. The parent P MUST be set to NULL, and the user MUST be warned.

## 5.3 No answer

This situation occurs when C has no parent in either the original hierarchy or the paracertificate hierarchy. In this case the parent P is NULL and path discovery terminates unsuccessfully. The NOCHAIN state bit must be set on C and all its descendants.

## 6 Implications for Revocation

In a standard implementation of revocation in a PKI, a valid CRL names a (sibling) certificate by serial number. That certificate is revoked and is purged from the local RPKI repository. The original certificate hierarchy and the paracertificate hierarchy created by applying the algorithms described above are closely related. It can thus be asked how revocation is handled in the presence of these two hierarchies. In particular do changes in one of the hierarchies trigger corresponding changes in the other hierarchy. There are four cases based on the state of the ORIGINAL and PARA bits. These are discussed in the subsections below. It should be noted that the existence of two hierarchies presents a particular challenge with respect to revocation. If a CRL arrives and is processed, that

processing can result in the destruction of one of the path chains. In the case of a single hierarchy this would mean that certain objects would fail to validate. In the presence of two hierarchies, however, a CRL revocation may force the preferred path to be destroyed. If the RP later determines that the CRL revocation should not have occurred, he is faced with an undesirable situation: the deprecated path will be discovered. In order to prevent this outcome, an RP MUST be able to configure one or more additional repository URIs in support of local trust anchor management.

#### 6.1 No state bits set

If the CRL names a certificate that has neither the ORIGINAL state bit set nor the PARA state bit set, revocation proceeds normally. All children of the revoked certificate have their state modified so that the NOCHAIN state bit is set.

#### 6.2 ORIGINAL state bit set

If the CRL names a certificate with the ORIGINAL state bit set and the PARA state bit clear, then this certificate is revoked as usual. If this original certificate also has the TARGET state bit set, then the corresponding paracertificate (if it exists) is not revoked; if this original certificate has the TARGET state bit clear, then the corresponding paracertificate is revoked as well. Note that since all the children of the original certificate have been re-parented to be children of the corresponding paracertificate, as described above, the revocation algorithm MUST NOT set the NOCHAIN state bit on these children unless the paracertificate is also revoked. Note also that if the original certificate is revoked but the paracertificate is not revoked, the paracertificate retains its PARA state bit. This is to ensure that path discovery proceeds preferentially through the paracertificate hierarchy, as described above.

#### 6.3 PARA state bit set

If the CRL names a certificate with the PARA state bit set and the ORIGINAL state bit clear, this CRL must have been issued, perforce, by the RP itself. This is because all the paracertificates are children of the RP's TA certificate. (Recall that a TA is not revoked via a CRL; it is merely removed from the repository.) The paracertificate is revoked and all children of the paracertificate have the NOCHAIN state bit set. No action is taken on the corresponding original certificate; in particular, its ORIGINAL state bit is not cleared.

Note that the serial numbers of paracertificates are synthesized according to the procedure given in Table 1, rather than being assigned by an algorithm under the control of the (original) issuer.

#### 6.4 Both ORIGINAL and PARA state bits set

This is an error. The revocation algorithm MUST alert the user and take no further action.

,

### 7 Security Considerations

The goal of the algorithm described in this document is to enable an RP to impose its own view of the RPKI, which is intrinsically a security function. An RP using a constraints file is trusting the assertions made in that file. Errors in the constraints file used by an RP can undermine the security offered by the RPKI, to that RP. In particular, since the paracertificate hierarchy is intended to trump the original certificate hierarchy for the purposes of path discovery, an improperly constructed paracertificate hierarchy could validate ROAs that would otherwise be invalid. It could also declare as invalid ROAs that would otherwise be valid. As a result, an RP must carefully consider the security implications of the constraints file being used, especially if the file is provided by a third party.

### 8 IANA Considerations

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

### 9 Acknowledgements

The authors would like to acknowledge the significant contributions of Charles Gardiner, who was the original author of an internal version of this document, and who contributed significantly to its evolution into the current version.

### 10 References

#### 10.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3513] Hinden, R., and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.
- [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, June 2004.
- [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.

- [RFC5396] Huston, G., and G. Michaelson, "Textual Representation of Autonomous System (AS) Numbers", RFC 5396, December 2008.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.
- [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Policy Structure", RFC 6481, February 2012.
- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, February 2012.

## 10.2 Informative References

None.

## Authors' Addresses

Stephen Kent  
Raytheon BBN Technologies  
10 Moulton St.  
Cambridge, MA 02138

Email: kent@bbn.com

Matthew Lepinski  
Raytheon BBN Technologies  
10 Moulton St.  
Cambridge, MA 02138

Email: mlepinsk@bbn.com

Mark Reynolds  
Island Peak Software  
328 Virginia Road  
Concord, MA 01742

Email: mcr@islandpeaksoftware.com

## Appendix A: Sample Constraints File

```
;
; Sample constraints file for TBO LTA Test Corporation.
;
; TBO manages its own local (10.x.x.x) address space
; via the target blocks in this file.
;

;
; Relying party subsection. TBO uses ssh-agent as
; a software cryptographic agent.
;

PRIVATEKEYMETHOD      OBO(ssh-agent)
TACERTIFICATE          tbomaster.cer

;
; Flags subsection
;
; Always use the resources in this file to augment
; certificate resources.
; Always process resource conflicts in the tree, even
; if the target certificate is missing.
; Always search the entire tree.
;

CONTROL  resource_nounion      FALSE
CONTROL  intersection_always   TRUE
CONTROL  treegrowth            TRUE

;
; Tags subsection
;
; Copy the original cert's validity dates.
; Use the default policy OID.
; Use our own CRLDP.
; Use our own AIA.
;

TAG      Xvalidity_dates      C
TAG      Xcp                  D
TAG      Xcrl dp              rsync://tbo_lta_test.com/pub/CRLs
TAG      Xaia                  rsync://tbo_lta_test.com/pub/repos

;
; Block subsection
```



```
;
;
; First block: TBO corporate
;

SKI 00112233445566778899998877665544332211
  IPv4
    10.2.3/24
    10.8/16
  IPv6
    2000:2:3:4:5:6/112
  AS#
    60123
    5507

;
; Second block: TBO LTA Test Enforcement Division
;

SKI 653420AF758421CF600029FF857422AA6833299F
  IPv4
    10.2.8/24
    10.47/16
  IPv6
  AS#
    60124

;
; Third block: TBO LTA Test Acceptance Corporation
; Quality financial services since sometime
; late yesterday.
;

SKI 19:82:34:90:8b:a0:9c:ef:00:af:a0:98:23:09:82:4b:ef:ab:98:09
  IPv4
    10.3.3/24
  IPv6
  AS#
    60125

; End of TBO constraints file
```

## Appendix B: Optional Sorting Algorithm for Ancestor Processing

Sorting is performed in an effort to eliminate any order dependencies in ancestor processing, as described in section 4.2.3 of this

document. The sorting algorithm does this by rearranging the processing of certificates such that if A is an ancestor of B, B is processed before A. The sorting algorithm is an OPTIONAL part of ancestor processing. Sorting proceeds as follows. The collection created at the beginning of ancestor processing is traversed and any certificate in the collection that is visited as a result of path discovery is temporarily marked. After the traversal, all unmarked certificates are moved to the beginning of the collection. The remaining marked certificates are unmarked, and a traversal again performed through this sub-collection of previously marked certificates. The sorting algorithm proceeds iteratively until all certificates have been sorted or until a predetermined fixed number of iterations has been performed. (Eight is suggested as a munificent value for the upper bound, since the number of sorting steps need not be any greater than the maximum depth of the tree.) Finally, the ancestor processing algorithm is applied in turn to each certificate in the remaining sorted collection. If the sorting algorithm fails to converge, that is if the maximum number of iterations has been reached and unsorted certificates remain, the implementation SHOULD warn the user.

Network Working Group  
Internet-Draft  
Updates: RFC 6490 (if approved)  
Intended status: Standards Track  
Expires: April 18, 2013

R. Gagliano  
Cisco Systems  
T. Manderson  
ICANN  
C. Martinez  
LACNIC  
October 15, 2012

Multiple Repository Publication Points support in the Resource Public  
Key Infrastructure (RPKI)  
draft-rogalia-sidr-multiple-publication-points-01

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 April 18, 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. Requirements notation . . . . .	3
2. Introduction . . . . .	4
3. Multiple Operators support in TAL files . . . . .	5
3.1. Update to RFC 6490 Section 2.1 . . . . .	5
3.2. Rules for Relying Parties (RP) . . . . .	6
4. Multiple Operators support in Certificates . . . . .	7
4.1. Rules for Relying Parties (RP) . . . . .	7
5. IANA Considerations . . . . .	8
6. Security Considerations . . . . .	9
7. Acknowledgements . . . . .	10
8. Normative References . . . . .	11
Authors' Addresses . . . . .	12

## 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

When thinking on how to scale the RPKI repository system described in [RFC6481] CA operators have a number of options such as:

- o Give the content to a Content Delivery Network (CDN) to have the content distributed (as long as the CDN supports the access method for the CA, which at this time is rsync).
- o 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.

When using any of these scaling technique to a unique CA Repository Publication Point URI, there is a dependency in the resolution of a single Fully Qualified Domain Name (FQDN). Also, when a single operator manages a RPKI Repository Publication Point, it is possible 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 idea of having multiple Repository Publication Points operators for a RPKI CA mitigates these risks and is complementary to any other scaling solution (as the ones described above).

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 remove this requirement by allowing one or more URI for each public key in a TAL file.

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] .

The addition of multiple Repository Publication Points operators for CAs in the RPKI introduces complexity for the RP. This documents provide some recommendations for RP implementors.

### 3. Multiple Operators support in TAL files

The idea of multiples operators support for a Trust Anchor 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
nbtxmLRW7B67SJCBSzfa5XpVyXYEgYAjk3fpmefU+AcxtxvvHB5OVPIa
BfPcs80ICMgHQX+fphvute9XLxjfJKJWkhZqZ0v7pZm2uhkcPx1PMGcrG
ee0WSDC3fr3erLueagpiLsFjwwpX6F+Ms8vqz45H+DKmYKvPSstZjCCq9
aJ0qANT9OtnfSDOS+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 in diversity, It is RECOMMENDED that 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](mailto:rogaglia@cisco.com)

Terry Manderson  
ICANN

Email: [terry.manderson@icann.org](mailto:terry.manderson@icann.org)

Carlos Martinez  
LACNIC

Email: [carlos@lacnic.net](mailto:carlos@lacnic.net)



Secure Inter-Domain Routing  
Internet-Draft  
Intended status: Informational  
Expires: March 26, 2013

K. Sriram  
D. Montgomery  
US NIST  
September 22, 2012

Design Discussion and Comparison of Replay-Attack Protection Mechanisms  
for BGPSEC  
draft-sriram-replay-protection-design-discussion-00

Abstract

The BGPSEC protocol requires a method for protection from replay attacks, at least to control the window of exposure. In the context of BGPSEC, a replay attack occurs when an adversary suppresses a prefix withdrawal (implicit or explicit) or replays a previously received BGPSEC announcement for a prefix that has since been withdrawn. This informational document provides design discussion and comparison of multiple alternative replay-attack protection mechanisms weighing their pros and cons. It is meant to be a companion document to the standards track I-D.-ietf-sidr-bgpsec-rollover that will specify a method to be used with BGPSEC for replay-attack protection.

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 March 26, 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. Definition of Replay Attack . . . . .	3
3. Classification of Solutions . . . . .	4
4. Expire Time Method . . . . .	5
5. Key Rollover Method . . . . .	6
5.1. Periodic Key Rollover Method . . . . .	7
5.2. Event-driven Key Rollover Method . . . . .	8
5.2.1. EKR-A: EKR where Update Expiry is Enforced by CRL . .	9
5.2.2. EKR-B: EKR where Update Expiry is Enforced by NotValidAfter Time . . . . .	10
5.2.3. EKR with Separate Key for Each Incoming-Outgoing Peering-Pair . . . . .	11
6. Summary of Pros and Cons . . . . .	12
7. Summary and Conclusions . . . . .	14
8. Acknowledgements . . . . .	16
9. IANA Considerations . . . . .	16
10. Security Considerations . . . . .	16
11. Informative References . . . . .	16
Authors' Addresses . . . . .	17

## 1. Introduction

The BGPSEC protocol [I-D.ietf-sidr-bgpsec-protocol] requires a method for protection from replay attacks, at least to control the window of exposure [I-D.ietf-sidr-bgpsec-reqs]. In the context of BGPSEC, a replay attack occurs when an adversary suppresses a prefix withdrawal or replays a previously received BGPSEC announcement for a prefix that has since been withdrawn.

In this informational document, we provide design discussion and comparison of various replay-attack protection mechanisms that may be used in conjunction with the BGPSEC protocol. It is meant to be a companion document to the standards track document [I-D.ietf-sidr-bgpsec-rollover] that will specify a method to be used with BGPSEC for replay-attack protection. Here we consider four alternative mechanisms - one based on the explicit Expire Time approach and three different variants based on the Key Rollover approach. We provide a detailed comparison between these mechanisms weighing their pros and cons. This document is meant to help inform the decision process leading to an exact description for the mechanism to be finalized and formally specified in [I-D.ietf-sidr-bgpsec-rollover].

## 2. Definition of Replay Attack

In the context of BGPSEC, a replay attack occurs when an adversary suppresses a prefix withdrawal (implicit or explicit). A replay attack occurs also when the adversary replays a previously received BGPSEC announcement for a prefix that has since been withdrawn. In the rest of this document, we will refer to either of these two situations as replay attack. The following are examples of replay attacks:

Example 1: AS1 has AS2 and AS3 as eBGPSEC peers. At time  $x$ , AS1 had announced a prefix  $P$  to AS2 and AS3. At a later time  $x+d$ , AS1 sends a Withdraw for prefix  $P$  to AS2. AS2 suppresses the Withdraw (does not send to its peers any explicit or implicit Withdraw). AS2 continues to attract some of the data for prefix  $P$  towards itself by pretending to still have a signed and valid route for  $P$ . In effect, AS2 can conduct a DOS attack on a server located at AS1 at prefix  $P$ . (See slide #2 in [replay-discussion] for an illustration.)

Example 2: AS1 has AS2 and AS3 as eBGPSEC peers. AS2 and AS3 are also eBGPSEC peers. At time  $x$ , AS1 had announced a prefix  $P$  to AS2 and AS3. AS3 also propagates to AS2 its route (via AS1) for prefix  $P$ . At a later time  $x+d$ , AS1 discontinues its peering with AS2. AS2 should propagate an alternate longer path via AS3 for prefix  $P$  and

thus send an implicit Withdraw. However, AS2 suppresses it. AS2 can thus make a significant part of traffic destined for prefix P to flow via itself and eavesdrop on the data but not cause a DOS attack. (See slide #3 in [replay-discussion] for an illustration.)

Example 3: AS1 has AS2 and AS3 as eBGPSEC peers. AS2 and AS3 are also eBGPSEC peers. At time x, AS1 had announced a prefix P to AS2 without prepending (Update: AS1{pCount=1} P) but announced the same prefix to AS3 with prepending (Update: AS1{pCount=2} P). Thus AS1 had preferred its ingress data traffic for prefix P to come in via AS2. At a later time x+d, AS1 switches ingress data path preference to AS3 over AS2 - announces prefix P without prepending (Update: AS1{pCount=1} P) to AS3 and with prepending (Update: AS1{pCount=2} P) to AS2. AS2 suppresses the new prepended path announcement (does not send to its peers any new update about P). Thus AS2 carries more of AS1's ingress data traffic and generates more revenue for itself at the expense of AS1. (See slide #4 in [replay-discussion] for an illustration.)

Thus the scenarios and motivations for replay attacks may differ as illustrated by the examples above.

A requirement for replay-attack protection can be stated as follows. The update that AS1 sent to AS2 at time x should expire at time x+w. That means, AS2 can suppress the Withdraw or possibly replay the update from AS1 for prefix P until at most x+w. This limits the replay vulnerability window. (Note: If no peering or policy change affecting prefix P occurs during the vulnerability window, then a typical solution would include a method for extending the validity period of the route(s) beyond x+w.)

### 3. Classification of Solutions

Mechanisms for replay-attack protection can be classified into two broad categories as follows:

- o Expire Time (ET) Method: This method uses an explicit expire time field in the BGPSEC update.
- o Key Rollover (KR) Method: In this method, the update expiry is enforced by a key rollover. Router rolls over to a new signing cert with a new pair of keys, and the previous router cert either expires or is revoked.

The Key Rollover method can be further characterized into the following sub categories:

- o Periodic Key Rollover (PKR): Key rollovers happen at periodic intervals.
- o Event-driven Key Rollover (EKR): Key rollovers happen only when peering or policy change events occur.
  - \* EKR-A: EKR where expiry of previous update is enforced by CRL.
  - \* EKR-B: EKR where expiry of previous update is controlled by NotValidAfter time.

In Section 4, Section 5, and Section 6 we describe the various methods listed above, and discuss their pros and cons.

#### 4. Expire Time Method

The details of the Expire Time (ET) method are as follow:

- o Explicit Expire Time is used for origin's signature.
- o Expire Time field is required in the BGPSEC update.
- o Periodic re-origination (beaconing) of prefixes is performed by origin ASes. The value in the ET field in the update is extended at beaconing time, and thereby the update is refreshed. Every prefix in the Internet is re-originated and propagates through the Internet once every 'beacon' interval.
- o These beacons are distributed actions by prefix owners and jittered in time by design to reduce burstiness. The beacon interval can be different at different originating ASes.
- o Beacon interval granularity: TBD but preferably in fairly granular units (days).

Discussion of Pros and Cons:

Pro: This method is easy on transit routers. In the event of peering or policy change, BGPSEC with the ET method behaves the same way as BGP-4 in terms of which prefix routes are propagated. That is, the router re-evaluates best paths factoring in peering or policy changes, and propagates only those prefix routes that have a change in best path. In other words, there is no necessity for the BGPSEC router to re-propagate and refresh prefixes on all peering links. This is because prefix updates are refreshed anyway once every beacon interval by all prefix originators. There is low steady-state traffic associated with beaconing (see Figure on slide #5 in

[replay-discussion]), but there are no huge bursts or spikes in workload due to peering or policy change events at transit routers.

Con: Equipment vendor can potentially facilitate unnecessary frequent beaconing if ISP urges and pays (dollar attack!). This possibility is mitigated by having a well thought-out granularity for ET, for example, if the unit of ET is one day (rather than one minute).

Con: A change in on-the-wire BGPSEC protocol would be needed in case the unit of the ET field (granularity) needs to be changed.

## 5. Key Rollover Method

Key Rollover (KR) method has three variations as outlined in Section 3. Those will be discussed later in this section. The following features are common to all variants of the KR method:

- o In the KR method, it is best if the BGPSEC router has two pairs of certs as follows: A pair of origination certs (current and next) for signing prefixes being originated by the AS of the router, and a pair of transit certs (current and next) for signing transit prefixes.
- o Note: If a BGPSEC router only originates prefixes (i.e., has no transit prefixes), then it needs to maintain only a pair of origination certs and need not maintain the extra pair of transit certs.
- o The three KR methods differ in how the rollover of certs (or keys) is done:
  - \* Cert rollovers are Periodic vs. Event-driven.
  - \* In the Event-driven method, the expiry of old update is (A) Enforced by CRL vs. (B) Controlled by NotValidAfter time.
  - \* In (A), cert's NotValidAfter field is set to a very large value and CRL is issued to revoke the cert when necessary. In (B), NotValidAfter field set to a permissible vulnerability window time and CRL to revoke cert is not required.

Discussion of Pros and Cons (common to all Key Rollover methods):

Pro: The KR method functions by manipulating the RPKI objects (certs, keys, NotValidAfter field in cert, etc.) to refresh updates or to cause expiry of previously propagated updates. Unlike the ET method, it does not rely on any explicit field in the update. Hence, an

advantage of the KR method over the ET method is that in case any parameters need to change or if the method itself is modified, then there is no impact on the BGPSEC protocol on the wire.

Con: The KR method introduces additional churn in the global RPKI system.

Con: There is also added update churn. The amount of update churn varies depending on the type of KR method used (see Section 5.1 and Section 5.2).

We will now describe and discuss in detail the variants of the KR method.

### 5.1. Periodic Key Rollover Method

The details of the Periodic Key Rollover (PKR) method are as follow.

- o Router's origination cert's NotValidAfter time is used as the implicit expire time for origin's signature.
- o Each origination router re-originates (i.e., beacons) before NotValidAfter time of the current cert. Beaconing is periodic re-origination of prefixes by origin ASes.
- o At beaconing time, next cert becomes the new current cert, and update is signed with the private key of this new current cert and re-originated.
- o A new 'next' cert is created and propagated at beaconing time. This can also be done with a good lead time. In practice, multiple 'next' certs can be kept in the pipeline. They must have contiguous or slightly overlapping validity periods.
- o Every prefix in the Internet is re-originated and propagates through the Internet once every 'beacon' interval.
- o The re-originations or beacons are distributed actions by prefix owners and jittered in time by design to reduce burstiness. The beacon interval can be different at different originating ASes.
- o Beacon (or re-origination) interval granularity: TBD but preferably in fairly granular units (days).
- o Transit certs can have very large NotValidAfter time (say ~years).
- o When a peering or policy change event occurs at a transit router, the router (i.e. BGPSEC router with PKR) does not perform any key

rollover. The router re-evaluates best paths factoring in peering or policy changes, and propagates only those prefix routes that have a change in best path (similar to BGP-4). There is no necessity for the BGPSEC router to re-propagate and refresh prefixes on all peering links. This is because prefix updates are refreshed anyway once every re-origination (i.e. beaconing) interval by all prefix originators.

#### Discussion of Pros and Cons:

Several of the same pros/cons of the Expire Time method also apply here for the PKR method.

Pro: The main pro for the PKR method is the same as that for the Expire Time (ET) method. That is, being easy on transit routers as discussed in Section 4. Just as in the ET method, there is low steady-state traffic associated with periodic re-originations (i.e. beaconing) (see Figure on slide 5 in [replay-discussion]), but there are no huge bursts or spikes in workload due to peering or policy change events at transit routers. (See comparisons with the EKR methods in Section 5.2.)

Pro: The pro discussed above for the KR method regarding parameter changes (e.g., beacon interval units) not requiring change of protocol on the wire is naturally applicable here.

Con: Churn in the RPKI is of concern. Every BGPSEC router rolls two origination certs (current and next) once in every beacon (i.e., re-origination) interval.

### 5.2. Event-driven Key Rollover Method

The common details of the Event-driven Key Rollover (EKR) methods are as follow.

- o Key rollover is reactive to events (not periodic).
- o If a peering or policy change event involves only prefixes being originated at the AS of the router, then the router rolls only the origination key.
- o If a peering change event involves transit prefixes at the AS of the router, then the router rolls the transit key as well as the origination key.
- o If a key rollover takes place, then a corresponding (origination or transit) new 'next' cert is propagated in RPKI.

#### Discussion of Pros and Cons:

Pro: As long as no triggering events occur, there is no added update churn in BGPSEC.

Con: Whenever the transit key is rolled, there is a storm of BGPSEC updates at routers in transit ASes. For example, consider BGPSEC capable transit AS5 that is connected to four BGPSEC non-stub customers (AS1, AS2, AS3, AS4). Assume each AS has a single BGPSEC router in it. AS1 through AS4 each receives almost full table (400K signed prefix updates) from AS5. Assume also that AS1 and its customers together originate 100 prefixes in total; likewise for AS2, AS3 and AS4. Now consider that an event occurs whereby the peering between AS1 and AS5 is discontinued. As a result of this event, in the EKR method, the AS5 router signs and re-propagates approximately  $3 \times 400K = 1.2$  Million signed prefix updates to AS2, AS3 and AS4 combined. In addition, it also sends  $4 \times 100 = 400$  Withdraws, which are negligible. In comparison, in the PKR method, following the same event, the router at AS5 sends only  $4 \times 100 = 400$  Withdraws and signs/re-propagates ZERO prefix updates. (An illustration can be found in slide #6 in [replay-discussion]. Also, additional peering change scenarios and quantitative comparisons can be found in slides #7 and #8 in [replay-discussion].)

It remains to be seen through measurement and modeling how the impact of such large bursts of workload in the ETR method at the time of event occurrence can be managed in route processors, e.g., by jittering and throttling the workload.

#### 5.2.1. EKR-A: EKR where Update Expiry is Enforced by CRL

EKR-A builds on the common principles as described for EKR above in Section 5.2. The additional details of EKR-A operation are as follow:

- o NotValidAfter time of origination and transit certs is set to a large value (~year).
- o Whenever key rollover (for origination or transit) occurs, then CRL is propagated for the old cert. So the old update expires (due to invalid state) only when the CRL propagates and reaches the relying router.
- o This method relies on end-to-end CRL propagation through the RPKI system to enforce expiry of a previous update whenever the need arises.



- o The cert CRL either propagates all the way to the relying router, or the RPKI cache server of the router receives the CRL and then sends a withdrawal of the {AS, SKI, Pub Key} tuple to the router. Either way, the CRL must in effect propagate all the way to the relying router.
- o Thus the attack vulnerability window with the EKR-A method is governed by the end-to-end CRL propagation time.

#### Discussion of Pros and Cons:

The following pro and con for the EKR-A method are in addition to the common pros and cons listed above for the KR and EKR methods (Section 5 and Section 5.2).

Pro: EKR-A has much less RPKI churn than PKR or EKR-B (see Section 5.2.2).

Con: Router needs to receive a CRL or a withdraw of {AS, SKI, Pub Key} tuple in order to know an update has expired. Hence, the replay-attack vulnerability window is determined by the CRL propagation time which can vary widely from one relying router to another router that may be in different regions. It is anticipated that this would be no worse than 24 hours, but needs to be confirmed by measurements in an operational or emulated RPKI systems [rpki-propagation].

#### 5.2.2. EKR-B: EKR where Update Expiry is Enforced by NotValidAfter Time

EKR-B builds on the common principles as described for EKR above in Section 5.2. The additional details of EKR-B operation are as follow:

- o NotValidAfter time of current origination and transit certs is set to a value determined by the desired vulnerability window (~day).
- o Update expiry is controlled by NotValidAfter time and CRL is not sent for the old cert when key rollover happens.
- o If no triggering event occurs to cause origination key rollover within a pre-set time (NotValidAfter), then new origination (current and next) certs are issued only to extend the NotValidAfter time but the corresponding key pairs and SKIs remain unchanged.
- o A previous update automatically becomes invalid at the earliest NotValidAfter time of the certs used in the signatures unless each of those certs' NotValidAfter time has been extended.

- o Likewise for the transit (current and next) certs and keys.
- o Changes in certs to extend their NotValidAfter time need not propagate end-to-end (all the way to the relying routers); they may propagate only up to the RPKI cache server of the relying router. RPKI cache server would send a withdraw for an {AS, SKI, Pub Key} tuple to a relying router if the NotValidAfter time of the cert has passed.
- o The changes in certs to advance NotValidAfter time can be scheduled and propagated in RPKI well in advance.

#### Discussion of Pros and Cons:

The following pro and con for EKR-B are in addition to the common pros and cons listed above for the KR and EKR methods (Section 5 and Section 5.2).

Pro: Update expiry is automatic in case the NotValidAfter time of any of the certs used to sign the update has not been extended. So the replay-attack vulnerability window is predictable and not influenced by the RPKI end-to-end propagation time.

Pro: Routers do not get any RPKI updates from the RPKI cache server when cert changes but the key pair and SKI remain unchanged. Routers do not receive NotValidAfter time from their RPKI cache server. There is no need for it. Instead, the RPKI cache server keeps track of NotValidAfter time, and provides to routers only valid {AS, SKI, Pub Key} tuples. This saves some RPKI state maintenance workload at the routers.

Con: EKR-B has much more RPKI churn than EKR-A because both origination and transit certs need to be reissued periodically to extend their validity time (in the absence of any events).

#### 5.2.3. EKR with Separate Key for Each Incoming-Outgoing Peering-Pair

This is a place holder section where we mention another variant of the EKR method. This idea has not been considered or whetted by the SIDR WG yet. So we only mention it here briefly.

As noted earlier, the EKR methods considered so far generate a huge spike in workload whenever the transit key rollover takes place at a router. One way to reduce that workload is to have a separate signing key for each incoming-outgoing peering pair. For example, consider a BGPSEC router in AS4 that has peers in AS1, AS2, and AS3. The router will hold six signing keys, one each corresponding to (AS1, AS2), (AS2, AS1), (AS1, AS3), (AS3, AS1), (AS2, AS3), and (AS3,

AS2) peering-pairs. Note that the directionality of peering is included here and is necessary. The key corresponding to (AS-i, AS-j) would only be used to sign updates received from AS-i and being forwarded to AS-j. In the general case, when the BGPSEC router has  $n$  peers, the number of transit keys will be  $n(n-1)$ . Since there would be a Current and a Next key (for rollover), the number of transit keys held in the router for signing will be actually  $2n(n-1)$ . When a peering or policy change occurs, the router would rollover only those specific keys that correspond to the peering-pairs over which the prefix updates are affected. In the above example, suppose a policy change between AS4 and AS1 causes AS4 to prepend prefixes sent to AS1 (pCount changed from 1 to 2). Then AS4 would do key rollover only for (AS2, AS1) and (AS3, AS1) peering-pairs, and not for any of the others. This would substantially reduce the quantity of prefix updates that are signed and re-propagated. In general, when peering or policy changes occur, this method will reduce the number of prefix updates to be re-propagated to exactly the same as that with normal BGP. That means that this method would also be on par with the ET and PKR methods in terms of update churn when a peering or policy change takes place. The downside of this method is that the router needs to maintain  $2n(n-1)$  key pairs if it has  $n$  BGPSEC peers.

Detailed discussion and comparison of this method with other methods can be provided in a later version of this document if the idea picks up interest in the WG.

## 6. Summary of Pros and Cons

Table 1 below summarizes the pros and cons for the various replay-attack protection methods. This summary follows from the discussion above in Section 4 and Section 5.

Method	Pros	Cons
Expire Time (ET)	<p>1. The background load due to beaconing is low and not bursty.</p> <p>---</p> <p>2. Transit AS does NOT have a huge spike in workload even when a peering or policy change happens at that AS. Beaconing facilitates this.</p> <p>---</p> <p>3. Does not add to RPKI churn.</p>	<p>1. Prefix owner can abuse by beaconing too frequently.</p> <p>---</p> <p>2. Any change to units (granularity) of the ET field entails a change to on-the-wire BGPSEC protocol.</p> <p>---</p>
Periodic Key Rollover (PKR)	<p>1. The background load due to beaconing is low and not bursty.</p> <p>---</p> <p>2. Transit AS does NOT have a huge spike in workload even when a peering change happens at that AS. Beaconing (i.e. periodic re-origination) facilitates this.</p> <p>---</p> <p>3. If the periodic re-origination (i.e., beaconing) interval units change, BGPSEC protocol on the wire remains unaffected.</p> <p>---</p> <p>4. Changes in the method (while still based on Key Rollover) can be accommodated without requiring any change to on-the-wire BGPSEC protocol.</p>	<p>1. Prefix owner can abuse by beaconing (i.e. re-originating) too frequently.</p> <p>---</p> <p>2. Adds to RPKI churn. A pair of certs (current and next) for each origination router are rolled once every beacon (i.e. re-origination) interval. Significantly more RPKI churn than that with EKR-A or EKR-B methods.</p> <p>---</p>

Event driven Key Rollover Type A (EKR-A)	1. No update churn for long periods when no peering or policy changes occur.	1. Whenever the transit key is rolled (in response to a peering or policy change event), there is a storm of BGPSEC updates, especially at routers in large transit ASes.
	---	---
	2. The added churn in RPKI is much lower than that in the EKR-B method.	2. The replay-attack vulnerability window is dependent on end-to-end CRL propagation. It may vary significantly from one relying router to another that may be in different regions.
Event driven Key Rollover Type B (EKR-B)	---	---
	3. Same as Pro #4 for the PKR method.	1. Same as Con #1 for the EKR-A method.
	---	---
	1. Same as Pro #1 for the EKR-A method.	1. Same as Con #1 for the EKR-A method.
	---	---
	2. The replay-attack vulnerability window is enforced by NotValidAfter time in certs and is therefore predictable.	2. The added churn in RPKI is much higher than that in the EKR-A method.
	---	---
	3. Same as Pro #4 for the PKR method.	
	---	

Table 1: Table with Summary of Pros and Cons

## 7. Summary and Conclusions

We have attempted to provide insights into the operation of multiple alternative methods for replay-attack protection. It is hoped that the SIDR WG will take the insights and trade-offs presented here as input for deciding on the choice of a mechanism for protection from replay attacks. Once that decision is made, the chosen mechanism would be included in the standards track document

[I-D.ietf-sidr-bgpsec-rollover].

Some important considerations for the decision making can be possibly listed as follow:

1. The Expire Time (ET) method is best (on par with the PKR method) in terms of preventing huge update workloads during peering and policy change events at transit routers with several peers. It has no added RPKI churn. But the ET method has the disadvantage of requiring on-the-wire protocol change if some parameters (e.g., the units of beacon interval) change.
2. The Periodic Key Rollover (PKR) method operates the same way as the ET method for preventing huge update workloads during peering and policy change events at transit routers with several peers. It does not have the disadvantage of requiring on-the-wire protocol change if some parameters (e.g., the units of beacon/re-origination periodicity) change. But it has the downside of added RPKI churn.
3. The Event-driven Key Roll (EKR-A and EKR-B) methods have significantly less RPKI churn than the PKR method. They also have no BGPSEC update churn during long quiet periods when no peering or policy change events occur. But they suffer the drawback of creating huge update workloads during peering and policy change events at transit routers with several peers. Can this workload be jittered or flow controlled to spread it over time without convergence delay concerns? May be - needs further study.
4. The EKR-A method relies on end-to-end CRL propagation through the RPKI system to enforce expiry of a previous update when needed. By contrast, in the EKR-B method the update expiry is controlled by NotValidAfter time of the certs used in update signatures. In EKR-B, previous update automatically becomes invalid at the earliest NotValidAfter time of the certs used in the signatures unless each of those certs' NotValidAfter time has been extended. In the latter method, changes in certs to extend their NotValidAfter time need not propagate end-to-end (all the way to the relying routers); they may propagate only up to the RPKI cache server of the relying router (see Section 5.2.2). The changes in certs to advance NotValidAfter time can be scheduled and propagated in RPKI well in advance.
5. Besides being out-of-band relative to the BGPSEC protocol on the wire, the other good thing about the Key Rollover method is that once the basics of the mechanism are implemented, there may be flexibility to implement PKR, EKR-A or EKR-B on top of it. It

may also be possible to switch from one method to another (within this class) if necessary based on operational experience; this transition would not require any change to on-the-wire BGPSEC protocol.

## 8. Acknowledgements

The authors would like to thank Brian Weis and Steve Kent for helpful discussions. Further, we are thankful to fellow NIST BGP team members for comments and suggestions.

## 9. IANA Considerations

This memo includes no request to IANA.

## 10. Security Considerations

This memo requires no security considerations of its own since it is targeted to be an informational RFC in support of [I-D.ietf-sidr-bgpsec-rollover] and [I-D.ietf-sidr-bgpsec-protocol]. The reader is therefore directed to the security considerations provided in those documents.

## 11. Informative References

[I-D.ietf-sidr-bgpsec-protocol]

Lepinski, M., "BGPSEC Protocol Specification",  
draft-ietf-sidr-bgpsec-protocol-05 (work in progress),  
September 2012.

[I-D.ietf-sidr-bgpsec-reqs]

Bellovin, S., Bush, R., and D. Ward, "Security  
Requirements for BGP Path Validation",  
draft-ietf-sidr-bgpsec-reqs-04 (work in progress),  
June 2012.

[I-D.ietf-sidr-bgpsec-rollover]

Gagliano, R., Patel, K., and B. Weis, "BGPSEC router key  
rollover as an alternative to beaconing",  
draft-ietf-sidr-bgpsec-rollover-00 (work in progress),  
August 2012.

[replay-discussion]

Sriram, K. and D. Montgomery, "Comparison of Replay-Attack

Mitigation Mechanisms for BGPSEC", September 2012, <<http://www.nist.gov/itl/anttd/upload/replay-discussion.pdf>>.

[rpki-propagation]

Bush, R, et al., "RPKI Propagation Emulation Measurement: an Early Report", July 2012, <<http://www.ietf.org/proceedings/interim/2012/07/27/sidr/slides/slides-interim-2012-sidr-5-4.pdf>>.

#### Authors' Addresses

Kotikalapudi Sriram  
US NIST

Email: [ksriram@nist.gov](mailto:ksriram@nist.gov)

Doug Montgomery  
US NIST

Email: [doug@nist.gov](mailto:doug@nist.gov)





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 02, 2013

R. Bush  
Internet Initiative Japan  
K. Patel  
P. Mehta  
A. Sreekantiah  
Cisco Systems  
L. Jalil  
Verizon  
October 2012

Authenticating L3VPN Origination Signaling  
draft-ymbk-l3vpn-origination-02

Abstract

A BGP-signaled Layer-3 VPN's prefix bindings sent over BGP are subject to unintentional errors, both by the legitimate originator and by non-legitimate origins. This is of special concern if the VPN traverses untrusted networks. This document describes how the sender of the Prefix/VPN binding may sign it so that recipient of the binding may authenticate it.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC2119] 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 02, 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.

This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

## Table of Contents

1. Introduction . . . . .	2
2. NLRI Deaggregation . . . . .	3
3. L3VPN Origination BGP Path Attribute (L3OPA) . . . . .	3
4. Validation of Routes Having an L3OPA . . . . .	4
5. L3VPN Deployment Scenarios . . . . .	5
5.1. End CE to CE Authentication . . . . .	5
5.2. Provider/ASBR Based Validation/Authentication . . . . .	5
5.3. PE-PE Based Validation . . . . .	6
6. Notes . . . . .	6
7. Security Considerations . . . . .	7
8. IANA Considerations . . . . .	7
9. Acknowledgements . . . . .	7
10. References . . . . .	7
10.1. Normative References . . . . .	7
10.2. Informative References . . . . .	8
Authors' Addresses . . . . .	8

## 1. Introduction

RFC 4364 [RFC4364] Section 7.4 describes how a Customer Edge (CE) router uses eBGP to announce to a Provider Edge (PE) router the address prefix(es) the customer provides to an L3VPN. It is possible that the originator of such an announcement could unintentionally announce prefixes they do not own.

```
Cust(West)-CE--PE-Provider(West)--TransitA-~
~-TransitB--Provider(East)-PE--CE-Cust(East)
```

This document describes how the PE receiving the CE's originating announcement, West, may sign the announcement so that the PE proximal to the destination CE, East, may authenticate the NLRI see RFC 4364 [RFC4364] Section 4.3.1. Alternatively, the originating CE router may sign the announcement so that the destination CE router may authenticate the NLRI.

It is assumed that the providers already have the key creation, storage, and distribution infrastructure needed. Keys might be configured on the routers, or in some shared PKI, or, for example, the Resource Public Key Infrastructure (RPKI) could be used, see RFC 6480 [RFC6480].

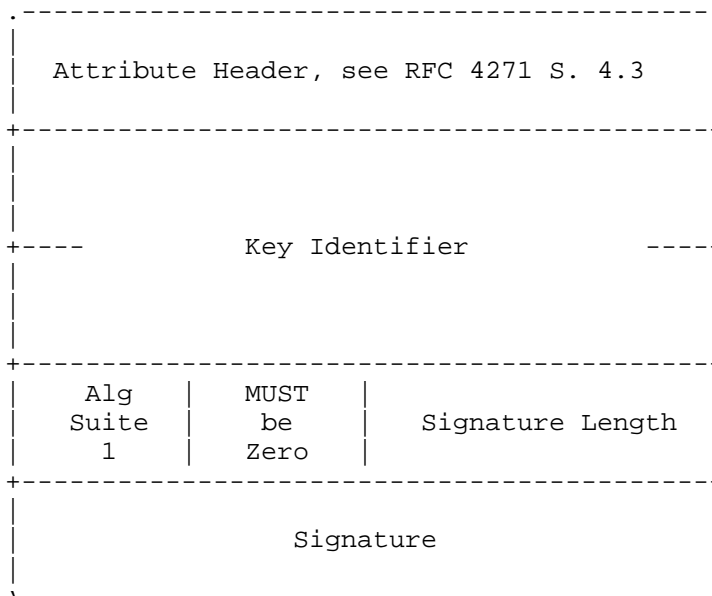
A new BGP PATH Attribute, called L3VPN Origination BGP PATH Attribute (L3OPA), is created to contain the necessary keying information and signature.

## 2. NLRI Deaggregation

Normally, a BGP Update may contain multiple NLRI which all share the identical set of attributes. As L3OPA signalling signs over the NLRI, and NLRI can become separated as they transit the network, separation would break the signature. Therefore, a BGP announcement using L3OPA signalling MUST contain one and only one NLRI.

## 3. L3VPN Origination BGP Path Attribute (L3OPA)

The L3OPA is a BGP optional transitive Path Attribute RFC 4271 [RFC4271]. BGP Path Attributes are Type/Length/Value tuples.



The Attribute Type is two octets, the first of which, Attribute Flags, MUST have the two high order bits set to signify that attribute is optional and transitive.

The second octet of the Attribute Flags, Attribute Type, MUST be set to 0xXX, as assigned by the IANA, see Section 8, to signal that this is an L3OPA.

The Length field is one or two octets with a value of the number of octets in the entire attribute. If the length of the L3OPA is less than 256 octets, only the first octet of the length field is used. Otherwise, both octets are used to represent the Length.. See RFC 4271 [RFC4271] Section 4.2 for another explanation of this byte saving.

The Key Identifier is an eight octet value identifying the key (pair) used for the Signature. It is used when the keying is not implied by the NLRI, as it would be, for example, if the RPKI was used. It is often the VPN Identifier. If not used to identify the key, it MUST be zero.

The Algorithm Suite is a one-octet identifier specifying the digest algorithm and digital signature algorithm used to produce the Signature. The values reference the IANA registry for Algorithm Identifiers from BGPsec, see [I-D.ietf-sidr-bgpsec-algs].

The Signature Length is two octets and is the number of octets in the Signature field.

The Signature field is a digital signature that covers the NLRI and the Key Identifier.

To compute the Signature, the digest algorithm for the specified Algorithm Suite is applied to the catenation of the NLRI and the Key Identifier. This is then fed to the signature algorithm for the specified algorithm suite and the resulting value is the Signature.

Signature = sign ( hash ( NLRI || Key Identifier ) )

#### 4. Validation of Routes Having an L3OPA

A BGP speaker receiving routes with an L3OPA MUST perform the necessary validation if configured to do so.

The digest algorithm for the specified Algorithm Suite is applied to the catenation of the NLRI and the Key Identifier. This is then fed to the signature algorithm for the specified algorithm suite and the resulting value is compared with the Signature.

If the signature value matches the Signature in the attribute, the route MUST be marked as Valid, otherwise it MUST be marked as Invalid.

A route received without an L3OPA SHOULD be marked as having an Unknown validity state.

If L3OPA marking is disabled in the router configuration, routes are

considered to have the Unknown validity state.

Configured local policy on the router may use the validity state markings to implement policy. For example, a route marked as Invalid or Unknown may be dropped or de-preferenced by appropriate use of normal BGP policy mechanisms.

Note that this is similar to announcement marking while allowing the user to control policy as described in RPKI-Based BGP origin validation, see [I-D.ietf-sidr-pfx-validate].

## 5. L3VPN Deployment Scenarios

The following L3VPN deployment scenarios illustrate use of the scheme. The examples use the language of symmetric keys which have been previously agreed upon between the signer of the route and the validator. Asymmetric keying, a PKI, etc. could also be used. Signing and validation are as described above.

### 5.1. End CE to CE Authentication

```
CE1 ---- PE1 ----- PE2 -- CE2
                AS1
```

```
CE1 ---- PE1 ----- ASBR1 ----- ASBR2 ----- PE2 ---- CE2
                AS1                      AS2
```

CE1 and CE2 are end CEs in the same VPN. PE1, PE2, ASBR1, ASBR2 are provider PE/ASBRs which are blindly propagating the announcement with the L3OPA as generated by CE1.

As the authorization is between the originating CE1 and the terminating CE2, the keying should not be known by the provider(s). The CEs are configured with the keying information, the originating CE1 creates and signs an L3OPA for each NLRI participating in the VPN.

An update received by CE2 without an L3OPA, or having an Invalid Signature would likely be dropped. Thus the CEs are protected from incorrect prefixes originating from a provider network or unauthorized CEs.

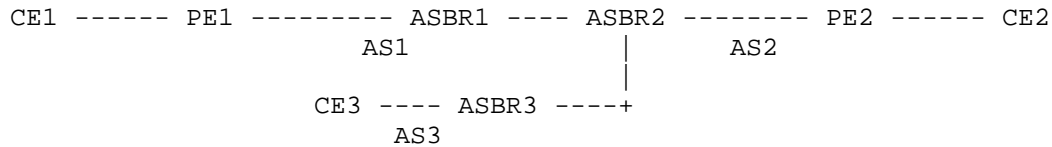
### 5.2. Provider/ASBR Based Validation/Authentication

```
CE1 ---- PE1 ----- ASBR1 ----- ASBR2 ----- PE2 ---- CE2
                AS1                      AS2
```

In the diagram, CE1 is the originating/signing CE. ASBR2 is the trusted provider with whom CE1 has collaborated. Updates generated by CE1 may be passed transparently through any number of intermediate providers, ASBR1s, which blindly propagate the L3OPA. Validation is performed when the announcement reaches the trusted validating provider, ASBR2.

Keying is agreed between CE1 and the trusted provider ASBR2, likely on per-VPN basis.

### 5.3. PE-PE Based Validation



Here PEs, possibly across ASes, agree on the keying. The Key Identifier and associated keys would normally be configured on a per VPN basis, with the PE1 signing and PE2 and PE3 validating similarly to the CEs in the previous examples.

CE1 originates an announcement, possibly with multiple NLRI, but without an L3OPA. PE1 de-aggregates the NLRI into separate announcements, signs each with the keying agreed with PE2 and PE3, and propagates them. Arbitrary providers carry the announcements toward PE2 and PE3, where the announcements have their Signatures validated, the L3OPAs removed, and are then propagated to CE2 and CE3.

### 6. Notes

The keying could either come from the Global RPKI or the customer or carrier running their own PKI. The keying is assumed to be asymmetric, but possibly could be symmetric. The keys can be statically configured (beware scaling and key-roll issues), dynamic, in some public or private infrastructure, etc.

If the RPKI is used, and the public key is taken from the CA certificate which owns the NLRI, the classic problem arises where all the NLRI on that certificate share fate. I.e. if one causes the need for a re-key, then all must re-key. RPKI-based origin validation solves this problem by a level of indirection, the CA certificate is used to sign an End Entity (EE) certificate which signs a Route Origin Authorization (ROA), see RFC 6480 [RFC6480] and RFC 6482 [RFC6482]. As the Key Identifier of an L3VPN signal is larger than the four octets of a ROA, a new RPKI object, for the moment let's call it a VOA, would have to be defined and then it would have to be carried in the RPKI-Router Protocol [I-D.ietf-sidr-rpki-rtr].

If the value of the signing key, as identified by the Key Identifier, is to be rolled, in case of compromise or security policy, the technique in RFC 4808 [RFC4808] should be used.

While it is poor security practice to trust a different entity for your security/authentication/..., should a non-validating router choose to trust a validating router, they could use normal policy and signaling mechanisms, e.g. communities, to signal validation status. This page is too small to enumerate the vulnerabilities this creates.

## 7. Security Considerations

Signing (NLRI || Key Identifier) with the key of the NLRI-owner or some other pre-agreed key, only says that the contents were produced by the owner of the key (NLRI or other), and that no one in between has changed the (NLRI || Key Identifier). This is not protection against attacks, only configuration errors, aka 'fat fingers'. If we were trying to protect against an attacking PE replaying a signed (NLRI || Key Identifier) it has no business announcing, this design does not help.

If Key Identifier based keying is used, then the Key Identifier, and hence the signing key, MUST be unique to the VPN.

Adding a VOA which binds ( NLRI || Key Identifier ) still could be replayed from anywhere so really offers nothing. Like RPKI-based origin validation, this only catches fat fingers, not black hats.

## 8. IANA Considerations

This document requests the IANA create a new entry in the BGP Path Attributes Registry as follows:

Value	Code	Reference
-----	-----	-----
TBD	L3VPN Origination	This Document

## 9. Acknowledgements

The authors would like to thank Eric Rosen, John Scudder, Russ Housley, and Sandy Murphy.

We note the long expired draft draft-ietf-l3vpn-auth by Ron Bonica, Yakov Rekhter, Eric Rosen, Robert Raszuk, and Dan Tappan.

## 10. References

### 10.1. Normative References

- [I-D.ietf-sidr-bgpsec-algs]  
Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", Internet-Draft draft-ietf-sidr-bgpsec-algs-03, September 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.



- [RFC4271]    Rekhter, Y., Li, T. and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4364]    Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.
- [RFC4808]    Bellovin, S., "Key Change Strategies for TCP-MD5", RFC 4808, March 2007.
- [RFC6480]    Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

#### 10.2.    Informative References

- [I-D.ietf-sidr-pfx-validate]  
          Mohapatra, P., Scudder, J., Ward, D., Bush, R. and R. Austein, "BGP Prefix Origin Validation", Internet-Draft draft-ietf-sidr-pfx-validate-10, October 2012.
- [I-D.ietf-sidr-rpki-rtr]  
          Bush, R. and R. Austein, "The RPKI/Router Protocol", Internet-Draft draft-ietf-sidr-rpki-rtr-26, February 2012.
- [RFC6482]    Lepinski, M., Kent, S. and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

#### Authors' Addresses

Randy Bush  
Internet Initiative Japan  
5147 Crystal Springs  
Bainbridge Island, Washington 98110  
US

Email: randy@psg.com

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

Email: keyupate@cisco.com

Pranav Mehta  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134  
USA

Email: pmehta@cisco.com

Arjun Sreekantiah  
Cisco Systems  
170 W. Tasman Drive  
San Jose, CA 95134  
USA

Email: [asreekan@cisco.com](mailto:asreekan@cisco.com)

Luay Jalil  
Verizon  
1201 E Arapaho Rd.  
Richardson, TX 75081  
USA

Email: [luay.jalil@verizon.com](mailto:luay.jalil@verizon.com)