Secure Inter-Domain Routing
Internet-Draft
Intended status: Best Current Practice
Expires: January 4, 2015

D. Mandelberg
BBN Technologies

July 3, 2014

Simplified Local internet nUmber Resource Management with the RPKI
draft-dseomn-sidr-slurm-01

Abstract

The Resource Public Key Infrastructure (RPKI) is a global authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. Internet Service Providers (ISPs) can use the RPKI to validate BGP route origination assertions. Some ISPs locally use BGP with private address space or private AS numbers (see RFC6890). These local BGP routes cannot be verified by the global RPKI, and SHOULD be considered invalid based on the global RPKI (see RFC6491). The mechanisms described below provide ISPs with a way to make local assertions about private (reserved) INRs while using the RPKI's assertions about all other INRs.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Resource Public Key Infrastructure (RPKI) is a global authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. For example, the holder of a block of IP(v4 or v6) addresses can issue a Route Origination Authorization (ROA) [RFC6482] to authorize an Autonomous System (AS) to originate routes for that block.

Internet Service Providers (ISPs) can then use the RPKI to validate BGP routes. However, some ISPs locally use BGP with private address space ([RFC1918], [RFC4193], [RFC6598]) or private AS numbers ([RFC1930], [RFC6996]). These local BGP routes cannot be verified by the global RPKI, and SHOULD be considered invalid when using the RPKI. For example, [RFC6491] recommends the creation of ROAs that would invalidate routes for reserved and unallocated address space.

This document specifies two new mechanisms to enable ISPs to make local assertions about some INRs while using the RPKI's assertions about all other INRs. These mechanisms support the second and third use cases in [I-D.ietf-sidr-lta-use-cases]. The second use case describes use of [RFC1918] addresses or use of public address space not allocated to the ISP that is using it. The third use case

describes a situation in which an ISP publishes a variant of the RPKI hierarchy (for its customers). In this variant some prefixes and/or AS numbers are different from what the RPKI repository system presents to the general ISP population. The result is that routes for consumers of this variant hierarchy will be re-directed (via routing).

Both mechanisms are specified in terms of abstract sets of assertions. For Origin Validation [RFC6483], an assertion is a tuple of {IP prefix, prefix length, maximum length, AS number} as used by rpki-rtr [RFC6810]. Output Filtering, described in Section 2, filters out any assertions by the RPKI about locally reserved INRs. Locally Adding Assertions, described in Section 3, adds local assertions about locally reserved INRs. Note that both of these mechanisms can later be extended to cover any assertions made by the RPKI for use in BGPSEC [I-D.ietf-sidr-bgpsec-protocol].

In general, the primary output of an RPKI relying party is the data it sends to routers over the rpki-rtr protocol. The rpki-rtr protocol enables routers to query a relying party for all Origin Validation assertions it knows about (Reset Query) or for an update of only the changes in Origin Validation assertions (Serial Query). The mechanisms specified in this document are to be applied to the result set for a Reset Query, and to both the old and new sets that are compared for a Serial Query. Relying party software MAY modify other forms of output in comparable ways, but that is outside the scope of this document.

This document is intended to supersede [I-D.ietf-sidr-ltamgmt] while focusing only on local management of private INRs. Another draft [I-D.kent-sidr-suspenders] focuses on the other aspects of local management.

1.1. Terminology

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

2. Validation Output Filtering

To prevent the global RPKI from affecting routes with locally reserved INRs, a relying party is locally configured with a list of IP prefixes and/or AS numbers that are used locally, and taken from the reserved INR spaces. Any Origin Validation assertions where the IP prefix is equal to or subsumed by a locally reserved IP prefix, are removed from the relying party’s output. Any Origin Validation

assertions where the IP prefix contains a locally reserved IP prefix are removed and the relying party software SHOULD issue a warning.

3. Locally Adding Assertions

Each relying party is locally configured with a (possibly empty) list of Origin Validation assertions. This list is added to the relying party's output.

4. Configuring SLURM

Relying party software SHOULD support the following configuration format for Validation Output Filtering and Locally Adding Assertions. The format is defined using the Augmented Backus-Naur Form (ABNF) notation and core rules from [RFC5234] and the rules <IPv4address> and <IPv6address> from Appendix A of [RFC3986]. Each <del> command specifies an INR to use for Validation Output Filtering. Each <add> command specifies an assertion to use for Locally Adding Assertions. See Appendix A for an example SLURM file.

```
   SLURMFile = header *line

   header = %x53.4c.55.52.4d SP "1.0" CRLF ; "SLURM 1.0"

   line =  *WSP [comment] CRLF
         / *WSP command [ 1*WSP [comment] ] CRLF

   comment = "#" *(VCHAR / WSP)

   command = add / del

   add = %x61.64.64 1*WSP IPprefixMaxLen 1*WSP ASnum

   del = %x64.65.6c 1*WSP inr

   inr = IPprefix / ASnum

   IPprefix = IPv4prefix / IPv6prefix

   IPprefixMaxLen = IPv4prefixMaxLen / IPv6prefixMaxLen

   IPv4prefix = IPv4address "/" 1*2DIGIT

   IPv6prefix = IPv6address "/" 1*3DIGIT

   ; In the following two rules, if the maximum length component is
   ; missing, it is treated as equal to the prefix length.
   IPv4prefixMaxLen = IPv4prefix ["-" 1*2DIGIT]
   IPv6prefixMaxLen = IPv6prefix ["-" 1*3DIGIT]

   ASnum = 1*DIGIT
```

5. Combining Mechanisms

In the typical use case, a relying party uses both output filtering and locally added assertions. In this case, the resulting assertions MUST be the same as if output filtering were performed before locally adding assertions. I.e., locally added assertions MUST NOT be removed by output filtering.

If a relying party chooses to use both SLURM and Suspenders [I-D.kent-sidr-suspenders], the SLURM mechanisms MUST be performed on the output of Suspenders.

6. IANA Considerations

TBD

7. Security Considerations

The mechanisms described in this document provide an ISP additional control over its own network. Care should be taken in how that control is used.

8. Acknowledgements

The author would like to thank Stephen Kent for his guidance and detailed reviews of this document.

9. References

9.1. Informative References

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPSEC Protocol Specification", draft-ietf-sidr-bgpsec-protocol-08 (work in progress), November 2013.

[I-D.ietf-sidr-lta-use-cases]
Bush, R., "RPKI Local Trust Anchor Use Cases", draft-ietf-sidr-lta-use-cases-01 (work in progress), June 2014.

[I-D.ietf-sidr-ltamgmt]
Reynolds, M., Kent, S., and M. Lepinski, "Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08 (work in progress), April 2013.

[I-D.kent-sidr-suspenders]
Kent, S. and D. Mandelberg, "Suspenders: A Fail-safe Mechanism for the RPKI", draft-kent-sidr-suspenders-01 (work in progress), March 2014.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

[RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, February 2012.

[RFC6491] Manderson, T., Vegoda, L., and S. Kent, "Resource Public Key Infrastructure (RPKI) Objects Issued by IANA", RFC 6491, February 2012.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6810] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, January 2013.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, April 2013.

[RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, July 2013.

9.2. Normative References

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

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

Appendix A. Example SLURM File

```
   SLURM 1.0

   # Reserve 192.0.2.0/24 and 2001:DB8::/32 for local use.
   del 192.0.2.0/24
   del 2001:DB8::/32

   # Allow either 65536 or 65537 to originate routes to 192.0.2.0/24.
   add 192.0.2.0/24 65536
   add 192.0.2.0/24 65537

   add 2001:DB8::/48-52 65536 # 65536 originates 2001:DB8::/48 and
                              # sub-prefixes down to length 52.
   add 2001:DB8:0:42::/64 65537 # However, 65537 originates
                                # 2001:DB8:0:42::/64.
   add 2001:DB8:1::/48 65537 # 65537 also originates 2001:DB8:1::/48
```

Author's Address

David Mandelberg
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: david@mandelberg.org

Secure Inter-Domain Routing
Internet-Draft
Intended status: Best Current Practice
Expires: November 14, 2015

D. Mandelberg
BBN Technologies

May 13, 2015

Simplified Local internet nUmber Resource Management with the RPKI
draft-dseomn-sidr-slurm-02

Abstract

The Resource Public Key Infrastructure (RPKI) is a global authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. Network operators, e.g., Internet Service Providers (ISPs), can use the RPKI to validate BGP route origination assertions. In the future, ISPs also will be able to use the RPKI to validate the path of a BGP route. Some ISPs locally use BGP with private address space or private AS numbers (see RFC6890). These local BGP routes cannot be verified by the global RPKI, and SHOULD be considered invalid based on the global RPKI (see RFC6491). The mechanisms described below provide ISPs with a way to make local assertions about private (reserved) INRs while using the RPKI's assertions about all other INRs.

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 November 14, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Resource Public Key Infrastructure (RPKI) is a global authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. For example, the holder of a block of IP(v4 or v6) addresses can issue a Route Origination Authorization (ROA) [RFC6482] to authorize an Autonomous System (AS) to originate routes for that block.

Internet Service Providers (ISPs) can then use the RPKI to validate BGP routes. (Validation of the origin of a route is described in [RFC6483], and validation of the path of a route is described in [I-D.ietf-sidr-bgpsec-overview].) However, some ISPs locally use BGP with private address space ([RFC1918], [RFC4193], [RFC6598]) or private AS numbers ([RFC1930], [RFC6996]). These local BGP routes cannot be verified by the global RPKI, and SHOULD be considered invalid when using the RPKI. For example, [RFC6491] recommends the creation of ROAs that would invalidate routes for reserved and unallocated address space.

This document specifies two new mechanisms to enable ISPs to make local assertions about some INRs while using the RPKI's assertions about all other INRs. These mechanisms support the second and third use cases in [I-D.ietf-sidr-lta-use-cases]. The second use case describes use of [RFC1918] addresses or use of public address space not allocated to the ISP that is using it. The third use case describes a situation in which an ISP publishes a variant of the RPKI hierarchy (for its customers). In this variant some prefixes and/or AS numbers are different from what the RPKI repository system presents to the general ISP population. The result is that routes for consumers of this variant hierarchy will be re-directed (via routing).

Both mechanisms are specified in terms of abstract sets of assertions. For Origin Validation [RFC6483], an assertion is a tuple of {IP prefix, prefix length, maximum length, AS number} as used by rpki-rtr version 0 [RFC6810] and version 1 [I-D.ietf-sidr-rpki-rtr-rfc6810-bis]. For BGPsec [I-D.ietf-sidr-bgpsec-overview], an assertion is a tuple of {AS number, subject key identifier, router public key} as used by rpki-rtr version 1. Output Filtering, described in Section 2, filters out any assertions by the RPKI about locally reserved INRs. Locally Adding Assertions, described in Section 3, adds local assertions about locally reserved INRs. The combination of both mechanisms is described in Section 5.

To ensure local consistency, the effect of SLURM MUST be atomic. That is, the output of the relying party must be either the same as if SLURM were not used, or it must reflect the entire SLURM configuration. For an example of why this is required, consider the case of two local routes for the same prefix but different origin AS numbers. Both routes are configured with Locally Adding Assertions. If neither addition occurs, then both routes could be in the unknown state [RFC6483]. If both additions occur then both routes would be in the valid state. However, if one addition occurs and the other does not, then one could be invalid while the other is valid.

In general, the primary output of an RPKI relying party is the data it sends to routers over the rpki-rtr protocol. The rpki-rtr protocol enables routers to query a relying party for all assertions it knows about (Reset Query) or for an update of only the changes in assertions (Serial Query). The mechanisms specified in this document are to be applied to the result set for a Reset Query, and to both the old and new sets that are compared for a Serial Query. Relying party software MAY modify other forms of output in comparable ways, but that is outside the scope of this document.

This document is intended to supersede [I-D.ietf-sidr-ltamgmt] while focusing only on local management of private INRs. Another draft [I-D.kent-sidr-suspenders] focuses on the other aspects of local management.

1.1. Terminology

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

2. Validation Output Filtering

To prevent the global RPKI from affecting routes with locally reserved INRs, a relying party may be locally configured with a list of IP prefixes and/or AS numbers that are used locally, and taken from reserved INR spaces. Any Origin Validation assertions where the IP prefix is equal to or subsumed by a locally reserved IP prefix, are removed from the relying party's output. Any Origin Validation assertions where the IP prefix contains a locally reserved IP prefix are removed; the relying party software SHOULD issue a warning when this action is taken. (Note that an Origin Validation assertion is not removed due to its AS number matching a locally reserved AS number.) Any BGPsec assertion where the AS number is equal to a locally reserved AS number is removed from the relying party's output.

3. Locally Adding Assertions

Each relying party is locally configured with a (possibly empty) list of assertions. This list is added to the relying party's output.

4. Configuring SLURM

Relying party software SHOULD support the following configuration format for Validation Output Filtering and Locally Adding Assertions. The format is defined using the Augmented Backus-Naur Form (ABNF) notation and core rules from [RFC5234] and the rules <IPv4address> and <IPv6address> from Appendix A of [RFC3986]. See Appendix A for an example SLURM file.

A SLURM configuration file, <SLURMFile>, consists of a head and a body. The head identifies the file as a SLURM configuration file, specifies the version of SLURM for which the file was written, and optionally contains other information described below. The body contains the configuration for Validation Output Filtering and Locally Adding Assertions.

```
SLURMFile = head body

head = firstLine *(commentLine / headLine)

body = *(commentLine / bodyLine)

firstLine = %x53.4c.55.52.4d SP "1.0" EOL ; "SLURM 1.0"

commentLine = *WSP [comment] EOL

headLine = *WSP headCommand [ 1*WSP [comment] ] EOL

bodyLine = *WSP bodyCommand [ 1*WSP [comment] ] EOL

comment = "#" *(VCHAR / WSP)

EOL = CRLF / LF
```

The head may specify a target. If present, the target string identifies the environment in which the SLURM file is intended to be used. The meaning of the target string, if any, is determined by the user. If a target is present, a relying party SHOULD verify that that the target is an acceptable value, and reject the SLURM file if the target is not acceptable. For example, the relying party could be configured to accept SLURM files only if they do not specify a target, have a target value of "hostname=rpki.example.com", or have a target value of "as=65536". If more than one target line is present, all targets must be acceptable to the RP.

```
headCommand = target

target =
   %x74.61.72.67.65.74 1*WSP ; "target"
   1*VCHAR
```

The body contains zero or more configuration lines for Validation Output Filtering and Locally Adding Assertions. Each <del> command specifies an INR to use for Validation Output Filtering. Each <add> command specifies an assertion to use for Locally Adding Assertions.

```
bodyCommand = add / del

add =
   %x61.64.64 1*WSP ; "add"
   addItem

del =
   %x64.65.6c 1*WSP ; "del"
```

```
      delItem

   addItem = addItemPrefixAS / addItemASKey

   ; Add a mapping from a prefix and max length to an AS number.
   addItemPrefixAS =
      %x6f.72.69.67.69.6e.61.74.69.6f.6e 1*WSP ; "origination"
      IPprefixMaxLen 1*WSP
      ASnum

   ; Add a mapping from an AS number to a router public key.
   addItemASKey =
      %x62.67.70.73.65.63 1*WSP ; "bgpsec"
      ASnum 1*WSP
      RouterSKI 1*WSP
      RouterPubKey

   delItem = delItemPrefix / delItemAS

   ; Filter prefix-AS mappings, using the given prefix
   delItemPrefix =
      %x6f.72.69.67.69.6e.61.74.69.6f.6e 1*WSP ; "origination"
      IPprefix

   ; Filter AS-key mappings for the given AS
   delItemAS =
      %x62.67.70.73.65.63 1*WSP ; "bgpsec"
      ASnum

   IPprefix = IPv4prefix / IPv6prefix

   IPprefixMaxLen = IPv4prefixMaxLen / IPv6prefixMaxLen

   IPv4prefix = IPv4address "/" 1*2DIGIT
   IPv6prefix = IPv6address "/" 1*3DIGIT

   ; In the following two rules, if the maximum length component is
   ; missing, it is treated as equal to the prefix length.
   IPv4prefixMaxLen = IPv4prefix ["-" 1*2DIGIT]
   IPv6prefixMaxLen = IPv6prefix ["-" 1*3DIGIT]

   ASnum = 1*DIGIT

   ; This is the Base64 [RFC4648] encoding of a router certificate's
   ; Subject Key Identifer, as described in
   ; [I-D.ietf-sidr-bgpsec-pki-profiles] and [RFC6487]. This is the
   ; value of the ASN.1 OCTET STRING without the ASN.1 tag or length
   ; fields.
```

```
RouterSKI = Base64

; This is the Base64 [RFC4648] encoding of a router public key's
; subjectPublicKeyInfo value, as described in
; [I-D.ietf-sidr-bgpsec-algs]. This is the full ASN.1 DER encoding
; of the subjectPublicKeyInfo, including the ASN.1 tag and length
; values of the subjectPublicKeyInfo SEQUENCE.
RouterPubKey = Base64

Base64 = 1*(ALPHA / DIGIT / "+" / "/") 0*2"="
```

An implementation MAY support the concurrent use of multiple SLURM files. In this case, the resulting inputs to Validation Output Filtering and Locally Adding Assertions are the respective unions of the inputs from each file. The typical use case for multiple files is when the files have distinct scopes. For example, an organization may belong to two separate networks that use different private-use IP prefixes and AS numbers. To detect conflict between multiple SLURM files, a relying party SHOULD issue a warning in the following cases:

1. There may be conflicting changes to Origin Validation assertions if there exists an IP address X and distinct SLURM files Y,Z such that X is contained by any prefix in any <addItemPrefixAS> or <delItemPrefix> in file Y and X is contained by any prefix in any <addItemPrefixAS> or <delItemPrefix> in file Z.

2. There may be conflicting changes to BGPsec assertions if there exists an AS number X and distinct SLURM files Y,Z such that X is used in any <addItemASKey> or <delItemAS> in file Y and X is used in any <addItemASKey> or <delItemAS> in file Z.

5. Combining Mechanisms

In the typical use case, a relying party uses both output filtering and locally added assertions. In this case, the resulting assertions MUST be the same as if output filtering were performed before locally adding assertions. I.e., locally added assertions MUST NOT be removed by output filtering.

If a relying party chooses to use both SLURM and Suspenders [I-D.kent-sidr-suspenders], the SLURM mechanisms MUST be performed on the output of Suspenders.

6. IANA Considerations

TBD

7. Security Considerations

The mechanisms described in this document provide a network operator with additional ways to control its own network while making use of RPKI data. These mechanisms are applied only locally; they do not influence how other network operators interpret RPKI data. Nonetheless, care should be taken in how these mechanisms are employed.

8. Acknowledgements

The author would like to thank Stephen Kent for his guidance and detailed reviews of this document. Thanks go to Wesley Wang for the idea behind the target command, to Declan Ma for the idea behind use of multiple SLURM files, and to Richard Hansen for his careful reviews.

9. References

9.1. Informative References

[I-D.ietf-sidr-bgpsec-overview]
Lepinski, M. and S. Turner, "An Overview of BGPsec", draft-ietf-sidr-bgpsec-overview-06 (work in progress), January 2015.

[I-D.ietf-sidr-lta-use-cases]
Bush, R., "RPKI Local Trust Anchor Use Cases", draft-ietf-sidr-lta-use-cases-02 (work in progress), December 2014.

[I-D.ietf-sidr-ltamgmt]
Reynolds, M., Kent, S., and M. Lepinski, "Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08 (work in progress), April 2013.

[I-D.ietf-sidr-rpki-rtr-rfc6810-bis]
Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", draft-ietf-sidr-rpki-rtr-rfc6810-bis-03 (work in progress), March 2015.

[I-D.kent-sidr-suspenders]
Kent, S. and D. Mandelberg, "Suspenders: A Fail-safe Mechanism for the RPKI", draft-kent-sidr-suspenders-03 (work in progress), April 2015.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

[RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, February 2012.

[RFC6491] Manderson, T., Vegoda, L., and S. Kent, "Resource Public Key Infrastructure (RPKI) Objects Issued by IANA", RFC 6491, February 2012.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6810] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, January 2013.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, April 2013.

[RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, July 2013.

9.2. Normative References

[I-D.ietf-sidr-bgpsec-algs]
Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", draft-ietf-sidr-bgpsec-algs-09 (work in progress), January 2015.

[I-D.ietf-sidr-bgpsec-pki-profiles]
Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPSEC Router Certificates, Certificate Revocation Lists, and Certification Requests", draft-ietf-sidr-bgpsec-pki-profiles-10 (work in progress), January 2015.

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

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

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

Appendix A. Example SLURM File

```
SLURM 1.0

# This file is only intended to be used on a relying party running
# on rpki.example.com.
target hostname=rpki.example.com # this is a comment

# Reserve IP prefixes for local use.
del origination 10.0.0.0/24
del origination fd0b:dd1d:2dcc::/48

# Reserve AS numbers for local use.
del bgpsec 64512
del bgpsec 64513

# Allow either 64512 or 64513 to originate routes to 10.0.0.0/24.
add origination 10.0.0.0/24 64512
add origination 10.0.0.0/24 64513

# 64512 originates fd0b:dd1d:2dcc::/52 and sub-prefixes up to length
# 56.
add origination fd0b:dd1d:2dcc::/52-56 64512

# However, 64513 originates fd0b:dd1d:2dcc:42::/64.
add origination fd0b:dd1d:2dcc:42::/64 64513

# 64513 also originates fd0b:dd1d:2dcc:100::/52
add origination fd0b:dd1d:2dcc:100::/52 64513

# Authorize router keys to sign BGPsec paths on behalf of the
# specified ASes. Note that the Base64 strings used in this
# example are not valid SKIs or router public keys, due to line
# length restrictions in RFCs.
add bgpsec 64512 Zm9v VGhpcyBpcyBub3QgYSByb3V0ZXIgcHVibGljIGtleQ==
add bgpsec 64512 YmFy b3IgYSBmbG9jayBvZiBkdWNrcw==
add bgpsec 64513 YWJj bWF5YmUgYSBkaWZmZXJlbnQgYXZpYW4gY2Fycmllcj8=
```

Author’s Address

David Mandelberg
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: david@mandelberg.org

Internet Engineering Task Force
Internet-Draft
Updates: 4271 (if approved)
Intended status: Standards Track
Expires: January 7, 2016

W. George
Time Warner Cable
S. Amante
Apple, Inc.

July 6, 2015

Autonomous System Migration Mechanisms and Their Effects on the BGP AS_PATH Attribute
draft-ietf-idr-as-migration-06

Abstract

This draft discusses some existing commonly-used BGP mechanisms for ASN migration that are not formally part of the BGP4 protocol specification. It is necessary to document these de facto standards to ensure that they are properly supported in future BGP protocol work.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Table of Contents

1. Introduction

This draft discusses some existing commonly-used BGP mechanisms for Autonomous System Number (ASN) migration that are not formally part of the BGP4 [RFC4271] protocol specification. These mechanisms are local to a given BGP Speaker and do not require negotiation with or cooperation of BGP neighbors. The deployment of these mechanisms do not need to interwork with one another to accomplish the desired results, so slight variations between existing vendor implementations exist, and will not necessarily be harmonized due to this document. However, it is necessary to document these de facto standards to ensure that new implementations can be successful, and any future protocol enhancements to BGP that propose to read, copy, manipulate or compare the AS_PATH attribute can do so without inhibiting the use of these very widely used ASN migration mechanisms.

The migration mechanisms discussed here are useful to ISPs and organizations of all sizes, but it is important to understand the business need for these mechanisms and illustrate why they are so critical for ISPs' operations. During a merger, acquisition or divestiture involving two organizations it is necessary to seamlessly migrate both internal and external BGP speakers from one ASN to a

second ASN. The overall goal in doing so is to simplify operations through consistent configurations across all BGP speakers in the combined network. In addition, given that the BGP Path Selection algorithm selects routes with the shortest AS_PATH attribute, it is critical that the ISP does not increase AS_PATH length during or after ASN migration, because an increased AS_PATH length would likely result in sudden, undesirable changes in traffic patterns in the network.

By default, the BGP protocol requires an operator to configure a router to use a single remote ASN for the BGP neighbor, and the ASN must match on both ends of the peering in order to successfully negotiate and establish a BGP session. Prior to the existence of these migration mechanisms, it would have required an ISP to coordinate an ASN change with, in some cases, tens of thousands of customers. In particular, as each router is migrated to the new ASN, to avoid an outage due to ASN mismatch, the ISP would have to force all customers on that router to change their router configurations to use the new ASN immediately after the ASN change. Thus, it becomes critical to allow the ISP to make this process a bit more asymmetric, so that it could seamlessly migrate the ASN within its network(s), but allow the customers to gradually migrate to the ISP's new ASN at their leisure, either by coordinating individual reconfigurations, or accepting sessions using either the old or new ASN to allow for truly asymmetric migration.

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].

1.2. Documentation note

This draft uses Autonomous System Numbers (ASNs) from the range reserved for documentation as described in RFC 5398 [RFC5398]. In the examples used here, they are intended to represent Globally Unique ASNs, not private use ASNs as documented in RFC 6996 [RFC6996] section 5.

2. ASN Migration Scenario Overview

The use case being discussed here is an ISP merging two or more ASNs, where eventually one ASN subsumes the other(s). In this use case, we will assume the most common case where there are two ISPs, A and B, that prior to the ASN migration use AS 64500 and 64510, respectively. AS 64500 will be the permanently retained ASN used across the consolidated set of both ISPs network equipment, and AS 64510 will be

retired. Thus, at the conclusion of the ASN migration, there will be a single ISP A' with all internal BGP speakers configured to use AS 64500. To all external BGP speakers, the AS_PATH length will not be increased.

In this same scenario, AS 64496 and AS 64499 represent two separate customer networks: C and D, respectively. Originally, customer C (AS 64496) is attached to ISP B, which will undergo ASN migration from AS 64510 to AS 64500. Furthermore, customer D (AS 64499) is attached to ISP A, which does not undergo ASN migration since the ASN for ISP A will remain constant, (AS 64500). Although this example refers to AS 64496 and 64499 as customer networks, either or both may be settlement-free or other types of peers. In this use case they are referred to as "customers" merely for convenience.

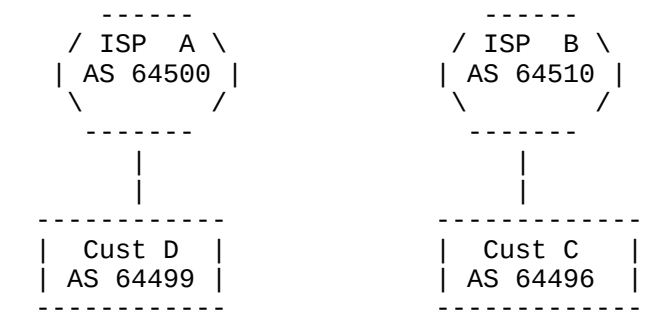


Figure 1: Before Migration

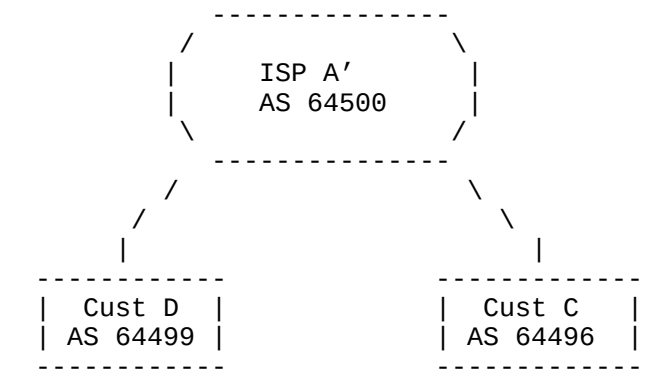


Figure 2: After Migration

The general order of operations, typically carried out in a single maintenance window by the network undergoing ASN migration (ISP B), are as follows. First, ISP B will change the global BGP ASN used by

a Provider Edge (PE) router, from ASN 64510 to 64500. At this point, the router will no longer be able to establish eBGP sessions toward the existing Customer Edge (CE) devices that are attached to it and still using AS 64510. Second, since ISP B needs to do this without coordinating the simultaneous change of its ASN with all of its eBGP peers, ISP B will configure two separate, but related ASN migration mechanisms discussed in this document on all eBGP sessions toward all CE devices. These mechanisms enable the router to establish BGP neighbors using the legacy ASN, modify the AS_PATH attribute received from a CE device when advertising it further, and modify AS_PATH when transmitted toward CE devices to achieve the desired effect of not increasing the length of the AS_PATH.

At the conclusion of the ASN migration, the CE devices at the edge of the network are not aware of the fact that their upstream router is now in a new ASN and do not observe any change in the length of the AS_PATH attribute. However, after the changes discussed in this document are put in place by ISP A', there is a change to the contents of the AS_PATH attribute to ensure the AS_PATH is not artificially lengthened while these AS migration parameters are used.

In this use case, neither ISP is using BGP Confederations RFC 5065 [RFC5065] internally.

3. External BGP Autonomous System Migration Mechanisms

The following section addresses optional capabilities that are specific to modifying the AS_PATH attribute at the Autonomous System Border Routers (ASBRs) of an organization, (typically a single Service Provider). This ensures that external BGP customers/peers are not forced to make any configuration changes on their CE routers before or during the exact time the Service Provider wishes to migrate to a new, permanently retained ASN. Furthermore, these mechanisms eliminate the artificial lengthening of the AS_PATH both transmitted from and received by the Service Provider that is undergoing AS Migration, which would have negative implications on path selection by external networks.

3.1. Modify Inbound BGP AS_PATH Attribute

The first instrument used in the process described above is called "Local AS". This allows the router to supersede the globally configured ASN in the "My Autonomous System" field of the BGP OPEN [RFC4271] with a locally defined AS value for a specific BGP neighbor or group of neighbors. This mechanism allows the PE router that was formerly in ISP B to establish an eBGP session toward the existing CE devices using the legacy AS, AS 64510. Ultimately, the CE devices (i.e.: customer C) are completely unaware that ISP B has reconfigured

its router to participate as a member of a new AS. Within the context of the former ISP B PE router, the second effect this specific mechanism has on AS_PATH is that, by default, it prepends all received BGP UPDATEs with the legacy AS of ISP B: AS 64510, while advertising it (Adj-RIB-Out) to other BGP speakers (A'). Within the Loc-RIB on ISP B prior to the migration, the AS_PATH of route announcements received from customer C would appear as: 64496, whereas the same RIB on ISP A' (ISP B routers post-migration) would contain AS_PATH: 64510 64496.

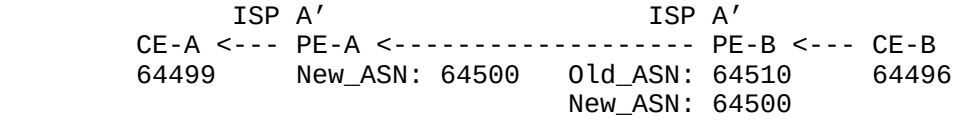
A second instrument, referred to as "No Prepend Inbound", is enabled on PE routers migrating from ISP B. The "No Prepend Inbound" capability causes ISP B's routers to not prepend the legacy AS, AS 64510, when advertising UPDATES received from customer C. This restores the AS_PATH within ISP A' for route announcements received from customer C so that it is just one ASN in length: 64496.

In the direction of CE -> PE (inbound):

1. "Local AS": Allows the local BGP router to generate a BGP OPEN to an eBGP neighbor with the old, legacy ASN value in the "My Autonomous System" field. When this capability is activated, it also causes the local router to prepend the <old_ASN> value to the AS_PATH when installing or advertising routes received from a CE to iBGP neighbors inside the Autonomous System.

2. "No Prepend Inbound (of Local AS)": the local BGP router does not prepend <old_ASN> value to the AS_PATH when installing or advertising routes received from the CE to iBGP neighbors inside the Autonomous System

PE-B is a PE that was originally in ISP B, and has a customer eBGP session to CE-B. PE-B has had its global configuration ASN changed from AS 64510 to AS 64500 to make it part of the permanently retained ASN. This now makes PE-B a member of ISP A'. PE-A is a PE that was originally in ISP A, and has a customer peer CE-A. Although its global configuration ASN remains AS 64500, throughout this exercise we also consider PE-A a member of ISP A'.



Note: Direction of BGP UPDATE as per the arrows.

Figure 3: Local AS and No Prepend BGP UPDATE Diagram

As a result using both the "Local AS" and "No Prepend Inbound" capabilities on PE-B, CE-A will see an AS_PATH of: 64500 64496. CE-A will not receive a BGP UPDATE containing AS 64510 in the AS_PATH. (If only the "Local AS" mechanism was configured without "No Prepend Inbound" on PE-B, then CE-A would have seen an AS_PATH of: 64500 64510 64496, which results in an unacceptable lengthening of the AS_PATH). NOTE: If there are still routers in the old ASN (64510), it is possible for them to accept these manipulated routes (i.e. those with 64510 removed from the AS_PATH by this command) as if they have not already passed through their ASN, potentially causing a loop, since BGP's normal loop-prevention behavior of rejecting routes that include its ASN in the path will not catch these. Careful filtering between routers remaining in the old ASN and routers migrated to the new ASN is necessary to minimize the risk of routing loops.

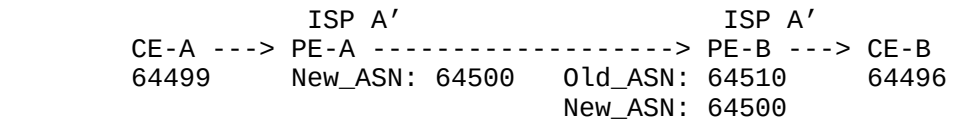
3.2. Modify Outbound BGP AS_PATH Attribute

The two aforementioned mechanisms, "Local AS" and "No Prepend Inbound", only modify the AS_PATH Attribute received by the ISP's PE's in the course of processing BGP UPDATEs from CE devices when CE devices still have an eBGP session established with the ISPs legacy AS (AS64510).

In some existing implementations, "Local AS" and "No Prepend Inbound" does not concurrently modify the AS_PATH Attribute for BGP UPDATEs that are transmitted by the ISP's PE's to CE devices. In these implementations, with "Local AS" and "No Prepend Inbound" used on PE-B, it automatically causes a lengthening of the AS_PATH in outbound BGP UPDATEs from ISP A' toward directly attached eBGP speakers, (Customer C in AS 64496). The externally observed result is that customer C, in AS 64496, will receive the following AS_PATH: 64510 64500 64499. Therefore, if ISP A' takes no further action, it will cause an unacceptable increase in AS_PATH length within customer's networks directly attached to ISP A'.

A tertiary mechanism, referred to as "Replace Old AS", is used to resolve this problem. This capability allows ISP A' to prevent routers from appending the globally configured ASN in outbound BGP UPDATEs toward directly attached eBGP neighbors that are using the "Local AS" mechanism. Instead, only the old (or previously used) AS will be prepended in the outbound BGP UPDATE toward the customer's network, restoring the AS_PATH length to what it what was before AS Migration occurred.

To re-use the above diagram, but in the opposite direction, we have:



Note: Direction of BGP UPDATE as per the arrows.

Figure 4: Replace AS BGP UPDATE Diagram

By default, without the use of "Replace Old AS", CE-B would see an AS_PATH of: 64510 64500 64499. After ISP A' changes PE-B to use "Replace Old AS", CE-B would receive an AS_PATH of: 64510 64499, which is the same AS_PATH length pre-AS migration.

3.3. Implementation

The mechanisms introduced in this section MUST be configurable on a per-neighbor or per neighbor group (i.e. a group of similar BGP neighbor statements that reuse some common configuration to simplify provisioning) basis to allow for maximum flexibility. When the "Local AS" capability is used, a local ASN will be provided in the configuration that is different from the globally-configured ASN of the BGP router. To implement this mechanism, a BGP speaker SHOULD send BGP OPEN [RFC4271] (see section 4.2) messages to the configured eBGP peer(s) using the local ASN configured for this session as the value sent in "My Autonomous System". The BGP router SHOULD NOT use the ASN configured globally within the BGP process as the value sent in "My Autonomous System" in the OPEN message. This will avoid causing the eBGP neighbor to unnecessarily generate a BGP OPEN Error message "Bad Peer AS". This method is typically used to re-establish eBGP sessions with peers expecting the legacy ASN after a router has been moved to a new ASN.

Implementations MAY support a more flexible model where the eBGP speaker attempts to open the BGP session using either the ASN configured as "Local AS" or the globally configured AS as discussed in BGP Alias (Section 4.2). If the session is successfully established to the globally configured ASN, then the modifications to AS_PATH described in this document SHOULD NOT be performed, as they are unnecessary. The benefit to this more flexible model is that it allows the remote neighbor to reconfigure to the new ASN without direct coordination between the ISP and the customer.

Note that this procedure will vary slightly if the locally or globally configured ASN is a 4-octet ASN. See section 3 of [RFC4893].

When the BGP router receives UPDATEs from its eBGP neighbor configured with the "Local AS" mechanism, it processes the UPDATE as described in RFC4271 section 5.1.2 [RFC4271]. However the presence of a second ASN due to "Local AS" adds the following behavior to processing UPDATEs received from an eBGP neighbor configured with this mechanism:

1. Internal: the router SHOULD append the configured "Local AS" ASN in the AS_PATH attribute before installing the route or advertising the UPDATE to an iBGP neighbor. The decision of when to append the ASN is an implementation detail outside the scope of this document. Some considerations factoring into this decision include consistency in the AS_PATH throughout the AS, and implementation of the loop detection mechanism.

2. External: the BGP router SHOULD first append the globally configured ASN to the AS_PATH immediately followed by the "Local AS" value before advertising the UPDATE to an eBGP neighbor.

Two options exist to manipulate the behavior of the basic "Local AS" mechanism. They modify the behavior as described below:

1. "No Prepend Inbound" - When the BGP router receives inbound BGP UPDATEs from its eBGP neighbor configured with this option, it MUST NOT append the "Local AS" ASN value in the AS_PATH attribute when installing the route or advertising that UPDATE to iBGP neighbors, but it MUST still append the globally configured ASN as normal when advertising the UPDATE to other local eBGP neighbors (i.e. those natively peering with the globally configured ASN).

2. "Replace Old AS", (outbound) - When the BGP router generates outbound BGP UPDATEs toward an eBGP neighbor configured with this option, the BGP speaker MUST NOT append the globally configured ASN from the AS_PATH attribute. The BGP router MUST append only the configured "Local AS" ASN value to the AS_PATH attribute before sending the BGP UPDATEs outbound to the eBGP neighbor.

4. Internal BGP Autonomous System Migration Mechanisms

The following section describes mechanisms that assist with a gradual and least service impacting migration of Internal BGP sessions from a legacy ASN to the permanently retained ASN. The following mechanism is very valuable to networks undergoing AS migration, but its use does not cause changes to the AS_PATH attribute.

4.1. Internal BGP AS Migration

In this case, all of the routers to be consolidated into a single, permanently retained ASN are under the administrative control of a single entity. Unfortunately, the traditional method of migrating all Internal BGP speakers, particularly within larger networks, is both time consuming and widely service impacting.

The traditional method to migrate Internal BGP sessions was strictly limited to reconfiguration of the global configuration ASN and, concurrently, changing all iBGP neighbors' remote ASN from the legacy ASN to the new, permanently retained ASN on each router within the legacy AS. These changes can be challenging to swiftly execute in networks with with more than a few dozen internal BGP routers. There is also the concomitant service interruptions as these changes are made to routers within the network, resulting in a reset of iBGP sessions and subsequent route reconvergence to reestablish optimal routing paths. Operators often cannot make such sweeping changes given the associated risks of a highly visible service interruption; rather, they require a more gradual method to migrate Internal BGP sessions, from one ASN to a second, permanently retained ASN, that is not visibly service-impacting to its customers.

With the "Internal BGP AS Migration" mechanism described herein, it allows an Internal BGP speaker to form a single iBGP session using either the old, legacy ASN or the new, permanently retained ASN. The benefits of using this mechanism are several fold. First, it allows for a more gradual and less service-impacting migration away from the legacy ASN to the permanently retained ASN. Second, it (temporarily) permits the coexistence of the legacy and permanently retained ASN within a single network, allowing for uniform BGP path selection among all routers within the consolidated network.

The iBGP router with the "Internal BGP AS Migration" capability enabled allows the receipt of a BGP OPEN message with either the legacy ASN value or the new, globally configured ASN value in the "My Autonomous System" field of the BGP OPEN message from iBGP neighbors. It is important to recognize that enablement of the "Internal BGP AS Migration" mechanism preserves the semantics of a regular iBGP session (i.e. using identical ASNs). Thus, the BGP attributes transmitted by and the acceptable methods of operation on BGP attributes received from iBGP sessions configured with "Internal BGP AS Migration" capability are no different than those exchanged across an iBGP session without "Internal BGP AS Migration" configured, as defined by [RFC4271] and [RFC4456].

Typically, in medium to large networks, BGP Route Reflectors [RFC4456] (RRs) are used to aid in reduction of configuration of iBGP

sessions and scalability with respect to overall TCP (and, BGP) session maintenance between adjacent iBGP routers. Furthermore, BGP Route Reflectors are typically deployed in pairs within a single Route Reflection cluster to ensure high reliability of the BGP Control Plane. As such, the following example will use Route Reflectors to aid in understanding the use of the "Internal BGP AS Migration" mechanism. Note that Route Reflectors are not a prerequisite to enable "Internal BGP AS Migration" and this mechanism can be enabled independent of the use of Route Reflectors.

The general order of operations is as follows:

1. Within the legacy network, (the routers comprising the set of devices that still have a globally configured legacy ASN), one member of a redundant pair of RRs has its global configuration ASN changed to the permanently retained ASN. Concurrently, the "Internal BGP AS Migration" capability is enabled on all iBGP sessions on that device. This will comprise Non-Client iBGP sessions to other RRs as well as Client iBGP sessions, typically to PE devices, both still utilizing the legacy ASN. Note that during this step there will be a reset and reconvergence event on all iBGP sessions on the RRs whose configuration was modified; however, this should not be service impacting due to the use of redundant RRs in each RR Cluster.

2. The above step is repeated for the other side of the redundant pair of RRs. The one alteration to the above procedure is that the "Internal BGP AS Migration" mechanism is now removed from the Non-Client iBGP sessions toward the other (previously reconfigured) RRs, since it is no longer needed. The "Internal BGP AS Migration" mechanism is still required on all RRs for all RR Client iBGP sessions. Also during this step, there will be a reset and reconvergence event on all iBGP sessions whose configuration was modified, but this should not be service impacting. At the conclusion of this step, all RRs should now have their globally configured ASN set to the permanently retained ASN and "Internal BGP AS Migration" enabled and in use toward RR Clients.

3. At this point, the network administrators would then be able to establish iBGP sessions between all Route Reflectors in both the legacy and permanently retained networks. This would allow the network to appear to function, both internally and externally, as a single, consolidated network using the permanently retained network.

4. To complete the AS migration, each RR Client (PE) in the legacy network still utilizing the legacy ASN is now modified.

Specifically, each legacy PE would have its globally configured ASN changed to use the permanently retained ASN. The ASN configured within the PE for the iBGP sessions toward each RR would be changed to use the permanently retained ASN. It is unnecessary to enable "Internal BGP AS Migration" mechanism on these migrated iBGP sessions. During the same maintenance window, External BGP sessions would be modified to include the above "Local AS", "No Prepend" and "Replace Old AS" mechanisms described in Section 3 above, since all of the changes are service interrupting to the eBGP sessions of the PE. At this point, all PEs will have been migrated to the permanently retained ASN.

5. The final step is to excise the "Internal BGP AS Migration" configuration from the Router Reflectors in an orderly fashion. After this is complete, all routers in the network will be using the new, permanently retained ASN for all iBGP sessions with no vestiges of the legacy ASN on any iBGP sessions.

The benefit of using the aforementioned "Internal BGP AS Migration" capability is that it is a more gradual and less externally service-impacting change to accomplish an AS migration. Previously, without "Internal BGP AS Migration", such an AS migration change would carry a high risk and need to be successfully accomplished in a very short timeframe (e.g.: at most several hours). In addition, it would likely cause substantial routing churn and rapid fluctuations in traffic carried -- potentially causing periods of congestion and resultant packet loss -- during the period the configuration changes are underway to complete the AS Migration. On the other hand, with "Internal BGP AS Migration", the migration from the legacy ASN to the permanently retained ASN can occur over a period of days or weeks with reduced customer disruption. (The only observable service disruption should be when each PE undergoes the changes discussed in step 4 above.)

4.2. Implementation

The mechanism introduced in this section MUST be configurable on a per-neighbor or per neighbor group basis to allow for maximum flexibility. When configured with this mechanism, a BGP speaker MUST accept BGP OPEN and establish an iBGP session from configured iBGP peers if the ASN value in "My Autonomous System" is either the globally configured ASN or a locally configured ASN provided when this capability is utilized. Additionally, a BGP router configured with this mechanism MUST send its own BGP OPEN [RFC4271] (see section 4.2) using either the globally configured or the locally configured ASN in "My Autonomous System" as follows. To avoid potential deadlocks when two BGP speakers are attempting to establish a BGP

peering session and are both configured with this mechanism, the speaker SHOULD send BGP OPEN using the globally configured ASN first, and only send a BGP OPEN using the locally configured ASN as a fallback if the remote neighbor responds with the BGP error "Bad Peer AS". In each case, the BGP speaker MUST treat UPDATEs sent and received to this peer as if this was a natively configured iBGP session, as defined by [RFC4271] and [RFC4456].

Note that this procedure will vary slightly if the locally or globally configured ASN is a 4-octet ASN. See section 3 of [RFC4893].

5. Additional Operational Considerations

This document describes several mechanisms to support ISPs and other organizations that need to perform ASN migrations. Other variations of these mechanisms may exist, for example, in legacy router software that has not been upgraded or reached End of Life, but continues to operate in the network. Such variations are beyond the scope of this document.

Companies routinely go through periods of mergers, acquisitions and divestitures, which in the case of the former cause them to accumulate several legacy ASNs over time. ISPs often do not have control over the configuration of customers' devices (i.e.: the ISPs are often not providing a managed CE router service, particularly to medium and large customers that require eBGP). Furthermore, ISPs are using methods to perform ASN migration that do not require coordination with customers. Ultimately, this means there is not a finite period of time after which legacy ASNs will be completely expunged from the ISP's network. In fact, it is common that legacy ASNs and the associated External BGP AS Migration mechanisms discussed in this document can and do persist for several years, if not longer. Thus, it is prudent to plan that legacy ASNs and associated External BGP AS Migration mechanisms will persist in a operational network indefinitely.

With respect to the Internal BGP AS Migration mechanism, all of the routers to be consolidated into a single, permanently retained ASN are under the administrative control of a single entity. Thus, completing the migration from iBGP sessions using the legacy ASN to the permanently retained ASN is more straightforward and could be accomplished in a matter of days to months. Finally, good operational hygiene would dictate that it is good practice to avoid using "Internal BGP AS Migration" capability over a long period of time for reasons of not only operational simplicity of the network, but also reduced reliance on that mechanism during the ongoing

lifecycle management of software, features and configurations that are maintained on the network.

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. This involves manipulating the AS_PATH Attribute with the intent of not increasing the AS_PATH length, which would typically cause the BGP route to no longer be selected by BGP's Path Selection Algorithm in others' networks. This could result in sudden and unexpected shifts in traffic patterns in the network, potentially resulting in congestion.

Given that these mechanisms can only be enabled through configuration of routers within a single network, standard security measures should be taken to restrict access to the management interface(s) of routers that implement these mechanisms. Additionally, BGP sessions SHOULD be protected using TCP Authentication Option [RFC5925] and the Generalized TTL Security Mechanism [RFC5082]

8. Acknowledgements

Thanks to Kotikalapudi Sriram, Stephane Litkowski, Terry Manderson, David Farmer, Jaroslaw Adam Gralak, Gunter Van de Velde, Juan Alcaide, Jon Mitchell, Thomas Morin, Alia Atlas, Alvaro Retana, and John Scudder for their comments.

9. References

9.1. Normative References

[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.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, April 2006.

9.2. Informative References

[ALU] Alcatel-Lucent, "BGP Local AS attribute", 2006-2012, <https://infoproducts.alcatel-lucent.com/html/0_add-h-f/93-0074-10-01/7750_SR_OS_Routing_Protocols_Guide/BGP-CLI.html#709567>.

[CISCO] Cisco Systems, Inc., "BGP Support for Dual AS Configuration for Network AS Migrations", 2003, <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-3s/asr1000/irg-xe-3s-asr1000-book/irg-dual-as.html>.

[JUNIPER] Juniper Networks, Inc., "Configuring the BGP Local Autonomous System Attribute", 2012, <http://www.juniper.net/techpubs/en_US/junos13.3/topics/concept/bgp-local-as-introduction.html>.

[RFC4893] Vohra, Q. and E. Chen, "BGP Support for Four-octet AS Number Space", RFC 4893, May 2007.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, August 2007.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, October 2007.

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, December 2008.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, July 2013.

Appendix A. Implementation report

As noted elsewhere in this document, this set of migration mechanisms has multiple existing implementations in wide use.

- Cisco [CISCO]
- Juniper [JUNIPER]
- Alcatel-Lucent [ALU]

This is not intended to be an exhaustive list, as equivalent features do exist in other implementations, however the authors were unable to find publicly available documentation of the vendor-specific implementation to reference.

Authors' Addresses

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

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

Shane Amante
Apple, Inc.
1 Infinite Loop
Cupertino, CA 95014
US

Email: samante@apple.com

SIDR
Internet-Draft
Intended status: Informational
Expires: November 10, 2014

W. George
Time Warner Cable
S. Murphy
SPARTA, Inc., a Parsons Company

May 9, 2014

BGPSec Considerations for AS Migration
draft-ietf-sidr-as-migration-01

Abstract

This draft discusses considerations and methods 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 November 10, 2014.

Copyright Notice

Table of Contents

1. Introduction

There is a 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 described in draft-ietf-idr-as-migration [I-D.ietf-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 BGPSec 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].

1.2. Documentation note

This draft uses Autonomous System Numbers (ASNs) from the range reserved for documentation as described in RFC 5398 [RFC5398]. In the examples used here, they are intended to represent Globally Unique ASNs, not private ASNs as documented in RFC 1930 [RFC1930] section 10.

2. General Scenario

This draft assumes that the reader has read and understood the ASN migration method discussed in draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration] including its examples, as they will be heavily referenced here. The use case being discussed in the referenced draft 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 allow the migrating PE 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 is not applicable in this exact case. The reason that this migration drives 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 affected signatures before propagating the route to its eBGP neighbors.

In the following examples, AS64510 is being subsumed by AS64500, and both ASNs represent a Service Provider (SP) network (see Figure 1 in draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration]). AS64496 and 64499 represent end customer networks. References to PE, CE, and P routers mirror the diagrams and references in the above cited draft.

3. RPKI Considerations

Since the methods and implementation discussed in draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration] are not technically a part of the BGP4 protocol implementation, but rather a vendor-specific

feature, 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 BGPSec [I-D.ietf-sidr-bgpsec-protocol] validation might interact.

One of the primary considerations for this draft and migration is that companies rarely stop after 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 (e.g. a key roll). This leaves many SPs with multiple legacy ASNs which don’t go away very quickly, if at all. As solutions were 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 been carefully considered. While 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 has been made to keep the additional complexity during the transition period to a minimum, on the assumption that it will likely be protracted.

3.1. Origin Validation

Origin Validation does not need a unique solution to enable migration, as the existing protocol and procedure allows for a solution. In the scenario discussed, AS64510 is being replaced by AS64500. If there are any existing routes originated by AS64510 on the router being moved into the new ASN, this simply requires generating new ROAs for the routes with the new ASN and treating them as new routes to be added to AS64500. However, we also need to consider the situation where one or more other PEs are still in AS64510, and are originating one or more routes that may be distinct from any that the router under migration is originating. PE1 (which is now a part of AS64500 and instructed to use replace-as to remove AS64510 from the path) needs to be able to properly handle routes originated from AS64510. If the route now shows up as originating from AS64500, any downstream peers’ validation check will fail unless a ROA is *also* available for AS64500 as the origin ASN, meaning that there will be overlapping ROAs until all routers originating prefixes from AS64510 are migrated to AS64500. Overlapping ROAs are

permissible per RFC 6480 [RFC6480] section 3.2, and so managing origin validation during a migration like this is merely applying the defined case where a set of prefixes are originated from more than one ASN. Therefore, for each ROA that authorizes AS64510 to originate a prefix, a new ROA SHOULD also be created that authorizes AS64500 to originate the same prefix.

3.2. Path Validation

BGPSec Path Validation requires that each router in the AS Path cryptographically sign its update to assert that "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." (see point #2 in intro of [I-D.ietf-sidr-bgpsec-protocol]) Since the referenced AS 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 AS64510 to AS64500, it will be provisioned with the appropriate keys for AS64500 to allow it to forward-sign routes using AS64500. 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 64510", the presence of "local-as 64510" 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 AS64500, there is no guidance as to whether the BGPSec validator on CE1 still consider those valid by default. RFC4271 [RFC4271] section 6.3 mentions this match between the ASN of the peer and the AS_PATH data, but it is listed as an optional validation, rather than a requirement. Assuming that this mismatch will be allowed by vendor implementations and using it as a means to solve this migration case is likely to be problematic.

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 AS64510, not AS64500. The BGPSec speaker cannot 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 64510 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 is undesirable as discussed in draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration].

4. Requirements

These requirements are written under the assumption that the currently vendor-specific implementations will be standardized via draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration], as it makes little sense to build support into a standard for something that is not actually a standard itself. However, should IETF choose not to standardize the discussed method of AS migration, it is possible that this draft could be considered implementation guidance for those vendors that have support for this method of AS migration and wish to support it in their BGPSec implementation. In order to be deployable, any solution to the described problem needs to consider the following requirements, listed in no particular order:

- BGPSec MUST support AS Migration for both inbound and outbound route announcements (see Section 3.2.1 and 3.2.2). It SHOULD do this without reducing BGPSec’s protections for route path
- MUST NOT require any reconfiguration on the remote eBGP neighbor (CE)
- SHOULD confine configuration changes to the migrating PEs e.g. can’t require global configuration changes to support migration
- MUST NOT lengthen AS Path during migration
- MUST operate within existing trust boundaries e.g. can’t expect remote side to accept pCount=0 (see Section 3 of [I-D.ietf-sidr-bgpsec-protocol]) from untrusted/non-confed neighbor

5. Solution

As noted in [I-D.ietf-sidr-bgpsec-protocol], section 4.2, BGPSec already has a solution for hiding ASNs where increasing the AS Path length is undesirable. So one might think that a simple solution would be to retain the keys for AS64510 on PE1, and forward-sign towards CE1 with AS64510 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 accept pCount=0 announcements only on the sessions where they are expected. 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 is not a workable solution.

A better solution presents itself when considering how to handle routes coming from the CE toward the PE, where the routes are forward-signed to AS64510, but will eventually need to show AS64500 in the outbound route announcement. Because both AS64500 and AS64510 are in the same administrative domain, a signature from AS64510 forward-signed to AS64500 with pCount=0 would be acceptable as it would be within the appropriate trust boundary so that each BGP speaker could be explicitly configured to accept pCount=0 where appropriate between the two ASNs. At the very simplest, this could potentially be used at the eBGP boundary between the two ASNs during migration. Since the AS_PATH manipulation described above usually happens at the PE router on a per-session basis, and does not happen network-wide simultaneously, it is not generally appropriate to apply this AS hiding technique across all routes exchanged between the two ASNs, as it may result in routing loops and other undesirable behavior. Therefore the most appropriate place to implement this is on the local PE that still has eBGP sessions associated with AS64510 (using the transition knobs detailed in the companion draft). Since that PE has been moved to AS64500, it is not possible for it to forward-sign AS64510 with pCount=0 without some minor changes to the BGPSec implementation to address this use case.

AS migration is using AS_PATH and remote-AS manipulation to act as if a PE under migration exists simultaneously in both ASNs even though it is only configured with one global ASN. This draft proposes applying a similar technique to the BGPSec signatures generated for routing updates processed through this migration machinery. Each routing update that is received from or destined to an eBGP neighbor that is still using the old ASN (64510) will be signed twice, once with the ASN to be hidden and once with the ASN that will remain visible. In essence, we are treating the update as if the PE had an internal BGP hop and the update was passed across an eBGP session between AS64500 and AS64510, configured to use and accept pCount=0, while eliminating the processing and storage overhead of creating an actual eBGP session between the two ASNs within the PE router. This will result in a properly secured AS Path in the affected route

updates, because the PE router will be provisioned with valid keys for both AS64500 and AS64510. An important distinction here is that while AS migration under standard BGP4 is manipulating the AS_PATH attribute, BGPSec uses an attribute called the Secure_Path (see Section 3 of [I-D.ietf-sidr-bgpsec-protocol]), and BGPSec capable neighbors do not exchange AS_PATH information in their route announcements. However, a BGPSec neighbor peering with a non-BGPSec-capable neighbor will use the information found in Secure_Path to reconstruct a standard AS_PATH for updates sent to that neighbor. Unlike in Secure_Path where the ASN to be hidden is still present, but ignored when considering AS Path (due to pCount=0), when reconstructing an AS_PATH for a non-BGPSec neighbor, the pCount=0 ASNs will not appear in the AS_PATH at all (see section 4.4 of the above-referenced draft). This draft is not changing existing AS_PATH reconstruction behavior, merely highlighting it for clarity.

The procedure to support AS Migration in BGPSec is slightly different depending on whether the PE under migration is receiving the routes from one of its eBGP peers ("inbound" as in section 3.2.2) or destined toward the eBGP peers ("outbound" as in section 3.2.1).

5.1. Outbound (PE->CE)

When a PE router receives an update destined for an eBGP neighbor that is locally configured with AS-migration knobs as discussed in draft-ietf-idr-as-migration [I-D.ietf-idr-as-migration], it MUST generate a valid BGPSec signature as defined in [I-D.ietf-sidr-bgpsec-protocol] for _both_ configured ASNs. It MUST generate a signature from the new (global) ASN forward signing to the old (local) ASN with pCount=0, and then it MUST generate a forward signature from the old (local) ASN to the target eBGP ASN with pCount=1 as normal.

5.2. Inbound (CE->PE)

When a PE router receives an update from an eBGP neighbor that is locally configured with AS-migration knobs (i.e. the opposite direction of the previous route flow), it MUST generate a signature from the old (local) ASN forward signing to the new (global) ASN with pCount=0. It is not necessary to generate the second signature from the new (global) ASN because the ASBR will generate that when it forward signs towards its eBGP peers as defined in normal BGPSec operation. This is a deviation from standard BGPSec behavior in that typically a signature is not added when a routing update is sent across an iBGP session, and the next signature is added by the ASBR when it forward-signs toward its eBGP peer as the routing update exits the ASN.

5.3. Other considerations

In this case, the PE is adding BGPSec attributes to routes received from or destined to an iBGP neighbor, and using pCount=0 to mask them. While this is not prohibited by the current BGPSec specification, routers that receive updates from iBGP neighbors MUST NOT reject updates with new (valid) BGPSec attributes, including the presence of pCount=0 on a previous signature, or they will interfere with this implementation. In similar fashion, any route-reflectors in the path of these updates MUST reflect them transparently to their clients.

In order to secure this set of signatures, the PE router MUST be provisioned with valid keys for _both_ configured ASNs (old and new), and the key for the old ASN MUST be kept valid until all eBGP sessions are migrated to the new ASN. Downstream neighbors will see this as a valid BGPSec path, as they will simply trust that their upstream neighbor accepted pCount=0 because it was explicitly configured to do so based on a trust relationship and business relationship between the upstream and its neighbor (the old and new ASNs).

5.4. Example

The following example will illustrate the method being used above. As with previous examples, PE1 is the router being migrated, AS64510 is the old AS, which is being subsumed by AS64500, the "keep" AS. 64505 is another external peer, used to demonstrate what the announcements will look like to a third party peer that is not part of the migration. Some additional notation is used to delineate the details of each signature as follows:

The origin BGPSEC signature attribute takes the form: sig(<Target ASN>, Origin ASN, pCount, NLRI Prefix) key

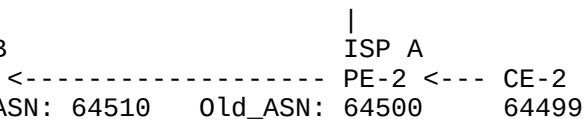
Intermediate BGPSEC signature attributes take the form: sig(<Target ASN>, Signer ASN, pCount, <most recent sig field>) key

Equivalent AS_PATH refers to what the AS_PATH would look like if it was reconstructed to be sent to a non-BGPSec peer, while Secure_Path shows the AS Path as represented between BGPSec peers.

Note: The representation of signature attribute generation is being simplified here somewhat for the sake of brevity; the actual details of the signing process are as described Sections 4.1 and 4.2 in [I-D.ietf-sidr-bgpsec-protocol]. For example, what is covered by the signature also includes Flags, Algorithm Suite ID, NLRI length, etc.

Also, the key is not carried in the update, instead the SKI is carried.

Before Merger



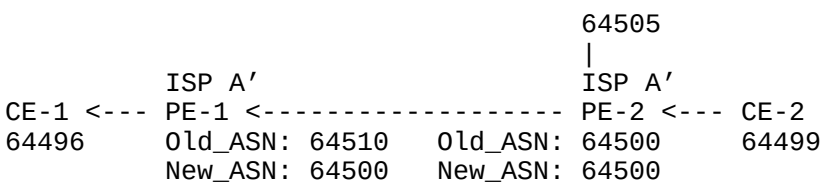
CE-2 to PE-2: sig(<64500>, O=64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64499)
Secure_Path=(64499)
length=sum(pCount)=1

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig2]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64500,64499)
Secure_Path=(64500,64499)
length=sum(pCount)=2

PE-2 to PE-1: sig(<64510>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig3]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64500,64499)
Secure_Path=(64500,64499)
length=sum(pCount)=2

PE-1 to CE-1: sig(<64496>, 64510, pCount=1, <sig3>)K_64510-PE1 [sig4]
sig(<64510>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig3]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH= (64510,64500,64499)
Secure_Path=(64510,64500,64499)
length=sum(pCount)=3

Migrating, route flow outbound PE-1 to CE-1



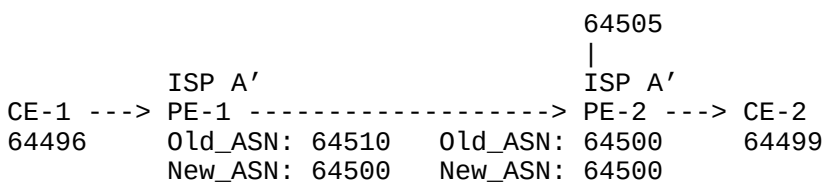

```
CE-2 to PE-2:  sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64499)
               Secure_Path=(64499)
               length=sum(pCount)=1

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig11>)K_64500-PE2  [sig12]
               sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64500,64499)
               Secure_Path=(64500,64499)
               length=sum(pCount)=2

PE-2 to PE-1:  sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64499)
               Secure_Path=(64499)
               length=sum(pCount)=1
#PE-2 sends to PE-1 (in iBGP) the exact same update as received from AS64499.


PE-1 to CE-1:  sig(<64496>, 64510, pCount=1, <sig13>)K_64510-PE1  [sig14]
               sig(<64510>, 64500, pCount=0, <sig11>)K_64500-PE2  [sig13]
               sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64510,64499)
               Secure_Path=(64510, 64500(pCount=0),64499)
               length=sum(pCount)=2 (length is NOT 3)
#PE1 adds [sig13] acting as AS64500
#PE1 accepts [sig13] with pCount=0 acting as AS64510,
#as it would if it received sig13 from an eBGP peer
```

Migrating, route flow inbound CE-1 to PE-1




```
CE-1 to PE-1:  sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64496)
               Secure_Path=(64496)
               length=sum(pCount)=1

PE-1 to PE-2:  sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64496)
               Secure_Path=(64510 (pCount=0),64496)
               length=sum(pCount)=1 (length is NOT 2)
#PE1 adds [sig22] acting as AS64510
#PE1 accepts [sig22] with pCount=0 acting as AS64500,
#as it would if it received sig22 from an eBGP peer

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig22>)K_64500-PE2  [sig23]
               sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64500,64496)
               Secure_Path=(64500,64510 (pCount=0), 64496)
               length=sum(pCount)=2 (length is NOT 3)

PE-2 to CE-2:  sig(<64499>, 64500, pCount=1, <sig22>)K_64500-PE2  [sig24]
               sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64500,64496)
               Secure_Path=(64500, 64510 (pCount=0), 64496)
               length=sum(pCount)=2 (length is NOT 3)
```

6. Acknowledgements

Thanks to Kotikalapudi Sriram, Shane Amante, Warren Kumari, and Terry Manderson for their review comments.

Additionally, the solution presented in this draft is an amalgam of several SIDR interim meeting discussions plus a discussion at IETF85, collected and articulated thanks to Sandy Murphy.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This draft discusses a process by which one ASN is migrated into and subsumed by another. Because this process involves manipulating the AS_Path in a BGP route to make it deviate from the actual path that it took through the network, this migration process is attempting to do exactly what BGPSec is working to prevent. BGPSec MUST be able to manage this legitimate use of AS_Path manipulation without generating a vulnerability in the RPKI route security infrastructure.

The solution discussed above is considered to be reasonably secure from exploitation by a malicious actor because it requires both signatures to be secured as if they were forward-signed between two eBGP neighbors. This requires any router using this solution to be provisioned with valid keys for both the migrated and subsumed ASN so that it can generate valid signatures for each of the two ASNs it is adding to the path. If the AS's keys are compromised, or zero-length keys are permitted, this does potentially enable an AS_PATH shortening attack, but this is not fundamentally altering the existing security risks for BGPSec.

9. References

9.1. Normative References

[I-D.ietf-idr-as-migration]
George, W. and S. Amante, "Autonomous System (AS) Migration Features and Their Effects on the BGP AS_PATH Attribute", draft-ietf-idr-as-migration-00 (work in progress), February 2014.

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M., "BGPSEC Protocol Specification", draft-ietf-sidr-bgpsec-protocol-08 (work in progress), November 2013.

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

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, December 2008.

9.2. Informative References

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[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.

Authors' Addresses

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

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

Sandy Murphy
SPARTA, Inc., a Parsons Company
7110 Samuel Morse Drive
Columbia, MD 21046
US

Phone: +1 443-430-8000
Email: sandy@tislabs.com

SIDR
Internet-Draft
Intended status: Standards Track
Expires: June 10, 2017

W. George
S. Murphy
SPARTA, Inc., a Parsons Company

December 7, 2016

BGPSec Considerations for AS Migration
draft-ietf-sidr-as-migration-06

Abstract

This document discusses considerations and methods 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 June 10, 2017.

Table of Contents

1. Introduction

A method of managing a BGP Autonomous System Number (ASN) migration is described in RFC7705 [RFC7705]. Since it concerns the handling of AS_PATH attributes, it is necessary to ensure that the process and features are properly supported in BGPSec [I-D.ietf-sidr-bgpsec-protocol], because BGPSec 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].

1.2. Documentation note

This document uses Autonomous System Numbers (ASNs) from the range reserved for documentation as described in RFC 5398 [RFC5398]. In the examples used here, they are intended to represent Globally Unique ASNs, not ASNs reserved for private use as documented in RFC 1930 [RFC1930] section 10.

2. General Scenario

This document assumes that the reader has read and understood the ASN migration method discussed in RFC7705 [RFC7705] including its examples (see section 2 of the referenced document), as they will be heavily referenced here. The use case being discussed in the referenced document is as follows: For whatever the reason, a provider is in the process of merging two or more ASes, where eventually one subsumes the other(s). BGP AS Confederations RFC 5065 [RFC5065] is not enabled between the ASes, but a mechanism is being used to modify BGP’s default behavior and allow the migrating Provider Edge router (PE) to masquerade as the old ASN for the Provider Edge to Customer Edge (PE-CE) eBGP session, or to manipulate the AS_PATH, or both. While BGPSec [I-D.ietf-sidr-bgpsec-protocol] does have a method to handle standard confederation implementations, it is not applicable in this exact case. This migration requires a slightly different solution in BGPSec than for a standard confederation because 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 would strip the affected signatures before propagating the route to its eBGP neighbors.

In the following examples (section 5.4) (Section 5.4), AS64510 is being subsumed by AS64500, and both ASNs represent a Service Provider (SP) network (see Figures 1 & 2 in RFC7705 [RFC7705]). AS64496 and 64499 represent end customer networks. References to PE, CE, and P routers mirror the diagrams and references in the above cited draft.

3. RPKI Considerations

The methods and implementation discussed in RFC7705 [RFC7705] are widely used during network integrations resulting from mergers and acquisitions, as well as network redesigns, and therefore it is necessary to support 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 BGPSec [I-D.ietf-sidr-bgpsec-protocol] validation might interact.

One of the primary considerations for this document and migration is that service providers (SPs) rarely stop after one merger/acquisition/divestiture, and end up accumulating several legacy ASNs over time. Since they are using methods to migrate that are transparent to and therefore 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 (e.g. a key roll). Because they are not forcing a simultaneous migration (i.e. both ends switch to the new ASN at an agreed-upon time), there is no incentive for a given customer to complete the move from the old ASN to the new. This leaves many SPs with multiple legacy ASNs which don't go away very quickly, if at all. As solutions were being proposed for RPKI implementations to solve this transition case, the WG carefully considered operational complexity and hardware scaling issues associated with maintaining multiple legacy ASN keys on routers throughout the combined network. While SPs who choose to remain in this transition phase indefinitely invite added risks because of the operational complexity and scaling considerations associated with maintaining multiple legacy ASN keys on routers throughout the combined network, saying "don't do this" is of limited utility as a solution. As a result, this solution attempts to minimize the additional complexity during the transition period, on the assumption that it will likely be protracted. Note: While this document primarily discusses service provider considerations, it is not solely applicable to SPs, as enterprises often migrate between ASNs using the same functionality. What follows is a discussion of origin and path validation functions and how they interact with ASN migrations.

3.1. Origin Validation

Route Origin Validation as defined by RFC 6480 [RFC6480] does not modification to enable AS migration, as the existing protocol and procedure allows for a solution. In the scenario discussed in RFC 7705 [RFC7705], AS64510 is being replaced by AS64500. If there are any existing routes originated by AS64510 on the router being moved into the new ASN, this simply requires generating new Route Origination Authorizations (ROAs) for the routes with the new ASN and treating them as new routes to be added to AS64500. However, we also need to consider the situation where one or more other PEs are still in AS64510, and are originating one or more routes that may be distinct from any that the router under migration is originating. PE1 (which is now a part of AS64500 and instructed to use Replace Old AS as defined in RFC 7705 [RFC7705] to remove AS64510 from the path) needs to be able to properly handle routes originated from AS64510. If the route now shows up as originating from AS64500, any downstream peers' validation check will fail unless a ROA is *also* available for AS64500 as the origin ASN. In addition to generating a ROA for

65400 for any prefixes originated by the router being moved, it may be necessary to generate ROAs for 65400 for prefixes that are originating on routers still in 65410, since the AS replacement function will change the origin AS in some cases. This means that there will be multiple ROAs showing different ASes authorized to orignate the same prefixes until all routers originating prefixes from AS64510 are migrated to AS64500. Multiple ROAs of this type are permissible per RFC 6480 [RFC6480] section 3.2, and so managing origin validation during a migration like this is merely applying the defined case where a set of prefixes are originated from more than one ASN. Therefore, for each ROA that authorizes the old ASN (e.g. AS64510) to originate a prefix, a new ROA MUST also be created that authorizes the replacing ASN (e.g. AS64500) to originate the same prefix.

3.2. Path Validation

BGPSec Path Validation requires that each router in the AS Path cryptographically sign its update to assert that "Every AS on the path of ASes listed in the update message has explicitly authorized the advertisement of the route to the subsequent AS in the path." (see intro of [I-D.ietf-sidr-bgpsec-protocol]) Since the referenced AS 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 AS64510 to AS64500, it will be provisioned with the appropriate keys for AS64500 to allow it to forward-sign routes using AS64500. However, there is no guidance in the BGPSec protocol specification [I-D.ietf-sidr-bgpsec-protocol] on whether or not the forward-signed ASN value is required to match the configured remote AS to validate properly. That is, if CE1’s BGP session is configured as "remote AS 64510", the presence of "local AS 64510" 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 AS64500, there is no guidance as to whether the BGPSec validator on CE1 still considers those valid by default. RFC4271 [RFC4271] section 6.3 mentions this match between the ASN of the peer and the AS_PATH data, but it is listed as an optional validation, rather than a requirement. We cannot assume that this mismatch will be allowed by vendor implementations and thus using it as a means to solve this migration case is likely to be problematic.

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 AS64510, not AS64500. The BGPSec speaker cannot 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 64510 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 is unacceptable as discussed in RFC7705 [RFC7705].

4. Requirements

In order to be deployable, any solution to the described problem needs to consider the following requirements, listed in no particular order. BGPSec:

- MUST support AS Migration for both inbound and outbound route announcements (see Section 3.2.1 and 3.2.2) without reducing BGPSec's protections for route path
- MUST NOT require any reconfiguration on the remote eBGP neighbor (CE)
- SHOULD NOT require global (i.e. network-wide) configuration changes to support migration. The goal is to limit required configuration changes to the devices (PEs) being migrated.
- MUST NOT lengthen AS Path during migration
- MUST operate within existing trust boundaries e.g. can't expect remote side to accept pCount=0 (see Section 4.2 of [I-D.ietf-sidr-bgpsec-protocol]) from untrusted/non-confed neighbor

5. Solution

As noted in [I-D.ietf-sidr-bgpsec-protocol], section 4.2, BGPSec already has a solution for hiding ASNs where increasing the AS Path length is undesirable. So a simple solution would be to retain the keys for AS64510 on PE1, and forward-sign towards CE1 with AS64510 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 accept pCount=0 announcements only on the sessions where they are expected. For the same reason that things like "Local AS" [RFC7705] 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 is not a workable solution.

A better solution presents itself when considering how to handle routes coming from the CE toward the PE, where the routes are forward-signed to AS64510, but will eventually need to show AS64500 in the outbound route announcement. Because both AS64500 and AS64510 are in the same administrative domain, a signature from AS64510 forward-signed to AS64500 with pCount=0 would be acceptable as it would be within the appropriate trust boundary so that each BGP speaker could be explicitly configured to accept pCount=0 where appropriate between the two ASNs. At the very simplest, this could potentially be used at the eBGP boundary between the two ASNs during migration. Since the AS_PATH manipulation described above usually happens at the PE router on a per-session basis, and does not happen network-wide simultaneously, it is not generally appropriate to apply this AS hiding technique across all routes exchanged between the two ASNs, as it may result in routing loops and other undesirable behavior. Therefore the most appropriate place to implement this is on the local PE that still has eBGP sessions with peers expecting to peer with AS64510 (using the transition mechanisms detailed in RFC7705 [RFC7705]). Since that PE has been moved to AS64500, it is not possible for it to forward-sign AS64510 with pCount=0 without some minor changes to the BGPSec behavior to address this use case.

AS migration is using AS_PATH and remote AS manipulation to act as if a PE under migration exists simultaneously in both ASNs even though it is only configured with one global ASN. This document describes applying a similar technique to the BGPSec signatures generated for routing updates processed through this migration machinery. Each routing update that is received from or destined to an eBGP neighbor that is still using the old ASN (64510) will be signed twice, once with the ASN to be hidden and once with the ASN that will remain visible. In essence, we are treating the update as if the PE had an internal BGP hop and the update was passed across an eBGP session between AS64500 and AS64510, configured to use and accept pCount=0, while eliminating the processing and storage overhead of creating an actual eBGP session between the two ASNs within the PE router. This will result in a properly secured AS Path in the affected route

updates, because the PE router will be provisioned with valid keys for both AS64500 and AS64510. An important distinction here is that while AS migration under standard BGP4 is manipulating the AS_PATH attribute, BGPSec uses an attribute called the Secure_Path (see Section 3.1 of [I-D.ietf-sidr-bgpsec-protocol]), and BGPSec capable neighbors do not exchange AS_PATH information in their route announcements. However, a BGPSec neighbor peering with a non-BGPSec-capable neighbor will use the information found in Secure_Path to reconstruct a standard AS_PATH for updates sent to that neighbor. Unlike in Secure_Path where the ASN to be hidden is still present, but ignored when considering AS Path (due to pCount=0), when reconstructing an AS_PATH for a non-BGPSec neighbor, the pCount=0 ASNs will not appear in the AS_PATH at all (see section 4.4 of the [I-D.ietf-sidr-bgpsec-protocol]). This document is not changing existing AS_PATH reconstruction behavior, merely highlighting it for clarity.

The procedure to support AS Migration in BGPSec is slightly different depending on whether the PE under migration is receiving the routes from one of its eBGP peers ("inbound" as in section 3.2.2) or destined toward the eBGP peers ("outbound" as in section 3.2.1).

5.1. Outbound (PE->CE)

When a PE router receives an update destined for an eBGP neighbor that is locally configured with AS-migration mechanisms as discussed in RFC7705 [RFC7705], it MUST generate a valid BGPSec signature as defined in [I-D.ietf-sidr-bgpsec-protocol] for _both_ configured ASNs. It MUST generate a signature from the new (global) ASN forward signing to the old (local) ASN with pCount=0, and then it MUST generate a forward signature from the old (local) ASN to the target eBGP ASN with pCount=1 as normal.

5.2. Inbound (CE->PE)

When a PE router receives an update from an eBGP neighbor that is locally configured with AS-migration mechanisms (i.e. the opposite direction of the previous route flow), it MUST generate a signature from the old (local) ASN forward signing to the new (global) ASN with pCount=0. It is not necessary to generate the second signature from the new (global) ASN because the Autonomous System Border Router (ASBR) will generate that when it forward signs towards its eBGP peers as defined in normal BGPSec operation. Note that a signature is not normally added when a routing update is sent across an iBGP session. The requirement to sign updates in iBGP represents a change to the normal behavior for this specific AS-migration scenario only.

5.3. Other considerations

In this case, the PE is adding BGPSec attributes to routes received from or destined to an iBGP neighbor, and using pCount=0 to mask them. While this is not prohibited by BGPSec [I-D.ietf-sidr-bgpsec-protocol], BGPSec-capable routers that receive updates from BGPSec-enabled iBGP neighbors MUST accept updates with new (properly-formed) BGPSec attributes, including the presence of pCount=0 on a previous signature, or they will interfere with this method. In similar fashion, any BGPSec-capable route-reflectors in the path of these updates MUST reflect them transparently to their BGPSec-capable clients.

In order to secure this set of signatures, the PE router MUST be provisioned with valid keys for _both_ configured ASNs (old and new), and the key for the old ASN MUST be kept valid until all eBGP sessions are migrated to the new ASN. Downstream neighbors will see this as a valid BGPSec path, as they will simply trust that their upstream neighbor accepted pCount=0 because it was explicitly configured to do so based on a trust relationship and business relationship between the upstream and its neighbor (the old and new ASNs).

Additionally, section 4 of RFC7705 [RFC7705] discusses methods in which AS migrations can be completed for iBGP peers such that a session between two routers will be treated as iBGP even if the neighbor ASN is not the same ASN on each peer’s global configuration. As far as BGPSec is concerned, this requires the same procedure as when the routers migrating are applying AS migration mechanisms to eBGP peers, but the router functioning as the "ASBR" between old and new ASN is different. In eBGP, the router being migrated has direct eBGP sessions to the old ASN and signs from old ASN to new with pCount=0 before passing the update along to additional routers in its global (new) ASN. In iBGP, the router being migrated is receiving updates (that may have originated either from eBGP neighbors or other iBGP neighbors) from its downstream neighbors in the old ASN, and MUST sign those updates from old ASN to new with pCount=0 before sending them on to other peers.

5.4. Example

The following example will illustrate the method being used above. As with previous examples, PE1 is the router being migrated, AS64510 is the old ASN, which is being subsumed by AS64500, the ASN to be permanently retained. 64505 is another external peer, used to demonstrate what the announcements will look like to a third party peer that is not part of the migration. Some additional notation is used to delineate the details of each signature as follows:

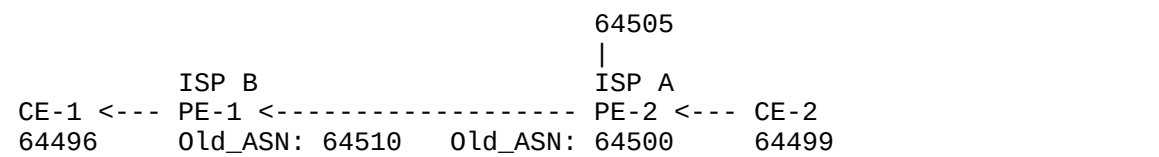
The origin BGPSEC signature attribute takes the form: sig(<Target ASN>, Origin ASN, pCount, NLRI Prefix) key

Intermediate BGPSEC signature attributes take the form: sig(<Target ASN>, Signer ASN, pCount, <most recent sig field>) key

Equivalent AS_PATH refers to what the AS_PATH would look like if it was reconstructed to be sent to a non-BGPSec peer, while Secure_Path shows the AS Path as represented between BGPSec peers.

Note: The representation of signature attribute generation is being simplified here somewhat for the sake of brevity; the actual details of the signing process are as described Sections 4.1 and 4.2 in [I-D.ietf-sidr-bgpsec-protocol]. For example, what is covered by the signature also includes Flags, Algorithm Suite ID, NLRI length, etc. Also, the key is not carried in the update, instead the SKI is carried.

Before Merger



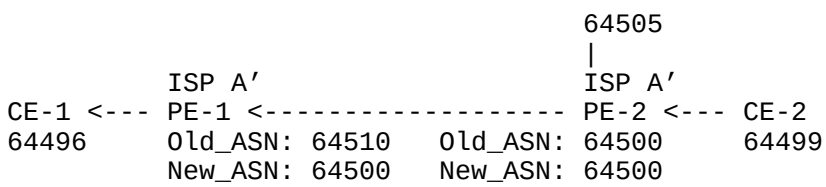
CE-2 to PE-2: sig(<64500>, O=64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64499)
Secure_Path=(64499)
length=sum(pCount)=1

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig2]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64500,64499)
Secure_Path=(64500,64499)
length=sum(pCount)=2

PE-2 to PE-1: sig(<64510>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig3]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH=(64500,64499)
Secure_Path=(64500,64499)
length=sum(pCount)=2

PE-1 to CE-1: sig(<64496>, 64510, pCount=1, <sig3>)K_64510-PE1 [sig4]
sig(<64510>, 64500, pCount=1, <sig1>)K_64500-PE2 [sig3]
sig(<64500>, 64499, pCount=1, N)K_64499-CE2 [sig1]
Equivalent AS_PATH= (64510,64500,64499)
Secure_Path=(64510,64500,64499)
length=sum(pCount)=3

Migrating, route flow outbound PE-1 to CE-1



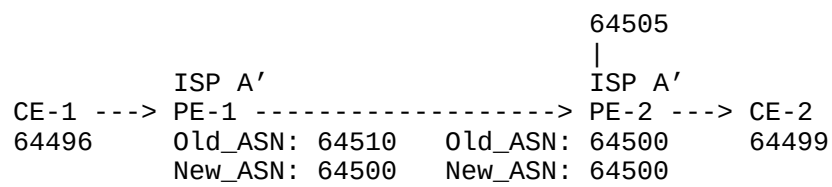

```
CE-2 to PE-2:  sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64499)
               Secure_Path=(64499)
               length=sum(pCount)=1

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig11>)K_64500-PE2  [sig12]
               sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64500,64499)
               Secure_Path=(64500,64499)
               length=sum(pCount)=2

PE-2 to PE-1:  sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64499)
               Secure_Path=(64499)
               length=sum(pCount)=1
#PE-2 sends to PE-1 (in iBGP) the exact same update
#as received from AS64499.


PE-1 to CE-1:  sig(<64496>, 64510, pCount=1, <sig13>)K_64510-PE1  [sig14]
               sig(<64510>, 64500, pCount=0, <sig11>)K_64500-PE2  [sig13]
               sig(<64500>, 64499, pCount=1, N)K_64499-CE2  [sig11]
               Equivalent AS_PATH=(64510,64499)
               Secure_Path=(64510, 64500(pCount=0),64499)
               length=sum(pCount)=2 (length is NOT 3)
#PE1 adds [sig13] acting as AS64500
#PE1 accepts [sig13] with pCount=0 acting as AS64510,
#as it would if it received sig13 from an eBGP peer
```

Migrating, route flow inbound CE-1 to PE-1




```
CE-1 to PE-1:  sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64496)
               Secure_Path=(64496)
               length=sum(pCount)=1

PE-1 to PE-2:  sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64496)
               Secure_Path=(64510 (pCount=0),64496)
               length=sum(pCount)=1 (length is NOT 2)
#PE1 adds [sig22] acting as AS64510
#PE1 accepts [sig22] with pCount=0 acting as AS64500,
#as it would if it received sig22 from an eBGP peer

PE-2 to 64505: sig(<64505>, 64500, pCount=1, <sig22>)K_64500-PE2  [sig23]
               sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64500,64496)
               Secure_Path=(64500,64510 (pCount=0), 64496)
               length=sum(pCount)=2 (length is NOT 3)

PE-2 to CE-2:  sig(<64499>, 64500, pCount=1, <sig22>)K_64500-PE2  [sig24]
               sig(<64500>, 64510, pCount=0, <sig21>)K_64510-PE1  [sig22]
               sig(<64510>, 64496, pCount=1, N)K_64496-CE1   [sig21]
               Equivalent AS_PATH=(64500,64496)
               Secure_Path=(64500, 64510 (pCount=0), 64496)
               length=sum(pCount)=2 (length is NOT 3)
```

6. Acknowledgements

Thanks to Kotikalapudi Sriram, Shane Amante, Warren Kumari, Terry Manderson, Keyur Patel, Alia Atlas, and Alvaro Retana for their review comments.

Additionally, the solution presented in this document is an amalgam of several SIDR interim meeting discussions plus a discussion at IETF85, collected and articulated thanks to Sandy Murphy.

7. IANA Considerations

This memo includes no request to IANA.

8. Note for RFC Editor

This section can be removed prior to publication.

RFC Editor - this document updates draft-ietf-sidr-bgpsec-protocol, but the normal Updates= metadata method cannot be used until an RFC number is assigned to the document being updated. Please ensure that the metadata is corrected when the bgpsec-protocol document has been assigned an RFC number.

9. Security Considerations

RFC7705 [RFC7705] discusses a process by which one ASN is migrated into and subsumed by another. Because this process involves manipulating the AS_Path in a BGP route to make it deviate from the actual path that it took through the network, this migration process is attempting to do exactly what BGPSec is working to prevent. BGPSec MUST be able to manage this legitimate use of AS_Path manipulation without generating a vulnerability in the RPKI route security infrastructure, and this document was written to define the method by which the protocol can meet this need.

The solution discussed above is considered to be reasonably secure from exploitation by a malicious actor because it requires both signatures to be secured as if they were forward-signed between two eBGP neighbors. This requires any router using this solution to be provisioned with valid keys for both the migrated and subsumed ASN so that it can generate valid signatures for each of the two ASNs it is adding to the path. If the AS’s keys are compromised, or zero-length keys are permitted, this does potentially enable an AS_PATH shortening attack, but these are existing security risks for BGPSec.

10. References

10.1. Normative References

[I-D.ietf-sidr-bgpsec-protocol]
Lepinski, M. and K. Sriram, "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol-20 (work in progress), December 2016.

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

[RFC7705] George, W. and S. Amante, "Autonomous System Migration Mechanisms and Their Effects on the BGP AS_PATH Attribute", RFC 7705, DOI 10.17487/RFC7705, November 2015, <http://www.rfc-editor.org/info/rfc7705>.

10.2. Informative References

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, DOI 10.17487/RFC1930, March 1996, <http://www.rfc-editor.org/info/rfc1930>.

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

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

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, DOI 10.17487/RFC5398, December 2008, <http://www.rfc-editor.org/info/rfc5398>.

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

Authors' Addresses

Wesley George

Email: wesgeorge@puck.nether.net

Sandy Murphy
SPARTA, Inc., a Parsons Company
7110 Samuel Morse Drive
Columbia, MD 21046
US

Phone: +1 443-430-8000
Email: sandy@tislabs.com

Secure Inter-Domain Routing Working Group
Internet-Draft
Updates: 6487 (if approved)
Intended status: Standard Track
Expires: July 9, 2017

M. Reynolds
IPSw
S. Turner
sn3rd
S. Kent
BBN

January 5, 2017

A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests
draft-ietf-sidr-bgpsec-pki-profiles-21

Abstract

This document defines a standard profile for X.509 certificates used to enable validation of Autonomous System (AS) paths in the Border Gateway Protocol (BGP), as part of an extension to that protocol known as BGPsec. BGP is the standard for inter-domain routing in the Internet; it is the "glue" that holds the Internet together. BGPsec is being developed as one component of a solution that addresses the requirement to provide security for BGP. The goal of BGPsec is to provide full AS path validation based on the use of strong cryptographic primitives. The end-entity (EE) certificates specified by this profile are issued to routers within an Autonomous System. Each of these certificates is issued under a Resource Public Key Infrastructure (RPKI) Certification Authority (CA) certificate. These CA certificates and EE certificates both contain the AS Identifier Delegation extension. An EE certificate of this type asserts that the router(s) holding the corresponding private key are authorized to emit secure route advertisements on behalf of the AS(es) specified in the certificate. This document also profiles the format of certification requests, and specifies Relying Party (RP) certificate path validation procedures for these EE certificates. This document extends the RPKI; therefore, this documents updates the RPKI Resource Certificates Profile (RFC 6487).

Status of this Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months

and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Table of Contents

1. Introduction

This document defines a profile for X.509 end-entity (EE) certificates [RFC5280] for use in the context of certification of Autonomous System (AS) paths in the BGPsec. Such certificates are termed "BGPsec Router Certificates". The holder of the private key associated with a BGPsec Router Certificate is authorized to send secure route advertisements (BGPsec UPDATEs) on behalf of the AS(es) named in the certificate. A router holding the private key is authorized to send route advertisements (to its peers) identifying the router's ASN as the source of the advertisements. A key property provided by BGPsec is that every AS along the AS PATH can verify that the other ASes along the path have authorized the advertisement of the given route (to the next AS along the AS PATH).

This document is a profile of [RFC6487], which is a profile of [RFC5280]; thus this document updates [RFC6487]. It establishes requirements imposed on a Resource Certificate that is used as a BGPsec Router Certificate, i.e., it defines constraints for certificate fields and extensions for the certificate to be valid in this context. This document also profiles the certification requests used to acquire BGPsec Router Certificates. Finally, this document specifies the Relying Party (RP) certificate path validation procedures for these certificates.

1.1. Terminology

It is assumed that the reader is familiar with the terms and concepts described in "A Profile for X.509 PKIX Resource Certificates" [RFC6487], "BGPsec Protocol Specification" [ID.sidr-bgpsec-protocol], "A Border Gateway Protocol 4 (BGP-4)" [RFC4271], "BGP Security Vulnerabilities Analysis" [RFC4272], "Considerations in Validating the Path in BGP" [RFC5123], and "Capability Advertisement with BGP-4" [RFC5492].

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

2. Describing Resources in Certificates

Figure 1 depicts some of the entities in the RPKI and some of the products generated by RPKI entities. IANA issues a Certification Authority (CA) certificate to each Regional Internet Registry (RIR). The RIR, in turn, issues a CA certificate to an Internet Service Provider (ISP). The ISP in turn issues EE Certificates to itself to enable verification of signatures on RPKI signed objects. The CA

also generates Certificate Revocation Lists (CRLs). These CA and EE certificates are referred to as "Resource Certificates", and are profiled in [RFC6487]. [RFC6480] envisioned using Resource Certificates to enable verification of Manifests [RFC6486] and Route Origin Authorizations (ROAs) [RFC6482]. ROAs and Manifests include the Resource Certificates used to verify them.

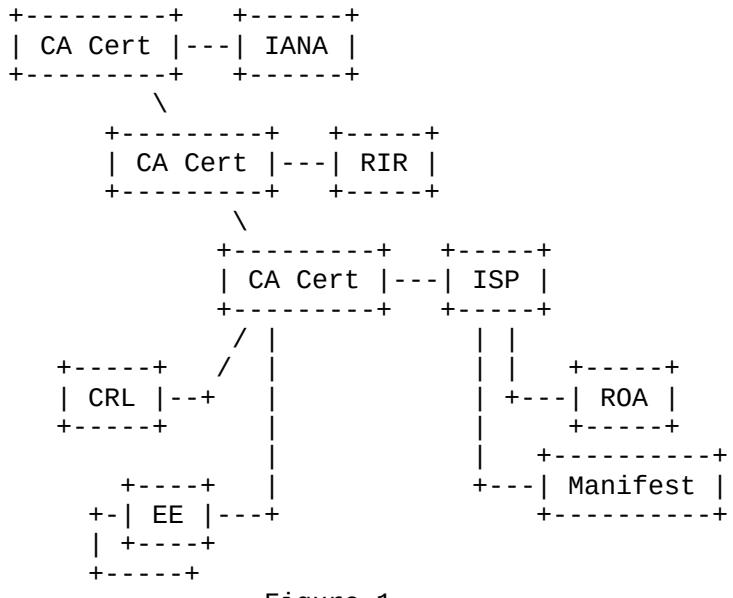


Figure 1

This document defines another type of Resource Certificate, which is referred to as a "BGPsec Router Certificate". The purpose of this certificate is explained in Section 1 and falls within the scope of appropriate uses defined within [RFC6484]. The issuance of BGPsec Router Certificates has minimal impact on RPKI CAs because the RPKI CA certificate and CRL profile remain unchanged (i.e., they are as specified in [RFC6487]). Further, the algorithms used to generate RPKI CA certificates that issue the BGPsec Router Certificates and the CRLs necessary to check the validity of the BGPsec Router Certificates remain unchanged (i.e., they are as specified in [RFC7935]). The only impact is that RPKI CAs will need to be able to process a profiled certificate request (see Section 5) signed with algorithms found in [ID.sidr-bgpsec-algs]. BGPsec Router Certificates are used only to verify the signature on the BGPsec certificate request (only CAs process these) and the signature on a BGPsec Update Message [ID.sidr-bgpsec-protocol] (only BGPsec routers process these); BGPsec Router Certificates are not used to process Manifests and ROAs or verify signatures on Certificates or CRLs.

This document enumerates only the differences between this profile and the profile in [RFC6487]. Note that BGPsec Router Certificates are EE certificates and as such there is no impact on process described in [RFC6916].

3. Updates to [RFC6487]

3.1 BGPsec Router Certificate Fields

A BGPsec Router Certificate is consistent with the profile in [RFC6487] as modified by the specifications in this section. As such, it is a valid X.509 public key certificate and consistent with the PKIX profile [RFC5280]. The differences between this profile and the profile in [RFC6487] are specified in this section.

3.1.1. Subject

Common name encoding options that are supported are printableString and UTF8String. For BGPsec Router Certificates, it is RECOMMENDED that the common name attribute contain the literal string "ROUTER-" followed by the 32-bit AS Number [RFC3779] encoded as eight hexadecimal digits and that the serial number attribute contain the 32-bit BGP Identifier [RFC4271] (i.e., the router ID) encoded as eight hexadecimal digits. If there is more than one AS number, the choice of which to include in the common name is at the discretion of the Issuer. If the same certificate is issued to more than one router (hence the private key is shared among these routers), the choice of the router ID used in this name is at the discretion of the Issuer.

3.1.2. Subject Public Key Info

Refer to section 3.1 of [ID.sidr-bgpsec-algs].

3.1.3. BGPsec Router Certificate Version 3 Extension Fields

3.1.3.1. Basic Constraints

BGPsec speakers are EEs; therefore, the Basic Constraints extension must not be present, as per [RFC6487].

3.1.3.2. Extended Key Usage

BGPsec Router Certificates MUST include the Extended Key Usage (EKU) extension. As specified in [RFC6487] this extension must be marked as non-critical. This document defines one EKU for BGPsec Router Certificates:

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

id-kp-bgpsec-router OBJECT IDENTIFIER ::= { id-kp 30 }
```

A BGPsec router MUST require the extended key usage extension to be present in a BGPsec Router Certificate it receives. If multiple KeyPurposeId values are included, the BGPsec routers need not recognize all of them, as long as the required KeyPurposeId value is present. BGPsec routers MUST reject certificates that do not contain the BGPsec Router EKU even if they include the anyExtendedKeyUsage OID defined in [RFC5280].

3.1.3.3. Subject Information Access

This extension is not used in BGPsec Router Certificates. It MUST be omitted.

3.1.3.4. IP Resources

This extension is not used in BGPsec Router Certificates. It MUST be omitted.

3.1.3.5. AS Resources

Each BGPsec Router Certificate MUST include the AS Resource Identifier Delegation extension, as specified in section 4.8.11 of [RFC6487]. The AS Resource Identifier Delegation extension MUST include one or more AS numbers, and the "inherit" element MUST NOT be specified.

3.2. BGPsec Router Certificate Request Profile

Refer to section 6 of [RFC6487]. The only differences between this profile and the profile in [RFC6487] are:

- The Basic Constraints extension:

  If included, the CA MUST NOT honor the cA boolean if set to TRUE.

- The Extended Key Usage extension:

  If included, id-kp-bgpsec-router MUST be present (see Section 3.1). If included, the CA MUST honor the request for id-kp-bgpsec-router.

- The Subject Information Access extension:

  If included, the CA MUST NOT honor the request to include the extension.

- The SubjectPublicKeyInfo field is specified in [ID.sidr-bgpsec-algs].

- The request is signed with the algorithms specified in [ID.sidr-bgpsec-algs].

3.3. BGPsec Router Certificate Validation

The validation procedure used for BGPsec Router Certificates is identical to the validation procedure described in Section 7 of [RFC6487] (and any RFC that updates that procedure), as modified below. For example, in step 3: "The certificate contains all fields that MUST be present" - refers to the fields that are required by this specification.

The differences are as follows:

- BGPsec Router Certificates MUST include the BGPsec Router EKU defined in Section 3.1.3.2.

- BGPsec Router Certificates MUST NOT include the SIA extension.

- BGPsec Router Certificates MUST NOT include the IP Resource extension.

- BGPsec Router Certificates MUST include the AS Resource Identifier Delegation extension.

- BGPsec Router Certificate MUST include the subjectPublicKeyInfo described in [ID.sidr-bgpsec-algs].

NOTE: BGPsec RPs will need to support the algorithms in [ID.sidr-bgpsec-algs], which are used to validate BGPsec signatures, as well as the algorithms in [RFC7935], which are needed to validate signatures on BGPsec certificates, RPKI CA certificates, and RPKI CRLs.

3.4. Router Certificates and Signing Functions in the RPKI

As described in Section 1, the primary function of BGPsec route certificates in the RPKI is for use in the context of certification of Autonomous System (AS) paths in the BGPsec protocol.

The private key associated with a router EE certificate may be used multiple times in generating signatures in multiple instances of the BGPsec_Path Attribute Signature Segments [ID.sidr-bgpsec-protocol]. I.e., the BGPsec router certificate is used to validate multiple signatures.

BGPsec router certificates are stored in the issuing CA's repository, where a repository following RFC6481 MUST use a .cer filename extension for the certificate file.

4. Design Notes

The BGPsec Router Certificate profile is based on the Resource Certificate profile as specified in [RFC7935]. As a result, many of the design choices herein are a reflection of the design choices that were taken in that prior work. The reader is referred to [RFC6484] for a fuller discussion of those choices.

CAs are required by the Certificate Policy (CP) [RFC6484] to issue properly formed BGPsec Router Certificates regardless of what is present in the certification request so there is some flexibility permitted in the certificate requests:

- o BGPsec Router Certificates are always EE certificates; therefore, requests to issue a CA certificate result in EE certificates;
- o BGPsec Router Certificates are always EE certificates; therefore, requests for Key Usage extension values keyCertSign and cRLSign result in certificates with neither of these values;
- o BGPsec Router Certificates always include the BGPsec Rouer EKU value; therefore, request without the value result in certificates with the value; and,
- o BGPsec Router Certificates never include the Subject Information Access extension; therefore, request with this extension result in certificates without the extension.

Note that this behavior is similar to the CA including the AS Resource Identifier Delegation extension in issued BGPsec Router Certificates despite the fact it is not present in the request.

5. Implementation Considerations

This document permits the operator to include a list of ASNs in a BGPsec Router Certificate. In that case, the router certificate would become invalid if any one of the ASNs is removed from any superior CA certificate along the path to a trust anchor. Operators could choose

to avoid this possibility by issuing a separate BGPsec Router Certificate for each distinct ASN, so that the router certificates for ASNs that are retained in the superior CA certificate would remain valid.

6. Security Considerations

The Security Considerations of [RFC6487] apply.

A BGPsec Router Certificate will fail RPKI validation, as defined in [RFC6487], because the cryptographic algorithms used are different. Consequently, a RP needs to identify the EKU to determine the appropriate Validation constraint.

A BGPsec Router Certificate is an extension of the RPKI [RFC6480] to encompass routers. It is a building block BGPsec and is used to validate signatures on BGPsec Signature-Segment origination of Signed-Path segments [ID.sidr-bgpsec-protocol]. Thus its essential security function is the secure binding of one or more AS numbers to a public key, consistent with the RPKI allocation/assignment hierarchy.

Hash functions [ID.sidr-bgpsec-algs] are used when generating the two key identifier extensions (i.e., Subject Key Identifier and Issuer Key Identifier) included in BGPsec certificates. However as noted in [RFC6818], collision resistance is not a required property of one-way hash functions when used to generate key identifiers. Regardless, hash collisions are unlikely, but they are possible and if detected an operator should be alerted. A subject key identifier collision might cause the incorrect certificate to be selected from the cache, resulting in a failed signature validation.

7. IANA Considerations

This document makes use of two object identifiers in the SMI Registry for PKIX. One is for the ASN.1 module in Appendix A and it comes from the SMI Security for PKIX Module Identifier IANA registry (id-mod-bgpsec-eku). The other is for the BGPsec router EKU defined in Section 3.1.3.2 and Appendix A and it comes from the SMI Security for PKIX Extended Key Purpose IANA registry. These OIDs were assigned before management of the PKIX Arc was handed to IANA. No IANA allocations are request of IANA, but please update the references in those registries when this document is published by the RFC editor.

8. Acknowledgements

We would like to thank Geoff Huston, George Michaelson, and Robert Loomans for their work on [RFC6487], which this work is based on. In

addition, the efforts of Matt Lepinski were instrumental in preparing this work. Additionally, we'd like to thank Rob Austein, Roque Gagliano, Richard Hansen, Geoff Huston, David Mandelberg, Sandra Murphy, and Sam Weiller for their reviews and comments.

9. References

9.1. Normative References

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

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <http://www.rfc-editor.org/info/rfc3779>.

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

[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, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

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

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

[RFC7935] Huston, G. and G. Michaelson, Ed., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure", RFC 7935, DOI 10.17487/RFC7935, August 2016, <http://www.rfc-editor.org/info/rfc7935>.

[ID.sidr-bgpsec-protocol] Lepinski, M. and K. Sriram, "BGPsec Protocol Specification", draft-ietf-sidr-bgpsec-protocol, work-in-progress.

[ID.sidr-bgpsec-algs] Turner, S., "BGP Algorithms, Key Formats, & Signature Formats", draft-ietf-sidr-bgpsec-algs, work-in-progress.

9.2. Informative References

[RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <http://www.rfc-editor.org/info/rfc4272>.

[RFC5123] White, R. and B. Akyol, "Considerations in Validating the Path in BGP", RFC 5123, DOI 10.17487/RFC5123, February 2008, <http://www.rfc-editor.org/info/rfc5123>.

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

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

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

[RFC6484] Kent, S., Kong, D., Seo, K., and R. Watro, "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, DOI 10.17487/RFC6484, February 2012, <http://www.rfc-editor.org/info/rfc6484>.

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

[RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, DOI 10.17487/RFC6916, April 2013, <http://www.rfc-editor.org/info/rfc6916>.

Appendix A. ASN.1 Module

```
BGPSECEKU { iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-bgpsec-eku(84) }

  DEFINITIONS EXPLICIT TAGS ::=

  BEGIN

  -- EXPORTS ALL --

  -- IMPORTS NOTHING --

  -- OID Arc --

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

  -- BGPsec Router Extended Key Usage --

  id-kp-bgpsec-router OBJECT IDENTIFIER ::= { id-kp 30 }

  END
```

Authors' Addresses

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

Email: mcr@islandpeaksoftware.com

Sean Turner
sn3rd

EMail: sean@sn3rd.com

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

Email: kent@bbn.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

M. Lepinski, Ed.
BBN

July 4, 2014

BGPSEC Protocol Specification
draft-ietf-sidr-bgpsec-protocol-09

Abstract

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

Requirements Language

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

Status of this Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

This document describes BGPSEC, a mechanism for providing path

security for Border Gateway Protocol (BGP) [2] route advertisements. That is, a BGP speaker who receives a valid BGPSEC update has cryptographic assurance that the advertised route has the following two properties:

1. The route was originated by an AS 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 listed in the update message has explicitly authorized the advertisement of the route to the subsequent AS in the path.

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

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

2. BGPSEC Negotiation

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

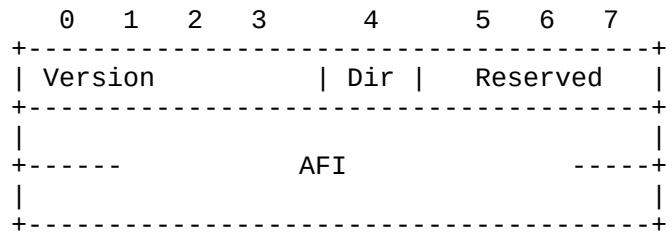
2.1. The BGPSEC Capability

This capability has capability code : TBD

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

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

BGPSEC Send Capability Value:



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

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

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

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

2.2. Negotiating BGPSEC Support

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

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

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

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

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

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

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

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

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

3. The BGPSEC_Path Attribute

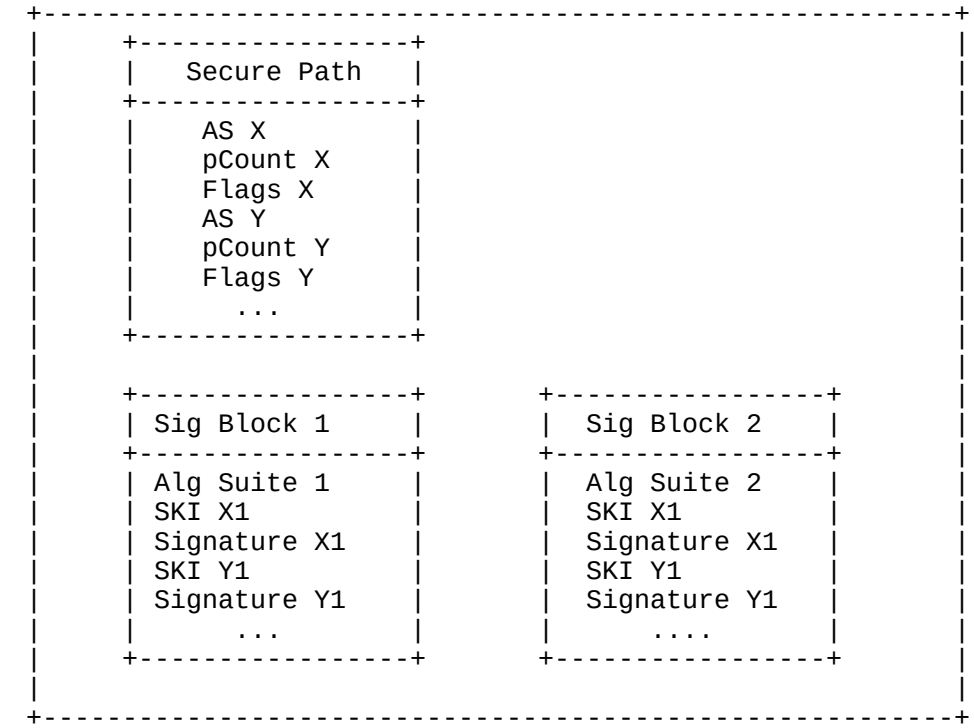
The BGPSEC_Path attribute is a new optional non-transitive BGP path attribute.

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

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

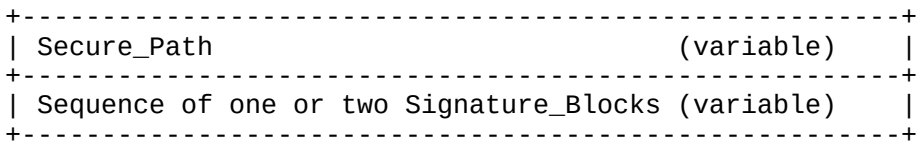
The BGPSEC_Path attribute is made up of several parts. The following high-level diagram provides an overview of the structure of the BGPSEC_Path attribute:

High-Level Diagram of the BGPSEC_Path Attribute



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

BGPSEC_Path Attribute



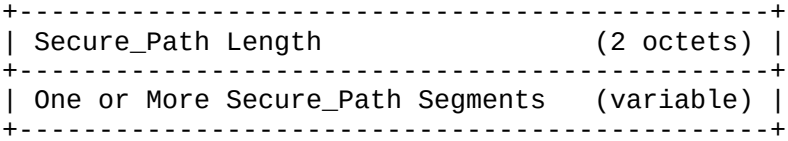
The Secure_Path contains AS path information for the BGPSEC update message. This is logically equivalent to the information that is contained in a non-BGPSEC AS_PATH attribute. A BGPSEC update message containing the BGPSEC_Path attribute MUST NOT contain the AS_PATH attribute. The Secure_Path is used 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

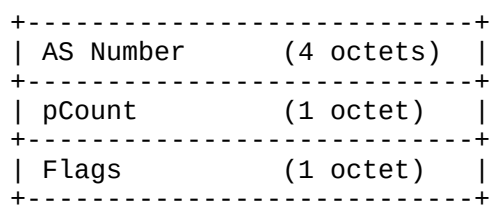


The Secure_Path Length contains the length (in octets) of the entire Secure_Path (including the two octets used to express this length field). As explained below, each Secure_Path segment is six octets long. Note that this means the Secure_Path Length is two greater

than six times the number Secure_Path Segments (i.e., the number of AS numbers in the path).

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

Secure_Path Segment



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

The pCount field contains the number of repetitions of the associated autonomous system number that the signature covers. This field enables a BGPSEC speaker to mimic the semantics of prepending multiple copies of their AS to the AS_PATH without requiring the speaker to generate multiple signatures.

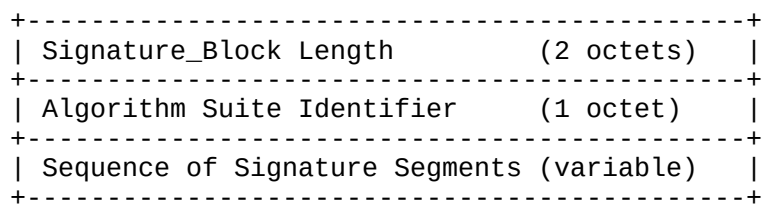
The first bit of the Flags field is the Confed_Segment flag. The Confed_Segment flag is set to one to indicate that the BGPSEC speaker that constructed this Secure_Path segment is sending the update message to a peer AS within the same Autonomous System confederation [5]. (That is, the Confed_Segment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

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

3.2. Signature_Block

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

Signature_Block

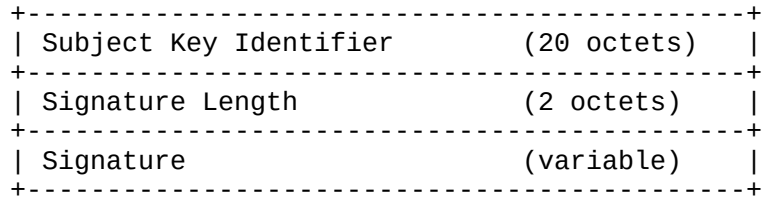


The Signature_Block Length is the total number of octets in the Signature_Block (including the two octets used to express this length field).

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

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

Signature Segments



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

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

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

4. Generating a BGPSEC Update

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

The remaining case is where the BGPSEC speaker sends the update message to an internal (iBGP) peer. When originating a new route advertisement and sending it to an internal peer, the BGPSEC speaker 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.)

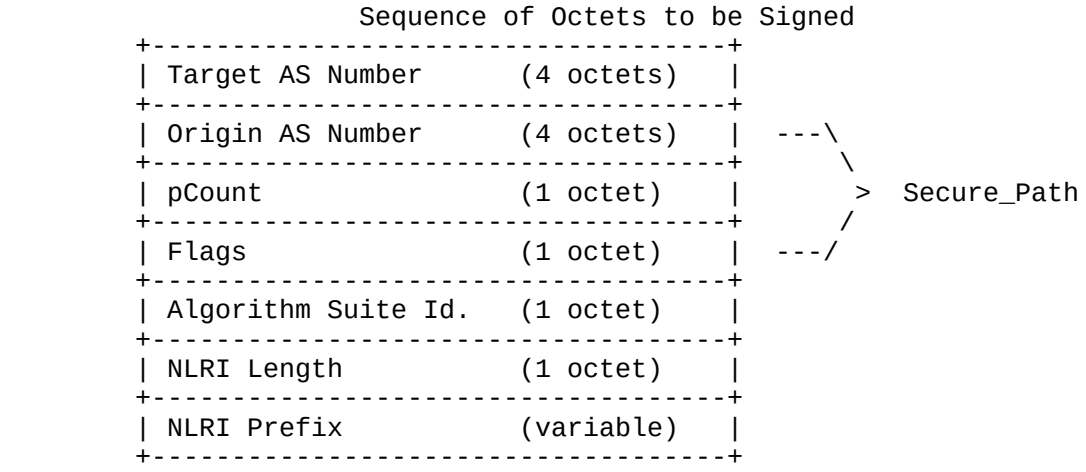
Typically, a BGPSEC speaker will use only a single algorithm suite, and thus create only a single Signature_Block in the BGPSEC_Path attribute. However, to ensure backwards compatibility during a period of transition from a 'current' algorithm suite to a 'new' algorithm suite, it will be necessary to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

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



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

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

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

4.2. Propagating a Route Advertisement

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

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

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

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

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

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

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

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

The pCount is typically set to the value 1. A BGPSEC speaker may set the pCount field to a value greater than 1. (See Section 4.1 for a discussion of setting pCount to a value greater than 1.) A route server that participates in the BGP control path, but does not act as a transit AS in the data plane, may choose to set pCount to 0. This option enables the route server to participate in BGPSEC and obtain the associated security guarantees without increasing the effective length of the AS path. (Note that BGPSEC speakers compute the effective length of the AS path by summing the pCount values in the BGPSEC_Path attribute, see Section 5.) However, when a route server

sets the pCount value to 0, it still inserts its AS number into the Secure_Path segment, as this information is needed to validate the signature added by the route server. 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 the BGPSEC_Path has two Signature_Blocks (corresponding to different algorithm suites), the validation algorithm (see Section 5.2) deems a BGPSEC update message to be 'Valid' if there is at least one supported algorithm suite (and corresponding Signature_Block) that is deemed 'Valid'. This means that a 'Valid' BGPSEC update message may contain a Signature_Block which is not deemed 'Valid' (e.g., contains signatures that the BGPSEC does not successfully verify). Nonetheless, such Signature_Blocks MUST NOT be removed. (See Section 7 for a discussion of the security ramifications of this design choice.)

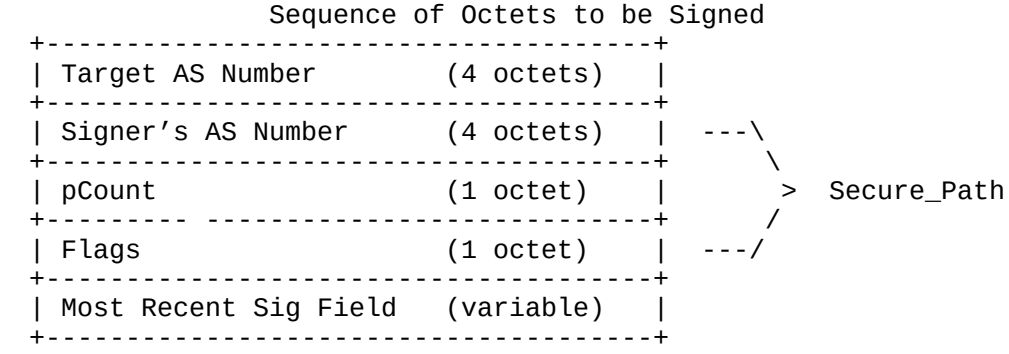
For each Signature_Block corresponding to an algorithm suite that the BGPSEC speaker does support, the BGPSEC speaker adds a new Signature Segment to the Signature_Block. This Signature Segment is prepended to the list of Signature Segments (placed in the first position) so that the list of Signature Segments appear in the same order as the

corresponding Secure_Path segments. The BGPSEC speaker populates the fields of this new signature segment as follows.

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

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

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



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

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

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

4.3. Processing Instructions for Confederation Members

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

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

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

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

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

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

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

When validating a received BGPSEC update message, confederation members need to make the following adjustment to the algorithm presented in Section 5.2. When a confederation member processes (validates) a Signature Segment and its corresponding Secure_Path Segment, the confederation member must note 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, the confederation member, during processing of a Signature_Segment, first checks whether the Confed_Sequence flag in the corresponding Secure_Path segment is set to one. If the Confed_Sequence flag is set to one in the corresponding Secure_Path segment, the confederation member does not perform any further checks on the Signature_Segment and immediately moves on to the next Signature_Segment (and checks its corresponding Secure_Path segment). Note that as specified in Section 5.2, it is an error when a BGPSEC speaker receives from a peer, who is not in the same AS confederation, a BGPSEC update containing a Confed_Sequence flag set to one. (As discussed in Section 5.2, any error in the BGPSEC_Path

attribute MUST be handled using the "treat-as-withdraw", approach as defined in RFC WXYZ [12].)

4.4. Reconstructing the AS_PATH Attribute

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

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

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

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

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

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

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

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

5. Processing a Received BGPSEC Update

Upon receiving a BGPSEC update message from an external (eBGP) peer, a BGPSEC speaker SHOULD validate the message to determine the authenticity of the path information contained in the BGPSEC_Path attribute. Section 5.1 provides an overview of BGPSEC validation and Section 5.2 provides a specific algorithm for performing such validation. (Note that an implementation need not follow the specific algorithm in Section 5.2 as long as the input/output behavior of the validation is identical to that of the algorithm in Section 5.2.) During exceptional conditions (e.g., the BGPSEC speaker receives an incredibly large number of update messages at once) a BGPSEC speaker MAY temporarily defer validation of incoming BGPSEC update messages. The treatment of such BGPSEC update messages, whose validation has been deferred, is a matter of local policy.

The validity of BGPSEC update messages is a function of the current RPKI state. When a BGPSEC speaker learns that RPKI state has changed (e.g., from an RPKI validating cache via the RTR protocol), the BGPSEC speaker MUST re-run validation on all affected update messages stored in its ADJ-RIB-IN. That is, when a given RPKI certificate ceases to be valid (e.g., it expires or is revoked), all update

messages containing a signature whose SKI matches the SKI in the given certificate must be re-assessed to determine if they are still valid. If this reassessment determines that the validity state of an update has changed then, depending on local policy, it may be necessary to re-run best path selection.

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

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

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

5.1. Overview of BGPSEC Validation

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

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

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

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

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

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

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

choose to perform its own validation.

5.2. Validation Algorithm

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

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

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

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

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

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

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

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

Next, the BGPSEC speaker 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.

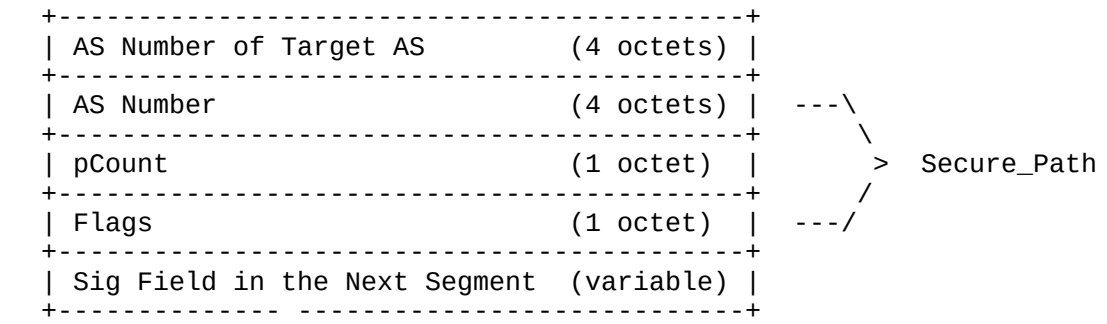
Finally, the BGPSEC speaker examines the Signature_Blocks in the BGPSEC_Path attribute. A Signature_Block corresponding to an algorithm suite that the BGPSEC speaker does not support is not considered in validation. If there is no Signature_Block corresponding to an algorithm suite that the BGPSEC speaker supports, then the BGPSEC speaker MUST treat the update message in the same manner that the BGPSEC speaker would treat an (unsigned) update message that arrived without a BGPSEC_Path attribute.

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

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

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

Sequence of Octets to be Hashed

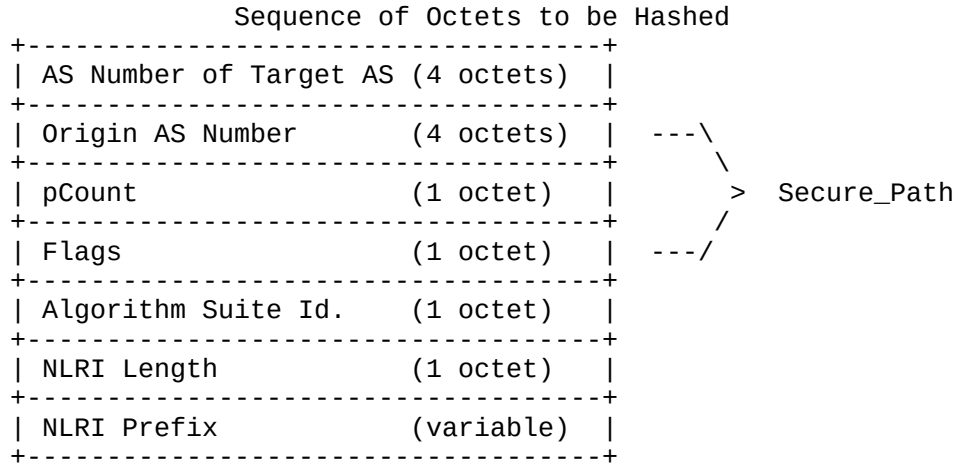


For the first segment to be processed (the most recently added segment), the 'AS Number of Target AS' is the AS number of the BGPSEC speaker validating the update message. Note that if a BGPSEC speaker uses multiple AS Numbers (e.g., the BGPSEC speaker is a member of a confederation), the AS number used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

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

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

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



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

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

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

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

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

6. Algorithms and Extensibility

6.1. Algorithm Suite Considerations

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

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

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

6.2. Extensibility Considerations

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

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

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

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

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

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

7. Security Considerations

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

7.1 Security Guarantees

A BGPSEC speaker who receives a valid BGPSEC update message, containing a route advertisement for a given prefix, is provided with the following security guarantees:

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

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

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

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

7.2 On the Removal of BGPSEC Signatures

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

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

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

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

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

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

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

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

7.3 Mitigation of Denial of Service Attacks

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

7.4 Additional Security Considerations

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

BGPSEC does not provide protection against attacks at the transport layer. 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
mlepinski.ietf@gmail.com

Stephen Kent
BBN Technologies
kent@bbn.com

Warren Kumari
Google
warren@kumari.net

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

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

Samuel Weiler
Sparta
weiler+ietf@watson.org

9.2. Acknowledgements

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

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

10. Normative References

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

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

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

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

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

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

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

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

[9] Patel, K., Ward, D., and R. Bush, "Extended Message support for BGP", draft-ietf-idr-bgp-extended-messages (work in progress), January 2014.

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

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

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

11. Informative References

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

[14] Kent, S., "Threat Model for BGP Path Security", draft-ietf-sidr-bgpsec-threats (work in progress), December 2013.

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

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

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

Author’s Address

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

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

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

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

April 27, 2017

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

1.1. Requirements Language

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

2. BGPsec Negotiation

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

2.1. The BGPsec Capability

This capability has capability code: TBD

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

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

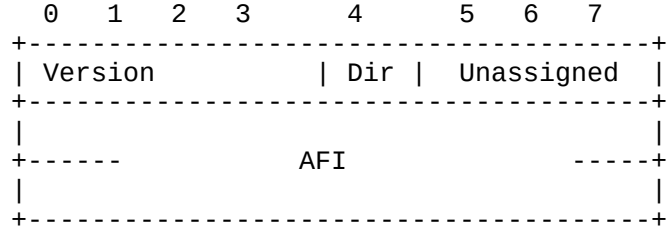


Figure 1: BGPsec Capability format.

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

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

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

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

2.2. Negotiating BGPsec Support

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

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

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

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

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

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

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

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

3. The BGPsec_Path Attribute

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

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

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

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

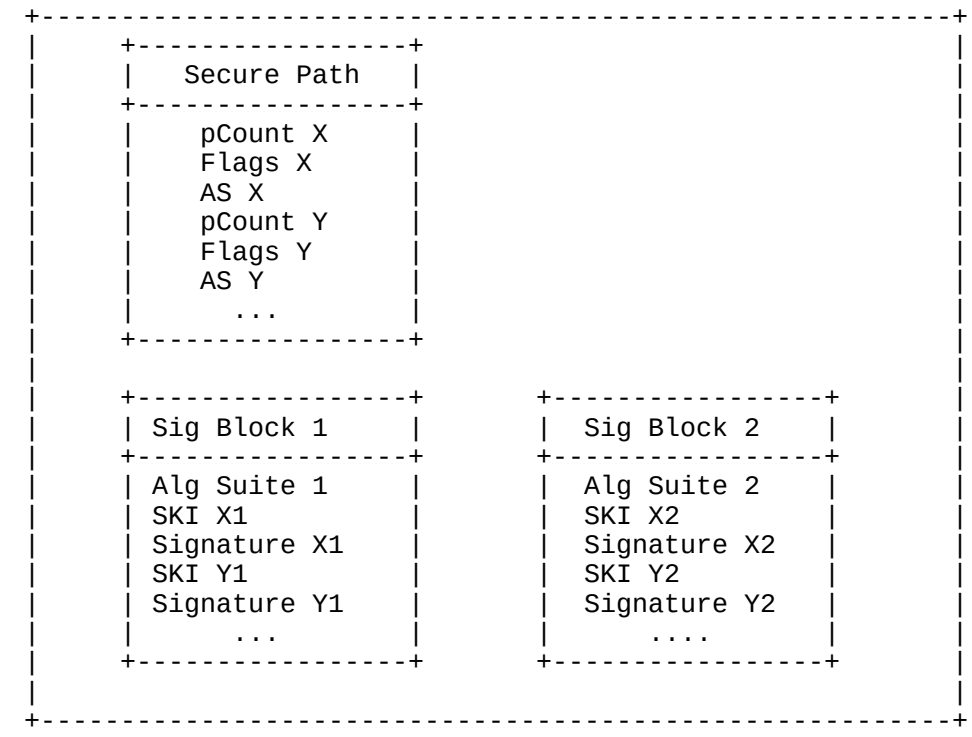


Figure 2: High-level diagram of the BGPsec_Path attribute.

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

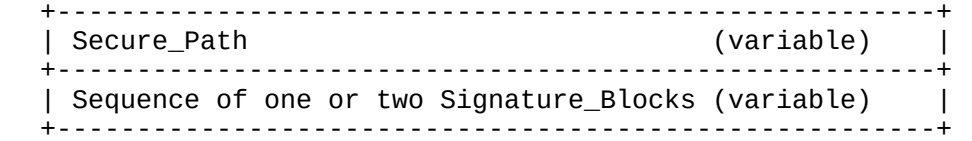


Figure 3: BGPsec_Path attribute format.

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

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

3.1. Secure_Path

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

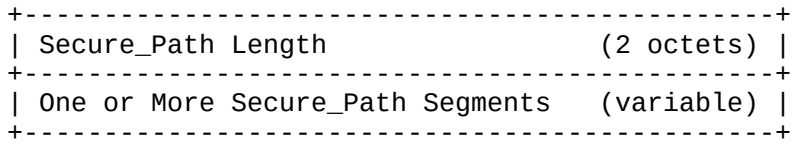


Figure 4: Secure_Path format.

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

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

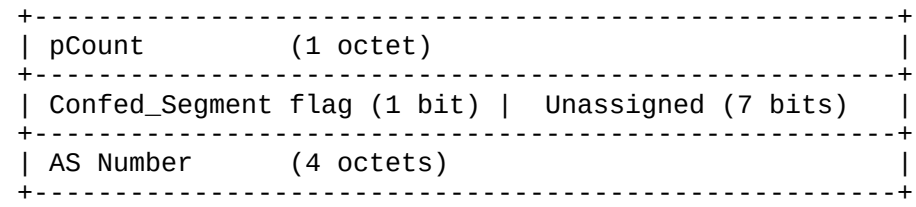


Figure 5: Secure_Path Segment format.

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

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

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

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

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

3.2. Signature_Block

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

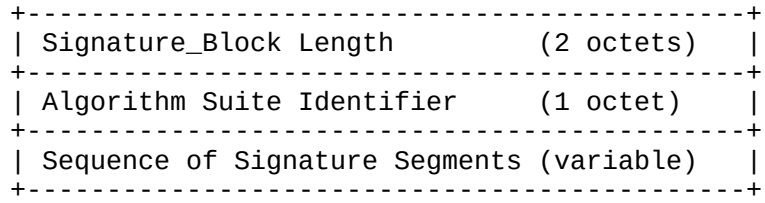


Figure 6: Signature_Block format.

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

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

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

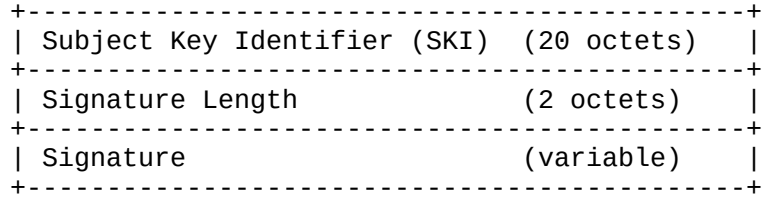


Figure 7: Signature Segment format.

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

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

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

4. BGPsec Update Messages

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

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

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

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

4.1. General Guidance

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2. Constructing the BGPsec_Path Attribute

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

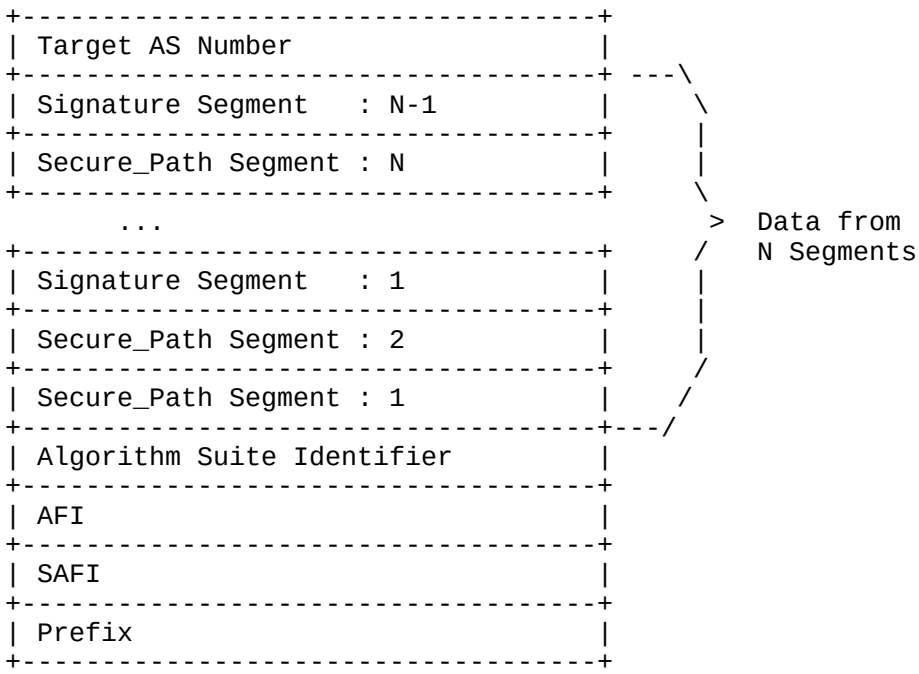


Figure 8: Sequence of octets to be hashed.

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

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

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

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

4.3. Processing Instructions for Confederation Members

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

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

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

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

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

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

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

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

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

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

4.4. Reconstructing the AS_PATH Attribute

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

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

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

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

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

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

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

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

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

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

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

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

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

5. Processing a Received BGPsec Update

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

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

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

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

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

5.1. Overview of BGPsec Validation

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

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

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

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

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

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

5.2. Validation Algorithm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

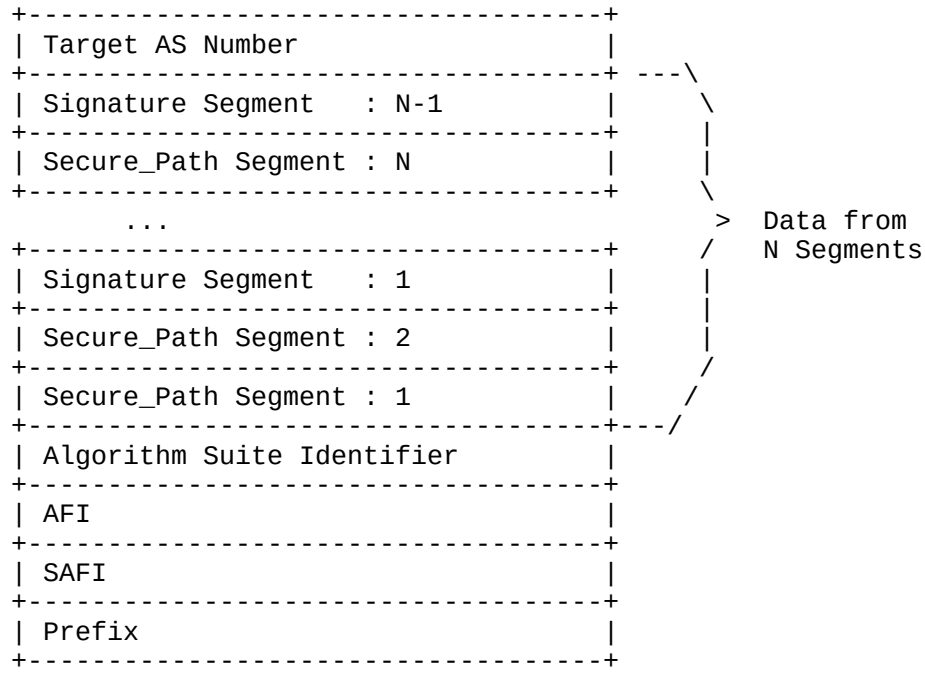


Figure 9: The Sequence of octets to be hashed for signature verification of Signature Segment N; N = 1,2, ..., K, where K is the number of AS hops in the BGPsec_Path attribute.

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

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

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

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

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

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

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

6. Algorithms and Extensibility

6.1. Algorithm Suite Considerations

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

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

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

6.2. Considerations for the SKI Size

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

6.3. Extensibility Considerations

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

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

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

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

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

7. Operations and Management Considerations

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

7.1. Capability Negotiation Failure

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

7.2. Preventing Misuse of pCount=0

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

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

7.3. Early Termination of Signature Verification

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

7.4. Non-Deterministic Signature Algorithms

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

7.5. Private AS Numbers

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

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

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

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

7.6. Robustness Considerations for Accessing RPKI Data

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

7.7. Graceful Restart

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

7.8. Robustness of Secret Random Number in ECDSA

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

7.9. Incremental/Partial Deployment Considerations

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

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

8. Security Considerations

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

8.1. Security Guarantees

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

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

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

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

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

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

8.2. On the Removal of BGPsec Signatures

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

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

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

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

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

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

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

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

8.3. Mitigation of Denial of Service Attacks

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

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

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

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

8.4. Additional Security Considerations

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

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

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

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

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

9. IANA Considerations

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

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

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

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

Figure 10: IANA registry for BGPsec Capability.

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

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

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

Figure 11: IANA registry for BGPsec_Path Flags field.

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

10. Contributors

10.1. Authors

Rob Austein
Dragon Research Labs
sra@hactrn.net

Steven Bellovin
Columbia University
smb@cs.columbia.edu

Randy Bush
Internet Initiative Japan
randy@psg.com

Russ Housley
Vigil Security
housley@vigilsec.com

Matt Lepinski
New College of Florida
mlepinski@ncf.edu

Stephen Kent
BBN Technologies
kent@bbn.com

Warren Kumari

Google
warren@kumari.net

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

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

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

10.2. Acknowledgements

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

11. References

11.1. Normative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11.2. Informative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

Email: mlepinski@ncf.edu

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

Email: kotikalapudi.sriram@nist.gov

Secure Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: October 2, 2013

M. Reynolds
IPSw
S. Kent
BBN
M. Lepinski
BBN

Apr 5, 2013

Local Trust Anchor Management for the Resource Public Key Infrastructure <draft-ietf-sidr-ltamgmt-08.txt>

Status of this Memo

Copyright and License Notice

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 extentions 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

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 sem but the collecting of certificates to populate a local cache. Thus, the common, bottom-up path discovery approach is not inconsistent with these RFCs. Morevoer, 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 constraings 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 comsumers of constraints files.

The constraints file consists of four logical subsections: the replying 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
Xcrldp
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 1are 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 Xcrldp 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 (state 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 in 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 are 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 descrution 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 ecurity 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, Feburary 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 C. 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   Xcrldp       rsync://tbo_lta_test.com/pub/CRLs
TAG   Xaia         rsync://tbo_lta_test.com/pub/repos

;
; Block subsection
;
```

```
;
; First block: TBO Corporate
;

; Resource Holder: TBO Corporation

SKI 00112233445566778899998877665544332211
  IPv4
    10.2.3/24
    10.8/16
  IPv6
    2001:db8::/32
  AS#
    60123
    5507

;
; Second block: TBO LTA Test Enforcement Division
;

; Resource Holder: TBO Corporation

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.
;

; Resource Holder: TBO Acceptance Corporation

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
Intended status: Standards Track
Expires: August 18, 2014

P. Mohapatra
Sproute Networks
K. Patel
Cisco Systems
J. Scudder
Juniper Networks
D. Ward
Cisco Systems
R. Bush
Internet Initiative Japan, Inc.
February 14, 2014

BGP Prefix Origin Validation State Extended Community
draft-ietf-sidr-origin-validation-signaling-04

Abstract

As part of the origination AS validation process, it can be desirable to automatically consider the validation state of routes in the BGP decision process. The purpose of this document is to provide a specification for doing so. The document also defines a new BGP opaque extended community to carry the validation state inside an autonomous system to influence the decision process of the IBGP speakers.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 18, 2014.

Table of Contents

1. Introduction

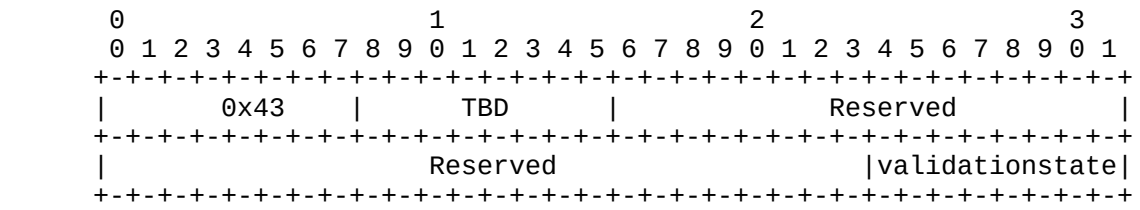
As part of the origination AS validation process, it can be desirable to automatically consider the validation state of routes in the BGP decision process. The purpose of this document is to provide a specification for doing so. The document defines a new BGP opaque extended community to carry the validation state inside an autonomous system to influence the decision process of the IBGP speakers.

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. Origin Validation State Extended Community

The origin validation state extended community is an opaque extended community [RFC4360] with the following encoding:



The value of the high-order octet of the extended Type Field is 0x43, which indicates it is non-transitive. The value of the low-order octet of the extended type field for this community is TBD. The last octet of the extended community encodes the route's validation state[RFC6811]. It can assume the following values:

Value	Meaning
0	Lookup result = "valid"
1	Lookup result = "not found"
2	Lookup result = "invalid"

If the router is configured to support the extensions defined in this draft, it SHOULD attach the origin validation state extended community to BGP UPDATE messages sent to IBGP peers by mapping the computed validation state in the last octet of the extended community. Similarly on the receiving IBGP speakers, the validation state of an IBGP route SHOULD be derived directly from the last octet of the extended community, if present.

3. Changes to the BGP Decision Process

If a BGP router supports prefix origin validation and is configured for the extensions defined in this document, the validation step SHOULD be performed prior to any of the steps defined in the decision process of [RFC4271]. The validation step is stated as follows:

> When comparing a pair of routes for a BGP destination, the route with the lowest "validation state" value is preferred.

In all other respects, the decision process remains unchanged.

3.1. Policy Control

It MUST be possible to enable or disable the validation step as defined in Section 3 through configuration. The default SHOULD be for the validation step to be disabled.

4. Deployment Considerations

In deployment scenarios where not all the speakers in an autonomous system are upgraded to support the extensions defined in this document, it is necessary to define policies that match on the origin validation extended community and set another BGP attribute [RFC6811] that influences the best path selection the same way as what would have been enabled by an implementation of this extension.

5. Acknowledgements

The authors would like to acknowledge the valuable review and suggestions from Wesley George and Roque Gagliano on this document.

6. IANA Considerations

IANA shall assign a new value from the "BGP Opaque Extended Community" type registry from the non-transitive range, to be called "BGP Origin Validation State Extended Community".

7. Security Considerations

This document introduces no new security concerns beyond what is described in [RFC6811].

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.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.

8.2. Informational References

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, January 2013.

Authors' Addresses

Pradosh Mohapatra
Sproute Networks

Email: mpradosh@yahoo.com

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

Email: keyupate@cisco.com

John Scudder
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
USA

Email: jgs@juniper.net

David Ward
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: dward@cisco.com

Randy Bush
Internet Initiative Japan, Inc.
5147 Crystal Springs
Bainbridge Island, Washington 98110
USA

Email: randy@psg.com

SIDR
Internet-Draft
Intended status: Standards Track
Expires: July 14, 2017

P. Mohapatra
Sproute Networks
K. Patel
Cisco
J. Scudder
Juniper Networks
D. Ward
Cisco
R. Bush
Internet Initiative Japan, Inc.
January 10, 2017

BGP Prefix Origin Validation State Extended Community draft-ietf-sidr-origin-validation-signaling-11

Abstract

This document defines a new BGP opaque extended community to carry the origination AS validation state inside an autonomous system. IBGP speakers that receive this validation state can configure local policies allowing it to influence their decision process.

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

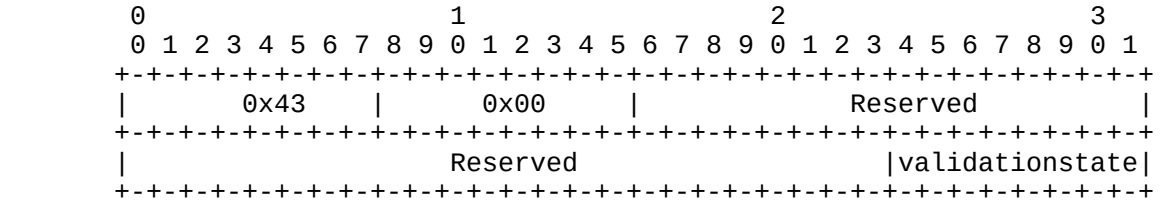
This document defines a new BGP opaque extended community to carry the origination AS validation state inside an autonomous system. IBGP speakers that receive this validation state can configure local policies allowing it to influence their decision process.

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. Origin Validation State Extended Community

The origin validation state extended community is an opaque extended community [RFC4360] with the following encoding:



The value of the high-order octet of the extended Type Field is 0x43, which indicates it is non-transitive. The value of the low-order octet of the extended type field as assigned by IANA is 0x00. The Reserved field MUST be set to 0 and ignored upon the receipt of this community. The last octet of the extended community is an unsigned integer that gives the route's validation state [RFC6811]. It can assume the following values:

Value	Meaning
0	Lookup result = "valid"
1	Lookup result = "not found"
2	Lookup result = "invalid"

If the router is configured to support the extensions defined in this draft, it SHOULD attach the origin validation state extended community to BGP UPDATE messages sent to IBGP peers by mapping the computed validation state in the last octet of the extended community. Similarly on the receiving IBGP speakers, the validation state of an IBGP route SHOULD be derived directly from the last octet of the extended community, if present.

An implementation SHOULD NOT send more than one instance of the origin validation state extended community. However, if more than one instance is received, an implementation MUST disregard all instances other than the one with the numerically-greatest validation state value. If the value received is greater than the largest specified value (2), the implementation MUST apply a strategy similar to attribute discard [RFC7606] by discarding the erroneous community and logging the error for further analysis.

By default, implementations MUST drop the origin validation state extended community if received from an EBGP peer, without further processing it. Similarly, by default an implementation SHOULD NOT send the community to EBGP peers. However it SHOULD be possible to configure an implementation to send or accept the community when warranted. An example of a case where the community would reasonably be received from, or sent to, an EBGP peer is when two adjacent ASes are under control of the same administration. A second example is documented in [I-D.ietf-sidr-route-server-rpki-light].

3. Deployment Considerations

In deployment scenarios where not all the speakers in an autonomous system are upgraded to support the extensions defined in this document, it is necessary to define policies that match on the origin

validation extended community and set another BGP attribute [RFC6811] that influences the best path selection the same way as what would have been enabled by an implementation of this extension.

4. Acknowledgements

The authors would like to acknowledge the valuable review and suggestions from Wesley George, Roque Gagliano and Bruno Decraene on this document.

5. IANA Considerations

IANA has assigned the value 0x00 from the "Non-Transitive Opaque Extended Community Sub-Types" registry. The value is called "BGP Origin Validation State Extended Community".

6. Security Considerations

Security considerations such as those described in [RFC4272] continue to apply. Since this document introduces an extended community that will generally be used to affect route selection, the analysis in Section 4.5 ("Falsification") of [RFC4593] is relevant. These issues are neither new, nor unique to the origin validation extended community.

The security considerations provided in [RFC6811] apply equally to this application of origin validation. In addition, this document describes a scheme where router A outsources validation to some router B. If this scheme is used, the participating routers should have the appropriate trust relationship -- B should trust A either because they are under the same administrative control or for some other reason (for example, consider [I-D.ietf-sidr-route-server-rpki-light]). The security properties of the propagation path between the two routers should also be considered. See [RFC7454] Section 5.1 for advice regarding protection of the propagation path.

7. References

7.1. Normative References

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

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

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <http://www.rfc-editor.org/info/rfc4360>.

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

7.2. Informative References

[I-D.ietf-sidr-route-server-rpki-light] King, T., Kopp, D., Lambrianidis, A., and A. Fenioux, "Signaling Prefix Origin Validation Results from a Route-Server to Peers", draft-ietf-sidr-route-server-rpki-light-01 (work in progress), December 2016.

[RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <http://www.rfc-editor.org/info/rfc4272>.

[RFC4593] Barbir, A., Murphy, S., and Y. Yang, "Generic Threats to Routing Protocols", RFC 4593, DOI 10.17487/RFC4593, October 2006, <http://www.rfc-editor.org/info/rfc4593>.

[RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <http://www.rfc-editor.org/info/rfc7454>.

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

Authors' Addresses

Pradosh Mohapatra
Sproute Networks

Email: mpradosh@yahoo.com

Keyur Patel
Cisco
170 W. Tasman Drive
San Jose, CA 95124

Email: keyupate@cisco.com

John Scudder
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089

Email: jgs@juniper.net

Dave Ward
Cisco
170 W. Tasman Drive
San Jose, CA 95124

Email: dward@cisco.com

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

Email: randy@psg.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2015

G. Huston
G. Michaelson
APNIC
C. Martinez
LACNIC
T. Bruijnzeels
RIPE NCC
A. Newton
ARIN
A. Aina
AFRINIC

July 2, 2014

RPKI Validation Reconsidered
draft-ietf-sidr-rpki-validation-reconsidered-00.txt

Abstract

This document reviews the certificate validation procedure specified in RFC6487 and highlights aspects of potentially acute operational fragility in the management of certificates in the RPKI in response to the movement of resources across registries, and the associated actions of Certification Authorities to maintain continuity of validation of certification of resources during this movement.

Table of Contents

1. Introduction

This document reviews the certificate validation procedure specified in RFC6487 and highlights aspects of potentially acute operational fragility in the management of certificates in the RPKI in response to the movement of resources across registries, and the associated actions of Certification Authorities to maintain continuity of validation of certification of resources during this movement.

2. Certificate Validation in the RPKI

As currently defined in section 7.2 of [RFC6487], validation of PKIX certificates that conform to the RPKI profile relies on the use of a path validation process where each certificate in the validation path is required to meet the certificate validation criteria. This can be considered to be a recursive validation process where, in the context of an ordered sequence of certificates, as defined by each pair of certificates in this sequence having a common Issuer and Subject Name respectively, a certificate is defined as valid if it satisfies basic validation criteria relating to the syntactic correctness, currency of validity dates and similar properties of the certificate itself, as described in [RFC5280], and also that it satisfies certain additional criteria with respect to the previous certificate in the sequence (the Issuer part of the pair), and that this previous certificate is itself a valid certificate using the same criteria. This definition applies recursively to all certificates in the sequence apart from the initial sequence element, which is required to be a Trust Anchor.

For RPKI certificates, the additional criteria relating to the previous certificate in this sequence is that the certificate’s number resource set, as defined in [RFC3779], is "encompassed" by the number resource set contained in the previous certificate.

Because [RFC6487] validation demands that all resources in a certificate be valid under the parent (and recursively, to the root), a digitally signed attestation, such as a Route Origin Authorization (ROA) object [RFC6482], which refers only to a subset of RFC3779-specified resources from that certificate validation chain can be concluded to be invalid, but not by virtue of the relationship between the RFC3779 extensions of the certificates on the putative certificate validation path and the resources in the ROA, but by other resources described in these certificates where the "encompassing" relationship of the resources does not hold. Any such invalidity along the certificate validation path can cause this outcome, not just at the immediate parent of the end entity certificate that attests to the key used to sign the ROA.

For example, in the certificate sequence:

Certificate 1:
Issuer A, Subject B, Resources 192.0.2.0/24, AS64496-AS64500

Certificate 2:
Issuer B, Subject C, Resources 192.0.2.0/24/24, AS64496-AS64511

Certificate 3:
Issuer C, Subject D, Resources 192.0.2.0/24

Certificate 3 is considered to be an invalid certificate, because the resources in Certificate 2 are not encompassed by the resources in Certificate 1, by virtue of certificate 2 describing the resources of the range AS64501 - AS64511 in this RFC3779 resource extension. Obviously, these Autonomous Systems numbers are not related to the IPv4 resources contained in Certificate 3.

Any non-encompassed resource set can cause invalidation, be it an ASN, IPv4 or IPv6 resource, if it is not encompassed by the resource set in the parent (Issuer) certificate.

The underlying observation here is that this definition of certificate validation treats a collection of resources as inseparable, so that a single certificate containing a bundle of number resources is semantically distinct from an equivalent set of certificates where each certificate contains a single number resource. This semantic distinction between the whole and the sum of its parts is an artifice introduced by the particular choice of a certificate validation procedure, as distinct from meeting any particular operational requirement, and the result is the introduction of operational fragility into the handling of RPKI certificates, particularly in the case where number resources are moved between the corresponding registries, as described here.

3. Operational Considerations

There are two areas of operational concern with the current RPKI validation definition.

The first is that of the robustness of the operational management procedures in the issuance of certificates. If a subordinate Certification Authority (CA) issues a certificate that contains an Internet Number Resource (INR) collection that is not either exactly equal to, or a strict subset of, its parent CA, then this issued certificate, and all subordinate certificates of this issued certificate are invalid. These certificates are not only defined as

invalid when being considered to validate an INR that is not in the parent CA certificate, but are defined as invalid for all INRs in the certificate.

This constraint creates a degree of operational fragility in the issuance of certificates, as all CA's are now required to exercise extreme care in the issuance and reissuance of certificates to ensure that at no time do they overclaim on the resources described in the parent CA, as the consequences of an operational lapse or oversight implies that all the subordinate certificates from the point of INR mismatch are invalid. It would be preferred if the consequences of such an operational lapse were limited in scope to the specific INRs that formed the mismatch, rather than including the entire set of INRs within the scope of damage from this point of mismatch downward across the entire sub-tree of descendant certificates in the RPKI certificate hierarchy.

The second operational consideration described here relates to the situation where a registry withdraws a resource from the current holder, and the resource to transferred to another registry, to be registered to a new holder in that registry. The reason why this is a consideration in operational deployments of the RPKI lies in the movement of the "home" registry of number resources during cases of mergers, acquisitions, business re-alignments, and resource transfers and the desire to ensure that during this movement all other resources can continue to be validated.

If the original registry's certification actions are simply to issue a new certificate for the current holder with a reduced resource set, and to revoke the original certificate, then there is a distinct possibility of encountering the situation illustrated by the example in the previous section. This is a result of an operational process for certificate issuance by the parent CA being de-coupled from the certificate operations of child CA.

This de-coupled operation of CAs introduces a risk of unintended third party damage: since a CA certificate can refer to holdings which relate to two or more unrelated subordinate certificates, if this CA certificate becomes invalid due to the reduction in the resources allocated to this CA relating to one subordinate resource set, all other subordinate certificates are invalid until the CA certificate is reissued with a reduced resource set.

In the example provided in the previous section, all subordinate certificates issued by CA B are invalid, including all certificates issued by CA C, until CA A issues a new certificate for CA B with a reduced resource set.

At the lower levels of the RPKI hierarchy the resource sets affected by such movements of resources may not encompass significantly large pools of resources. However, as one ascends through this certification hierarchy towards the apex, the larger the resource set that is going to be affected by a period of invalidity by virtue of such uncoordinated certificate management actions. In the case of a Regional Internet Registry (RIR) or National Internet Registry (NIR), the potential risk arising from uncoordinated certification actions relating to a transfer of resources is that the entire set of subordinate certificates that refer to resources administered by the RIR or the NIR cannot be validated during this period.

Avoiding such situations requires that CA's adhere to a very specific ordering of certificate issuance. In this framework, the common registry CA that describes (directly or indirectly) the resources being shifted from one registry to the other, and also contains in subordinate certificates (direct or indirect) the certificates for both registries who are parties to the resource transfer has to coordinate a specific sequence of actions.

This common registry CA has to first issue a new certificate towards the "receiving" registry that adds to the RFC3779 extension resource set the specific resource being transferred into this receiving registry. The common registry CA then has to wait until all registries in the subordinate certificate chain to the receiving registry have also performed a similar issuance of new certificates, and in each case a registry must await the issuance of the immediate superior certificate with the augmented resource set before it, in turn, can issue its own augmented certificate to its subordinate CA. This is a "top down" issuance sequence."

It is possible for the common registry to issue a certificate to the "sending" registry with the reduced resource set at any time, but it should not revoke the previously issued certificate, nor overwrite this previously issued certificate in its repository publication point without specific coordination. Only when the common registry is assured that the top down certificate issuance process to the receiving registry CA chain has been completed can the common registry commence the revocation of the original certificate for the sending registry, However, it should not so until it is assured that the immediate subordinate registry CA in the path to the sending registry has issued a certificate with a reduced resource set, and so on. This implies that on the sending side the certificate issuance and revocation is a "bottom up" process.

If this process is not carefully followed, then the risk is that some or all or the subordinate certificates of this common registry CA will be unable to be validated until the entire process of

certificate issuance and revocation has been completed. While this sequenced process is intended to preserve validity of certificates in the RPKI, it is a complex, fragile and operationally cumbersome process.

The underlying consideration here is that the operational coordination of these certificate issuance and revocation actions to effect a smooth resource transfer across registries is mandated by the nature of the particular choice of certificate validation process described in [RFC6487].

4. Alternatives Approaches

If the current definition of the RPKI certificate validation procedure is considered to introduce unacceptable levels of fragility and risk into the operational environment, what alternatives exist?

One approach is to remove the semantic requirement to consider the collection of resources in the extension field of the RPKI certificate as an indivisible bundle. This would allow for a certificate to be considered as valid for some subset of the resources listed in this extension, without necessarily being considered as valid for all such described resources. The implications of this approach is that any mismatch between parent and subordinate over resources where the subordinate certificate lists resources that are not contained in the parent certificate would affect validity questions relating to only those particular resources, rather than invaliding the subordinate certificate for all resources, and all of its subordinate products. This would appear to offer a relatively precise match tot he defined problem space, and limits the scope of consequent third party damage in the event of a INR mismatch in the RPKI certification hierarchy.

Another approach may involve the alteration of the RPKI provisioning protocol [RFC6492] to include a specific signal from child to parent ("bottom up") relating to readiness for certificate revocation. At this stage it is entirely unclear how this signalling mechanism would operate, nor is it clear that it would alter the elements of operational fragility nor mitigate to any meaningful extent the risks of failure to ensure strict INR consistency at all times. This is a topic for further study.

5. Security Considerations

The Security Considerations of [RFC6487] and [RFC6492] do not address the topic described here. Obviously, within the current RPKI

validation procedure, any inconsistency in certificates located towards the apex of the RPKI hierarchy would invalidate the entirety of the sub-tree located below the point of this inconsistency. If the RPKI was used to control inter-domain routing in the context of a secure routing protocol, then the implications of this large scale invalidation of certificates would have a corresponding massive impact on the stability of routing. This appears to be a serious situation.

6. IANA Considerations

No updates to the registries are suggested by this document.

7. Acknowledgements

TBA.

8. References

8.1. Normative References

[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.

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

8.2. Informative References

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, February 2012.

[RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, February 2012.

Authors' Addresses

Geoff Huston
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: gih@apnic.net

George Michaelson
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: ggm@apnic.net

Carlos M. Martinez
Latin American and Caribbean IP Address Regional Registry
Rambla Mexico 6125
Montevideo 11400
Uruguay

Phone: +598 2604 2222
Email: carlos@lacnic.net

Tim Bruijnzeels
RIPE Network Coordination Centre
Singel 258
Amsterdam 1016 AB
The Netherlands

Email: tim@ripe.net

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

Email: andy@arin.net

Alain Aina
African Network Information Centre (AFRINIC)
11th Floor, Raffles Tower
Cybercity, Ebene
Mauritius

Phone: +230 403 51 00
Email: aalain@afrinic.net

SIDR Working Group
Internet-Draft
Intended status: BCP
Expires: November 24, 2014

S. Turner
IECA, Inc.
K. Patel
Cisco Systems
R. Bush
Internet Initiative Japan, Inc.

May 23, 2014

Router Keying for BGPsec
draft-ietf-sidr-rtr-keying-07

Abstract

BGPsec-speaking routers are provisioned with private keys to sign BGP messages; the corresponding public keys are published in the global RPKI (Resource Public Key Infrastructure) thereby enabling verification of BGPsec messages. This document describes two ways of provisioning the public-private key-pairs: router-driven and operator-driven.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 27, 2013.

Copyright Notice

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.

1. Introduction

BGPsec-speaking routers are provisioned with private keys, which allow them to digitally sign BGP messages. To verify the signature, the public key, in the form of a certificate [I-D.ietf-sidr-bgpsec-pki-profiles], is published in the RPKI (Resource Public Key Infrastructure). This document describes two methods for provisioning the necessary public-private key-pairs: router-driven and operator-driven.

The difference between the two methods is where the keys are generated: on the router in the router-driven method and elsewhere in the operator-driven method. Routers are expected to support either one, the other, or both methods to work in various deployment environments. Some routers may not allow the private key to be off-loaded while other routers may. Off-loading of private keys would support swapping of routing engines which could then have the same private key installed in the soon-to-be online engine that had previously been installed in the soon-to-be removed card.

The remainder of this document describes how operators can use the two methods to provision new and existing routers.

Note: [I-D.ietf-sidr-bgpsec-pki-profiles] specifies the format for the PKCS #10 request and [I-D.ietf-sidr-bgpsec-algs] specifies the algorithms used to generate the signature.

2. Terminology

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.

Readers are assumed to be familiar with the BGPsec protocol [I-D.ietf-sidr-bgpsec-overview][I-D.ietf-sidr-bgpsec-protocol] and the RPKI [RFC6480] as well as the BGPsec-specific PKI (Public Key Infrastructure) specifications [I-D.ietf-sidr-bgpsec-pki-profiles][I-D.ietf-sidr-bgpsec-algs].

3. Provisioning a New Router

Depending on the options supported by the new router, operators are free to use either the router-driven or operator-driven methods. Regardless of the method chosen, operators first establish a secure communication channel (e.g., via SSH (Secure Shell)) between the operator's management platform and the router to allow the operator to securely use the Command Line Interface (CLI). How this channel is established is router-specific and is not in scope of this document. Though other configuration mechanisms might be used, e.g. NetConf (see [RFC6470]), in the remainder of this document, the secure communication channel between the server and the router is assumed to be an SSH-protected CLI.

Encryption, integrity, authentication, and key exchange algorithms used by the secure communication channel SHOULD be of comparable strength to BGPsec keys, which currently is 128-bit, or stronger than BGPsec keys. In other words for the encryption algorithm, do not use export grade crypto (40-56 bits of security), do not use Triple DES (112 bits of security), do use something like or better than AES-128: aes128-cbc [RFC4253] and AEAD_AES_128_GCM [RFC5647]; for integrity use something like hmac-sha2-256 [RFC6668] or AESAD_AES_128_GCM [RFC5647]; for authentication use something like ecdsa-sha2-nistp256 [RFC5656], and; for key exchange use something like ecdh-sha2-nistp256 [RFC5656].

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

3.1. Router-Generated Keys

In the router-driven method, once the SSH-protected CLI session is established between the operator and the router, the operator issues a command, or commands, for the router to generate the public/private key pair, to generate the PKCS#10 request, and to sign the PKCS#10 with the private key. Once generated, the PKCS#10, which includes the public key the router wants certified, is transmitted to the RPKI CA for the CA to certify. This can be via a number of means, two of which might be as follows:

- Through the SSH-protected CLI session with the operator's RPKI management platform: The operator off-loads the PKCS#10 and uploads the request to the CA. If the CA is operated by an external entity, external network connectivity likely is required.

- Between the router and the CA: The operator, through a command or commands, prompts the router to send/transfer the PKCS#10 request to the CA over the network. Obviously for this to work, the router requires network connectivity with the CA and if the CA is operated by an external entity external network connectivity may be required.

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

- Publishes the certificate in the Global RPKI. The CA must have connectivity to the relevant publication point, which in turn must have external network connectivity as it is part of the Global RPKI.

- Returns the certificate to the operator's management station or to the router, normally packaged in a PKCS#7, using the corresponding method by which it received the certificate request.

With network connectivity, the router and CA can exchange the certificate request and the certificate using the application/pkcs10 media type [RFC5967] and application/pkcs7-mime [RFC5751], respectively, with the FTP [RFC2585], the HTTP [RFC2585], or the EST (Enrollment over Secure Transport) [RFC7030].

The router SHOULD extract the certificate from the PCKCS#7 and verify that the private key it holds corresponds to the returned public key. The router SHOULD inform the operator that the certificate was received; by some mechanism which is out of scope of this document. The router SHOULD inform the operator whether or not the keys correspond, again by a mechanism which is out of scope for this document.

The router SHOULD also verify that the returned certificate validates back to a trust anchor. To perform this verification either the CA's certificate needs to be installed on the router via the CLI or the CA's certificate needs to be returned along with the router's certificate in the PKCS#7. The router SHOULD inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope. After performing these checks, the router need not retain the CA's certificate because the certificate is not transmitted as part of BGPsec messages.

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

3.2. Operator-Generated Keys

In the operator-driven method, the operator generates the public/private key pair and installs the private key into the router over the SSH-protected CLI session. Note that cut/copy and paste operations for keys over a certain sizes are error-prone.

The operator uses RPKI management tools to generate the keys, the PKCS#10 certification request, the certificate, and the PKCS#7 certification response, as well as publishing the certificate in the Global RPKI. External network connectivity may be needed if the certificate is to be published in the Global RPKI.

Along with the PKCS#7, the operator returns the private key. The private key is encapsulated in a PKCS #8 [RFC5958], the PKCS#8 is further encapsulated in CMS (Cryptographic Message Syntax) SignedData [RFC5652], and signed by the AS's EE (End Entity) certificate.

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

The router SHOULD extract the certificate from the PKCS#7 and verify that the private key corresponds to the returned public key. The router SHOULD inform the operator whether it successfully received the certificate; this mechanism is out of scope. The router should inform the operator whether or not the keys correspond; this mechanism is out of scope. The router SHOULD also verify the returned certificate back to a trust anchor, but to perform this verification either the CA's certificate needs to be installed on the router via the CLI or the CA's certificate needs to be returned along with the router's certificate in the PKCS#7. The router SHOULD inform the operator whether or not the signature validates to a trust anchor; this notification mechanism is out of scope. After performing these checks, the router need not retain the CA certificate.

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

4. Key Management

An operator's responsibilities do not end after key generation, key

provisioning, certificate issuance, and certificate distribution. They persist for as long as the operator wishes to operate the BGPsec-speaking router.

Paramount to maintaining a router that can be a continuous BPGsec speaker is ensuring that the router has a valid certificate at all times. To ensure this, the operator needs to ensure the router always has a non-expired certificate. That is the key used when BGP-speaking always has an associated certificate whose expiry time is after the current time.

Ensuring this is not terribly difficult but requires that either:

- The router has a mechanism to notify the operator that the certificate has an impending expiration, and/or
- The operator notes the expiry time of the certificate and uses a calendaring program to remind them of the expiry time. It is advisable that the expiration warning happen well in advance of the actual expiry time, and/or
- The RPKI CA warns the operaor of pending expiration, and/or
- Use some other kind of automated process to search for and track the expiry times of router certificates.

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

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

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

Whether the certificate is rekeyed (i.e., different key in the certificate with a new expiry time) or renewed (i.e., the same key in the certificate with a new expiry time) depends on the key’s lifetime and operational use. Arguably, rekeying the router’s BGPsec certificate every time the certificate expires is more secure than

renewal because it limits the private key's exposure. However, if the key is not compromised the certificate could be renewed as many times as allowed by the operator's security policy. Routers that support only one key can use renewal to ensure continuous operation, assuming the certificate is renewed and distributed prior to the operational's certificate expiry time.

Certain unfortunate circumstances exist when the operator will need to revoke the router's BGPsec certificate. When this occurs, the operator needs to use the RPKI CA system to revoke the certificate by placing the router's BGPsec certificate on the CRL (Certificate Revocation List) as well as rekeying the router's certificate.

When it is decided that an active router key is to be revoked, the process of requesting the CA to revoke, the process of the CA actually revoking the router's certificate, and then the process of rekeying/renewing the router's certificate, (possibly distributing a new key and certificate to the router), and distributing the status takes time during which the operator must decide how they wish to maintain continuity of operations, with or without the compromised private key, or whether they wish to bring the router offline to address the compromise.

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

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

5. Other Use Cases

Current router code generates private keys for uses such as SSH, but the private keys may not be seen or off-loaded via the SSH-protected CLI session or any other means. While this is good security, it creates difficulties when a routing engine or whole router must be replaced in the field and all software which accesses the router must be updated with the new keys. Also, any network based initial contact with a new routing engine requires trust in the public key presented on first contact.

To allow operators to quickly replace routers without requiring update and distribution of the corresponding public keys in the RPKI, routers SHOULD allow the private BGPsec key to be off-loaded via the SSH-protected CLI, NetConf (see [RFC6470]), SNMP, etc. This lets the operator upload the old private key via the mechanism used for operator-generated keys, see Section 3.2.

6. Security Considerations

Operator-generated keys could be intercepted in transport and the recipient router would have no way of knowing a substitution had been made or that the key had been disclosed by a monkey in the middle. Hence transport security is strongly RECOMMENDED. As noted in Section 3, the level of security provided by the transport security SHOULD be commensurate with the BGPsec key. Additionally, operators SHOULD ensure the transport security implementation is up to date and addresses all known implementation bugs.

All generated key pairs MUST be generated from a good source of non-deterministic random input [RFC4086] and the private key MUST be protected in a secure fashion. Disclosure of the private key leads to masquerade [RFC4949]. The local storage format for the private key is a local matter.

Though the CA’s certificate is installed on the router and used to verify the returned certificate is in fact signed by the CA, the revocation status of the CA’s certificate is not checked. The operator MUST ensure that installed CA certificate is valid.

Operators need to manage their SSH keys to ensure only those authorized to access the router may do so. As employees no longer need access to the router, their keys SHOULD be removed from the router.

7. IANA Considerations

This document has no IANA Considerations.

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.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, August 2010.

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

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

8.2. Informative References

[I-D.ietf-sidr-bgpsec-overview] Lepinski, M. and S. Turner, "An Overview of BGPSEC", draft-ietf-sidr-bgpsec-overview (work in progress).

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

[RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI

36, RFC 4949, August 2007.

[RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, August 2009.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, December 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, August 2010.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, March 2011.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, February 2012.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, 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.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, July 2012.

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

Authors' Addresses

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, Virginia 22031
US

Email: turners@ieca.com

Keyur Patel
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: keyupate@cisco.com

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

Phone: +1 206 780 0431 x1
Email: randy@psg.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 3, 2019

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

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

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 16, 2017.

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

2. Management / Router Communication

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

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

3. Exchange Certificates

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

- Use application/pkcs10 media type [RFC5967] to extract certificate requests and application/pkcs7-mime [I-D.lamps-rfc5751-bis] to return the issued certificate,

- Use FTP or HTTP per [RFC2585], and

- Use Enrollment over Secure Transport (EST) protocol per [RFC7030].

4. Set-Up

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

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

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

5. Generate PKCS#10

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

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

used to generate the signature on the CSR.

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

5.1. Router-Generated Keys

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

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

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

5.2. Operator-Generated Keys

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

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

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

5.2.1. Using PKCS#8 to Transfer Private Key

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

(EE) private key.

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

6. Send PKCS#10 and Receive PKCS#7

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

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

1. Publishes the certificate in the Global RPKI. The CA must have connectivity to the relevant publication point, which in turn must have external network connectivity as it is part of the Global RPKI.

2. Returns the certificate to the operator's management station, packaged in a PKCS#7 certs-only message, using the corresponding method by which it received the certificate request. It SHOULD include the certificate chain below the TA Certificate so that the router can validate the router certificate.

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

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

7. Install Certificate

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

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

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

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

8. Advanced Deployment Scenarios

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

The operator burden shifts here to include:

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

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

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

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

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

9. Key Management

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

9.1. Key Validity

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

Ensuring this is not terribly difficult but requires that either:

1. The router has a mechanism to notify the operator that the certificate has an impending expiration, and/or

2. The operator notes the expiry time of the certificate and uses a calendaring program to remind them of the expiry time, and/or

3. The RPKI CA warns the operator of pending expiration, and/or

4. The operator uses some other kind of automated process to search for and track the expiry times of router certificates.

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

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

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

9.2. Key Roll-Over

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

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

9.3. Key Revocation

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

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

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

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

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

9.4. Router Replacement

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

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

10. Security Considerations

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

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

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

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

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

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

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

have access to a router.

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

11. IANA Considerations

This document has no IANA Considerations.

12. References

12.1. Normative References

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

[I-D.lamps-rfc5751-bis]
Schaad, J., Ramsdell, B, S. Turner, "Secure/Multipurpose Internet Mail Extension (S/MIME) Version 4.0", draft-ietf-lamps-rfc5751-bis (work in progress), July 2018.

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

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

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009,

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

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8208] Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, and Signature Formats", RFC 8208, DOI 10.17487/RFC8208, September 2017, <https://www.rfc-editor.org/info/rfc8208>.

[RFC8209] Reynolds, M., Turner, S., and S. Kent, "A Profile for BGPsec Router Certificates, Certificate Revocation Lists, and Certification Requests", RFC 8209, DOI 10.17487/RFC8209, September 2017, <https://www.rfc-editor.org/info/rfc8209>.

[802.1AR] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

12.1. Informative References

[I-D.ylonen-sshkeybcp] Ylonen, T. and G. Kent, "Managing SSH Keys for Automated Access - Current Recommended Practice", draft-ylonen-sshkeybcp (work in progress), April 2013.

[RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, DOI 10.17487/RFC2585, May 1999, <https://www.rfc-editor.org/info/rfc2585>.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, DOI 10.17487/RFC3766, April 2004, <https://www.rfc-editor.org/info/rfc3766>.

[RFC5273] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC): Transport Protocols", RFC 5273, DOI

10.17487/RFC5273, June 2008, <https://www.rfc-editor.org/info/rfc5273>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

[RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, DOI 10.17487/RFC5647, August 2009, <https://www.rfc-editor.org/info/rfc5647>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC5967] Turner, S., "The application/pkcs10 Media Type", RFC 5967, DOI 10.17487/RFC5967, August 2010, <https://www.rfc-editor.org/info/rfc5967>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[RFC6484] Kent, S., Kong, D., Seo, K., and R. Watro, "Certificate Policy (CP) for the Resource Public Key Infrastructure (RPKI)", BCP 173, RFC 6484, DOI 10.17487/RFC6484, February 2012, <https://www.rfc-editor.org/info/rfc6484>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

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

[RFC8205] Lepinski, M., Ed., and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

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

Appendix A. Management/Router Channel Security

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

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

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

Appendix B. The n00b Guide to BGPsec Key Management

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

Email: randy@psg.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

Keyur Patel
Arrcus, Inc.

Email: keyur@arrcus.com

Secure Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2015

S. Kent
D. Mandelberg
BBN Technologies

July 3, 2014

Suspenders: A Fail-safe Mechanism for the RPKI
draft-kent-sidr-suspenders-02

Abstract

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). The mechanisms described below, and referred to as "Suspenders" are intended to address this risk.

Suspenders enables an INR holder to publish information about changes to objects it signs and publishes in the RPKI repository system. This information is made available via a file that is external to the RPKI repository, so that Relying Parties (RPs) can detect erroneous or malicious changes related to these objects. RPs can then decide, individually, whether to accept changes that are not corroborated by independent assertions by INR holders, or to revert to previously verified RPKI data.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2015.

Table of Contents

1. Overview

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet number resources (INRs) to make verifiable statements about those resources. For example, the holder of a block of IP(v4 or v6) addresses can issue a Route Origination Authorization (ROA) to authorize an autonomous system to originate routes for that block.

The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a

CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). Suspenders is a collection of mechanisms designed to address this potential problem. It addresses the first use case described in [I-D.ietf-sidr-lta-use-cases]. This use case describes a scenario in which an RIR is compelled to remove or modify RPKI data signed by the RIR, but the community of network operators wants to continue using the RPKI as though these actions had not taken place.

Assertions by INR holders about their resources, and about bindings among resources, are realized by publishing RPKI signed objects via the RPKI repository system [RFC6481]. For example, authorization to originate a route for a prefix is accomplished by issuing a ROA. Changes in the RPKI can have an adverse impact on routing in the Internet, by changing the set of (valid) signed objects for a resource. Invalidating a ROA could cause the origin authorized by the ROA in question to be less preferred; adding a ROA for a more specific prefix could enable an unauthorized party to represent itself as the legitimate origin for traffic for that prefix.

The goal of Suspenders is to minimize the likelihood that changes to the RPKI will adversely affect INR holders, irrespective of whether the changes are inadvertent or malicious. Suspenders should work when an INR holder acts as its own CA (and manages its own publication point), and when the INR holder has outsourced these management functions. Suspenders allows each INR holder to assert a "lock" on selected objects at its publication point, to protect the bindings asserted by these objects. Changes to protected objects are confirmed by the INR holder, via a file published outside the repository system. Changes to the validity of protected objects, effected by changes to any other objects in the RPKI, are presumed to be unauthorized (and thus suspicious), unless independently confirmed by the INR holder.

Detection of potentially adverse changes is carried out by each INR holder for its own resources, and by each RP that elects to implement Suspenders. It is critical that an INR holder be able to quickly detect adverse changes that affect its own resources, so that it can initiate actions to remedy the problem. RPs should be able to detect potentially adverse changes, that are not authorized by INR holders, so that they can (at their discretion) use cached, validated data in lieu of such changes. The model adopted here is to assume that changes to previously-validated data should not be accepted, unless authorized by the relevant INR holder. Thus RPs who detect changes need to be able to verify that these changes are authorized by the INR holder. Because not all INR holders manage their own CAs and publication points, an external mechanism is used to signal authorized changes to RPs.

2. Terminology

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

3. The LOCK Record and INRD File

An INR holder that elects to protect its resources and resource bindings creates a LOCK record in its publication point. The INR holder also generates and signs an Internet Number Resource Declaration (INRD) file, and publishes it at a location independent of the RPKI repository system. The LOCK record consists of a URL that points to the INRD file, and a public key used to verify a signature on the content of that file. (This public key is distinct from any used by the INR holder in the RPKI context.) The INRD file contains the date at which the most recent changes were made, and enumerates those changes. The formats of the LOCK record and INRD file are described below.

3.1. LOCK Record Format and Semantics

The LOCK record conforms to the signed object specification from [RFC6488], which, in turn, uses the CMS [RFC5652] signed-data object format. See [RFC6488] for the top-level signed-data format and the constraints imposed on that format for use in the RPKI context. The LOCK encapsulated content is defined below:

```
EncapsulatedContentInfo ::= SEQUENCE {
    eContentType ContentType,
    eContent [0] EXPLICIT OCTET STRING OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

The eContentType for an LOCK record is defined as id-ct-rpkiLOCK and it has the numeric value 1.2.840.113549.1.9.16.1.XX.

```
id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                 rsadsi(113549) pkcs(1) pkcs9(9) 16 }

id-ct OBJECT IDENTIFIER ::= { id-smime 1 }

id-ct-rpkiLOCK OBJECT IDENTIFIER ::= { id-ct XX }
```

The eContent for an LOCK record is defined by the following ASN.1

```
LOCK ::= SEQUENCE {
    version     [0] INTEGER DEFAULT 0,
    outsourced  BOOLEAN,
    uRL         IA5String,
    publicKey   SubjectPublicKeyInfo }

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm         AlgorithmIdentifier,
    subjectPublicKey  BIT STRING  }
```

The EE certificate embedded in the LOCK record MUST use the inherit flag in the [RFC3779] extensions. (The content of the LOCK is independent of the 3779 extensions in the EE certificate, so it is appropriate to use the inherit flag here.)

The version number of the LOCK record determines the set of RPKI object types that it protects. Version 0 protects the LOCK record itself, ROAs, (subordinate) CA certificates, and router certificates (if present).

The algorithm and subjectPublicKey fields in the publicKey MUST conform to the guidance in Section 3 of [RFC6485].

If an RP elects to process a LOCK record, it verifies the signature on the record using the procedure described in [RFC6488]. If the signature verification fails, it ignores the record. (If the RP has a previously validated LOCK record, it continues to use that record instance.)

If the signature verification succeeds, the RP extracts the version number and verifies that the RP is prepared to process this version of the record. If not, it ignores the record. If it is prepared to process this version, it extracts the URL and public key fields. The URL is used to fetch the corresponding INRD file, and the public key is used to verify the signature on that file.

If the RP has a copy of an INRD file for this publication point, and if the RP detects no material changes to the protected records at the publication point, the RP SHOULD NOT fetch the INRD file. (A material change is one that affects the semantics of the object. For example, for a ROA, only changes to the prefixes and/or ASN are material.) If the RP does not hold a copy of the INRD file, or if a protected record has changed, the RP fetches a new INRD file using the URL, and proceeds as described in Section 3.2.

When an INR holder has outsourced management of its RPKI CA function and publication point, it is susceptible to attacks in which the LOCK record itself is changed. This is because the entity providing these

functions could create a new LOCK record containing a new URL and public key, thus defeating the LOCK/INRD mechanism. An authorized change to the content of a LOCK record should be very rare. A location selected as a home for an INRD file should be stable, and thus the URL should rarely change. The public key used to verify the signature on an INRD file should also be constant for long intervals. The LOCK record contains a flag that indicates whether the INR holder has outsourced CA and publication point management. If this flag is FALSE, an RP will accept changes to the LOCK record (see Section 5) just as it would changes to any other object at a protected publication point. If the flag is TRUE, then any change to a LOCK record is regarded as suspicious by RPs. In such cases the RP delays accepting the new LOCK record and associated INRD file, as discussed in Section 5.

3.2. INRD File Format and Semantics

The INRD file is a DER-encoded ASN.1 file that contains data associated with a single INR holder (publication point owner). The file is encoded using ASN.1, since most of the values it holds will be compared to data from RPKI objects that also are ASN.1 encoded, and because it is a signed object.

```
INRD ::= SEQUENCE {
    tbsINRD    TBSINRD,
    algorithm  AlgorithmIdentifier,
    signature  OCTET STRING
    }

TBSINRD ::= SEQUENCE {
    version       [0] INTEGER DEFAULT 0,
    lastChange        UTCTime,
    changeWindow      ENUMERATED
        {
        1week (7) DEFAULT
        2week (14)
        4week (28)
        },
    additions     [1] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    deletions     [2] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    keyRollover   [3] OCTET STRING OPTIONAL,
    algRollover   [4] OCTET STRING OPTIONAL
    }

ProtectedObject ::= CHOICE {
    cmsObject    [0] EncapsulatedContentInfo,
    rtrCert      [1] RtrCertInfo,
    cACert       [2] CACertInfo
    }

RtrCertInfo ::= SEQUENCE {
    subjKeyId  OCTET STRING,
    aSNum      INTEGER
    }

CACertInfo ::= SEQUENCE {
    subjKeyId          OCTET STRING,
    ipAddrBlocks   [0] IPAddrBlocks OPTIONAL,
    aSIdentifiers  [1] ASIdentifiers OPTIONAL
    }

-- See [RFC3779] for the definitions of IPAddrBlocks and
-- ASIdentifiers.
```

The lastChange and changeWindow values are used to bound the set of additions and deletions stored in an INRD file. The default is a one week window, but two and four week values also may be expressed. The window determines the oldest time at which changes to protected

objects at the publication point are represented in the additions and deletions portions of the file.

The additions element is used by an INR holder to list all protected objects that have been added to the publication point over the interval defined by the change window. If no objects have been added during this interval, the element is omitted. Similarly, the deletions element is used by an INR holder to list all protected objects that have been removed from the publication point over the interval defined by the change window. If no objects have been removed during this interval, the element is omitted.

A LOCK or ROA is listed in the additions and/or deletions fields by putting its EncapsulatedContentInfo in the cmsObject field of a ProtectedObject. A router certificate is listed by putting its SKI and AS number in the rtrCert field. A CA certificate is listed by putting its SKI and [RFC3779] resources in the cACert field. If the outsourced flag in the LOCK record is FALSE, then no CA certificates should be included in the additions or deletions elements. If any CA certificates are included in these elements, they are ignored. RPs SHOULD accept all valid CA certificates issued at this publication point when the outsourced flag is FALSE.

The key rollover element is present only during the time when a key rollover [RFC6489] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" key. During key rollover each CA will have its own LOCK record, that points to its own INRD file. The old CA will list the new CA’s SKI here; the new CA will not include this field. Key rollover is a transient condition, so the need for both LOCK records and INRD files ought not be very long.

The algorithm rollover element is present during the time when an algorithm rollover [RFC6916] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" algorithm. During algorithm rollover each CA will have its own LOCK record, that points to its own INRD file, and each of them will list the other CA’s SKI here. (Note that the SKI value is compared against the SKI in the CA certificate in question. An RP does not compute an SKI. This means that changes to the hash algorithm used to compute an SKI do not affect how an RP processes this field. An RP MUST be prepared to deal with an SKI length other than the 20 octet value in common use today.) Algorithm transition is a long process, so both sets of LOCK records and INRD files will persist for an extended period.

An RP fetches an INRD file using a URL from a LOCK record, as noted above. If the file cannot be located, the RP software logs an error and regards any changes to the publication point as suspicious. If the file is located, the RP verifies the signature on the file using the public key (and indicated algorithms) from the same LOCK record. If the signature fails, the RP software logs an error and regards any changes to the publication point as suspicious. If the signature is valid, the RP extracts the data elements from the INRD file.

If this is the first time that an INRD file is fetched for this publication point, the file is accepted, and its content is used to populate the RP's expanded local cache. If the INRD file is replacing a previously acquired instance for this publication point, the content is used to confirm changes to protected objects at this publication point. If an RP detects changes to protected publication point objects that occurred after the lastChange time, these changes are treated as suspicious.

Authorizing changes to subordinate CA certificates in an INRD file is critical when an INR holder outsources CA and publication point management. Listing these CAs and their associated 3779 extension data enables an RP to detect creation of unauthorized CAs that could then create competing ROAs or router certificates. However, if an INR holder operates its own CA and manages its publication point, it is not necessary to protect against such attacks. To signal this situation, the "outsourced" flag in the LOCK record is set to FALSE. Under this condition, an RP will not impose change control checks on subordinate CA certificates for the publication point.

Some classes of INR holders need not publish a LOCK record and INRD file. IANA, RIRs, and NIRs, are principally delegators of resources. Each of these RPKI entities SHOULD create one publication point for the resources used by the entity for its own networking needs, and a separate publication point under which all resource delegations take place. The first publication point MAY be protected by a LOCK record, so that ROAs and router certificates associated with those resources can be protected. However, the second publication point OUGHT not include a LOCK record. If this convention is followed, these classes of INR holders need not update an INRD file every time a new subordinate CA is created or modified, as a result of delegation. If an INR holder follows this convention, and includes a LOCK record in its superior publication point, that record, and the associated INRD file, conveys some degree of protection for the subordinate CA resources, even if the INR holders of these resources do not publish LOCK records.

4. Self-checking by RPs

It is easy for an INR holder, acting as an RP, to determine if any of its resource bindings have been undermined via the RPKI. It knows what resources it holds, and what assertions it has made relevant to those resources. Any conflicting RPKI objects represent a problem! It is more difficult for an RP to detect problems with another INR holder's resources, because it lacks the knowledge that the other INR holder has. The mechanisms described in Section 5 are designed to enable RPs to detect these problems. This section describes the procedures each RP executes to detect adverse changes to its own data in the RPKI repository system. Note that the procedures in this section do not require use of the LOCK record or INRD file.

When and INR downloads RPKI data, as it normally does, it SHOULD perform the checks noted below, to detect problems. To enable such checking, each INR holder's RP software MUST be configured with data about the ROAs, and other protected objects, of this INR holder. If any of these objects are missing or fail to validate, then the INR holder has detected a problem, and is notified.

The semantics of ROAs require an additional check; if other ROAs for the same or more specific prefixes are found anywhere in the RPKI repository system, this too indicates a problem, and the INR holder is notified.

The semantics of router certificates, require a separate, additional check. A router certificate binds a public key (and a router ID) to an ASN. Thus, if an INR holder discovers router certificates for its ASN, that it did not authorize, this indicates a problem.

As additional objects are protected via this mechanism, it will be necessary to perform additional checks to detect the latter sort of adverse changes, based on the semantics of the protected objects.

In any case, RP software SHOULD inform the INR holder of the apparent cause and source of the problem, e.g., a revoked or expired certificate or a manifest problem, and guide the INR holder to the responsible CAs (e.g., using Ghostbusters [RFC6493] records).

When an INR holder is alerted to a change adversely affecting its own resources, it is expected to contact the appropriate RPKI entities to rectify the error in a timely fashion. If the changes are determined to be intentional (and not authorized by the INR holder), the INR holder can inform the Internet operations community (via an out of band mechanism), which can then decide, individually, how to respond.

Remedying a problem detected by an INR holder is not likely to be instantaneous, even if the problem is just an error. To avoid adversely affecting routing, the mechanisms described in Section 5 enable RPs to detect a change that adversely affects INR holders, and to reject it, reverting to previously validated INR data. This gives the INR holder time to resolve the problem. Reverting to an earlier version of INR data is conceptually easy for RPs, because they already cache RPKI data. The mechanisms described below require augmenting the RPKI local cache maintained by each RP, to detect adverse changes, making use of information gleaned from LOCK records and INRD files. The next section describes how the LOCK and INRD data is used.

5. Detection & Remediation

The design described in this section assumes that an RP has acquired a snapshot of the RPKI repository, validated and extracted INR holding and binding data, and considers this data to be "good". The detection and remediation algorithm is initialized by acquiring a complete download of RPKI repository data, and by fetching INRD files for all publication points that contain a LOCK record. (Prior to this initialization step, it is not possible for an RP to detect and respond to adverse changes to the RPKI, using the technique described below.)

Each RP already maintains a cache of RPKI data [RFC6480], [RFC6481]; this document extends that cache. For every publication point that contains a LOCK record, the content of that record, and the corresponding INRD file content, become part of the data maintained by each RP.

An RP acquires and validates all changed RPKI objects as usual. An RP does not update its cache with the changes, until additional checks, described below, are performed. Before accepting any changes the RP MUST process every pub point where there is (or was) a LOCK record. For each of these pub points, if there are changes to protected objects, these changes must be confirmed by the corresponding INRD file before they are accepted. If any of these checks fail, the changes are held in escrow, waiting for a timeout (or an updated INRD file?). After all protected pub point changes have been processed, then changes for unprotected pub point can be accepted. The checks will detect pending changes that would whack or compete with protected objects, and place them in escrow.

After validating all changed objects downloaded from the RPKI repository, an RP performs the following additional checks for every publication point that has (or had) a LOCK record:

- ROA and router certificate whacking
- INR sub-delegation changes
- ROA competition
- router certificate competition
- LOCK record changes

A ROA or a router certificate has been whacked (see Appendix A) if it was valid and is now missing or invalid, and if it is not indicated as deleted in the INRD file of its issuer. Any previously valid ROA that is no longer valid (or missing) is checked against the INRD file for the ROA issuer, to determine if the ROA or (router certificate) certificate has been legitimately revoked/removed. If the INRD file confirms the action, the old ROA (or router certificate) is removed from the local cache. If not, the old ROA (or router certificate) is retained, but marked as suspicious.

Changes to INR sub-delegation occur when the INR holder issues a new CA certificate, an existing child CA certificate expires, or any other change affects the status of a child CA certificate. These changes are accepted by an RP only if they are confirmed in the INR holder's INRD file.

A newly issued ROA is in competition (see Appendix A) with an existing ROA if the new ROA specifies the same or a more specific prefix than the older ROA, the new ROA is not issued by one of the existing ROA's issuer's descendants, and the new ROA was not authorized by the INRD file of the existing ROA. A competing ROA is not accepted as valid by an RP.

A newly-issued router certificate competes with an existing router certificate, if the new certificate includes the same ASN and was not authorized by the INRD file covering the existing router certificate. A competing router certificate is not accepted as valid by an RP. (Such a certificate would be accepted if the INRD file of the issuer of the original certificate indicates that the old certificate has been deleted, and not replaced with a new router certificate associated with the same entity. In this case, the newly-issued certificate would not be in competition.)

As noted above, any change to a LOCK record is viewed as suspicious unless the outsourced flag is FALSE. If the record is for a publication point that is not outsourced, then a changed LOCK record is accepted as valid if the corresponding INRD file authorizes the new record. (If the INR holder has changed the public key for the

INRD file, it is RECOMMENDED that the URL also change. This allows the INR holder to publish a new INRD file that authorizes the new LOCK record, minimizing the potential race condition between updating an INRD file and a LOCK record.)

If the LOCK record shows that the publication point is outsourced, an RP examines the changes made to the LOCK record. If the URL has changed, but the public key and the outsource flag are unchanged, the new LOCK record may be accepted, if the new INRD file authorizes the change. If not, the new LOCK record is rejected. If the public key has been changed, a delay is imposed on accepting the new LOCK record, even if the INRD file authorizes the change. (should we establish a global delay, or should each INR holder publish its own delay preference in the INRD file?)

Remediation for all of the whacking and competition events consists of NOT making a change in the local cache when an unconfirmed change is encountered.

6. INRD Management Scenarios

Common wisdom notes that we cannot choose our parents, but we can choose our friends, and we should do so wisely. In the RPKI context, and INR holder cannot, generally choose its CA, but Suspenders allows the INR holder to choose its INRD file server. It should do so wisely.

An INRD file is published outside of the RPKI repository system, and is verified using a public key that is also independent of the RPKI. The motivation for these two measures is to insulate this part of the Suspenders system from possible manipulation by an entity to whom CA and publication point services have been outsourced. If an INR holder acts as its own CA, and manages its own publication point, it can publish its INRD file on the same machines as its publication point, but not in the publication point. In this case the independence features are not critical, but they also don’t cause harm for this class of INR holder.

Every INR holder needs to choose a location for the INRD file that is highly available. When an INR holder has outsourced CA and publication point management, independent publication of the INRD file is critical. The INR holder needs to choose a location for the INRD file that is highly available. It also is appropriate to consider placing the file outside of the geopolitical region in which the INR holder (and its RIR) operate. Here too the motivation is to insulate the INR holder from a malicious action by the CA service provider, or, perhaps, an RIR above it.

Organizations may arise to offer hosting for INRD files, as a service for INR holders. They could offer just file storage, or they might offer more extensive services. For example, an organization might monitor an INR holder's publication point and create the INRD file data, and even sign it for the INR holder. (In this case the organization would provide the public key to the INR holder for inclusion in the LOCK record.) Various other arrangements between the INR holder and a organization that assists in managing INRD files are possible, and are a local matter between the INR holder and the organization.

A country might elect to mandate use of Suspenders, as a means to protect the INRs of its ISPs and other organizations that run BGP with in the country. The motivation is similar to that cited above, i.e., protecting INRs against errors or malicious actions by RPKI entities. In this case the country itself generally is not an INR holder per se, so the relationship is somewhat different from that discussed above. Nonetheless, the mechanisms described above apply.

For example, Elbonia might mandate that every INR holder within the country make use of Suspenders. Every Elbonian INR holder will be required to include a LOCK record in its publication point, no matter where that publication point is realized. The URL in each LOCK points to a file on a server managed by an Elbonian government organization. Each Elbonian ISP would be required to follow the procedures described in Section 5, when managing its local cache.

7. IANA Considerations

This document registers the following in the "RPKI Signed Object" registry created by [RFC6488]:

Name: LOCK
OID: 1.2.840.113549.1.9.16.1.XX
Reference: [RFCxxxx] (this document)

This document also registers the following three-letter filename extension in the "RPKI Repository Name Schemes" registry created by [RFC6481]:

Filename extension: lck
RPKI Object: LOCK
Reference: [RFCxxxx] (this document)

8. Security Considerations

This document specifies Suspenders, a set of security-focused mechanisms designed to protect INR holders against accidental and malicious changes to RPKI repository data, and to enable RPs to detect and respond to such changes. More text to be provided later.

9. Acknowledgements

Richard Barnes provided the motivation to develop Suspenders, after he identified a problem with the LTAM [I-D.ietf-sidr-ltamgmt] design.

10. References

10.1. Informative References

[I-D.ietf-sidr-lta-use-cases]
Bush, R., "RPKI Local Trust Anchor Use Cases", draft-ietf-sidr-lta-use-cases-01 (work in progress), June 2014.

[I-D.ietf-sidr-ltamgmt]
Reynolds, M., Kent, S., and M. Lepinski, "Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08 (work in progress), April 2013.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

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

[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.

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, February 2012.

10.2. Normative References

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

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

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

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

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

[RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, April 2013.

Appendix A. RPKI Object Whacking and Competition

There are two ways that an RPKI object can be adversely affected. We term these actions "whacking" and "competition."

Any object in the RPKI can become invalid or inaccessible (to RPs) via various actions by CAs and/or publication point maintainers along the certificate path from the object's EE certificate to a trust anchor (TA). Any action that causes an object to become invalid or inaccessible is termed "whacking". Revocation of the EE certificate for an object whacks it. Revocation of any CA certificate along the certificate path for the object (without reissuance) has the same effect. Reissuance of any CA certificate along the certificate path, with changes in [RFC3779] extensions in any of these certificates to exclude the resources cited in a targeted object, also constitutes whacking. Changing a manifest along the certificate path might whack an object (depending on how RPs deal with manifest changes), and removing an object from the RPKI repository system also potentially whacks it. Unless an action that causes an object to be whacked is authorized by the creator of an object, whacking is an attack against the INR holder that created the whacked object.

A different form of attack is termed object "competition". The details of object competition are determined by the semantics of the object. In the general case, one object competes with another object (of the same type), if the newer object creates a binding that adversely affects the binding expressed in the original object. So, for example, a newly issued ROA competes with an existing ROA if the new ROA contains the same or more specific prefixes than the older ROA. Competition does not always indicate an attack; the transfer of resources in a "make before break" model implies ROA competition. A newly issued router certificate competes with a previously issued one if the new certificate binds the same ASN to a public key issued by a

different entity. (If key rollover or algorithm transition is in progress, such competition is explicitly authorized via the INRD file.)

Competition that is not authorized by the issuer of the original router certificate is viewed as an attack against that certificate.

Appendix B. Design Criteria (do we still need this section?)

Several criteria were employed in developing the mechanisms described in this document.

1. It is anticipated that object whacking and competition, and analogous forms of errors that adversely impact INR holders, will be infrequent. Thus the detection mechanisms employed by RPs to detect such anomalies ought to be efficient (in terms of data fetching, processing, and storage) for the normal case.

2. RPs may elect to ignore/reject adverse changes to objects if they perceive such changes as suspicious. If an RP elects to reject a change to an object it must have access to previously validated objects for the INR holder question.

3. Transfers of "live" address space will occur, although not frequently. INR holders engaged in such transfers must be able to signal to RPs that such transfers are authorized, so that the transfers are not rejected as suspicious.

4. Routes for a prefix may be legitimately originated by more than one AS (MOA). The design MUST enable an INR holder to inform RPs when this situation is authorized.

5. Many INR holders may choose to outsource CA and publication point management functions. INR holders who choose to outsource these functions should be offered equivalent protection against ROA invalidation and competition as INR holders who perform these functions for themselves.

6. Any new RPKI repository objects used with the mechanisms defined here MUST conform to the format specified in [RFC6488].

7. The decision to process any additional data associated with the mechanisms described in this document is local to each RP. RPs that choose to not implement these mechanisms will incur minimal additional data fetching, storage, and processing burdens.

8. The decision to employ the mechanisms described here to protect INR holdings and binding is a local one made by each INR holder.

(INR holders who outsource CA and publication point management functions will require the providers of these services to support creation and publication of one new RPKI object. As a result, all such providers must support generation and maintenance of the new RPKI object so that their clients have the option to utilize these capabilities.)

9. Revocation and expiration of RPKI object MUST continue to work as they do currently, for all objects that have not been adversely affected.

Authors' Addresses

Stephen Kent
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: kent@bbn.com

David Mandelberg
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: david@mandelberg.org

Secure Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

S. Kent
D. Mandelberg
BBN Technologies

October 19, 2015

Suspenders: A Fail-safe Mechanism for the RPKI
draft-kent-sidr-suspenders-04

Abstract

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet Number Resources (INRs) to make verifiable statements about those resources. The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). The mechanisms described below, and referred to as "Suspenders" are intended to address this risk.

Suspenders enables an INR holder to publish information about changes to objects it signs and publishes in the RPKI repository system. This information is made available via a file that is external to the RPKI repository, so that Relying Parties (RPs) can detect erroneous or malicious changes related to these objects. RPs can then decide, individually, whether to accept changes that are not corroborated by independent assertions by INR holders, or to revert to previously verified RPKI data.

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 21, 2016.

Table of Contents

1. Overview

The Resource Public Key Infrastructure (RPKI) is an authorization infrastructure that allows the holder of Internet number resources (INRs) to make verifiable statements about those resources. For example, the holder of a block of IP(v4 or v6) addresses can issue a Route Origination Authorization (ROA) to authorize an autonomous system to originate routes for that block.

The certification authorities (CAs) in the RPKI issue certificates to match their allocation of INRs. These entities are trusted to issue certificates that accurately reflect the allocation state of resources as per their databases. However, there is some risk that a

CA will make inappropriate changes to the RPKI, either accidentally or deliberately (e.g., as a result of some form of "government mandate"). Suspenders is a collection of mechanisms designed to address this potential problem. It addresses the first use case described in [I-D.ietf-sidr-lta-use-cases]. This use case describes a scenario in which an RIR is compelled to remove or modify RPKI data signed by the RIR, but the community of network operators wants to continue using the RPKI as though these actions had not taken place.

Assertions by INR holders about their resources, and about bindings among resources, are realized by publishing RPKI signed objects via the RPKI repository system [RFC6481]. For example, authorization to originate a route for a prefix is accomplished by issuing a ROA. Changes in the RPKI can have an adverse impact on routing in the Internet, by changing the set of (valid) signed objects for a resource. Invalidating a ROA could cause the origin authorized by the ROA in question to be less preferred; adding a ROA for a more specific prefix could enable an unauthorized party to represent itself as the legitimate origin for traffic for that prefix.

The goal of Suspenders is to minimize the likelihood that changes to the RPKI will adversely affect INR holders, irrespective of whether the changes are inadvertent or malicious. Suspenders should work when an INR holder acts as its own CA (and manages its own publication point), and when the INR holder has outsourced these management functions. Suspenders allows each INR holder to assert a "lock" on selected objects at its publication point, to protect the bindings asserted by these objects. Changes to protected objects are confirmed by the INR holder, via a file published outside the repository system. Changes to the validity of protected objects, effected by changes to any other objects in the RPKI, are presumed to be unauthorized (and thus suspicious), unless independently confirmed by the INR holder.

Detection of potentially adverse changes is carried out by each INR holder for its own resources, and by each RP that elects to implement Suspenders. It is critical that an INR holder be able to quickly detect adverse changes that affect its own resources, so that it can initiate actions to remedy the problem. RPs should be able to detect potentially adverse changes, that are not authorized by INR holders, so that they can (at their discretion) use cached, validated data in lieu of such changes. The model adopted here is to assume that changes to previously-validated data should not be accepted, unless authorized by the relevant INR holder. Thus RPs who detect changes need to be able to verify that these changes are authorized by the INR holder. Because not all INR holders manage their own CAs and publication points, an external mechanism is used to signal authorized changes to RPs.

2. Terminology

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

3. The LOCK Record and INRD File

An INR holder that elects to protect its resources and resource bindings creates a LOCK record in its publication point. The INR holder also generates and signs an Internet Number Resource Declaration (INRD) file, and publishes it at a location independent of the RPKI repository system. The LOCK record consists of a URL that points to the INRD file, and a public key used to verify a signature on the content of that file. (This public key is distinct from any used by the INR holder in the RPKI context.) The INRD file contains the date at which the most recent changes were made, and enumerates those changes. The formats of the LOCK record and INRD file are described below.

3.1. LOCK Record Format and Semantics

The LOCK record conforms to the signed object specification from [RFC6488], which, in turn, uses the CMS [RFC5652] signed-data object format. See [RFC6488] for the top-level signed-data format and the constraints imposed on that format for use in the RPKI context. The LOCK encapsulated content is defined below:

```
EncapsulatedContentInfo ::= SEQUENCE {
    eContentType ContentType,
    eContent [0] EXPLICIT OCTET STRING OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

The eContentType for an LOCK record is defined as id-ct-rpkiLOCK and it has the numeric value 1.2.840.113549.1.9.16.1.XX.

```
id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                 rsadsi(113549) pkcs(1) pkcs9(9) 16 }

id-ct OBJECT IDENTIFIER ::= { id-smime 1 }

id-ct-rpkiLOCK OBJECT IDENTIFIER ::= { id-ct XX }
```

The eContent for an LOCK record is defined by the following ASN.1

```
LOCK ::= SEQUENCE {
    version     [0] INTEGER DEFAULT 0,
    outsourced  BOOLEAN,
    uRL         IA5String,
    publicKey   SubjectPublicKeyInfo }

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm         AlgorithmIdentifier,
    subjectPublicKey  BIT STRING  }
```

The EE certificate embedded in the LOCK record MUST use the inherit flag in the [RFC3779] extensions. (The content of the LOCK is independent of the 3779 extensions in the EE certificate, so it is appropriate to use the inherit flag here.)

The version number of the LOCK record determines the set of RPKI object types that it protects. Version 0 protects the LOCK record itself, ROAs, (subordinate) CA certificates, and router certificates (if present).

The algorithm and subjectPublicKey fields in the publicKey MUST conform to the guidance in Section 3 of [RFC6485].

If an RP elects to process a LOCK record, it verifies the signature on the record using the procedure described in [RFC6488]. If the signature verification fails, it ignores the record. (If the RP has a previously validated LOCK record, it continues to use that record instance.)

If the signature verification succeeds, the RP extracts the version number and verifies that the RP is prepared to process this version of the record. If not, it ignores the record. If it is prepared to process this version, it extracts the URL and public key fields. The URL is used to fetch the corresponding INRD file, and the public key is used to verify the signature on that file.

If the RP has a copy of an INRD file for this publication point, and if the RP detects no material changes to the protected records at the publication point, the RP SHOULD NOT fetch the INRD file. (A material change is one that affects the semantics of the object. For example, for a ROA, only changes to the prefixes and/or ASN are material.) If the RP does not hold a copy of the INRD file, or if a protected record has changed, the RP fetches a new INRD file using the URL, and proceeds as described in Section 3.2.

When an INR holder has outsourced management of its RPKI CA function and publication point, it is susceptible to attacks in which the LOCK record itself is changed. This is because the entity providing these

functions could create a new LOCK record containing a new URL and public key, thus defeating the LOCK/INRD mechanism. An authorized change to the content of a LOCK record should be very rare. A location selected as a home for an INRD file should be stable, and thus the URL should rarely change. The public key used to verify the signature on an INRD file should also be constant for long intervals. The LOCK record contains a flag that indicates whether the INR holder has outsourced CA and publication point management. If this flag is FALSE, an RP will accept changes to the LOCK record (see Section 5) just as it would changes to any other object at a protected publication point. If the flag is TRUE, then any change to a LOCK record is regarded as suspicious by RPs. In such cases the RP delays accepting the new LOCK record and associated INRD file, as discussed in Section 5.

3.2. INRD File Format and Semantics

The INRD file is a DER-encoded ASN.1 file that contains data associated with a single INR holder (publication point owner). The file is encoded using ASN.1, since most of the values it holds will be compared to data from RPKI objects that also are ASN.1 encoded, and because it is a signed object.

```
INRD ::= SEQUENCE {
    tbsINRD    TBSINRD,
    algorithm  AlgorithmIdentifier,
    signature  OCTET STRING
    }

TBSINRD ::= SEQUENCE {
    version       [0] INTEGER DEFAULT 0,
    lastChange        UTCTime,
    changeWindow      ENUMERATED
        {
        1week (7) DEFAULT
        2week (14)
        4week (28)
        },
    additions     [1] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    deletions     [2] SEQUENCE SIZE (1..MAX) OF
                          ProtectedObject OPTIONAL,
    keyRollover   [3] OCTET STRING OPTIONAL,
    algRollover   [4] OCTET STRING OPTIONAL
    }

ProtectedObject ::= CHOICE {
    cmsObject    [0] EncapsulatedContentInfo,
    rtrCert      [1] RtrCertInfo,
    cACert       [2] CACertInfo
    }

RtrCertInfo ::= SEQUENCE {
    subjKeyId  OCTET STRING,
    aSNum      INTEGER
    }

CACertInfo ::= SEQUENCE {
    subjKeyId          OCTET STRING,
    ipAddrBlocks   [0] IPAddrBlocks OPTIONAL,
    aSIdentifiers  [1] ASIdentifiers OPTIONAL
    }

-- See [RFC3779] for the definitions of IPAddrBlocks and
-- ASIdentifiers.
```

The lastChange and changeWindow values are used to bound the set of additions and deletions stored in an INRD file. The lastChange is the timestamp of the most recent addition or deletion in the INRD file; the changeWindow determines the oldest time at which changes to protected objects at the publication point are represented in the

additions and deletions portions of the file. The default is a one week window (i.e., the least recent addition or deletion is no older than one week before lastChange), but two and four week values also may be expressed.

The additions element is used by an INR holder to list all protected objects that have been added to the publication point over the interval defined by the change window. If no objects have been added during this interval, the element is omitted. Similarly, the deletions element is used by an INR holder to list all protected objects that have been removed from the publication point over the interval defined by the change window. If no objects have been removed during this interval, the element is omitted. A substantial change to a protected object is considered to be a deletion followed by an addition. Therefore, the version of the object prior to the change is listed in the deletions element and the version of the object after the change is listed in the additions element. To prevent race conditions, a CA MUST list additions and/or deletions in the INRD before those additions and/or deletions are visible at the CA's publication point.

A LOCK or ROA is listed in the additions and/or deletions fields by putting its EncapsulatedContentInfo in the cmsObject field of a ProtectedObject. A router certificate is listed by putting its SKI and AS number in the rtrCert field. A CA certificate is listed by putting its SKI and [RFC3779] resources in the cACert field. If the outsourced flag in the LOCK record is FALSE, then no CA certificates should be included in the additions or deletions elements. If any CA certificates are included in these elements, they are ignored. RPs SHOULD accept all valid CA certificates issued at this publication point when the outsourced flag is FALSE.

The key rollover element is present only during the time when a key rollover [RFC6489] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" key. During key rollover each CA will have its own LOCK record, that points to its own INRD file. The old CA will list the new CA's SKI here; the new CA will not include this field. Key rollover is a transient condition, so the need for both LOCK records and INRD files ought not be very long.

The algorithm rollover element is present during the time when an algorithm rollover [RFC6916] is taking place. It signals to RPs that an additional set of objects exist that would ordinarily be viewed as competing with the objects protected by this INRD file. The SKI contained here is that of the CA for the "other" algorithm. During

algorithm rollover each CA will have its own LOCK record, that points to its own INRD file, and each of them will list the other CA's SKI here. (Note that the SKI value is compared against the SKI in the CA certificate in question. An RP does not compute an SKI. This means that changes to the hash algorithm used to compute an SKI do not affect how an RP processes this field. An RP MUST be prepared to deal with an SKI length other than the 20 octet value in common use today.) Algorithm transition is a long process, so both sets of LOCK records and INRD files will persist for an extended period.

An RP fetches an INRD file using a URL from a LOCK record, as noted above. If the file cannot be located, the RP software logs an error and regards any changes to the publication point as suspicious. If the file is located, the RP verifies the signature on the file using the public key (and indicated algorithms) from the same LOCK record. If the signature fails, the RP software logs an error and regards any changes to the publication point as suspicious. If the signature is valid, the RP extracts the data elements from the INRD file.

If this is the first time that an INRD file is fetched for this publication point, the file is accepted, and its content is used to populate the RP's expanded local cache. If the INRD file is replacing a previously acquired instance for this publication point, the content is used to confirm changes to protected objects at this publication point. If an RP detects changes to protected publication point objects that occurred after the lastChange time, these changes are treated as suspicious.

Authorizing changes to subordinate CA certificates in an INRD file is critical when an INR holder outsources CA and publication point management. Listing these CAs and their associated 3779 extension data enables an RP to detect creation of unauthorized CAs that could then create competing ROAs or router certificates. However, if an INR holder operates its own CA and manages its publication point, it is not necessary to protect against such attacks. To signal this situation, the "outsourced" flag in the LOCK record is set to FALSE. Under this condition, an RP will not impose change control checks on subordinate CA certificates for the publication point.

Some classes of INR holders need not publish a LOCK record and INRD file. IANA, RIRs, and NIRs, are principally delegators of resources. Each of these RPKI entities SHOULD create one publication point for the resources used by the entity for its own networking needs, and a separate publication point under which all resource delegations take place. The first publication point MAY be protected by a LOCK record, so that ROAs and router certificates associated with those resources can be protected. However, the second publication point OUGHT not include a LOCK record. If this convention is followed,

these classes of INR holders need not update an INRD file every time a new subordinate CA is created or modified, as a result of delegation. If an INR holder follows this convention, and includes a LOCK record in its superior publication point, that record, and the associated INRD file, conveys some degree of protection for the subordinate CA resources, even if the INR holders of these resources do not publish LOCK records.

4. Self-checking by RPs

It is easy for an INR holder, acting as an RP, to determine if any of its resource bindings have been undermined via the RPKI. It knows what resources it holds, and what assertions it has made relevant to those resources. Any conflicting RPKI objects represent a problem! It is more difficult for an RP to detect problems with another INR holder's resources, because it lacks the knowledge that the other INR holder has. The mechanisms described in Section 5 are designed to enable RPs to detect these problems. This section describes the procedures each RP executes to detect adverse changes to its own data in the RPKI repository system. Note that the procedures in this section do not require use of the LOCK record or INRD file.

When and INR downloads RPKI data, as it normally does, it SHOULD perform the checks noted below, to detect problems. To enable such checking, each INR holder's RP software MUST be configured with data about the ROAs, and other protected objects, of this INR holder. If any of these objects are missing or fail to validate, then the INR holder has detected a problem, and is notified.

The semantics of ROAs require an additional check; if other ROAs for the same or more specific prefixes are found anywhere in the RPKI repository system, this too indicates a problem, and the INR holder is notified.

The semantics of router certificates, require a separate, additional check. A router certificate binds a public key (and a router ID) to an ASN. Thus, if an INR holder discovers router certificates for its ASN, that it did not authorize, this indicates a problem.

As additional objects are protected via this mechanism, it will be necessary to perform additional checks to detect the latter sort of adverse changes, based on the semantics of the protected objects.

In any case, RP software SHOULD inform the INR holder of the apparent cause and source of the problem, e.g., a revoked or expired certificate or a manifest problem, and guide the INR holder to the responsible CAs (e.g., using Ghostbusters [RFC6493] records).

When an INR holder is alerted to a change adversely affecting its own resources, it is expected to contact the appropriate RPKI entities to rectify the error in a timely fashion. If the changes are determined to be intentional (and not authorized by the INR holder), the INR holder can inform the Internet operations community (via an out of band mechanism), which can then decide, individually, how to respond.

Remedying a problem detected by an INR holder is not likely to be instantaneous, even if the problem is just an error. To avoid adversely affecting routing, the mechanisms described in Section 5 enable RPs to detect a change that adversely affects INR holders, and to reject it, reverting to previously validated INR data. This gives the INR holder time to resolve the problem. Reverting to an earlier version of INR data is conceptually easy for RPs, because they already cache RPKI data. The mechanisms described below require augmenting the RPKI local cache maintained by each RP, to detect adverse changes, making use of information gleaned from LOCK records and INRD files. The next section describes how the LOCK and INRD data is used.

5. Detection & Remediation

The design described in this section assumes that an RP has acquired a snapshot of the RPKI repository, validated and extracted INR holding and binding data, and considers this data to be "good". The detection and remediation algorithm is initialized by acquiring a complete download of RPKI repository data, and by fetching INRD files for all publication points that contain a LOCK record. (Prior to this initialization step, it is not possible for an RP to detect and respond to adverse changes to the RPKI, using the technique described below.)

Each RP already maintains a cache of RPKI data [RFC6480], [RFC6481]; this document extends that cache. For every publication point that contains a LOCK record, the content of that record, and the corresponding INRD file content, become part of the data maintained by each RP.

An RP acquires and validates all changed RPKI objects as usual. An RP does not update its cache with the changes, until additional checks, described below, are performed. Before accepting any changes the RP MUST process every pub point where there is (or was) a LOCK record. For each of these pub points, if there are changes to protected objects, these changes must be confirmed by the corresponding INRD file before they are accepted. If any of these checks fail, the changes are held in escrow, waiting for a timeout (or an updated INRD file?). After all protected pub point changes have been processed, then changes for unprotected pub point can be

accepted. The checks will detect pending changes that would whack or compete with protected objects, and place them in escrow.

After validating all changed objects downloaded from the RPKI repository, an RP performs the following additional checks for every publication point that has (or had) a LOCK record:

- ROA and router certificate whacking
- INR sub-delegation changes
- ROA competition
- router certificate competition
- LOCK record changes

A ROA or a router certificate has been whacked (see Appendix A) if it was valid and is now missing or invalid, and if it is not indicated as deleted in the INRD file of its issuer. Any previously valid ROA that is no longer valid (or missing) is checked against the INRD file for the ROA issuer, to determine if the ROA or (router certificate) certificate has been legitimately revoked/removed. If the INRD file confirms the action, the old ROA (or router certificate) is removed from the local cache. If not, the old ROA (or router certificate) is retained, but marked as suspicious.

Changes to INR sub-delegation occur when the INR holder issues a new CA certificate, an existing child CA certificate expires, or any other change affects the status of a child CA certificate. These changes are accepted by an RP only if they are confirmed in the INR holder's INRD file.

A newly issued ROA is in competition (see Appendix A) with an existing ROA if the new ROA specifies the same or a more specific prefix than the older ROA, the new ROA is not issued by one of the existing ROA's issuer's descendants, and the new ROA was not authorized by the INRD file of the existing ROA. A competing ROA is not accepted as valid by an RP.

A newly-issued router certificate competes with an existing router certificate, if the new certificate includes the same ASN and was not authorized by the INRD file covering the existing router certificate. A competing router certificate is not accepted as valid by an RP. (Such a certificate would be accepted if the INRD file of the issuer of the original certificate indicates that the old certificate has been deleted, and not replaced with a new router certificate

associated with the same entity. In this case, the newly-issued certificate would not be in competition.)

As noted above, any change to a LOCK record is viewed as suspicious unless the outsourced flag is FALSE. If the record is for a publication point that is not outsourced, then a changed LOCK record is accepted as valid if the corresponding INRD file authorizes the new record. (If the INR holder has changed the public key for the INRD file, it is RECOMMENDED that the URL also change. This allows the INR holder to publish a new INRD file that authorizes the new LOCK record, minimizing the potential race condition between updating an INRD file and a LOCK record.)

If the LOCK record shows that the publication point is outsourced, an RP examines the changes made to the LOCK record. If the URL has changed, but the public key and the outsource flag are unchanged, the new LOCK record may be accepted, if the new INRD file authorizes the change. If not, the new LOCK record is rejected. If the public key has been changed, a delay is imposed on accepting the new LOCK record, even if the INRD file authorizes the change. (should we establish a global delay, or should each INR holder publish its own delay preference in the INRD file?)

Remediation for all of the whacking and competition events consists of NOT making a change in the local cache when an unconfirmed change is encountered.

6. INRD Management Scenarios

Common wisdom notes that we cannot choose our parents, but we can choose our friends, and we should do so wisely. In the RPKI context, and INR holder cannot, generally choose its CA, but Suspenders allows the INR holder to choose its INRD file server. It should do so wisely.

An INRD file is published outside of the RPKI repository system, and is verified using a public key that is also independent of the RPKI. The motivation for these two measures is to insulate this part of the Suspenders system from possible manipulation by an entity to whom CA and publication point services have been outsourced. If an INR holder acts as its own CA, and manages its own publication point, it can publish its INRD file on the same machines as its publication point, but not in the publication point. In this case the independence features are not critical, but they also don’t cause harm for this class of INR holder.

Every INR holder needs to choose a location for the INRD file that is highly available. When an INR holder has outsourced CA and

publication point management, independent publication of the INRD file is critical. The INR holder needs to choose a location for the INRD file that is highly available. It also is appropriate to consider placing the file outside of the geopolitical region in which the INR holder (and its RIR) operate. Here too the motivation is to insulate the INR holder from a malicious action by the CA service provider, or, perhaps, an RIR above it.

Organizations may arise to offer hosting for INRD files, as a service for INR holders. They could offer just file storage, or they might offer more extensive services. For example, an organization might monitor an INR holder's publication point and create the INRD file data, and even sign it for the INR holder. (In this case the organization would provide the public key to the INR holder for inclusion in the LOCK record.) Various other arrangements between the INR holder and a organization that assists in managing INRD files are possible, and are a local matter between the INR holder and the organization.

A country might elect to mandate use of Suspenders, as a means to protect the INRs of its ISPs and other organizations that run BGP with in the country. The motivation is similar to that cited above, i.e., protecting INRs against errors or malicious actions by RPKI entities. In this case the country itself generally is not an INR holder per se, so the relationship is somewhat different from that discussed above. Nonetheless, the mechanisms described above apply.

For example, Elbonia might mandate that every INR holder within the country make use of Suspenders. Every Elbonian INR holder will be required to include a LOCK record in its publication point, no matter where that publication point is realized. The URL in each LOCK points to a file on a server managed by an Elbonian government organization. Each Elbonian ISP would be required to follow the procedures described in Section 5, when managing its local cache.

7. IANA Considerations

This document registers the following in the "RPKI Signed Object" registry created by [RFC6488]:

Name: LOCK
OID: 1.2.840.113549.1.9.16.1.XX
Reference: [RFCxxxx] (this document)

This document also registers the following three-letter filename extension in the "RPKI Repository Name Schemes" registry created by [RFC6481]:

```
Filename extension: lck
RPKI Object: LOCK
Reference: [RFCxxxx] (this document)
```

8. Security Considerations

This document specifies Suspenders, a set of security-focused mechanisms designed to protect INR holders against accidental and malicious changes to RPKI repository data, and to enable RPs to detect and respond to such changes. More text to be provided later.

9. Acknowledgements

Richard Barnes provided the motivation to develop Suspenders, after he identified a problem with the LTAM [I-D.ietf-sidr-ltamgmt] design.

10. References

10.1. Informative References

[I-D.ietf-sidr-lta-use-cases]
Bush, R., "RPKI Local Trust Anchor Use Cases", draft-ietf-sidr-lta-use-cases-03 (work in progress), June 2015.

[I-D.ietf-sidr-ltamgmt]
Reynolds, M., Kent, S., and M. Lepinski, "Local Trust Anchor Management for the Resource Public Key Infrastructure", draft-ietf-sidr-ltamgmt-08 (work in progress), April 2013.

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

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <http://www.rfc-editor.org/info/rfc6481>.

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

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493, February 2012, <http://www.rfc-editor.org/info/rfc6493>.

10.2. Normative References

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

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <http://www.rfc-editor.org/info/rfc3779>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)", RFC 6485, DOI 10.17487/RFC6485, February 2012, <http://www.rfc-editor.org/info/rfc6485>.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <http://www.rfc-editor.org/info/rfc6488>.

[RFC6916] Gagliano, R., Kent, S., and S. Turner, "Algorithm Agility Procedure for the Resource Public Key Infrastructure (RPKI)", BCP 182, RFC 6916, DOI 10.17487/RFC6916, April 2013, <http://www.rfc-editor.org/info/rfc6916>.

Appendix A. RPKI Object Whacking and Competition

There are two ways that an RPKI object can be adversely affected. We term these actions "whacking" and "competition."

Any object in the RPKI can become invalid or inaccessible (to RPs) via various actions by CAs and/or publication point maintainers along the certificate path from the object’s EE certificate to a trust anchor (TA). Any action that causes an object to become invalid or inaccessible is termed "whacking". Revocation of the EE certificate for an object whacks it. Revocation of any CA certificate along the certificate path for the object (without reissuance) has the same effect. Reissuance of any CA certificate along the certificate path, with changes in [RFC3779] extensions in any of these certificates to exclude the resources cited in a targeted object, also constitutes whacking. Changing a manifest along the certificate path might whack an object (depending on how RPs deal with manifest changes), and

removing an object from the RPKI repository system also potentially whacks it. Unless an action that causes an object to be whacked is authorized by the creator of an object, whacking is an attack against the INR holder that created the whacked object.

A different form of attack is termed object "competition". The details of object competition are determined by the semantics of the object. In the general case, one object competes with another object (of the same type), if the newer object creates a binding that adversely affects the binding expressed in the original object. So, for example, a newly issued ROA competes with an existing ROA if the new ROA contains the same or more specific prefixes than the older ROA. Competition does not always indicate an attack; the transfer of resources in a "make before break" model implies ROA competition. A newly issued router certificate competes with a previously issued one if the new certificate binds the same ASN to a public key issued by a different entity. (If key rollover or algorithm transition is in progress, such competition is explicitly authorized via the INRD file.)

Competition that is not authorized by the issuer of the original router certificate is viewed as an attack against that certificate.

Appendix B. Design Criteria (do we still need this section?)

Several criteria were employed in developing the mechanisms described in this document.

1. It is anticipated that object whacking and competition, and analogous forms of errors that adversely impact INR holders, will be infrequent. Thus the detection mechanisms employed by RPs to detect such anomalies ought to be efficient (in terms of data fetching, processing, and storage) for the normal case.

2. RPs may elect to ignore/reject adverse changes to objects if they perceive such changes as suspicious. If an RP elects to reject a change to an object it must have access to previously validated objects for the INR holder question.

3. Transfers of "live" address space will occur, although not frequently. INR holders engaged in such transfers must be able to signal to RPs that such transfers are authorized, so that the transfers are not rejected as suspicious.

4. Routes for a prefix may be legitimately originated by more than one AS (MOA). The design MUST enable an INR holder to inform RPs when this situation is authorized.

5. Many INR holders may choose to outsource CA and publication point management functions. INR holders who choose to outsource these functions should be offered equivalent protection against ROA invalidation and competition as INR holders who perform these functions for themselves.

6. Any new RPKI repository objects used with the mechanisms defined here MUST conform to the format specified in [RFC6488].

7. The decision to process any additional data associated with the mechanisms described in this document is local to each RP. RPs that choose to not implement these mechanisms will incur minimal additional data fetching, storage, and processing burdens.

8. The decision to employ the mechanisms described here to protect INR holdings and binding is a local one made by each INR holder. (INR holders who outsource CA and publication point management functions will require the providers of these services to support creation and publication of one new RPKI object. As a result, all such providers must support generation and maintenance of the new RPKI object so that their clients have the option to utilize these capabilities.)

9. Revocation and expiration of RPKI object MUST continue to work as they do currently, for all objects that have not been adversely affected.

Authors' Addresses

Stephen Kent
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: kent@bbn.com

David Mandelberg
BBN Technologies
10 Moulton St.
Camridge, MA 02138
US

Email: david@mandelberg.org