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

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: June 14, 2018

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

December 11, 2017

BGPsec Router Certificate Rollover
draft-ietf-sidrops-bgpsec-rollover-04

Abstract

Certification Authorities (CAs) within the Resource Public Key Infrastructure (RPKI) manage BGPsec router certificates as well as RPKI certificates. The rollover of BGPsec router certificates must be carefully performed in order to synchronize the distribution of router public keys with BGPsec Update messages verified with those router public keys. This document describes a safe rollover process, as well as discussing when and why the rollover of BGPsec router certificates are necessary. When this rollover process is followed the rollover will be performed without routing information being lost.

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 14, 2018.

Table of Contents

1. Requirements notation

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

2. Introduction

In BGPsec, a key rollover (or re-key) is the process of changing a router's BGPsec key pair (or key pairs), issuing the corresponding new BGPsec router certificate and (if the old certificate is still valid) revoking the old certificate. This process will need to happen at regular intervals, normally due to policies of the local network. This document describes a safe rollover process that results in a BGPsec receiver always having the needed verification keys. Certificate Practice Statements (CPS) documents may reference this memo. This memo only addresses changing of a router's BGPsec key pair within the RPKI. Refer to [RFC6489] for a procedure to rollover RPKI Certification Authority key pairs.

When a router receives or creates a new key pair (using a key provisioning mechanism), this key pair will be used to sign new BGPsec updates [RFC8205] that are originated or that transit through the BGP speaker. Additionally, the BGP speaker will refresh its outbound BGPsec Update messages to include a signature using the new key (replacing the old key). When the rollover process finishes, the old BGPsec router certificate (and its key) will no longer be valid, and thus any BGPsec Update that includes a signature performed by the old key will be invalid. Consequently, if the router does not refresh its outbound BGPsec Update messages, previously sent routing information may be treated as unauthenticated after the rollover process is finished. It is therefore extremely important that new BGPsec router certificates have been distributed throughout the RPKI before the router begin signing BGPsec updates with a new private key.

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

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

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

3. Key rollover in BGPsec

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

Scheduled rollover: BGPsec router certificates have an expiration date (NotValidAfter) that requires a frequent rollover process to refresh certificates or issue new certificates. The validity period for these certificates is typically expressed in the CA's CPS document.

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

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

Protection against withdrawal suppression and replay attacks: An AS may determine that withdrawn BGPsec updates are being propagated instead of the most recently propagated BGPsec updates. Changing the BGPsec router signing key, distributing a new BGPsec router certificate, and revoking the old BGPsec router certificate will invalidate the replayed BGPsec updates.

In some of these cases it is possible to generate a new certificate without changing the key pair. This practice simplifies the rollover process as the BGP speakers receiving BGPsec Updates do not even need to be aware of the change of certificate. However, not replacing the certificate key for a long period of time increases the risk that a compromised router private key may be used by an attacker to deliver unauthorized or false BGPsec Updates. Distributing the old public key in a new certificate is NOT RECOMMENDED when the rollover event is due to a compromised key, or when it is suspected that withdrawn BGPsec updates are being distributed.

3.1. Rollover Process

The key rollover process is dependent on the key provisioning mechanisms adopted by an AS [I-D.ietf-sidr-rtr-keying]. An automatic provisioning mechanism such as EST will allow router key management procedures to include automatic re-keying methods with minimum development cost.

A safe BGPsec router key rollover process is as follows.

1. New Certificate Publication: The first step in the rollover mechanism is to publish the new certificate. If required, a new key pair will be generated for the BGPsec router. A new certificate will be generated and the certificate published at the appropriate RPKI repository publication point. The details of this process will vary as they depend on whether the keys are

assigned per-BGPsec speaker or shared among multiple BGPsec speakers, whether the keys are generated on each BGPsec speaker or in a central location, and whether the RPKI repository is locally or externally hosted.

2. Staging Period: A staging period will be required from the time a new certificate is published in the RPKI global repository until the time it is fetched by RPKI caches around the globe. The exact minimum staging time will be dictated by the conventional interval chosen between repository fetches. If rollovers will be done more frequently, an administrator can provision two certificates for every router concurrently with different valid start times. In this case when the rollover operation is needed, the relying parties around the globe would already have the new router public keys. However, if an administrator has not previously provisioned the next certificate then a staging period may not be possible to implement during emergency key rollover. If there is no staging period, routing may be disrupted due to the inability of a BGPsec router to validate BGPsec updates signed with a new private key.

3. Twilight: At this moment, the BGPsec speaker holding the rolled-over private key will stop using the old key for signing and start using the new key. Also, the router will generate appropriate refreshed BGPsec updates just as in the typical operation of refreshing out-bound BGP polices. This operation may generate a great number of BGPsec updates. A BGPsec speaker may vary the Twilight moment for every peer in order to distribute the system load (e.g., skewing the rollover for different peers by a few minutes each would be sufficient and effective).

4. Certificate Revocation: This is an optional step, but SHOULD be taken when the goal is to invalidate BGPsec updates signed with the old key. Reasons to invalidate old BGPsec updates include: (a) the AS has reason to believe that the router signing key has been compromised, and (b) the AS needs to invalidate already propagated BGPsec updates signed with the old key. As part of the rollover process, a CA MAY decide to revoke the old certificate by publishing its serial number on the CA’s CRL. Alternatively, the CA will just let the old certificate expire and not revoke it. This choice will depend on the reasons that motivated the rollover process.

5. RPKI-Router Protocol Withdrawals: At the expiration of the old certificate’s validation, the RPKI relying parties around the globe will need to communicate to their router peers that the old certificate’s public key is no longer valid (e.g., using the

RPKI-Router Protocol described in [RFC8210]). A router's reaction to a message indicating withdrawal of a router key in the RPKI-Router Protocol SHOULD include the removal of any RIB entries (i.e., BGPsec updates) signed with that key and the generation of the corresponding BGP WITHDRAWALs (either implicit or explicit).

This rollover mechanism depends on the existence of an automatic provisioning process for BGPsec router certificates. It requires a staging mechanism based on the RPKI propagation time (typically a 24 hour period at the time this document was published), and an AS is REQUIRED to re-sign all originated and transited BGPsec updates that were previously signed with the old key.

The first two steps (New Certificate Publication and Staging Period) may happen in advance of the rest of the process. This will allow a network operator to perform its subsequent key rollover in an efficient and timely manner.

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

4. BGPsec router key rollover as a measure against replay attacks

There are two typical generic measures to mitigate replay attacks in any protocol: the addition of a timestamp or the addition of a serial number. However, neither BGP nor BGPsec provide either measure. The timestamp approach was originally proposed for BGPsec [I-D.sriram-replay-protection-design-discussion] but later dropped in favor of the key rollover approach. This section discusses the use of using a key rollover as a measure to mitigate replay attacks.

4.1. BGP UPDATE window of exposure requirement

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

The BGPsec Operational Considerations document [RFC8207] gives some idea of requirements for the size of the window of exposure to replay

attacks. It states that the requirement will be in the order of a day or longer.

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

Since the window requirement is on the order of a day (as documented in [RFC8207]) and the BGP speaker performing re-keying is the edge router of the origin AS, it is feasible to use key rollover to mitigate replays. In this case it is important to complete the full process (i.e., the old and new certificates do not share the same key). By re-keying, an AS is letting the BGPsec router certificate validation time be a type of "timestamp" to mitigate replay attacks. However, the use of frequent key rollovers comes with an additional administrative cost and risks if the process fails. As documented before, re-keying should be supported by automatic tools, and for the great majority of the Internet it will be done with good lead time to ensure that the public key corresponding to the new router certificate will be available to validate the corresponding BGPsec updates when received.

If a transit AS also originates BGPsec updates for its own prefixes and it wishes to mitigate replay attacks on those prefixes, then the transit AS SHOULD be provisioned with two unique key pairs and certificates. One of the key pairs is used to sign BGPsec updates for prefixes originated from the transit AS, and can have a replay protection policy applied to it. The other key pair is used to sign BGPsec updates in transit and SHOULD NOT have replay protection policy applied to it. Because the transit AS is not likely to know or care what is the policy of origin ASes elsewhere, there is no value for the transit AS to perform key rollovers to mitigate replay attacks against prefixes originated elsewhere. If the transit AS were instead to perform replay protection for all updates that it signs, its key rollover process would generate a large number of BGPsec UPDATE messages, even in the complete Default Free Zone (DFZ). Therefore, it is best to let each AS independently manage the replay attack vulnerability window for the prefixes it originates.

Advantages to re-keying as replay attack protection mechanism are as follows:

1. All expiration policies are maintained in the RPKI.

2. Much of the additional administrative cost is paid by the provider that wants to protect its infrastructure, as it bears the cost of creating and initiating distribution of new router key pairs and BGPsec router certificates. (It is true that the cost of relying parties will be affected by the new objects, but

their responses should be completely automated or otherwise routine.)

3. The re-keying can be implemented in coordination with planned topology changes by either origin ASes or transit ASes (e.g., if an AS changes providers, it completes a key rollover).

Disadvantages to Re-keying as replay attack protection mechanism are as follows:

1. Frequent rollovers add administrative and BGP processing loads, although the required frequency is not clear. Some initial ideas are found in [RFC8207].

2. The minimum replay vulnerability is bounded by the propagation time for RPKI caches to obtain the new certificate and CRL (2x propagation time because first the new certificate and then the CRL need to propagate through the RPKI system). If provisioning is done ahead of time, the minimum replay vulnerability window size is reduced to 1x propagation time (i.e., propagation of the CRL). However, these bounds will be better understood when RPKI and RPs are well deployed, as well as the propagation time for objects in the RPKI is better understood.

3. Re-keying increases the dynamics and size of the RPKI repository.

5. IANA Considerations

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

6. Security Considerations

This document does not contain a protocol update to either the RPKI or BGPsec. It describes a process for managing BGPsec router certificates within the RPKI.

Routers participating in BGPsec will need to rollover their signing keys as part of conventional certificate management processes. However, because rolling over signing keys will also have an effect of invalidating BGPsec updates signatures, the rollover process must be carefully orchestrated to ensure that valid BGPsec updates are not treated as invalid. This situation could affect Internet routing. This document describes a safe method for rolling over BGPsec router certificates. It takes into account both normal and emergency key rollover requirements.

Additionally, the key rollover method described in this document can be used as a measure to mitigate BGP update replay attacks, in which an entity in the routing system is suppressing current BGPsec updates and replaying withdrawn updates. When the key used to sign the withdrawn updates has been rolled over, the withdrawn updates will be considered invalid. When certificates containing a new public key are provisioned ahead of time, the minimum replay vulnerability window size is reduced to the propagation time of a CRL invalidating the certificate containing an old public key. For a discussion of the difficulties deploying a more effectual replay protection mechanism for BGPSEC, see [I-D.sriram-replay-protection-design-discussion].

7. Acknowledgments

Randy Bush, Kotikalapudi Sriram, Stephen Kent and Sandy Murphy each provided valuable suggestions resulting in an improved document. Kotikalapudi Sriram contributed valuable guidance regarding the use of key rollovers to mitigate BGP update replay attacks.

8. References

8.1. Normative References

[I-D.ietf-sidr-rtr-keying]
Bush, R., Turner, S., and K. Patel, "Router Keying for BGPsec", draft-ietf-sidr-rtr-keying-14 (work in progress), October 2017.

[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>.

[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>.

8.2. Informative References

[I-D.sriram-replay-protection-design-discussion]
Sriram, K. and D. Montgomery, "Design Discussion and Comparison of Protection Mechanisms for Replay Attack and Withdrawal Suppression in BGPsec", draft-sriram-replay-protection-design-discussion-09 (work in progress), October 2017.

[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, <https://www.rfc-editor.org/info/rfc6489>.

[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>.

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

[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>.

[RFC8207] Bush, R., "BGPsec Operational Considerations", BCP 211, RFC 8207, DOI 10.17487/RFC8207, September 2017, <https://www.rfc-editor.org/info/rfc8207>.

[RFC8210] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", RFC 8210, DOI 10.17487/RFC8210, September 2017, <https://www.rfc-editor.org/info/rfc8210>.

Authors' Addresses

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

Email: bew@cisco.com

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

Email: rogaglia@cisco.com

Keyur Patel
Arrcus, Inc.

Email: keyur@arrcus.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 4, 2019

R. Bush
Internet Initiative Japan

October 1, 2018

Use Cases for Localized Versions of the RPKI
draft-ietf-sidrops-lta-use-cases-04

Abstract

There are a number of critical circumstances where a localized routing domain needs to augment or modify its view of the Global RPKI. This document attempts to outline a few of them.

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 https://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 4, 2019.

Copyright Notice

Table of Contents

1. Introduction

Today RPKI-based Origin Validation, [RFC6811], relies on widespread deployment of the Global Resource Public Key Infrastructure (RPKI), [RFC6480]. In the future, RPKI-based Path Validation, [I-D.ietf-sidr-bgpsec-overview], will be even more reliant on the Global RPKI.

But there are critical circumstances in which a local, clearly-scoped, administrative and/or routing domain will want to augment and/or modify their internal view of the Global RPKI.

This document attempts to lay out a few of those use cases. It is not intended to be authoritative, complete, or to become a standard. It merely tries to lay out a few critical examples to help frame the issues.

2. Suggested Reading

It is assumed that the reader understands the RPKI, see [RFC6480], the RPKI Repository Structure, see [RFC6481], Route Origin Authorizations (ROAs), see [RFC6482], and GhostBusters Records, see [RFC6493].

3. What is 'Local'

The RPKI is a distributed database containing certificates, CRLs, manifests, ROAs, and GhostBusters Records as described in [RFC6481]. Policies and considerations for RPKI object generation and maintenance are discussed elsewhere.

Like the DNS, the Global RPKI tries to present a single global view, although only a loosely consistent view, depending on timing,

updating, fetching, etc. There is no 'fix' for this, it is not broken, it is the nature of distributed data with distributed caches.

There are critical uses of the RPKI where a local administrative and/or routing domain, e.g. an end-user site, a particular ISP or content provider, an organization, a geo-political region, ... may wish to have a specialized view of the RPKI.

For the purposes of this exploration, we refer to this localized view as a 'Local Trust Anchor', mostly for historical reasons, but also because implementation would likely require the local distribution of one or more specialized trust anchors, [RFC6481].

4. Example Uses

We explore this space using three examples.

Carol, a resource holder (LIR, PI holder, ...), operates outside of the country in which her RIR is based. Someone convinces the RIR's local court to force the RIR to remove or modify some or all of Carol's certificates, ROAs, etc. or the resources they represent, and the operational community wants to retain the ability to route to Carol's network(s). There is need for some channel through which operators can exchange local trust, command, and data collections necessary to propagate patches local to all their RPKI views.

Bob has a multi-AS network under his administration and some of those ASs use private ([RFC1918]) or 'borrowed' address space which is not announced on the global Internet (not to condone borrowing), and he wishes to certify them for use in his internal routing.

Alice is responsible for the trusted routing for a large organization, commercial or geo-political, in which management requests routing engineering to redirect their competitors' prefixes to socially acceptable data. Alice is responsible for making the CA hierarchy have validated certificates for those redirected resources as well as the rest of the Internet.

5. Some Approaches

In these examples, it is ultimately the ROAs, not the certificates, which one wants to modify or replace. But one probably can not simply create new ROAs as one does not have the private keys needed to sign them. Hence it is likely that one has to also do something about the [RFC6480] certificates.

The goal is to modify, create, and/or replace ROAs and GhostBuster Records which are needed to present the localized view of the RPKI data.

One wants to reproduce only as much of the Global RPKI as needed. Replicating more than is needed would amplify tracking and maintenance.

One can not reissue down from the root trust anchor at the IANA or from the RIRs' certificates because one does not have the private keys required. So one has to create a new trust anchor which, for ease of use, will contain the new/modified certificates and ROAs as well as the unmodified remainder of the Global RPKI.

Because Alice, Bob, and Carol want to be able to archive, reproduce, and send to other operators the data necessary to reproduce their modified view of the global RPKI, there will need to be a formally defined set of data which is input to a well-defined process to take an existing Global RPKI tree and produce the desired modified re-anchored tree.

It is possible that an operator may need to accept and process modification data from more than one source. Hence there is a need to merge modification 'recipes'.

6. Security Considerations

Though the above use cases are all constrained to local contexts, they violate the model of a single global PKI, albeit to meet real operational needs. Hence they MUST be implemented to assure the local constraint.

Authentication of modification 'recipes' will be needed.

7. IANA Considerations

This document has no IANA Considerations.

8. Acknowledgments

The author thanks Chris Morrow, Karen Seo, Rob Austein, and Steve Kent for comments and suggestions.

9. References

9.1. Normative References

[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>.

[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>.

[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>.

[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>.

9.2. Informative References

[I-D.ietf-sidr-bgpsec-overview] Lepinski, M. and S. Turner, "An Overview of BGPSEC", draft-ietf-sidr-bgpsec-overview-02 (work in progress), May 2012.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

Author’s Address

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

Email: randy@psg.com

Network Working Group
Internet-Draft
Updates: 6811 (if approved)
Intended status: Standards Track
Expires: February 21, 2019

R. Bush
Internet Initiative Japan

August 20, 2018

BGP RPKI-Based Origin Validation Clarifications
draft-ietf-sidrops-ov-clarify-05

Abstract

Deployment of Resource Public Key Infrastructure (RPKI) based BGP origin validation is hampered by, among other things, vendor mis-implementations in two critical areas: which routes are validated and whether policy is applied when not specified by configuration. This document is meant to clarify possible misunderstandings causing those mis-implementations; and thus updates RFC 6811 by clarifying that all prefixes should have their validation state set, and that policy must not be applied without operator configuration.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC8174] 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 https://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 February 21, 2019.

1. Introduction

Deployment of RPKI-based BGP origin validation is hampered by, among other things, vendor mis-implementations in two critical areas: which routes are validated and whether policy is applied when not specified by configuration. This document is meant to clarify possible misunderstandings causing those mis-implementations.

When a route is distributed into BGP, the origin validation state is set to NotFound, Valid, or Invalid per [RFC6811]. Operational testing has shown that the specifications of that RFC were not sufficient to avoid divergent implementations. This document attempts to clarify two areas which seem to cause confusion.

The implementation issues seem not to be about how to validate, i.e., how to decide if a route is NotFound, Valid, or Invalid. The issues seem to be which routes should be evaluated and have their evaluation state set, and whether to apply policy without operator configuration.

2. Suggested Reading

It is assumed that the reader understands BGP, [RFC4271], the RPKI, [RFC6480], Route Origin Authorizations (ROAs), [RFC6482], and RPKI-based Prefix Validation, [RFC6811].

3. Evaluate ALL Prefixes

Significant Clarification: A router MUST evaluate and set the validation state of all routes in BGP coming from any source (eBGP, iBGP, or redistribution from static, connected, etc.), unless specifically configured otherwise by the operator. Else the operator does not have the ability to drop Invalid routes coming from every

potential source; and is therefore liable to complaints from neighbors about propagation of Invalid routes. For this reason, [RFC6811] says:

"When a BGP speaker receives an UPDATE from a neighbor, it SHOULD perform a lookup as described above for each of the Routes in the UPDATE message. The lookup SHOULD also be applied to routes that are redistributed into BGP from another source, such as another protocol or a locally defined static route."

[RFC6811] goes on to say "An implementation MAY provide configuration options to control which routes the lookup is applied to."

When redistributing into BGP from connected, static, IGP, iBGP, etc., there is no AS_PATH in the input to allow RPKI validation of the originating AS. In such cases, the router MUST use the AS of the router's BGP configuration. If that is ambiguous because of confederation, AS migration, or other multi-AS configuration, then the router configuration MUST provide a means of specifying the AS to be used on the redistribution, either per redistribution or globally.

4. Set State, Don't Act

Significant Clarification: Once routes are evaluated and have their state set, the operator should be in complete control of any policy applied based on the evaluation state. Absent specific operator configuration, policy MUST NOT be applied.

Automatic origin validation policy actions such as those described in [RFC8097], BGP Prefix Origin Validation State Extended Community, MUST NOT be carried out or otherwise applied unless specifically configured by the operator.

5. Security Considerations

This document does not create security considerations beyond those of [RFC6811].

6. IANA Considerations

This document has no IANA Considerations.

7. Acknowledgments

Many thanks to John Scudder who had the patience to give constructive review multiple times, and to Keyur Patel who noted that the AS might have to be specified. George Michaelson, Jay Borkenhagen, John Heasley, and Matthias Waehlisch kindly helped clean up loose wording.

8. Normative References

[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>.

[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>.

[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>.

[RFC8097] Mohapatra, P., Patel, K., Scudder, J., Ward, D., and R. Bush, "BGP Prefix Origin Validation State Extended Community", RFC 8097, DOI 10.17487/RFC8097, March 2017, <http://www.rfc-editor.org/info/rfc8097>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

Author’s Address

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

Email: randy@psg.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 12, 2017

T. King
D. Kopp
DE-CIX
A. Lambrianidis
AMS-IX
A. Fenioux
France-IX

April 10, 2017

Signaling Prefix Origin Validation Results from a Route Server to Peers
draft-ietf-sidrops-route-server-rpki-light-02

Abstract

This document defines the usage of the BGP Prefix Origin Validation State Extended Community [RFC8097] to signal prefix origin validation results from a route server to its peers. Upon reception of prefix origin validation results peers can use this information in their local routing decision process.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 12, 2017.

Table of Contents

1. Introduction

RPKI-based prefix origin validation [RFC6480] can be a significant operational burden for BGP peers to implement and adopt. In order to boost acceptance and usage of prefix origin validation and ultimately increase the security of the Internet routing system, IXPs may provide RPKI-based prefix origin validation at the route server [RFC7947]. The result of this prefix origin validation is signaled to peers by using the BGP Prefix Origin Validation State Extended Community as introduced in [RFC8097].

Peers receiving the prefix origin validation result from the route server(s) can use this information in their local routing decision

process for acceptance, rejection, preference, or other traffic engineering purposes of a particular route.

2. BGP Prefix Origin Validation State Utilized at Route-Servers

A route server that is aware of a BGP Prefix Origin Validation state for a certain route can handle this information in one of the following modes of operation:

Simple Tagging: The prefix origin validation state is tagged to the route as described in Section 3.
This mode of operation is like the tradional way route servers work, however, the prefix origin validation state information is additionally available for peers.

Dropping and Tagging: Routes for which the prefix origin validation state is "invalid" (according to [RFC6811]) are dropped by the route server. Routes which show a prefix origin validation state of "not found" and "valid" (according to [RFC6811]) are tagged accordingly to Section 3.
Security is higher rated than questionable reachability of a prefix by this mode of operation.

Prioritizing and Tagging: If the route server learned for a particular prefix more than one route it removes firstly the set of "invalid" routes and secondly the "not found" routes unless the set of routes is empty. Based on the set of routes left over the BGP best path section algorithm is executed. The selected route is marked accordinly to Section 3.
The BGP best path selection algorithm is changed by this mode of operation in such a way that "valid" routes are preferred even if they are unfavorable by the traditional best path selection algorithm. This puts prefix origin validation on top of the best path selection.

A route server MUST support the Simple Tagging mode of operation. Other modes of operation are OPTIONAL. The mode of operation MAY be configured by the route server operator for a route server instance or for each BGP session with a peer seperately.

These mode of operations might be used in combination with [RFC7911] in order to allow a peer to receive all routes and take the routing decision by itself.

3. Signaling Prefix Origin Validation Results from a Route Server to Peers

The BGP Prefix Origin Validation State Extended Community (as defined in [RFC8097]) is utilized for signaling prefix origin validation result from a route server to peers.

[RFC8097] proposes an encoding of the prefix origin validation result [RFC6811] as follows:

Value	Meaning
0	Lookup result = "valid"
1	Lookup result = "not found"
2	Lookup result = "invalid"

Table 1

This encoding is re-used. Route servers providing RPKI-based prefix origin validation set the validation state according to the prefix origin validation result (see [RFC6811]).

4. Operational Recommendations

4.1. Local Routing Decision Process

A peer receiving prefix origin validation results from the route server MAY use the information in its own local routing decision process. The local routing decision process SHOULD apply to the rules as described in section 5 [RFC6811].

A peer receiving a prefix origin validation result from the route server MAY redistribute this information within its own AS.

4.2. Route Server Receiving the BGP Prefix Origin Validation State Extended Community

An IXP route server receiving routes from its peers containing the BGP Prefix Origin Validation State Extended Community MUST remove the extended community before the route is re-distributed to its peers. This is required regardless of whether the route server is executing prefix origin validation or not.

Failure to do so would allow opportunistic peers to advertise routes tagged with arbitrary prefix origin validation results via a route

server, influencing maliciously the decision process of other route server peers.

4.3. Information about Validity of a BGP Prefix Origin Not Available at a Route-Server

In case information about the validity of a BGP prefix origin is not available at the route server (e.g., error in the ROA cache, CPU overload) the route server MUST NOT add the BGP Prefix Origin Validation State Extended Community to the route.

4.4. Error Handling at Peers

A route sent by a route server SHOULD only contain none or one BGP Prefix Origin Validation State Extended Community.

A peer receiving a route from a route server containing more than one BGP Prefix Origin Validation State Extended Community SHOULD only consider the largest value (as described in Table 1) in the validation result field and disregard the other values. Values larger than two in the validation result field MUST be disregarded.

5. IANA Considerations

None.

6. Security Considerations

All security considerations described in RFC RFC6811 [RFC6811] fully apply to this document.

Additionally, threat agents polluting ROA cache server(s) run by IXP operators could cause significant operational impact, since multiple route server clients could be affected. Peers should be vigilant as to the integrity and authenticity of the origin validation results, as they are provided by a third party, namely the IXP operator hosting both the route server as well as any ROA cache server(s).

Therefore, a route server could be misused to spread malicious prefix origin validation results. However, peers already trust the route server for the collection, filtering (e.g. IRR database filtering), and redistribution of BGP routing information to other peers. So, no change in the trust level is needed for this proposal.

To facilitate trust and help with peers establishing appropriate controls in mitigating the risks mentioned above, IXPs SHOULD provide out-of-band means for peers to ensure that the ROA validation process has not been compromised or corrupted.

While beeing under DDoS attacks, it is a common practice for peers connected to an IXP to make use of blackholing services (see [RFC7999]). Peers are using blackholing to drop traffic, typically by announcing a more specific prefix, which is under attack. A peer SHOULD make sure that this prefix is covered by an appropriate ROA.

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>.

[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>.

[RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <http://www.rfc-editor.org/info/rfc7911>.

[RFC8097] Mohapatra, P., Patel, K., Scudder, J., Ward, D., and R. Bush, "BGP Prefix Origin Validation State Extended Community", RFC 8097, DOI 10.17487/RFC8097, March 2017, <http://www.rfc-editor.org/info/rfc8097>.

7.2. Informative References

[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>.

[RFC7947] Jasinska, E., Hilliard, N., Raszuk, R., and N. Bakker, "Internet Exchange BGP Route Server", RFC 7947, DOI 10.17487/RFC7947, September 2016, <http://www.rfc-editor.org/info/rfc7947>.

[RFC7999] King, T., Dietzel, C., Snijders, J., Doering, G., and G. Hankins, "BLACKHOLE Community", RFC 7999, DOI 10.17487/RFC7999, October 2016, <http://www.rfc-editor.org/info/rfc7999>.

Authors' Addresses

Thomas King
DE-CIX Management GmbH
Lichtstrasse 43i
Cologne 50825
DE

Email: thomas.king@de-cix.net

Daniel Kopp
DE-CIX Management GmbH
Lichtstrasse 43i
Cologne 50825
DE

Email: daniel.kopp@de-cix.net

Aristidis Lambrianidis
Amsterdam Internet Exchange
Frederiksplein 42
Amsterdam 1017 XN
NL

Email: aristidis.lambrianidis@ams-ix.net

Arnaud Fenioux
France-IX
88 Avenue Des Ternes
Paris 75017
FR

Email: afenioux@franceix.net

SIDROPS
Internet-Draft
Intended status: Informational
Expires: August 1, 2019

D. Ma
ZDNS
S. Kent
Independent

January 28, 2019

Requirements for Resource Public Key Infrastructure (RPKI) Relying Parties
draft-ietf-sidrops-rp-03

Abstract

This document provides a single reference point for requirements for Relying Party (RP) software for use in the Resource Public Key Infrastructure (RPKI) in the context of securing Internet routing. It cites requirements that appear in several RPKI RFCs, making it easier for implementers to become aware of these requirements that are segmented with orthogonal functionalities.

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 https://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 1, 2019.

Table of Contents

1. Introduction

The RPKI Relying Party (RP) software is used by network operators and others to acquire and verify Internet Number Resource (INR) data stored in the RPKI repository system. RPKI data, when verified, allow an RP to verify assertions about which Autonomous Systems (ASes) are authorized to originate routes for IP address prefixes. RPKI data also establishes binding between public keys and BGP routers, and indicates the AS numbers that each router is authorized to represent.

Noting that the essential requirements imposed on RPs to support securing Internet routing ([RFC6480]) are scattered throughout numerous RFC documents that are protocol specific or provide best practices, as follows:

RFC 6481 (Repository Structure)
RFC 6482 (ROA format)
RFC 6486 (Manifests)
RFC 6487 (Certificate and CRL profile)
RFC 6488 (RPKI Signed Objects)
RFC 6489 (Key Rollover)
RFC 6810 (RPKI to Router Protocol)
RFC 6916 (Algorithm Agility)
RFC 7935 (Algorithms)
RFC 8209 (Router Certificates)
RFC 8210 (RPKI to Router Protocol,Version 1)
RFC 8360 (Certificate Validation Procedure)
I-D.ietf-sidrops-https-tal (Trust Anchor Locator)

This makes it hard for an implementer to be confident that he/she has addressed all of these generalized requirements. Besides, software engineering calls for how to segment the RP system into components with orthogonal functionalities, so that those components could be distributed across the operational timeline of the user. Taxonomy of generalized RP requirements is going to help have 'the role of the RP' well framed.

To consolidate RP requirements in one document, with pointers to all the relevant RFCs, this document outlines a set of baseline requirements imposed on RPs and provides a single reference point for requirements for RP software for use in the RPKI, as segmented with orthogonal functionalities:

- Fetching and Caching RPKI Repository Objects
- Processing Certificates and CRLs
- Processing RPKI Repository Signed Objects
- Distributing Validated Cache of the RPKI Data

This document will be update to reflect new or changed requirements as these RFCs are updated, or new RFCs are written.

2. Fetching and Caching RPKI Repository Objects

RP software uses synchronization mechanisms supported by targeted repositories (e.g., [rsync], RRDP [RFC8182]) to download all RPKI changed data objects in the repository system and cache them locally. The software validates the RPKI data and uses it to generate authenticated data identifying which ASes are authorized to originate

routes for address prefixes, and which routers are authorized to sign BGP updates on behalf of ASes.

2.1. TAL Acquisition and Processing

In the RPKI, each RP chooses its own set of trust anchors (TAs). Consistent with the extant INR allocation hierarchy, the IANA and/or the five RIRs are obvious candidates to be default TAs for the RP.

An RP does not retrieve TAs directly. A set of Trust Anchor Locators (TALs) is used by each RP to retrieve and verify the authenticity of each TA.

TAL acquisition and processing are specified in Section 3 of [I-D.ietf-sidrops-https-tal].

2.2. Locating RPKI Objects Using Authority and Subject Information Extensions

The RPKI repository system is a distributed one, consisting of multiple repository instances. Each repository instance contains one or more repository publication points. An RP discovers publication points using the Subject Information Access (SIA) and the Authority Information Access (AIA) extensions from (validated) certificates.

Section 5 of [RFC6481] specifies how an RP locates all RPKI objects by using the SIA and AIA extensions. Detailed specifications of SIA and AIA extensions in a resource certificate are described in Section 4 of [RFC6487].

2.3. Dealing with Key Rollover

An RP takes the key rollover period into account with regard to its frequency of synchronization with RPKI repository system.

RP requirements in dealing with key rollover are described in Section 3 of [RFC6489] and Section 3 of [I-D.ietf-sidrops-bgpsec-rollover].

2.4. Dealing with Algorithm Transition

The set of cryptographic algorithms used with the RPKI is expected to change over time. Each RP is expected to be aware of the milestones established for the algorithm transition and what actions are required at every juncture.

RP requirements for dealing with algorithm transition are specified in Section 4 of [RFC6916].

2.5. Strategies for Efficient Cache Maintenance

Each RP is expected to maintain a local cache of RPKI objects. The cache needs to be as up to date and consistent with repository publication point data as the RP's frequency of checking permits.

The last paragraph of Section 5 of [RFC6481] provides guidance for maintenance of a local cache.

3. Certificate and CRL Processing

The RPKI make use of X.509 certificates and CRLs, but it profiles these standard formats [RFC6487]. The major change to the profile established in [RFC5280] is the mandatory use of a new extension to X.509 certificate [RFC3779].

3.1. Verifying Resource Certificate and Syntax

Certificates in the RPKI are called resource certificates, and they are required to conform to the profile [RFC6487]. An RP is required to verify that a resource certificate adheres to the profile established by Section 4 of [RFC6487]. This means that all extensions mandated by Section 4.8 of [RFC6487] must be present and value of each extension must be within the range specified by this RFC. Moreover, any extension excluded by Section 4.8 of [RFC6487] must be omitted.

Section 7.1 of [RFC6487] gives the procedure that the RP should follow to verify resource certificate and syntax.

3.2. Certificate Path Validation

In the RPKI, issuer can only assign and/or allocate public INRs belong to it, thus the INRs in issuer's certificate are required to encompass the INRs in the subject's certificate. This is one of necessary principles of certificate path validation in addition to cryptographic verification i.e., verification of the signature on each certificate using the public key of the parent certificate).

Section 7.2 of [RFC6487] gives the procedure that the RP should follow to perform certificate path validation.

Certificate Authorities that want to reduce aspects of operational fragility will migrate to the new OIDs [RFC8360], informing the RP of using an alternative RPKI validation algorithm. An RP is expected to support the amended procedure to handle with accidental over-claim.

3.3. CRL Processing

The CRL processing requirements imposed on CAs and RP are described in Section 5 of [RFC6487]. CRLs in the RPKI are tightly constrained; only the AuthorityKeyIndetifier and CRLNumber extensions are allowed, and they MUST be present. No other CRL extensions are allowed, and no CRLEntry extensions are permitted. RPs are required to verify that these constraints have been met. Each CRL in the RPKI MUST be verified using the public key from the certificate of the CA that issued the CRL.

In the RPKI, RPs are expected to pay extra attention when dealing with a CRL that is not consistent with the Manifest associated with the publication point associated with the CRL.

Processing of a CRL that is not consistent with a manifest is a matter of local policy, as described in the fourth paragraph of Section 6.6 of [RFC6486].

4. Processing RPKI Repository Signed Objects

4.1. Basic Signed Object Syntax Checks

Before an RP can use a signed object from the RPKI repository, the RP is required to check the signed object syntax.

Section 3 of [RFC6488] lists all the steps that the RP is required to execute in order to validate the top level syntax of a repository signed object.

Note that these checks are necessary, but not sufficient. Additional validation checks must be performed based on the specific type of signed object.

4.2. Syntax and Validation for Each Type of Signed Object

4.2.1. Manifest

To determine whether a manifest is valid, the RP is required to perform manifest-specific checks in addition to those specified in [RFC6488].

Specific checks for a Manifest are described in Section 4 of [RFC6486]. If any of these checks fails, indicating that the manifest is invalid, then the manifest will be discarded and treated as though no manifest were present.

4.2.2. ROA

To validate a ROA, the RP is required perform all the checks specified in [RFC6488] as well as the additional ROA-specific validation steps. The IP address delegation extension [RFC3779] present in the end-entity (EE) certificate (contained within the ROA), must encompass each of the IP address prefix(es) in the ROA.

More details for ROA validation are specified in Section 2 of [RFC6482].

4.2.3. Ghostbusters

The Ghostbusters Record is optional; a publication point in the RPKI can have zero or more associated Ghostbuster Records. If a CA has at least one Ghostbuster Record, RP is required to verify that this Ghostbusters Record conforms to the syntax of signed object defined in [RFC6488].

The payload of this signed object is a (severely) profiled vCard. An RP is required to verify that the payload of Ghostbusters conforms to format as profiled in [RFC6493].

4.2.4. Verifying BGPsec Router Certificate

A BGPsec Router Certificate is a resource certificate, so it is required to comply with [RFC6487]. Additionally, the certificate must contain an AS Identifier Delegation extension, and must not contain an IP Address Delegation extension. The validation procedure used for BGPsec Router Certificates is identical to the validation procedure described in Section 7 of [RFC6487], but using the constraints applied come from specification of Section 7 of [RFC8209].

Note that the cryptographic algorithms used by BGPsec routers are found in [RFC8208]. Currently, the algorithms specified in [RFC8208]and [RFC7935] are different. BGPsec RPs will need to support algorithms that are used to validate BGPsec signatures as well as the algorithms that are needed to validate signatures on BGPsec certificates, RPKI CA certificates, and RPKI CRLs.

4.3. How to Make Use of Manifest Data

For a given publication point, the RP ought to perform tests, as specified in Section 6.1 of [RFC6486], to determine the state of the Manifest at the publication point. A Manifest can be classified as either valid or invalid, and a valid Manifest is either current and

stale. An RP decides how to make use of a Manifest based on its state, according to local (RP) policy.

If there are valid objects in a publication point that are not present on a Manifest, [RFC6486] does not mandate specific RP behavior with respect to such objects. However, most RP software ignores such objects and this document recommends that this behavior be adopted uniformly.

In the absence of a Manifest, an RP is expected to accept all valid signed objects present in the publication point. If a Manifest is stale or invalid (see [RFC6486]) and an RP has no way to acquire a more recently valid Manifest, the RP is expected to contact the repository manager via Ghostbusters record and thereafter make decision according to local (RP) policy.

4.4. What to Do with Ghostbusters Information

An RP may encounter a stale Manifest or CRL, or an expired CA certificate or ROA at a publication point. An RP is expected to use the information from the Ghostbusters record to contact the maintainer of the publication point where any stale/expired objects were encountered. The intent here is to encourage the relevant CA and/or repository manager to update the slate or expired objects.

5. Distributing Validated Cache

On a periodic basis, BGP speakers within an AS request updated validated origin AS data and router/ASN data from the validated cache of RPKI data. The RP may either transfer the validated data to the BGP speakers directly, or it may transfer the validated data to a cache server that is responsible for provisioning such data to BGP speakers. The specification of the protocol designed to deliver validated cache data to a BGP Speaker is provided in [RFC6810] and [RFC8210].

6. Security Considerations

The RP links the RPKI provisioning side and the routing system, establishing the local view of global RPKI data to BGP speakers. The security of the RP is critical to BGP messages exchanging. The RP implementation is expected to offer cache backup management to facilitate recovery from outage state. The RP implementation also should support application of secure transport (e.g., IPsec [RFC4301]) that is able to protect validated cache delivery in a unsafe environment.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The authors thank David Mandelberg, Wei Wang and Tim Bruijnzeels for their review, feedback and editorial assistance in preparing this document.

9. References

9.1. Normative References

[I-D.ietf-sidrops-https-tal]
Huston, G., Weiler, S., Michaelson, G., Kent, S., and T. Bruijnzeels, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", draft-ietf-sidrops-https-tal-06 (work in progress), January 2019.

[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, <https://www.rfc-editor.org/info/rfc3779>.

[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, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

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

[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, <https://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, <https://www.rfc-editor.org/info/rfc6487>.

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

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493, February 2012, <https://www.rfc-editor.org/info/rfc6493>.

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

[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, <https://www.rfc-editor.org/info/rfc7935>.

[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>.

[RFC8210] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", RFC 8210, DOI 10.17487/RFC8210, September 2017, <https://www.rfc-editor.org/info/rfc8210>.

[RFC8360] Huston, G., Michaelson, G., Martinez, C., Bruijnzeels, T., Newton, A., and D. Shaw, "Resource Public Key Infrastructure (RPKI) Validation Reconsidered", RFC 8360, DOI 10.17487/RFC8360, April 2018, <https://www.rfc-editor.org/info/rfc8360>.

9.2. Informative References

[I-D.ietf-sidrops-bgpsec-rollover]
Weis, B., Gagliano, R., and K. Patel, "BGPsec Router Certificate Rollover", draft-ietf-sidrops-bgpsec-rollover-04 (work in progress), December 2017.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

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

[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, <https://www.rfc-editor.org/info/rfc6489>.

[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, <https://www.rfc-editor.org/info/rfc6916>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[rsync] "rsync web page", <http://rsync.samba.org/>.

Authors' Addresses

Di Ma
ZDNS
4 South 4th St. Zhongguancun
Haidian, Beijing 100190
China

Email: madi@zdns.cn

Stephen Kent
Independent

Email: kent@alum.mit.edu

SIDR Operations
Internet-Draft
Intended status: Informational
Expires: March 20, 2019

O. Muravskiy
RIPE NCC
T. Bruijnzeels
NLNetLabs

September 16, 2018

RPKI Certificate Tree Validation by the RIPE NCC RPKI Validator
draft-ietf-sidrops-rpki-tree-validation-03

Abstract

This document describes the approach to validate the content of the RPKI certificate tree, as it is implemented in the RIPE NCC RPKI Validator. This approach is independent of a particular object retrieval mechanism. This allows it to be used with repositories available over the rsync protocol, the RPKI Repository Delta Protocol, and repositories that use a mix of both.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on March 20, 2019.

Table of Contents

1. Scope of this document

This document describes how the RIPE NCC RPKI Validator version 2.23 has been implemented. Source code to this software can be found at [github]. The purpose of this document is to provide transparency to users of (and contributors to) this software tool, as well as serve to be subjected to scrutiny by the SIDR Operations Working Group. It is not intended as a document that describes a standard or best practices on how validation should be done in general.

2. Introduction

In order to use information published in RPKI repositories, Relying Parties (RP) need to retrieve and validate the content of certificates, certificate revocation lists (CRLs), and other RPKI signed objects. To validate a particular object, one must ensure that all certificates in the certificate chain up to the Trust Anchor (TA) are valid. Therefore the validation of a certificate tree is performed top-down, starting from the TA certificate and descending down the certificate chain, validating every encountered certificate and its products. The result of this process is a list of all encountered RPKI objects with a validity status attached to each of them. These results may later be used by a Relying Party in taking routing decisions, etc.

Traditionally RPKI data is made available to RPs through the repositories [RFC6481] accessible over [rsync] protocol. Relying parties are advised to keep a local copy of repository data, and perform regular updates of this copy from the repository (Section 5 of [RFC6481]). The RPKI Repository Delta Protocol [RFC8182] introduces another method to fetch repository data and keep the local copy up to date with the repository.

This document describes how the RIPE NCC RPKI Validator discovers RPKI objects to download, builds certificate paths, and validates RPKI objects, independently from what repository access protocol is used. To achieve this, it puts downloaded RPKI objects in an object store, where each RPKI object can be found by its URI, the hash of its content, value of its Authority Key Identifier (AKI) extension, or a combination of these. It also keeps track of the download and the validation time for every object, to decide which locally stored objects are not used in the RPKI tree validation and could be removed.

3. General Considerations

3.1. Hash comparisons

This algorithm relies on the collision resistance properties of the file hash algorithm (defined in [RFC7935]) to compute the hash of repository objects. It assumes that any two objects for which the hash value is the same, are identical.

The hash comparison is used when matching objects in the repository with entries on the manifest (Section 4.2.2), and when looking up objects in the object store (Section 6).

3.2. Discovery of RPKI objects issued by a CA

There are several possible ways of discovering potential products of a CA certificate: one could use all objects located in a repository directory designated as a publication point for a CA, or only objects mentioned on the manifest located at that publication point (see Section 6 of[RFC6486]), or use all known repository objects whose AKI extension matches the Subject Key Identifier (SKI) extension (Section 4.2.1 of[RFC5280]) of a CA certificate.

For publication points whose content is consistent with the manifest and issuing certificate all of these approaches should produce the same result. For inconsistent publication points the results might be different. Section 6 of [RFC6486] leaves the decision on how to deal with inconsistencies to a local policy.

The implementation described here does not rely on content of repository directories, but uses the Authority Key Identifier (AKI) extension of a manifest and a certificate revocation list (CRL) to find in an object store (Section 6) a manifest and a CRL issued by a particular Certification Authority (CA) (see Section 4.2.1). It further uses the hashes of manifest’s fileList entries (Section 4.2.1 of [RFC6486]) to find other objects issued by the CA, as described in Section 4.2.2.

3.3. Manifest entries versus repository content

Since the current set of RPKI standards requires use of the manifest [RFC6486] to describe the content of a publication point, this implementation requires strict consistency between the publication point content and manifest content. (This is a more stringent requirement than established in [RFC6486].) Therefore it will not process objects that are found in the publication point but do not match any of the entries of that publication point’s manifest (see Section 4.2.2). It will also issue warnings for all found

mismatches, so that the responsible operators could be made aware of inconsistencies and fix them.

4. Top-down Validation of a Single Trust Anchor Certificate Tree

1. The validation of a Trust Anchor (TA) certificate tree starts from its TA certificate. To retrieve the TA certificate, a Trust Anchor Locator (TAL) object is used, as described in Section 4.1.

2. If the TA certificate is retrieved, it is validated according to Section 7 of [RFC6487] and Section 2.2 of [RFC7730]. Otherwise the validation of certificate tree is aborted and an error is issued.

3. If the TA certificate is valid, then all its subordinate objects are validated as described in Section 4.2. Otherwise the validation of certificate tree is aborted and an error is issued.

4. For each repository object that was validated during this validation run, its validation timestamp is updated in the object store (see Section 6.1.7).

5. Outdated objects are removed from the store as described in Section 4.3. This completes the validation of the TA certificate tree.

4.1. Fetching the Trust Anchor Certificate Using the Trust Anchor Locator

The following steps are performed in order to fetch a Trust Anchor Certificate:

1. (Optional) If the Trust Anchor Locator contains a "prefetch.uris" field, pass the URIs contained in that field to the fetcher (see Section 5.1.1). (This field is a non-standard addition to the TAL format. It helps fetching non-hierarchical rsync repositories more efficiently.)

2. Extract the first TA certificate URI from the TAL's URI section (see Section 2.1 of [RFC7730]) and pass it to the object fetcher (Section 5.1.2). If the fetcher returns an error, repeat this step for every URI in the URI section, until no error is encountered, or no more URIs left.

3. Retrieve from the object store (see Section 6.1.3) all certificate objects, for which the URI matches the URI extracted from the TAL in the previous step, and the public key matches the

subjectPublicKeyInfo extension of the TAL (see Section 2.1 of [RFC7730]).

4. If no, or more than one such objects are found, issue an error and abort certificate tree validation process with an error. Otherwise, use the single found object as the Trust Anchor certificate.

4.2. CA Certificate Validation

The following steps describe the validation of a single CA Resource certificate:

1. If both the caRepository (Section 4.8.8.1 of [RFC6487]), and the id-ad-rpkiNotify (Section 3.2 of [RFC8182]) SubjectInfoAccess (SIA) pointers are present in the CA certificate, use a local policy to determine which pointer to use. Extract the URI from the selected pointer and pass it to the object fetcher (that will then fetch all objects available from that repository, see Section 5.1.1).

2. For the CA certificate, find the current manifest and certificate revocation list (CRL), using the procedure described in Section 4.2.1. If no such manifest and CRL could be found, stop validation of this certificate, consider it invalid, and issue an error.

3. Compare the URI found in the id-ad-rpkiManifest field (Section 4.8.8.1 of [RFC6487]) of the SIA extension of the certificate with the URI of the manifest found in the previous step. If they are different, issue a warning, but continue validation process using the manifest found in the previous step. (This warning indicates that there is a mismatch between the expected and the actual location of an object in a repository. See Section 9 for the explanation of this mismatch and the decision taken.)

4. Perform manifest entries discovery and validation as described in Section 4.2.2.

5. Validate all resource certificate objects found on the manifest, using the CRL object found on the manifest:

   * if the strict validation option is enabled by the operator, the validation is performed according to Section 7 of [RFC6487],

* otherwise, the validation is performed according to Section 7 of [RFC6487], with the exception of the resource certification path validation, that is performed according to Section 4.2.4.4 of [RFC8360].

(Note that this implementation uses the operator configuration to decide which algorithm to use for path validation. It applies the selected algorithm to all resource certificates, rather than applying appropriate algorithm per resource certificate, based on the object identifier (OID) for the Certificate Policy found in that certificate, as specified in [RFC8360].)

6. Validate all Route Origin Authorization (ROA) objects found on the manifest, using the CRL object found on the manifest, according to Section 4 of [RFC6482].

7. Validate all Ghostbusters Record objects found on the manifest, using the CRL object found on the manifest, according to Section 7 of [RFC6493].

8. For every valid CA certificate object found on the manifest, apply the procedure described in this section (Section 4.2), recursively, provided that this CA certificate (identified by its SKI) has not yet been validated during current tree validation run.

4.2.1. Finding the most recent valid manifest and CRL

1. Fetch from the store (see Section 6.1.4) all objects of type manifest, whose certificate's AKI extension matches the SKI of the current CA certificate. If no such objects are found, stop processing the current CA certificate and issue an error.

2. Find among found objects the manifest object with the highest manifestNumber field (Section 4.2.1 of [RFC6486]), for which all following conditions are met:

   * There is only one entry in the manifest for which the store contains exactly one object of type CRL, the hash of which matches the hash of the entry.

   * The manifest's certificate AKI equals the above CRL's AKI.

   * The above CRL is a valid object according to Section 6.3 of [RFC5280].

   * The manifest is a valid object according to Section 4.4 of [RFC6486], and its EE certificates is not in the CRL found above.

3. If there is an object that matches above criteria, consider this object to be the valid manifest, and the CRL found at the previous step - the valid CRL for the current CA certificate's publication point.

4. Report an error for every other manifest with a number higher than the number of the valid manifest.

4.2.2. Manifest entries validation

For every entry in the manifest object:

1. Construct an entry's URI by appending the entry name to the current CA's publication point URI.

2. Get all objects from the store whose hash attribute equals entry's hash (see Section 6.1.2).

3. If no such objects are found, issue an error for this manifest entry and progress to the next entry. This case indicates that the repository does not have an object at the location listed in the manifest, or that the object's hash does not match the hash listed in the manifest.

4. For every found object, compare its URI with the URI of the manifest entry.

   * For every object with a non-matching URI issue a warning. This case indicates that the object from the manifest entry is (also) found at a different location in a (possibly different) repository.

   * If no objects with a matching URI are found, issue a warning. This case indicates that there is no object found in the repository at the location listed in the manifest entry (but there is at least one matching object found at a different location).

5. Use all found objects for further validation as per Section 4.2.

Please note that the above steps will not reject objects whose hash matches the hash listed in the manifest, but the URI does not. See Section 9.3 for additional information.

4.3. Object Store Cleanup

At the end of every TA tree validation some objects are removed from the store using the following rules:

1. Given all objects that were encountered during the current validation run, remove from the store (Section 6.1.6) all objects whose URI attribute matches the URI of one of the encountered objects, but the content's hash is different. This removes from the store objects that were replaced in the repository by their newer versions with the same URIs.

2. Remove from the store all objects that were last encountered during validation a long time ago (as specified by the local policy). This removes objects that do not appear on any valid manifest anymore (but possibly are still published in a repository).

3. Remove from the store all objects that were downloaded recently (as specified by the local policy), but have never been used in the validation process. This removes objects that have never appeared on any valid manifest.

Shortening the time interval used in step 2 will free more disk space used by the store, at the expense of downloading removed objects again if they are still published in the repository.

Extending the time interval used in step 3 will prevent repeated downloads of repository objects, with the risk that such objects, if created massively by mistake or by an adversary, will fill up local disk space, if they are not cleaned up promptly.

5. Remote Objects Fetcher

The fetcher is responsible for downloading objects from remote repositories (described in Section 3 of [RFC6481]) using rsync protocol ([rsync]), or RPKI Repository Delta Protocol (RRDP) ([RFC8182]).

5.1. Fetcher Operations

For every visited URI the fetcher keeps track of the last time a successful fetch occurred.

5.1.1. Fetch repository objects

This operation receives one parameter - a URI. For an rsync repository this URI points to a directory. For an RRDP repository it points to the repository's notification file.

The fetcher performs following steps:

1. If data associated with the URI has been downloaded recently (as specified by the local policy), skip following steps.

2. Download remote objects using the URI provided (for an rsync repository use recursive mode). If the URI contains schema "https" and download has failed, issue a warning, replace "https" schema in the URI by "http", and try to download objects again, using the resulting URI.

3. If remote objects can not be downloaded, issue an error and skip following steps.

4. Perform syntactic verification of fetched objects. The type of every object (certificate, manifest, CRL, ROA, or Ghostbusters record), is determined based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively). The syntax of the object is described in Section 4 of [RFC6487] for resource certificates, step 1 of Section 3 of [RFC6488] for signed objects, and specifically, Section 4 of [RFC6486] for manifests, [RFC5280] for CRLs, Section 3 of [RFC6482] for ROAs, and Section 5 of [RFC6493] for Ghostbusters records.

5. Put every downloaded and syntactically correct object in the object store (Section 6.1.1).

The time interval used in the step 1 should be chosen based on the acceptable delay in receiving repository updates.

5.1.2. Fetch single repository object

This operation receives one parameter - a URI that points to an object in a repository.

The fetcher performs following operations:

1. Download remote object using the URI provided. If the URI contains "https" schema and download failed, issue a warning, replace "https" schema in the URI by "http", and try to download the object using the resulting URI.

2. If the remote object can not be downloaded, issue an error and skip following steps.

3. Perform syntactic verification of fetched object. The type of object (certificate, manifest, CRL, ROA, or Ghostbusters record), is determined based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively). The syntax of the object is described in Section 4 of [RFC6487] for resource certificates, step 1 of Section 3 of [RFC6488] for signed objects, and specifically, Section 4 of [RFC6486] for manifests, [RFC5280] for CRLs, Section 3 of [RFC6482] for ROAs, and Section 5 of [RFC6493] for Ghostbusters records.

4. If the downloaded object is not syntactically correct, issue an error and skip further steps.

5. Delete all objects from the object store (Section 6.1.5) whose URI matches the URI given.

6. Put the downloaded object in the object store (Section 6.1.1).

6. Local Object Store

6.1. Store Operations

6.1.1. Store Repository Object

Put given object in the store, along with its type, URI, hash, and AKI, if there is no record with the same hash and URI fields. Note that in the (unlikely) event of hash collision the given object will not replace the object in the store.

6.1.2. Get objects by hash

Retrieve all objects from the store whose hash attribute matches the given hash.

6.1.3. Get certificate objects by URI

Retrieve from the store all objects of type certificate, whose URI attribute matches the given URI.

6.1.4. Get manifest objects by AKI

Retrieve from the store all objects of type manifest, whose AKI attribute matches the given AKI.

6.1.5. Delete objects for a URI

For a given URI, delete all objects in the store with matching URI attribute.

6.1.6. Delete outdated objects

For a given URI and a list of hashes, delete all objects in the store with matching URI, whose hash attribute is not in the given list of hashes.

6.1.7. Update object's validation time

For all objects in the store whose hash attribute matches the given hash, set the last validation time attribute to the given timestamp.

7. Acknowledgements

This document describes the algorithm as it is implemented by the software development team at the RIPE NCC, which included over time: Mikhail Puzanov, Erik Rozendaal, Miklos Juhasz, Misja Alma, Thiago da Cruz Pereira, Yannis Gonianakis, Andrew Snare, Varesh Tapadia, Paolo Milani, Thies Edeling, Hans Westerbeek, Rudi Angela, and Constantijn Visinescu. The authors would also like to acknowledge contributions by Carlos Martinez, Andy Newton, Rob Austein, and Stephen Kent.

8. IANA Considerations

This document has no actions for IANA.

9. Security Considerations

9.1. Hash collisions

This implementation will not detect possible hash collisions in the hashes of repository objects (calculated using the file hash algorithm specified in [RFC7935]). It considers objects with same hash values as identical.

9.2. Algorithm agility

This implementation only supports hash algorithms and key sizes specified in [RFC7935]). Algorithm agility described in [RFC6916] is not supported.

9.3. Mismatch between the expected and the actual location of an object in the repository

According to Section 2 of [RFC6481], all objects issued by a particular CA certificate are expected to be located in one repository publication point, specified in the SIA extension of that CA certificate. The manifest object issued by that CA certificate enumerates all other issued objects, listing their file names and content hashes.

However, it is possible that an object whose content hash matches the hash listed in the manifest, has either a different file name, or is located at a different publication point in a repository.

On the other hand, all RPKI objects, either explicitly or within their embedded EE certificate, have an Authority Key Identifier extension that contains the key identifier of their issuing CA certificate. Therefore it is always possible to perform an RPKI validation of the object whose expected location does not match its actual location, provided that the certificate that matches the AKI of the object in question is known to the system that performs validation.

In case of a mismatch described above this implementation will not exclude an object from further validation merely because its actual location or file name does not match the expected location or file name. This decision was chosen because the actual location of a file in a repository is taken from the repository retrieval mechanism, which, in case of an rsync repository, does not provide any cryptographic security, and in case of an RRDP repository, provides only a transport layer security, with the fallback to unsecured transport. On the other hand, the manifest is an RPKI signed object, and its content could be verified in the context of the RPKI validation.

9.4. Manifest content versus publication point content

This algorithm uses the content of a manifest object to determine other objects issued by a CA certificate. It verifies that the manifest is located in the publication point designated in the CA Certificate’s SIA extension. However, if there are other (not listed in the manifest) objects located in the same publication point directory, they are ignored, even if they might be valid and issued by the same CA as the manifest. (This RP behavior is allowed, but not required, by [RFC6486].)

9.5. Possible denial of service

The store cleanup procedure described in Section 4.3 tries to minimise removal and subsequent re-fetch of objects that are published in a repository, but not used in the validation. Once such objects are removed from the remote repository, they will be discarded from the local object store after a period of time specified by a local policy. By generating an excessive amount of syntactically valid RPKI objects, a man-in-the-middle attack between a validating tool and a repository could force an implementation to fetch and store those objects in the object store (see Section 5.1.1) before they are validated and discarded, leading to an out-of-memory or out-of-disk-space conditions, and, subsequently, a denial of service.

10. References

10.1. Normative References

[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, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

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

[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, <https://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, <https://www.rfc-editor.org/info/rfc6487>.

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

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493, February 2012, <https://www.rfc-editor.org/info/rfc6493>.

[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, <https://www.rfc-editor.org/info/rfc6916>.

[RFC7730] Huston, G., Weiler, S., Michaelson, G., and S. Kent, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 7730, DOI 10.17487/RFC7730, January 2016, <https://www.rfc-editor.org/info/rfc7730>.

[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, <https://www.rfc-editor.org/info/rfc7935>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[RFC8360] Huston, G., Michaelson, G., Martinez, C., Bruijnzeels, T., Newton, A., and D. Shaw, "Resource Public Key Infrastructure (RPKI) Validation Reconsidered", RFC 8360, DOI 10.17487/RFC8360, April 2018, <https://www.rfc-editor.org/info/rfc8360>.

10.2. Informative References

[github] "RIPE NCC RPKI Validator on GitHub", <https://github.com/RIPE-NCC/rpki-validator>.

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

Authors' Addresses

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net
URI: https://www.ripe.net/

Tim Bruijnzeels
NLNetLabs

Email: tim@nlnetlabs.nl
URI: https://www.nlnetlabs.nl/

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2019

T. Bruijnzeels
NLnet Labs
C. Martinez
LACNIC
R. Austein
Dragon Research Labs

October 16, 2018

RPKI Signed Object for Trust Anchor Keys
draft-ietf-sidrops-signed-tal-02

Abstract

Trust Anchor Locators (TALs) [I-D.ietf-sidrops-https-tal] are used by Relying Parties in the RPKI to locate and validate Trust Anchor certificates used in RPKI validation. This document defines an RPKI signed object for Trust Anchor Keys (TAK), that can be used by Trust Anchors to signal their set of current keys and the location(s) of the accompanying CA certiifcates to Relying Parties, as well as changes to this set in the form of revoked keys and new keys, in order to support both planned and unplanned key rolls without impacting RPKI validation.

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 https://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 19, 2019.

Table of Contents

1. Requirements notation

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

2. Overview

Trust Anchor Locators (TALs) [I-D.ietf-sidrops-https-tal] are used by Relying Parties in the RPKI to locate and validate Trust Anchor (TA) certificates used in RPKI validation. However, until now there has been no formal way of notifying Relying Parties (RP) of updates to a TAL. Such updates may be needed in particular in case a Trust Anchor needs to perform a planned, or unplanned, key roll.

This document defines a new RPKI signed object that can be used to document the current set of keys and the loctaion(s) of the accompanying CA certificates, as well as any changes to this set. This allows RPs to be notified automatically of such changes, and enables Trust Anchors to pre-stage a number of operational keys so that planned and unplanned key rolls can be performed without risking the invalidation of the RPKI tree under the TA. We call this object the Trust Anchor Keys (TAK) object.

When Relying Parties (RPs) are first bootstrapped, they use any current TAL to discover a key and location(s) of the TA certificate(s) for a TA. The RP can then retreive and validating the TA certificate, and subsequently validate the manifest [RFC6486] and CRL [section 5 of @!RFC6487]. However, before processing any other objects it will then first validate the TAK object, if present. All enumarated new keys (and locations) are then added to a new list of current TA keys for this TA. The RP will then recursively fetch and validate the TA certificates, manifest, CRL and TAK objects for each of these keys. As a part of this process the RP will also compile a list of revoked keys enumarated by any of the validly signed TAK objects. As the final step the RP will then filter out any revoked TA keys from its new set. This new set now replaces the previous set.

If the key used to start this process is still considered current, then validation continues. But if the key was revoked, then validation is restarted using one of the remaining keys in the set.

This process allows Trust Anchors to operate a set of N current keys, where any key can effectively revoke any or all of the other keys to perform either a planned, or an unplanned, key roll. This also

allows Trust Anchors to produce long lived TAK objects as forward pointers to RPs, and retire its old key when doing a key roll.

While the generic process is quite involved, the amount of work needed to support an envisioned normal key roll is fairly limited. Under normal circumstances a TA will typically have two current keys, so that is can perform an emergency roll over in case one of the keys is lost. This means that the RP will need to validate two TAK objects. However, typically these files will agree that both keys are current and validation continues.

When a key roll is executed a TA will remove one old key, and introduce one new (back-up) key. The RP will remove the old key from its set, and it will not be queried again, and it will add the new key and its TA certifcate location(s).

Only in a situation where an RP is very outdated can it be expected that the RP will have to discover several chained TAK object. But, since it will remove the outdated TALs in this process, this presents a one time cost only.

Note that in theory a TA can revoke all of its keys and make itself obsolete. In practice however, a well operated TA will have measures in place to prevent this. Furthermore they can protect themselves against key loss to adversaries through the use of such as the use of a Hardware Security Module (HSM) to protect keys. Protecting against this mis-operation would incur complexity and guesswork on the RPs. Therefore it is believed that it is best to keep the process straightforward, and offer a solution for the more likely issues of loss of a key, e.g. because an HSM or card set is broken, and planned key rolls.

3. TAK Object definition

The TAK object makes use of the template for RPKI digitally signed objects [RFC6488], which defines a Crytopgraphic Message Syntax (CMS) [RFC5652] wrapper for the Signed TALs content as well as a generic validation procedure for RPKI signed objects. Therefore, to complete the specification of the TAK object (see Section 4 of [RFC6488]), this document defines:

- The OID defined in Section 3.1 that identifies the signed object as being a TAK. (This OID appears within the eContentType in the encapContentInfo object as well as the content-type signed attribute in the signerInfo object).

- The ASN.1 syntax for the TAK eContent defined in Section 3.2.

- Additional steps to the validation steps specified in [RFC6488] required to validate the TAK, defined in Section 3.3.

3.1. The TAK Object Content Type

This document requests an OID for TAK objects as follows:

```
signed-Tal OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
           rsadsi(113549) pkcs(1) pkcs9(9) 16 id-smime (1) TBD }
```

This OID MUST appear both within the eContentType in the encapContentInfo object as well as the content-type signed attribute in the signerInfo object (see [RFC6488])

3.2. The TAK Object eContent

The content of a TAK object is ASN.1 encoded using the Distinguished Encoding Rules (DER) [X.690], and is defined as follows:

```
TAK ::= SEQUENCE {
   version   INTEGER DEFAULT 0,
   current   ::= SEQUENCE SIZE (1..MAX) OF CurrentKey,
   revoked   ::= SEQUENCE OF SubjectPublicKeyInfo
}

CurrentKey ::= SEQUENCE {
   certificateURIs       SEQUENCE SIZE (1..MAX) OF CertificateURI,
   subjectPublicKeyInfo  SubjectPublicKeyInfo
}

CertificateURI ::= IA5String

SubjectPublicKeyInfo ::= SEQUENCE {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING
}
```

3.2.1. version

The version number of the TAK object MUST be 0.

3.2.2. current

This field defines the set of current keys (CurrentKey) according to the signer of this Signed TALs object.

3.2.2.1. CurrentKey

This field defines a current TA Key, equivalent to [I-D.ietf-sidrops-https-tal]. This structure contains a sequence of one or more URIs and a SubjectPublicKeyInfo.

3.2.2.1.1. certificateURIs

This field is equivalent to the URI section in section 2.1 of [I-D.ietf-sidrops-https-tal]. It MUST contain at least one CertificateURI element. Each CertificateURI element contains the IA5String representation of either an rsync URI [RFC5781], or an HTTPS URI [RFC7230].

3.2.2.1.2. subjectPublicKeyInfo

This field contains a SubjectPublicKeyInfo [section 4.1.2.7 or @!RFC5280] in DER format [X.690].

3.2.3. revoked

This field contains the list of keys, identified by SubjectPublicKeyInfo, that are no longer to be used according to the signer of this document.

3.3. TAK Object Validation

To determine whether a TAK object is valid, the RP MUST perform the following steps in addition to those specified in [RFC6488]:

- The eContentType OID matches the OID described in Section 3.1
- The TAK object appears as the product of a Trust Anchor CA certificate.
- This Trust Anchor CA has published only one TAK object in its repository for this key, and this object appears on the Manifest as the only entry using the ".tak" extension (see [RFC6481]). In case more than one TAK object is found, all such objects MUST be considered invalid.
- The EE certificate of this TAK object describes its Internet Number Resources (INRs) using the "inherit" attribute
- The decoded TAK content conforms to the format defined in Section 3.2.

If the above procedure indicates that the manifest is invalid, then the TAK object MUST be discarded and treated as though no TAK object were present.

4. Maintaining multiple TA keys

As described in Section 6 a TA will most likely choose to operate two keys at any one time in order to be prepared for an emergency key roll. When a TA operates multiple keys, each key MUST use its own CA repository publication point as described in [RFC6481]. The CRL and Manifest [RFC6486] for each of these keys will be unique to each key, but the TA MUST ensure that equivalent CA certificates and RPKI signed objects are issued under each key. Note that this is similar to how such certificates and RPKI signed objects are re-issued as part of a lower level CA key roll, described in section 4 of [RFC6489].

4.1. Prepare a new TA key

The Trust Anchors MUST generate a new key pair and generate a new TA Certificate. For the Subject Information Access (see section 4.8.8.1 of [RFC6487]) this MUST use URIs that will be used by the new key to publish objects. These URIs MUST be uniqe for use by this new key only. The Internet Number Resources on this new certificate MUST be equivalent to those found on the current certificate.

The new TA certificate MUST be published under one or more new Certificate URIs for use by this new key only.

As decribed above, the TA MUST issue and publish equivalent CA certificates and RPKI signed objects under this new key.

It is RECOMMENDED that the TA now generates a new TAL [I-D.ietf-sidrops-https-tal] and verifies that the new Trust Anchor certificate can be retrieved from all locations, and that it generates the same results when it is used for top-down validation instead of (any of) the current TA key(s).

Note that the TA MAY choose to make this TAL available to Relying Parties, in particular to those that do not support TAK objects, and for inclusion in the distribution of RP software in order to minimise the overhead in bootstrapping fresh installations.

4.2. Publishing for Multiple TA Keys

If a TA uses a single remote publication server for its keys using the RPKI publication protocol [RFC8181], then it MUST include all <publish/> and <withdraw/> PDUs for the products of each of its keys

in a single query in order to ensure that they will reflect the same content at all times.

If a TA uses multiple publication servers then it is by definition inevitable that the content of different keys will be out of sync at times. In such cases the TA SHOULD ensure that the duration of these moments are limited to the shortest possible time. Furthermore the following should be observed:

- It is strongly RECOMMENDED that TAs do not issue any RPKI Signed Objects, such as ROAs [RFC6482], but limit their operations to maintaining a CRL, Manifest and CA certificates only. If an organisation maintaining a TA has an operational need for such objects then it is strongly RECOMMENDED that they operate a separate non-TA CA as a child of their TA for these operations. If this approach is used the remaining issues regarding temporary inconsistencies between multiple TA key repository publication points is greatly reduced.

- In cases where a CA certificate is revoked completely, or replaced by a certifcate with a reduced set of resources, these changes will not take effect fully until all the TA keys repository publication points have been updated. Given that TA key operations are normally performed infrequently we don't expect that this is a problem. I.e. if the revocation or shrinking of an issued CA certificate is staged for days, or weeks anyway, then experiencing a delay of several minutes for the repository publication points to all be updated is fairly insignificant.

- In cases where a CA certificate is replaced by a certifcate with an extend set of resources the TA MUST inform the receiving CA only after all its repository publication points have been updated. This ensures that the receiveing CA will not issue any products that could be invalid if an RP uses a TA key just before the CA certificate was due to be updated.

5. TAK Object Generation and Publication

A TA MAY choose to use TAK objects to communicate its set of current, and revoked keys. If a TA chooses to use TAK objects, then it SHOULD generate and publish TAK objects under each of its current keys. An exception to this rule exists when a TA has lost permanent access to one of its keys or the accompanying repository publication point. In such cases however, the key in question MUST be revoked as described below in Section 6.

A non-normative guideline for naming this object is that the filename chosen for the Signed TAL Object in the publication repository be a

value derived from the public key part of the entity's key pair, using the algorithm described for CRLs in section 2.2 of [RFC6481] for generation of filenames. The filename extension of ".tak" MUST be used to denote the object as a TAK. Note that this is in-line with filename extensions defined in section 7.2 of [RFC6481]

In order to generate the TAK Objects, the TA MUST perform the following actions:

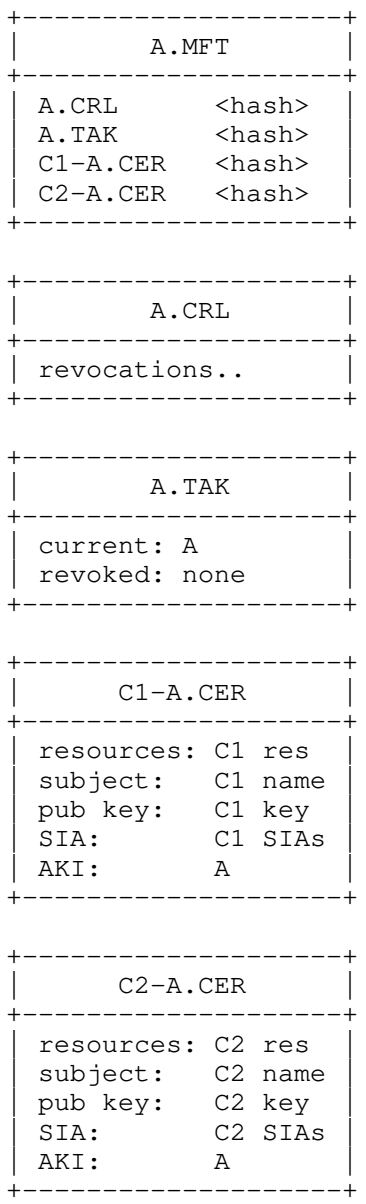
- The TA MUST generate a key pair for a "one-time-use" EE certificate to use for the TAK
- The TA MUST generate a one-time-use EE certificate for the TAK
- This EE certificate MUST have an SIA extension access description field with an accessMethod OID value of id-ad-signedobject, where the associated accessLocation references the publication point of the TAK as an object URL.
- As described in [RFC6487], an [RFC3779] extension is required in the EE certificate used for this object. However, because the resource set is irrelevant to this object type, this certificate MUST describe its Internet Number Resources (INRs) using the "inherit" attribute, rather than explicit description of a resource set.
- This EE certificate MUST have a "notBefore" time that matches, or predates the moment that the TAK will be published.
- This EE certificate MUST have a "notAfter" time that reflects the intended duration for which this TAK will be published. If the EE certificate for a Signed TAL is expired, it MUST no longer be published, but it MAY be replaced by a newly generated TAK object with equivalent content and an updated "notAfter" time.
- The same set of current keys (see Section 3.2.2) MUST be included on each TAK object for each current key.
- The TAK object MUST include all revoked keys (see Section 3.2.3) that became revoked while the key signing the TAK in question was current.

6. Performing TA Key Rolls

6.1. Opting in to Key Rolls

6.1.1. Trust Anchor

For simplicitly let's start with a situation where a TA has only one key. The TA wants to start using TAK objects to perform key rolls in future, so it introduces a TAK object under its single key 'A'. The repository structure looks as follows (irrelevant details omitted):



So, the TA publishes a CRL and MFT under its key A, listing a TAK object and in this case two certificates issued to children 'C1' and 'C2' signed using key A. The TAK object lists key 'A' as the only current key, and has no revoked keys.

6.1.2. Relying Parties

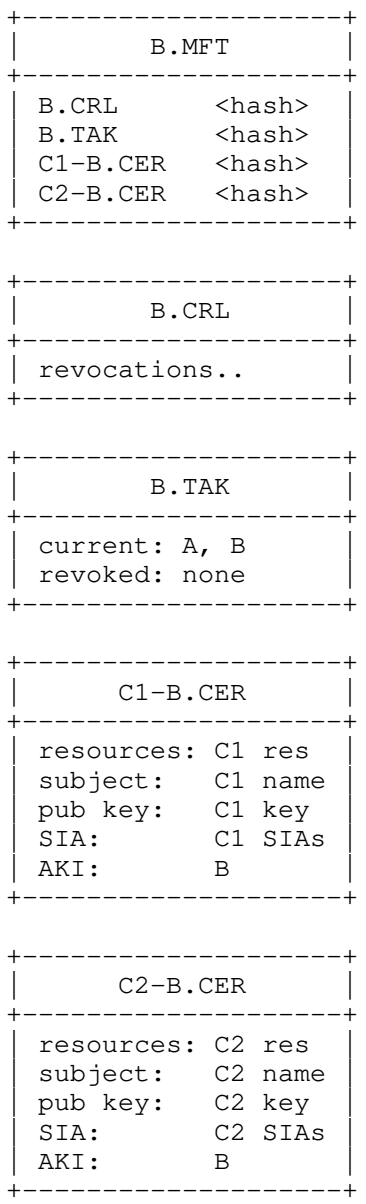
Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. If the RP does support TAK objects it will conclude that key 'A' is the one and only current key, and will proceed to validate the remaining RPKI tree as usual.

6.2. Pre-stage a New Key

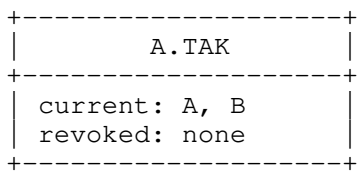
6.2.1. Trust Anchor

Now the TA prestages a new key 'B' and produces equivalent CA certificates for children 'C1' and 'C2', i.e. the resources, subject name, public key and SIA etc are all equivalent, but these certificates are signed under key 'B'. (See Section 4 for a more thorough description of this). The TAK object for key 'B' recognises both keys 'A' and 'B' as current.

The repostory structure and TAK object for key B are then as follows:



When the TA is certain that the content for key 'B' is correct, it can also update the TAK object for key 'A' to include 'B':



One way to do this is by generating a TAL [I-D.ietf-sidrops-https-tal] for key B and verifying that validation using this yields the same results as validation using the TAL for key A would. However, note, that it is preferred that this is done as part of an automated process that is sufficiently well tested, and that the contents of the repositories for keys 'A' and 'B' are updated as a single delta if the publication protocol [RFC8181] is used (see also: Section 5).

6.2.2. Relying Parties

Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. If the RP does support TAK objects it will conclude that there are now two keys 'A' and 'B', and no revoked keys that it should be aware of. Since key 'A' is still current, the RP will continue to validate the RPKI tree structure using the repository for key 'A', ignoring the non-TAK objects in the repository for key 'B'.

The result will be the same for Relying Parties who have a TAL for key 'B' configured, because both keys are equivalent at this time.

6.3. Planned Key Revocation

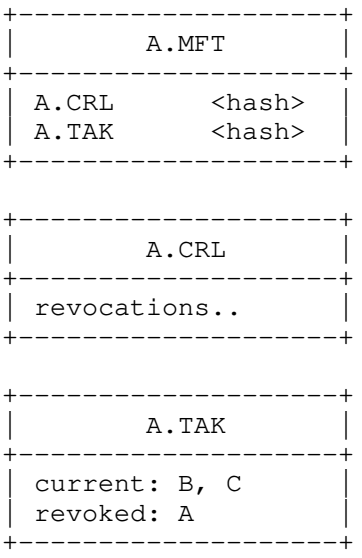
6.3.1. Trust Anchor

The TA has now decided that key 'A' must be revoked. It still has access to this key and the repository, so it can perform a planned key roll. In addition to revoking key 'A', the TA will also generate new key 'C' to ensure that it has at least two current keys at all times for redundancy.

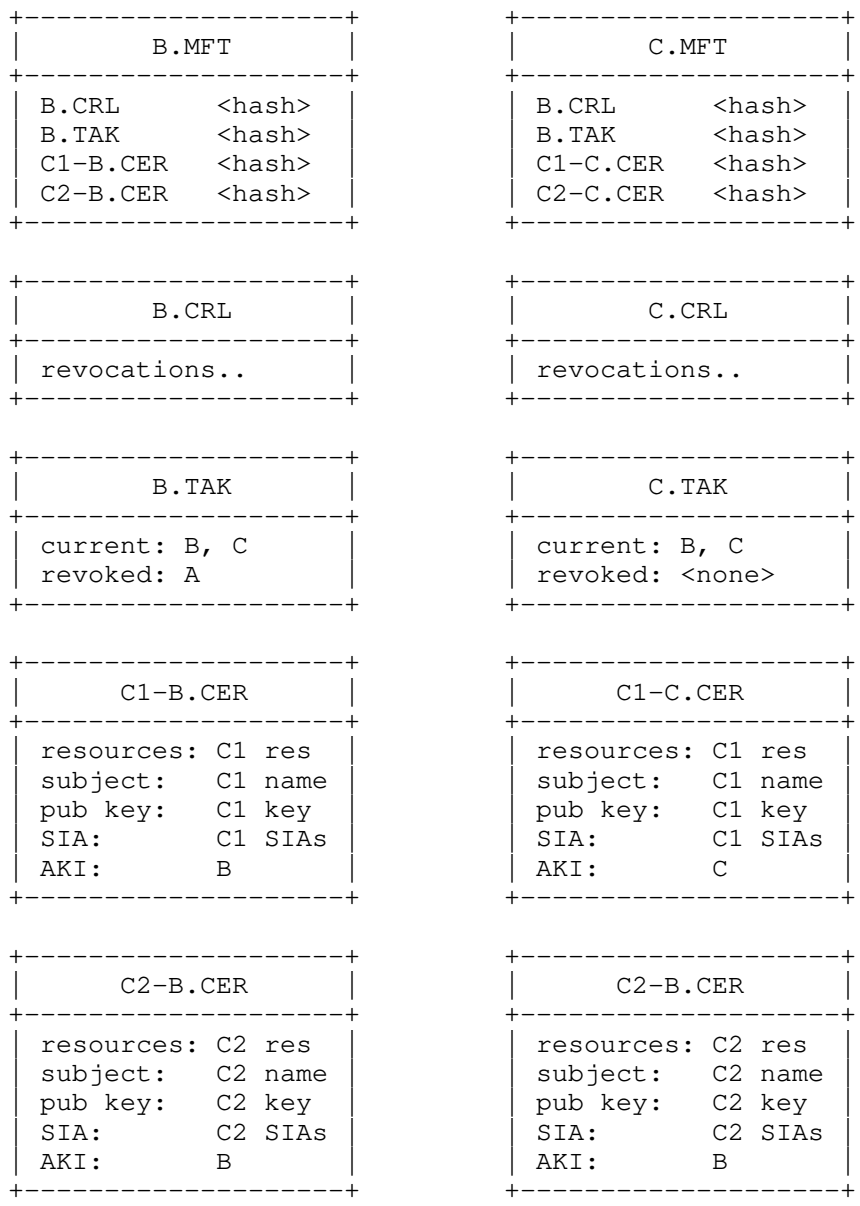
Keys 'B' and 'C' will become current keys on the TAK objects for all keys: 'A', 'B' and 'C'. Key 'A' will become part of the revoked keys on the TAK objects for keys 'A' and 'B'. Note that it is not needed to list key 'A' as revoked on the TAK file for key 'C', because RPs will only learn about key 'C' at the same time as learning about the revocation of key 'A' (see also below).

The TA will publish a long-lived TAK file and MFT and CRL only for key 'A' and publish these objects as waypointers for RPs that have a TAL pointing at key 'A' before destroying key 'A'.

The resulting structure for key 'A' will be as follows:



The resulting structures for keys 'B' and 'C' will be as follows:



In addition to this the TA SHOULD reach out to RP vendors so that they can update the TAL included in the RP software distribution to use key 'B'.

6.3.2. Relying Parties

Relying Parties who have a TAL for key 'A' configured will discover the TAK object. If the RP does not support this object, it will reject this object but continue to validate the remaining RPKI tree as usual. In this case that means that validation will stop, because there are no more objects under key 'A'. Therefore it is important that RPs that do not support TAK files are updated to use the TAL for key 'B' through some other process.

If the RP uses a TAL for key 'A' and it supports TAK objects, it will discover that the TAL for key 'A' has keys 'B' and 'C' as current, and revokes itself. It will then proceed to process keys 'B' and 'C' and find TALs which list the same current keys. So, it will now replace its notion of the current key set for this TA based on its TAL (key 'A') with what it learned. To keep things simple the RP will now conclude that it should re-start validation using a remaining current key, in this case key either 'B' or 'C' may be used.

If the RP already had a TAL for key 'B' and it supports TAK objects, or it simply started with key 'B' because it added it to its set of current keys when this key was pre-staged (see Section 6.2), it will learn that key 'A' is revoked and therefore will not attempt to verify the TAK file for key 'A'. It will also learn about key 'C' and inspect this key's TAL, and discover that only keys 'B' and 'C' are considered current. Since it started the validation process with a key that is still current, it can proceed to validate the RPKI tree using the repository under key 'B'.

6.4. Unplanned revocation

6.4.1. Trust Anchor

Now keys 'B' and 'C' are current. The TA may have intended to revoke key 'B', essentially rolling over to key 'C' and a new key 'D', but let us suppose that the TA lost access to key 'C'. In this case the TA will simply revoke key 'C' instead, and still introduce a new key 'D'.

The major difference with the process described above for planned rolls, is that now the TA will not be able to update the TAK object, MFT or CRL for key 'C'. However, because all TAL objects for current keys are evaluated before tree validation is performed, it is safe to leave these objects in a repository. Keys 'B' and 'D' will simply mark key 'C' as being revoked.

If an RP still has a TAL pointing at key 'C' it will discover that key 'D' is added, and that key 'B' has been revoked through the TAK object published for keys 'B' and 'D'. At least, as long as the the MFT and TAK EE certificates have not expired, and the CRL and MFT are not stale.

If the TA is absolutely sure that the TAL for key 'C' never shipped with any RP distribution, then it would also be safe to delete the repository key 'C' altogether. RPs will learn that 'C' is revoked, and therefore will not even attempt to download the TAK object. However, it is hard to be certain of this and there this is NOT RECOMMENDED.

7. Deployment Considerations

Including Signed TAL objects while RP tools do not support this standard will result in these RPs rejecting these objects. It is not expected that this will result in the invalidation of any other object under a Trust Anchor.

That said, the flagging mechanism introduced here can only be relied on once a majority of RPs support it. Defining when that moment arrives is by definition something that cannot be established at the time of writing this document. Until such time, TAs SHOULD continue to generate unsigned TAL files [I-D.ietf-sidrops-https-tal], and indicate which should be considered their current TAL, and communicate them to RPs through other means.

However, once a majority of RPs support this mechanism it would be RECOMMENDED that Trust Anchor operators perform key rolls regularly. The most assured way to know that such key rolls will work is by making them a part of normal operations. Determining when this moment arrives is by definition out of scope for this document, as it should be based on operational experience.

8. IANA Considerations

8.1. OID

IANA is to add the following to the "RPKI Signed Objects" registry:

Decimal	Description	References
TBD	Trust Anchor Keys	[section 3.1]

8.2. File Extension

IANA is to add an item for the Signed TAL file extension to the "RPKI Repository Name Scheme" created by [RFC6481] as follows:

Extension	RPKI Object	References
.tak	Trust Anchor Keys	[this document]

9. Security Considerations

TBD

10. Acknowledgements

The authors wish to thank Martin Hoffmann for a thourough review of this document.

11. References

11.1. Normative References

[I-D.ietf-sidrops-https-tal]
Huston, G., Weiler, S., Michaelson, G., Kent, S., and T. Bruijnzeels, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", draft-ietf-sidrops-https-tal-05 (work in progress), October 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>.

[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, <https://www.rfc-editor.org/info/rfc3779>.

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

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

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

[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, <https://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, <https://www.rfc-editor.org/info/rfc6487>.

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

[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, <https://www.rfc-editor.org/info/rfc6489>.

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

[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>.

[RFC8181] Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", RFC 8181, DOI 10.17487/RFC8181, July 2017, <https://www.rfc-editor.org/info/rfc8181>.

[X.690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", 2002.

11.2. Informative References

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

Authors' Addresses

Tim Bruijnzeels
NLnet Labs

Email: tim@nlnetlabs.nl
URI: https://www.nlnetlabs.nl/

Carlos Martinez
LACNIC

Email: carlos@lacnic.net
URI: https://www.lacnic.net/

Rob Austein
Dragon Research Labs

Email: sra@hactrn.net