Workgroup: Network Working Group
Internet-Draft:
draft-ietf-lsvr-l3dl-signing-05
Published: 25 July 2023
Intended Status: Standards Track
Expires: 26 January 2024
Authors: R. Bush (Arrcus & IIJ)
R. Housley (Vigil Security)
R. Austein (Arrcus)

Layer-3 Discovery and Liveness Signing

Abstract

The Layer-3 Discovery and Liveness protocol OPEN PDU may contain a public key and a certificate, which can be used to verify signatures on subsequent PDUs. This document describes two mechanisms based on digital signatures, one that is Trust On First Use (TOFU), and one that uses a trust anchor signture over the public key to provide authentication as well as session integrity.

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 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 26 January 2024.

Copyright Notice

Table of Contents

1. Introduction

The Layer-3 Discovery and Liveness protocol [I-D.ietf-lsvr-l3dl] OPEN PDU contains an algorithm identifier, a key, and a L3DL certificate, which can be used to verify signatures on subsequent PDUs. This document describes two methods of key generation and signing for use by L3DL, Trust On First Use (TOFU) and a PKI-based mechanism to provide authentication as well as session integrity.

The Key in the OPEN PDU SHOULD be the public key of an asymmetric key pair. The sender signs with the private key, of course. The device sending the OPEN PDU may use one key for all links, a different key for each link, or some mix(es) thereof.

In the TOFU method the key sent in the OPEN PDU is generated on the sending device, is believed without question by the receiver, and used to verify all subsequent PDUs from the same sender with the same public key and algorithm.

With the PKI method, an enrollment step is performed. The public key is signed by the the operational environment's trust anchor. In this way, the relying party can be confident that the public key is under control of the identified L3DL protocol entity.

As part of enrollment or before hand, all relying parties must have received the trust anchor in an authentic manner.

To the receiver verifying signatures on PDUs, the two methods are indistinguishable; the key provided in the OPEN PDU is used to verify the signatures of subsequent PDUs. The difference that PKI-based keys may be verified against the trust anchor when the OPEN PDU is received.

In the PKI method the public key in the OPEN PDU MUST be verified against the trust anchor for the operational domain. The OPEN PDU public key is then used to verify all subsequent PDUs in the session. A mechanism for 'rolling' from the current public key to a fresh one is described in Section 6.

2. Signature Algorithm Identifiers

To avoid the creation of yet another IANA registry for digital signature algorithm identifiers, this specification makes use of the existing IANA registry for "DNS Security Algorithm Numbers" [IANA]. In this registry, each signature algorithm is identified by an 8-bit value. The entries in this registry with "Y" in the "Zone Signing" column are appropriate for use with this protocol.

For interoperability, all implementations of this protocol MUST support the RSASHA256 algorithm (identified by the value 0x08). Implementation MAY support any other registered "Zone Signing" signature algorithms.

3. Trust On First Use Method

There are three parts to using a key: signing PDUs, verifying the OPEN PDU, and verifying subsequent PDUs.

3.1. Signing a PDU

All signed PDUs are generated in the same way:

* Compose the PDU, with all fields including "Sig Algo" and "Signature Length" set, but omitting the trailing "Signature" field itself. The Certificate Length should be zero and the Certificate field should be empty. This is the "message to be signed" for purposes of the signature algorithm.

*Generate the signature as specified for the chosen algorithm, using the private key of the asymmetric key pair. In general, this will involve first hashing the "message to be signed" then signing the hash, but the precise details may vary with the specific signature algorithm. The result will be a sequence of octets, the length of which MUST be equal to the value in the "Signature Length" field.

*Construct the complete message by appending the signature octets to the otherwise complete message composed above.

In the case of the OPEN PDU, the message to be signed will include the public member of the asymmetric keypair, but as far as the signature algorithm is concerned that's just payload, no different from any other PDU content.

3.2. Verifying the OPEN PDU

The process for verifying an OPEN PDU is slightly different from the process for verifying other PDU types, because the OPEN PDU also establishes the session key.

*Verify that the PDU is syntactically correct, and extract the Auth Type, Key, Sig Type, and Signature fields.

*Verify that Auth Type and Sig Type refer to the same algorithm suite, and that said algorithm suite is one that the implementation understands.

*Construct the "message to be verified" by truncating the PDU to remove the Signature field (in practice this should not require copying any data, just subtract the signature length from the PDU length).

*Verify the message constructed above against the public key using the rules for the specific signature suite.

*Record Auth Type and Key as this sessions's authentication type and session key, for use in verifying subseuqent PDUs.

If any of the above verification steps fail, generate an error using error code 2 ("Authorization failure in OPEN").

3.3. Verifying Other PDUs

The process for verifying non-OPEN PDUs is slightly simpler, but follows the same basic pattern as for OPEN PDUs.

*Verify that the PDU is syntactically correct, and extract the Sig Type and Signature fields.

*Verify that Sig Type refers to the same algorithm suite as the Auth Type recorded during verification of the OPEN PDU.

*Construct the "message to be verified" by truncating the PDU to remove the Signature field.

*Verify the message constructed above against the recorded session key using the rules for the specific signature suite.

If any of the above verification steps fail, generate an error using error code 3 ("Signature failure in PDU").

4. Public Key Infrastructure Method

Using a PKI is almost the same as using TOFU, but with one additional step: during verification of an OPEN PDU, after extracting the Key field from the PDU but before attempting to use it to verify the OPEN PDU signature, the receiver MUST verify the received key against the PKI to confirm that it's an authorized key.

Generating an OPEN PDU using the PKI method requires a certificate, which must be supplied via out of band configuration. The certificate is a signature of the public key to be sent in the Key field of the OPEN PDU, signed by the trust anchor private key.

Verifying an OPEN PDU using the PKI method requires the public key of the trust anchor, which the receiver uses to verify the certificate, thereby demonstrating that the supplied public key represents an authorized L3DL speaker in this administrative domain.

We use the term "certificate" here in the generic sense, not as defined in [RFC5280]. X.509 certificates are not used here; X.509 certificates are more complicated than needed for L3DL. The L3DL certificates are just signatures of one key (the public key supplied in the Key field of the OPEN PDU) that can be verified by another trusted public key (the trust anchor).

4.1. Signing OPEN PDU with PKI

Generating and signing the OPEN PDU with the PKI method is almost the same as in Section 3.1. The only difference is that the PKI method MUST supply the appropriate certificate in the Certificate field.

Note that the Auth Type field applies to both the Key and Certificate fields. That is: the certificate uses the same certificate suite as the session keys, L3DL does not support cross-algorithm-suite certification.

4.2. Verifying OPEN PDU with PKI

Verifying the OPEN PDU with PKI is similar to verifying with TOFU as described in Section 3.2, but includes one critical extra step:

After extracting the Key field from the PDU but before verifying the Signature, extract the Certificate field and verfiy that the Certificate is a valid signature of the Key field, according to the rules for the signature suite specified by Auth Type. If this step fails, handle as in Section 3.2.

5. Local Policy

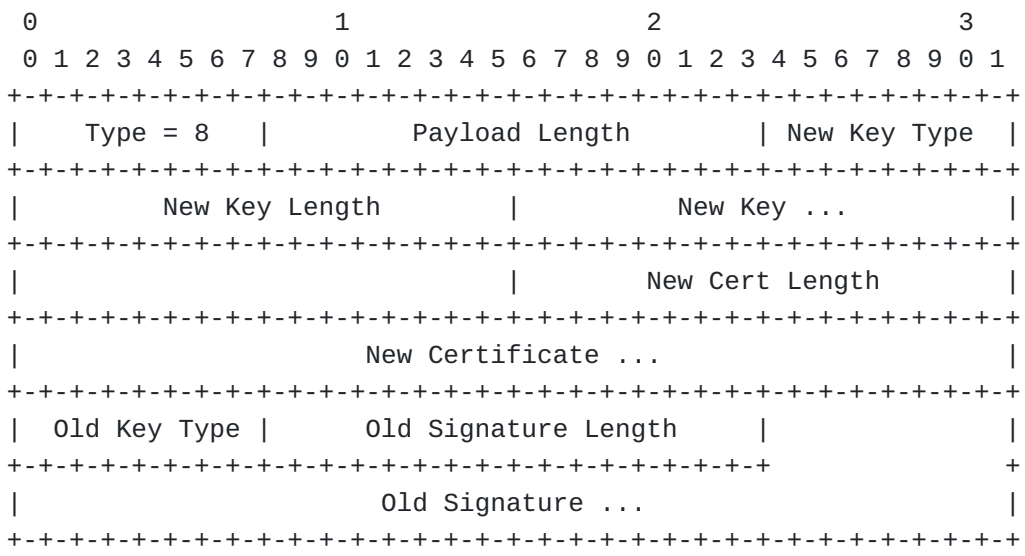
Whether to use TOFU, PKI, or no signatures at all is a matter of local policy, to be decided by the operator. The useful policy combinations for Key and Certificate are probably:

- Not signing: sender need not sign, receiver does not check.
- Require TOFU: sender MUST supply key and receiver MUST check, but L3DL certificates not needed and ignored if sent.
- Allow TOFU: sender MUST supply key and receiver MUST check, receiver SHOULD check certificate if supplyed by sender.
- Require PKI: sender MUST supply key and L3DL certificate, receiver MUST check signature and verify the L3DL certificate.

6. NEWKEY, Key Roll

Modern key management allows for agility in 'rolling' to a new key or even algorithm in case of key expiry, key compromise, or merely prudence. Declaring a new key with an L3DL OPEN PDU would cause serious churn in topology as a new OPEN PDU may cause a withdraw of previously announced encapsulations. Therefore, a gentler rekeying is needed.

Prior to 'rolling' to a new key or new algorithm, a new public/private key pair is generated. If PKI is being used, then the trust anchor also signs the new public key to create a new L3DL certificate.



The New Key Type, New Key Length, New Key, New Cert Length, and New Certificate fields declare the replacement algorithm, key, and L3DL certificate.

The NEWKEY PDU is signed using the current (soon to be old) algorithm and key.

The sender and the receiver should be cautious of signature algorithm downgrade attacks.

To avoid possible race conditions, the receiver SHOULD accept signatures using either the new or old key for a configurable time (default 30 seconds). This is intended to accommodate situations such as senders with high peer out-degree and a single per-device asymmetric key.

If the sender does not receive an ACK in the normal window, including retransmission, then the sender MAY choose to allow a session reset by either issuing a new OPEN PDU or by letting the receiver eventually have a signature failure (error code 3) on a PDU.

The rekeying operation changes the session key and the associated algorithm described in Section 3.3. The NEWKEY PDU itself is verified using the old algorithm and session key. After the NEWKEY PDU has been accepted, subsequent PDUs are verified with the new algorithm and the new session key.

7. Security Considerations

The TOFU method requires a leap of faith to accept the key in the OPEN PDU, as it can not be verified against any authority. Hence it is jokingly referred to as Married On First Date. The assurance it does provide is that subsequent signed PDUs are from the same peer.

And data integrity is a positive side effect of the signature covering the payload.

The PKI method offers assurance that the L3DL certificate, and hence the public key, provided in the OPEN PDU are authorized by a central authority, e.g. the network's security team. The onward assurance of talking to the same peer and data integrity are the same as in the TOFU method.

With the PKI method, automated device provisioning could restrict which L3DL certificates are allowed from which peers on a per interface basis. This would complicate key rolls. Where one draws the line between rigidity, flexibility, and security varies.

The REKEY PDU is open to abuse to create a signature algorithm downgrade attack.

8. IANA Considerations

This document requests the IANA create a new entry in the L3DL PDU Type registry as follows:

PDU Code	PDU Name
8	NEWKEY

This document requests the IANA add registry entries for "TOFU - Trust On First Use" and "PKI" to the L3DL-Signature-Type registry as follows:

Number	Name
1	TOFU - Trust On First Use
2	PKI

9. References

9.1. Normative References

[I-D.ietf-lsvr-l3dl] Bush, R., Austein, R., and K. Patel, "Layer-3 Discovery and Liveness", Work in Progress, Internet-

Draft, draft-ietf-lsvr-l3dl-09, 2 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lsvr-l3dl-09>.

[IANA] "DNS Security Algorithm Numbers", <https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <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>.

9.2. Informative 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>.

Authors' Addresses

Randy Bush
Arrcus & IIJ
5147 Crystal Springs
Bainbridge Island, WA 98110
United States of America

Email: randy@psg.com

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America

Email: housley@vigilsec.com

Rob Austein
Arrcus, Inc.

Email: sra@hactrn.net