        Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality

Abstract

   This document describes a mechanism for encrypting traffic
   encapsulated using the Locator/ID Separation Protocol (LISP). The
   design describes how key exchange is achieved using existing LISP
   control-plane mechanisms as well as how to secure the LISP data plane
   from third-party surveillance attacks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   This document describes a mechanism for encrypting LISP-encapsulated
   traffic. The design describes how key exchange is achieved using
   existing LISP control-plane mechanisms as well as how to secure the
   LISP data plane from third-party surveillance attacks.

   The Locator/ID Separation Protocol [RFC6830] defines a set of
   functions for routers to exchange information used to map from
   non-routable Endpoint Identifiers (EIDs) to routable Routing Locators
   (RLOCs). LISP Ingress Tunnel Routers (ITRs) and Proxy Ingress Tunnel
   Routers (PITRs) encapsulate packets to Egress Tunnel Routers (ETRs)
   and Re-encapsulating Tunnel Routers (RTRs). Packets that arrive at
   the ITR or PITR may not be encrypted, which means no protection or
   privacy of the data is added. When the source host encrypts the data
   stream, encapsulated packets do not need to be encrypted by LISP.
   However, when plaintext packets are sent by hosts, this design can
   encrypt the user payload to maintain privacy on the path between the
   encapsulator (the ITR or PITR) to a decapsulator (ETR or RTR). The
   encrypted payload is unidirectional. However, return traffic uses
   the same procedures but with different key values by the same xTRs or
   potentially different xTRs when the paths between LISP sites are
   asymmetric.

   This document has the following requirements (as well as the general
   requirements from [RFC6973]) for the solution space:

   o Do not require a separate Public Key Infrastructure (PKI) that is
      out of scope of the LISP control-plane architecture.

   o The budget for key exchange MUST be one round-trip time. That is,
      only a two-packet exchange can occur.

   o Use symmetric keying so faster cryptography can be performed in
      the LISP data plane.

   o Avoid a third-party trust anchor if possible.

   o Provide for rekeying when secret keys are compromised.

   o Support Authenticated Encryption with packet integrity checks.

   o Support multiple Cipher Suites so new crypto algorithms can be
      easily introduced.

Satisfying the above requirements provides the following benefits:

o Avoiding a PKI reduces the operational cost of managing a secure
   network. Key management is distributed and independent from any
   other infrastructure.

o Packet transport is optimized due to fewer packet headers. Packet
   loss is reduced by a more efficient key exchange.

o Authentication and privacy are provided with a single mechanism
   thereby providing less per-packet overhead and therefore more
   resource efficiency.

2. Requirements Notation

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

3. Definition of Terms

   AEAD: Authenticated Encryption with Associated Data [RFC5116]

   ICV: Integrity Check Value

   LCAF: LISP Canonical Address Format [RFC8060]

   xTR: A general reference to ITRs, ETRs, RTRs, and PxTRs

4. Overview

   The approach proposed in this document is NOT to rely on the LISP
   mapping system (or any other key-infrastructure system) to store
   security keys. This will provide for a simpler and more secure
   mechanism. Secret shared keys will be negotiated between the ITR and
   the ETR in Map-Request and Map-Reply messages. Therefore, when an
   ITR needs to obtain the RLOC of an ETR, it will get security material
   to compute a shared secret with the ETR.

   The ITR can compute three shared secrets per ETR the ITR is
   encapsulating to. When the ITR encrypts a packet before
   encapsulation, it will identify the key it used for the crypto
   calculation so the ETR knows which key to use for decrypting the
   packet after decapsulation. By using key-ids in the LISP header, we
   can also get fast rekeying functionality.

   The key management described in this document is unidirectional from
   the ITR (the encapsulator) to the ETR (the decapsultor).

5. Diffie-Hellman Key Exchange

   LISP will use a Diffie-Hellman [RFC2631] key exchange sequence and
   computation for computing a shared secret. The Diffie-Hellman
   parameters will be passed via Cipher Suite code-points in Map-Request
   and Map-Reply messages.

   Here is a brief description how Diffie-Hellman works:

```
+-------------------------+---------+---------------------------+
|             ITR         |         |            ETR            |
+------+--------+---------+---------+-----------+--------+------+
|Secret| Public | Calculates | Sends | Calculates | Public |Secret|
+------+--------+---------+---------+-----------+--------+------+
|  i   |  p,g   |         | p,g --> |           |        |  e   |
+------+--------+---------+---------+-----------+--------+------+
|  i   | p,g,I  |g^i mod p=I|  I --> |           | p,g,I  |  e   |
+------+--------+---------+---------+-----------+--------+------+
|  i   | p,g,I  |         |  <-- E  |g^e mod p=E|  p,g   |  e   |
+------+--------+---------+---------+-----------+--------+------+
| i,s  |p,g,I,E |E^i mod p=s|        |I^e mod p=s|p,g,I,E | e,s  |
+------+--------+---------+---------+-----------+--------+------+
```

         Public-Key Exchange for Computing a Shared Private Key [DH]

   Diffie-Hellman parameters 'p' and 'g' must be the same values used by
   the ITR and ETR. The ITR computes public key 'I' and transmits 'I'
   in a Map-Request packet. When the ETR receives the Map-Request, it
   uses parameters 'p' and 'g' to compute the ETR's public key 'E'. The
   ETR transmits 'E' in a Map-Reply message. At this point, the ETR has
   enough information to compute 's', the shared secret, by using 'I' as
   the base and the ETR's private key 'e' as the exponent. When the ITR
   receives the Map-Reply, it uses the ETR's public key 'E' with the
   ITR's private key 'i' to compute the same 's' shared secret the ETR
   computed. The value 'p' is used as a modulus to create the width of
   the shared secret 's' (see Section 6).

6. Encoding and Transmitting Key Material

   The Diffie-Hellman key material is transmitted in Map-Request and
   Map-Reply messages. Diffie-Hellman parameters are encoded in the
   LISP Security Key LCAF Type [RFC8060].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           AFI = 16387         |     Rsvd1     |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 11   |     Rsvd2     |             6 + n             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Key Count   |     Rsvd3     | Cipher Suite  |    Rsvd4    |R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Key Length          |      Public Key Material ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   ... Public Key Material                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             AFI = x           |      Locator Address ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
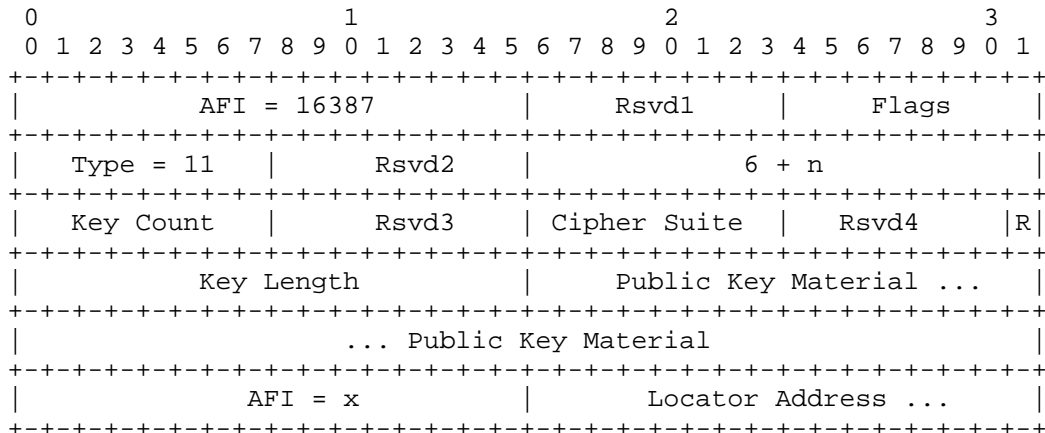
   Cipher Suite Field Contains DH Key Exchange and Cipher/Hash Functions

   The Key Count field encodes the number of {'Key-Length',
   'Key-Material'} fields included in the encoded LCAF. The maximum
   number of keys that can be encoded is three, each identified by
   key-id 1, followed by key-id 2, and finally key-id 3.

   The R bit is not used for this use case of the Security Key LCAF Type
   but is reserved for [LISP-DDT] security. Therefore, the R bit SHOULD
   be transmitted as 0 and MUST be ignored on receipt.

Cipher Suite 0:
    Reserved

Cipher Suite 1 (LISP_2048MODP_AES128_CBC_SHA256):
    Diffie-Hellman Group: 2048-bit MODP [RFC3526]
    Encryption: AES with 128-bit keys in CBC mode [AES-CBC]
    Integrity: Integrated with AEAD_AES_128_CBC_HMAC_SHA_256 [AES-CBC]
    IV length: 16 bytes
    KDF: HMAC-SHA-256

Cipher Suite 2 (LISP_EC25519_AES128_CBC_SHA256):
    Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
    Encryption: AES with 128-bit keys in CBC mode [AES-CBC]
    Integrity: Integrated with AEAD_AES_128_CBC_HMAC_SHA_256 [AES-CBC]
    IV length: 16 bytes
    KDF: HMAC-SHA-256

Cipher Suite 3 (LISP_2048MODP_AES128_GCM):
    Diffie-Hellman Group: 2048-bit MODP [RFC3526]
    Encryption: AES with 128-bit keys in GCM mode [RFC5116]
    Integrity: Integrated with AEAD_AES_128_GCM [RFC5116]
    IV length: 12 bytes
    KDF: HMAC-SHA-256

Cipher Suite 4 (LISP_3072MODP_AES128_GCM):
    Diffie-Hellman Group: 3072-bit MODP [RFC3526]
    Encryption: AES with 128-bit keys in GCM mode [RFC5116]
    Integrity: Integrated with AEAD_AES_128_GCM [RFC5116]
    IV length: 12 bytes
    KDF: HMAC-SHA-256

Cipher Suite 5 (LISP_256_EC25519_AES128_GCM):
    Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
    Encryption: AES with 128-bit keys in GCM mode [RFC5116]
    Integrity: Integrated with AEAD_AES_128_GCM [RFC5116]
    IV length: 12 bytes
    KDF: HMAC-SHA-256

Cipher Suite 6 (LISP_256_EC25519_CHACHA20_POLY1305):
    Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
    Encryption: Chacha20-Poly1305 [CHACHA-POLY] [RFC7539]
    Integrity: Integrated with AEAD_CHACHA20_POLY1305 [CHACHA-POLY]
    IV length: 8 bytes
    KDF: HMAC-SHA-256

The Public Key Material field contains the public key generated by one of the Cipher Suites defined above. The length of the key, in octets, is encoded in the Key Length field.

When an ITR, PITR, or RTR sends a Map-Request, they will encode their own RLOC in the Security Key LCAF Type format within the ITR-RLOCs field. When an ETR or RTR sends a Map-Reply, they will encode their RLOCs in Security Key LCAF Type format within the RLOC-record field of each EID-record supplied.

If an ITR, PITR, or RTR sends a Map-Request with the Security Key LCAF Type included and the ETR or RTR does not want to have encapsulated traffic encrypted, they will return a Map-Reply with no RLOC-records encoded with the Security Key LCAF Type. This signals to the ITR, PITR, or RTR not to encrypt traffic (it cannot encrypt traffic anyway since no ETR public key was returned).

Likewise, if an ITR or PITR wishes to include multiple key-ids in the Map-Request, but the ETR or RTR wishes to use some but not all of the key-ids, it returns a Map-Reply only for those key-ids it wishes to use.

7. Shared Keys Used for the Data Plane

When an ITR or PITR receives a Map-Reply accepting the Cipher Suite sent in the Map-Request, it is ready to create data-plane keys. The same process is followed by the ETR or RTR returning the Map-Reply.

The first step is to create a shared secret, using the peer's shared Diffie-Hellman Public Key Material combined with the device's own private keying material, as described in Section 5. The Diffie-Hellman parameters used are defined in the Cipher Suite sent in the Map-Request and copied into the Map-Reply.

The resulting shared secret is used to compute an AEAD-key for the algorithms specified in the Cipher Suite. A Key Derivation Function (KDF) in counter mode, as specified by [NIST-SP800-108], is used to generate the data-plane keys. The amount of keying material that is derived depends on the algorithms in the Cipher Suite.

The inputs to the KDF are as follows:

o KDF function. This is HMAC-SHA-256 in this document, but generally specified in each Cipher Suite definition.

o A key for the KDF function. This is the computed Diffie-Hellman shared secret.

   o Context that binds the use of the data-plane keys to this session.
      The context is made up of the following fields, which are
      concatenated and provided as the data to be acted upon by the KDF
      function. A Context is made up of the following components:

      * A counter, represented as a two-octet value in network byte
         order.

      * The null-terminated string "lisp-crypto".

      * The ITR's nonce from the Map-Request the Cipher Suite was
         included in.

      * The number of bits of keying material required (L), represented
         as a two-octet value in network byte order.

   The counter value in the context is first set to 1. When the amount
   of keying material exceeds the number of bits returned by the KDF
   function, then the KDF function is called again with the same inputs
   except that the counter increments for each call. When enough keying
   material is returned, it is concatenated and used to create keys.

   For example, AES with 128-bit keys requires 16 octets (128 bits) of
   keying material, and HMAC-SHA1-96 requires another 16 octets (128
   bits) of keying material in order to maintain a consistent 128 bits
   of security. Since 32 octets (256 bits) of keying material are
   required, and the KDF function HMAC-SHA-256 outputs 256 bits, only
   one call is required. The inputs are as follows:

   key-material = HMAC-SHA-256(dh-shared-secret, context)

       where: context = 0x0001 || "lisp-crypto" || <itr-nonce> || 0x0100

   In contrast, a Cipher Suite specifying AES with 256-bit keys requires
   32 octets (256 bits) of keying material, and HMAC-SHA256-128 requires
   another 32 octets (256 bits) of keying material in order to maintain
   a consistent 256 bits of security. Since 64 octets (512 bits) of
   keying material are required, and the KDF function HMAC-SHA-256
   outputs 256 bits, two calls are required.

    key-material-1 = HMAC-SHA-256(dh-shared-secret, context)

       where: context = 0x0001 || "lisp-crypto" || <itr-nonce> || 0x0200

    key-material-2 = HMAC-SHA-256(dh-shared-secret, context)

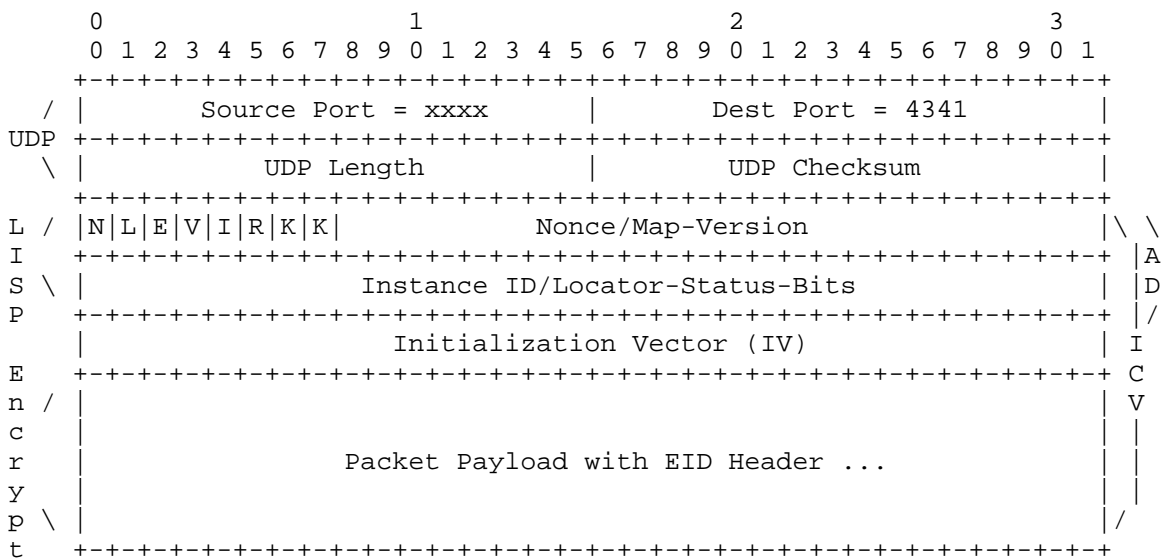   where: context = 0x0002 || "lisp-crypto" || <itr-nonce> || 0x0200

    key-material = key-material-1 || key-material-2

    If the key-material is longer than the required number of bits (L),
    then only the most significant L bits are used.

    From the derived key-material, the most significant 256 bits are used
    for the AEAD-key by AEAD ciphers. The 256-bit AEAD-key is divided
    into a 128-bit encryption key and a 128-bit integrity check key
    internal to the cipher used by the ITR.

8. Data-Plane Operation

    The LISP encapsulation header [RFC6830] requires changes to encode
    the key-id for the key being used for encryption.

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /  |       Source Port = xxxx      |       Dest Port = 4341        |
 UDP +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    \  |           UDP Length          |        UDP Checksum           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 L /   |N|L|E|V|I|R|K|K|            Nonce/Map-Version                  |\ \
 I     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |A
 S \   |                 Instance ID/Locator-Status-Bits              | |D
 P     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |/
       |                   Initialization Vector (IV)                 | I
 E     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ C
 n /   |                                                              | V
 c     |                                                              | |
 r     |            Packet Payload with EID Header ...                | |
 y     |                                                              | |
 p \   |                                                              |/
 t     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

       K-bits Indicate When a Packet Is Encrypted and Which Key Is Used

When the KK bits are 00, the encapsulated packet is not encrypted.
When the value of the KK bits is 1, 2, or 3, it encodes the key-id of
the secret keys computed during the Diffie-Hellman
Map-Request/Map-Reply exchange. When the KK bits are not 0, the
payload is prepended with an Initialization Vector (IV). The length
of the IV field is based on the Cipher Suite used. Since all Cipher
Suites defined in this document do Authenticated Encryption with
Associated Data (AEAD), an ICV field does not need to be present in
the packet since it is included in the ciphertext. The Additional
Data (AD) used for the ICV is shown above and includes the LISP
header, the IV field, and the packet payload.

When an ITR or PITR receives a packet to be encapsulated, the device
will first decide what key to use, encode the key-id into the LISP
header, and use that key to encrypt all packet data that follows the
LISP header. Therefore, the outer header, UDP header, and LISP
header travel as plaintext.

At the time of writing, there is an open working group item to
discuss if the data encapsulation header needs change for encryption
or any new applications. This document proposes changes to the
existing header so experimentation can continue without making large
changes to the data plane at this time. This document allocates two
bits of the previously unused three flag bits (note the R-bit above
is still a reserved flag bit, as documented in [RFC6830]) for the KK
bits.

9. Procedures for Encryption and Decryption

When an ITR, PITR, or RTR encapsulates a packet and has already
computed an AEAD-key (detailed in Section 7) that is associated with
a destination RLOC, the following encryption and encapsulation
procedures are performed:

1. The encapsulator creates an IV and prepends the IV value to the
    packet being encapsulated. For GCM and ChaCha20 Cipher Suites,
    the IV is incremented for every packet (beginning with a value of
    1 in the first packet) and sent to the destination RLOC. For CBC
    Cipher Suites, the IV is a new random number for every packet
    sent to the destination RLOC. For the ChaCha20 Cipher Suite, the
    IV is an 8-byte random value that is appended to a 4-byte counter
    that is incremented for every packet (beginning with a value of 1
    in the first packet).

2. Next encrypt with cipher function AES or ChaCha20 using the AEAD-
    key over the packet payload following the AEAD specification
    referenced in the Cipher Suite definition. This does not include
    the IV. The IV must be transmitted as plaintext so the decrypter

can use it as input to the decryption cipher. The payload should
be padded to an integral number of bytes a block cipher may
require. The result of the AEAD operation may contain an ICV,
the size of which is defined by the referenced AEAD
specification. Note that the AD (i.e., the LISP header exactly
as will be prepended in the next step and the IV) must be given
to the AEAD encryption function as the "associated data"
argument.

3. Prepend the LISP header. The key-id field of the LISP header is
    set to the key-id value that corresponds to key-pair used for the
    encryption cipher.

4. Lastly, prepend the UDP header and outer IP header onto the
    encrypted packet and send packet to destination RLOC.

When an ETR, PETR, or RTR receives an encapsulated packet, the
following decapsulation and decryption procedures are performed:

1. The outer IP header, UDP header, LISP header, and IV field are
    stripped from the start of the packet. The LISP header and IV
    are retained and given to the AEAD decryption operation as the
    "associated data" argument.

2. The packet is decrypted using the AEAD-key and the IV from the
    packet. The AEAD-key is obtained from a local-cache associated
    with the key-id value from the LISP header. The result of the
    decryption function is a plaintext packet payload if the cipher
    returned a verified ICV. Otherwise, the packet is invalid and is
    discarded. If the AEAD specification included an ICV, the AEAD
    decryption function will locate the ICV in the ciphertext and
    compare it to a version of the ICV that the AEAD decryption
    function computes. If the computed ICV is different than the ICV
    located in the ciphertext, then it will be considered tampered.

3. If the packet was not tampered with, the decrypted packet is
    forwarded to the destination EID.

10. Dynamic Rekeying

Since multiple keys can be encoded in both control and data messages,
an ITR can encapsulate and encrypt with a specific key while it is
negotiating other keys with the same ETR. As soon as an ETR or RTR
returns a Map-Reply, it should be prepared to decapsulate and decrypt
using the new keys computed with the new Diffie-Hellman parameters
received in the Map-Request and returned in the Map-Reply.

RLOC-probing can be used to change keys or Cipher Suites by the ITR
at any time. And when an initial Map-Request is sent to populate the
ITR's map-cache, the Map-Request flows across the mapping system
where a single ETR from the Map-Reply RLOC-set will respond. If the
ITR decides to use the other RLOCs in the RLOC-set, it MUST send a
Map-Request directly to negotiate security parameters with the ETR.
This process may be used to test reachability from an ITR to an ETR
initially when a map-cache entry is added for the first time, so an
ITR can get both reachability status and keys negotiated with one
Map-Request/Map-Reply exchange.

A rekeying event is defined to be when an ITR or PITR changes the
Cipher Suite or public key in the Map-Request. The ETR or RTR
compares the Cipher Suite and public key it last received from the
ITR for the key-id, and if any value has changed, it computes a new
public key and Cipher Suite requested by the ITR from the Map-Request
and returns it in the Map-Reply. Now a new shared secret is computed
and can be used for the key-id for encryption by the ITR and
decryption by the ETR. When the ITR or PITR starts this process of
negotiating a new key, it must not use the corresponding key-id in
encapsulated packets until it receives a Map-Reply from the ETR with
the same Cipher Suite value it expects (the values it sent in a Map-
Request).

Note when RLOC-probing continues to maintain RLOC reachability and
rekeying is not desirable, the ITR or RTR can either not include the
Security Key LCAF Type in the Map-Request or supply the same key
material as it received from the last Map-Reply from the ETR or RTR.
This approach signals to the ETR or RTR that no rekeying event is
requested.

11. Future Work

   For performance considerations, newer Elliptic-Curve Diffie-Hellman
   (ECDH) groups can be used as specified in [RFC4492] and [RFC6090] to
   reduce CPU cycles required to compute shared secret keys.

   For better security considerations as well as to be able to build
   faster software implementations, newer approaches to ciphers and
   authentication methods will be researched and tested. Some examples
   are ChaCha20 and Poly1305 [CHACHA-POLY] [RFC7539].

12. Security Considerations

12.1. SAAG Support

   The LISP working group received security advice and guidance from the
   Security Area Advisory Group (SAAG). The SAAG has been involved
   early in the design process, and their input and reviews have been
   included in this document.

   Comments from the SAAG included:

   1. Do not use asymmetric ciphers in the data plane.

   2. Consider adding ECDH early in the design.

   3. Add Cipher Suites because ciphers are created more frequently
       than protocols that use them.

   4. Consider the newer AEAD technology so authentication comes with
       doing encryption.

12.2. LISP-Crypto Security Threats

   Since ITRs and ETRs participate in key exchange over a public
   non-secure network, a man in the middle (MITM) could circumvent the
   key exchange and compromise data-plane confidentiality. This can
   happen when the MITM is acting as a Map-Replier and provides its own
   public key so the ITR and the MITM generate a shared secret key
   between them. If the MITM is in the data path between the ITR and
   ETR, it can use the shared secret key to decrypt traffic from the
   ITR.

   Since LISP can secure Map-Replies by the authentication process
   specified in [LISP-SEC], the ITR can detect when a MITM has signed a
   Map-Reply for an EID-prefix for which it is not authoritative. When
   an ITR determines that the signature verification fails, it discards
   and does not reuse the key exchange parameters, avoids using the ETR
   for encapsulation, and issues a severe log message to the network
   administrator. Optionally, the ITR can send RLOC-probes to the
   compromised RLOC to determine if the authoritative ETR is reachable.
   And when the ITR validates the signature of a Map-Reply, it can begin
   encrypting and encapsulating packets to the RLOC of ETR.

13. IANA Considerations

   This document describes a mechanism for encrypting LISP-encapsulated
   packets based on Diffie-Hellman key exchange procedures. During the
   exchange, the devices have to agree on a Cipher Suite to be used
   (i.e., the cipher and hash functions used to encrypt/decrypt and to
   sign/verify packets). The 8-bit Cipher Suite field is reserved for
   such purpose in the security material section of the Map-Request and
   Map-Reply messages.

   IANA has created a new registry (as outlined in [RFC5226]) titled
   "LISP Crypto Cipher Suite". Initial values for the registry are
   provided below. Future assignments are to be made on a "First Come,
   First Served" basis [RFC5226].

```
   +-----+---------------------------------------------+------------+
   |Value| Suite                                       | Reference  |
   +-----+---------------------------------------------+------------+
   |  0  | Reserved                                    | Section 6  |
   +-----+---------------------------------------------+------------+
   |  1  | LISP_2048MODP_AES128_CBC_SHA256             | Section 6  |
   +-----+---------------------------------------------+------------+
   |  2  | LISP_EC25519_AES128_CBC_SHA256              | Section 6  |
   +-----+---------------------------------------------+------------+
   |  3  | LISP_2048MODP_AES128_GCM                    | Section 6  |
   +-----+---------------------------------------------+------------+
   |  4  | LISP_3072MODP_AES128_GCM                    | Section 6  |
   +-----+---------------------------------------------+------------+
   |  5  | LISP_256_EC25519_AES128_GCM                 | Section 6  |
   +-----+---------------------------------------------+------------+
   |  6  | LISP_256_EC25519_CHACHA20_POLY1305          | Section 6  |
   +-----+---------------------------------------------+------------+
```

                       LISP Crypto Cipher Suites

14. References

14.1. Normative References

   [NIST-SP800-108]
             National Institute of Standards and Technology,
             "Recommendation for Key Derivation Using Pseudorandom
             Functions", NIST Special Publication SP 800-108,
             DOI 10.6028/NIST.SP.800-108, October 2009.

   [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>.

   [RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method",
             RFC 2631, DOI 10.17487/RFC2631, June 1999,
             <http://www.rfc-editor.org/info/rfc2631>.

   [RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP)
             Diffie-Hellman groups for Internet Key Exchange (IKE)",
             RFC 3526, DOI 10.17487/RFC3526, May 2003,
             <http://www.rfc-editor.org/info/rfc3526>.

   [RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B.
             Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites
             for Transport Layer Security (TLS)", RFC 4492,
             DOI 10.17487/RFC4492, May 2006,
             <http://www.rfc-editor.org/info/rfc4492>.

   [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated
             Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008,
             <http://www.rfc-editor.org/info/rfc5116>.

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

   [RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic
             Curve Cryptography Algorithms", RFC 6090,
             DOI 10.17487/RFC6090, February 2011,
             <http://www.rfc-editor.org/info/rfc6090>.

   [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
             Locator/ID Separation Protocol (LISP)", RFC 6830,
             DOI 10.17487/RFC6830, January 2013,
             <http://www.rfc-editor.org/info/rfc6830>.

   [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J.,
              Morris, J., Hansen, M., and R. Smith, "Privacy
              Considerations for Internet Protocols", RFC 6973,
              DOI 10.17487/RFC6973, July 2013,
              <http://www.rfc-editor.org/info/rfc6973>.

   [RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF
              Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015,
              <http://www.rfc-editor.org/info/rfc7539>.

   [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical
              Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060,
              February 2017, <http://www.rfc-editor.org/info/rfc8060>.

14.2. Informative References

   [AES-CBC] McGrew, D., Foley, J., and K. Paterson, "Authenticated
              Encryption with AES-CBC and HMAC-SHA", Work in Progress,
              draft-mcgrew-aead-aes-cbc-hmac-sha2-05, July 2014.

   [CHACHA-POLY]
              Langley, A. and W. Chang, "ChaCha20 and Poly1305 based
              Cipher Suites for TLS", Work in Progress,
              draft-agl-tls-chacha20poly1305-04, November 2013.

   [CURVE25519]
              Bernstein, D., "Curve25519: new Diffie-Hellman speed
              records", DOI 10.1007/11745853_14,
              <http://www.iacr.org/cryptodb/archive/2006/
              PKC/3351/3351.pdf>.

   [DH]   Wikipedia, "Diffie-Hellman key exchange", January 2017,
              <https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%9
              3Hellman_key_exchange&oldid=759611604>.

   [LISP-DDT] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A.
              Smirnov, "LISP Delegated Database Tree", Work in
              Progress, draft-ietf-lisp-ddt-08, September 2016.

   [LISP-SEC] Maino, F., Ermagan, V., Cabellos, A., and D. Saucez,
              "LISP-Security (LISP-SEC)", Work in Progress,
              draft-ietf-lisp-sec-12, November 2016.

Acknowledgments

   The authors would like to thank Dan Harkins, Joel Halpern, Fabio
   Maino, Ed Lopez, Roger Jorgensen, and Watson Ladd for their interest,
   suggestions, and discussions about LISP data-plane security. An
   individual thank you to LISP WG Chair Luigi Iannone for shepherding
   this document as well as contributing to the IANA Considerations
   section.

   The authors would like to give a special thank you to Ilari Liusvaara
   for his extensive commentary and discussion. He has contributed his
   security expertise to make lisp-crypto as secure as the state of the
   art in cryptography.

   In addition, the support and suggestions from the SAAG working group
   were helpful and appreciated.

Authors' Addresses

   Dino Farinacci
   lispers.net
   San Jose, California 95120
   United States of America

   Phone: 408-718-2001
   Email: farinacci@gmail.com

   Brian Weis
   Cisco Systems
   170 West Tasman Drive
   San Jose, California 95124-1706
   United States of America

   Phone: 408-526-4796
   Email: bew@cisco.com