Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: December 2, 2016

D. Farinacci
lispers.net
B. Weis
cisco Systems
May 31, 2016

LISP Data-Plane Confidentiality
draft-ietf-lisp-crypto-04

Abstract

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.

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 December 2, 2016.

Table of Contents

1. Introduction

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 ITRs and PITRs encapsulate packets to ETRs and RTRs. Packets that arrive at the ITR or PITR are typically not modified. Which means no protection or privacy of the data is added. If the source host encrypts the data stream then the encapsulated packets can be encrypted but would be redundant. 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 draft has the following requirements for the solution space:

- Do not require a separate Public Key Infrastructure (PKI) that is out of scope of the LISP control-plane architecture.
- The budget for key exchange MUST be one round-trip time. That is, only a two packet exchange can occur.
- Use symmetric keying so faster cryptography can be performed in the LISP data plane.
- Avoid a third-party trust anchor if possible.
- Provide for rekeying when secret keys are compromised.
- Support Authenticated Encryption with packet integrity checks.
- Support multiple cipher suites so new crypto algorithms can be easily introduced.

2. Overview

The approach proposed in this draft is to NOT 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.

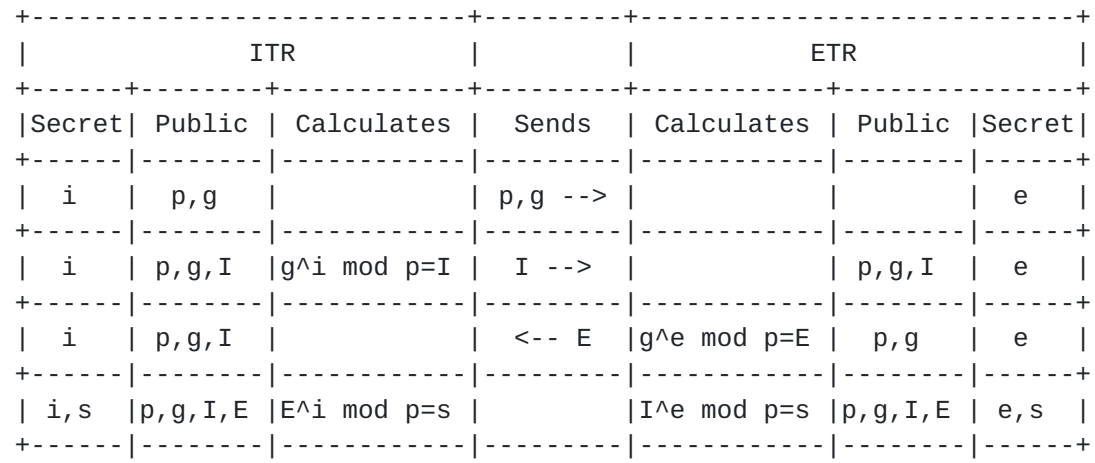
The ITR can compute 3 shared-secrets per ETR the ITR is encapsulating to. And when the ITR encrypts a packet before encapsulation, it will identify the key it used for the crypto calculation so the ETR knows which key to use for decrypting the packet after decapsulation. By using key-ids in the LISP header, we can also get real-time rekeying functionality.

When an ETR (when it is also an ITR) encapsulates packets to this ITR (when it is also an ETR), a separate key exchange and shared-secret computation is performed. The key management described in this documemnt is unidirectional from the ITR (the encapsulator) to the ETR (the decapsultor).

3. 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 Diff-Hellman works:

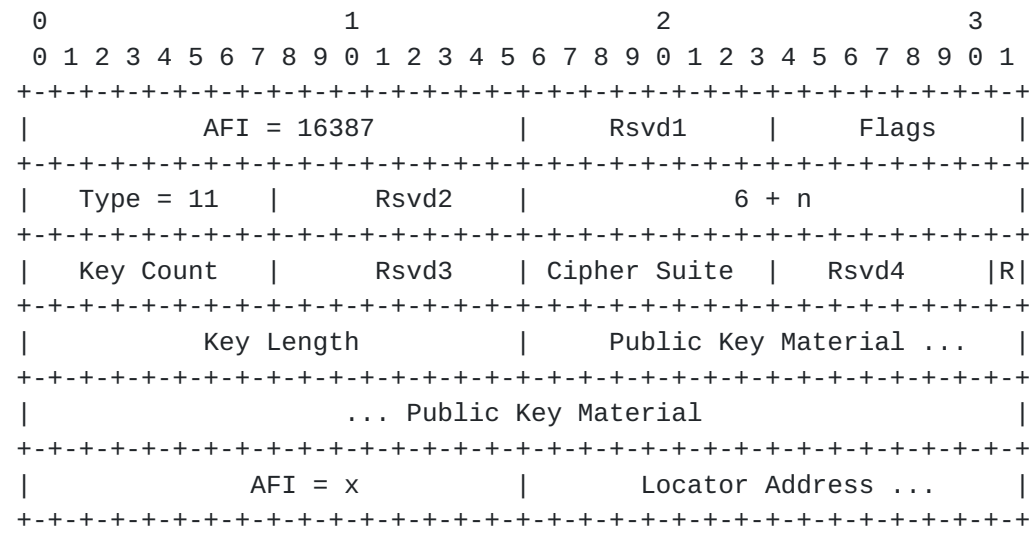


Public-key exchange for computing a shared private key [DH]

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'.

4. 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 Type LCAF [LCAF].



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 are 3, each identified by key-id 1, followed by key-id 2, an finally key-id 3.

The 'R' bit is not used for this use-case of the Security Type LCAF but is reserved for [LISP-DDT] security.

Cipher Suite 0:
Reserved

Cipher Suite 1:
Diffie-Hellman Group: 2048-bit MODP [RFC3526]
Encryption: AES with 128-bit keys in CBC mode [AES-CBC]
Integrity: Integrated with [AES-CBC] AEAD [RFC5116] encryption
IV length: 16 bytes

Cipher Suite 2:
Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
Encryption: AES with 128-bit keys in CBC mode [AES-CBC]
Integrity: HMAC-SHA1-96 [RFC2404]
IV length: 16 bytes

Cipher Suite 3:
Diffie-Hellman Group: 2048-bit MODP [RFC3526]
Encryption: AES with 128-bit keys in GCM mode [AES-GCM]
Integrity: Integrated with [AES-GCM] AEAD [RFC5116] encryption
IV length: 12 bytes

Cipher Suite 4:
Diffie-Hellman Group: 3072-bit MODP [RFC3526]
Encryption: AES with 128-bit keys in GCM mode [AES-GCM]
Integrity: Integrated with [AES-GCM] AEAD [RFC5116] encryption
IV length: 12 bytes

Cipher Suite 5:
Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
Encryption: AES with 128-bit keys in GCM mode [AES-GCM]
Integrity: Integrated with [AES-GCM] AEAD [RFC5116] encryption
IV length: 12 bytes

Cipher Suite 6:
Diffie-Hellman Group: 256-bit Elliptic-Curve 25519 [CURVE25519]
Encryption/Integrity: Chacha20-Poly1305 [CHACHA-POLY] [RFC7539]
Integrity: Integrated with Chacha20-Poly1305 AEAD [RFC1116] encryption
IV length: 8 bytes

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 or PITR send a Map-Request, they will encode their own RLOC in the Security Type LCAF format within the ITR-RLOCs field. When a ETR or RTR sends a Map-Reply, they will encode their RLOCs in

Security Type LCAF format within the RLOC-record field of each EID-record supplied.

If an ITR or PITR sends a Map-Request with the Security Type LCAF 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 Type LCAF. This signals to the ITR or PITR that it should not encrypt traffic (it cannot encrypt traffic anyways since no ETR public-key was returned).

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

5. 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 device's own private keying material as described in Section 3. The Diffie-Hellman group used is 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:

- KDF function. This is HMAC-SHA-256.
- A key for the KDF function. This is the computed Diffie-Hellman shared secret.
- 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.

Context:

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

- The null-terminated string "lisp-crypto".
- The ITR's nonce from the 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

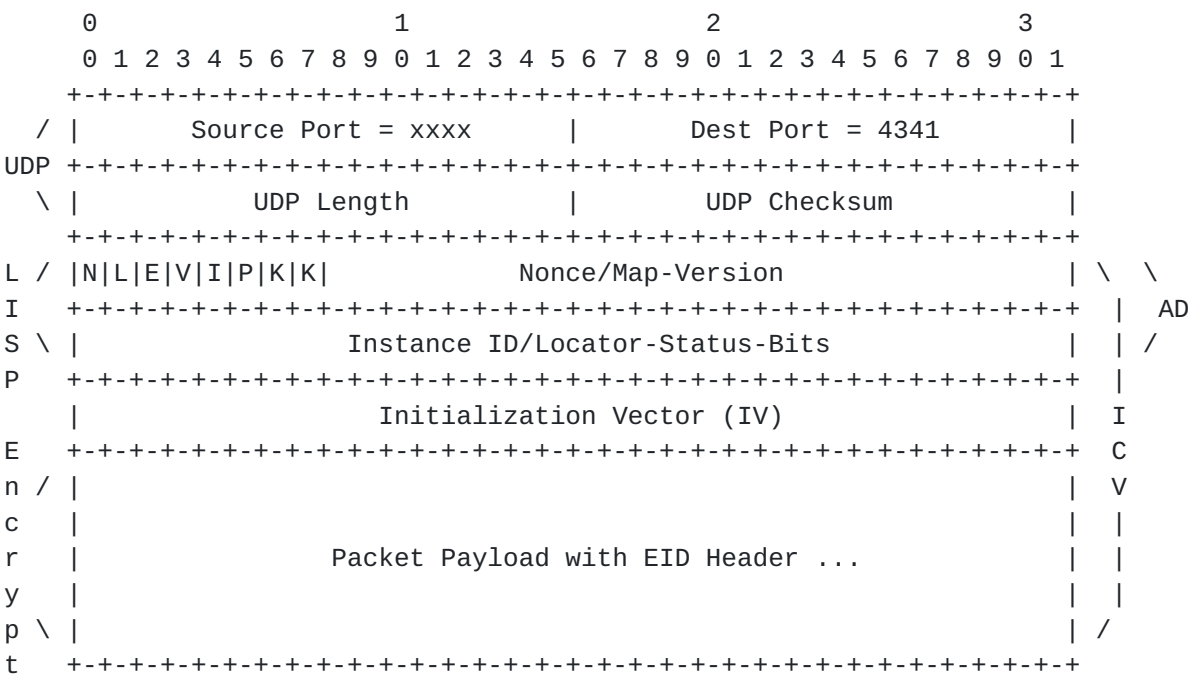
If the key-material is longer than the required number of bits (L), then only the most significant L bits are used.

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.

6. Data-Plane Operation

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



K-bits indicate when packet is encrypted and which key used

When the KK bits are 00, the encapsulated packet is not encrypted. When the value of the KK bits are 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 (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, they 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.

There is an open working group item to discuss if the data encapsulation header needs change for encryption or any new applications. This draft proposes changes to the existing header so experimentation can continue without making large changes to the data-plane at this time.

7. Procedures for Encryption and Decryption

When an ITR, PITR, or RTR encapsulate a packet and have already computed an AEAD-key (detailed in section Section 5) 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 Chacha 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 Chacha 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 receive 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 has been tampered with, is dropped, and an optional log message may be issued. 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.

8. 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. 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 Type LCAF 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.

9. 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].

10. Security Considerations

10.1. SAAG Support

The LISP working group has and will continue to seek help from the SAAG working group for security advice. The SAAG has been involved early in the design process so they have early input and review.

10.2. 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, provides its own public key so the ITR and the MITM generate a shared secret key among each other. 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 it is not authoritative for. When an ITR determines 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 can reach the authoritative ETR.

And when the ITR validates the signature of a Map-Reply, it can begin encrypting and encapsulating packets to the RLOC of ETR.

11. IANA Considerations

This draft may require the use of the registry that selects Security parameters. Rather than convey the key exchange parameters and crypto functions directly in LISP control packets, the cipher suite values can be assigned and defined in a registry. For example, Diffie-Hellman group-id values can be used from [RFC2409] and [RFC3526].

This draft specifies how the 7-bit cipher suite values from the Security Type LCAF are partitioned. The partitions are:

```
0:      Reserved
1-96:   Allocated by registry, but first 3 values defined in this document
97-127: Private use
```

12. References

12.1. Normative References

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <http://www.rfc-editor.org/info/rfc2409>.

[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>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

[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>.

[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>.

[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>.

12.2. Informative References

[AES-CBC] McGrew, D., Foley, J., and K. Paterson, "Authenticated Encryption with AES-CBC and HMAC-SHA", draft-mcgrew-aead-aes-cbc-hmac-sha2-05.txt (work in progress).

[CHACHA-POLY] Langley, A., "ChaCha20 and Poly1305 based Cipher Suites for TLS", draft-agl-tls-chacha20poly1305-00 (work in progress).

[CURVE25519] Bernstein, D., "Curve25519: new Diffie-Hellman speed records", Publication http://www.iacr.org/cryptodb/archive/2006/PKC/3351/3351.pdf.

[DH] "Diffie-Hellman key exchange", Wikipedia http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange.

[LCAF] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format", draft-ietf-lisp-lcaf-13.txt (work in progress).

[LISP-DDT] Fuller, V., Lewis, D., Ermaagan, V., and A. Jain, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-06 (work in progress).

[LISP-SEC]
Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Secuirty (LISP-SEC)", draft-ietf-lisp-sec-10 (work in progress).

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

Appendix A. 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.

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 appreciative.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-lisp-crypto-04.txt

- Posted May 2016.
- Update document timer from expiration.

B.2. Changes to draft-ietf-lisp-crypto-03.txt

- Posted December 2015.
- Changed cipher suite allocations. We now have 2 AES-CBC cipher suites for compatibility, 3 AES-GCM cipher suites that are faster ciphers that include AE and a Chacha20-Poly1305 cipher suite which is the fastest but not totally proven/accepted..
- Remove 1024-bit DH keys for key exchange.
- Make clear that AES and chacha20 ciphers use AEAD so part of encrytion/decryption does authentication.
- Make it more clear that separate key pairs are used in each direction between xTRs.

- Indicate that the IV length is different per cipher suite.

- Use a counter based IV for every packet for AEAD ciphers. Previously text said to use a random number. But CBC ciphers, use a random number.

- Indicate that key material is sent in network byte order (big endian).

- Remove A-bit from Security Type LCAF. No need to do authentication only with the introduction of AEAD ciphers. These ciphers can do authentication. So you get ciphertext for free.

- Remove language that refers to "encryption-key" and "integrity-key". Used term "AEAD-key" that is used by the AEAD cipher suites that do encryption and authenticaiton internal to the cipher.

B.3. Changes to draft-ietf-lisp-crypto-02.txt

- Posted September 2015.

- Add cipher suite for Elliptic Curve 25519 DH exchange.

- Add cipher suite for Chacha20/Poly1305 ciphers.

B.4. Changes to draft-ietf-lisp-crypto-01.txt

- Posted May 2015.

- Create cipher suites and encode them in the Security LCAF.

- Add IV to beginning of packet header and ICV to end of packet.

- AEAD procedures are now part of encrpytion process.

B.5. Changes to draft-ietf-lisp-crypto-00.txt

- Posted January 2015.

- Changing draft-farinacci-lisp-crypto-01 to draft-ietf-lisp-crypto-00. This draft has become a working group document

- Add text to indicate the working group may work on a new data encapsulation header format for data-plane encryption.

B.6. Changes to draft-farinacci-lisp-crypto-01.txt

- Posted July 2014.
- Add Group-ID to the encoding format of Key Material in a Security Type LCAF and modify the IANA Considerations so this draft can use key exchange parameters from the IANA registry.
- Indicate that the R-bit in the Security Type LCAF is not used by lisp-crypto.
- Add text to indicate that ETRs/RTRs can negotiate less number of keys from which the ITR/PITR sent in a Map-Request.
- Add text explaining how LISP-SEC solves the problem when a man-in-the-middle becomes part of the Map-Request/Map-Reply key exchange process.
- Add text indicating that when RLOC-probing is used for RLOC reachability purposes and rekeying is not desired, that the same key exchange parameters should be used so a reallocation of a pubic key does not happen at the ETR.
- Add text to indicate that ECDH can be used to reduce CPU requirements for computing shared secret-keys.

B.7. Changes to draft-farinacci-lisp-crypto-00.txt

- Initial draft posted February 2014.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, California 95120
USA

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

Brian Weis
cisco Systems
170 West Tasman Drive
San Jose, California 95124-1706
USA

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