

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: March 19, 2020

D. Farinacci  
lispers.net  
E. Nordmark  
Zededa  
September 16, 2019

**LISP Control-Plane ECDSA Authentication and Authorization**  
**draft-ietf-lisp-ecdsa-auth-02**

**Abstract**

This draft describes how LISP control-plane messages can be individually authenticated and authorized without a a priori shared-key configuration. Public-key cryptography is used with no new PKI infrastructure required.

**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 19, 2020.

**Copyright Notice**

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                             |   |                    |
|-----------------------------|---|--------------------|
| <a href="#">1.</a>          | Introduction . . . . .  | <a href="#">2</a>  |
| <a href="#">2.</a>          | Definition of Terms . . . . .   | <a href="#">4</a>  |
| <a href="#">3.</a>          | Overview . . . . .  | <a href="#">5</a>  |
| <a href="#">4.</a>          | Public-Key Hash . . . . .   | <a href="#">6</a>  |
| <a href="#">5.</a>          | Hash-EID Mapping Entry . . . . .  | <a href="#">7</a>  |
| <a href="#">6.</a>          | Hash-EID Structure . . . . .  | <a href="#">7</a>  |
| <a href="#">7.</a>          | Keys and Signatures . . . . .   | <a href="#">8</a>  |
| <a href="#">8.</a>          | Signed Map-Register Encoding . . . . .                                  | <a href="#">8</a>  |
| <a href="#">9.</a>          | Signed Map-Request Encoding . . . . .                                   | <a href="#">9</a>  |
| <a href="#">10.</a>         | Signed Map-Notify Encoding . . . . .                                    | <a href="#">10</a> |
| <a href="#">11.</a>         | Other Uses . . . . .  | <a href="#">10</a> |
| <a href="#">12.</a>         | EID Authorization . . . . .   | <a href="#">11</a> |
| <a href="#">13.</a>         | Security Considerations . . . . .                                       | <a href="#">13</a> |
| <a href="#">14.</a>         | IANA Considerations . . . . .   | <a href="#">13</a> |
| <a href="#">15.</a>         | References . . . . .  | <a href="#">13</a> |
| <a href="#">15.1.</a>       | Normative References . . . . .  | <a href="#">13</a> |
| <a href="#">15.2.</a>       | Informative References . . . . .  | <a href="#">15</a> |
| <a href="#">Appendix A.</a> | Acknowledgments . . . . .   | <a href="#">15</a> |
| <a href="#">Appendix B.</a> | Document Change Log . . . . .   | <a href="#">16</a> |
| <a href="#">B.1.</a>        | Changes to <a href="#">draft-ietf-lisp-ecdsa-auth-02</a> . . . . .      | <a href="#">16</a> |
| <a href="#">B.2.</a>        | Changes to <a href="#">draft-ietf-lisp-ecdsa-auth-01</a> . . . . .      | <a href="#">16</a> |
| <a href="#">B.3.</a>        | Changes to <a href="#">draft-ietf-lisp-ecdsa-auth-00</a> . . . . .      | <a href="#">16</a> |
| <a href="#">B.4.</a>        | Changes to <a href="#">draft-farinacci-lisp-ecdsa-auth-03</a> . . . . . | <a href="#">16</a> |
| <a href="#">B.5.</a>        | Changes to <a href="#">draft-farinacci-lisp-ecdsa-auth-02</a> . . . . . | <a href="#">16</a> |
| <a href="#">B.6.</a>        | Changes to <a href="#">draft-farinacci-lisp-ecdsa-auth-01</a> . . . . . | <a href="#">16</a> |
| <a href="#">B.7.</a>        | Changes to <a href="#">draft-farinacci-lisp-ecdsa-auth-00</a> . . . . . | <a href="#">17</a> |
|                             | Authors' Addresses . . . . .  | <a href="#">17</a> |

**1. Introduction**

The LISP architecture and protocols [[RFC6830](#)] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which provide an architecture to build overlays on top of the underlying Internet. Mapping EIDs to RLOC-sets is accomplished with a Mapping Database System. EIDs and RLOCs come in many forms than just IP addresses, using a general syntax that includes Address Family Identifier (AFI) [[RFC1700](#)]. Not only IP addresses, but other addressing information have privacy requirements. Access to private information is granted only to those who are authorized and authenticated. Using asymmetric keying with public key cryptography enforces authentication for entities that read from and write to the mapping system. The proposal described in this document takes advantage of the latest in Elliptic Curve Cryptography.

In this proposal the EID is derived from a public key, and the corresponding private key is used to authenticate and authorize Map-



Register messages. Thus only the owner of the corresponding private key can create and update mapping entries from the EID. Furthermore, the same approach is used to authenticate Map-Request messages. This in combination with the mapping database containing authorization information for Map-Requests is used to restrict which EIDs can lookup up the RLOCs for another EID.

This specification introduces how to use the Distinguished-Name AFI [[AFI](#)] and the [[RFC8060](#)] LCAF JSON Type to encode public keys and signatures in the LISP mapping database. The information in the mapping database is used to verify cryptographic signatures in LISP control-plane messages such as the Map-Request and Map-Register.



## **2. Definition of Terms**

**Crypto-EID:** is an IPv6 EID where part of the EID includes a hash value of a public-key. An IPv6 EID is a Crypto-EID when the Map-Server is configured with an Crypto-EID Prefix that matches the IPv6 EID.

**Crypto-EID Hash Length:** is the number of low-order bits in a Crypto-EID which make up the hash of a public-key. The hash length is determined by the Map-Server when it is configured with a Crypto-EID Prefix.

**Crypto-EID Prefix:** is a configuration parameter on the Map-Server that indicates which IPv6 EIDs are Crypto-EIDs and what is the Crypto-EID Hash Length for the IPv6 EID. This can be different for different LISP Instance-IDs.

**Hash-EID:** is a distinguished name EID-record stored in the mapping database. The EID format is 'hash-<pubkey-hash>'. When a key-pair is generated for an endpoint, the produced private-key does not leave the xTR that will register the Crypto-EID. A hash of the public-key is used to produce a Crypto-EID and a Hash-EID. The Crypto-EID is assigned to the endpoint and the xTR that supports the LISP-site registers the Crypto-EID. Another entity registers the Hash-EID mapping with the public-key as an RLOC-record.

**Public-Key RLOC:** is a JSON string that encodes a public-key as an RLOC-record for a Hash-EID mapping entry. The format of the JSON string is '{ "public-key" : "<pubkey>" }'.

**Control-Plane Signature:** a Map-Request or Map-Register sender signs the message with its private key. The format of the signature is a JSON string that includes sender information and the signature value. The JSON string is included in Map-Request and Map-Register messages.

**Signature-ID:** is a Crypto-EID used for a Control-Plane signature to register or request any type of EID. The Signature-ID is included with the JSON-encoded signature in Map-Request and Map-Register messages.

**Multi-Signatures:** multiple signatures are used in LISP when an entity allows and authorized another entity to register an EID. There can be more than one authorizing entities that allow a registering entity to register an EID. The authorizing entities sign their own RLOC-records that are registered and merged into the registering entity's Hash-EID public-key mapping. And when



the registering entity registers the EID, all authorizing entity signatures must be verified by the Map-Server before the EID is accepted.

### **3. Overview**

LISP already has several message authentication mechanisms. They can be found in [[I-D.ietf-lisp-rfc6833bis](#)], [[I-D.ietf-lisp-sec](#)], and [[RFC8061](#)]. The mechanisms in this draft are providing a more granular level of authentication as well as a simpler way to manage keys and passwords.

A client of the mapping system can be authenticated using public-key cryptography. The client is required to have a private/public key-pair where it uses the private-key to sign Map-Requests and Map-Registers. The server, or the LISP entity, that processes Map-Requests and Map-Registers uses the public-key to verify signatures.





The following describes how the mapping system is used to implement the public-key crypto system:

1. An entity registers Hash-EID to Public-Key RLOC mappings. A third-party entity that provides a service can register or the client itself can register.
2. Anyone can lookup the Hash-EID mappings. These mappings are not usually authenticated with the mechanisms in this draft but use the shared configured password mechanisms from [\[I-D.ietf-lisp-rfc6833bis\]](#) that provide group level authentication.
3. When a Crypto-EID, or any EID type, is registered to the mapping system, a signature is included in the Map-Register message. When a non-Crypto-EID is registered a Signature-ID is also included in the Map-Register message.
4. The Map-Server processes the registration by constructing the Hash-EID from the registered Crypto-EID, looks up the Hash-EID in the mapping system, obtains the public-key from the RLOC-record and verifies the signature. If Hash-EID lookup fails or the signature verification fails, the Map-Register is not accepted.
5. When a Crypto-EID, or any EID type, is looked up in the mapping system, a signature is included with a Signature-ID in the Map-Request message.
6. The Map-Server processes the request for a Crypto-EID by constructing the Hash-EID from the Signature-ID included in the Map-Request. The signer-ID is a Crypto-EID that accompanies a signature in the Map-Request. The Hash-EID is looked up in the mapping system, obtains the public-key from the RLOC-record and verifies the Map-Request signature. If the Hash-EID lookup fails or the signature verification fails, the Map-Request is not accepted and a Negative Map-Reply is sent back with an action of "auth-failure".

#### **4. Public-Key Hash**

When a private/public key-pair is created for a node, its IPv6 EID is pre-determined based on the public key generated. Note if the key-pair is compromised or is changed for the node, a new IPv6 EID is assigned for the node.

The sha256 [\[RFC6234\]](#) hex digest function is used to compute the hash. The hash is run over the following hex byte string:



<iid><prefix><pubkey>

Where each field is defined to be:

<iid>: is a 4-byte (leading zeroes filled) binary value of the Instance-ID the EID will be registered with in the mapping database. For example, if the instance-id is 171, then the 4-byte value is 0x000000ab.

<prefix>: is a variable length IPv6 prefix in binary format (with no colons) and IS quad-nibble zero-filled. The length of the prefix is 128 minus the Crypto-EID hash bit length. For example, if the prefix is 2001:5:3::/48, then the 6 byte value is 0x200100050003.

<pubkey>: is a DER [[RFC7468](#)] encoded public-key.

The public-key hash is used to construct the Crypto-EID and Hash-EID.

## 5. Hash-EID Mapping Entry

A Hash-EID is formatted in an EID-record as a Distinguished-Name AFI as specified in [[I-D.farinacci-lisp-name-encoding](#)]. The format of the string is:

EID-record: 'hash-<hash-eid>'

Where <hash-eid> is a public-key hash as described in [Section 4](#). The RLOC-record to encode and store the public-key is in LCAF JSON Type format of the form:

RLOC-record: '{ "public-key" : "<pubkey-base64>" }'

Where <pubkey-base64> is a base64 [[RFC4648](#)] encoding of the public-key generated for the system that is assigned the Hash-EID.

## 6. Hash-EID Structure

Since the Hash-EID is formatted as a distinguished-name AFI, the format of the <hash-eid> for EID 'hash-<hash-eid>' needs to be specified. The format will be an IPv6 address [[RFC3513](#)] where colons are used between quad-nibble characters when the hash bit length is a multiple of 4. And when the hash bit length is not a multiple of 4 but a multiple of 2, a leading 2 character nibble-pair is present. Here are some examples for different hash bit lengths:



Crypto-EID: 2001:5::1111:2222:3333:4444, hash length 64:  
Hash-EID is: 'hash-1111:2222:3333:4444'

Crypto-EID: 2001:5::11:22:33:44, hash length 64:  
Hash-EID is: 'hash-0011:0022:0033:0044'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 80:  
Hash-EID is: 'hash-bbbb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 72:  
Hash-EID is: 'hash-bb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:22:33:4444, hash length 72:  
Hash-EID is: 'hash-bb:1111:0022:0033:4444'

Note when leading zeroes exist in a IPv6 encoded quad between colons, the zeros are included in the quad for the Hash-EID string.

The entity that creates the hash, the entity that registers the Crypto-EID and the Map-Server that uses the hash for Hash-EID lookups MUST agree on the hash bit length.

## **7. Keys and Signatures**

Key generation, message authentication with digital signatures, and signature verification will use the Elliptic Curve Digital Signature Algorithm or ECDSA [[X9.62](#)]. For key generation curve 'NIST256p' is used and recommended.

Signatures are computed over signature data that depends on the type of LISP message sent. See [Section 8](#) and [Section 9](#) for each message type. The signature data is passed through a sha256 hash function before it is signed or verified.

## **8. Signed Map-Register Encoding**

When a ETR registers its Crypto-EID or any EID type to the mapping system, it builds a LISP Map-Register message. The mapping includes an EID-record which encodes the Crypto-EID, or any EID type, and an RLOC-set. One of the RLOC-records in the RLOC-set includes the the ETR's signature and signature-ID. The RLOC-record is formatted with a LCAF JSON Type, in the following format:

```
{ "signature" : "<signature-base64>", "signature-id" : "<signer-id>" }
```

Where <signature-base64> is a base64 [[RFC4648](#)] encoded string over the following ascii [[RFC0020](#)] string signature data:



```
[<iid>]<crypto-eid>
```

Where <iid> is the decimal value of the instance-ID the Crypto-EID is registering to and the <crypto-eid> is in the form of [\[RFC3513\]](#) where quad-nibbles between colons ARE NOT zero-filled.

The Map-Server that process an EID-record with a Crypto-EID and a RLOC-record with a signature extracts the public-key hash value from the Crypto-EID to build a Hash-EID. The Map-Server looks up the Hash-EID in the mapping system to obtain the public-key RLOC-record. The Map-Server verifies the signature over the signature data to determine if it should accept the EID-record registration.

## 9. Signed Map-Request Encoding

When an xTR (an ITR, PITR, or RTR), sends a Map-Request to the mapping system to request the RLOC-set for a Crypto-EID, it signs the Map-Request so it can authenticate itself to the Map-Server the Crypto-EID is registered to. The Map-Request target-EID field will contain the Crypto-EID and the source-EID field will contain an LCAF JSON Type string with the following signature information:

```
{
  "source-eid" : "<seid>",
  "signature-id" : "<signer-id>",
  "signature" : "<signature-base64>"
}
```

Where <signer-id> is an IPv6 encoded string according to [\[RFC3513\]](#) where quad-nibbles between colons ARE NOT zero-filled. The <seid> is the source EID from the data packet that is invoking the Map-Request or the entire key/value pair for "source-eid" can be excluded when a data packet did not invoke the Map-Request (i.e. lig or an API request). The <signer-id> is the IPv6 Crypto-EID of the xTR that is providing the Map-Request signature.

The signature string <signature-base64> is a base64 [\[RFC4648\]](#) encoded string over the following signature data:

```
<nonce><source-eid><crypto-eid>
```

Where <nonce> is a hex string [\[RFC0020\]](#) of the nonce used in the Map-Request and the <source-eid> and <crypto-eid> are hex strings [\[RFC0020\]](#) of an IPv6 address in the form of [\[RFC3513\]](#) where quad-nibbles between colons ARE NOT zero-filled. When <seid> is not included in the Map-Request, string "0::0" is used for <source-eid>.





## **10. Signed Map-Notify Encoding**

When a Map-Server originates a Map-Notify message either as an acknowledgment to a Map-Register message, as a solicited [[I-D.ietf-lisp-pubsub](#)] notification, or an unsolicited [[RFC8378](#)] notification, the receiver of the Map-Notify can verify the message is from an authenticated Map-Server.

An RLOC-record similar to the one used to sign Map-Register messages is used to sign the Map-Notify message:

```
{ "signature" : "<signature-base64>", "signature-id" : "<signer-id>" }
```

Where the "signature-id" is an IPv6 crypto-EID used by the Map-Server to sign the RLOC-record. The signature data and the encoding format of the signature is the same as for a Map-Register message. See details in [Section 8](#).

A receiver of a Map-Notify message will lookup the signature-id in the mapping system to obtain a public-key to verify the signature. The Map-Notify is accepted only if the verification is successful.

## **11. Other Uses**

The mechanisms described within this document can be used to sign other types of LISP messages. And for further study is how to use these mechanisms to sign LISP encapsulated data packets in a compressed manner to reduce data packet header overhead.

In addition to authenticating other types of LISP messages, other types of EID-records can be encoded as well and is not limited to IPv6 EIDs. It is possible for a LISP xTR to register and request non IPv6 EIDs but use IPv6 Crypto-EIDs for the sole purpose of signing and verifying EID-records.

Examples of other EID types that can be authenticated in Map-Request and Map-Register messages are:



| EID-Type              | Format Definition                                  |
|-----------------------|--|
| IPv4 address prefixes | <a href="#">[RFC1123]</a>                          |
| Distinguished-Names   | <a href="#">[I-D.farinacci-lisp-name-encoding]</a> |
| Geo-Coordinates       | <a href="#">[I-D.farinacci-lisp-geo]</a>           |
| LCAF defined EIDs     | <a href="#">[RFC8060]</a>                          |

## 12. EID Authorization

When a Crypto-EID is being used for IPv6 communication, it is implicit that the owner has the right to use the EID since it was generated from the key-pair provisioned for the owner. For other EID types that are not directly associated with signature keys, they must be validated for use by the mapping system they are registered to. This policy information for the mapping system must be configured in the Map-Servers the EID owner registers to or a signed authorization provided by a third-party entity.

To achieve signed authorization, an entity that allows another entity to register an EID, must authorize the registering entity. It does so by adding RLOC-records to the registering entity's Hash-EID public-key mapping. The format of the RLOC-record is a JSON encoded record as follows:

```
{
  "allow-eid" : "<eid>",
  "signature-id" : "<signer-id>",
  "signature" : "<signature-base64>"
}
```

The format of the <signer-id> and <signature-base64> values are the same as described in [Section 8](#). The <eid> value is in the same string format as the signature data described in [Section 8](#). For other non-IPv6 EID types, the conventions in [\[RFC8060\]](#) are used. In all cases, the string encoding format of instance-ID '[<iid>]' prepended is to the EID string.

This entry is added to the RLOC-set of the registering entity's Hash-EID 'hash-<hash>' registration. The authorizing entity does signs the Map-Register and sends it with merge-semantics. The Map-Server accepts the registration after the signature is verified and merges the RLOC-record into the existing RLOC-set. The 'signature' is optional and when not included means the authorizing entity has not



yet allowed the registering entity to register the EID <eid>. Note multiple entities can register RLOC-records with the same <eid> meaning that signature verification for all of them is required before the Map-Server accepts the registration.

When the Map-Server receives a Map-Register for <eid>, it looks up 'hash-<hash>' EID in the mapping system. If not found, the Map-Register EID-record is not processed and the next EID-record is retrieved from the Map-Register message, if it exists. If the Hash-EID entry is found, the registering entity's signature is verified first. If the verification fails, the Map-Register EID-record is not accepted. Otherwise, a search for the RLOC-set is done to look for all matches of the EID being registered with <eid>, for those entries found, if any of them do not have a "signature" JSON item, the EID-record is not accepted. Otherwise, the signature-id is looked up in the mapping system to retrieve the public-key of the authorizing entity. If the verification is successful, then a lookup for the next RLOC-record signature-id is done. Only when all signature verifications are verified, the Map-Register EID-record is accepted.

The Map-Server should reject an RLOC-record with a signature-id that contains the Hash-EID of the entry disallowing a registering entity to self authorize itself.

Here is an example of a Hash-EID mapping stored in the mapping system:

EID-record: [1000]'hash-1111:2222:3333:4444', RLOC-Set (count is 4):

```
RLOC-record: { "public-key" : "<pubkey-base64>" }
RLOC-record: { "allow-eid" : "[1000]1.1.1.1/32", "signature" : "<sig>",
               "signature-id" : "[1000]2001:5:3::1111" }
RLOC-record: { "allow-eid" : "[1000]1.1.1.1/32", "signature" : "<sig>",
               "signature-id" : "[1000]2001:5:3::2222" }
RLOC-record: { "allow-eid" : "37-16-46-N-121-52-4-W",
               "signature-id" : "[1000]2001:5:3::5555" }
```

This mapping stores the public-key of the registering entity with Hash-EID 1111:2222:3333:4444. The registering entry registered this RLOC-record. There are two authorizing entities, :1111 and :2222, who allow it to register IPv4 EID 1.1.1.1/32. They each registered their respective RLOC-records. And a third authorizing entity :5555 that registers an RLOC-record that has not yet authorized the registering entity to register Geo-Coordinate 37-16-46-N-121-52-4-W. Note the mapping and the signature-IDs are all within the context of instance-ID 1000.



### **13. Security Considerations**

The mechanisms within this specification are intentionally using accepted practices and state of the art public-key cryptography.

Crypto-EIDs can be made private when control messages are encrypted, for instance, using [\[RFC8061\]](#).

The topological or physical location of a Crypto-EID is only available to the other Crypto-EIDs that register in the same LISP Instance-ID and have their corresponding Hash-EIDs registered.

This draft doesn't address reply attacks directly. If a man-in-the-middle captures Map-Register messages, it could send such captured packets at a later time which contains signatures of the source. In which case, the Map-Server verifies the signature as good and interprets the contents to be valid where in fact the contents can contain old mapping information. This problem can be solved by encrypting the contents of Map-Registers using a third-party protocol like DTLS [\[RFC6347\]](#) or LISP-Crypto [\[RFC8061\]](#) directly by encapsulating Map-Registers in LISP data packets (using port 4341).

Map-Reply message signatures and authentication are not in scope for this document. This document focuses on authentication between xTRs and mapping system components. Map-Reply authentication, which is performed between xTRs is described in [\[I-D.ietf-lisp-sec\]](#).

### **14. IANA Considerations**

Since there are no new packet formats introduced for the functionality in this specification, there are no specific requests for IANA.

### **15. References**

#### **15.1. Normative References**

- [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, [RFC 20](#), DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/info/rfc20>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, [RFC 1123](#), DOI 10.17487/RFC1123, October 1989, <<https://www.rfc-editor.org/info/rfc1123>>.





- [RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", [RFC 1700](#), DOI 10.17487/RFC1700, October 1994, <<https://www.rfc-editor.org/info/rfc1700>>.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", [RFC 3513](#), DOI 10.17487/RFC3513, April 2003, <<https://www.rfc-editor.org/info/rfc3513>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", [RFC 6234](#), DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", [RFC 6830](#), DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", [RFC 6833](#), DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", [RFC 7468](#), DOI 10.17487/RFC7468, April 2015, <<https://www.rfc-editor.org/info/rfc7468>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", [RFC 8060](#), DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", [RFC 8061](#), DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.



- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", [RFC 8378](#), DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

## **15.2. Informative References**

- [AFI] "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS <http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml?>, February 2007.
- [I-D.farinacci-lisp-geo] Farinacci, D., "LISP Geo-Coordinate Use-Cases", [draft-farinacci-lisp-geo-07](#) (work in progress), April 2019.
- [I-D.farinacci-lisp-name-encoding] Farinacci, D., "LISP Distinguished Name Encoding", [draft-farinacci-lisp-name-encoding-08](#) (work in progress), September 2019.
- [I-D.ietf-lisp-pubsub] Rodriguez-Natal, A., Ermagan, V., Leong, J., Maino, F., Cabellos-Aparicio, A., Barkai, S., Farinacci, D., Boucadair, M., Jacquenet, C., and S. Secci, "Publish/Subscribe Functionality for LISP", [draft-ietf-lisp-pubsub-04](#) (work in progress), September 2019.
- [I-D.ietf-lisp-rfc6833bis] Farinacci, D., Maino, F., Fuller, V., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", [draft-ietf-lisp-rfc6833bis-25](#) (work in progress), June 2019.
- [I-D.ietf-lisp-sec] Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", [draft-ietf-lisp-sec-19](#) (work in progress), July 2019.
- [X9.62] "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", NIST ANSI X9.62-2005, November 2005.

## **Appendix A. Acknowledgments**

A special thanks goes to Sameer Merchant and Colin Cantrell for their ideas and technical contributions to the ideas in this draft.



## **Appendix B. Document Change Log**

[RFC Editor: Please delete this section on publication as RFC.]

### **B.1. Changes to [draft-ietf-lisp-ecdsa-auth-02](#)**

- o Posted September 2019.
- o Update references and document timer.

### **B.2. Changes to [draft-ietf-lisp-ecdsa-auth-01](#)**

- o Posted March IETF week 2019.
- o Update references and document timer.

### **B.3. Changes to [draft-ietf-lisp-ecdsa-auth-00](#)**

- o Posted mid-September 2018.
- o Make [draft-farinacci-lisp-ecdsa-auth-03](#) a LISP working group docuemnt.

### **B.4. Changes to [draft-farinacci-lisp-ecdsa-auth-03](#)**

- o Posted September 2018.
- o Change all occurrences of signature-EID to signature-ID.
- o Document how Map-Servers sign Map-Notify messages so they can be verified by xTRs.
- o Add multi-signatures to mappings so a 3rd-party can allow an entity to register any type of EID.

### **B.5. Changes to [draft-farinacci-lisp-ecdsa-auth-02](#)**

- o Draft posted April 2018.
- o Generalize text to allow Map-Requesting and Map-Registering for any EID type with a proper signature-EID and signature encoded together.

### **B.6. Changes to [draft-farinacci-lisp-ecdsa-auth-01](#)**

- o Draft posted October 2017.



- o Make it more clear what values and format the EID hash is run over.
- o Update references to newer RFCs and Internet Drafts.

**B.7. Changes to [draft-farinacci-lisp-ecdsa-auth-00](#)**

- o Initial draft posted July 2017.

Authors' Addresses

Dino Farinacci  
lispers.net  
San Jose, CA  
USA

Email: farinacci@gmail.com

Erik Nordmark  
Zededa  
Santa Clara, CA  
USA

Email: erik@zededa.com



