Content Delivery Networks Interconnection
Internet-Draft
Intended status: Standards Track
Expires: 26 August 2026

B. Rosenblum
Vecima
G. Goldstein
Lumen Technologies

22 February 2026

CDNI Protected Secrets Metadata
draft-ietf-cdni-protected-secrets-metadata-06

Abstract

This document defines a simple mechanism for protected secret data (such as salt values or encryption keys) that may be embedded in configuration metadata or capabilities advertisements.

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 August 2026.

Copyright Notice

Table of Contents

1. Introduction

Certain objects in both the FCI and MI interfaces encapsulate sensitive values, such as credentials and access keys, which should not necessarily be accessible to all parties that can view the advertisement and configuration payloads.

This document defines two mechanisms to enclose secret values in the context of other FCI and MI objects that may only be viewed by the intended recipients, including embedded secrets encrypted using a certificate supplied by a counterparty and secrets stored in an external service (support defined in this draft is specifically for HashiCorp Vault), which are accessed via a specified path and a key ID. Refer to [HCVAULT] documentation for details.

Either side can share secrets, and the functionality is the same for both sides, so the FCI capabilities are wrappers around the MI objects, similar to how FCI footprints (used in [RFC8008]) reutilize the MI.Footprint and registry defined in [RFC8006]

The public certificate for the downstream content delivery network (dCDN) is shared via FCI.SecretCertificate and the certificate for the upstream content delivery network (uCDN) is shared via MI.SecretCertificate.

Overview of the workflow for embedded secrets:

* uCDN retrieves an advertisement with FCI.SecretStore and FCI.SecretCertificate. As the uCDN has not yet provided a certificate, any embedded secret values in the advertisement are omitted; the dCDN is not yet able to encrypt the secrets, so no secrets are present.

* uCDN provides configuration including MI.SecretStore and MI.SecretValue with values encrypted using the dCDN certificate.

* uCDN receives an updated capabilities advertisement; having provided an MI.SecretCertificate, the advertisement SHOULD now contain populated MI.SecretValue objects if secret values are utilized, encrypted with the provided certificate.

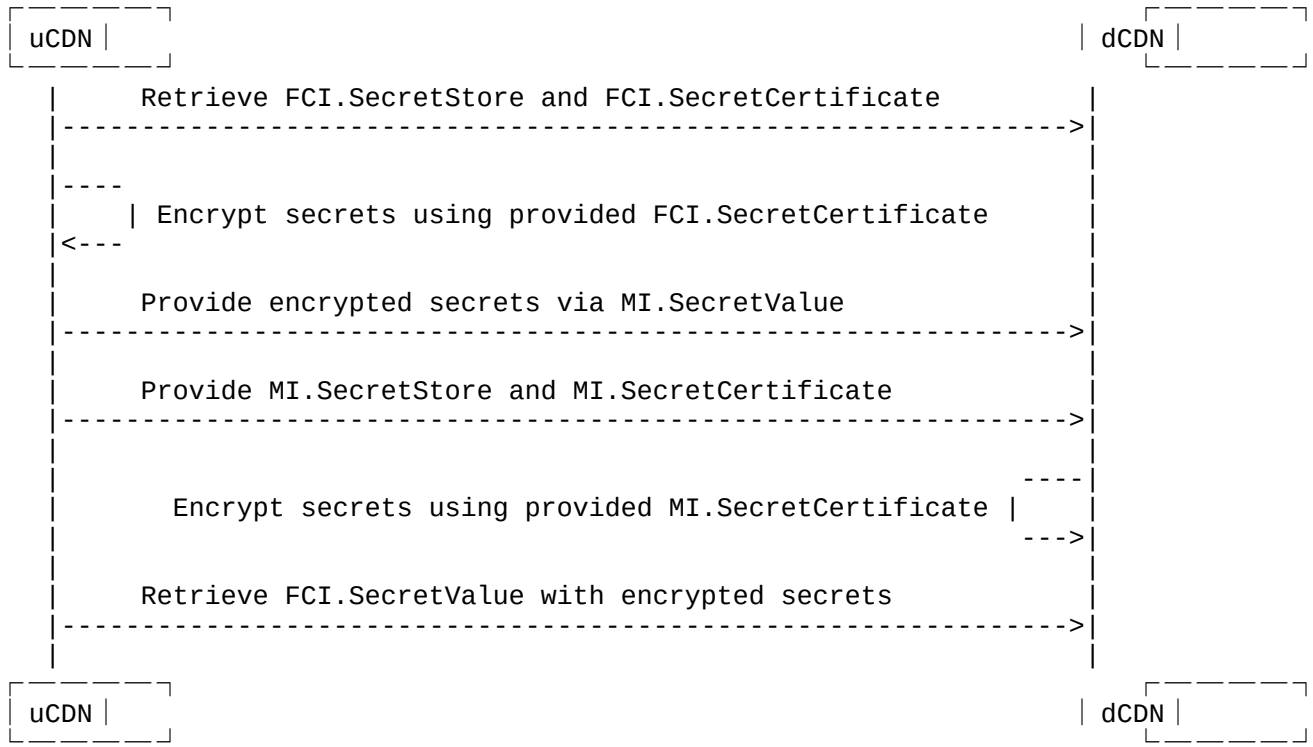

Figure 1: Bi-directional Protect Secrets Workflow, Embedded Store Type

Detailed workflow examples, including modes that reference external services or contain secret values in plaintext, are available in the Workflow Examples (Section 5) section.

The MI.SecretValue objects are utilized in the FCI and MI interfaces, where secrets must be referenced.

Certificates can be validated based on signatures in production environments, and self-signed certificates can be accepted in testing/lab environments. With this model, no out-of-band communication is required to share secrets.

2. Requirements

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.

3. Metadata Objects

3.1. MI.SecretStore

MI.SecretStore instructs the counterparty how to dereference the value of any MI.SecretValue objects linked to the store.

For embedded stores, MI.SecretStore identifies the certificate used for encrypting the values. For external stores (e.g., HashiCorp Vault), MI.SecretStore specifies the service endpoint that SHOULD be used in conjunction with the MI.SecretValue key path to obtain the secure data.

Property: secret-store-id

* Description: An identifier for this store configuration that is referenced from linked MI.SecretValue objects. This value MUST be unique within the context of the metadata objects and the FCI advertisement.

* Type: String

* Mandatory-to-Specify: Yes

Property: secret-store-type

* Description: A type discriminator for the configuration object, this property specifies whether the linked MI.SecretValue objects contain embedded secret objects or reference an external store.

* Type: String. Either "MI.SecretStoreTypeEmbedded" or "MI.SecretStoreTypeHashiCorpVault".

* Mandatory-to-Specify: Yes

Property: secret-store-config

* Description: The appropriate configuration object for the specified store type.

* Type: Object, with type specified by the secret-store-type property.

* Mandatory-to-Specify: Yes

3.2. MI.SecretStoreTypeEmbedded

MI.SecretStoreTypeEmbedded contains the configuration necessary to decrypt embedded secrets in MI.SecretValue.

The only currently supported encrypted message format is Cryptographic Message Syntax (CMS) as defined in [RFC5652] Messages MUST be CMS type "EnvelopedData". Messages MUST be encoded as Base64 per section 4 of [RFC4648] Messages MAY also be optionally formatted with encapsulation boundaries as per [RFC7468] It is strongly preferred to utilize the [RFC7468] message formatting when capable.

A cleartext format is also defined for testing purposes. In this case, the value of an MI.SecretValue object’s secret-value property is the cleartext secret. It is NOT RECOMMENDED to use the cleartext format in production environments.

Property: format

* Description: The format of the embedded encrypted message.

* Type: String. Either "cms" or "cleartext".

* Mandatory-to-Specify: Yes

Property: secret-certificate-id

* Description: The ID of the MI.SecretCertificate used to encrypt secret messages linked with this store configuration.

* Type: String

* Mandatory-to-Specify: Yes when format is "cms", otherwise No.

The following is an example of MI.SecretStoreTypeEmbedded specifying use of CMS:

```
{
  "secret-store-id": "store-1",
  "secret-store-type": "MI.SecretStoreTypeEmbedded",
  "secret-store-config": {
      "format": "cms",
      "secret-certificate-id": "cert-1"
  }
}
```

Figure 2

3.3. MI.SecretStoreTypeHashiCorpVault

MI.SecretStoreTypeHashiCorpVault contains the configuration necessary to reference secrets stored in an external instance of a HashiCorp Vault KV store [HCVAULT].

MI.SecretValue objects reference secrets stored in the Vault using the secret-path property to identify the path and property key. See the MI.SecretValue (Section 3.4) section for details.

Property: endpoint

* Description: The base URL of the Vault instance.

* Type: String

* Mandatory-to-Specify: Yes

Property: namespace

* Description: The Vault namespace in which secret lookups should be performed.

* Type: String

* Mandatory-to-Specify: Yes

Property: version

* Description: The Vault KV version.

* Type: Integer. Valid values: 1 or 2.

* Mandatory-to-Specify: Yes

The following is an example of MI.SecretStoreTypeHashiCorpVault specifying a version KV-V1 Vault:

```
{
  "secret-store-id": "store-2-vaultv1",
  "secret-store-type": "MI.SecretStoreTypeHashiCorpVault",
  "secret-store-config": {
      "endpoint": "https://vault.example.com/v1/secret",
      "version": 1,
      "namespace": "customer-1"
  }
}
```

Figure 3

The following is an example of MI.SecretStoreTypeHashiCorpVault specifying a version KV-V2 Vault:

```
{
  "secret-store-id": "store-3-vaultv2",
  "secret-store-type": "MI.SecretStoreTypeHashiCorpVault",
  "secret-store-config": {
      "endpoint": "https://vault.example.com/v1/secret",
      "version": 2,
      "namespace": "customer-1"
  }
}
```

Figure 4

3.4. MI.SecretValue

MI.SecretValue may be used in any FCI or MI object where sensitive data must be transmitted only to intended recipients. Only one of secret-value or secret-path MUST be specified, depending on the store type.

Property: secret-store-id

* Description: The ID of MI.SecretStore that contains the configuration defining how to decrypt or access the referenced secret.

* Type: String

* Mandatory-to-Specify: Yes

Property: secret-value

* Description: Used only for embedded secrets. Format is one of the following:

  - The cleartext string

  - The PEM encoded or Base64 encoded value of a CMS message. If the message begins with an [RFC7468] pre-encapsulation boundary of "---BEGIN CMS---", the message MUST be decoded according to [RFC7468] If no pre-encapsulation boundary is present, the message MUST be decoded as per section 4 of [RFC4648] Future versions of this specification will deprecate support for plain [RFC4648] encoding.

* Type: String

* Mandatory-to-Specify: Yes, if secret-path is not specified.

Property: secret-path

* Description: Used only for HashiCorp Vault secrets, the path, not including namespace, to the secret, including the key of the particular property to access as the last path parameter.

* Type: String

* Mandatory-to-Specify: Yes, if secret-value is not specified.

Property: timeout

* Description: If this property is present and the elapsed time since last retrieving the secret value has exceeded this timeout, any cached instance of this secret value MUST be discarded and fetched again from the associated secret store. This value has no effect when using an embedded store.

* Type: Integer duration in seconds.

* Mandatory-to-Specify: No

The following is an example of MI.SecretValue specifying an embedded value:

```
{
  "secret-store-id": "store-1-cms",
  "secret-value": "---BEGIN CMS—
MIIBiQYJKoZIhvcNAQcDoIIBejCCAXYCAQAxggEhMIIBHQIBAD
AFMAACAQEwDQYJKoZIhvcNAQEBBQAEggEApJeXzsUS1jbAyNtQiJ9um9IMIHW5B2g+gHn
XdNSTyd33OEfTR6yLSZihBlFbHpY3qSzK1CX7RF5Oz3SqLDW+r3i1D/aHbVXwQbviWHEv
Hterql8l9VDm2FCNaDx5vihdbtvng3+/vdJNNMMhmovwZL5uhPsK81DkKwZCvznMMWt8Y
dNSFGT62f73ash7Eg/mS54IUyYOJHYrXEkRLSjvl0j+JqcIR8hCOCA78+5bS4MgfdsS9x
xSwQTrPru6EdTivMDKE/jlKg7li8lWdirWqtv0za5gLmH5T+zslXIoklwERAE50Jj8FxZ
D98EikKH8DAa+JeFsBm6Z1+yVFsWucTBMBgkqhkiG9w0BBwEwHQYJYIZIAWUDBAEqBBBw
s1riXA6m336zRbsiKtrVgCA267133v2zD/wjFQHXrKSJfd/2YJaxPskgdmQaVlgWCw==
--END CMS--"
}
```

Figure 5

The following is an example of MI.SecretValue for a Vault:

```
{
  "secret-store-id": "store-2-vaultv1",
  "secret-path": "bar/baz/importantsecret",
  "timeout": 3600,
}
```

Figure 6

3.5. MI.SecretCertificate

MI.SecretCertificate is used to share an [X.509] certificate to be utilized for encrypting embedded secret messages.

In lab and testing environments, this certificate MAY be self-signed, depending on participant agreement.

In production environments, this SHOULD be a certificate signed by an appropriate certificate authority (CA) and validated by the counterparty.

Property: certificate-id

* Description: A unique ID for this certificate that can be referenced from a corresponding MI.SecretStore configuration defined by a counterparty.

* Type: String

* Mandatory-to-Specify: Yes

Property: certificate-value

* Description: The PEM encoded OR Base64 encoded certificate.If the value begins with an [RFC7468] pre-encapsulation boundary of "---BEGIN CERTIFICATE---", the message MUST be decoded according to [RFC7468] If no pre-encapsulation boundary is present, the message MUST be decoded as per section 4 of [RFC4648] Future versions of this specification will deprecate support for plain [RFC4648] encoding.

* Type: String

* Mandatory-to-Specify: Yes

The following is an example of MI.SecretCertificate:

```
{
  "certificate-id": "store-1",
  "certificate-value": "---BEGIN CERTIFICATE---
MIIDZTCCAk2gAwIBAgIUFJokJzAxDgUGsBd8uhSblpMwS
LAwDQYJKoZIhvcNAQELBQAwQjELMAkGA1UEBhMCVVMxEDAOBgNVBAgMB0dlb3JnaWExIT
AfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDAeFw0yMzAxMjMyMDM2MDNaFw0
yMzAyMjIyMDM2MDNaMEIxCzAJBgNVBAYTAlVTMRAwDgYDVQQIDAdHZW9yZ2lhMSEwHwYD
VQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggEiMA0GCSqGSIb3DQEBAQUAA4IBD
wAwggEKAoIBAQCT11o9yebJmjiq7mXbLtnr5THpTnyahNpKECI+N8YZSl5+cS9hGa06zK
QV3MNxbjJ15smmeWbgynYGwqhs5ZXGUjzd8S1/M1A08z1VFhEJiODQ00f3BOocpIn25RQ
zFz/BOLREW7sLkrhuz/WVBR3bzp6T1gu3nKcRSNuNxO1p9490gS1LhsZYQKfNvncuxBCP
0GTNbUOXd6xkQ+EX5cEKoODUYWzOMdMAMlEEFb4jUjxYbbJoygwTMHpG2yGAQ2IXpB2/w
rrawivxDHlMHGpML+Ie8o6YBR4PDiOJmlCg9uIsirf65R1zhfcCxmNQ7z/IggC0WNQjZw
ymeZT9cFDdAgMBAAGjUzBRMB0GA1UdDgQWBBQb5eJeYLEpErJetb1eid5BgsS3uTAfBgN
VHSMEGDAWgBQb5eJeYLEpErJetb1eid5BgsS3uTAPBgNVHRMBAf8EBTADAQH/MA0GCSqG
SIb3DQEBCwUAA4IBAQBURnrjVbHVwfV/u/xjzK8p4dTke0xb0oKt0J5YeH95sRa66m3tQ
JYf0jbMNQ8InfXK0IzGM/uUOJX3daeOMQxMbJvaUDZV64kuU6IgkEQuLwkOP5k0Rc9+Su
RMlvWOB2exiyQkd2iHJtURuEtvB39LIr4pPDsicBAYxsm5ybIWCmqNMPkVl8Qks3lAXeF
+xvH11tmciTJSYP0Ud2psbV3lduD76UT2bzDGkr690KqroNS57WbQrHxEhtMbdq0cPfzq
FlxyhckqNYrcw2v1igQDhplQ2eUc4ye0Mvimj1Me2mWjPvilhvS3vDGhrmcx9mlishlI/
RFy6yDI1gtkF7eS
---END CERTIFICATE---"
}
```

Figure 7

4. Capabilities Objects

These objects are simple capability wrappers around the MI objects defined in this specification.

4.1. FCI.SecretStore

FCI.SecretStore instructs the uCDN how to dereference the value of any MI.SecretValue objects linked to the store from other FCI objects via an embedded MI.SecretValue object. For further details, see the MI.SecretStore (Section 3.1) section.

The following is an example of FCI.SecretStore:

```
{
  "capabilities": [
    {
      "capability-type": "FCI.SecretStore",
      "capability-value": {
        "secret-store-id": "store-1",
        "secret-store-type": "MI.SecretStoreTypeEmbedded",
        "secret-store-config": {
           "format": "cms"
        }
      }
    }
  ]
}
```

Figure 8

4.2. FCI.SecretCertificate

FCI.SecretCertificate is used to share an [X.509] certificate to be utilized for encrypting embedded secret messages via an embedded MI.SecretCertificate object. For further details, see the MI.SecretCertificate (Section 3.5) section.

The following is an example of FCI.SecretCertificate:

```
{
  "capabilities": [
    {
      "capability-type": "FCI.SecretCertificate",
      "capability-value": {
        "certificate-id": "store-1",
        "certificate-value": "---BEGIN CERIFICATE--
MIIDZTCCAk2gAwIBAgIUFJokJzAxDgUGsBd8uhS
blpMwSLAwDQYJKoZIhvcNAQELBQAwQjELMAkGA1UEBhMCVVMxEDAOBgNVBAgMB0dlb3Jn
aWExITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDAeFw0yMzAxMjMyMDM2M
DNaFw0yMzAyMjIyMDM2MDNaMEIxCzAJBgNVBAYTAlVTMRAwDgYDVQQIDAdHZW9yZ2lhMS
EwHwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggEiMA0GCSqGSIb3DQEBAQU
AA4IBDwAwggEKAoIBAQCT11o9yebJmjiq7mXbLtnr5THpTnyahNpKECI+N8YZSl5+cS9h
Ga06zKQV3MNxbjJ15smmeWbgynYGwqhs5ZXGUjzd8S1/M1A08z1VFhEJiODQ00f3BOocp
In25RQzFz/BOLREW7sLkrhuz/WVBR3bzp6T1gu3nKcRSNuNxO1p9490gS1LhsZYQKfNvn
cuxBCP0GTNbUOXd6xkQ+EX5cEKoODUYWzOMdMAMlEEFb4jUjxYbbJoygwTMHpG2yGAQ2I
XpB2/wrrawivxDHlMHGpML+Ie8o6YBR4PDiOJmlCg9uIsirf65R1zhfcCxmNQ7z/IggC0
WNQjZwymeZT9cFDdAgMBAAGjUzBRMB0GA1UdDgQWBBQb5eJeYLEpErJetb1eid5BgsS3u
TAfBgNVHSMEGDAWgBQb5eJeYLEpErJetb1eid5BgsS3uTAPBgNVHRMBAf8EBTADAQH/MA
0GCSqGSIb3DQEBCwUAA4IBAQBURnrjVbHVwfV/u/xjzK8p4dTke0xb0oKt0J5YeH95sRa
66m3tQJYf0jbMNQ8InfXK0IzGM/uUOJX3daeOMQxMbJvaUDZV64kuU6IgkEQuLwkOP5k0
Rc9+SuRMlvWOB2exiyQkd2iHJtURuEtvB39LIr4pPDsicBAYxsm5ybIWCmqNMPkVl8Qks
3lAXeF+xvH11tmciTJSYP0Ud2psbV3lduD76UT2bzDGkr690KqroNS57WbQrHxEhtMbdq
0cPfzqFlxyhckqNYrcw2v1igQDhplQ2eUc4ye0Mvimj1Me2mWjPvilhvS3vDGhrmcx9ml
ishlI/RFy6yDI1gtkF7eS
---END CERTIFICATE---"
      }
    }
  ]
}
```

Figure 9

5. Workflow Examples

The facilities in this document can be used for simple and bidirectional exchange of secret values between uCDN and dCDN participants in an Open Caching system. The embedded model provides for a secret exchange without reference to out-of-band services, while the Vault support allows for external reference to secrets stored in a HashiCorp Vault.

Participants utilizing a secret distribution method or service not supported here MAY define a Private Feature MI object [SVTA2038] with the necessary configuration for that method or service and then utilize that MI object within MI.SecretStore and FCI.SecretStore

Provided below are workflow examples for uCDN -> dCDN and dCDN -> uCDN exchanges of secret values.

Consideration is needed when addressing key rollover, expiration, and revocation in the embedded model. The RECOMMENDED workflow for key rollover is as follows:

* When the recipient of a secret provides an updated configuration that no longer contains an MI.SecretCertificate with an ID referenced in an MI.SecretStore used by MI.SecretValue objects, those MI.SecretValue objects SHOULD be reduced to an object with no contained secret-value or secret-path property as they would be in the initial state before any certificate had been provided.

* If the recipient of a secret then provides a new MI.SecretCertificate object, the sender of the secret SHOULD update its MI.SecretStore to reference the new certificate-id and then update any referencing MI.SecretValue objects to include an updated secret-value property that contains the newly encrypted values.

5.1. Workflow: dCDN -> uCDN Embedded

1. The dCDN advertises FCI.SecretStore with a store-type of MI.SecretStoreTypeEmbedded; other FCI objects may contain MI.SecretValue objects that reference the store-id. MI.SecretValue objects do not presently contain a secret-value property.

2. The uCDN pushes the MI configuration with an MI.SecretCertificate.

3. The dCDN updates the advertised FCI.SecretStore with a certificate-id property that references the dCDN MI.SecretCertificate; any MI.SecretValue objects in other FCI objects now contain a secret-value property with the CMS encrypted secret.

Figure 10: Figure 2: dCDN -> uCDN Workflow, Embedded Store Type

5.2. Workflow: uCDN -> dCDN Embedded

1. The dCDN advertises an FCI.SecretCertificate.

2. The uCDN pushes the MI configuration containing MI.SecretStore with a store-type of MI.SecretStoreTypeEmbedded and a certificate-id referencing the FCI.SecretCertificate advertised by the uCDN. Other MI objects may contain MI.SecretValue objects with a secret-value property containing the CMS encrypted secret.

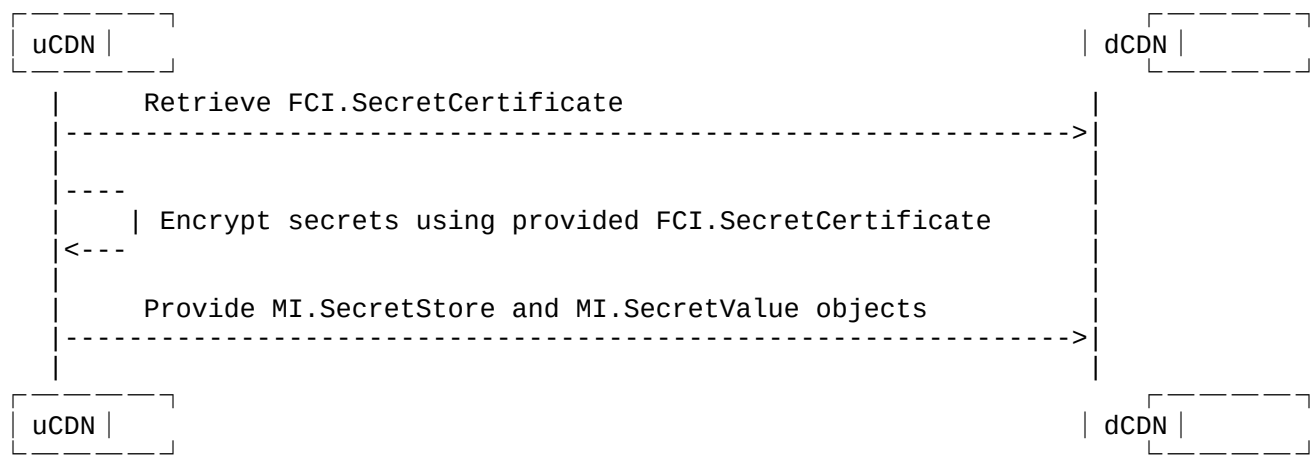

Figure 11: Figure 3: Figure 3: uCDN -> dCDN Workflow, Embedded Store

5.3. Workflow: Embedded Cleartext (uCDN and dCDN)

1. An MI.SecretStoreTypeEmbedded has a defined format of "cleartext".

2. Any MI.SecretValue objects that reference the cleartext store contain a secret-value property with the unencrypted secret.

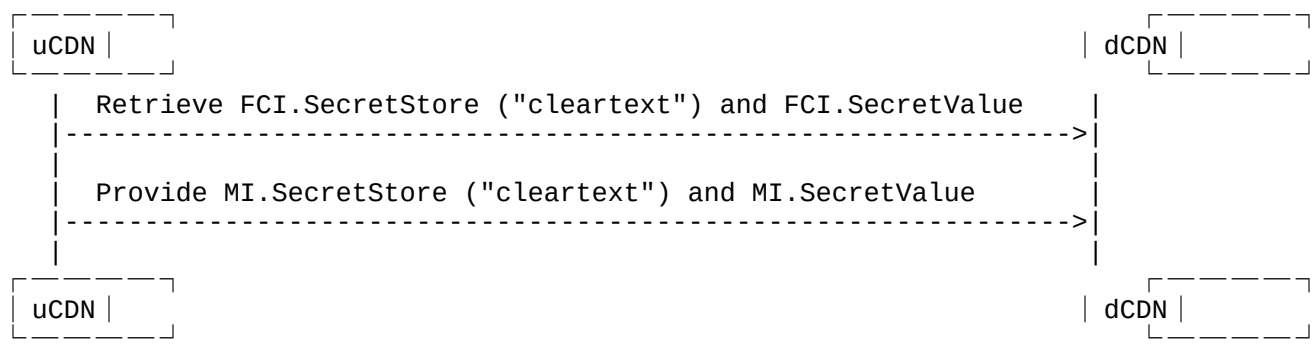

Figure 12: Figure 4: Embedded Cleartext Workflow

5.4. Workflow: dCDN -> uCDN Vault

The dCDN advertises an FCI.SecretStore with an appropriate configuration for accessing an instance of a HashiCorp Vault accessible to the uCDN. Other FCI objects may contain MI.SecretValue objects that reference the FCI.SecretStore and a secret-path property specifying the secret to retrieve.

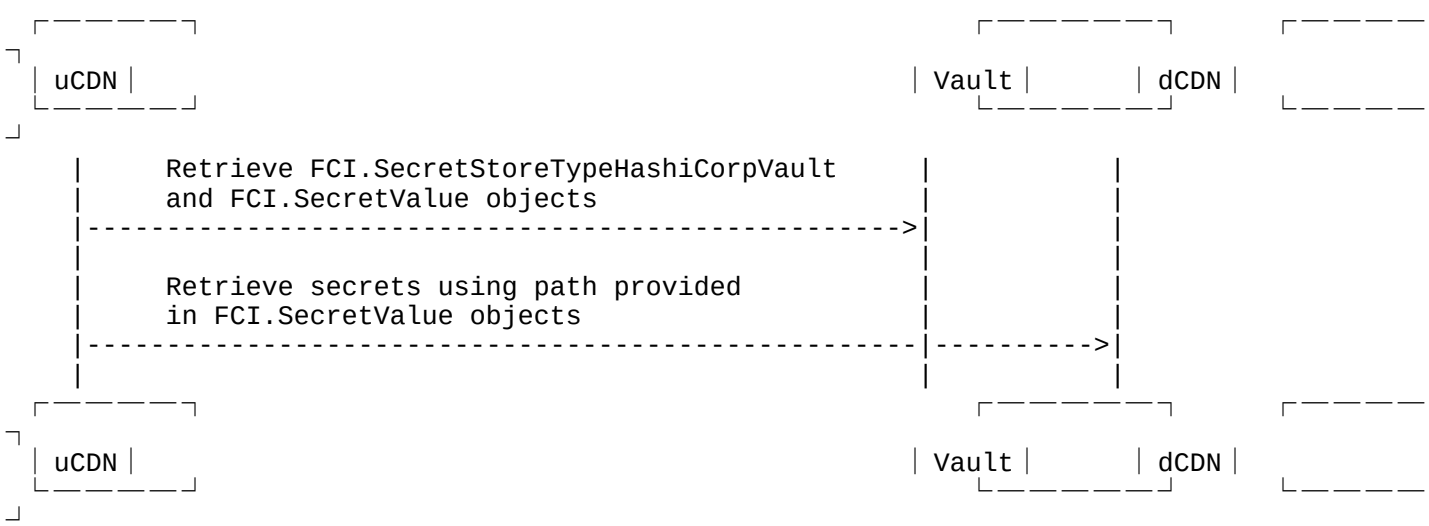

Figure 13: Figure 5: dCDN -> uCDN Workflow, HashiCorp Vault Store

5.5. Workflow: uCDN -> dCDN Vault

The uCDN provides the MI configuration, including an MI.SecretStore with appropriate configuration for accessing an instance of a HashiCorp Vault accessible to the dCDN. Other MI objects may contain MI.SecretValue objects that reference the MI.SecretStore and a secret-path property specifying the secret to retrieve.

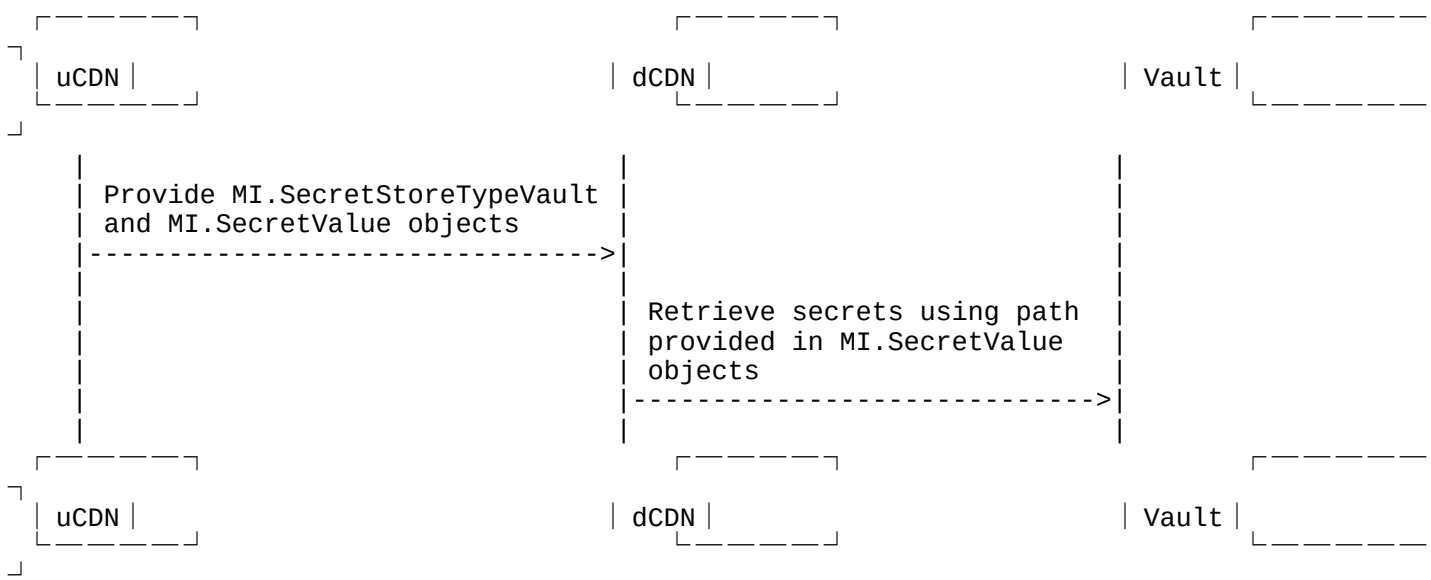

Figure 14: Figure 6: uCDN -> dCDN Workflow, HashiCorp Vault Store

6. Security Considerations

The FCI and MI objects defined in this document are transferred via the interfaces defined in CDNI [RFC8006] which describes how to secure these interfaces by protecting integrity and confidentiality while ensuring the authenticity of the dCDN and uCDN.

7. IANA Considerations

7.1. CDNI Payload Types

This document requests the registration of the following entries under the "CDNI Payload Types" registry hosted by IANA:

Payload Type	Specification
MI.SecretStore	RFCthis
MI.SecretStoreTypeEmbedded	RFCthis
MI.SecretStoreTypeVault	RFCthis
MI.SecretValue	RFCthis
MI.SecretCertificate	RFCthis
FCI.SecretStore	RFCthis
FCI.SecretCertificate	RFCthis

Table 1: CDNI Payload Types

8. Acknowledgements

The authors would like to express their gratitude to the members of the Streaming Video Technology Alliance [SVTA] Open Caching Working Group for their contributions and guidance.

The following people contributed to the content of this draft:

* Chris Lemmons (Comcast)

* Rajeev RK (picoNETS)

* Yoav Gressel (Qwilt)

* Arnon Warshavsky (Qwilt)

* Alfonso Siloniz (Telefonica)

9. Normative References

[HCVAULT] HashiCorp, "Vault Documentation", <https://developer.hashicorp.com/vault/docs>.

[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>.

[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>.

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

[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>.

[RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <https://www.rfc-editor.org/info/rfc8006>.

[RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <https://www.rfc-editor.org/info/rfc8008>.

[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>.

[X.509] ITU, "X.509", February 2011, <http://www.itu.int/rec/T-REC-X.509>.

10. Informative References

[SVTA] SVTA, "Streaming Video Technology Alliance Home Page", <https://www.svta.org>.

[SVTA2038] SVTA, "Private Features Metadata", <https://svta.org/documents/SVTA2038>.

Authors' Addresses

Ben Rosenblum
Vecima
United States of America
Email: ben@rosenblum.dev

Glenn Goldstein
Lumen Technologies
United States of America
Email: glenng1215@gmail.com