CDNI
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2016

K. Leung
F. Le Faucheur
Cisco Systems
R. van Brandenburg
TNO
B. Downey
Verizon Labs
M. Fisher
Limelight Networks
June 28, 2016

URI Signing for CDN Interconnection (CDNI)
draft-ietf-cdni-uri-signing-09

Abstract

This document describes how the concept of URI signing supports the content access control requirements of CDNI and proposes a URI signing scheme.

The proposed URI signing method specifies the information needed to be included in the URI and the algorithm used to authorize and to validate access requests for the content referenced by the URI. The mechanism described can be used both in CDNI and single CDN scenarios.

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 30, 2016.

Table of Contents

1. Introduction

This document describes the concept of URI Signing and how it can be used to provide access authorization in the case of redirection between interconnected CDNs (CDNI) and between a Content Service Provider (CSP) and a CDN. The primary goal of URI Signing is to make sure that only authorized User Agents (UAs) are able to access the content, with a CSP being able to authorize every individual request. It should be noted that URI Signing is not a content protection scheme; if a CSP wants to protect the content itself, other mechanisms, such as DRM, are more appropriate. In addition to access control, URI Signing also has benefits in reducing the impact of denial-of-service attacks.

The overall problem space for CDN Interconnection (CDNI) is described in CDNI Problem Statement [RFC6707]. In this document, along with the CDNI Requirements [RFC7337] document and the CDNI Framework [RFC7336] the need for interconnected CDNs to be able to implement an access control mechanism that enforces the CSP's distribution policy is described.

Specifically, CDNI Framework [RFC7336] states:

"The CSP may also trust the CDN operator to perform actions such as ..., and to enforce per-request authorization performed by the CSP using techniques such as URI signing."

In particular, the following requirement is listed in CDNI Requirements [RFC7337]:

"MI-16 [HIGH] The CDNI Metadata Distribution interface shall allow signaling of authorization checks and validation that are to be performed by the surrogate before delivery. For example, this could potentially include:

* need to validate URI signed information (e.g., Expiry time, Client IP address)."

This document proposes a URI Signing scheme that allows Surrogates in interconnected CDNs to enforce a per-request authorization performed by the CSP. Splitting the role of performing per-request authorization by CSP and the role of validation of this authorization by the CDN allows any arbitrary distribution policy to be enforced across CDNs without the need of CDNs to have any awareness of the actual CSP distribution policy.

1.1. Terminology

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].

This document uses the terminology defined in CDNI Problem Statement [RFC6707].

This document also uses the terminology of Keyed-Hashing for Message Authentication (HMAC) [RFC2104].

In addition, the following terms are used throughout this document:

- URI Signature: Message digest or digital signature that is computed with an algorithm for protecting the URI.
- Full Original URI: The URI before URI Signing is applied.
- Signed URI: Any URI that contains a URI Signature.
- Target CDN URI: Embedded URI created by the CSP to direct UA towards the Upstream CDN. The Target CDN URI can be signed by the CSP and verified by the Upstream CDN.
- Redirection URI: URI created by the Upstream CDN to redirect UA towards the Downstream CDN. The Redirection URI can be signed by the Upstream CDN and verified by the Downstream CDN. In a

cascaded CDNI scenario, there can be more than one Redirection URI.

1.2. Background and overview on URI Signing

A CSP and CDN are assumed to have a trust relationship that enables the CSP to authorize access to a content item by including a set of attributes in the URI before redirecting a UA to the CDN. Using these attributes, it is possible for a CDN to check an incoming content request to see whether it was authorized by the CSP (e.g., based on the UA's IP address or a time window). Of course, the attributes need to be added to the URI in a way that prevents a UA from changing the attributes, thereby leaving the CDN to think that the request was authorized by the CSP when in fact it wasn't. For this reason, a URI Signing mechanism includes in the URI a message digest or digital signature that allows a CDN to check the authenticity of the URI. The message digest or digital signature can be calculated based on a shared secret between the CSP and CDN or using CSP's asymmetric public/private key pair, respectively.

Figure 1, shown below, presents an overview of the URI Signing mechanism in the case of a CSP with a single CDN. When the UA browses for content on CSP's website (#1), it receives HTML web pages with embedded content URIs. Upon requesting these URIs, the CSP redirects to a CDN, creating a Target CDN URI (#2) (alternatively, the Target CDN URI itself is embedded in the HTML). The Target CDN URI is the Signed URI which may include the IP address of the UA and/or a time window and always contains the URI Signature which is generated by the CSP using the shared secret or a private key. Once the UA receives the response with the embedded URI, it sends a new HTTP request using the embedded URI to the CDN (#3). Upon receiving the request, the CDN checks to see if the Signed URI is authentic by verifying the URI signature. If applicable, it checks whether the IP address of the HTTP request matches that in the Signed URI and if the time window is still valid. After these values are confirmed to be valid, the CDN delivers the content (#4).

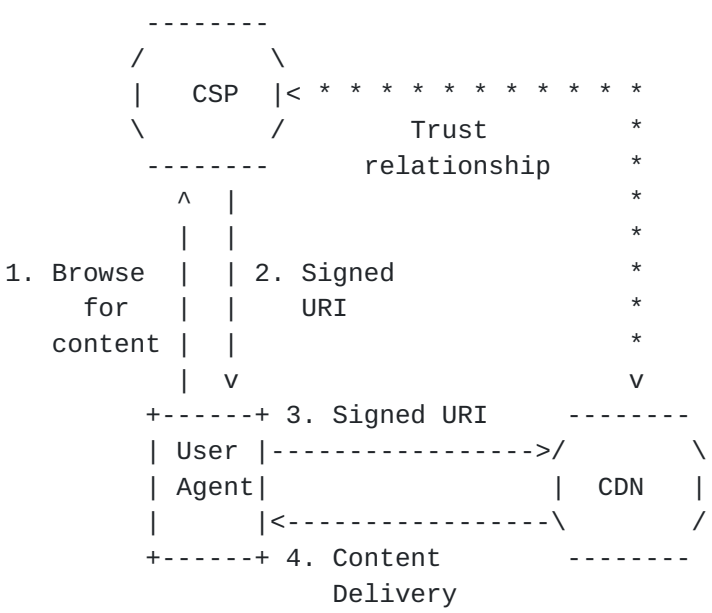
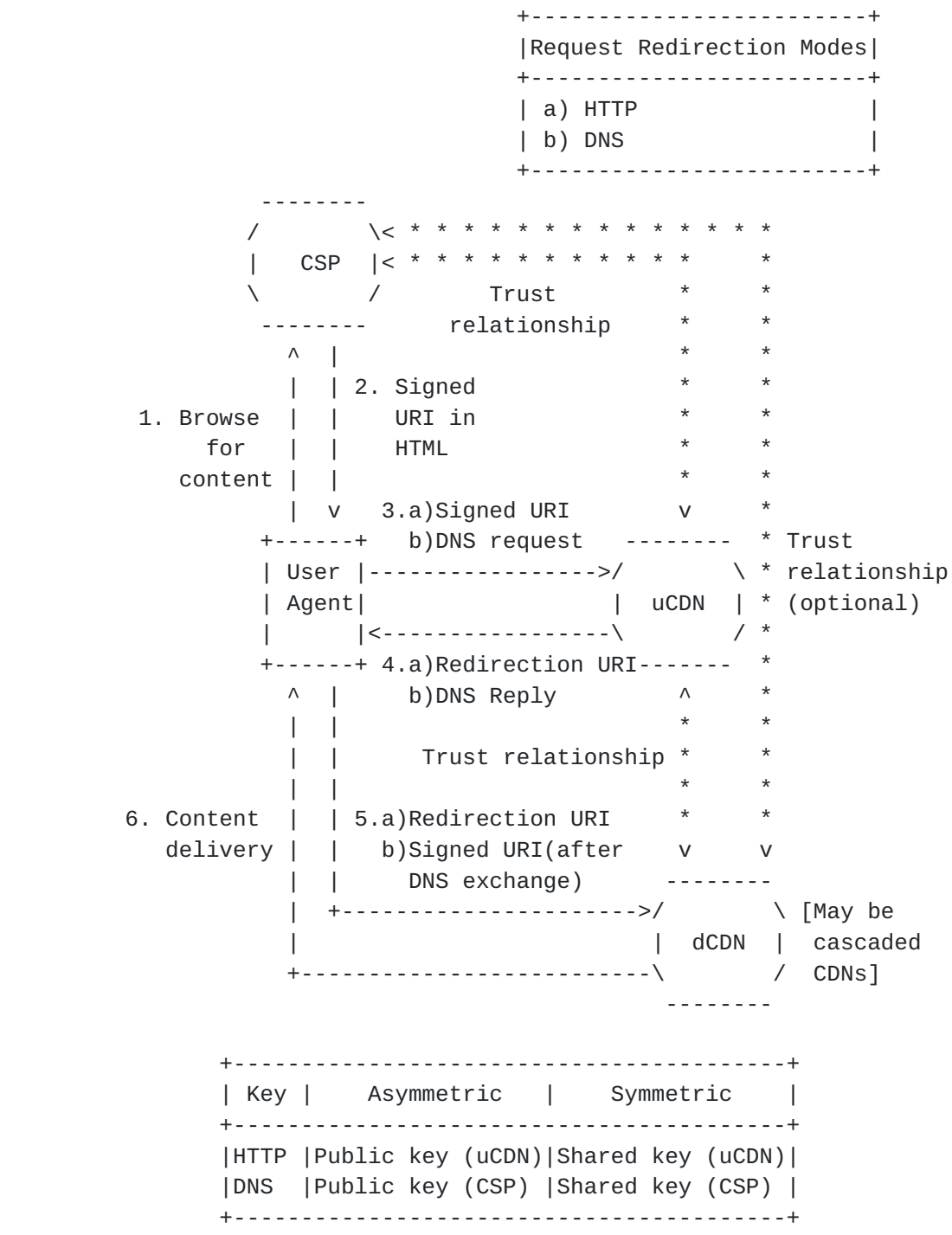


Figure 1: Figure 1: URI Signing in a CDN Environment

1.3. CDNI URI Signing Overview

In a CDNI environment, URI Signing operates the same way in the initial steps #1 and #2 but the later steps involve multiple CDNs in the process of delivering the content. The main difference from the single CDN case is a redirection step between the Upstream CDN and the Downstream CDN. In step #3, UA may send HTTP request or DNS request. Depending on whether HTTP-based or DNS-based request routing is used, the Upstream CDN responds by directing the UA towards the Downstream CDN using either a Redirection URI (which is a Signed URI generated by the Upstream CDN) or a DNS reply, respectively (#4). Once the UA receives the response, it sends the Redirection URI/Target CDN URI to the Downstream CDN (#5). The received URI is validated by the Downstream CDN before delivering the content (#6). This is depicted in the figure below. Note: The CDNI call flows are covered in Detailed URI Signing Operation (Section 6).



Key	Asymmetric	Symmetric
HTTP	Public key (uCDN)	Shared key (uCDN)
DNS	Public key (CSP)	Shared key (CSP)

Figure 2: URI Signing in a CDNI Environment

The trust relationships between CSP, Upstream CDN, and Downstream CDN have direct implications for URI Signing. In the case shown in Figure 2, the CDN that the CSP has a trust relationship with is the Upstream CDN. The delivery of the content may be delegated to the

Downstream CDN, which has a relationship with the Upstream CDN but may have no relationship with the CSP.

In CDNI, there are two methods for request routing: DNS-based and HTTP-based. For DNS-based request routing, the Signed URI (i.e., Target CDN URI) provided by the CSP reaches the Downstream CDN directly. In the case where the Downstream CDN does not have a trust relationship with the CSP, this means that only an asymmetric public/private key method can be used for computing the URI Signature because the CSP and Downstream CDN are not able to exchange symmetric shared secret keys. Since the CSP is unlikely to have relationships with all the Downstream CDNs that are delegated to by the Upstream CDN, the CSP may choose to allow the Authoritative CDN to redistribute the shared key to a subset of their Downstream CDNs .

For HTTP-based request routing, the Signed URI (i.e., Target CDN URI) provided by the CSP reaches the Upstream CDN. After this URI has been verified to be correct by the Upstream CDN, the Upstream CDN creates and signs a new Redirection URI to redirect the UA to the Downstream CDN. Since this new URI also has a new URI Signature, this new signature can be based around the trust relationship between the Upstream CDN and Downstream CDN, and the relationship between the Downstream CDN and CSP is not relevant. Given the fact that such a relationship between Upstream CDN and Downstream CDN always exists, both asymmetric public/private keys and symmetric shared secret keys can be used for URI Signing. Note that the signed Redirection URI MUST maintain the same, or higher, level of security as the original Signed URI.

1.4. URI Signing in a non-CDNI context

While the URI signing scheme defined in this document was primarily created for the purpose of allowing URI Signing in CDNI scenarios, e.g., between a uCDN and a dCDN or between a CSP and a dCDN, there is nothing in the defined URI Signing scheme that precludes it from being used in a non-CDNI context. As such, the described mechanism could be used in a single-CDN scenario such as shown in Figure 1 in Section 1.2, for example to allow a CSP that uses different CDNs to only have to implement a single URI Signing mechanism.

2. Signed URI Information Elements

The concept behind URI Signing is based on embedding in the Target CDN URI/Redirection URI a number of information elements that can be validated to ensure the UA has legitimate access to the content. These information elements are appended, in an encapsulated form, to the original URI.

For the purposes of the URI signing mechanism described in this document, three types of information elements may be embedded in the URI:

- Enforcement Information Elements: Information Elements that are used to enforce a distribution policy defined by the CSP. Examples of enforcement attributes are IP address of the UA and time window.

- Signature Computation Information Elements: Information Elements that are used by the CDN to verify the URI signature embedded in the received URI. In order to verify a URI Signature, the CDN requires some information elements that describe how the URI Signature was generated. Examples of Signature Computation Elements include the used HMACs hash function and/or the key identifier.

- URI Signature Information Elements: The information elements that carry the actual message digest or digital signature representing the URI signature used for checking the integrity and authenticity of the URI. A typical Signed URI will only contain one embedded URI Signature Information Element.

In addition, the this document specifies the following URI attribute:

- URI Signing Package Attribute: The URI attribute that encapsulates all the URI Signing information elements in an encoded format. Only this attribute is exposed in the Signed URI as a URI query parameter or as URL path parameter.

Two types of keys can be used for URI Signing: asymmetric keys and symmetric keys. Asymmetric keys are based on a public/private key pair mechanism and always contain a private key only known to the entity signing the URI (either CSP or uCDN) and a public key for the verification of the Signed URI. With symmetric keys, the same key is used by both the signing entity for signing the URI as well as by the validating entity for validating the Signed URI. Regardless of the type of keys used, the validating entity has to obtain the key (either the public or the symmetric key). There are very different requirements for key distribution (out of scope of this document) with asymmetric keys and with symmetric keys. Key distribution for symmetric keys requires confidentiality to prevent another party from getting access to the key, since it could then generate valid Signed URIs for unauthorized requests. Key distribution for asymmetric keys does not require confidentiality since public keys can typically be distributed openly (because they cannot be used for URI signing) and private keys are kept by the URI signing function.

Note that all the URI Signing information elements and the URI query attribute are mandatory to implement, but not mandatory to use.

2.1. Enforcement Information Elements

This section identifies the set of information elements that may be needed to enforce the CSP distribution policy. New information elements may be introduced in the future to extend the capabilities of the distribution policy.

In order to provide flexibility in distribution policies to be enforced, the exact subset of information elements used in the URI Signature of a given request is a deployment decision. The defined keyword for each information element is specified in parenthesis below.

The following information elements are used to enforce the distribution policy:

- Expiry Time (ET) [optional] - Time when the Signed URI expires. This is represented as an integer denoting the number of seconds since midnight 1/1/1970 UTC (i.e., UNIX epoch). The request is rejected if the received time is later than this timestamp. Note: The time, including time zone, on the entities that generate and validate the signed URI need to be in sync. In the CDNI case, this means that servers at both the CSP, uCDN and dCDN need to be time-synchronized. It is RECOMMENDED to use NTP for this.

- Client IP (CIP) [optional] - IP address, or IP prefix, for which the Signed URI is valid. This is represented in CIDR notation, with dotted decimal format for IPv4 or canonical text representation for IPv6 addresses [RFC5952]. The request is rejected if sourced from a client outside of the specified IP range.

- Original URI Container (OUC) [optional] - Container for holding the Full Original URI while the URI signature is calculated. The Original URI Container information element is not transmitted as part of the URI Signing Package Attribute. If the Original URI Container information element is used, the URI Pattern Sequence information element MUST NOT be used.

- URI Pattern Container (UPC) [optional] - Percent-encoded container for one or more URI Patterns that describes for which content the Signed URI is valid. The URI Pattern Container contains an expression to match against the requested URI to check whether the requested content is allowed to be requested. Multiple URI Patterns may be concatenated in a single URI Pattern Container

information element by separating them with a semi-colon (';') character. Each URI Pattern follows the [RFC3986] URI format, including the '://' that delimits the URI scheme from the hierarchy part. The pattern may include the wildcards '*' and '?', where '*' matches any sequence of characters (including the empty string) and '?' matches exactly one character. The three literals '$', '*' and '?' should be escaped as '$$', '$*' and '$?'. All other characters are treated as literals. The following is an example of a valid URI Pattern: '*://*/folder/content-83112371/quality_*/segment????.mp4'. In its final percent-encoded form, this is equal to '%2A%3A%2F%2F%2A%2Ffolder%2Fcontent-83112371%2Fquality_%2A%2Fsegment%3F%3F%3F%3F.mp4'. An example of two concatenated URI Patterns is the following: 'http://*/folder/content-83112371/manifest/*.xml;http://*/folder/content-83112371/quality_*/segment????.mp4', which in percent-encoded form is: 'http%3A%2F%2F%2A%2Ffolder%2Fcontent-83112371%2Fmanifest%2F%2A.xml%3Bhttp%3A%2F%2F%2A%2Ffolder%2Fcontent-83112371%2Fquality_%2A%2Fsegment%3F%3F%3F%3F.mp4' If the UPC is used, the Original URI Container information element MUST NOT be used.

The Expiry Time Information Element ensures that the content authorization expires after a predetermined time. This limits the time window for content access and prevents replay of the request beyond the authorized time window.

The Client IP Information Element is used to restrict content access to a particular IP address or set of IP addresses based on the IP address for whom the content access was authorized. The URI Signing mechanism described in this document will communicate the IP address in the URI. To prevent the IP address from being logged, the Client IP information element is transmitted in encrypted form.

The Original URI Container is used to limit access to the Original URI only.

The URI Pattern Container Information Element is used to restrict content access to a particular set of URIs.

In order to increase performance of string parsing of the UPC, implementations can check often-used UPC prefixes to quickly check whether certain URI components can be ignored. For example, UPC prefixes '*://*/' or '*://*:*' will be used in case the scheme and authority components of the URI are ignored for purposes of UPC enforcement.

Note: See the Security Considerations (Section 9) section on the limitations of using an expiration time and client IP address for distribution policy enforcement.

2.2. Signature Computation Information Elements

This section identifies the set of information elements that may be needed to verify the URI (signature). New information elements may be introduced in the future if new URI signing algorithms are developed.

The defined keyword for each information element is specified in parenthesis below.

The following information elements are used to validate the URI by recreating the URI Signature.

- Version (VER) [optional] - An 8-bit unsigned integer used for identifying the version of URI signing method. If this Information Element is not present in the URI Signing Package Attribute, the default version is 1.

- Key ID (KID) [optional] - A string used for obtaining the key (e.g., database lookup, URI reference) which is needed to validate the URI signature. The KID and KID_NUM information elements MUST NOT be present in the same URI Signing Package Attribute.

- Numerical Key ID (KID_NUM) [optional] - A 64-bit unsigned integer used as an optional alternative for KID. The KID and KID_NUM information elements MUST NOT be present in the same URI Signing Package Attribute.

- Hash Function (HF) [optional] - A string used for identifying the hash function to compute the URI signature with HMAC. If this Information Element is not present in the URI Signing Package Attribute, the default hash function is "SHA-256". For interoperability purposes, any hash function signalled via this Information Element SHALL use the notation as used by NIST (e.g. "SHA-256" instead of "SHA256", as defined in [FIPS.180-1.1995]).

- Digital Signature Algorithm (DSA) [optional] - Algorithm used to calculate the Digital Signature. If this Information Element is not present in the URI Signing Package Attribute, the default is "ECDSA". For interoperability purposes, any digital signature algorithm signalled via this Information Element SHALL use the notation as used by NIST (e.g. "ECDSA" instead of "EC-DSA", as defined in [FIPS.186-4.2013]).

- Client IP Encryption Algorithm (CEA) [optional] - Algorithm used to encrypt the Client IP. If this Information Element is not present in the URI Signing Package Attribute, the default is "AES-128". For interoperability purposes, any encryption algorithm signalled via this Information Element SHALL use the notation as used by NIST (e.g. "AES-128" instead of "AES128", as defined in [FIPS.197.2001]").

- Client IP Key ID (CKI) [optional] - A 64-bit unsigned integer used for obtaining the key (e.g., database lookup) used for encrypting/decrypting the Client IP.

The Version Information Element indicates which version of URI signing scheme is used (including which attributes and algorithms are supported). The present document specifies Version 1. If the Version attribute is not present in the Signed URI, then the version is obtained from the CDNI metadata, else it is considered to have been set to the default value of 1. More versions may be defined in the future.

The Key ID Information Element is used to retrieved the key which is needed as input to the algorithm for validating the Signed URI. The method used for obtaining the actual key from the reference included in the Key ID Information Element is outside the scope of this document. Instead of using the KID element, which is a string, it is possible to use the KID_NUM element for numerical Key identifiers instead. The KID_NUM element is a 64-bit unsigned integer. In cases where numerical KEY IDs are used, it is RECOMMENDED to use KID_NUM instead of KID.

The Hash Function Information Element indicates the hash function to be used for HMAC-based message digest computation. The Hash Function Information Element is used in combination with the Message Digest Information Element defined in section Section 2.3.

The Digital Signature Algorithm Information Element indicates the digital signature function to be in the case asymmetric keys are used. The Digital Signature Algorithm Information Element is used in combination with the Digital Signature Information Element defined in section Section 2.3.

The Client IP Encryption Algorithm Information Element indicates the encryption algorithm to be used for the Client IP. The Client IP Encryption Algorithm Information Element is used in combination with the Client IP Information Element defined in section Section 2.1.

The Client IP Key ID is used to retrieved the key which is used for encrypting and decrypting the Client IP. The method used for

obtaining the actual key from the reference included in the Key ID Information Element is outside the scope of this document. The Client IP Encryption Algorithm Information Element is used in combination with the Client IP Information Element defined in section Section 2.1.

2.3. URI Signature Information Elements

This section identifies the set of information elements that carry the URI Signature that is used for checking the integrity and authenticity of the URI.

The defined keyword for each information element is specified in parenthesis below.

The following information elements are used to carry the actual URI Signature.

- Message Digest (MD) [mandatory for symmetric key] - A string used for the message digest generated by the URI signing entity.

- Digital Signature (DS) [mandatory for asymmetric keys] - A string used for the digital signature provided by the URI signing entity.

The Message Digest attribute contains the message digest used to validate the Signed URI when symmetric keys are used.

The Digital Signature attribute contains the digital signature used to verify the Signed URI when asymmetric keys are used.

In the case of symmetric key, HMAC algorithm is used for the following reasons: 1) Ability to use hash functions (i.e., no changes needed) with well understood cryptographic properties that perform well and for which code is freely and widely available, 2) Easy to replace the embedded hash function in case faster or more secure hash functions are found or required, 3) Original performance of the hash function is maintained without incurring a significant degradation, and 4) Simple way to use and handle keys. The default HMAC algorithm used is SHA-256.

In the case of asymmetric keys, Elliptic Curve Digital Signature Algorithm (EC DSA) - a variant of DSA - is used because of the following reasons: 1) Key size is small while still offering good security, 2) Key is easy to store, and 3) Computation is faster than DSA or RSA.

2.4. URI Signing Package Attribute

The URI Signing Package Attribute is an encapsulation container for the URI Signing Information Elements defined in the previous sections. The URI Signing Information Elements are encoded and stored in this attribute. URI Signing Package Attribute is appended to the Original URI to create the Signed URI.

The primary advantage of the URI Signing Package Attribute is that it avoids having to expose the URI Signing Information Elements directly in the query string of the URI, thereby reducing the potential for a namespace collision space within the URI query string (or the URL path in case path parameters are used). A side-benefit of the attribute is the obfuscation performed by the URI Signing Package Attribute hides the information (e.g., client IP address) from view of the common user, who is not aware of the encoding scheme. Obviously, this is not a security method since anyone who knows the encoding scheme is able to obtain the clear text. Note that any parameters appended to the query string after the URI Signing Package Attribute are not validated and hence do not affect URI Signing.

The following attribute is used to carry the encoded set of URI Signing attributes in the Signed URI.

- URI Signing Package (URISigningPackage) - The encoded attribute containing all the CDNI URI Signing Information Elements used for URI Signing.

The URI Signing Package Attribute contains the URI Signing Information Elements in the Base-64 encoding with URL and Filename Safe Alphabet (a.k.a. "base64url") as specified in the Base-64 Data Encoding [RFC4648] document. The URI Signing Package Attribute is the only URI Signing attribute exposed in the Signed URI. If the Signed URI is communicated via the URI query string, the attribute MUST be the last parameter in the query string of the URI when the Signed URI is generated. However, a client or CDN may append other query parameters unrelated to URI Signing to the Signed URI. Such additional query parameters SHOULD NOT use the same name as the URI Signing Package Attribute to avoid namespace collision and potential failure of the URI Signing validation.

The parameter name of the URI Signing Package Attribute shall be defined in the CDNI Metadata interface. If the CDNI Metadata interface is not used, or does not include a parameter name for the URI Signing Package Attribute, the parameter name is set by configuration (out of scope of this document).

2.5. User Agent Attributes

For some use cases, such as logging, it might be useful to allow the UA, or another entity, add one or more attributes to the Signed URI for purposes other than URI Signing without causing URI Signing to fail. In order to do so, such attributes MUST be appended after the URI Signing Packacke Attribute. Any attributes appended in such way after the URI Signature has been calculated are not validated for the purpose of content access authorization. Adding any such attributes to the Signed URI before the URI Signing Packacke Attribute will cause the URI Signing validation to fail.

Note that a malicious UA might potentially use the ability to append attributes to the Signed URI in order to try to influence the content that is delivered. For example, the UA might append '&quality=HD' to try to make the dCDN deliver an HD version of the requested content. Since such an additional attribute is appended after the URI Signing Package Attribute it is not validated and will not affect the outcome of the URI validation. In order to deal with this vulnerability, a dCDN is RECOMMENDED to ignore any query strings appended after the URI Signing Package Attribute for the purpose of content selection.

3. Create a Signed URI

The following procedure for signing a URI defines the algorithms in this version of URI Signing. Note that some steps may be skipped if the CSP does not enforce a distribution policy and the Enforcement Information Elements are therefore not necessary. A URI (as defined in URI Generic Syntax [RFC3986]) contains the following parts: scheme name, authority, path, query, and fragment. If the Original URI Container information element is used, all components except for the scheme part are protected by the URI Signature. This allows the URI signature to be validated correctly in the case when a client performs a fallback to another scheme (e.g., HTTP) for a content item referenced by a URI with a specific scheme (e.g., RTSP). In case the URI Pattern Container information element is used, the CSP has full flexibility to specify which elements of the URI (including the scheme part) are protected by the URI.

The process of generating a Signed URI can be divided into four sets of steps: 1) Compose URI Signing IEs with original URI / URI pattern, 2) Compute the URI Signature, 3) Encode the URI Signing Package, and 4) Assemble the parts to create the Signed URI. Note it is possible to use some other algorithm and implementation as long as the same result is achieved. An example for the Full Original URI, "http://example.com/content.mov", is used to clarify the steps.

3.1. Compose URI Signing IEs with Protected URI

Calculate the URI Signature by following the procedure below.

1. Create an empty buffer for performing the operations below.

2. If the version is not the default value (i.e. "1"), perform this step. Specify the version by appending the string "VER=#" to the buffer, where '#' represents the new version number. The following steps in the procedure are based on the initial version of URI Signing specified by this document. For other versions, reference the associated RFC for the URI signing procedure.

3. If time window enforcement is needed, perform this step.

   A. If an information element was added to the buffer, append an "&" character. Append the string "ET=". Note in the case of re-signing a URI, the information element MUST be carried over from the received Signed URI.

   B. Get the current time in seconds since epoch (as an integer). Add the validity time in seconds as an integer. Note in the case of re-signing a URI, the value MUST remain the same as the received Signed URI.

   C. Convert this integer to a string and append to the buffer.

4. If client IP enforcement is needed, perform this step.

   A. Skip this step when the Client IP Encryption Algorithm used is the default ("AES-128"). If an information element was added to the buffer, append an "&" character. Append the string "CEA=". Append the string for the Client IP Encryption Algorithm to be used.

   B. If the Client IP Key Identifier is needed, perform this step. If an information element was added to the buffer, append an "&" character. Append the string "CKI=". Append the Client IP key identifier (e.g., "56128239") needed by the entity to locate the shared key for decrypting the Client IP.

   C. If an information element was added to the buffer, append an "&" character. Append the string "CIP=".

   D. Convert the client's IP address in CIDR notation (dotted decimal format for IPv4 or canonical text representation for IPv6 [RFC5952]) to a string and encrypt it using AES-128 (in ECB mode) or another algorithm if specified by the CEA

Information Element. Note in the case of re-signing a URI, the client IP that is encrypted MUST be equal to the unencrypted value of the Client IP as received in the Signed URI, see step 1 in Section 4.5.

E. Convert the encrypted Client IP to its equivalent hexadecimal format.

F. Append the value computed in the previous step to the buffer.

5. If a Key ID information element is needed, perform this step. If an information element was added to the buffer, append an "&" character. Append the string "KID=" in case a string-based Key ID is used, or "KID_NUM=" in case a numerical Key ID is used. Append the key identifier (e.g. "example:keys:123" or "56128239") needed by the entity to locate the shared key for validating the URI signature.

6. If symmetric shared key is used, perform this step. However, skip this step when the hash function for the HMAC uses the default value ("SHA-256"). If an information element was added to the buffer, append an "&" character. Append the string "HF=". Append the string for the new hash function to be used. Note that re-signing a URI MUST use the same hash function as the received Signed URI or one of the allowable hash functions designated by the CDNI metadata.

7. If asymmetric private/public keys are used, perform this step. However, skip this step when the digital signature algorithm uses the default value ("ECDSA"). If an information element was added to the buffer, append an "&" character. Append the string "DSA=". Append the string for the digital signature function. Note that re-signing a URI MUST use the same digital signature algorithm as the received Signed URI or one of the allowable digital signature algorithms designated by the CDNI metadata.

8. Depending on the type of URI enforcement used (Full Original URI or URI Pattern), add the appropriate information element.

A. If enforcement based on the Full Original URI, perform this step. If an information element was added to the buffer, append an "&" character. Append the string "OUC=". Append the Original URI, excluding the "scheme name" part and the "://" delimiter, to the buffer. Note: the Original URI Container information element MUST be the last information element in the buffer before the signature information element.

B. If enforcement based on a URI Pattern, perform this step. If an information element was added to the buffer, append an "&" character. Append the string "UPC=". Append the URI Pattern Container in the form of a percent-encoded string to the buffer.

3.2. Compute URI Signature

Compute the URI Signature by following the procedure below. The buffer from the previous section is used.

1. If symmetric shared key is used, perform this step.

   A. Obtain the shared key to be used for signing the URI.

   B. Append the string "MD=". The buffer now contains the complete section of the URI that is protected (e.g. "ET=1209422976&CKI=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&OUC=example.com/content.mov&MD=").

   C. Compute the message digest using the HMAC algorithm and the default SHA-256 hash function, or another hash function if specified by the HF Information Element, with the shared key and message as the two inputs to the hash function.

   D. Convert the message digest to its equivalent hexadecimal format.

   E. Append the string for the message digest (e.g. "ET=1209422976&CKI=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&OUC=example.com/content.mov&MD=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bddaf").

2. If asymmetric private/public keys are used, perform this step.

   A. Obtain the private key to be used for signing the URI.

   B. If an information element was added to the buffer, append an "&" character. Append the string "DS=". The buffer now contains the complete section of the URI that is protected. (e.g. "ET=1209422976&CKI=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&OUC=example.com/content.mov&DS=").

   C. Compute the message digest using SHA-1 (without a key) for the buffer. Note: The digital signature generated in the next step is calculated over the SHA-1 message digest, instead of over the full cleartext buffer. This is done to reduce the length of the digital signature, the URI Signing

Package Attribute, and the resulting Signed URI. Since SHA-1 is not used for cryptographic purposes here, the security concerns around SHA-1 do not apply.

D. Compute the digital signature, using the EC-DSA algorithm by default, or another algorithm if specified by the DSA Information Element, with the private EC key and message digest (obtained in previous step) as inputs.

E. Convert the digital signature to its equivalent hexadecimal format.

F. Append the string for the digital signature. In the case where EC-DSA algorithm is used, this string contains the values for the 'r' and 's' parameters, delimited by ':' (e.g. "ET=1209422976&CKI=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&OUC=example.com/content.mov&DS=r:CFB03EDB33810AB6C79EE3C47FBD86D227D702F25F66C01CF03F59F1E005668D:s:57ED0E8DF7E786C87E39177DD3398A7FB010E6A4C0DC8AA71331A929A29EA24E")

3.3. Encode the URI Signing Package

Encode the URI Signing Package by following the procedure below. The buffer from the previous section is used.

1. If enforcement is based on the Full Original URI, this step is performed. Remove the Original URI Container Attribute from the buffer, including the preceding "&" character (e.g. "ET=1209422976&CKI=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&MD=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bddaf"). Note: This attribute is not needed in the encoded URI Signing Package because the Full Original URI is part of the Signed URI.

2. Compute the URI Signing Package Attribute using Base-64 Data Encoding [RFC4648] on the message (e.g. "RVQ9MTIwOTQyMjk3NiZhbXA7Q0tJPTMxMSZhbXA7Q0lQPTkwQzkxMzk3NzkzM0ZDNjUwRTcxODYzNjFBOTNENkMzJmFtcDtLSUQ9ZXhhbXBsZTprZXlzOjEyMyZhbXA7TUQ9MWVjYjE0NDZhNjQzMTM1MmFhYjBmYjZlMGRjYTMwZTMwMzU2NTkzYTk3YWNiOTcyMjAyMTIwZGM0ODJiZGRhZg=="). Note: This is the value for the URI Signing Package Attribute.

3.4. Assemble the Signed URI

Assemble the parts to create the Signed URI by following the procedure below.

1. Copy the entire Full Original URI into a new empty buffer.

2. If the Signed URI is communicated via the URI query string, perform this step.

   A. Check if the Full Original URI already contains a query string. If not, append a "?" character. If yes, append an "&" character.

   B. Append the parameter name used to indicate the URI Signing Package Attribute, as communicated via the CDNI Metadata interface, followed by an "=". If none is communicated by the CDNI Metadata interface, it defaults to "URISigningPackage". For example, if the CDNI Metadata interface specifies "SIG", append the string "SIG=" to the message.

   C. Append the URI Signing Package that was generated in previous section (e.g. "http://example.com/content.mov?URISigningPackage=RVQ9MTIwOTQyMjk3NiZhbXA7Q0tJPTMxMSZhbXA7Q0lQPTkwQzkxMzk3NzkzM0ZDNjUwRTcxODYzNjFBOTNENkMzJmFtcDtLSUQ9ZXhhbXBsZTprZXlzOjEyMyZhbXA7TUQ9MWVjYjE0NDZhNjQzMTM1MmFhYjBmYjZlMGRjYTMwZTMwMzU2NTkzYTk3YWNiOTcyMjAyMTIwZGM0ODJiZGRhZg=="). Note: this is the completed Signed URI.

3. If the Signed URI is communicated via a URL path parameter, perform this step.

   A. Check if the Full Original URI already contains a path parameter. If not, add "/;" before the last path component indicating the file to be retrieved. If yes, character at the last append a "?" character. If yes, append an ";" character after the last path parameter.

   B. Append the parameter name used to indicate the URI Signing Package Attribute, as communicated via the CDNI Metadata interface, after the inserted ";" character. If none is communicated by the CDNI Metadata interface, it defaults to "URISigningPackage". Append an "=" character. For example, if the CDNI Metadata interface specifies "SIG" as the parameter name, append the string "SIG=" to the message.

   C. Append the URI Signing Package that was generated in previous section after the "=" character (e.g. "http://example.com/;URISigningPackage=RVQ9MTIwOTQyMjk3NiZhbXA7Q0tJPTMxMSZhbXA7Q0lQPTkwQzkxMzk3NzkzM0ZDNjUwRTcxODYzNjFBOTNENkMzJmFtcDtLSUQ9ZXhhbXBsZTprZXlzOjEyMyZhbXA7TUQ9MWVjYjE0NDZhNjQzMTM1MmFhYjBmYjZlMGR

jYTMwZTMwMzU2NTkzYTk3YWNiOTcyMjAyMTIwZGM0ODJiZGRhZg==/content
.mov"). Note: this is the completed Signed URI.

4. Validate a Signed URI

The process of validating a Signed URI can be divided into five sets of steps: 1) Extract and decode URI Signing Package from the Signed URI, 2) Extract the URI Signing information elements, 3) Obtain the Protected URI, 4) Validate URI signature to ensure integrity of Signed URI, and 5) Ensure proper enforcement of the distribution policy. The integrity of the Signed URI is confirmed before distribution policy enforcement because validation procedure will detect first if the URI has been tampered with. Note it is possible to use some other algorithm and implementation as long as the same result is achieved.

4.1. Extract and Decode URI Signing Package

Extract the encoded URI Signing Package Attribute from the Signed URI. The attribute is decoded for subsequent processing by the Downstream CDN.

1. Extract the value from 'URISigningPackage' attribute. This value is the encoded URI Signing Package Attribute. If there are multiple instances of this attribute, the first one is used and the remaining ones are ignored. This ensures that the Signed URI can be validated despite a client appending another instance of the 'URISigningPackage' attribute.

2. Decode the string using Base-64 Data Encoding [RFC4648] to obtain all the URI Signing information elements (e.g. "ET=1209422976&CK I=311&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&M D=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bdda f").

4.2. Extract URI Signing IEs

Extract the information elements in the URI Signing Package Attribute. Note that some steps are to be skipped if the corresponding URI Signing information elements are not embedded in the attribute. Some of the information elements will be used to validate the URI signature in the subsequent section.

1. Extract the value from "VER" if the information element exists in the decoded URI Signing Package. Determine the version of the URI Signing algorithm used to process the Signed URI. If the CDNI Metadata interface is used, check to see if the used version of the URI Signing algorithm is among the allowed set of

URI Signing versions specified by the metadata. If this is not the case, the request is denied. If the information element is not in the URI, then obtain the version number in another manner (e.g., configuration, CDNI metadata or default value).

2. Extract the value from "MD" if the information element exists in the decoded URI Signing Package. The existence of this information element indicates a symmetric key is used.

3. Extract the value from "DS" if the information element exists in the decoded URI Signing Package. The existence of this information element indicates an asymmetric key is used.

4. If neither "MD" or "DS" attribute is in the decoded URI Signing Package, then no URI Signature exists and the request is denied. If both the "MD" and the "DS" information elements are present, the Signed URI is considered to be malformed and the request is denied.

5. Extract the value from "UPC" if the information element exists in the decoded URI Signing Package and convert it from its percent-encoded form to a regular string. The existence of this information element indicates content delivery is enforced based on a (set of) URI pattern(s) instead of the Full Original URI.

6. Extract the value from "CIP" if the information element exists in the decoded URI Signing Package. The existence of this information element indicates content delivery is enforced based on client IP address.

7. Extract the value from "ET" if the information element exists in the decoded URI Signing Package. The existence of this information element indicates content delivery is enforced based on time.

8. Extract the value from the "KID" or "KID_NUM" information element, if they exist. The existence of either of these information elements indicates a key can be referenced. If both the "KID" and the "KID_NUM" information elements are present, the Signed URI is considered to be malformed and the request is denied.

9. Extract the value from the "HF" information element, if it exists. The existence of this information element indicates a different hash function than the default.

10. Extract the value from the "DSA" information element, if it exists. The existence of this information element indicates a different digital signature algorithm than the default.

11. Extract the value from the "CEA" information element, if it exists. The existence of this information element indicates a different Client IP Encryption Algorithm than the default.

12. Extract the value from the "CKI" information element, if it exists. The existence of this information element indicates a key can be referenced using which the Client IP was encrypted.

4.3. Obtain URI Signing IEs with Protected URI

Obtain the message that contains the URI Signing Information Elements and Protected URI (either Full Original URI or URI pattern). This is the content that was used to generate the URI signature, which is validated by Downstream CDN in the next section.

1. Copy the decoded URI Signing Package into a new buffer to hold the message for performing the operations below. Note: The attribute contains all the URI Signing Information Elements and may also include the URI Pattern Container.

2. Remove the value part of the "MD" or "DS" information element from the message. The part of information element that remains is "MD=" or "DS=".

3. When UPC information element exists, the Protected URI is a set of URIs (i.e., URI Pattern which is conveyed in the value of the UPC IE). Otherwise, the Protected URI is the Full Original URI.

   A. For URI Pattern, the message already contains the Protected URI. Therefore, no additional operation is needed to create the protected URI.

   B. For Full Original URI, the message is missing the Full Original URI in the URI Signing Package. Perform the following steps.

      1. Remove the string "MD=" or "DS=".

      2. Append the string "OUC=". Append the Full Original URI, excluding the "scheme name" part and the "://" delimiter, to the buffer.

      3. Append the "&" character. Append "MD=" or "DS=", depending on which of the two was present in the URI

Signing Package. The message is ready for validation of the message digest (e.g. "ET=1209422976&CIP=90C913977933FC650E7186361A93D6C3&KID=example:keys:123&OUC=example.com/content.mov&MD=").

4.4. Validate URI Signature

Validate the URI Signature for the Signed URI. The message used for computation is obtained from previous section.

1. The received message signature is the value extracted from the "MD" or "DS" information element. Convert the message signature to binary format. This will be used to compare with the computed value later.

2. Based on the presence of either the MD or DS information element in the URI Signing Package, validate the message digest or digital signature for symmetric key or asymmetric keys, respectively.

   A. For MD, an HMAC algorithm is used.

      1. If either the "KID" or "KID_NUM" information element exists, validate that the key identifier is in the allowable KID set as listed in the CDNI metadata or configuration. The request is denied when the key identifier is not allowed. If neither the "KID" or "KID_NUM" information element is present in the Signed URI, obtain the shared key via CDNI metadata or configuration.

      2. If "HF" information element exists, validate that the hash function is in the allowable "HF" set as listed in the CDNI metadata or configuration. The request is denied when the hash function is not allowed. Otherwise, the "HF" information element is not in the Signed URI. In this case, the default hash function is SHA-256.

      3. Compute the message digest using the HMAC algorithm with the shared key and message as the two inputs to the hash function.

      4. Compare the result with the received message signature to validate the Signed URI.

   B. For DS, a digital signature function is used.

1. If either the "KID" or "KID_NUM" information element exists, validate that the key identifier is in the allowable KID set as listed in the CDNI metadata or configuration. The request is denied when the key identifier is not allowed. If neither the "KID" or "KID_NUM" information element is present in the Signed URI, obtain the public key via CDNI metadata or configuration.

2. If "DSA" information element exists, validate that the digital signature algorithm is in the allowable "DSA" set as listed in the CDNI metadata or configuration. The request is denied when the DSA is not allowed. Otherwise, the "DSA" information element is not in the Signed URI. In this case, the default DSA is EC-DSA.

3. Compute the message digest using SHA-1 (without a key) for the message.

4. Verify the digital signature using the digital signature function (e.g., EC-DSA) with the public key, received digital signature, and message digest (obtained in previous step) as inputs. This validates the Signed URI.

4.5. Distribution Policy Enforcement

Note that the absence of a given Enforcement Information Element indicates enforcement of its purpose is not necessary in the CSP's distribution policy.

1. If the "CIP" information element does not exist, this step can be skipped.

   A. Obtain the key for decrypting the Client IP, as indicated by the Client IP Key Index information element or set via configuration.

   B. Decrypt the encrypted Client IP address obtained in step 6 using AES-128, or the algorithm specified by the Client IP Encryption Algorithm information element.

   C. Verify, using CIDR matching, that the request came from an IP address within the range indicated by the decrypted Client IP information element. If the IP address is incorrect, the request is denied.

2. If the "ET" information element exists, validate that the request arrived before expiration time based on the "ET" information element. If the time expired, then the request is denied.

3. If the "UPC" information element exists, validate that the requested resource is in the allowed set by matching the received URI against each of the Patterns in the URI Pattern Container information element until a match is found. If there is no match, the request is denied.

5. Relationship with CDNI Interfaces

Some of the CDNI Interfaces need enhancements to support URI Signing. As an example: A Downstream CDN that supports URI Signing needs to be able to advertise this capability to the Upstream CDN. The Upstream CDN needs to select a Downstream CDN based on such capability when the CSP requires access control to enforce its distribution policy via URI Signing. Also, the Upstream CDN needs to be able to distribute via the CDNI Metadata interface the information necessary to allow the Downstream CDN to validate a Signed URI. Events that pertain to URI Signing (e.g., request denial or delivery after access authorization) need to be included in the logs communicated through the CDNI Logging interface (Editor's Note: Is this within the scope of the CDNI Logging interface?).

5.1. CDNI Control Interface

URI Signing has no impact on this interface.

5.2. CDNI Footprint & Capabilities Advertisement Interface

The Downstream CDN advertises its capability to support URI Signing via the CDNI Footprint & Capabilities Advertisement interface (FCI). The supported version of URI Signing needs to be included to allow for future extensibility.

In general, new information elements introduced to enhance URI Signing requires a draft and a new version.

For Enforcement Information Elements, there is no need to advertise the based information elements such as "CIP" and "ET".

For Signature Computation Information Elements:

No need to advertise "VER" Information Element unless it's not "1". In this case, a draft is needed to describe the new version.

Advertise value of the "HF" Information Element (i.e. SHA-256) to indicate support for the hash function; Need IANA assignment for new hash function.

Advertise value of the "DSA" Information Element (i.e. "ECDSA") to indicate support for the DSA; Need IANA assignment for new digital signature algorithm.

Advertise "MD" Information Element (i.e., SHA-256) to indicate support for symmetric key method; A new draft is needed for an alternative method.

Advertise "DS" Information Element (i.e., "ECDSA") to indicate support for asymmetric key method; A new draft is needed for an alternative method.

For URI Signing Package Attribute, there is no need to advertise the base attribute.

5.3. CDNI Request Routing Redirection Interface

The CDNI Request Routing Redirection Interface [I-D.ietf-cdni-redirection] describes the recursive request redirection method. For URI Signing, the Upstream CDN signs the URI provided by the Downstream CDN. This approach has the following benefits:

Consistency with interative request routing method

URI Signing is fully operational even when Downstream CDN does not have the signing function (which may be the case when the Downstream CDN operates only as a delivering CDN)

Upstream CDN can act as a conversion gateway for the requesting routing interface between Upstream CDN and CSP and request routing interface between Upstream CDN and Downstream CDN since these two interfaces may not be the same

5.4. CDNI Metadata Interface

The CDNI Metadata Interface [I-D.ietf-cdni-metadata] describes the CDNI metadata distribution in order to enable content acquisition and delivery. For URI Signing, additional CDNI metadata objects are specified. In general, an Empty set means "all". These are the CDNI metadata objects used for URI Signing.

The UriSigning Metadata object contains information to enable URI signing and validation by a dCDN. The UriSigning properties are defined below.

Property: enforce

Description: URI Signing enforcement flag. Specifically, this flag indicates if the access to content is subject to URI Signing. URI Signing requires the Downstream CDN to ensure that the URI must be signed and validated before content delivery. Otherwise, Downstream CDN does not perform validation regardless if URI is signed or not.

Type: Boolean

Mandatory-to-Specify: No. If a UriSigning object is present in the metadata for a piece of content (even if the object is empty), then URI signing should be enforced. If no UriSigning object is present in the metadata for a piece of content, then the URI signature should not be validated.

Property: key-id

Description: Designated key identifier used for URI Signing computation when the Signed URI does not contain the Key ID information element.

Type: String

Mandatory-to-Specify: No. A Key ID is not essential for all implementations of URI signing.

Property: key-id-set

Description: Allowable Key ID set that the Signed URI's Key ID information element can reference.

Type: List of Strings

Mandatory-to-Specify: No. Default is to allow any Key ID.

Property: hash-function

Description: Designated hash function used for URI Signing computation when the Signed URI does not contain the Hash Function information element.

Type: String (limited to the hash function strings in the registry defined by the IANA Considerations (Section 8) section)

Mandatory-to-Specify: No. Default is SHA-256.

Property: hash-function-set

Description: Allowable Hash Function set that the Signed URI's Hash Function information element can reference.

Type: List of Strings

Mandatory-to-Specify: No. Default is to allow any hash function.

Property: digital-signature-algorithm

Description: Designated digital signature function used for URI Signing computation when the Signed URI does not contain the Digital Signature Algorithm information element.

Type: String (limited to the digital signature algorithm strings in the registry defined by the IANA Considerations (Section 8) section).

Mandatory-to-Specify: No. Default is "ECDSA".

Property: digital-signature-algorithm-set

Description: Allowable digital signature function set that the Signed URI's Digital Signature Algorithm information element can reference.

Type: List of Strings

Mandatory-to-Specify: No. Default is to allow any DSA.

Property: version

Description: Designated version used for URI Signing computation when the Signed URI does not contain the VER attribute.

Type: Integer

Mandatory-to-Specify: No. Default is 1.

Property: version-set

Description: Allowable version set that the Signed URI's VER attribute can reference.

Type: List of Integers

Mandatory-to-Specify: No. Default is to allow any version.

Property: package-attribute

Description: Overwrite the default name for the URL Signing Package Attribute.

Type: String

Mandatory-to-Specify: No. Default is "URISigningPackage".

Note that the Key ID information element is not needed if only one key is provided by the CSP or the Upstream CDN for the content item or set of content items covered by the CDNI Metadata object. In the case of asymmetric keys, it's easy for any entity to sign the URI for content with a private key and provide the public key in the Signed URI. This just confirms that the URI Signer authorized the delivery. But it's necessary for the URI Signer to be the content owner. So, the CDNI Metadata interface or configuration MUST provide the allowable Key ID set to authorize the Key ID information element embedded in the Signed URI.

The following is an example of a URI Signing metadata payload with all default values:

```
{
  "generic-metadata-type": "MI.UriSigning.v1"
  "generic-metadata-value": {}
}
```

The following is an example of a URI Signing metadata payload with explicit values:

```
{
  "generic-metadata-type": "MI.UriSigning.v1"
  "generic-metadata-value":
    {
      "enforce": true,
      "key-id": "1",
      "key-id-set": ["1", "2", "3"],
      "hash-function": "SHA-512",
      "hash-function-set": ["SHA-384", "SHA-512"],
      "digital-signature-algorithm": "ECDSA",
      "digital-signature-algorithm-set": ["ECDSA"],
      "version": 1,
      "version-set": [1],
      "package-attribute": "usp"
    }
}
```

5.5. CDNI Logging Interface

For URI Signing, the Downstream CDN reports that enforcement of the access control was applied to the request for content delivery. When the request is denied due to enforcement of URI Signing, the reason is logged.

The following CDNI Logging field for URI Signing SHOULD be supported in the HTTP Request Logging Record as specified in CDNI Logging Interface [I-D.ietf-cdni-logging].

- s-uri-signing (mandatory):
  - format: 3DIGIT
  - field value: this characterises the URI signing validation performed by the Surrogate on the request. The allowed values are:
    - "000" : no URI signature validation performed
    - "200" : URI signature validation performed and validated
    - "400" : URI signature validation performed and rejected because of incorrect signature
    - "401" : URI signature validation performed and rejected because of Expiration Time enforcement

+ "402" : URI signature validation performed and rejected because of Client IP enforcement

+ "403" : URI signature validation performed and rejected because of URI Pattern enforcement

+ "500" : unable to perform URI signature validation because of malformed URI

+ "501" : unable to perform URI signature validation because of unsupported version number

* occurrence: there MUST be zero or exactly one instance of this field.

o s-uri-signing-deny-reason (optional):

* format: QSTRING

* field value: a string for providing further information in case the URI signature was rejected, e.g., for debugging purposes.

* occurrence: there MUST be zero or exactly one instance of this field.

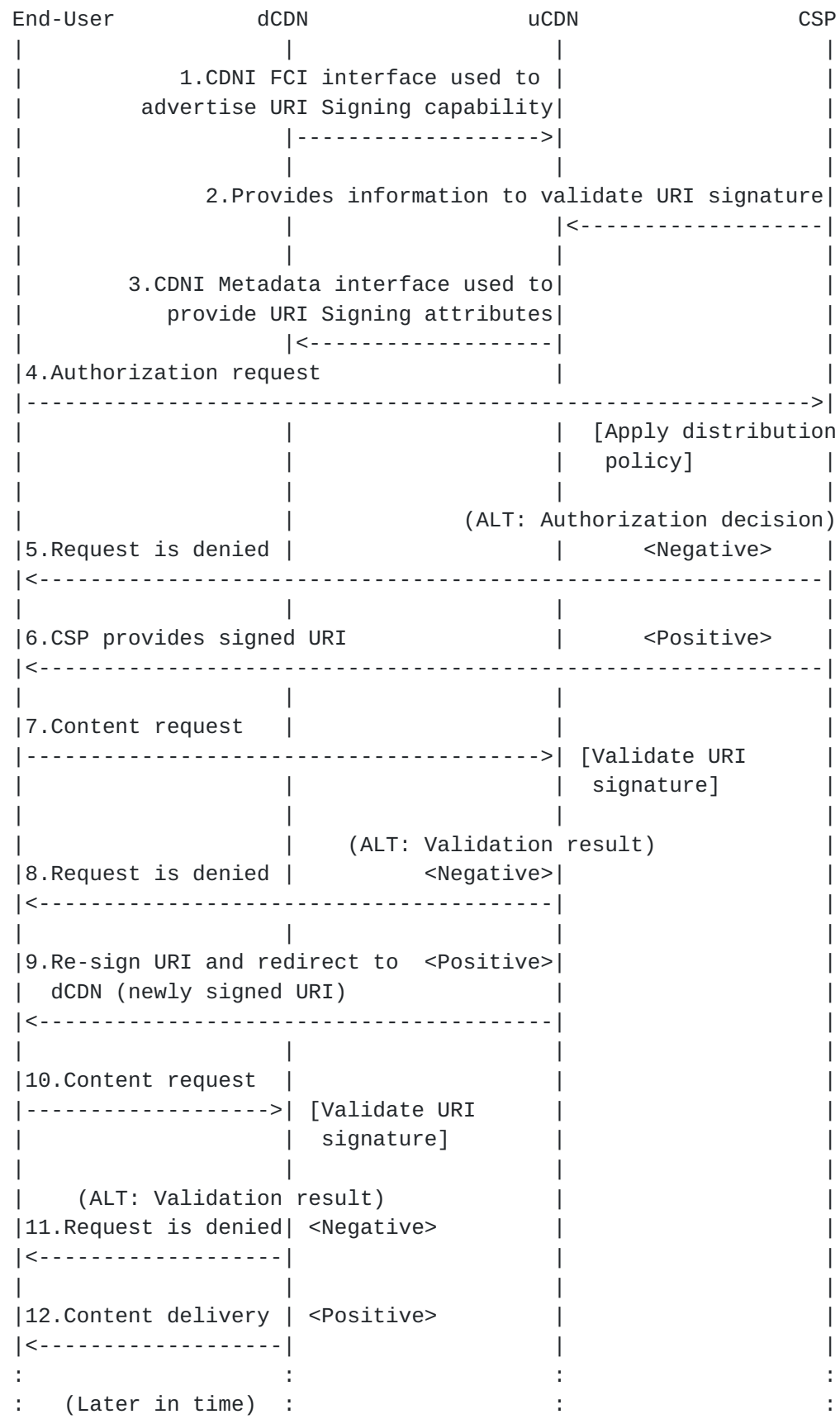
6. URI Signing Message Flow

URI Signing supports both HTTP-based and DNS-based request routing. HMAC [RFC2104] defines a hash-based message authentication code allowing two parties that share a symmetric key or asymmetric keys to establish the integrity and authenticity of a set of information (e.g., a message) through a cryptographic hash function.

6.1. HTTP Redirection

For HTTP-based request routing, HMAC is applied to a set of information that is unique to a given end user content request using key information that is specific to a pair of adjacent CDNI hops (e.g. between the CSP and the Authoritative CDN, between the Authoritative CDN and a Downstream CDN). This allows a CDNI hop to ascertain the authenticity of a given request received from a previous CDNI hop.

The URI signing scheme described below is based on the following steps (assuming HTTP redirection, iterative request routing and a CDN path with two CDNs). Note that Authoritative CDN and Upstream CDN are used exchangeably.


End-User
dCDN
uCDN
CSP
1.CDNI FCI interface used to advertise URI Signing capability
2.Provides information to validate URI signature
3.CDNI Metadata interface used to provide URI Signing attributes
4.Authorization request
[Apply distribution policy]
(ALT: Authorization decision)
5.Request is denied
<Negative>
6.CSP provides signed URI
<Positive>
7.Content request
[Validate URI signature]
(ALT: Validation result)
8.Request is denied
<Negative>
9.Re-sign URI and redirect to dCDN (newly signed URI)
<Positive>
10.Content request
[Validate URI signature]
(ALT: Validation result)
11.Request is denied
<Negative>
12.Content delivery
<Positive>
(Later in time)

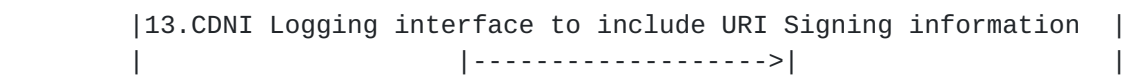


Figure 3: HTTP-based Request Routing with URI Signing

1. Using the CDNI Footprint & Capabilities Advertisement interface, the Downstream CDN advertises its capabilities including URI Signing support to the Authoritative CDN.

2. CSP provides to the Authoritative CDN the information needed to validate URI signatures from that CSP. For example, this information may include a hashing function, algorithm, and a key value.

3. Using the CDNI Metadata interface, the Authoritative CDN communicates to a Downstream CDN the information needed to validate URI signatures from the Authoritative CDN for the given CSP. For example, this information may include the URI query string parameter name for the URI Signing Package Attribute, a hashing algorithm and/or a key corresponding to the trust relationship between the Authoritative CDN and the Downstream CDN.

4. When a UA requests a piece of protected content from the CSP, the CSP makes a specific authorization decision for this unique request based on its arbitrary distribution policy

5. If the authorization decision is negative, the CSP rejects the request.

6. If the authorization decision is positive, the CSP computes a Signed URI that is based on unique parameters of that request and conveys it to the end user as the URI to use to request the content.

7. On receipt of the corresponding content request, the authoritative CDN validates the URI Signature in the URI using the information provided by the CSP.

8. If the validation is negative, the authoritative CDN rejects the request

9. If the validation is positive, the authoritative CDN computes a Signed URI that is based on unique parameters of that request and provides to the end user as the URI to use to further request the content from the Downstream CDN

10. On receipt of the corresponding content request, the Downstream CDN validates the URI Signature in the Signed URI using the information provided by the Authoritative CDN in the CDNI Metadata

11. If the validation is negative, the Downstream CDN rejects the request and sends an error code (e.g., 403) in the HTTP response.

12. If the validation is positive, the Downstream CDN serves the request and delivers the content.

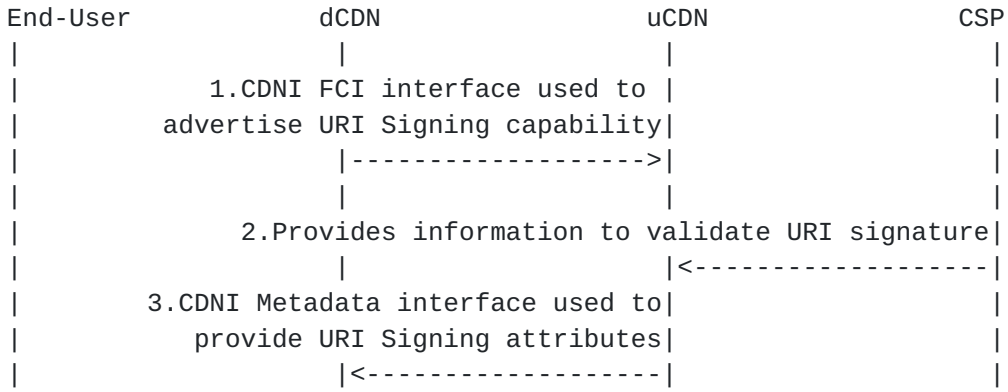
13. At a later time, Downstream CDN reports logging events that includes URI signing information.

With HTTP-based request routing, URI Signing matches well the general chain of trust model of CDNI both with symmetric key and asymmetric keys because the key information only need to be specific to a pair of adjacent CDNI hops.

6.2. DNS Redirection

For DNS-based request routing, the CSP and Authoritative CDN must agree on a trust model appropriate to the security requirements of the CSP's particular content. Use of asymmetric public/private keys allows for unlimited distribution of the public key to Downstream CDNs. However, if a shared secret key is preferred, then the CSP may want to restrict the distribution of the key to a (possibly empty) subset of trusted Downstream CDNs. Authorized Delivery CDNs need to obtain the key information to validate the Signed UR, which is computed by the CSP based on its distribution policy.

The URI signing scheme described below is based on the following steps (assuming iterative DNS request routing and a CDN path with two CDNs). Note that Authoritative CDN and Upstream CDN are used exchangeably.



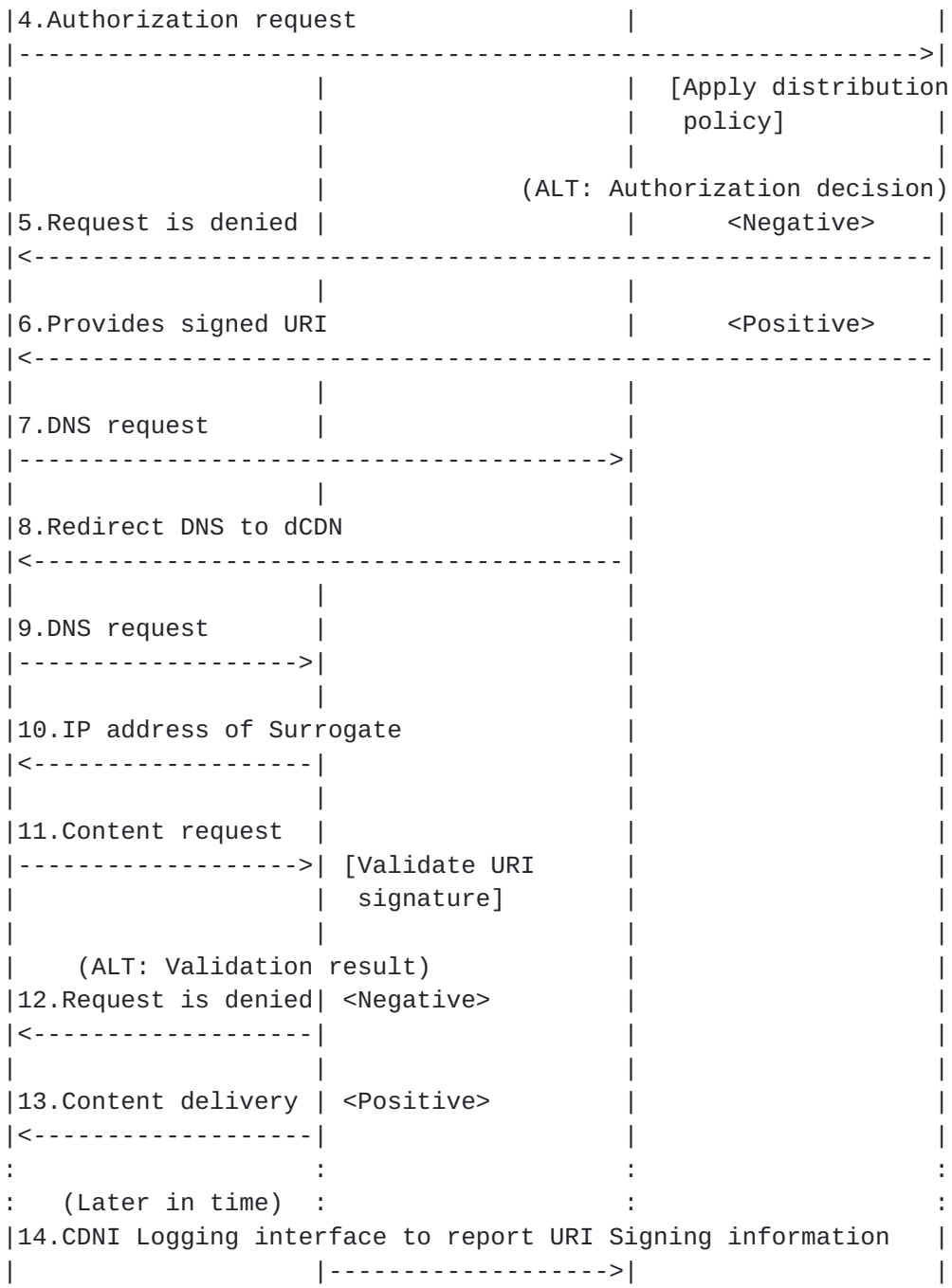


Figure 4: DNS-based Request Routing with URI Signing

1. Using the CDNI Footprint & Capabilities Advertisement interface, the Downstream CDN advertises its capabilities including URI Signing support to the Authoritative CDN.

2. CSP provides to the Authoritative CDN the information needed to validate cryptographic signatures from that CSP. For example,

this information may include a hash function, algorithm, and a key.

3. Using the CDNI Metadata interface, the Authoritative CDN communicates to a Downstream CDN the information needed to validate cryptographic signatures from the CSP (e.g., the URI query string parameter name for the URI Signing Package Attribute). In the case of symmetric key, the Authoritative CDN checks if the Downstream CDN is allowed by CSP to obtain the shared secret key.

4. When a UA requests a piece of protected content from the CSP, the CSP makes a specific authorization decision for this unique request based on its arbitrary distribution policy.

5. If the authorization decision is negative, the CSP rejects the request

6. If the authorization decision is positive, the CSP computes a cryptographic signature that is based on unique parameters of that request and includes it in the URI provided to the end user to request the content.

7. End user sends DNS request to the authoritative CDN.

8. On receipt of the DNS request, the authoritative CDN redirects the request to the Downstream CDN.

9. End user sends DNS request to the Downstream CDN.

10. On receipt of the DNS request, the Downstream CDN responds with IP address of one of its Surrogates.

11. On receipt of the corresponding content request, the Downstream CDN validates the cryptographic signature in the URI using the information provided by the Authoritative CDN in the CDNI Metadata

12. If the validation is negative, the Downstream CDN rejects the request and sends an error code (e.g., 403) in the HTTP response.

13. If the validation is positive, the Downstream CDN serves the request and delivers the content.

14. At a later time, Downstream CDN reports logging events that includes URI signing information.

With DNS-based request routing, URI Signing matches well the general chain of trust model of CDNI when used with asymmetric keys because the only key information that need to be distributed across multiple CDNI hops including non-adjacent hops is the public key, that is generally not confidential.

With DNS-based request routing, URI Signing does not match well the general chain of trust model of CDNI when used with symmetric keys because the symmetric key information needs to be distributed across multiple CDNI hops including non-adjacent hops. This raises a security concern for applicability of URI Signing with symmetric keys in case of DNS-based inter-CDN request routing.

7. HTTP Adaptive Streaming

The authors note that in order to perform URI signing for individual content segments of HTTP Adaptive Bitrate content, specific URI signing mechanisms are needed. Such mechanisms are currently out-of-scope of this document. More details on this topic is covered in Models for HTTP-Adaptive-Streaming-Aware CDNI [RFC6983]. In addition, [I-D.brandenburg-cdni-uri-signing-for-has] provides an extension to the algorithm defined in this document that deals specifically with URI signing of segmented content.

8. IANA Considerations

8.1. CDNI Payload Type

This document requests the registration of the following CDNI Payload Type under the IANA "CDNI Payload Type" registry:

Payload Type	Specification
MI.UriSigning.v1	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

8.1.1. CDNI UriSigning Payload Type

Purpose: The purpose of this payload type is to distinguish UriSigning MI objects (and any associated capability advertisement).

Interface: MI/FCI

Encoding: see Section 5.4

8.2. CDNI Logging Record Type

This document requests the registration of the following CDNI Logging record-type under the IANA "CDNI Logging record-types" registry:

record-types	Reference	Description
cdni_http_request_v2	RFCthis	Extension to CDNI Logging Record version 1 for content delivery using HTTP, to include URI Signing logging fields

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

8.2.1. CDNI Logging Record Version 2 for HTTP

The "cdni_http_request_v2" record-type supports all of the fields supported by the "cdni_http_request_v1" record-type [I-D.ietf-cdni-logging] plus the two additional fields "s-uri-signing" and "s-uri-signing-deny-reason", registered by this document in Section 8.3. The name, format, field value, and occurence information for the two new fields can be found in Section 5.5 of this document.

8.3. CDNI Logging Field Names

This document requests the registration of the following CDNI Logging fields under the IANA "CDNI Logging Field Names" registry:

Field Name	Reference
s-uri-signing	RFCthis
s-uri-signing-deny-reason	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

8.4. CDNI Metadata Auth Type

This document requests the registration of the following CDNI Metadata Auth type under the IANA "CDNI Metadata Auth Types" registry:

Auth type	Description	Specification
MI.UriSigning.v1	URI Signing version 1	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

8.5. CDNI URI Signing Enforcement Information Elements

The IANA is requested to create a new "CDNI URI Signing Enforcement Information Elements" subregistry in the "Content Delivery Networks Interconnection (CDNI) Parameters" registry. The "CDNI URI Signing Enforcement Information Elements" namespace defines the valid Enforcement Information Elements that may be included in a URI Signing token. Additions to the Enforcement Information Elements namespace conform to the "Specification Required" policy as defined in [RFC5226].

The following table defines the initial Enforcement Information Elements:

Element	Description	RFC
ET	Expiry Time	RFCthis
CIP	Client IP Address	RFCthis
OUC	Original URI Container	RFCthis
URI Pattern Container	Client IP Address	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

[Ed Note: are there any special instructions to the designated expert reviewer?]

8.6. CDNI URI Signing Signature Computation Information Elements

The IANA is requested to create a new "CDNI URI Signing Signature Computation Information Elements" subregistry in the "Content Delivery Networks Interconnection (CDNI) Parameters" registry. The "CDNI URI Signing Signature Computation Information Elements" namespace defines the valid Signature Computation Information Elements that may be included in a URI Signing token. Additions to the Signature Computation Information Elements namespace conform to the "Specification Required" policy as defined in [RFC5226].

The following table defines the initial Signature Computation Information Elements:

Element	Description	RFC
VER	Version Number	RFCthis
KID	Non-numerical Key Identifier	RFCthis
KID_NUM	Numerical Key Identifier	RFCthis
HF	Hash Function	RFCthis
DSA	Digital Signature Algorithm	RFCthis
CEA	Client IP Encryption Algorithm	RFCthis
CKI	Client IP Encryption Key Identifier	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

[Ed Note: are there any special instructions to the designated expert reviewer?]

8.7. CDNI URI Signing Signature Information Elements

The IANA is requested to create a new "CDNI URI Signing Signature Information Elements" subregistry in the "Content Delivery Networks Interconnection (CDNI) Parameters" registry. The "CDNI URI Signing Signature Information Elements" namespace defines the valid Signature Information Elements that may be included in a URI Signing token. Additions to the Signature Information Elements namespace conform to the "Specification Required" policy as defined in [RFC5226].

The following table defines the initial Signature Information Elements:

Element	Description	RFC
MD	Message Digest for Symmetric Key	RFCthis
DS	Digital Signature for Asymmetric Keys	RFCthis

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

[Ed Note: are there any special instructions to the designated expert reviewer?]

9. Security Considerations

This document describes the concept of URI Signing and how it can be used to provide access authorization in the case of interconnected CDNs (CDNI). The primary goal of URI Signing is to make sure that only authorized UAs are able to access the content, with a Content Service Provider (CSP) being able to authorize every individual request. It should be noted that URI Signing is not a content protection scheme; if a CSP wants to protect the content itself, other mechanisms, such as DRM, are more appropriate.

In general, it holds that the level of protection against illegitimate access can be increased by including more Enforcement Information Elements in the URI. The current version of this document includes elements for enforcing Client IP Address and Expiration Time, however this list can be extended with other, more complex, attributes that are able to provide some form of protection against some of the vulnerabilities highlighted below.

That said, there are a number of aspects that limit the level of security offered by URI signing and that anybody implementing URI signing should be aware of.

Replay attacks: Any (valid) Signed URI can be used to perform replay attacks. The vulnerability to replay attacks can be reduced by picking a relatively short window for the Expiration Time attribute, although this is limited by the fact that any HTTP-based request needs a window of at least a couple of seconds to prevent any sudden network issues from preventing legitimate UAs access to the content. One way to reduce exposure to replay attacks is to include in the URI a unique one-time access ID. Whenever the Downstream CDN receives a request with a given unique access ID, it adds that access ID to the list of 'used' IDs. In the case an illegitimate UA tries to use the same URI through a replay attack, the Downstream CDN can deny the request based on the already-used access ID.

Illegitimate client behind a NAT: In cases where there are multiple users behind the same NAT, all users will have the same IP address from the point of view of the Downstream CDN. This results in the Downstream CDN not being able to distinguish between the different users based on Client IP Address and illegitimate users being able to access the content. One way to reduce exposure to this kind of attack is to not only check for Client IP but also for other attributes that can be found in the HTTP headers.

The shared key between CSP and Authoritative CDN may be distributed to Downstream CDNs - including cascaded CDNs. Since this key can be used to legitimately sign a URL for content access authorization, it's important to know the implications of a compromised shared key.

In the case where asymmetric keys are used, the KID information element might contain the URL to the public key. To prevent malicious clients from signing their own URIs and inserting the associated public key URL in the KID field, thereby passing URI validation, it is important that CDNs check whether the URI conveyed in the KID field is in the allowable set of KIDs as listed in the CDNI metadata or set via configuration.

10. Privacy

The privacy protection concerns described in CDNI Logging Interface [I-D.ietf-cdni-logging] apply when the client's IP address (CIP attribute) is embedded in the Signed URI. For this reason, the mechanism described in Section 3.1 encrypts the Client IP before including it in the URI Signing Package (and thus the URL itself).

11. Acknowledgements

The authors would like to thank the following people for their contributions in reviewing this document and providing feedback: Scott Leibrand, Kevin Ma, Ben Niven-Jenkins, Thierry Magnien, Dan York, Bhaskar Bhupalam, Matt Caulfield, Samuel Rajakumar, Iuniana Oprescu, Leif Hedstrom, Phil Sorber and Gancho Tenev. In addition, Matt Caulfield provided content for the CDNI Metadata Interface section.

12. References

12.1. Normative References

[I-D.ietf-cdni-logging]
Faucheur, F., Bertrand, G., Oprescu, I., and R. Peterkofsky, "CDNI Logging Interface", draft-ietf-cdni-logging-27 (work in progress), June 2016.

[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>.

[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>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <http://www.rfc-editor.org/info/rfc6707>.

12.2. Informative References

[FIPS.180-1.1995]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, April 1995, <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.

[FIPS.186-4.2013]
National Institute of Standards and Technology, "Digital Signature Standard", FIPS PUB 186-1, December 1998, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.184-4.pdf>.

[FIPS.197.2001]
National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[I-D.brandenburg-cdni-uri-signing-for-has]
Brandenburg, R., "URI Signing for HTTP Adaptive Streaming (HAS)", draft-brandenburg-cdni-uri-signing-for-has-03 (work in progress), June 2016.

[I-D.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "CDN Interconnection Metadata", draft-ietf-cdni-metadata-18 (work in progress), June 2016.

[I-D.ietf-cdni-redirection]
Niven-Jenkins, B. and R. Brandenburg, "Request Routing Redirection interface for CDN Interconnection", draft-ietf-cdni-redirection-18 (work in progress), April 2016.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

[RFC6983] van Brandenburg, R., van Deventer, O., Le Faucheur, F., and K. Leung, "Models for HTTP-Adaptive-Streaming-Aware Content Distribution Network Interconnection (CDNI)", RFC 6983, DOI 10.17487/RFC6983, July 2013, <http://www.rfc-editor.org/info/rfc6983>.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <http://www.rfc-editor.org/info/rfc7336>.

[RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <http://www.rfc-editor.org/info/rfc7337>.

Authors' Addresses

Kent Leung
Cisco Systems
3625 Cisco Way
San Jose 95134
USA

Phone: +1 408 526 5030
Email: kleung@cisco.com

Francois Le Faucheur
Cisco Systems
Greenside, 400 Avenue de Roumanille
Sophia Antipolis 06410
France

Phone: +33 4 97 23 26 19
Email: flefauch@cisco.com

Ray van Brandenburg
TNO
Anna van Buerenplein 1
Den Haag 2595DC
the Netherlands

Phone: +31 88 866 7000
Email: ray.vanbrandenburg@tno.nl

Bill Downey
Verizon Labs
60 Sylvan Road
Waltham, Massachusetts 02451
USA

Phone: +1 781 466 2475
Email: william.s.downey@verizon.com

Michel Fisher
Limelight Networks
222 S Mill Ave
Tempe, AZ 85281
USA

Phone: +1 360 419 5185
Email: mfisher@llnw.com