CDNI
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2015

K. Leung
F. Le Faucheur
Cisco Systems
R. van Brandenburg
TNO
B. Downey
Verizon Labs
M. Fisher
Limelight Networks
June 1, 2015

URI Signing for CDN Interconnection (CDNI)
draft-ietf-cdni-uri-signing-04

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 algorithm includes specific provisions for allowing access control of HTTP Adaptive Streaming content that is characterized by independent chunks referenced by a manifest file.

The proposed mechanism is also applicable in a non-CDNI context.

Status of This Memo

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 interconnected CDNs (CDNI). The primary goal of URI Signing is to make sure that only authorized User Agents (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.

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

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] including the following terms (reproduced here for convenience):

- o MAC: message authentication code.

- o HMAC: Hash-based message authentication code (HMAC) is a specific construction for calculating a MAC involving a cryptographic hash function in combination with a secret key.

- HMAC-SHA256: HMAC instantiation using SHA-256 as the cryptographic hash function.

- SHA-1: Secure Hash Algorithm 1 (SHA-1) [RFC3174] is the cryptographic hash function.

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 and/or a set of URI Signing Information Elements.

- URI Signing Information Element: An element containing information used in the URI Signature validation process that is signalled along with the URI Signature in either the Signed URI or in a Signed Token. It is protected by the URI Signature.

- Original URI: The URI before URI Signing is applied.

- Signed URI: Any URI that contains a URI Signature. It can be used to retrieve one particular resource, indicated by the URI.

- Signed Token: A set of URI Signing Information Elements protected by a URI Signature that can be used to retrieve a pre-determined set of resources. It can be communicated via a URL, an HTTP Header or a Cookie. A Signed Token differs from a Signed URI in the sense that the token is valid for multiple resources and it's embedded URI Signature thus does not protect a particular URI. It can be used to form a Signed Token Chain in the case of HTTP Adaptive Streaming content.

- 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 on URI Signing

The next section provides an overview of how URI Signing works in a CDNI environment. As background information, URI Signing is first explained in terms of a single CDN delivering content on behalf of a CSP.

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. In addition, 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).

```
                    --------
                  /          \
                  |   CSP    |< * * * * * * * * * * *
                  \          /        Trust          *
                    --------       relationship      *
                       ^   |                         *
                       |   |                         *
         1. Browse     |   | 2. Signed               *
              for      |   |    URI                  *
            content    |   |                         *
                       |   v                         v
                    +-------+ 3. Signed URI       --------
                    | User  |----------------->/          \
                    | Agent|                  |   CDN    |
                    |       |<-----------------\          /
                    +-------+ 4. Content          --------
                                 Delivery
```

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 7).

```
                                    +-------------------------+
                                    |Request Redirection Modes|
                                    +-------------------------+
                                    | a) HTTP                 |
```

```
                                      | b) DNS                  |
                                      +--------------------------+
                   --------
                  /        \< * * * * * * * * * * * * * *
                  |   CSP  |< * * * * * * * * * * *     *
                  \        /        Trust         *     *
                   --------      relationship     *     *
                     ^  |                         *     *
                     |  | 2. Signed               *     *
          1. Browse  |  |    URI in               *     *
               for   |  |    HTML                 *     *
             content |  |                         *     *
                     |  v   3.a)Signed URI        v     *
                   +------+   b)DNS request   --------  * Trust
                   | User |----------------->/        \ * relationship
                   | Agent|                  |  uCDN  | * (optional)
                   |      |<-----------------\        / *
                   +------+ 4.a)Redirection URI-------  *
                     ^  |     b)DNS Reply         ^     *
                     |  |                         *     *
                     |  |      Trust relationship *     *
                     |  |                         *     *
         6. Content  |  | 5.a)Redirection URI     *     *
            delivery |  |   b)Signed URI(after    v     v
                     |  |     DNS exchange)     --------
                     |  +---------------------->/        \ [May be
                     |                          |  dCDN  |  cascaded
                     +--------------------------\        /  CDNs]
                                                 --------
```

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 SHOULD maintain the same 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. URI Signing and HTTP Adaptive Streaming

For content that is delivered via an HTTP Adaptive Streaming (HAS) protocol, such as MPEG DASH [Editor's Note: Include reference], special provisions need to be made in order to ensure URI Signing can be applied. In general, HAS protocols work by breaking large objects (e.g. videos), into a sequence of small, independent chunks. Such chunks are then referenced by a separate manifest file, which either includes a list of URLs to the chunks or specifies an algorithm through which a User Agent can construct the URLs to the chunks. Requests for chunks therefore originate from the manifest file and, unless the URLs in the manifest file point to the CSP, are not subjected to redirection and URI Signing. This opens up the vulnerability of malicious User Agents sharing the manifest file and deep-linking to the chunks.

One method for dealing with this vulnerability would be to include in the manifest itself Signed URIs that point to the individual chunks. There exist a number of issues with that approach. First, it requires the CDN delivering the manifest to rewrite the manifest file for each User Agent, which would require the CDN to be aware of the exact HAS protocol and version used. Secondly, it would require the expiration time of the Signed URIs to be valid for at least the full duration of the content described by the manifest. Since it is not uncommon for a manifest file to contain a video item of more than 30 minutes in length, this would require the Signed URIs to be valid for a long time, thereby reducing their effectiveness and that of the URI Signing mechanism in general. For a more detailed analysis of how HAS protocols affect CDNI, see Models for HTTP-Adaptive-Streaming-Aware CDNI [RFC6983].

In order to allow for effective access control of HAS content, the URI signing scheme defined in this document instead supports a mechanism through which subsequent chunk requests can be chained together. As part of the URI validation procedure, the CDN can generate a Signed Token that the UA can use to do a subsequent request. More specifically, whenever a UA successfully retrieves a chunk, it receives, in the HTTP 2xx Successful message, a Signed Token that it can use whenever it requests the next chunk. As long as each Signed Token in the chain is correctly validated before a new one is generated, the chain is not broken and the User Agent can successfully retrieve additional chunks. Given the fact that with HAS protocols, it is usually not possible to determine a priori which chunk will be requested next (i.e. to allow for seeking within the content and for switching to a different quality level), the Signed Token includes a scoping mechanism that allows it to be valid for more than one URL.

In order for this chaining of Signed Tokens to work, it is necessary for a UA to extract the Signed Token from the HTTP 2xx Successful message of an earlier request and use it to retrieve the next chunk. The exact mechanism by which the client does this depends on the exact HAS protocol and since this document is only concerned with the generation and validation of incoming request, this process is outside the scope of this document. However, in order to also support legacy UAs that do not include any specific provisions for the handling of Signed Tokens, this document does define a legacy mechanism using HTTP Cookies that allows such UAs to support the concept of chained Signed Tokens without requiring any support on the UA side.

3. Signed URI Information Elements

The concept behind URI Signing is based on embedding in either the Signed URI or Signed Token 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 to form the Signed URI. Alternatively, the encapsulated elements are placed in the Signed Token.

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

- o 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. Another example is the Path Pattern information element that is used to define the scope of a Signed Token.

- o 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.

o 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 Signed URI or Signed Token. A typical Signed URI or Token will only contain one embedded URI Signature Information Element.

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

o URI Signing Package Attribute: The URI attribute is a container that encapsulates all the URI Signing Information Elements in an encoded format. In the case of a Signed URI, only this attribute is exposed as a URI query string parameter. In the case of a Signed Token, this attribute constitutes the Signed Token and is communicated in either the URI, a dedicated header, or as an HTTP Cookie.

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 Signing Package Attribute query string attribute are mandatory to implement, but not mandatory to use.

3.1. 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 a Signed URI or Signed Token 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:

o Expiry Time (ET) [optional] - Time when the Signed URI or Signed Token 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 or Signed Token need to be in sync (e.g. NTP is used).

o Client IP (CIP) [optional] - IP address of the client for which this Signed URI or Signed Token is generated. This is represented in dotted decimal format for IPv4 or canonical text representation for IPv6 address [RFC5952] . The request is rejected if sourced from a client with a different IP address.

o Path Pattern (PP) [mandatory for Signed Token] - Path Pattern that describes for which content the Signed Token is valid. The Path Pattern contains an expression to match against the requested URI path to check whether the requested content is allowed to be requested with the Signed Token. A Path Pattern contains a sequence of one or more path segments seperated by a slash ('/') character. 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 Path Pattern: '*/folder/content-83112371/quality_*/segment????.mp4'. The Path Pattern Information Element MUST NOT be used in a Signed URI.

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 User Agent, based on its IP address for whom the content access was authorized.

The Path Pattern Information Element is used to restrict content access to a particular set of URLs, based on the path component of the URI on which it is available. This is primarily useful for content delivered via an HTTP Adaptive Streaming protocol using a manifest file, where it is often not known a priori which segment will be requested next.

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

3.2. Signature Computation Information Elements

This section identifies the set of information elements that may be needed to verify the URI Signature and/or calculate a new Signed Token. 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 and/or calculate a new Signed Token.

o 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 Signed URI or Signed Token, the default version is 1.

o 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 Signed URI or Signed Token.

o 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 Signed URI or Signed Token.

o 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 Signed URI or Signed Token, the default hash function is SHA-256.

o Digital Signature Algorithm (DSA) [optional] - Algorithm used to calculate the Digital Signature. If this Information Element is not present in the Signed URI or Signed Token, the default is EC-DSA.

o URI Signing Cookie Flag (USCF) [optional] - The presence of this flag indicates the URI Signing Information Elements contents of the URI Signing Package Attribute are communicated via the Cookie header of the HTTP request instead of via the query string component of the URI.

o Expiration Time Setting (ETS) [optional] - An 16-bit unsigned integer (in seconds) used for setting the value of the Expiry Time information element in newly generated Signed Tokens in case a chain of Signed Tokens is used.

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 or Signed Token, 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 or Signed Token. 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 3.3.

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 3.3.

The URI Signing Cookie Flag Information Element is used to indicate the contents of the URI Signing Package attribute is communicated via the Cookie header of the HTTP request instead of via the query string component of the URI. The primary use case for this is the case where the chained Signed Token mechanism described in Section 2 is used in combination with legacy UAs.

The Expiration Time Setting Information Element is used to communicate to the CDN to which duration the Expiry Time information element should be set whenever a new Signed Token is generated. This information element is only used in combination with the chained Signed Token mechanism for HTTP Adaptive Streaming content as described in Section 2.

3.3. 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.

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

o 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 or Signed Token when symmetric keys are used.

The Digital Signature attribute contains the digital signature used to verify the Signed URI or Signed Token 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.

3.4. 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. In the case of a Signed URI, the URI Signing Package Attribute is appended to the Original URI to create the Signed URI. In the case of a Signed Token, the URI Signing Package Attribute can be communicated via the query string component of a URI, via a dedicated HTTP header, or as an HTTP Cookie.

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. 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 Information Elementsin the Signed URI or the Signed Token.

- o 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 Information Element exposed in a Signed URI. The attribute MUST be the last parameter in the query string component 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. In the case of a Signed Token, the URI Signing Package Attribute fully constitutes the Signed Token.

The query string 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).

3.5. 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.

4. Generating a URI Signature

This section describes the process of generating a URI Signature. The nature of this process depends on whether a Signed URI or Signed Token is being generated. A Signed Token will typically be used as part of a chain of Signed Tokens for the delivery of HTTP Adaptive Streaming content (see Section 2). Section 4.1 defines creating a regular Signed URI. Section 4.2 defines creating the initial Signed Token for HAS content.

4.1. Creating a Signed URI

The following procedure defines the URI Signing algorithm for creating a Signed URI for regular (i.e. non-HTTP Adaptive Streaming) content in which the Path Pattern information element is not used. 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. In a Signed URI, the entire URI except the "scheme name" part is 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). The benefit is that the content access is protected regardless of the type of transport used for delivery. If the CSP wants to ensure a specific protocol is used for content delivery, that information is passed by CDNI metadata. Note: Support for changing of the URL scheme requires that the default port is used, or that the protocols must both run on the same non-standard port.

The process of generating a Signed URI can be divided into two sets of steps: first, calculating the URI Signature and then, packaging the URI Signature along with the URI Signing Information Elements into a URI Signing Package Attribute and appending it to the Original

URI. Note it is possible to use some other algorithm and implementation as long as the same result is achieved. An example for the Original URI, "http://example.com/content.mov", is used to clarify the steps.

4.1.1. Calculating the URI Signature (Signed URI)

Calculate the URI Signature for use with a Signed URI by following the procedure below.

1. Copy the Original URI, excluding the "scheme name" part, into a buffer to hold the message for performing the operations below.

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

3. If the version is the default value (i.e. "1"), skip this step. Otherwise, specify the version by appending the string "VER=#", where '#' represents the new version number. The following steps in the procedure is based on the initial version of URI Signing specified by this document. For other versions, reference the associated RFC for the URI signing procedure.

4. If time window enforcement is not needed, skip this step.

    A. If an information element was added to the message, append an "&" character. Append the string "ET=". Note in the case of re-signing a URI, the information element is 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 message.

5. If client IP enforcement is not needed, skip this step.

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

B. Convert the client's IP address in dotted decimal notation format (i.e. for IPv4 address) or canonical text representation (for IPv6 address [RFC5952]) to a string and append to the message. Note in the case of re-signing an

URI, the value MUST remain the same as the received Signed URI.

6. Depending on the type of key used to sign the URI, compute the message digest or digital signature for symmetric key or asymmetric keys, respectively.

A. For symmetric key, HMAC is used.

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

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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.

3. Optional: If the hash function for the HMAC uses the default value ("SHA-256"), skip this step. If an information element was added to the message, append an "&" character. append the string "HF=". Append the string for the new type of 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.

4. If an information element was added to the message, append an "&" character. Append the string "MD=". The message now contains the complete section of the URI that is protected (e.g. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=example:keys:123&MD=").

5. 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.

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

7. Append the string for the message digest (e.g. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=example:keys:123&MD=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bddaf").

B. For asymmetric keys, EC DSA is used.

1. Generate the EC private and public key pair. Store the EC public key in a location that's reachable for any entity that needs to validate the URI signature.

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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. "http://example.com/public/keys/123") needed by the entity to locate the shared key for validating the URI signature. Note that in the case the Key ID URI is a URL to a public key, the Key ID URI SHOULD only contain the "scheme name", "authority", and "path" parts (i.e. query string is not allowed).

3. Optional: If the digital signature algorithm uses the

default value ("EC-DSA"), skip this step. If an information element was added to the message, append an "&" character. Append the string "DSA=". Append the string denoting the new digital signature function.

4. If an information element was added to the message, append an "&" character. Append the string "DS=". The message now contains the complete section of the URI that is protected. (e.g. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=http://example.com/public/keys/123&DS=").

5. Compute the message digest using SHA-1 (without a key) for the message. Note: The digital signature generated in the next step is calculated over the SHA-1 message digest, instead of over the cleartype message, to reduce the length of the digital signature, and thereby the length of 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.

6. 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.

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

8. 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. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=http://example.com/public/keys/123&DS=r:CFB03EDB33810AB6C79EE3C47FBD86D227D702F25F66C01CF03F59F1E005668D:s:57ED0E8DF7E786C87E39177DD3398A7FB010E6A4C0DC8AA71331A929A29EA24E")

4.1.2. Packaging the URI Signature (Signed URI)

Apply the URI Signing Package Attribute by following the procedure below to generate the Signed URI.

1. Remove the Original URI portion from the message to obtain all the URI Signing Information Elements, including the URI signature (e.g. "ET=1209422976&CIP=192.0.2.1&KID=example:keys:123&MD=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bddaf").

2. Compute the URI Signing Package Attribute using Base-64 Data Encoding [RFC4648] on the message (e.g. "VkVSPTEmRVQ9MTIwOTQyMjk3NiZDSVA9MTkyLjAuMi4xJktJRD1leGFtcGxlOmtleXM6MTIzJk1EPTFlY2IxNDQ2YTY0MzEzNTJhYWIwZmI2ZTBkY2EzMGUzMDM1NjU5M2E5N2FjYjk3MjIwMjEyMGRjNDgyYmRkYWY="). Note: This is the value for the URI Signing Package Attribute.

3. Copy the entire Original URI into a buffer to hold the message.

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

5. 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.

6. Append the URI Signing token to the message (e.g. "http://example.com/content.mov?URISigningPackage=VkVSPTEmRVQ9MTIwOTQyMjk3NiZDSVA9MTkyLjAuMi4xJktJRD1leGFtcGxlOmtleXM6MTIzJk1EPTFlY2IxNDQ2YTY0MzEzNTJhYWIwZmI2ZTBkY2EzMGUzMDM1NjU5M2E5N2FjYjk3MjIwMjEyMGRjNDgyYmRkYWY="). Note: this is the completed Signed URI.

4.2. Creating an initial Signed Token

The following procedure defines the algorithm for creating the initial Signed Token of a Signed Token chain as for example used with HTTP Adaptive Streaming content. Note that the process described in this section is only performed for creating the initial Signed Token of a particular chain. Subsequent Signed Tokens forming the same chain are generated as part of the URI Signature Validation process described in Section 5. The creation of the initial Signed Token will typically be done by the CSP the first time a particular UA requests the manifest file. Choosing appropriate values of the Enforcement Information Elements in the initial Signed Token requires some knowledge of the structure of the HTTP Adapative Streaming content that is being requested.

In contrast with a Signed URI, where the URI Signature is calculated over the Original URI, a Signed Token does not protect the Original URI. Instead, the Path Pattern Information Element is used to convey the set of resources for which the particular Signed Token is valid.

The process of generating a initial Signed Token can be divided into two sets of steps: first, calculating the URI Signature and then, packaging the URI Signature along with the URI Signing Information Elements into a URI Signing Package to construct a Signed Token and appending the Signed Token to the message. Note it is possible to use some other algorithm and implementation as long as the same result is achieved. An example for the Original URI, "http://example.com/folder/content-83112371/manifest.xml", is used to clarify the steps.

4.2.1. Calculating the URI Signature (initial Signed Token)

Calculate the URI Signature for use with a Signed Token by following the procedure below.

1. Create a new buffer for constructing the Signed Token in the steps below.

2. If the URI Signing version used is the default one (i.e. "1"), skip this step. Otherwise, specify the version by appending the string "VER=#", where '#' represents the new version number. The following steps in the procedure is 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 not needed, this step can be skipped.

A. If an information element was added to the message, append an "&" character. Append the string "ET=".

B. Get the current time in seconds since epoch (as an integer). Add the validity time of the initial Signed Token in seconds as an integer.

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

D. Append an "&" character. Append the string "ETS=".

E. Append the Expiration Time Setting (in seconds) in the form of a string to the message. Note: the length of the Expiration Time Setting should be appropriate given the segment duration of the HTTP Adaptive Streaming content in question. As an example, if the segment duration is 10 seconds, the Expiration Time Setting should be at minimum 10 seconds, and preferably a bit more.

4. If client IP enforcement is not needed, this step can be skipped.

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

   B. Convert the client's IP address in dotted decimal notation format (i.e. for IPv4 address) or canonical text representation (for IPv6 address [RFC5952]) to a string and append to the message.

5. If an information element was added to the message, append an "&" character. Append the string "PP=".

6. Append the value of the Path Pattern in the form of a string to the message. Note: the value of the Path Pattern element should be appropriate given the file and folder structure of the HTTP Adaptive Streaming content in question. As an example, if the Original URI for the manifest file is 'http://example.com/folder/content-83112371/manifest.xml', a suitable Path Pattern might be '*/content-83112371/*/segment????.mp4'.

7. Depending on the type of key used to sign the URI, compute the message digest or digital signature for symmetric key or asymmetric keys, respectively.

   A. For symmetric key, HMAC is used.

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

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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.

3. Optional: If the hash function for the HMAC uses the default value ("SHA-256"), skip this step. If an information element was added to the message, append an "&" character. append the string "HF=". Append the string for the new type of hash function to be used. Note one MUST use the same hash function as communicated in the original concatenated information element string or one of the allowable hash functions designated by the CDNI metadata.

4. If an information element was added to the message, append an "&" character. Append the string "MD=". The message now contains the complete set of URI Signing Information Elements over which the URI Signature is computed (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&MD=").

5. 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.

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

7. Append the string for the message digest (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&MD=d6117d7db8a68bd59f6e7e3343484831acd8f23bbaa7f44b285a2f3bb6f02cfd").

B. For asymmetric keys, EC DSA is used.

1. Generate the EC private and public key pair. Store the EC public key in a location that's reachable for any entity that needs to validate the URI signature.

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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. "http://example.com/public/keys/123") needed by the entity to locate the shared key for validating the URI signature. Note that in the case the Key ID URI is a URL to a public key, the Key ID URI SHOULD only contain the "scheme name", "authority", and "path" parts (i.e. query string is not allowed).

3. Optional: If the digital signature algorithm uses the default value ("EC-DSA"), skip this step. If an information element was added to the message, append an "&" character. Append the string "DSA=". Append the string denoting the new digital signature function.

4. If an information element was added to the message, append an "&" character. Append the string "DS=". The message now contains the complete set of URI Signing Information Elements over which the URI Signature is computed (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&DS=").

5. Compute the message digest using SHA-1 (without a key) for the message. Note: The digital signature generated in the next step is calculated over the SHA-1 message digest, instead of over the cleartype message, to reduce the length of the digital signature, and thereby the length of the Signed Token. Since SHA-1 is not used for cryptographic purposes here, the security concerns around SHA-1 do not apply.

6. 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.

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

8. 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&ETS=15&CIP=192.0.2.1&PP=*/content-8
3112371/*/segment????.mp4&KID=example:keys:123&DS=r:CFB03

EDB33810AB6C79EE3C47FBD86D227D702F25F66C01CF03F59F1E00566
8D:s:57ED0E8DF7E786C87E39177DD3398A7FB010E6A4C0DC8AA71331
A929A29EA24E")

4.2.2. Packaging the URI Signature (initial Signed Token)

The following steps depend on whether the Signed Token is communicated to the UA via an HTTP 3xx Redirection message or via an HTTP 2xx Successful message. In the case of a redirection response, the Signed Token can be communicated as part of the query string component of the URL signalled via the Location header of the Redirection message. In the case of a 2xx Successful message, the Signed Token can be communicated via either a dedicated header or, for legacy UAs, via the Set-Cookie header.

4.2.2.1. Communicating the Signed Token in a HTTP 3xx Redirection message

The following steps describe how the Signed Token can be communicated to the UA via an HTTP 3xx Redirection message.

1. Copy the entire Original URI into a buffer to hold the message.

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

3. 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.

4. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMj
k3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQ
tODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6
MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmM
jNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

5. Append the URI Signing token to the message (e.g. "http://example.com/folder/content-83112371/manifest.xml?URISigningPackage=RVQ9MTIwOTQyMjk3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4
xJmFtcDtQUD0qL2NvbnRlbnQtODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1w
O0tJRD1leGFtcGxlOmtleXM6MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2Z
TdlMzM0MzQ4NDgzMWFjZDhmMjNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

6. Place the message in the Location header of the HTTP 3xx Redirection message returned to the UA.

4.2.2.2. Communicating the Signed Token in a HTTP 2xx Successful message

The following steps describe how the Signed Token can be communicated to the UA via an HTTP 2xx Successful message.

4.2.2.2.1. Header-based

The following steps are used in case the UA is expected to implement a mechanisms for extracting the Signed Token from the dedicated URI Signing HTTP Header and place in the query string component of the next request.

1. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMj
k3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQ
tODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6
MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmM

jNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

2. Place the value of the encoded Signed Token in the URI Signing Header of the HTTP 2xx Successful message of the content being returned to the UA. Note: The HTTP Header name to use is communicated via the CDNI Metadata Interface or set via configuration. Otherwise, it defaults to 'URISigningPackage'.

4.2.2.2.2. Cookie-based

The following steps are used in combination with legacy UAs that do not support a dedicated URI Signing HTTP header.

1. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMj
k3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQ
tODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6
MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmM
jNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

2. Add a 'URISigningPackage' cookie to the HTTP 2xx Successful message of the content being returned to the UA, with the value set to the encoded Signed Token.

5. Validating a URI Signature

The process of validating a Signed URI or Signed Token can be divided into three sets of steps: first, extraction of the URI Signing information elements, then validation of the URI signature to ensure the integrity of the Signed URI or Signed Token, and finally, validation of the information elements to ensure proper enforcement of the distribution policy. In case the chained Signed Token mechanism for HTTP Adaptive Streaming, as defined in Section 2, is used, a fourth step for constructing and communicating the next Signed Token, is added. Note the first three steps in the algorithm described below apply irrespective of whether a Signed URI or Signed Token is received.

In the algorithm below, the integrity of the Signed URI or Signed Token is confirmed before distribution policy enforcement because validation procedure would detect the right event when the URI is tampered with. Note it is possible to use some other algorithm and implementation as long as the same result is achieved.

5.1. Information Element Extraction

Extract the information elements embedded in the Signed URI or Signed Token. Note that some steps are to be skipped if the corresponding URI Signing Information Elements are not embedded in the Signed URI or Signed Token.

1. Check if the query string component of the received URI contains the 'URISigningPackage' 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 URI Signing Package attribute. If the query string component of the received URI does not contain the URI Signing Package attribute, check if the HTTP request contains a 'URISigningPackage' cookie and use that as the URI Signing Package in the following steps. If the request does not contain the URI Signing Package query string attribute and does not contain a URISigningPackage cookie, the request is denied.

2. Decode the URI Signing Package using Base-64 Data Encoding [RFC4648] to obtain all the URI Signing Information Elements in the form of a concatenated string (e.g. "ET=1209422976&CIP=192.0.2.1&KID=example:keys:123&MD=1ecb1446a6431352aab0fb6e0dca30e30356593a97acb972202120dc482bddaf").

3. Extract the value from "VER", if the information element exists. Determine the version of the URI Signing algorithm used to

process the Signed URI or Signed Token. 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 information elements string, then obtain the version number in another manner (e.g. configuration, CDNI metadata or default value).

4. Extract the value from "MD", if the information element exists. The existence of this information element indicates a symmetric key is used.

5. Extract the value from "DS", if the information element exists. The existence of this information element indicates an asymmetric key is used.

6. If neither the "MD" or "DS" attribute exists, 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.

7. Extract the value from "CIP", if the information element exists. The existence of this information element indicates content delivery is enforced based on client IP address.

8. Extract the value from "ET", if the information element exists. The existence of this information element indicates content delivery is enforced based on time.

9. Extract the value from "PP", if the information element exists. The existence of this information element indicates content delivery is enforced based on whether there is a match between the path of the requested resource and the Path Pattern information element. The existence of this element further indicates that a chain of Signed Tokens is used, and a new Signed Token should be generated and communicated upon successful validation.

10. 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.

11. 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.

12. 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.

13. Extract the value from "USCF", if the information element exists. The existence of this information element indicates cookie-based communication for legacy UAs should be used for signalling the next Signed Token in case a HTTP 2xx Succcessful message is sent to the user.

14. Extract the value from "ETS", if the information element exists. This information element indicates the validity time of the next Signed Token in the chain.

5.2. Signature Validation

Validate the URI Signature for the Signed URI or Signed Token.

1. Create a new buffer for validating the Signed URI or Signed Token in the steps below.

2. If the received URI Signing Package contains the Path Pattern Information Element, this step can be skipped.

    A. Copy the received URI, excluding the "scheme name" part, into the buffer.

    B. Remove the "URISigningPackage" attribute from the message, if it exists. Remove any subsequent part of the query string after the "URISigningPackage" attribute.

    C. Append the decoded value from the "URISigningPackage" attribute (which contains all the URI Signing Information Elements).

3. If the received URI Signing Package does NOT contain the Path Pattern Information Element, this step can be skipped. Copy the decoded contents of the Signed Token in the buffer.

4. Depending on the type of key used to create the Signed URI or Signed Token, validate the message digest or digital signature for symmetric key or asymmetric keys, respectively.

    A. For symmetric key, HMAC algorithm is used.

a. 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 received URI Signing Package, obtain the shared key via CDNI metadata or configuration.

b. If the "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. If the "HF" information element is not in the received URI Signing Package, the default hash function is SHA-256.

c. Extract the value from the "MD" information element. This is the received message digest.

d. Convert the message digest to binary format. This will be used to compare with the computed value later.

e. Remove the value part of the "MD" information element (but not the '=' character) from the message. The message is ready for validation of the message digest (e.g. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=example:keys:123&MD=").

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

g. Compare the result with the received message digest to validate the Signed URI or Signed Token. If there is no match, the request is denied.

B. For asymmetric keys, a digital signature function is used.

a. 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 received URI Signing Package, obtain the public key via CDNI metadata or configuration.

b. If the "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. If the "DSA" information element is not in the received URI Signing Package, the default DSA is EC-DSA.

c. Extract the value from the "DS" information element. This is the received digital signature.

d. Convert the digital signature to binary format. This will be used for verification later.

e. Remove the value part of the "DS" information element (but not the '=' character) from the message. The message is ready for validation of the digital signature (e.g. "://example.com/content.mov?ET=1209422976&CIP=192.0.2.1&KID=http://example.com/public/keys/123&DS=").

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

g. 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 or Signed Token. If signature is determined to be invalid, the request is denied.

5.3. Distribution Policy Enforcement

Note the associated steps are to be skipped if the corresponding URI Signing Information Elements are not in the received URI Signing Package. The absence of a given Enforcement Information Element indicates enforcement of its purpose is not necessary in the CSP's distribution policy. The exception is the Path Pattern Information

Element, which is mandatory for Signed Tokens.

1. If the "CIP" information element exists, validate that the request came from the same IP address as indicated in the "CIP" 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, the request is denied.

3. If the "PP" information element exists, validate that the requested resource is in the allowed set by matching the received URI against the Path Pattern information element. If there is no match, the request is denied.

5.4. Subsequent Signed Token Generation

The following steps describe how to generate a subsequent Signed Token in a chain of Signed Tokens and are to be skipped in case the chained Signed Token mechanism for HTTP Adaptive Streaming content is not used. Note that the process for generating an initial Signed Token is described in Section 4.2 and the process below is used for generating all subsequent tokens after the initial one.

The process of generating a subsequent Signed Token can be divided into two sets of steps: first, calculating the URI Signature and then, packaging the URI Signature along with the URI Signing Information Elements into a URI Signing Package to construct a new Signed Token and appending the Signed Token to the message. Note it is possible to use some other algorithm and implementation as long as the same result is achieved.

5.4.1. Calculating the URI Signature (subsequent Signed Token)

Calculate the URI Signature for use with the new Signed Token by following the procedure below.

1. Create a new buffer for constructing the new Signed Token in the steps below.

2. If the URI Signing version used in the received URI Signing Package is the default one (i.e. "1"), skip this step. Otherwise, specify the version by appending the string "VER=#", where '#' represents the version number. The following steps in the procedure is 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 the received URI Signing Package does not contain the "ET" information element, skip this step.

    A. If an information element was added to the message, append an "&" character. Append the string "ET=".

    B. If the received URI Signing Package contains the "ETS" information element, perform this step.

        1. Get the value of the "ETS" information element and convert it to an integer.

        2. Get the current time in seconds sinds epoch (as an integer) and add the value of the "ETS" information element as seconds.

        3. Convert the result to a string and append it to the message.

        4. Append the "&" character and the "ETS=" string.

        5. Append the value of the "ETS" information element in the received URI Signing Package.

    C. If the received URI Signing Package does not contain the "ETS" information element, perform this step. Get the value of the "ET" information element from the original concatenated information element string and append it to the message.

4. If the received URI Signing Package does not contain the "CIP"

information element, skip this step.

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

B. Append the value of the "CIP" information element in the received URI Signing Package.

5. If an information element was added to the message, append an "&" character. Append the string "PP=". Append the value of the "PP" information element in the received URI Signing Package.

6. Depending on the type of key used to sign the received Signed Token, compute the message digest or digital signature for symmetric key or asymmetric keys, respectively.

A. For symmetric key, HMAC is used.

1. Obtain the shared key to be used for signing the Signed Token.

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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.

3. Optional: If the hash function for the HMAC uses the default value ("SHA-256"), skip this step. If an information element was added to the message, append an "&" character. append the string "HF=". Append the string for the new type of hash function to be used. Note one MUST use the same hash function as communicated in the received URI Signing Package or one of the allowable hash functions designated by the CDNI metadata.

4. If an information element was added to the message, append an "&" character. Append the string "MD=". The

message now contains the complete set of URI Signing Information Elements over which the URI Signature is computed (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&MD=").

5. 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.

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

7. Append the string for the message digest (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&MD=d6117d7db8a68bd59f6e7e3343484831acd8f23bbaa7f44b285a2f3bb6f02cfd").

B. For asymmetric keys, EC DSA is used.

1. Generate the EC private and public key pair. Store the EC public key in a location that's reachable for any entity that needs to validate the URI signature.

2. If the key identifier is not needed, skip this step. If an information element was added to the message, 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. "http://example.com/public/keys/123") needed by the entity to locate the shared key for validating the URI

signature. Note that in the case the Key ID URI is a URL to a public key, the Key ID URI SHOULD only contain the "scheme name", "authority", and "path" parts (i.e. query string is not allowed).

3. Optional: If the digital signature algorithm uses the default value ("EC-DSA"), skip this step. If an information element was added to the message, append an

"&" character. Append the string "DSA=". Append the string denoting the new digital signature function.

4. If an information element was added to the message, append an "&" character. Append the string "DS=". The message now contains the complete set of URI Signing Information Elements over which the URI Signature is computed (e.g. "ET=1209422976&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&DS=").

5. Compute the message digest using SHA-1 (without a key) for the message. Note: The digital signature generated in the next step is calculated over the SHA-1 message digest, instead of over the cleartype message, to reduce the length of the digital signature, and thereby the length of 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.

6. 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.

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

8. 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&ETS=15&CIP=192.0.2.1&PP=*/content-83112371/*/segment????.mp4&KID=example:keys:123&DS=r:CFB03EDB33810AB6C79EE3C47FBD86D227D702F25F66C01CF03F59F1E005668D:s:57ED0E8DF7E786C87E39177DD3398A7FB010E6A4C0DC8AA71331A929A29EA24E")

5.4.2. Packaging the URI Signature (subsequent Signed Token)

The following steps depend on whether the Signed Token is communicated to the UA via an HTTP 3xx Redirection message or via an HTTP 2xx Successful message. In the case of a redirection response, the Signed Token can be communicated as part of the query string component of the URL signalled via the Location header of the Redirection message. In the case of a 2xx Successful message, the Signed Token can be communicated via either a dedicated HTTP header or, for legacy UAs, via the Set-Cookie header. If the received URI Signing Package contains the 'USCF' Information Element, the new Signed Token MUST be communicated via the Cookie method. If the received URI Signing Package does NOT contain the 'USCF' Information Element, the new Signed Token SHALL be communicated via the dedicated HTTP header.

5.4.2.1. Communicating the Signed Token in a HTTP 3xx Redirection message

The following steps describe how the new Signed Token can be communicated to the UA via an HTTP 3xx Redirection message.

1. Copy the target URI of the HTTP 3xx Redirection message into a buffer to hold the message.

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

3. 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.

4. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMjk3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQtODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmMjNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

5. Append the URI Signing token to the message (e.g. "http://example.com/folder/content-83112371/manifest.xml?URISigningPackage=RVQ9MTIwOTQyMjk3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQtODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmMjNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

6. Place the message in the Location header of the HTTP 3xx Redirection message returned to the UA.

5.4.2.2. Communicating the Signed Token in a HTTP 2xx Successful message

The following steps describe how the new Signed Token can be communicated to the UA via an HTTP 2xx Successful message.

5.4.2.2.1. Header-based

If the received URI Signing Package does NOT contain the 'USCF' Information Element, the new Signed Token SHALL be communicated via the following method.

1. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMjk3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQtODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmMjNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

2. Place the value of the encoded Signed Token in the URI Signing Header of the HTTP 2xx Successful message of the content being returned to the UA. Note: The HTTP Header name to use is communicated via the CDNI Metadata Interface or set via configuration. Otherwise, it defaults to 'URISigningPackage'.

5.4.2.2.2. Cookie-based

If the received URI Signing Package contains the 'USCF' Information Element, the new Signed Token MUST be communicated via the following method.

1. Encode the Signed Token by applying Base-64 Data Encoding [RFC4648] on the value of the Signed Token (e.g. "RVQ9MTIwOTQyMjk3NiZhbXA7RVRTPTE1JmFtcDtDSVA9MTkyLjAuMi4xJmFtcDtQUD0qL2NvbnRlbnQtODMxMTIzNzEvKi9zZWdtZW50Pz8/Py5tcDQmYW1wO0tJRD1leGFtcGxlOmtleXM6MTIzJmFtcDtNRD1kNjExN2Q3ZGI4YTY4YmQ1OWY2ZTdlMzM0MzQ4NDgzMWFjZDhmMjNiYmFhN2Y0NGIyODVhMmYzYmI2ZjAyY2Zk").

2. Add a 'URISigningPackage' cookie to the HTTP 2xx Successful message of the content being returned to the UA, with the value set to the encoded Signed Token.

6. 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?).

6.1. CDNI Control Interface

URI Signing has no impact on this interface.

6.2. 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. ForInformation Elements,

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. EC-DSA) to indicate support for the DSA; Need IANA assignment for new digital signature algorithm.

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

Advertise "DS" Information Element (i.e. EC-DSA) 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.

6.3. 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

6.4. 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 EC-DSA.

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.

6.5. 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].

- o 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:

    - + "0" : no uri signature validation performed

    - + "1" : uri signature validation performed and validated

    - + "2" : uri signature validation performed and rejected

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

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

  - * format: QSTRING

  - * field value: the rejection reason when uri signature performed by the Surrogate on the request. Examples:

+ "invalid client IP address"

+ "expired signed URI"

+ "incorrect URI signature"

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

7. 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.

7.1. 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  <Positive>|                    |
        |  dCDN (newly signed URI)                |                    |
        |<----------------------------------------|                    |
        |                    |                    |                    |
        |10.Content request  |                    |                    |
        |------------------->| [Validate URI      |                    |
        |                    |  signature]        |                    |
        |                    |                    |                    |
        |    (ALT: Validation result)             |                    |
        |11.Request is denied| <Negative>         |                    |
        |<-------------------|                    |                    |
        |                    |                    |                    |
        |12.Content delivery | <Positive>         |                    |
        |<-------------------|                    |                    |
        :                    :                    :                    :
        :   (Later in time)  :                    :                    :
        |13.CDNI Logging interface to include URI Signing information  |
        |                    |------------------->|                    |
```

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.

7.2. 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.

```
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.Provides signed URI                    |      <Positive>    |
        |<-------------------------------------------------------------|
        |                    |                    |                    |
```



```
        |7.DNS request       |                    |                    |
        |---------------------------------------->|                    |
        |                    |                    |                    |
        |8.Redirect DNS to dCDN                   |                    |
        |<----------------------------------------|                    |
        |                    |                    |                    |
        |9.DNS request       |                    |                    |
        |------------------->|                    |                    |
        |                    |                    |                    |
        |10.IP address of Surrogate               |                    |
        |<-------------------|                    |                    |
        |                    |                    |                    |
        |11.Content request  |                    |                    |
        |------------------->| [Validate URI      |                    |
        |                    |  signature]        |                    |
        |                    |                    |                    |
        |    (ALT: Validation result)             |                    |
        |12.Request is denied| <Negative>         |                    |
        |<-------------------|                    |                    |
        |                    |                    |                    |
        |13.Content delivery | <Positive>         |                    |
        |<-------------------|                    |                    |
        :                    :                    :                    :
        :   (Later in time)  :                    :                    :
        |14.CDNI Logging interface to report URI Signing information   |
        |                    |------------------->|                    |
```

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.

8. IANA Considerations

[Editor's note: (Is there a need to) register default value for URI Signing Package Attribute URI query string parameter name (i.e. URISigningPackage) to be used for URI Signing? Need anything from IANA?]

[Editor's note: To do: Convert to proper IANA Registry format]

This document requests IANA to create three new URI Signing registries for the Information Elements and their defined values to be used for URI Signing.

The following Enforcement Information Element names are allocated:

o ET (Expiry time)

o CIP (Client IP address)

The following Signature Computation Information Element names are allocated:

o VER (Version): 1 (Base)

o KID (Key ID)

o KID_NUM (Numerical Key ID)

o HF (Hash Function): "SHA-256"

o DSA (Digital Signature Algorithm): "EC-DSA"

The following URI Signature Information Element names are allocated:

o MD (Message Digest for Symmetric Key)

o DS (Digital Signature for Asymmetric Keys)

The IANA is requested to allocate a new entry to the CDNI Logging Field Names Registry as specified in CDNI Logging Interface [I-D.ietf-cdni-logging] in accordance to the "Specification Required" policy [RFC5226]

o s-url-signing

o s-url-signing-deny-reason

The IANA is requested to allocate a new entry to the "CDNI GenericMetadata Types" Registry as specified in CDNI Metadata Interface [I-D.ietf-cdni-metadata] in accordance to the "Specification Required" policy [RFC5226]:

Type name	Specification	Version	MTE	STR
UriSigning	RFCthis	1	true	true

The IANA is also requested to allocate a new MIME type under the IANA MIME Media Type registry for the UriSigning metadata object:

application/cdni.UriSigning.v1

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. This means that, when anonymization is enabled, the value of the URI Signing Package Attribute MUST be removed from the logging record.

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 and Leif Hedstrom. 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-18 (work in progress), March 2015.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, September 2012.

12.2. Informative References

[I-D.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "CDN Interconnection Metadata", draft-ietf-cdni-metadata-09 (work in progress), March 2015.

[I-D.ietf-cdni-redirection]
Niven-Jenkins, B. and R. Brandenburg, "Request Routing Redirection Interface for CDN Interconnection", draft-ietf-cdni-redirection-09 (work in progress), April 2015.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[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, July 2013.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, August 2014.

[RFC7337] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, August 2014.

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