           URI Signing for Content Delivery Network Interconnection (CDNI)

Abstract

   This document describes how the concept of URI Signing supports the
   content access control requirements of Content Delivery Network
   Interconnection (CDNI) and proposes a URI Signing method as a JSON
   Web Token (JWT) profile.

   The proposed URI Signing method specifies the information needed to
   be included in the URI to transmit the signed JWT, as well as the
   claims needed by the signed JWT to authorize a User Agent (UA). The
   mechanism described can be used both in CDNI and single Content
   Delivery Network (CDN) scenarios.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc9246.

Table of Contents

1. Introduction

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

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

    Specifically, the CDNI Framework [RFC7336] states:

       The CSP may also trust the CDN operator to perform actions such as
       delegating traffic to additional downstream CDNs, and to enforce
       per-request authorization performed by the CSP using techniques
       such as URI Signing.

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

       | MI-16 {HIGH} The CDNI Metadata 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 the need to validate
          |     information (e.g., Expiry time, Client IP address) required
          |     for access authorization.
```

This document defines a method of signing URIs that allows Surrogates
in interconnected CDNs to enforce a per-request authorization
initiated by the CSP. Splitting the role of initiating per-request
authorization by the CSP and the role of verifying 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
specific CSP distribution policies.

The method is implemented using signed JSON Web Tokens (JWTs)
[RFC7519].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

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

This document also uses the terminology of the JSON Web Token (JWT)
[RFC7519].

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

FCI: Footprint & Capabilities Advertisement interface

Signed URI: A URI for which a signed JWT is provided.

Target CDN URI: A URI created by the CSP to direct a UA towards the
   upstream CDN (uCDN). The Target CDN URI can be signed by the CSP
   and verified by the uCDN and possibly further downstream CDNs
   (dCDNs).

Redirection URI: A URI created by the uCDN to redirect a UA towards
   the dCDN. The Redirection URI can be signed by the uCDN and
   verified by the dCDN. In a cascaded CDNI scenario, there can be
   more than one Redirection URI.

Signed Token Renewal: A series of signed JWTs that are used for
   subsequent access to a set of related resources in a CDN, such as
   a set of HTTP Adaptive Streaming files. Every time a signed JWT
   is used to access a particular resource, a new signed JWT is sent

along with the resource that can be used to request the next
resource in the set. When generating a new signed JWT in Signed
Token Renewal, parameters are carried over from one signed JWT to
the next.

1.2. Background and Overview on URI Signing

   A CSP and CDN are assumed to have a trust relationship that enables
   the CSP to authorize access to a content item, which is realized in
   practice by including a set of claims in a signed JWT 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 a time window or
   pattern matching the URI). To prevent the UA from altering the
   claims, the JWT MUST be signed.

   Figure 1 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.
   The Signed URI always contains a signed JWT generated by the CSP
   using a shared secret or private key. Once the UA receives the
   response with the Signed URI, it sends a new HTTP request using the
   Signed URI to the CDN (3). Upon receiving the request, the CDN
   authenticates the Signed URI by verifying the signed JWT. If
   applicable, the CDN checks whether the time window is still valid in
   the Signed URI and the pattern matches the URI of the request. After
   these claims are verified, the CDN delivers the content (4).

   Note: While using a symmetric shared key is supported, it is NOT
   RECOMMENDED. See the Security Considerations (Section 7) about the
   limitations of shared keys.

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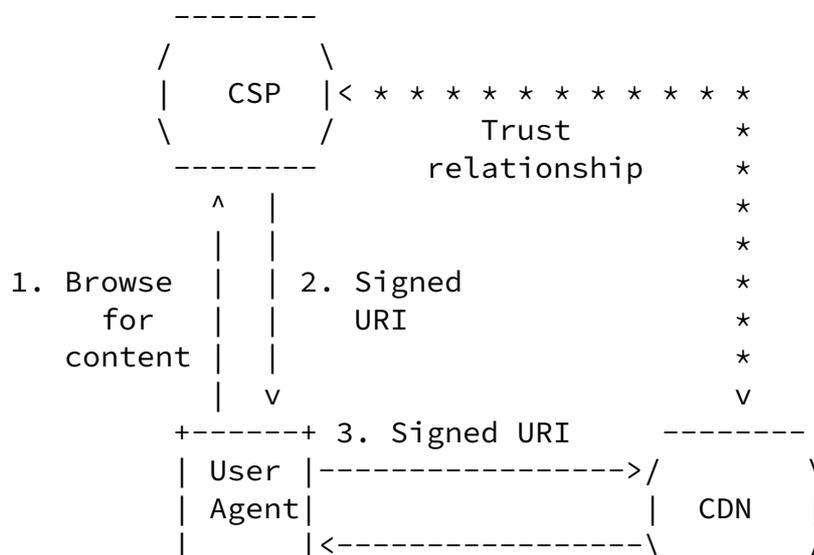
```
               +------+ 4. Content         --------
                        Delivery

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

1.3. CDNI URI Signing Overview

In a CDNI environment, as shown in Figure 2 below, URI Signing
operates the same way in the initial steps 1 and 2, but the later
steps involve multiple CDNs delivering the content. The main
difference from the single CDN case is a redirection step between the
uCDN and the dCDN. In step 3, the UA may send an HTTP request or a
DNS request, depending on whether HTTP-based or DNS-based request
routing is used. The uCDN responds by directing the UA towards the
dCDN using either a Redirection URI (i.e., a Signed URI generated by
the uCDN) or a DNS reply, respectively (4). Once the UA receives the
response, it sends the Redirection URI/Target CDN URI to the dCDN
(5). The received URI is verified by the dCDN before delivering the
content (6).

Note: The CDNI call flows are covered in URI Signing Message Flow
(Section 5).

```
                                        +-------------------------+
                                        |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]
                                     --------
```

Figure 2: URI Signing in a CDNI Environment

The trust relationships between CSP, uCDN, and dCDN have direct
implications for URI Signing. In the case shown in Figure 2, the CSP
has a trust relationship with the uCDN. The delivery of the content
may be delegated to a dCDN, which has a relationship with the uCDN
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., the
Target CDN URI) provided by the CSP reaches the CDN directly. In the
case where the dCDN does not have a trust relationship with the CSP,
this means that either an asymmetric public/private key method needs
to be used for computing the signed JWT (because the CSP and dCDN are
not able to exchange symmetric shared secret keys). Shared keys MUST
NOT be redistributed.

For HTTP-based request routing, the Signed URI (i.e., the Target CDN
URI) provided by the CSP reaches the uCDN. After this URI has been
verified by the uCDN, the uCDN creates and signs a new Redirection
URI, redirecting the UA to the dCDN. Since this new URI can have a
new signed JWT, the relationship between the dCDN and CSP is not
relevant. Because a relationship between uCDN and dCDN always
exists, either asymmetric public/private keys or symmetric shared
secret keys can be used for URI Signing with HTTP-based request
routing. Note that the signed Redirection URI MUST maintain HTTPS as
the scheme if it was present in the original, and it MAY be upgraded
from "http:" to "https:".

Two types of keys can be used for URI Signing: asymmetric keys and
symmetric shared keys. Asymmetric keys are based on a public/private
key pair mechanism and always contain a private key known only 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 and the verifying
entity for verifying the Signed URI. Regardless of the type of keys
used, the verifying entity has to obtain the key in a manner that
allows trust to be placed in the assertions made using that key
(either the public or the symmetric key). There are very different
requirements (outside the scope of this document) for distributing
asymmetric keys and symmetric keys. Key distribution for symmetric
keys requires confidentiality to prevent third parties from getting
access to the key, since they 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 to sign URIs) and the corresponding private keys are kept secret by the URI signer.

Note: While using a symmetric shared key is supported, it is NOT RECOMMENDED. See the Security Considerations (Section 7) about the limitations of shared keys.
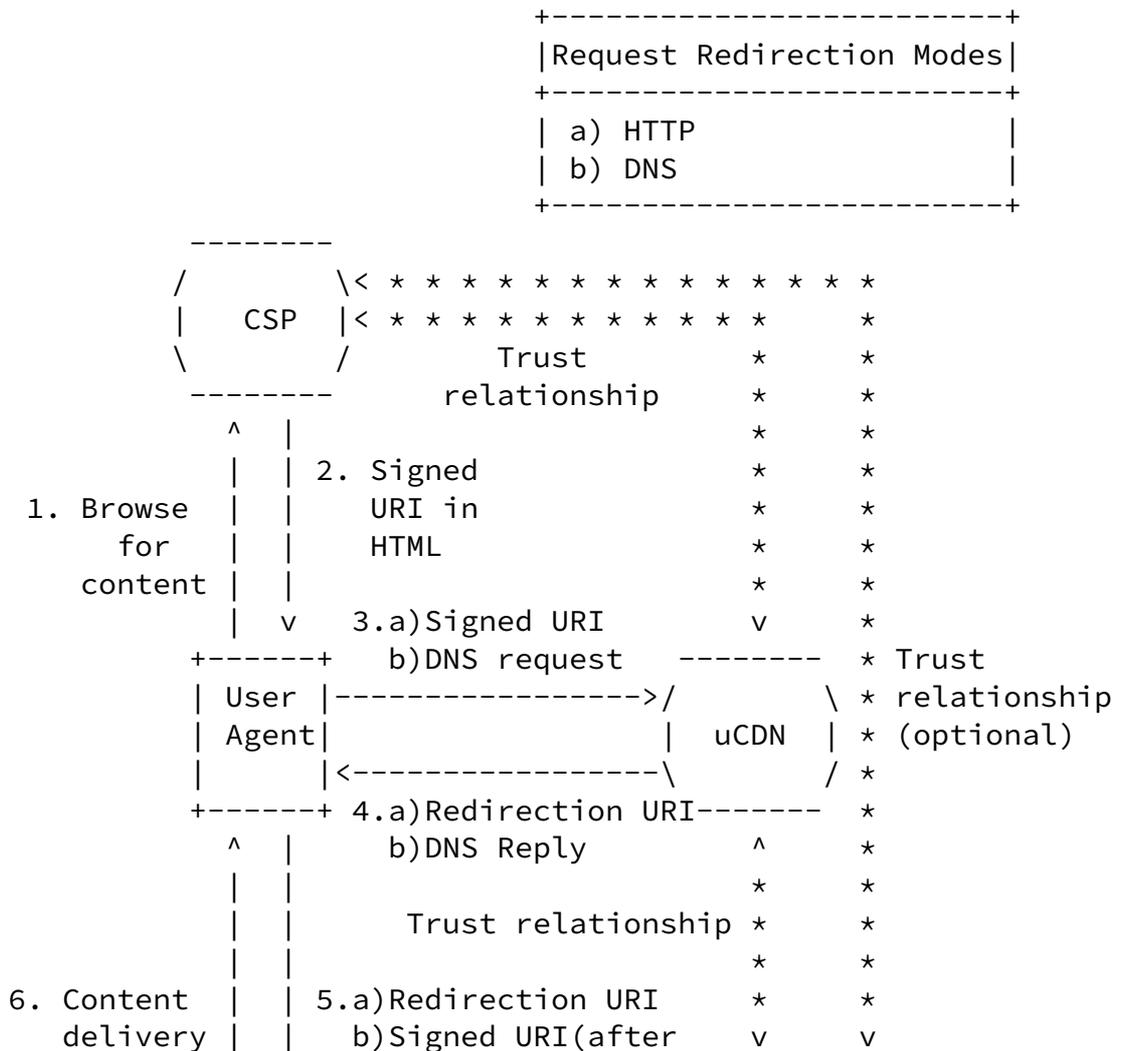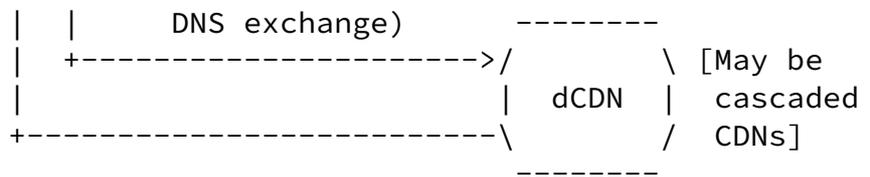
1.4. URI Signing in a Non-CDNI Context

While the URI Signing method defined in this document was primarily created for the purpose of allowing URI Signing in CDNI scenarios, i.e., between a uCDN and a dCDN, there is nothing in the defined URI Signing method 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. JWT Format and Processing Requirements

The concept behind URI Signing is based on embedding a signed JSON Web Token (JWT) [RFC7519] in an HTTP or HTTPS URI [RFC7230] (see Section 2.7 of [RFC7230]). The signed JWT contains a number of claims that can be verified to ensure the UA has legitimate access to the content.

This document specifies the following attribute for embedding a signed JWT in a Target CDN URI or Redirection URI:

URI Signing Package (URISigningPackage): The URI attribute that encapsulates all the URI Signing claims in a signed JWT encoded format. This attribute is exposed in the Signed URI as a path-style parameter or a form-style parameter.

The parameter name of the URI Signing Package Attribute is defined in the CDNI Metadata (Section 4.4). If the CDNI Metadata interface is not used, or does not include a parameter name for the URI Signing Package Attribute, the parameter name can be set by configuration (out of scope of this document).

The URI Signing Package will be found by parsing any path-style parameters and form-style parameters looking for a key name matching the URI Signing Package Attribute. Both parameter styles MUST be supported to allow flexibility of operation. The first matching parameter SHOULD be taken to provide the signed JWT, though providing more than one matching key is undefined behavior. Path-style parameters generated in the form indicated by Section 3.2.7 of [RFC6570] and Form-style parameters generated in the form indicated by Sections 3.2.8 and 3.2.9 of [RFC6570] MUST be supported.

The following is an example where the URI Signing Package Attribute

name is "token" and the signed JWT is "SIGNEDJWT":

http://example.com/media/path?come=data&token=SIGNEDJWT&other=data

2.1. JWT Claims

This section identifies the set of claims that can be used to enforce
the CSP distribution policy. New claims can be introduced in the
future to extend the distribution policy capabilities.

In order to provide distribution policy flexibility, the exact subset
of claims used in a given signed JWT is a runtime decision. Claim
requirements are defined in the CDNI Metadata (Section 4.4). If the
CDNI Metadata interface is not used, or does not include claim
requirements, the claim requirements can be set by configuration (out
of scope of this document).

The following claims (where the "JSON Web Token Claims" registry
claim name is specified in parentheses below) are used to enforce the
distribution policies. All of the listed claims are mandatory to
implement in a URI Signing implementation but are not necessarily
mandatory to use in a given signed JWT. (The "optional" and
"mandatory" identifiers in square brackets refer to whether or not a
given claim MUST be present in a URI Signing JWT.)

Note: The time on the entities that generate and verify the Signed
URI MUST be in sync. In the CDNI case, this means that CSP, uCDN,
and dCDN servers need to be time synchronized. It is RECOMMENDED to
use NTP [RFC5905] for time synchronization.

Note: See the Security Considerations (Section 7) on the limitations
of using an expiration time and Client IP address for distribution
policy enforcement.

2.1.1. Issuer (iss) Claim

Issuer (iss) [optional] - The semantics in Section 4.1.1 of [RFC7519]
MUST be followed. If this claim is used, it MUST be used to identify
the Issuer (signer) of the JWT. In particular, the recipient will
have already received, in trusted configuration, a mapping of Issuer
name to one or more keys used to sign JWTs and must verify that the
JWT was signed by one of those keys. If this claim is used and the
CDN verifying the signed JWT does not support Issuer verification, or
if the Issuer in the signed JWT does not match the list of known
acceptable Issuers, or if the Issuer claim does not match the key
used to sign the JWT, the CDN MUST reject the request. If the
received signed JWT contains an Issuer claim, then any JWT
subsequently generated for CDNI redirection MUST also contain an
Issuer claim, and the Issuer value MUST be updated to identify the
redirecting CDN. If the received signed JWT does not contain an
Issuer claim, an Issuer claim MAY be added to a signed JWT generated

for CDNI redirection.

2.1.2. Subject (sub) Claim

   Subject (sub) [optional] - The semantics in Section 4.1.2 of
   [RFC7519] MUST be followed. If this claim is used, it MUST be a JSON
   Web Encryption (JWE [RFC7516]) Object in compact serialization form,
   because it contains personally identifiable information. This claim
   contains information about the Subject (for example, a user or an
   agent) that MAY be used to verify the signed JWT. If the received
   signed JWT contains a Subject claim, then any JWT subsequently
   generated for CDNI redirection MUST also contain a Subject claim, and
   the Subject value MUST be the same as in the received signed JWT. A
   signed JWT generated for CDNI redirection MUST NOT add a Subject
   claim if no Subject claim existed in the received signed JWT.

2.1.3. Audience (aud) Claim

   Audience (aud) [optional] - The semantics in Section 4.1.3 of
   [RFC7519] MUST be followed. This claim is used to ensure that the
   CDN verifying the JWT is an intended recipient of the request. The
   claim MUST contain an identity belonging to the chain of entities
   involved in processing the request (e.g., identifying the CSP or any
   CDN in the chain) that the recipient is configured to use for the
   processing of this request. A CDN MAY modify the claim as long it
   can generate a valid signature.

2.1.4. Expiry Time (exp) Claim

   Expiry Time (exp) [optional] - The semantics in Section 4.1.4 of
   [RFC7519] MUST be followed, though URI Signing implementations MUST
   NOT allow for any time-synchronization "leeway". If this claim is
   used and the CDN verifying the signed JWT does not support Expiry
   Time verification, or if the Expiry Time in the signed JWT
   corresponds to a time equal to or earlier than the time of the
   content request, the CDN MUST reject the request. If the received
   signed JWT contains an Expiry Time claim, then any JWT subsequently
   generated for CDNI redirection MUST also contain an Expiry Time
   claim, and the Expiry Time value MUST be the same as in the received
   signed JWT. A signed JWT generated for CDNI redirection MUST NOT add
   an Expiry Time claim if no Expiry Time claim existed in the received
   signed JWT.

2.1.5. Not Before (nbf) Claim

   Not Before (nbf) [optional] - The semantics in Section 4.1.5 of
   [RFC7519] MUST be followed, though URI Signing implementations MUST
   NOT allow for any time-synchronization "leeway". If this claim is
   used and the CDN verifying the signed JWT does not support Not Before
   time verification, or if the Not Before time in the signed JWT
   corresponds to a time later than the time of the content request, the

CDN MUST reject the request. If the received signed JWT contains a
Not Before time claim, then any JWT subsequently generated for CDNI
redirection MUST also contain a Not Before time claim, and the Not
Before time value MUST be the same as in the received signed JWT. A
signed JWT generated for CDNI redirection MUST NOT add a Not Before
time claim if no Not Before time claim existed in the received signed
JWT.

2.1.6. Issued At (iat) Claim

Issued At (iat) [optional] - The semantics in Section 4.1.6 of
 [RFC7519] MUST be followed. If the received signed JWT contains an
Issued At claim, then any JWT subsequently generated for CDNI
redirection MUST also contain an Issued At claim, and the Issued At
value MUST be updated to identify the time the new JWT was generated.
If the received signed JWT does not contain an Issued At claim, an
Issued At claim MAY be added to a signed JWT generated for CDNI
redirection.

2.1.7. JWT ID (jti) Claim

JWT ID (jti) [optional] - The semantics in Section 4.1.7 of [RFC7519]
MUST be followed. A JWT ID can be used to prevent replay attacks if
the CDN stores a list of all previously used values and verifies that
the value in the current JWT has never been used before. If the
signed JWT contains a JWT ID claim and the CDN verifying the signed
JWT either does not support JWT ID storage or has previously seen the
value used in a request for the same content, then the CDN MUST
reject the request. If the received signed JWT contains a JWT ID
claim, then any JWT subsequently generated for CDNI redirection MUST
also contain a JWT ID claim, and the value MUST be the same as in the
received signed JWT. If the received signed JWT does not contain a
JWT ID claim, a JWT ID claim MUST NOT be added to a signed JWT
generated for CDNI redirection. Sizing of the JWT ID is application
dependent given the desired security constraints.

2.1.8. CDNI Claim Set Version (cdniv) Claim

CDNI Claim Set Version (cdniv) [optional] - The CDNI Claim Set
Version (cdniv) claim provides a means within a signed JWT to tie the
claim set to a specific version of this specification. The cdniv
claim is intended to allow changes in and facilitate upgrades across
specifications. The type is a JSON integer and the value MUST be set
to "1" for this version of the specification. In the absence of this
claim, the value is assumed to be "1". For future versions, this
claim will be mandatory. Implementations MUST reject signed JWTs
with unsupported CDNI Claim Set versions.

2.1.9. CDNI Critical Claims Set (cdnicrit) Claim

CDNI Critical Claims Set (cdnicrit) [optional] - The CDNI Critical

Claims Set (cdnicrit) claim indicates that extensions to this
specification are being used that MUST be understood and processed.
Its value is a comma-separated listing of claims in the Signed JWT
that use those extensions. If any of the listed extension claims are
not understood and supported by the recipient, then the Signed JWT
MUST be rejected. Producers MUST NOT include claim names defined by
this specification, duplicate names, or names that do not occur as
claim names within the Signed JWT in the cdnicrit list. Producers
MUST NOT use the empty list "" as the cdnicrit value. Recipients MAY
consider the Signed JWT to be invalid if the cdnicrit list contains
any claim names defined by this specification or if any other
constraints on its use are violated. This claim MUST be understood
and processed by all implementations.

2.1.10. Client IP Address (cdniip) Claim

Client IP Address (cdniip) [optional] - The Client IP Address
(cdniip) claim holds an IP address or IP prefix for which the Signed
URI is valid. This is represented in CIDR notation with dotted
decimal format for IPv4 addresses [RFC0791] or canonical text
representation for IPv6 addresses [RFC5952]. The request MUST be
rejected if sourced from a client outside the specified IP range.
Since the Client IP is considered personally identifiable
information, this field MUST be a JSON Web Encryption (JWE [RFC7516])
Object in compact serialization form. If the CDN verifying the
signed JWT does not support Client IP verification, or if the Client
IP in the signed JWT does not match the source IP address in the
content request, the CDN MUST reject the request. The type of this
claim is a JSON string that contains the JWE. If the received signed
JWT contains a Client IP claim, then any JWT subsequently generated
for CDNI redirection MUST also contain a Client IP claim, and the
Client IP value MUST be the same as in the received signed JWT. A
signed JWT generated for CDNI redirection MUST NOT add a Client IP
claim if no Client IP claim existed in the received signed JWT.

It should be noted that use of this claim can cause issues, for
example, in situations with dual-stack IPv4 and IPv6 networks, MPTCP,
QUIC, and mobile clients switching from Wi-Fi to Cellular networks
where the client's source address can change, even between address
families. This claim exists mainly for legacy feature parity
reasons; therefore, use of this claim should be done judiciously. An
example of a reasonable use case would be making a signed JWT for an
internal preview of an asset where the end consumer understands that
they must be originated from the same IP for the entirety of the
session. Using this claim at large is NOT RECOMMENDED.

2.1.11. CDNI URI Container (cdniuc) Claim

URI Container (cdniuc) [mandatory] - The URI Container (cdniuc) holds
the URI representation before a URI Signing Package is added. This
representation can take one of several forms detailed in

Section 2.1.15. If the URI Container used in the signed JWT does not
match the URI of the content request, the CDN verifying the signed
JWT MUST reject the request. When comparing the URI, the percent
encoded form as defined in Section 2.1 of [RFC3986] MUST be used.
When redirecting a URI, the CDN generating the new signed JWT MAY
change the URI Container to comport with the URI being used in the
redirection.

2.1.12. CDNI Expiration Time Setting (cdniets) Claim

CDNI Expiration Time Setting (cdniets) [optional] - The CDNI
Expiration Time Setting (cdniets) claim provides a means for setting
the value of the Expiry Time (exp) claim when generating a subsequent
signed JWT in Signed Token Renewal. Its type is a JSON numeric
value. It denotes the number of seconds to be added to the time at
which the JWT is verified that gives the value of the Expiry Time
(exp) claim of the next signed JWT. The CDNI Expiration Time Setting
(cdniets) SHOULD NOT be used when not using Signed Token Renewal and
MUST be present when using Signed Token Renewal.

2.1.13. CDNI Signed Token Transport (cdnistt) Claim

CDNI Signed Token Transport (cdnistt) [optional] - The CDNI Signed
Token Transport (cdnistt) claim provides a means of signaling the
method through which a new signed JWT is transported from the CDN to
the UA and vice versa for the purpose of Signed Token Renewal. Its
type is a JSON integer. Values for this claim are defined in
Section 6.5. If using this claim, you MUST also specify a CDNI
Expiration Time Setting (cdniets) as noted above.

2.1.14. CDNI Signed Token Depth (cdnistd) Claim

CDNI Signed Token Depth (cdnistd) [optional] - The CDNI Signed Token
Depth (cdnistd) claim is used to associate a subsequent signed JWT,
generated as the result of a CDNI Signed Token Transport claim, with
a specific URI subset. Its type is a JSON integer. Signed JWTs MUST
NOT use a negative value for the CDNI Signed Token Depth claim.

If the transport used for Signed Token Transport allows the CDN to
associate the path component of a URI with tokens (e.g., an HTTP
Cookie Path as described in Section 4.1.2.4 of [RFC6265]), the CDNI
Signed Token Depth value is the number of path segments that should
be considered significant for this association. A CDNI Signed Token
Depth of zero means that the client SHOULD be directed to return the
token with requests for any path. If the CDNI Signed Token Depth is
greater than zero, then the CDN SHOULD send the client a token to
return for future requests wherein the first CDNI Signed Token Depth
segments of the path match the first CDNI Signed Token Depth segments
of the Signed URI path. This matching MUST use the URI with the
token removed, as specified in Section 2.1.15.

If the URI path to match contains fewer segments than the CDNI Signed Token Depth claim, a signed JWT MUST NOT be generated for the purposes of Signed Token Renewal. If the CDNI Signed Token Depth claim is omitted, it means the same thing as if its value were zero. If the received signed JWT contains a CDNI Signed Token Depth claim, then any JWT subsequently generated for CDNI redirection or Signed Token Transport MUST also contain a CDNI Signed Token Depth claim, and the value MUST be the same as in the received signed JWT.

2.1.15. URI Container Forms

The URI Container (cdniuc) claim takes one of the following forms: 'hash:' or 'regex:'. More forms may be added in the future to extend the capabilities.

Before comparing a URI with contents of this container, the following steps MUST be performed:

* Prior to verification, remove the signed JWT from the URI. This removal is only for the purpose of determining if the URI matches; all other purposes will use the original URI. If the signed JWT is terminated by anything other than a sub-delimiter (as defined in Section 2.2 of [RFC3986]), everything from the reserved character (as defined in Section 2.2 of [RFC3986]) that precedes the URI Signing Package Attribute to the last character of the signed JWT will be removed, inclusive. Otherwise, everything from the first character of the URI Signing Package Attribute to the sub-delimiter that terminates the signed JWT will be removed, inclusive.

* Normalize the URI according to Section 2.7.3 of [RFC7230] and Sections 6.2.2 and 6.2.3 of [RFC3986]. This applies to both generation and verification of the signed JWT.

2.1.15.1. URI Hash Container (hash:)

Prefixed with 'hash:', this string is a URL Segment form (Section 5 of [RFC6920]) of the URI.

2.1.15.2. URI Regular Expression Container (regex:)

Prefixed with 'regex:', this string is any regular expression compatible with POSIX (Section 9 of [POSIX.1]) Extended Regular Expression used to match against the requested URI. These regular expressions MUST be evaluated in the POSIX locale (Section 7.2 of [POSIX.1]).

Note: Because '\' has special meaning in JSON [RFC8259] as the escape character within JSON strings, the regular expression character '\' MUST be escaped as '\\'.

An example of a 'regex:' is the following:

```
[^:]*\\://[^/]*/dir/content/quality_[^/]*/segment.{3}\\.mp4(\\?.*)?
```

Note: Due to computational complexity of executing arbitrary regular
expressions, it is RECOMMENDED to only execute after verifying the
JWT to ensure its authenticity.

2.2. JWT Header

The header of the JWT MAY be passed via the CDNI Metadata interface
instead of being included in the URISigningPackage. The header value
MUST be transmitted in the serialized encoded form and prepended to
the JWT payload and signature passed in the URISigningPackage prior
to verification. This reduces the size of the signed JWT token.

3. URI Signing Token Renewal

3.1. Overview

For content that is delivered via HTTP in a segmented fashion, such
as MPEG-DASH [MPEG-DASH] or HTTP Live Streaming (HLS) [RFC8216],
special provisions need to be made in order to ensure URI Signing can
be applied. In general, segmented protocols work by breaking large
objects (e.g., videos) into a sequence of small independent segments.
Such segments are then referenced by a separate manifest file, which
either includes a list of URLs to the segments or specifies an
algorithm through which a User Agent can construct the URLs to the
segments. Requests for segments 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
a vulnerability to malicious User Agents sharing the manifest file
and deep linking to the segments.

One method for dealing with this vulnerability would be to include,
in the manifest itself, Signed URIs that point to the individual
segments. 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 segmentation protocol used. Secondly, it could also
require the expiration time of the Signed URIs to be valid for an
extended duration if the content described by the manifest is meant
to be consumed in real time. For instance, if the manifest file were
to contain a segmented video stream of more than 30 minutes in
length, Signed URIs would require to be valid for at least 30
minutes, thereby reducing their effectiveness and that of the URI
Signing mechanism in general. For a more detailed analysis of how
segmented protocols such as HTTP Adaptive Streaming protocols affect
CDNI, see Models for HTTP-Adaptive-Streaming-Aware Content
Distribution Network Interconnection (CDNI) [RFC6983].

The method described in this section allows CDNs to use URI Signing for segmented content without having to include the Signed URIs in the manifest files themselves.

3.2. Signed Token Renewal Mechanism

In order to allow for effective access control of segmented content, the URI Signing mechanism defined in this section is based on a method through which subsequent segment requests can be linked together. As part of the JWT verification procedure, the CDN can generate a new signed JWT that the UA can use to do a subsequent request. More specifically, whenever a UA successfully retrieves a segment, it receives, in the HTTP 2xx Successful message, a signed JWT that it can use whenever it requests the next segment. As long as each successive signed JWT is correctly verified before a new one is generated, the model is not broken, and the User Agent can successfully retrieve additional segments. Given the fact that with segmented protocols it is usually not possible to determine a priori which segment will be requested next (i.e., to allow for seeking within the content and for switching to a different representation), the Signed Token Renewal uses the URI Regular Expression Container scoping mechanisms in the URI Container (cdniuc) claim to allow a signed JWT to be valid for more than one URL.

In order for this renewal of signed JWTs to work, it is necessary for a UA to extract the signed JWT from the HTTP 2xx Successful message of an earlier request and use it to retrieve the next segment. The exact mechanism by which the client does this 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 JWTs, Section 3.3 defines a mechanism using HTTP Cookies [RFC6265] that allows such UAs to support the concept of renewing signed JWTs without requiring any additional UA support.

3.2.1. Required Claims

The cdnistt claim (Section 2.1.13) and cdniets claim (Section 2.1.12) MUST both be present for Signed Token Renewal. cdnistt MAY be set to a value of '0' to mean no Signed Token Renewal, but there still MUST be a corresponding cdniets that verifies as a JSON number. However, if use of Signed Token Renewal is not desired, it is RECOMMENDED to simply omit both.

3.3. Communicating a Signed JWT in Signed Token Renewal

This section assumes the value of the CDNI Signed Token Transport (cdnistt) claim has been set to 1.

When using the Signed Token Renewal mechanism, the signed JWT is transported to the UA via a 'URISigningPackage' cookie added to the HTTP 2xx Successful message along with the content being returned to

the UA, or to the HTTP 3xx Redirection message in case the UA is
redirected to a different server.

3.3.1. Support for Cross-Domain Redirection

   For security purposes, the use of cross-domain cookies is not
   supported in some application environments. As a result, the Cookie-
   based method for transport of the Signed Token described in
   Section 3.3 might break if used in combination with an HTTP 3xx
   Redirection response where the target URL is in a different domain.
   In such scenarios, Signed Token Renewal of a signed JWT SHOULD be
   communicated via the query string instead, in a similar fashion to
   how regular signed JWTs (outside of Signed Token Renewal) are
   communicated. Note the value of the CDNI Signed Token Transport
   (cdnistt) claim MUST be set to 2.

   Note that the process described herein only works in cases where both
   the manifest file and segments constituting the segmented content are
   delivered from the same domain. In other words, any redirection
   between different domains needs to be carried out while retrieving
   the manifest file.

4. Relationship with CDNI Interfaces

   Some of the CDNI Interfaces need enhancements to support URI Signing.
   A dCDN that supports URI Signing needs to be able to advertise this
   capability to the uCDN. The uCDN needs to select a dCDN based on
   such capability when the CSP requires access control to enforce its
   distribution policy via URI Signing. Also, the uCDN needs to be able
   to distribute via the CDNI Metadata interface the information
   necessary to allow the dCDN to verify a Signed URI. Events that
   pertain to URI Signing (e.g., request denial or delivery after an
   access authorization decision has been made) need to be included in
   the logs communicated through the CDNI Logging interface.

4.1. CDNI Control Interface

   URI Signing has no impact on this interface.

4.2. CDNI Footprint & Capabilities Advertisement Interface

   The CDNI Request Routing: Footprint and Capabilities Semantics
   document [RFC8008] defines support for advertising CDNI Metadata
   capabilities via CDNI Payload Type. The CDNI Payload Type registered
   in Section 6.1 can be used for capability advertisement.

4.3. CDNI Request Routing Redirection Interface

   The CDNI Request Routing Redirection Interface [RFC7975] describes
   the recursive request redirection method. For URI Signing, the uCDN
   signs the URI provided by the dCDN. URI Signing therefore has no

impact on this interface.

. CDNI Metadata Interface

The CDNI Metadata Interface [RFC8006] describes the CDNI Metadata
distribution needed to enable content acquisition and delivery. For
URI Signing, a new CDNI Metadata object is specified.

The UriSigning Metadata object contains information to enable URI
Signing and verification 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 dCDN to ensure that the
         URI is signed and verified before delivering content.
         Otherwise, the dCDN does not perform verification,
         regardless of whether or not the URI is signed.

      Type: Boolean

      Mandatory-to-Specify: No. The default is true.

   Property: issuers

      Description: A list of valid Issuers against which the Issuer
         claim in the signed JWT may be cross-referenced.

      Type: Array of Strings

      Mandatory-to-Specify: No. The default is an empty list. An
         empty list means that any Issuer in the trusted key store of
         Issuers is acceptable.

   Property: package-attribute

      Description: The attribute name to use for the URI Signing
         Package.

      Type: String

      Mandatory-to-Specify: No. The default is "URISigningPackage".

   Property: jwt-header

      Description: The header part of JWT that is used for verifying
         a signed JWT when the JWT token in the URI Signing Package
         does not contain a header part.

```
      Type:  String

      Mandatory-to-Specify:  No.  By default, the header is assumed
         to be included in the JWT token.
```

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

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

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

```
{
  "generic-metadata-type": "MI.UriSigning"
  "generic-metadata-value":
    {
      "enforce": true,
      "issuers": ["csp", "ucdn1", "ucdn2"],
      "package-attribute": "usp",
      "jwt-header":
        {
            "alg": "ES256",
            "kid": "P5UpOv0eMq1wcxLf7WxIg09JdSYGYFDOWkldueaImf0"
        }
    }
}
```

4.5. CDNI Logging Interface

   For URI Signing, the dCDN 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 "Content
   Distribution Network Interconnection (CDNI) Logging Interface"
   [RFC7937] using the new "cdni_http_request_v2" record-type registered
   in Section 6.2.1.

   * s-uri-signing (mandatory):

      Format: 3DIGIT

      Field value: this characterizes the URI Signing verification
         performed by the Surrogate on the request. The allowed values
         are registered in Section 6.4.

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

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

      Format: QSTRING

      Field value: a string for providing further information in case
         the signed JWT was rejected, e.g., for debugging purposes.

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

5. URI Signing Message Flow

   URI Signing supports both HTTP-based and DNS-based request routing.
   JSON Web Token (JWT) [RFC7519] defines a compact, URL-safe means of
   representing claims to be transferred between two parties. The
   claims in a Signed JWT are encoded as a JSON object that is used as
   the payload of a JSON Web Signature (JWS) structure enabling the
   claims to be digitally signed or integrity protected with a Message
   Authentication Code (MAC).

5.1. HTTP Redirection

   For HTTP-based request routing, a set of information that is unique
   to a given end user content request is included in a Signed JWT,
   using key information that is specific to a pair of adjacent CDNI
   hops (e.g., between the CSP and the uCDN or between the uCDN and a
   dCDN). This allows a CDNI hop to ascertain the authenticity of a
   given request received from a previous CDNI hop.

   The URI Signing method (assuming HTTP redirection, iterative request
   routing, and a CDN path with two CDNs) includes the following steps:

```
    End-User              dCDN                 uCDN                 CSP
     |                     |                    |                    |
     |              1.CDNI FCI used to          |                    |
     |          advertise URI Signing capability |                    |
     |                     |------------------->|                    |
     |                     |                    |                    |
     |            2.Provides information to verify Signed JWT        |
     |                     |                    |<-------------------|
     |                     |                    |                    |
     |          3.CDNI Metadata interface used to|                    |
     |               provide URI Signing attributes|                 |
     |                     |<-------------------|                    |
     :                     :                    :                    :
     :    (Later in time)  :                    :                    :
     |4.Authorization request                   |                    |
```

```
|----------------------------------------------------------->|
|                     |                    |  [Apply distribution
|                     |                    |         policy]     |
|                     |                    |                     |
|                     |                    |  (ALT: Authorization decision)
|5.Request is denied  |                    |         <Negative>   |
|<-----------------------------------------------------------|
|                     |                    |                     |
|6.CSP provides Signed URI                 |       <Positive>    |
|<-----------------------------------------------------------|
|                     |                    |                     |
|7.Content request    |                    |                     |
|---------------------------------------->|  [Verify URI       |
|                     |                    |    signature]       |
|                     |                    |                     |
|                     |    (ALT: Verification result)          |
|8.Request is denied  |         <Negative>|                     |
|<---------------------------------------|                     |
|                     |                    |                     |
|9.Re-sign URI and redirect to  <Positive>|                     |
|  dCDN (newly Signed URI)                 |                     |
|<---------------------------------------|                     |
|                     |                    |                     |
|10.Content request   |                    |                     |
|------------------->| [Verify URI        |                     |
|                     |   signature]       |                     |
|                     |                    |                     |
|     (ALT: Verification 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 dCDN advertises its capabilities including URI Signing
         support to the uCDN.

    2.   CSP provides to the uCDN the information needed to verify Signed
         URIs from that CSP. For example, this information will include
         one or more keys used for validation.

    3.   Using the CDNI Metadata interface, the uCDN communicates to a
         dCDN the information needed to verify Signed URIs from the uCDN
         for the given CSP. For example, this information may include

the URI query string parameter name for the URI Signing Package Attribute in addition to keys used for validation.

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 local distribution policy.

5. If the authorization decision is negative, the CSP rejects the request and sends an error code (e.g., 403 Forbidden) in the HTTP response.

6. If the authorization decision is positive, the CSP computes a Signed JWT 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 uCDN verifies the Signed JWT in the URI using the information provided by the CSP.

8. If the verification result is negative, the uCDN rejects the request and sends an error code 403 Forbidden in the HTTP response.

9. If the verification result is positive, the uCDN computes a Signed JWT that is based on unique parameters of that request and provides it to the end user as the URI to use to further request the content from the dCDN.

10. On receipt of the corresponding content request, the dCDN verifies the Signed JWT in the Signed URI using the information provided by the uCDN in the CDNI Metadata.

11. If the verification result is negative, the dCDN rejects the request and sends an error code 403 Forbidden in the HTTP response.

12. If the verification result is positive, the dCDN serves the request and delivers the content.

13. At a later time, the dCDN reports logging events that include URI Signing information.

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

Note: While using a symmetric shared key is supported, it is NOT RECOMMENDED. See the Security Considerations (Section 7) about the limitations of shared keys.

5.2. DNS Redirection

For DNS-based request routing, the CSP and uCDN 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 dCDNs. However, if a
shared secret key is required, then the distribution SHOULD be
performed by the CSP directly.

Note: While using a symmetric shared key is supported, it is NOT
RECOMMENDED. See the Security Considerations (Section 7) about the
limitations of shared keys.

The URI Signing method (assuming iterative DNS request routing and a
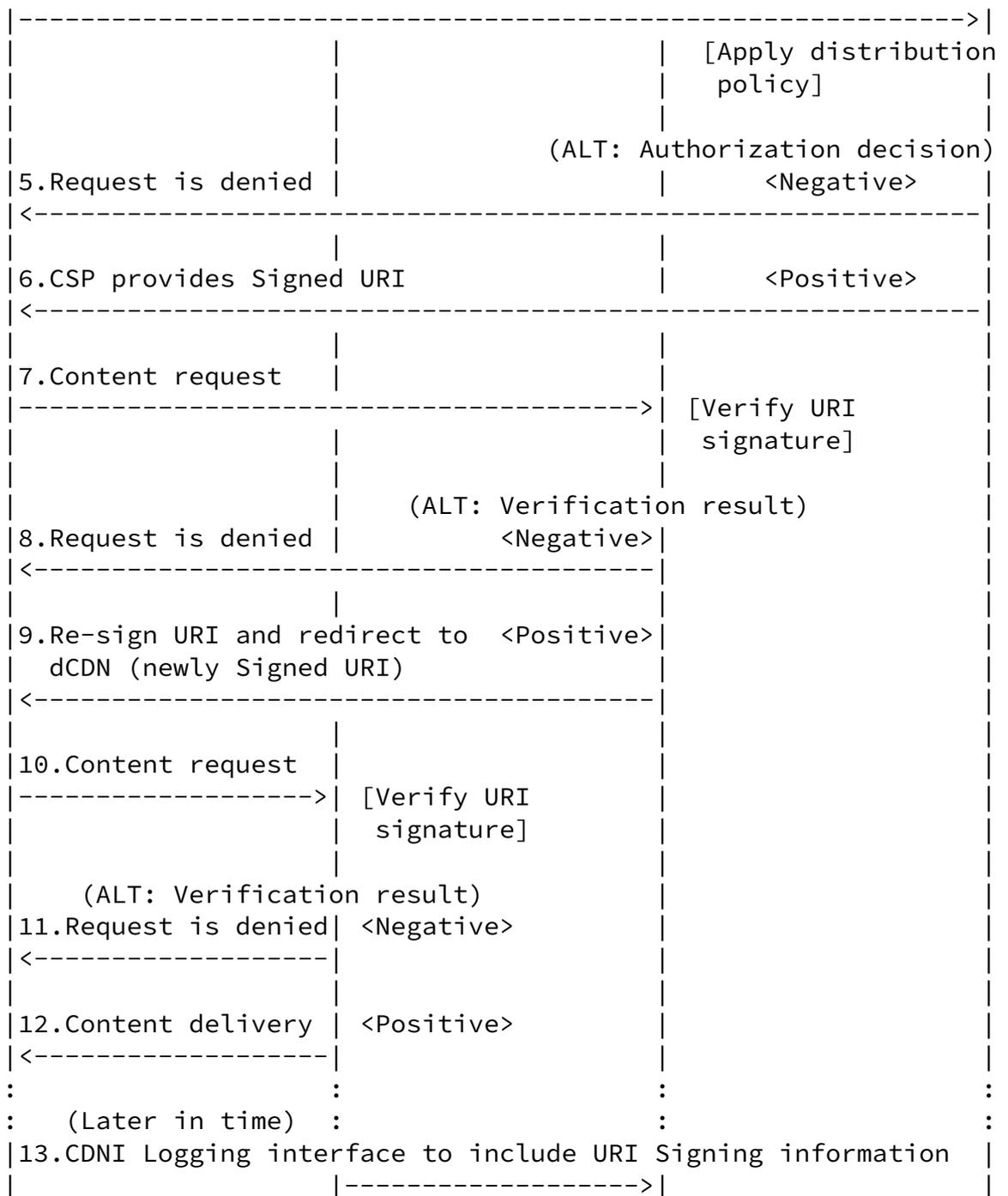CDN path with two CDNs) includes the following steps.

```
        End-User            dCDN                uCDN              CSP
         |                  |                   |                 |
         |            1.CDNI FCI used to        |                 |
         |        advertise URI Signing capability |              |
         |                  |------------------>|                 |
         |                  |                   |                 |
         |           2.Provides information to verify Signed JWT  |
         |                  |                   |<----------------|
         |           3.CDNI Metadata interface used to|           |
         |             provide URI Signing attributes|           |
         |                  |<------------------|                 |
         :                  :                   :                 :
         :   (Later in time) :                  :                 :
         |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 |                    |                   |
|------------------->| [Verify URI       |                   |
|                   |   signature]       |                   |
|                   |                    |                   |
|    (ALT: Verification result)          |                   |
|12.Request is denied| <Negative>        |                   |
|<------------------|                    |                   |
|                   |                    |                   |
|13.Content delivery | <Positive>        |                   |
|<------------------|                    |                   |
:                   :                    :                   :
:   (Later in time) :                    :                   :
|14.CDNI Logging interface to report URI Signing information |
|                   |------------------->|                   |
```
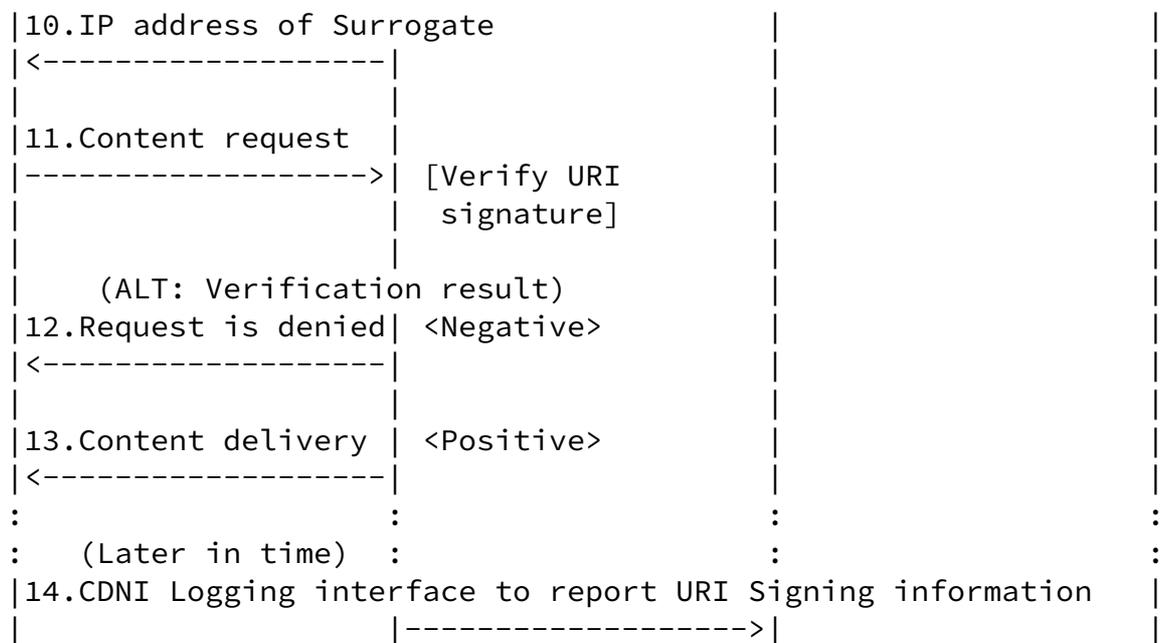
Figure 4: DNS-based Request Routing with URI Signing

1. Using the CDNI Footprint & Capabilities Advertisement interface,
   the dCDN advertises its capabilities including URI Signing
   support to the uCDN.

2. CSP provides to the uCDN the information needed to verify Signed
   JWTs from that CSP. For example, this information will include
   one or more keys used for validation.

3. Using the CDNI Metadata interface, the uCDN communicates to a
   dCDN the information needed to verify Signed JWTs from the CSP
   (e.g., the URI query string parameter name for the URI Signing
   Package Attribute). In the case of symmetric shared key, the
   uCDN MUST NOT share the key with a dCDN.

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 local distribution policy.

5. If the authorization decision is negative, the CSP rejects the
   request and sends an error code (e.g., 403 Forbidden) in the
   HTTP response.

6. If the authorization decision is positive, the CSP computes a
   Signed JWT 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. The end user sends a DNS request to the uCDN.

8. On receipt of the DNS request, the uCDN redirects the request to
   the dCDN.

9. The end user sends a DNS request to the dCDN.

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

11. On receipt of the corresponding content request, the dCDN
    verifies the Signed JWT in the URI using the information
    provided by the uCDN in the CDNI Metadata.

12. If the verification result is negative, the dCDN rejects the
    request and sends an error code 403 Forbidden in the HTTP
    response.

13. If the verification result is positive, the dCDN serves the
    request and delivers the content.

14. At a later time, dCDN 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 needs to be distributed across
multiple, possibly untrusted, CDNI hops is the public key, which is
generally not confidential.

With DNS-based request routing, URI Signing does not match well with
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 to CDNs with which the CSP may not have a
trust relationship. This raises a security concern for applicability
of URI Signing with symmetric keys in case of DNS-based inter-CDN
request routing. Due to these flaws, this architecture MUST NOT be
implemented.

Note: While using a symmetric shared key is supported, it is NOT
RECOMMENDED. See the Security Considerations (Section 7) about the
limitations of shared keys.

6. IANA Considerations

6.1. CDNI Payload Type

IANA has registered the following CDNI Payload Type under the IANA
"CDNI Payload Types" registry:

```
              +===============+===============+
              | Payload Type  | Specification |
              +===============+===============+
              | MI.UriSigning | RFC 9246      |
              +---------------+---------------+
```

Table 1

6.1.1. CDNI UriSigning Payload Type

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

   Interface: MI/FCI

   Encoding: see Section 4.4

6.2. CDNI Logging Record Type

   IANA has registered the following CDNI Logging record-type under the
   IANA "CDNI Logging record-types" registry:

   +======================+===========+===========================+
   | record-types | Reference | Description |
   +======================+===========+===========================+
cdni_http_request_v2	RFC 9246	Extension to CDNI Logging
		Record version 1 for
		content delivery using
		HTTP, to include URI
		Signing Logging fields
   +----------------------+-----------+---------------------------+

                              Table 2

6.2.1. CDNI Logging Record Version 2 for HTTP

   The "cdni_http_request_v2" record-type supports all of the fields
   supported by the "cdni_http_request_v1" record-type [RFC7937] plus
   the two additional fields "s-uri-signing" and "s-uri-signing-deny-
   reason", registered by this document in Section 6.3. The name,
   format, field value, and occurrence information for the two new
   fields can be found in Section 4.5 of this document.

6.3. CDNI Logging Field Names

   IANA has registered the following CDNI Logging fields under the IANA
   "CDNI Logging Field Names" registry:

              +============================+===========+
              | Field Name | Reference |
              +============================+===========+
              | s-uri-signing | RFC 9246 |
              +----------------------------+-----------+
              | s-uri-signing-deny-reason | RFC 9246 |
              +----------------------------+-----------+

Table 3

6.4. CDNI URI Signing Verification Code

IANA has created a new "CDNI URI Signing Verification Code" subregistry in the "Content Delivery Networks Interconnection (CDNI) Parameters" registry. The "CDNI URI Signing Verification Code" namespace defines the valid values associated with the s-uri-signing CDNI Logging field. The CDNI URI Signing Verification Code is a 3DIGIT value as defined in Section 4.5. Additions to the CDNI URI Signing Verification Code namespace will conform to the "Specification Required" policy as defined in [RFC8126]. Updates to this subregistry are expected to be infrequent.

Value	Reference	Description
=======	===========	=====================================
000	RFC 9246	No signed JWT verification performed
200	RFC 9246	Signed JWT verification performed and verified
400	RFC 9246	Signed JWT verification performed and rejected because of incorrect signature
401	RFC 9246	Signed JWT verification performed and rejected because of Issuer enforcement
402	RFC 9246	Signed JWT verification performed and rejected because of Subject enforcement
403	RFC 9246	Signed JWT verification performed and rejected because of Audience enforcement
404	RFC 9246	Signed JWT verification performed and rejected because of Expiration Time enforcement
405	RFC 9246	Signed JWT verification performed and rejected because of Not Before enforcement
406	RFC 9246	Signed JWT verification performed and rejected because only one of CDNI Signed Token Transport or CDNI

| | | | Expiration Time Setting present |
+-------+----------+----------------------------------+
407	RFC 9246	Signed JWT verification performed
		and rejected because of JWT ID
		enforcement
+-------+----------+----------------------------------+
408	RFC 9246	Signed JWT verification performed
		and rejected because of Version
		enforcement
+-------+----------+----------------------------------+
409	RFC 9246	Signed JWT verification performed
		and rejected because of Critical
		Extension enforcement
+-------+----------+----------------------------------+
410	RFC 9246	Signed JWT verification performed
		and rejected because of Client IP
		enforcement
+-------+----------+----------------------------------+
411	RFC 9246	Signed JWT verification performed
		and rejected because of URI
		Container enforcement
+-------+----------+----------------------------------+
500	RFC 9246	Unable to perform signed JWT
		verification because of malformed
		URI
+-------+----------+----------------------------------+

Table 4

6.5. CDNI URI Signing Signed Token Transport

IANA has created a new "CDNI URI Signing Signed Token Transport"
subregistry in the "Content Delivery Networks Interconnection (CDNI)
Parameters" registry. The "CDNI URI Signing Signed Token Transport"
namespace defines the valid values that may be in the Signed Token
Transport (cdnistt) JWT claim. Additions to the Signed Token
Transport namespace conform to the "Specification Required" policy as
defined in [RFC8126]. Updates to this subregistry are expected to be
infrequent.

The following table defines the initial Enforcement Information
Elements:

+=======+===+==========+
| Value | Description | RFC |
+=======+===+==========+
| 0 | Designates token transport is not enabled | RFC 9246 |
+-------+---+----------+
| 1 | Designates token transport via cookie | RFC 9246 |
+-------+---+----------+
| 2 | Designates token transport via query string | RFC 9246 |

```
   +-------+------------------------------------------+----------+
```

                              Table 5

6.6. JSON Web Token Claims Registration

   This specification registers the following claims in the IANA "JSON
   Web Token Claims" registry [IANA.JWT.Claims] established by
   [RFC7519].

6.6.1. Registry Contents

   Claim Name: cdniv
   Claim Description: CDNI Claim Set Version
   Change Controller: IESG
   Specification Document(s): Section 2.1.8 of RFC 9246

   Claim Name: cdnicrit
   Claim Description: CDNI Critical Claims Set
   Change Controller: IESG
   Specification Document(s): Section 2.1.9 of RFC 9246

   Claim Name: cdniip
   Claim Description: CDNI IP Address
   Change Controller: IESG
   Specification Document(s): Section 2.1.10 of RFC 9246

   Claim Name: cdniuc
   Claim Description: CDNI URI Container
   Change Controller: IESG
   Specification Document(s): Section 2.1.11 of RFC 9246

   Claim Name: cdniets
   Claim Description: CDNI Expiration Time Setting for Signed Token
      Renewal
   Change Controller: IESG
   Specification Document(s): Section 2.1.12 of RFC 9246

   Claim Name: cdnistt
   Claim Description: CDNI Signed Token Transport Method for Signed
      Token Renewal
   Change Controller: IESG
   Specification Document(s): Section 2.1.13 of RFC 9246

   Claim Name: cdnistd
   Claim Description: CDNI Signed Token Depth
   Change Controller: IESG
   Specification Document(s): Section 2.1.14 of RFC 9246

6.7. Expert Review Guidance

Generally speaking, we should determine the registration has a
rational justification and does not duplicate a previous
registration. Early assignment should be permissible as long as
there is a reasonable expectation that the specification will become
formalized. Expert Reviewers should be empowered to make
determinations, but generally speaking they should allow new claims
that do not otherwise introduce conflicts with implementation or
things that may lead to confusion. They should also follow the
guidelines of Section 5 of [RFC8126] when sensible.

7. Security Considerations

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

   CDNI URI Signing Signed Tokens leverage JSON Web Tokens and thus,
   guidelines in [RFC8725] are applicable for all JWT interactions.

   In general, it holds that the level of protection against
   illegitimate access can be increased by including more claims in the
   signed JWT. The current version of this document includes claims for
   enforcing Issuer, Client IP Address, Not Before time, 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: A (valid) Signed URI may be used to perform replay
      attacks. The vulnerability to replay attacks can be reduced by
      picking a relatively short window between the Not Before time and
      Expiration Time attributes, although this is limited by the fact
      that any HTTP-based request needs a window of at least a couple of
      seconds to prevent sudden network issues from denying legitimate
      UAs access to the content. One may also reduce exposure to replay
      attacks by including a unique one-time access ID via the JWT ID
      attribute (jti claim). Whenever the dCDN receives a request with
      a given unique ID, it adds that ID to the list of 'used' IDs. In
      the case an illegitimate UA tries to use the same URI through a
      replay attack, the dCDN can deny the request based on the already
      used access ID. This list should be kept bounded. A reasonable
      approach would be to expire the entries based on the exp claim
      value. If no exp claim is present, then a simple Least Recently
      Used (LRU) cache could be used; however, this would allow values

to eventually be reused.

Illegitimate clients 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 dCDN. This results in the dCDN not
being able to distinguish between different users based on Client
IP Address, which can lead to 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, e.g., attributes that can be found in HTTP headers.
However, this may be easily circumvented by a sophisticated
attacker.

A shared key distributed between CSP and uCDN is more likely to be
compromised. Since this key can be used to legitimately sign a URL
for content access authorization, it is important to know the
implications of a compromised shared key. While using a shared key
scheme can be convenient, this architecture is NOT RECOMMENDED due to
the risks associated. It is included for legacy feature parity and
is highly discouraged in new implementations.

If a shared key usable for signing is compromised, an attacker can
use it to perform a denial-of-service attack by forcing the CDN to
evaluate prohibitively expensive regular expressions embedded in a
URI Container (cdniuc) claim. As a result, compromised keys should
be timely revoked in order to prevent exploitation.

The URI Container (cdniuc) claim can be given a wildcard value.
This, combined with the fact that it is the only mandatory claim,
means you can effectively make a skeleton key. Doing this does not
sufficiently limit the scope of the JWT and is NOT RECOMMENDED. The
only way to prevent such a key from being used after it is
distributed is to revoke the signing key so it no longer validates.

8. Privacy

The privacy protection concerns described in "Content Distribution
Network Interconnection (CDNI) Logging Interface" [RFC7937] apply
when the client's IP address (cdniip) or Subject (sub) is embedded in
the Signed URI. For this reason, the mechanism described in
Section 2 encrypts the Client IP or Subject before including it in
the URI Signing Package (and thus the URL itself).

9. References

9.1. Normative References

   [POSIX.1]   The Open Group, "IEEE Standard for Information Technology
               -- Portable Operating System Interface (POSIX(TM)) Base
               Specifications, Issue 7", (Revision of IEEE Std
               1003.1-2008), IEEE Std 1003.1-2017, January 2018,

                    <https://pubs.opengroup.org/onlinepubs/9699919799/>.

[RFC0791]   Postel, J., "Internet Protocol", STD 5, RFC 791,
            DOI 10.17487/RFC0791, September 1981,
            <https://www.rfc-editor.org/info/rfc791>.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC5905]   Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,
            "Network Time Protocol Version 4: Protocol and Algorithms
            Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010,
            <https://www.rfc-editor.org/info/rfc5905>.

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

[RFC6265]   Barth, A., "HTTP State Management Mechanism", RFC 6265,
            DOI 10.17487/RFC6265, April 2011,
            <https://www.rfc-editor.org/info/rfc6265>.

[RFC6570]   Gregorio, J., Fielding, R., Hadley, M., Nottingham, M.,
            and D. Orchard, "URI Template", RFC 6570,
            DOI 10.17487/RFC6570, March 2012,
            <https://www.rfc-editor.org/info/rfc6570>.

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

[RFC6920]   Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B.,
            Keranen, A., and P. Hallam-Baker, "Naming Things with
            Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013,
            <https://www.rfc-editor.org/info/rfc6920>.

[RFC7230]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
            Protocol (HTTP/1.1): Message Syntax and Routing",
            RFC 7230, DOI 10.17487/RFC7230, June 2014,
            <https://www.rfc-editor.org/info/rfc7230>.

[RFC7516]   Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)",

              RFC 7516, DOI 10.17487/RFC7516, May 2015,
              <https://www.rfc-editor.org/info/rfc7516>.

   [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token
              (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015,
              <https://www.rfc-editor.org/info/rfc7519>.

   [RFC7937] Le Faucheur, F., Ed., Bertrand, G., Ed., Oprescu, I., Ed.,
              and R. Peterkofsky, "Content Distribution Network
              Interconnection (CDNI) Logging Interface", RFC 7937,
              DOI 10.17487/RFC7937, August 2016,
              <https://www.rfc-editor.org/info/rfc7937>.

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

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
              Interchange Format", STD 90, RFC 8259,
              DOI 10.17487/RFC8259, December 2017,
              <https://www.rfc-editor.org/info/rfc8259>.

9.2. Informative References

   [IANA.JWT.Claims]
              IANA, "JSON Web Token (JWT)",
              <https://www.iana.org/assignments/jwt>.

   [MPEG-DASH]
              ISO, "Information technology -- Dynamic adaptive streaming
              over HTTP (DASH) -- Part 1: Media presentation description
              and segment formats", ISO/IEC 23009-1:2019, Edition 4,
              December 2019, <https://www.iso.org/standard/79329.html>.

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

   [RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed.,

              "Framework for Content Distribution Network
              Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336,
              August 2014, <https://www.rfc-editor.org/info/rfc7336>.

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

   [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517,
              DOI 10.17487/RFC7517, May 2015,
              <https://www.rfc-editor.org/info/rfc7517>.

   [RFC7975] Niven-Jenkins, B., Ed. and R. van Brandenburg, Ed.,
              "Request Routing Redirection Interface for Content
              Delivery Network (CDN) Interconnection", RFC 7975,
              DOI 10.17487/RFC7975, October 2016,
              <https://www.rfc-editor.org/info/rfc7975>.

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

   [RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming",
              RFC 8216, DOI 10.17487/RFC8216, August 2017,
              <https://www.rfc-editor.org/info/rfc8216>.

   [RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best
              Current Practices", BCP 225, RFC 8725,
              DOI 10.17487/RFC8725, February 2020,
              <https://www.rfc-editor.org/info/rfc8725>.

Appendix A. Signed URI Package Example

   This section contains three examples of token usage: a simple example
   with only the required claim present, a complex example that
   demonstrates the full JWT claims set, including an encrypted Client
   IP Address (cdniip), and one that uses a Signed Token Renewal.

   Note: All of the examples have empty space added to improve
   formatting and readability, but are not present in the generated
   content.

   All examples use the following JWK Set [RFC7517]:

   { "keys": [
     {
       "kty": "EC",
       "kid": "P5UpOv0eMq1wcxLf7WxIg09JdSYGYFDOWkldueaImf0",

```
      "use": "sig",
      "alg": "ES256",
      "crv": "P-256",
      "x": "be807S4O7dzB6I4hTiCUvmxCI6FuxWba1xYBlLSSsZ8",
      "y": "rOGC4vI69g-WF9AGEVI37sNNwbjIzBxSjLvIL7f3RBA"
    },
    {
      "kty": "EC",
      "kid": "P5UpOv0eMq1wcxLf7WxIg09JdSYGYFDOWkldueaImf0",
      "use": "sig",
      "alg": "ES256",
      "crv": "P-256",
      "x": "be807S4O7dzB6I4hTiCUvmxCI6FuxWba1xYBlLSSsZ8",
      "y": "rOGC4vI69g-WF9AGEVI37sNNwbjIzBxSjLvIL7f3RBA",
      "d": "yaowezrCLTU6yIwUL5RQw67cHgvZeMTLVZXjUGb1A1M"
    },
    {
      "kty": "oct",
      "kid": "f-WbjxBC3dPuI3d24kP2hfvos7Qz688UTi6aB0hN998",
      "use": "enc",
      "alg": "A128GCM",
      "k": "4uFxxV7fhNmrtiah2d1fFg"
    }
  ]}
```

   Note: They are the public signing key, the private signing key, and
   the shared secret encryption key, respectively. The public and
   private signing keys have the same fingerprint and only vary by the
   'd' parameter that is missing from the public signing key.

A.1. Simple Example

   This example is a simple common usage example containing a minimal
   subset of claims that the authors find most useful.

   The JWT Claim Set before signing:

   Note: "sha-256;2tderfWPa86Ku7YnzW51YUp7dGUjBS_3SW3ELx4hmWY" is the
   URL Segment form (Section 5 of [RFC6920]) of
   "http://cdni.example/foo/bar".

```
{
  "exp": 1646867369,
  "iss": "uCDN Inc",
  "cdniuc":
    "hash:sha-256;2tderfWPa86Ku7YnzW51YUp7dGUjBS_3SW3ELx4hmWY"
}
```

   The signed JWT:

   eyJhbGciOiJFUzI1NiIsImtpZCI6IlA1VXBPdjBlTXExd2N4TGY3V3hJZzA5SmRTWU

```
dZRkRPV2tsZHVlYUltZjAifQ.eyJleHAiOjE2NDY4NjczNjksImlzcyI6InVDRE4gS
W5jIiwiY2RuaXVjIjoiaGFzaGGEtMjU2OzJ0ZGVyZldQYTg2S3U3WW56VzUxWVV
wN2RHVWpCU18zU1czRUx4NGhtV1kifQ.TaNlJM3D96i_9J9XvlICO6FUIDFTqt3E2Y
JkEUOLfcH0b89wYRKTbJ9Yj6h_GRgSoZoQO0cps3yUPcWGK3smCw
```

. Complex Example

   This example uses all fields except for those dealing with Signed
   Token Renewal, including Client IP Address (cdniip) and Subject
   (sub), which are encrypted. This significantly increases the size of
   the signed JWT token.

   JWE for Client IP Address (cdniip) of [2001:db8::1/32]:

```
eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XYmp4QkMzZFB1ST
NkMjRrUDJoZnZvczdRejY4OFVaTZhQjBoTjk5OCJ9..aUDDFEQBIc3nWjOb.bGXWT
HPkntmPCKn0pPPNEQ.iyTttnFybO2YBLqwl_YSjA
```

   JWE for Subject (sub) of "UserToken":

```
eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XYmp4QkMzZFB1ST
NkMjRrUDJoZnZvczdRejY4OFVaTZhQjBoTjk5OCJ9..CLAu80xclc8Bp-Ui.6P1A3
F6ip2Dv.CohdtLLpgBnTvRJQCFuz-g
```

   The JWT Claim Set before signing:

```
{
  "aud": "dCDN LLC",
  "sub": "eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XYmp4
QkMzZFB1STNkMjRrUDJoZnZvczdRejY4OFVaTZhQjBoTjk5OCJ9..CLAu80xclc8B
p-Ui.6P1A3F6ip2Dv.CohdtLLpgBnTvRJQCFuz-g",
  "cdniip": "eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIiwia2lkIjoiZi1XY
mp4QkMzZFB1STNkMjRrUDJoZnZvczdRejY4OFVaTZhQjBoTjk5OCJ9..aUDDFEQBI
c3nWjOb.bGXWTHPkntmPCKn0pPPNEQ.iyTttnFybO2YBLqwl_YSjA",
  "cdniv": 1,
  "exp": 1646867369,
  "iat": 1646694569,
  "iss": "uCDN Inc",
  "jti": "5DAafLhZAfhsbe",
  "nbf": 1646780969,
  "cdniuc": "regex:http://cdni\\.example/foo/bar/[0-9]{3}\\.png"
}
```

   The signed JWT:

```
eyJhbGciOiJFUzI1NiIsImtpZCI6IlA1VXBPdjBlTXExd2N4TGY3V3hJZzA5SmRTWU
dZRkRPV2tsZHVlYUltZjAifQ.eyJhdWQiOiJkQ0ROIExMQyIsInN1YiI6ImV5Smxib
U1pT2lKQk1USTRSME5OSWl3aVlXeG5Jam9pWkdseUlpd2lhMmxrSWpvaVppMVhZbXA
0UWtNelpGQjFTVE5rTWpSclVESm9ablo2Y3pkUmVqWTRPFVaVGhRakJvVGprNU9DSjT
0NKOS4uQ0xBdTgweGNsYzhCcC1VaS42UDFBM0Y2aXAyRHYuQ29oZHRMTHBnQm5UdlJ
KUUNGdXotZyIsImNkbmlpcCI6ImV5SmxibU1pT2lKQk1USTRSME5OSWl3aVlXeG5Ja
```

m9pWkdseUlpd2lhMmxrSWpvaVppMVhZbXA0UWtNelpGQjFTVE5rTWpSclVESm9ablp
2Y3pkUmVqWTRPRlZVYVRaaFFqQm9Uams1T0NKOS4uYVVVEREZFUUJJYzNuV2pPYi5iR
1hXVEhQa250bVBDS24wcFBQTkVRLml5VHR0bkZ5Yk8yWUJMcXdsX1lTakEiLCJjZG5
pdiI6MSwiZXhwIjoxNjQ2ODY3MzY5LCJpYXQiOjE2NDY2OTQ1NjksImlzcyI6InVDR
E4gSW5jIiwianRpIjoiNURBYWZMaFpBZmhzYmUiLCJuYmYiOjE2NDY3ODA5Njks ImN
kbml1YyI6InJlZ2V4Omh0dHA6Ly9jZG5pXFwuZXhhbXBsZS9mb28vYmFyL1swLTlde
zN9XFwucG5nIn0.IjmVX0uD5MYqArc-M08uEsEeoDQn8kuYXZ9HGHDmDDxsHikT0c8
jcX8xYD0z3LzQclMG65i1kT2sRbZ7isUw8w

A.3. Signed Token Renewal Example

This example uses fields for Signed Token Renewal.

The JWT Claim Set before signing:

```
{
  "cdniets": 30,
  "cdnistt": 1,
  "cdnistd": 2,
  "exp": 1646867369,
  "cdniuc": "regex:http://cdni\\.example/foo/bar/[0-9]{3}\\.ts"
}
```

The signed JWT:

eyJhbGciOiJFUzI1NiIsImtpZCI6IlA1VXBPdjBlTXExd2N4TGY3V3hJZzA5SmRTWU
dZRkRPV2tsZHVlYUltZjAifQ.eyJjZG5pZXRzIjozMCwiY2RuaXN0dCI6MSwiY2Rua
XN0ZCI6MiwiZXhwIjoxNjQ2ODY3MzY5LCJjZG5pdWMiOiJyZWdleDpodHRwOi8vY2R
uaVxcLmV4YW1wbGUvZm9vL2Jhci9bMC05XXszfVxcLnRzIn0.tlPvoKw3BCClw4Lx9
PQu7MK6b2IN55ZoCPSaxovGK0zS53Wpb1MbJBow7G8LiGR39h6-2Iq7PWUSr3MdTIz
HYw

Once the server verifies the signed JWT it will return a new signed
JWT with an updated Expiry Time (exp) as shown below. Note the
Expiry Time is increased by the expiration time setting (cdniets)
value.

The JWT Claim Set before signing:

```
{
  "cdniets": 30,
  "cdnistt": 1,
  "cdnistd": 2,
  "exp": 1646867399,
  "cdniuc": "regex:http://cdni\\.example/foo/bar/[0-9]{3}\\.ts"
}
```

The signed JWT:

eyJhbGciOiJFUzI1NiIsImtpZCI6IlA1VXBPdjBlTXExd2N4TGY3V3hJZzA5SmRTWU
dZRkRPV2tsZHVlYUltZjAifQ.eyJjZG5pZXRzIjozMCwiY2RuaXN0dCI6MSwiY2Rua

XN0ZCI6MiwiZXhwIjoxNjQ2ODY3Mzk5LCJjZG5pdWMiOiJyZWdleDpodHRwOi8vY2R
uaVxcLmV4YW1wbGUvZm9vL2Jhci9bMC05XXszfVxcLnRzIn0.ivY5d_fKGd-OHTpUs
8uJUrnHvt-rduzu5H4zM7167pUUAghub53FqDQ5G16jRYX2sY73mA_uLpYDdb-CPts
8FA

Acknowledgements

Contributors

   In addition, the authors would also like to make special mentions for
   certain people who contributed significant sections to this document.

   * Matt Caulfield provided content for [Section 4.4](#), "CDNI Metadata
      Interface".

   * Emmanuel Thomas provided content for HTTP Adaptive Streaming.

   * Matt Miller provided consultation on JWT usage as well as code to
      generate working JWT examples.

Authors' Addresses

   Ray van Brandenburg
   Tiledmedia
   Anna van Buerenplein 1
   2595DA Den Haag
   Netherlands
   Phone: +31 88 866 7000
   Email: ray@tiledmedia.com

   Kent Leung
   Email: mail4kentl@gmail.com

   Phil Sorber
   Apple, Inc.
   Suite 410
   1800 Wazee Street
   Denver, CO 80202
   United States
   Email: sorber@apple.com