Workgroup: TODO Working Group
Internet-Draft:
draft-pfeairheller-cesr-proof-01
Published: 31 July 2023
Intended Status: Informational
Expires: 1 February 2024
Authors: P. Feairheller
GLEIF

CESR Proof Signatures

Abstract

CESR Proof Signatures are an extension to the Composable Event Streaming Representation [CESR] that provide transposable cryptographic signature attachments on self-addressing data (SAD) [SAID]. Any SAD, such as an Authentic Chained Data Container (ACDC) Verifiable Credential [ACDC] for example, may be signed with a CESR Proof Signature and streamed along with any other CESR content. In addition, a signed SAD can be embedded inside another SAD and the CESR proof signature attachment can be transposed across envelope boundaries and streamed without losing any cryptographic integrity.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/trustoverip/tswg-cesr-proof-specification.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Composable Event Streaming Representation (CESR) is a dual text-binary encoding format that has the unique property of text-binary concatenation composability. The CESR specification not only provides the definition of the streaming format but also the attachment codes needed for differentiating the types of cryptographic material (such as signatures) used as attachments on all event types for the Key Event Receipt Infrastructure (KERI) [KERI]. While all KERI event messages are self-addressing data (SAD), there is a broad class of SADs that are not KERI events but that require signature attachments. ACDC Verifiable credentials fit into this class of SADs. With more complex data structures represented as SADs, such as verifiable credentials, there is a need to provide signature attachments on nested subsets of SADs. Similar to indices in indexed controller signatures in KERI that specify the location of the public key they represent, nested SAD signatures need a path mechanism to specify the exact location of the nested content that they are signing. CESR Proof Signatures provide this mechanism with the CESR SAD Path Language and new CESR attachment codes, detailed in this specification.

1.1. Streamable SADs

A primary goal of CESR Proof Signatures is to allow any signed self-addressing data (SAD) to be streamed inline with any other CESR content. In support of that goal, CESR Proof Signatures leverage CESR attachments to define a signature scheme that can be attached to any SAD content serialized as JSON [JSON], MessagePack [MGPK] or CBOR [CBOR]. Using this capability, SADs signed with CESR Proof Signatures can be streamed inline in either the text (T) or binary (B) domain alongside any other KERI event message over, for example TCP or UDP. In addition, signed SADs can be transported via HTTP as a CESR HTTP Request (todo: reference needed).

1.2. Nested Partial Signatures

CESR Proof Signatures can be used to sign as many portions of a SAD as needed, including the entire SAD. The signed subsets are either SADs themselves or the self-addressing identifer (SAID) of a SAD that will be provided out of band. A new CESR count code is included with this specification to allow for multiple signatures on nested portions of a SAD to be grouped together under one attachment. By including a SAD Path in the new CESR attachment for grouping signatures, the entire group of signatures can be transposed across envelope boundaries by changing only the root path of the group attachment code.

1.3. Transposable Signature Attachments

There are several events in KERI that can contain context specific embedded self-addressing data (SADs). Exchange events (exn) for peer-to-peer communication and Replay events (rpy) for responding to data requests as well as Expose events (exp) for providing anchored data are all examples of KERI events that contain embedded SADs as part of their payload. If the SAD payload for one of these event types is signed with a CESR attachment, the resulting structure is not embeddable in one of the serializations of map or dictionary like data models. (JSON, CBOR, MessagePack) supported by CESR. To solve this problem, CESR Proof Signatures are transposable across envelope boundaries in that a single SAD signature or an entire signature group on any given SAD can be transposed to attach to the end of an enveloping SAD without losing its meaning. This unique feature is provided by the SAD Path language used in either a SAD signature or the root path designation in the outermost attachment code of any SAD signature group. These paths can be updated to point to the embedded location of the signed SAD inside the envelope. Protocols for verifiable credential issuance and proof presentation can be defined using this capability to embed the same verifiable credential SAD at and location in an enveloping exn message as appropriate for the protocol without having to define a unique signature scheme for each protocol.

2. CESR SAD Path Language

CESR Proof Signatures defines a SAD Path Language to be used in signature attachments for specifying the location of the SAD content within the signed SAD that a signature attachment is verifying. This path language has a more limited scope than alternatives like JSONPtr [RFC6901] or JSONPath [JSONPath] and is therefore simpler and more compact when encoding in CESR signature attachments. SADs in CESR and therefore CESR Proof Signatures require static field ordering of all maps. The SAD path language takes advantage of this feature to allow for a Base64 compatible syntax into SADs even when a SAD uses non-Base64 compatible characters for field labels.

2.1. Description and Usage

The SAD path language contains a single reserved character, the - (dash) character. Similar to the / (forward slack) character in URLs, the - in the SAD Path Language is the path separator between components of the path. The - was selected because it is a one of the valid Base64 characters.

The simplest path in the SAD Path Language is a single - character representing the root path which specifies the top level of the SAD content.

Root Path

-

After the root path, path components follow, delimited by the - character. Path components may be integer indices into field labels or arrays or may be full field labels. No wildcards are supported by the SAD Path Language.

An example SAD Path using only labels that resolve to map contexts follows:

-a-personal

In addition, integers can be specified and their meaning is dependent on the context of the SAD.

-1-12-personal-0

The rules for a SAD Path Language processor are simple. If a path consists of only a single -, it represents the root of the SAD and therefore the entire SAD content. Following any - character is a path component that points to a field if the current context is a map in the SAD or is an index of an element if the current context is an array. It is an error for any sub-path to resolve to a value this is not a map or an array. Any trailing - character in a SAD Path can be ignored.

The root context (after the initial -) is always a map. Therefore, the first path component represents a field of that map. The SAD is traversed following the path components as field labels or indexes in arrays until the end of the path is reached. The value at the end of the path is then returned as the resolution of the SAD Path. If the current context is a map and the path component is an integer, the path component represents an index into fields of the map. This feature takes advantage of the static field ordering of SADs and is used against any SAD that contains field labels that use non-Base64 compatible characters or the - character. Any combination of integer and field label path components can be used when the current context is a map. All path components **MUST** be an integer when the current context is an array.

2.2. CESR Encoding for SAD Path Language

SAD Paths are variable raw size primitives that require CESR variable size codes. We will use the A small variable size code for SAD Paths which has 3 code entries being added to the Master Code Table, 4A##, 5A## and 6A## for SAD Paths with 0 lead bytes, 1 lead byte and 2 lead bytes respecively. This small variable size code is reserved for text values that only contain valid Base64 characters.

These codes are detailed in Table 2 below. The selector not only encodes the table but also implicitly encodes the number of lead bytes. The variable size is measured in quadlets of 4 characters each in the T domain and equivalently in triplets of 3 bytes each in the B domain. Thus computing the number of characters when parsing or off-loading in the T domain means multiplying the variable size by 4. Computing the number of bytes when parsing or off-loading in the B domain means multiplying the variable size by 3. The two Base64 size characters provide value lengths in quadlets/triplets from 0 to 4095 (64**2 -1). This corresponds to path lengths of up to 16,380 characters (4095 * 4) or 12,285 bytes (4095 * 3).

2.3. SAD Path Examples

This section provides some more examples for SAD Path expressions. The examples are based on Authentic Chained Data Containers (ACDCs) representing verifiable credentials.

```
{
  "v": "ACDC10JSON00011c_",
  "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a": {
    "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
    "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
    "dt": "2021-06-09T17:35:54.169967+00:00",
    "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
    "LEI": "254900OPPU84GM83MG36",
    "personal": {
      "legalName": "John Doe",
      "home-city": "Durham"
    }
  },
  "p": [
    {
      "qualifiedIssuerCredential": {
        "d": "EIl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA",
        "i": "Et2DOOu4ivLsjpv89vgv6auPntSLx4CvOhGUxMhxPS24"
      }
    },
    {
      "certifiedLender": {
        "d": "EglG9JLG6UhkLrrv012NPuLEc1F3ne5vPH_sHGP_QPN0",
        "i": "E8YrUcVIqrMtDJHMHDde7LHsrBOpvN38PLKe_JCDzVrA"
      }
    }
  ]
}
```

Figure 1. Example ACDC Credential SAD

The examples in Table 1 represent all the features of the SAD Path language when referring to the SAD in Figure 1. along with the CESR text encoding.

SAD Path	Result	CESR T Domain Encoding
-	The root of the SAD	6AABAAA-
-a-personal	The personal map of the a field	4AADA-a-personal
-4-5	The personal map of the a field	4AAB-4-5
-4-5-legalName	"John Doe"	5AAEAA-4-5-legalName
-a-personal-1	"Durham"	6AAEAAA-a-personal-1
-p-1	The second element in the p array	4AAB-p-1
-a-LEI	"254900OPPU84GM83MG36"	5AACAA-a-LEI
-p-0-0-d	"EIl3MORH...6DZA"	4AAC-p-0-0-d
-p-0-certifiedLender-i	"E8YrUcVI...zVrA"	5AAGAA-p-0-certifiedLender-i

Table 1

2.4. Alternative Pathing / Query Languages

The SAD Path language was chosen over alternatives such as JSONPtr and JSONPath in order to create a more compact representation of a pathing language in the text domain. Many of the features of the alternatives are not needed for CESR Proof Signatures. The only token in the language (-) is Base64 compatible. The use of field indices in SADs (which require staticly ordered fields) allows for Base64 compatible pathing even when the field labels of the target SAD are not Base64 compatible. The language accomplishes the goal of uniquely locating any path in a SAD using minimally sufficient means in a manner that allows it to be embedded in a CESR attachment as Base64. Alternative syntaxes would need to be Base64 encoded to be used in a CESR attachment in the text domain thus incurring the additional bandwidth cost of such an encoding.

3. CESR Attachments

This specification adds 2 *Counter Four Character Codes* to the Master Code Table and uses 1 *Small Variable Raw Size Code Type* and 1 *Large Variable Raw Size Code Type* from the Master Code Table (each of which have 3 code entries).

3.1. Counter Four Character Codes

The SAD Path Signature counter code is represented by the four character code -J##. The first two characters reserve this code for attaching the couplet (SAD Path, Signature Group). The second two characters represent the count in hexidecimal of SAD path signatures are in this attachment. The path is attached in the T domain using the codes described in the next section. The signature group is from either a transferable identifier or a non-transferable identifier and therefore attached using the CESR codes -F## or -C## respectively as described in the CESR Specification [CESR].

3.2. Variable Size Codes

The code A is reserved as a Small Variable Raw Size Code and AAA as a Large Variable Raw Size Code for Base64 URL safe strings. SAD Paths are Base64 URL safe strings and so leverage these codes when encoded in the CESR T domain. To account for the variable nature of path strings, the variable size types reserve 3 codes each with prefix indicators of lead byte size used for adjusting the T domain encoding to multiples of 4 characters and the B domain to multiples of 3 bytes. For the *Small* codes the prefix indicators are 4, 5 and 6 representing 0, 1 and 2 lead bytes respectively and for *Large* codes the prefix indicators are 7, 8, and 9 representing 0, 1 and 2 lead bytes respectively. The resulting 6 code entries are displayed in the table that follows.

The additions to the Master Code Table of CESR is shown below:

Code	Description	Code Length	Count or Index Length	Total Length
	Counter Four Character Codes			
-J##	Count of attached qualified Base64 SAD path sig groups path+sig group (trans or non-trans)	2	2	4
-K##	Count of attached qualified Base64 SAD Path groups	2	2	4
	Small Variable Raw Size Code			
4A##	String Base64 Only with 0 Lead Bytes	2	2	4
5A##	String Base64 Only with 1 Lead Byte	2	2	4
6A##	String Base64 Only with 2 Lead Bytes	2	2	4
	Large Variable Raw Size Code			
7AAA####	String Base64 Only with 0 Lead Bytes	4	4	8

Code	Description	Code Length	Count or Index Length	Total Length
8AAA####	String Base64 Only with 1 Lead Byte	4	4	8
9AAA####	String Base64 Only with 2 Lead Bytes	4	4	8

Table 2

3.3. CESR Signature Attachments

CESR defines several counter codes for attaching signatures to serialized CESR event messages. For KERI event messages, the signatures in the attachments apply to the entire serialized content of the KERI event message. As all KERI event messages are SADs, the same rules for signing a KERI event message applies to signing SADs for CESR Proof Signatures. A brief review of CESR signatures for transferable and non-transferable identifiers follows. In addition, signatures on nested content must be specified.

3.3.1. Signing SAD Content

Signatures on SAD content require signing the serialized encoding format of the data ensuring that the signature applies to the data over the wire. The serialization for any SAD is identified in the version string which can be found in the v field of any KERI event message or ACDC credential. An example version string follows:

```
{
    "v": "KERI10JSON00011c_"
}
```

where KERI is the identifier of KERI events followed by the hexidecimal major and minor version code and then the serialized encoding format of the event, JSON in this case. KERI and ACDC support JSON, MessagePack and CBOR currently. Field ordering is important when apply cryptographic signatures and all serialized encoding formats must support static field ordering. Serializing a SAD starts with reading the version string from the SAD field (v for KERI and ACDC events message) to determine the serialized encoding format of the message. The serialized encoding format is used to generate the SAID at creation and can not be changed. The event map is serialized using a library that ensures the static field order perserved across serialization and deserialization and the private keys are used to generate the qualified cryptographic material that represents the signatures over the SAD content.

The same serialized encoding format must be used when nesting a SAD in another SAD. For example, an ACDC credential that was issued

using JSON can only be embedded and presented in a KERI exn presentation event message that uses JSON as its serialized encoding format. That same credential can not be transmitted using CBOR or MessagePack. Controllers can rely on this restriction when verifying signatures of embedded SADs. When processing the signature attachments and resolving the data at a given SAD path, the serialization of the outter most SAD can be used at any depth of the traversal. New verison string processing does not need to occur at nested paths. However, if credential signature verification is pipelined and processed in parallel to the event message such that the event message is not avaiable, the version string of the nested SAD will still be valid and can be used if needed.

Each attached signature is accompanied by a SAD Path that indicates the content that is signed. The path must resolve within the enveloping SAD to either a nested SAD (map) or a SAID (string) of an externally provided SAD. This of course, includes a root path that resolves to the enveloping SAD itself.

3.3.2. Signatures with Non-Transferable Identifiers

Non-transferable identifiers only ever have one public key. In addition, the identifier prefix is identical to the qualified cryptographic material of the public key and therefore no KEL is required to validate the signature of a non-transferable identifier [KERI]. The attachment code for witness receipt couplets, used for CESR Proof Signatures, takes this into account. The four character couner code -C## is used for non-transferable identifiers and contains the signing identfier prefix and the signature [CESR]. Since the verification key can be extracted from the identifier prefix and the identifier can not be rotated, all that is required to validate the signature is the identifier prefix, the data signed and the signature.

3.3.3. Signatures with Transferable Identifiers

Transferable identifiers require full KEL resolution and verfication to determine the correct public key used to sign some content [KERI]. In addition, the attachment code used for transferable identifiers, -F## must specify the location in the KEL at which point the signature was generated [CESR]. To accomplish this, this counter code includes the identifier prefix, the sequence number of the event in the KEL, the digest of the event in the KEL and the indexed signatures (transferable identifiers support multiple public/private keys and require index signatures). Using all the values, one can verify the signature(s) by retrieving the KEL of the identifier prefix and determine the key state at the sequence number along with validating the digest of the event against the actual

event. Then using the key(s) at the determined key state, validate the signature(s).

3.4. Additional Count Codes

This specification adds two Counter Four Character Codes to the CESR Master Code Table for attaching and grouping transposable signatures on SAD and nested SAD content. The first code (-J##) is reserved for attaching a SAD path and the associated signatures on the content at the resolution of the SAD Path (either a SAD or its associated SAID). The second reserved code (-K##) is for grouping all SAD Path signature groups under a root path for a given SAD. The root path in the second grouping code provides signature attachment transposability for embedding SAD content in other messages.

3.4.1. SAD Path Signature Group

The SAD Path Signature Group provides a four character counter code, -J##, for attaching an encoded variable length SAD Path along with either a transferable index signature group or non-transferable identifer receipt couplets. The SAD Path identifies the content that this attachment is signing. The path must resolve to either a nested SAD (map) or a SAID (string) of an externally provided SAD within the context of the SAD and root path against which this attachment is applied. Using the following ACDC SAD embedded in a KERI exn message:

```
{
  "v": "KERI10JSON00011c_",
  "t": "exn",
  "dt": "2020-08-22T17:50:12.988921+00:00",
  "r": "/credential/offer",
  "a": {
    "credential": { // SIGNATURE TARGET OF TRANSPOSED SAD PATH GROUP
      "v": "ACDC10JSON00011c_",
      "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
      "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
      "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
      "a": {
        "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
        "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
        "dt": "2021-06-09T17:35:54.169967+00:00",
        "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
        "LEI": "254900OPPU84GM83MG36",
        "personal": {
          "legalName": "John Doe",
          "home": "Durham"
        }
      }
    }
  }
}
```

the following signature applies to the nested credential SAD signed by a transferable identifier using the transferable index signature group. The example is annotated with spaces and line feeds for clarity and an accompanied table is provided with comments.

```
-JAB
6AAEAAA-a-credential
-FAB
E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y
-EAB0AAAAAAAAAAAAAAAAAAAAAB
EwmQtlcszNoEIDfqD-Zih3N6o5B3humRKvBBln2juTEM
-AAD
AA5267UlFg1jHee4Dauht77SzGl8WUC_0oimYG5If3SdIOSzWM8Qs9SFajAilQcozXJVnbkY
ABBgeqntZW3Gu4HL0h3odYz6LaZ_SMfmITL-Btoq_7OZFe3L16jmOe49Ur108wH7mnBaq2E_
ACTD7NDX93ZGTkZBBuSeSGsAQ7u0hngpNTZTK_Um7rUZGnLRNJvo5oOnnC1J2iBQHuxoq8Py
```

code	description
-JAB	SAD path signature group counter code 1 following the group
6AAEAAA-a-credential	encoded SAD path designation
-FAB	Trans Indexed Sig Groups counter code 1 following group

code	description
E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y	trans prefix of signer for sigs
-EAB0AAAAAAAAAAAAAAAAAAAAAB	sequence number of est event of signer's public keys for sigs
EwmQtlcszNoEIDfqD-Zih3N6o5B3humRKvBBln2juTEM	digest of est event of signer's public keys for sigs
-AAD	Controller Indexed Sigs counter code 3 following sigs
AA5267...4AQ	sig 0
ABBgeq...pAQ	sig 1
ACTD7N...2Cg	sig 2

Table 3

The next example demostrates the use of a non-transferable identifier to sign SAD content. In this example, the entire nested SAD located at the a field is signed by the non-transferable identfier:

```
-JAB
5AABAA-a
-CAB
BmMfUwIOywRkyc5GyQXfgDA4UOAMvjvnXcaK9G939ArM
0BT7b5PzUBmts-lblgOBzdThIQjKCbq8gMinhymgr4_dD0JyfN6CjZhsOqqUYFmRhABQ-vPy
```

code	description
-JAB	SAD path signature group counter code 1 following the group
5AABAA-a	encoded SAD path designation
-CAB	NonTrans witness receipt couplet
BmMfUwIOywRkyc5GyQXfgDA4UOAMvjvnXcaK9G939ArM	non-trans prefix of signer of sig
0BT7b5... aBg	sig

Table 4

3.4.2. SAD Path Groups

The SAD Path Group provides a four character counter code, -K##, for attaching encoded variable length **root** SAD Path along with 1 or more SAD Path Signature Groups. The root SAD Path identifies the root context against which the paths in all included SAD Path Signature Groups are resolved. When parsing a SAD Path Group, if the root path is the single - character, all SAD paths are treated as absolute paths. Otherwise, the root path is prepended to the SAD paths in

each of the SAD Path Signature Groups. Given the following snippet of a SAD Path Group:

```
-KAB6AABAAA--JAB5AABAA-a...
```

The root path is the single - character meaning that all subsequent SAD Paths are absolute and therefore the first path is resolved as the a field of the root map of the SAD as seen in the following example:

```
{
  "v": "ACDC10JSON00011c_",
  "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a": {  // SIGNATURE TARGET OF SAD PATH GROUP
    "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
    "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
    "dt": "2021-06-09T17:35:54.169967+00:00",
    "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
    "LEI": "254900OPPU84GM83MG36",
    "personal": {
      "legalName": "John Doe",
      "city": "Durham"
    }
  }
}
```

3.4.2.1. Transposable Signature Attachments

To support nesting of signed SAD content in other SAD content the root path of SAD Path Groups or the path of a SAD Path Signature Group provides transposability of CESR SAD signatures such that a single SAD Path Signature Group or an entire SAD Path Group attachment can be transposed across envelope boundaries by updating the single path or root path to indicate the new location. Extending the example above, the SAD content is now embedded in a KERI exn event message as follows:

```
{
  "v": "KERI10JSON00011c_",
  "t": "exn",
  "dt": "2020-08-22T17:50:12.988921+00:00"
  "r": "/credential/offer"
  "a": {
    "v": "ACDC10JSON00011c_",
    "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
    "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
    "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
    "a": { // SIGNATURE TARGET OF TRANSPOSED SAD PATH GROUP
      "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
      "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
      "dt": "2021-06-09T17:35:54.169967+00:00",
      "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
      "LEI": "254900OPPU84GM83MG36",
      "personal": {
        "legalName": "John Doe",
        "city": "Durham"
      }
    }
  }
}
```

The same signature gets transposed to the outer exn SAD by updating the root path of the -K## attachment:

```
-KAB5AABAA-a-JAB5AABAA-a...
```

Now the SAD Path of the first signed SAD content resolves to the a field of the a field of the streamed exn message

3.5. Small Variable Raw Size SAD Path Code

The small variable raw side code reserved for SAD Path encoding is A which results in the addition of 3 entries (4A##, 5A## and 6A##) in the Master Code Table for each lead byte configuration. These codes and their use are discussed in detail in CESR Encoding for SAD Path Language.

4. Nested Partial Signatures

Additional signatures on nested content can be included in a SAD Path Group and are applied to the serialized data at the resolution of a SAD path in a SAD. Signatures can be applied to the SAID or an entire nested SAD. When verifying a CESR Proof Signature, the content at the resolution of the SAD path is the data that was signed. The choice to sign a SAID or the full SAD effects how the

data may be used in presentations and the rules for verifying the signature.

4.1. Signing Nested SADs

When signing nested SAD content, the serialization used at the time of signing is the only serialization that can be used when presenting the signed data. When transposing the signatures and nesting the signed data, the enveloping SAD must use the same serialization that was used to create the signatures. This is to ensure that all signatures apply to the data over the wire and not a transformation of that data. The serialization can be determined from the version field (v) of the nested SAD or any parent of the nested SAD as they are guaranteed to be identical. Consider the following ACDC Credential SAD:

```
{
  "v": "ACDC10JSON00011c_",
  "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a": {   // SIGNATURE TARGET OF SAD PATH GROUP
    "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
    "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
    "dt": "2021-06-09T17:35:54.169967+00:00",
    "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
    "LEI": "254900OPPU84GM83MG36",
    "personal": {
      "d": "E2X8OLaLnM0XRQEYgM5UV3bZmWg3UUn7CP4SoKkvsl-s",
        "first": "John",
        "last": "Doe"
    }
  }
}
```

To sign the SAD located at the path -a, JSON serialization would be used because the SAD at that path does not have a version field so the version field of its parent is used. The serialization rules (spacing, field ordering, etc) for a SAD would be used for the SAD and the serialization encoding format and the signature would be applied to the bytes of the JSON for that map. Any presentation of the signed data must always include the fully nested SAD. The only valid nesting of this credential would be as follows:

```
{
  "v": "KERI10JSON00011c_",
  "t": "exn",
  "dt": "2020-08-22T17:50:12.988921+00:00"
  "r": "/credential/apply"
  "a": {
    "v": "ACDC10JSON00011c_",
    "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
    "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
    "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
    "a": {   // FULL SAD MUST BE PRESENT
      "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
      "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
      "dt": "2021-06-09T17:35:54.169967+00:00",
      "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
      "LEI": "254900OPPU84GM83MG36",
      "legalName": {
        "d": "E2X8OLaLnM0XRQEYgM5UV3bZmWg3UUn7CP4SoKkvsl-s",
        "first": "John",
        "last": "Doe"
      }
    }
  }
}
```

4.2. Signing SAIDs

Applying signatures to a SAD with SAIDs in place of fully expanded nested SAD content enables compact credentials for domains with bandwidth restrictions such as IoT. Consider the following fully expanded credential:

```
{
    "v": "ACDC10JSON00011c_",
    "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
    "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
    "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
    "a": {
      "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
      "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
      "dt": "2021-06-09T17:35:54.169967+00:00",
      "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
      "LEI": "254900OPPU84GM83MG36",
      "legalName": {
        "d": "E2X8OLaLnM0XRQEYgM5UV3bZmWg3UUn7CP4SoKkvsl-s",
        "n": "sKHtYSiCdlibuLDS2PTJg1AZXtPhaySZ9O3DoKrRXWY",
        "first": "John
        "middle": "William"
        "last": "Doe"
      },
      "address": {
        "d": "E-0luqYSg6cPcMFmhiAz8VBQObZLmTQPrgsr7Z1j6CA4",
        "n": "XiSoVDNvqV8ldofPyTVqQ-EtVPlkIIQTln9Ai0yI05M",
        "street": "123 Main St",
        "city": "Salt Lake City",
        "state": "Utah",
        "zipcode": "84157"
      },
      "phone": {
        "d": "E6lty8H2sA_1acq8zg89_kqF194DbF1cDpwA7UPtwjPQ",
        "n": "_XKNVntbcIjp12DmsAGhv-R7JRwuzjD6KCHC7Fw3zvU"
        "mobile": "555-121-3434",
        "home": "555-121-3435",
        "work": "555-121-3436",
        "fax": "555-121-3437"
      }
    }
  }
}
```


The three nested blocks of the a block legalName, address and phone are SADs with a SAID in the d field and are candidates for SAID replacement in an issued credential. A compact credential can be created and signed by replacing those three nested blocks with the SAID of each nested SAD. The schema for this verifiable credential would need to specify conditional subschema for the field labels at each nesting location that requires the full schema of the nested SAD or a string for the SAID. The commitment to a SAID in place of a SAD contains nearly the same cryptographic integrity as a commitment to the SAD itself since the SAID is the qualified cryptographic material of a digest of the SAD. The same credential could be

converted to a compact credential containing the SAIDs of each nested block and signed as follows:


```
{
   "v": "ACDC10JSON00011c_",
   "d": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
   "i": "EmkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
   "s": "E46jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
   "a": {
     "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
     "i": "EQzFVaMasUf4cZZBKA0pUbRc9T8yUXRFLyM1JDASYqAA",
     "dt": "2021-06-09T17:35:54.169967+00:00",
     "ri": "EymRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
     "LEI": "254900OPPU84GM83MG36",
     "legalName": "E2X8OLaLnM0XRQEYgM5UV3bZmWg3UUn7CP4SoKkvsl-s",
     "address": "E-0luqYSg6cPcMFmhiAz8VBQObZLmTQPrgsr7Z1j6CA4",
     "phone": "E6lty8H2sA_1acq8zg89_kqF194DbF1cDpwA7UPtwjPQ"
   }
}
```


It is important to note that if this version of the credential is the one issued to the holder and the signature over the entire credential is on the serialized data of this version of the credential it is the only version that can be presented. The full SAD data of the three nested blocks would be delivered out of band from the signed credential. The top level schema would describe the blocks with conditional subschema for each section. The credential signature becomes a cryptographic commitment to the contents of the overall credential as well as the content of each of the blocks and will still validate the presented credential with significantly less bandwidth.

With this approach, credential presentation request and exchange protocols can be created that modify the schema with the conditional subschema, removing the conditions that allow for SAIDs in place of the required (or presented) nested blocks. The modified schema can be used in such a protocol to indicate the required sections to be delivered out of bounds or as a commitment to provide the nested blocks after the crendential presentation has occurred.

5. Conventions and Definitions

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.

6. Security Considerations

TODO Security

7. IANA Considerations

The Internet Assigned Numbers Authority (IANA) is a standards organization that oversees global IP address allocation, autonomous system number allocation, root zone management in the Domain Name System (DNS), media types, and other Internet Protocol-related symbols and Internet numbers.

This document has no IANA actions.

8. References

8.1. Normative References

[ACDC] Smith, S., "Authentic Data Chained Containers", 2021, <https://datatracker.ietf.org/doc/draft-ssmith-acdc/>.

[CESR] Smith, S., "Composable Event Streaming Representation (CESR)", 2021, <https://datatracker.ietf.org/doc/draft-ssmith-cesr/>.

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

[SAID] Smith, S., "Self-Addressing IDentifier (SAID)", 2021, <https://datatracker.ietf.org/doc/draft-ssmith-said/>.

8.2. Informative References

[CBOR] "CBOR Mapping Object Codes", n.d., <https://en.wikipedia.org/wiki/CBOR>.

[JSON] "JavaScript Object Notation Delimeters", n.d., <https://www.json.org/json-en.html>.

[JSONPath] Gössner, S., Normington, G., and C. Bormann, "JSONPath - Query expressions for JSON", 25 October 2021, <https://datatracker.ietf.org/doc/draft-ietf-jsonpath-base/>.

[KERI] Smith, S., "Key Event Receipt Infrastructure (KERI)", 2021, <https://arxiv.org/abs/1907.02143>.

[MGPK] "Msgpack Mapping Object Codes", n.d., <https://github.com/msgpack/msgpack/blob/master/spec.md>.

[RFC6901] Bryan, P. C., Zyp, K., and M. Nottingham, "JavaScript Object Notation (JSON) Pointer", 2003, <https://datatracker.ietf.org/doc/html/rfc6901>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", 4 December 2020, <https://datatracker.ietf.org/doc/rfc8949/>.

Acknowledgments

Dr Sam Smith, Kevin Griffin and the Global Legal Entity Identifier Foundation (GLEIF)

Author's Address

Phil Feairheller
GLEIF

Email: Philip.Feairheller@gleif.org