CESR
Composable Event Streaming Representation

Composable cryptographic material primitives and groups in dual text and binary streaming protocol design

Samuel M. Smith Ph.D.
Sam@prosapien.com
2022/04/26

KERI

Resources

Documentation:

https://github.com/WebOfTrust/ietf-cesr
https://github.com/WebOfTrust/ietf-cesr-proof
https://github.com/trustoverip/tswg-acdc-specification
https://docs.google.com/presentation/d/1mO1EZa9BcjAjWEzw7DWi124uMfyNyDeM3HuajsGNoTo/edit#slide=id.g124effd16ae_0_164
https://keri.one/keri-resources/
https://arxiv.org/abs/1907.02143 (KERI White Paper)

Community: (meetings, open source code, IETF internet drafts)

https://github.com/WebOfTrust
https://github.com/WebOfTrust/keri
https://wiki.trustoverip.org/display/HOME/ACDC+%28Authentic+Chained+Data+Container%29+Task+Force

Protocol Formats

Internet Protocols

Binary

UDP, TCP, DNS, RTP, RTCP, NTP, SNMP, BGP, BGMP, ARP, IGMP, RIP, PING, WebRTC

Text *(line based)*

Syslog, SMTP, POP, Telnet, NNTP, RTSP, IRC

Text *(header framed)*

HTTP, SIP

Inflection Point in Protocol Design *(dual representation)*

XML, JSON (text self describing map)

MGPK, CBOR (binary self describing map)

Cryptographic Protocols

JSON or Hybrid JSON/MGPK

RAET, Secure Scuttlebutt, DIDCom

Fixed Binary

RLPx (Ethereum)

Bitcoin

Flexible Concatenable Binary Crypto Material

Noise (Signal)

Flexible Composable Concatenable Text/Binary Crypto Material

CESR (KERI) Composable Event Streaming Representation

Binary vs Text Based Protocol Features

Binary Format

Advantages:

Compact, efficient, and performant

Disadvantages:

Difficult to develop, test, debug (over the wire), prove compliance, and extremely difficult to fix and version

Requires custom tooling especially over the wire debug, difficult to understand, and *hard to gain adoption.*

Text Format (especially self-describing hash map based)

Advantages:

Easy to develop, test, debug (especially over the wire), and prove compliance, and extremely easy to fix and version

Requires little custom tooling especially over the wire debug, easy to understand and *easy to gain adoption.*

Disadvantages:

Verbose, Inefficient, Non-performant

Hybrid Both Text & Binary Formats (self describing map, Text=JSON and Binary=MGPK or CBOR)

Advantages:

Zero cost to switch between text and binary. Text for development and adoption. Binary for production use.

Easy to develop, test, debug (especially over the wire), and prove compliance when text

Requires little custom tooling especially over the wire debug, easy to understand

Extremely easy to fix and version

Easy to adopt as text with no additional barrier to binary adoption

Fairly compact when binary, Fairly efficient when binary, fairly performant when binary

Composable Concatenable (Text/Binary) Event Streaming Representation (CESR)

Hybrid Flexible Concatenated Compact Text Base64 & Binary Formats

Advantages:

Composable *(any concatenated block in one format may be converted as a block to the other without loss, round trippable)*

Zero cost to switch between text and binary. Text for development and adoption. Binary for production use.

Flexible concatenation of heterogenous crypto material that preserves byte/char boundaries between primitives

Pipelined parsing and processing

Stream or Datagram

Fully qualified self framing derivation codes for primitives.

Fully qualified self framing count codes for groups of primitives or groups of groups.

Fully qualified self framing count codes for pipelining groups.

More compact text and binary than hybrid text and binary self describing map based formats

Fairly Easy to develop, test, debug (especially over the wire), and prove compliance when text

Fairly easy to fix and version

Requires little custom tooling especially over the wire debug when text

Fairly easy to understand when text

Archivable audit compliant text format

Fairly easy to adopt as text with no additional barrier to binary adoption

Compact when binary, Efficient when binary, Performant when binary

Disadvantages:

Not as but almost as compact, efficient, performant as non-composable tuned binary.

Three native domains and formats

Raw Domain (separated code and raw binary) (cryptographic operations)

Namespace Domain (fully qualified text) (name-spaceable text) (streamable text) (archivable text) (envelopable text)

Compact Domain (fully qualified binary) (streaming binary)

Raw Domain = Raw binary representation that crypto libraries use

Compact Domain = Fully qualified binary representation of cryptographic material for efficient over the wire streaming

Namespace Domain = Fully qualified text representations of cryptographic material: identifiers, digests, signatures etc

Includes any textual use of cryptographic material, Documents, VCs, Archives, Audit logs etc

Usable in over the wire streaming for development and debug

```
BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0
did:keri:prefix[:options][/path][?query][#fragment]
did:keri:BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0/path/to/resource?name=bob#really
```

```
{
  "id": "did:keri:Eewfge7gf78sgfivsf/vLEIGLEIFCredential", // DID of the verifiable credential itself
  "type":
  [
    "VerifiableCredential",
    "vLEIGLEIFCredential"
  ], // type of the verifiable credential
  "issuer": "did:keri:Eewfge7gf78sgfivsf/DelegatedGLEIFRootID", // issuer of the verifiable credential
  "issuanceDate": "2021-02-10T17:50:24Z", // date of issuance
  "credentialSubject":
  {
    "id": "did:keri:Eewfge7gf78sgfivsf/GLEIFRootID", // DID of the issuee / holder
    "lei": "506700GE1G29325QX363" // LEI
  },
  "proof":
  {
    "signature": "AAmdI8OSQkMJ9r-xigjEByEjIua7LHH3AOJ22PQKqljMhuhcgh9nGRcKnsz5KvKd7K_H9-1298F4Id1DxvIoEmCQ"
 }
}
```

Round Trippable Closed loop Transformation

Namespace to/from Compact to/from Raw Domain to/from Namespace

Fully qualified means prepended derivation code.

In namespace domain, readability is enhanced if prepended derivation code is stable and is not changed by post-pended crypto material value

Namespace domain, T, is fully qualified Base64. Streaming binary domain, S, is fully qualified base 2 equivalent (conversion) of T.

Composability Property: transformation (round trip) between T and S of concatenated primitives does not cross primitive boundaries. Separable parseability is preserved.

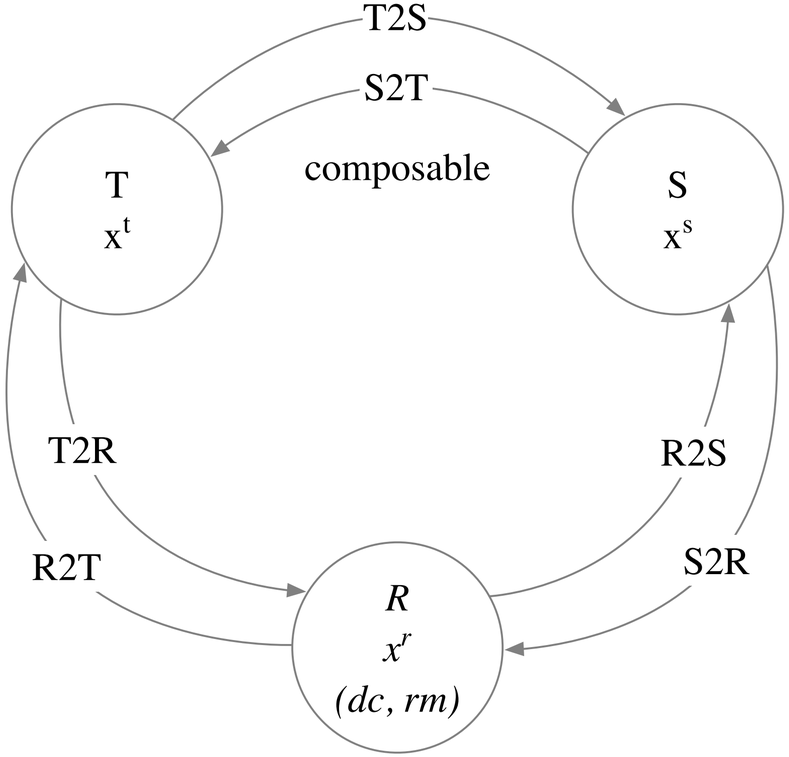
Normally composability requires pad characters (pad bytes) on each Base64 (Base2) primitive.

KERI replaces pad characters with prepend derivation codes whose length preserves composability.

By comparison did:key is not stable, did:peer is, neither are composable.

In general Multi-Codec is not stable nor composable except fro serendipitous combinations of code and value

(dc, rm) -> R2T -> t -> T2S -> s -> S2R -> (dc, rm)
(dc, rm) -> R2S -> s -> S2T -> t -> T2R -> (dc, rm)



$$x^t$$

$$x^s$$

$$x^t = T\left(x^s\right)$$

$$x^s = S\left(x^t\right)$$

$$x^t + y^t = T\left(x^s + y^s\right)$$

$$x^s + y^s = S\left(x^t + y^t\right)$$

$$S\left(\sum_{i=0}^{N-1} x_i^t\right) = \sum_{i=0}^{N-1} x_i^s$$

$$T\left(\sum_{i=0}^{N-1} x_i^s\right) = \sum_{i=0}^{N-1} x_i^t$$

Code Tables

3 classes of stable composable derivation codes:

Basic material primitives,

Indexed signature primitives or variable length values,

Grouping count codes.

Tritet Parsing

CESR Parser Tritet Table

Starting Tritet	Serialization	Character
0b000		
0b001	CESR *T* Domain Count (Group) Code	-
0b010	CESR *T* Domain Op Code	_
0b011	JSON	{
0b100	MGPK	
0b101	CBOR	
0b110	MGPK	
0b111	CESR *B* Domain	

CESR Code Table Schemes

Selector	Selector	Type Chars	Value Size Chars	Code Size	Lead Bytes	Pad Size	Format (Minimal Size)
[A-Z,a-z]		1*	0	1	0	1	$&&&
0		1	0	2	0	2	0$&&
1		3	0	4	0	0	1$$$&&&&
2		3	0	4	1	1	2$$$&&&&
3		3	0	4	2	2	3$$$&&&&
4		1	2	4	0	0	4$##&&&&
5		1	2	4	1	1	5$##&&&&
6		1	2	4	2	2	6$##&&&&
7		3	4	8	0	0	7$$$####&&&&
8		3	4	8	1	1	8$$$####&&&&
9		3	4	8	2	2	9$$$####&&&&
-	[A-Z,a-z]	1*	0	4	0	0	-$##
-	0	2	0	8	0	0	-0$$####
_	TBD	TBD	TBD	TBD	TBD	TBD	_

* selector character is also type character¶
$ means type code character from subset of Base64 [A-Z,a-z,0-9,-,_].
# means a Base64 digit as part of a base 64 integer that determines the number of following quadlets or triplets in the primitive or when part of a count code, the count of following primitives or groups of primitives.
& represents one or more Base64 value characters representing the converted raw binary value included lead bytes when applicable. The actual number of chars is determined by the prep-ended text code.

Parsing Size Tables

Size Table Selector

selector	hs
B	1
0	2
5	2

Parse Size Table

hard sized index	hs	ss	vs	fs	ls	ps
B	1	0	43*	44	0	1
0B	2	0	86*	88	0	2*
5A	2	2	#	#	1	1*

`*` size may be calculated from other sizes.
`#` size may be calculated from extracted code characters given by other sizes.¶

hs means hard size in chars.
ss means soft size in chars.
cs means code size where *cs = hs + ss*.
vs means value size in chars.
fs means full size in chars where *fs = hs + ss + vs*.
ls means lead size in bytes.
ps means pad size in chars.
rs means raw size in bytes of binary value.
bs means binary size in bytes where *bs = as + rs*.¶

Special Context Tables

Special Indexed Primitive Table

Selector	Selector	Type Chars	Index Chars	Code Size	Lead Bytes	Pad Size	Format
[A-Z,a-z]		1*	1	2	0	2	$#&&
0		1	2	4	0	0	0$##&&&&

* selector character is also type character

$ means type code character from subset of Base64 [A-Z,a-z,0-9,-,_].
# means a Base64 digit as part of a base 64 integer that determines the index.
& represents one or more Base64 value characters representing the converted raw binary value included lead bytes when applicable. The actual number of chars is determined by the prep-ended text code.

Master Table

Master Table

Code	Description	Code Length	Count, Size, or Index Length	Leader Length	Total Length
	Basic One Character Codes				
A	Random seed of Ed25519 private key of length 256 bits	1			44
B	Ed25519 non-transferable prefix public signing verification key. Basic derivation.	1			44
C	X25519 public encryption key. May be converted from Ed25519 public signing verification key.	1			44
D	Ed25519 public signing verification key. Basic derivation.	1			44
E	Blake3-256 Digest. Self-addressing derivation.	1			44
F	Blake2b-256 Digest. Self-addressing derivation.	1			44
G	Blake2s-256 Digest. Self-addressing derivation.	1			44
H	SHA3-256 Digest. Self-addressing derivation.	1			44
I	SHA2-256 Digest. Self-addressing derivation.	1			44
J	Random seed of ECDSA secp256k1 private key of length 256 bits	1			44
K	Random seed of Ed448 private key of length 448 bits	1			76
L	X448 public encryption key. May be converted from Ed448 public signing verification key.	1			76
M	Short value of length 16 bits	1			4

Master Table

Master Table

Code	Description	Code Length	Count, Size, or Index Length	Leader Length	Total Length
	Basic Two Character Codes				
0A	Random salt, seed, private key, or sequence number of length 128 bits	2			24
0B	Ed25519 signature. Self-signing derivation.	2			88
0C	ECDSA secp256k1 signature. Self-signing derivation.	2			88
0D	Blake3-512 Digest. Self-addressing derivation.	2			88
0E	Blake2b-512 Digest. Self-addressing derivation.	2			88
0F	SHA3-512 Digest. Self-addressing derivation.	2			88
0G	SHA2-512 Digest. Self-addressing derivation.	2			88
0H	Long value of length 32 bits	2			8
	Basic Four Character Codes				
1AAA	ECDSA secp256k1 non-transferable prefix public signing verification key. Basic derivation.	4			48
1AAB	ECDSA secp256k1 public signing verification or encryption key. Basic derivation.	4			48
1AAC	Ed448 non-transferable prefix public signing verification key. Basic derivation.	4			80
1AAD	Ed448 public signing verification key. Basic derivation.	4			80
1AAE	Ed448 signature. Self-signing derivation.	4			156
1AAF	Tag Base64 4 chars or 3 byte number	4			8
1AAG	DateTime Base64 custom encoded 32 char ISO-8601 DateTime	4			36
1AAH	X25519 Cipher Salt Cipher of 24 char Salt	4			100

Master Table

Master Table

Code	Description	Code Length	Count, Size, or Index Length	Leader Length	Total Length
	Variable Sized Character Codes				
4A	String Base64 Only Leader Size 0	2	2	0	
5A	String Base64 Only Leader Size 1	2	2	1	
6A	String Base64 Only Leader Size 2	2	2	2	
7AAA	String Base64 Only Leader Size 0	4	4	0	
8AAA	String Base64 Only Leader Size 1	4	4	1	
9AAA	String Base64 Only Leader Size 2	4	4	2	
	Indexed Two Character Codes				
A#	Ed25519 indexed signature	2	1		88
B#	ECDSA secp256k1 indexed signature	2	1		88
	Indexed Four Character Codes				
0A##	Ed448 indexed signature	4	2		156
0B##	Label Base64 chars of variable length L=N*4 where N is value of index total = L+4	4	2		Variable

Master Table

Master Table

Code	Description	Code Length	Count, Size, or Index Length	Leader Length	Total Length
	Counter Four Character Codes				
-A##	Count of attached qualified Base64 indexed controller signatures	4	2		4
-B##	Count of attached qualified Base64 indexed witness signatures	4	2		4
-C##	Count of attached qualified Base64 nontransferable identifier receipt couples pre+sig	4	2		4
-D##	Count of attached qualified Base64 transferable identifier receipt quadruples pre+snu+dig+sig	4	2		4
-E##	Count of attached qualified Base64 first seen replay couples fn+dt	4	2		4
-F##	Count of attached qualified Base64 transferable indexed sig groups pre+snu+dig + idx sig group	4	2		4
-U##	Count of qualified Base64 groups or primitives in message data	4	2		4
-V##	Count of total attached grouped material qualified Base64 4 char quadlets	4	2		4
-W##	Count of total message data grouped material qualified Base64 4 char quadlets	4	2		4
-X##	Count of total group message data plus attachments qualified Base64 4 char quadlets	4	2		4
-Y##	Count of qualified Base64 groups or primitives in group. (context dependent)	4	2		4
-Z##	Count of grouped material qualified Base64 4 char quadlets (context dependent)	4	2		4

Example

```
# Trans Indexed Sig Groups counter code 1 following group
-FAB

# trans prefix of signer for sigs
E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y

# sequence number of est event of signer's public keys for sigs
-EAB0AAAAAAAAAAAAAAAAAAAAAB

# digest of est event of signer's public keys for sigs
EwmQtlcszNoEIDfqD-Zih3N6o5B3humRKvBBln2juTEM

# Controller Indexed Sigs counter code 3 following sigs
-AAD

# sig 0
AA5267UlFg1jHee4Dauht77SzGl8WUC_0oimYG5If3SdIOSzWM8Qs9SFajAilQcozXJVnbkY5stG_K4NbKdNB4AQ

# sig 1
ABBgeqntZW3Gu4HL0h3odYz6LaZ_SMfmITL-Btoq_7OZFe3L16jmOe49Ur108wH7mnBaq2E_0U0N0c5vgrJtDpAQ

# sig 2
ACTD7NDX93ZGTkZBBuSeSGsAQ7u0hngpNTZTK_Um7rUZGnLRNJvo5oOnnC1J2iBQHuxoq8PyjdT3BHS2LiPrs2Cg
```

Stream Parsing Rules

Stream start, cold restart, message end, group end:

Examine tritet (3 bits).

Each stream must start (restart) with one of four things:

- Framing count code in either Base64 or Binary.
- Framing opcode in either Base64 or Binary
- JSON encoded mapping.
- CBOR encoded Mapping.
- MGPK encoded mapping.
- (1 remaining unused tritet)

A parser merely needs to examine the first tritet (3 bits) of the first byte of the stream start to determine which one of the five it is.

When the first tritet indicates its JSON, CBOR, or MGPK, then the included version string provides the remaining and confirming information needed to fully parse the associated encoded message.

When the first tritet is a framing code then, the remainder of framing code itself will include the remaining information needed to parse the attached group.

The framing code may be in either Base64 or binary.

At the end of the stream start, the stream must resume with one of the 5 things, either a new JSON, CBOR, or MGPK encoded mapping or another of two types of framing codes expressed in either Base64 or binary.

Why Composability

Verifiable Text Stream = Verifiable Binary Stream (no loss of verifiability in mass conversion)

Amount of cryptographic material in attachments far exceeds cryptographic material in message bodies. Controller signatures, witness signatures, other receipt signatures, endorsement signatures.

Replay of KELs must replay messages (key events) plus signatures

Want this replay to be compact, performant, and supported by bare metal protocols (TCP, UDP)

Verifiable Credential world is Text

Verifiable Authentic Data world human facing side is Text

Verifiable Digitally Signed Contract world is Text

Verifiable Audit Trail World is Text

Archival Preservation

The ISO ISO 14641:2018 standard for the preservation of electronic documents, lists four important features of a legally defensible archive [33][34]. These are long-term preservation, data integrity, data security, and traceability.
A KERI Key Event Receipt Log (KERL) is already in a form that provides data integrity, data security, and traceability.
All that is needed is to ensure long-term preservation capability.
The standard formats for long-term document archival are by-in-large text based (with some exceptions) [30].
What this means is that a KERI event log stream in a native text format is inherently compatible with archival requirements. Indeed because a KERL text stream with composition operators only uses the Base64URL character set, that is,[A-Z,a-z,0-9,-,_]

`Annotated KELs`

Any of the other characters in the ASCII set may be used to loss-lessly annotate a KERI text event stream.
These include white space characters.
Primitives and groups of framed primitives within the text KERL could then be line delimited and white spaced and comments added using the “#” symbol for example.
A text parser could easily strip all non Base-64 characters, line delimiters and comments to reconstitute the KERL stream which could then be converted en mass for transmission.
The archivist never need access or convert to the raw binary format.

Legally Binding Digital Signatures (Contracts)

The relevant legislation for legally compliant electronic signatures are the USA Electronic Signatures in Global and National Commerce Act (ESIGN), the USA Uniform Eletronic Transactions Act (UETA) and the EU Regulation for Electronic Identification and Electronic Trust Services (eIDAS) [27][28][29].
The legislations have similar conditions at least those relevant to KERI.
When the set of legally defined conditions are met, a cryptographic digital signature based on asymmetric key pairs has legal standing.
Key pairs used to control a KERI self-certifying identifier may also be used to create legally binding electronic signatures on electronic documents.
Establishment of control authority is provided by cryptographically verifiable proof of key state, that is, a proof that a given set of key pairs are the authoritative ones.
This proof is expressed as a KERI KERL (Key Event Receipt Log) which is a hash chained signed data structure.
The most relevant condition (to KERI) that a legally binding signature must satisfy is proof or assurance that the entity creating the electronic signature is also the entity that is the controller of the associated private keys.
A KERI KEL may be used to support that proof or assurance.
An annotated text domain KEL could be attached to the signed legal document as an appendix or provided to an electronic signature notary.
This text based annotated KEL appendix could be archived with all the associated legal documents.
This greatly facilitates the broader adoption of electronic signatures and KERI.
This could also help reduce trust transaction costs for the authentic data economy.

Audit Trails

An audit log is used to provenance something.
An annotated text domain KERL could be used to support a securely attributed tamper evident archival audit trail [24][25][26].
Composability would allow archival in streaming text form while also enabling efficient transmission in streaming binary form.
Electronic Code of Federal Regulations: Electronic Signatures (E-CFR) regulation requires non-repudiable audit trial attribution[25]:
E-CFR
(e) Use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records and shall be available for agency review and copying.
(f) Use of operational system checks to enforce permitted sequencing of steps and events, as appropriate.
(g) Use of authority checks to ensure that only authorized individuals can use the system, electronically sign a record, access the operation or computer system input or output device, alter a record, or perform the operation at hand.
(j) The establishment of, and adherence to, written policies that hold individuals accountable and responsible for actions initiated under their electronic signatures, in order to deter record and signature falsification.
The Alliance for Telecommunications Industry Solutions (ATIS) provides various definitions for audit trail as follows [26]:
A set of records that collectively provide documentary evidence of processing used to aid in tracing from original transactions forward to related records and reports, and/or backwards from records and reports to their component source transactions.
KERI supports secure attribution via non-repudiable signatures on any data in such a compliance log.
The key events in the text KEL may be interspersed in the audit trail so that the timing of key events that rotate keys may be correlated to the audit log signed data. The audit trail may be viewed as an annotated KEL.
A text processor could extract the KERL from any archived audit trail and then convert it to streaming binary for compact transmission to a processor to perform the cryptographic verification operations.
This provides efficient scalable provenance.