Enabling global identity
Protecting digital trust

vLEI Q&A

Public
Version 1.2
2023-08-30

Questions and Answers regarding the verifiable LEI (vLEI)

How can the LEI and the vLEI solve identity and authentication challenges?

Using the LEI within the vLEI will address the need for secure, certain and verifiable organizational identity. When dealing business to business, business to consumer and private sector to public sector, it is critical to know with certainly the entity with which you are dealing.

We all are familiar with the negative realities of not knowing with certainty the entity with which you are dealing - identity impersonation, fraud, social engineering in the form of phishing/smishing and robocalls.

The LEI as a unique, permanent and persistent identifier based on an international standard can be useful as our world undergoes digital transformation

- Digital engagement
- Digital transactions
- In both 'local' and 'global' interactions

With the vLEI, there an opportunity for secure, verifiable organizational identity to be realized more cheaply, with certainty, by leveraging improvements in automated cryptographic verification with the potential to decrease, and in some cases, eliminate risks.

The vLEI will give public sector government entities, companies and consumers worldwide the capacity to use the LEI's non-repudiable identification data in a growing number of digital business activities, such as approving business transactions and contracts, onboarding customers, transacting within import/export and supply chain business networks and submitting regulatory filings and reports.

More information is included in the Press Release about the launch of the vLEI Ecosystem and Infrastructure in December 2022: https://www.gleif.org/en/newsroom/press-releases/first-suite-of-vlei-services-to-enable-digital-signing-and-automated-verification-of-corporate-caller-ids

What makes GLEIF the 'Root of trust' for the vLEI?

Governance has been an important cornerstone of the Global LEI System and in the establishing GLEIF's role in operating and managing this system of LEIs. GLEIF is subject to oversight by global regulators and according to our statutes has established strong governance for the process of validating legal entities for LEI issuance and the maintenance and updating of legal entity reference data

Building on this strong governance which begins with the issuance and maintenance of the LEI itself, GLEIF will be the anchor of the vLEI ecosystem, at the root of the governance that will position the LEI as a key component in building a trust layer for identification and verification of legal entities as the LEI allows authentication that the legal entity is indeed who it claims to be and that those who act on its behalf, can.

To document the governance for the vLEI Ecosystem and Infrastructure, GLEIF chose the to leverage the Ecosystem Governance Framework of the Trust over IP Foundation *(link to ToIP website),* which GLEIF joined as a Contributor member at its inception.

The ToIP Technology Stack covers technical trust. The ToIP Governance Stack covers human trust which includes an Ecosystem Governance Framework which documents requirements at the business, legal, and social layers. To the Self-Sovereign Identity experts engaged in ToIP, the LEI, as a key component in building a trust layer and technical infrastructure for identification and verification of organizations, is the basis for a textbook example of being able to satisfy organizational identity requirements for any number of use cases or domains through the use of vLEIs.

Also, see the related question in the Technical Question section below, '**How was GLEIF established as the root-of-trust for the vLEI ecosystem?'**

Technical Questions regarding the vLEI Ecosystem Infrastructure

What does GLEIF see as important technical features that the technical solution for vLEI ecosystem infrastructure must have?

GLEIF think that the technical solution for vLEI ecosystem infrastructure must be interoperable and portable.

- Open source/non-proprietary
- Interoperable
- Works with all technical solutions – blockchain/Distributed Ledger Technology (DLT) as well as cloud-based solutions

Incorporating these features would allow the vLEI infrastructure to be integrated with existing implementations/use cases using blockchain/DLT technology without the users needing to abandon their existing technical solutions. The new approach also could accommodate existing and new implementations not based on blockchain/DLT technology. GLEIF anticipates interfaces of the vLEI technical infrastructure to Hyperledger, Ethereum, Quorum and Corda based networks as well as cloud applications.

What is KERI (Key Event Receipt Infrastructure)?

Key Event Receipt Infrastructure (KERI) is a ledgerless approach to identity that enables a universal decentralized key management infrastructure (DKMI). KERI takes the objectives that conventional Public Key Infrastructure (PKI) tries to achieve with centralized, administrative roots of trust and achieves them with decentralized, cryptographic roots of trust. This solves many problems: security, privacy, scalability, performance, cost, governance, and more. *(Link to KERI white paper: Smith, S. M., "Key Event Receipt Infrastructure (KERI) Design", Revised 2020/09/06, 2019/07/03)*

What Decentralized Identifier (DID) methods will the vLEI infrastructure use?

As part of supporting the vLEI ecosystem, GLEIF is sheppanding the development of the did:keri DID Method. This will be a native DID method that will leverage the GLEIF provided Witnesses and Watchers (these KERI roles are described below) for discovery. The native did:keri method provides a compatibility hook for DID Resolvers allowing any application that requires a fully complaint DID Document can receive one for the KERI identifiers of the vLEI ecosystem. Integration with KERI Witnesses and Watchers provides discovery of any identifier that is known to those components. Ecosystems (like the vLEI) can have global "Super Watchers" that monitor the entire ecosystem to provide visibility of all relevant identifiers.

A vLEI "Super Watcher" will be made available by GLEIF to provide this access for anyone interested in resolving the key state of any identifier. Anyone interested in receiving DID Documents for a vLEI identifier can run an instance of the did:keri did method resolver pointing at this vLEI "Super Watcher".

Click here for the link to the W3C DID standard.

What are the roles in the KERI design?

Controller
A Controller is a controlling entity of an identifier. At any point in time an identifier has at least one but may have more than one controlling entity. Let L be the number of controlling entities. This set of controlling entities constitute the Controller. All proper key management events on the identifier must include a signature from the sole controlling entity when there is only one member in the set of controlling entities or a least one signature from one of the controlling entities when there is more than one. This signature may be expressed as a single collective signature when a collective signing scheme is used. Without loss of generality, when the context is unambiguous, the term Controller may refer either to the whole set or a member of the set of controlling entities.

Typically, when there is more than one controlling entity, control is established via L signatures, one from each entity Controller. This is called multi-signature or multi-sig for short. Alternatively, with a K of L threshold control scheme, where $K \leq L$, control is established via any set of at least K signatures each one from a subset of at least size K of the L Controllers. A more sophisticated scheme may use fractional weighted multiple signatures. These multiple signatures under a threshold control scheme may be expressed as a single collective threshold signature from an appropriate collective threshold signing scheme. The description of the KERI protocol assumes the simplest case of individual not collective signatures, but it is anticipated that the protocol may be extended to support collective multi-signature schemes.

Verifier
A Verifier is an entity or component that cryptographically verifies the signature(s) on an event message. In order to verify a signature, a Verifier must first determine which set of keys are or were the controlling set for an identifier when an event was issued. In other words, a Verifier must first establish control authority for an identifier. For identifiers that are declared as non-transferable at inception this control establishment merely requires a copy of the inception event for the identifier. For identifiers that are declared transferable at inception this control establishment requires a complete copy of the sequence of key operation events (inception and all rotations) for the identifier up to the time at which the statement was issued.

Validator
A Validator is an entity or component that determines that a given signed statement associated with an identifier was valid at the time of its issuance. Validation first requires that the statement is verifiable, that is, has a verifiable signature from the current controlling key-pair(s) at the time of its issuance. Therefore, a Validator must first act as a Verifier in order to establish the root authoritative set of keys. Once verified, the Validator may apply other criteria or constraints to the statement in order to determine its validity for a given use case. This use-case specific validation logic may be associated with interaction event statements.

Witness
A Witness is an entity or component designated (trusted) by the Controller of an identifier. The primary role of a Witness is to verify, sign, and keep events associated with an identifier. A witness is the Controller of its own self-referential identifier which may or may not be the same as the identifier to which it is a Witness. As a special case a Controller may serve as its own Witness. Witness designations are included in key (establishment) events. As a result, the role of a Witness may be verified using the identifier's rotation history. When designated, a Witness becomes part of the supporting infrastructure establishing and maintaining control authority over an identifier. An identifier Witness therefore is part of its trust basis and may be controlled (but not necessarily so) by its Controller.

The purpose of a pool of Witnesses is to protect the Controller from external exploit of its identifier. A Witness may use the controlling key-pairs of its own self-referential identifier to create digital signatures on event messages it has received but are associated with identifiers not necessarily under its control. To clarify, a Witness controls its own self-referential identifier and acts as a witness of event messages for some identifier not necessarily under its control. A Witness may receive, verify, and store an event on an identifier. Verify means verify the signature attached to the event using the current controlling key-pairs for the event at the time of event issuance.

Thus, a Witness first acts as an event Verifier. It determines current control authority of the event's identifier with respect to the sequence of key (establishment) events it has so far received for that identifier. The Witness follows a policy explained in more detail later for how it treats different versions of an event it may receive. Simply, it always gives priority to the first version of an event it receives (first seen). The Witness signifies this by only signing and keeping the first successfully verified version of an event it receives. To restate, a Witness will never sign any other conflicting version of the same event in an event sequence. The event sequence kept by a Witness for an identifier must therefore be internally consistent.

Watcher
A Watcher is an entity or component that keeps a copy of a Key Event Receipt Log (KERL) for an identifier but is not designated by the Controller thereof as one of its Witnesses. To clarify, a Watcher is not designated in the associated identifier's key events. A Watcher is the Controller of its own self-referential identifier which may not be the same as the identifier to which it is a Watcher. An identifier Watcher may be part of the trust basis of a Validator and may also be controlled (but not necessarily so) by the Validator's controlling entity. A Watcher may sign copies of its KERL or parts of its KERL but because a Watcher is not a designated Witness these are not witnessed receipts. They may be considered Watcher receipts or ersatz receipts.

Resolver
A Resolver is an entity or component that provides discovery for identifiers. A Resolver is the Controller of its own self-referential identifier which may not be the same as the identifier to which it is a Resolver. A Resolver primarily maps identifiers to the Uniform Resource Locators (URLs) or Internet Protocol (IP) addresses of components of the trust bases for identifiers. These components include Controllers, Witnesses and Watchers. Given the URL or IP address of a component, a user may there from obtain or be directed to the associated event histories (Key Event Logs (KELs), Key

Event Receipt Logs (KERLs), and Duplicitous Event Logs (DELs)) in order that the user may establish current (root) control authority for the identifier. The Resolver may cache these event histories or key event subsequences as end verifiable proofs of root control authority. A Watcher may act as a Resolver.

A Resolver provides a bootstrap discovery mechanism for KERI identifiers (AIDs). With respect to KERI, the target data for discovery is different for the two operative classes of identifier in KERI, that are, transferable and non-transferable identifier prefixes. In the case of a non-transferable identifier prefix, such as that of a Witness or Watcher, the target data may include a mapping from the non-transferable identifier prefix to a service endpoint Uniform Resource Locator (URL) or directly to the Internet Protocol (IP) address of the Witness or Watcher KERL service. In this case, a Validator could query the resultant IP address for a copy of the full KERL for the transferable identifier prefix to which the Witness or Watcher is entrained.

In the case of transferable identifiers, discovery may provide a mapping of the identifier prefix to a cached copy of either its full KEL or a copy of its inception event plus the latest rotation event or equivalently the latest key event state. From this copy, one may extract the identifier prefixes of the current Witness set and then use discovery to access the KERLs for those Witnesses.

Because KELs and KERLs are end-verifiable, almost any method of internet discovery is viable for KERI because discovery is merely a bootstrap mechanism. The end-verification that happens post discovery ensures that the discovered material is securely attributed.

How does KERI Key Management work?

All ledger-based system security is based on signing keys not becoming compromised. The difference in KERI is that pre-rotated keys enable recovery of compromise of signing keys. And delegated identifier keys may be recovered by any delegating identifiers. Multiple levels of delegation provide enhanced recovery. KERI's delegation is unique and is described as cooperative delegation. See the glossary below for a detailed summary of cooperative delegation. But the most important feature of cooperative delegation is the nested levels of delegation provide protection via recovery rotations of key compromise both of signing keys and pre-rotated keys of the levels below.

Best practices for key management are assumed by any user of public-private key pairs. But like any key management system one must protect one's secrets. KERI's pre-rotation makes those best practices all the more secure and nested delegation even more so. KERI specifically addresses the weakest link in conventional key management systems, that is key rotation. The pre-rotated keys never need be exposed to side channel attacks against event signing infrastructure until they are actually used for a rotation. And pre-rotated are one-time-use rotation keys. This is a best practice and not subject to a host of side channel attacks from which non-one-time-use rotation keys must be protected. The pre-rotated keys are also protected with a post-quantum proof hash thereby future proofing the system employed by KERI. Other key rotation algorithms that do not employ pre-rotation use much more complicated mechanisms for post-quantum proofing or not at all. Moreover, the nested delegation of identifiers that KERI employs enables enhanced security as the delegated identifiers are protected by the keys of the delegating identifiers. This means that the root keys need only be used once to delegate and never again unless the delegated keys become compromised. With multiple levels of delegation this minimizes potential attacks on the root keys.

KERI key management provides multiple layers of threshold structures that serve to multiply the number of attack surfaces that must be simultaneously compromised for a successful exploit. These are pre-rotation, multi-sig, and nested delegation. Moreover, Witnesses and Watchers provide threshold structures for protecting event signing and event signing verification. Unlike many other schemes that merely bolt on multi-signature support, KERI's support for multi-sig is built in as a first-class citizen. With nested delegated design of the vLEI infrastructure, a successful attacker must simultaneously compromise multiple nested multi-signatures on multiple sets of pre-rotated keys. Indeed, one may say that the vLEI infrastructure KERI implementation by GLEIF employs not just best practices for key management but best in class key management.

But to summarize, compromised pre-rotated keys on any but the root level the layer above may merely perform a rotation to invalidate compromised pre-rotated keys. Given that the upper layers at least will be multi-sig, such a compromise would be extremely difficult and such compromises of multi-sig systems are extremely rare. For example, so far there is no published case of a successful exploit of the Gnosis multi-sig wallet. And this does not even employ pre-rotation or multiple levels of delegation like KERI. So, KERI would be orders of magnitude more difficult to exploit. Such an exploit would be equivalent in difficulty to the compromise of the root keys for the most secure critical infrastructure. This is not at all equivalent to the compromise of a Domain Name Servers

(DNS) certificate which is based on the well-known vulnerabilities in the outdated, flawed insecure DNS/CA (Certificate Authority) system.

In addition, as per the KERI architecture, any participant in the vLEI ecosystem will add reliability/availability by operating their own Watcher pools which will keep copies of the GLEIF Key Event Logs (KELs). The following slide presentation provides diagrams of the notional witness network:

https://github.com/SmithSamuelM/Papers/blob/master/presentations/GLEIF_with_KERI.web.pdf

An even more extreme case would be a total compromise of all keys including the root pre-rotated keys. In this case the root identifier must be abandoned, and a new root created. This is equivalent to the total compromise of a ledger.

More on the security implications of KERI

The following white paper discusses in more detail the security implications of threshold structures and key management and root-of-trust with respect to KERI in a GLEIF class application:

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/IdentifierTheory_web.pdf

It is important to make apples to apples comparisons between key management schemes especially with regards the root-of-trust. It is easy to confuse the security difference between a convenient scheme for the recovery of a weak root-of-trust and more inconvenient recovery scheme but for a much stronger root-of-trust. The latter may be magnitudes more secure than the former. The goal is to make compromise of the root-of-trust vanishingly remote as opposed to enabling the more convenient but also more common recovery of a weak root-of-trust. The cumulative harm to the system is much greater for the weak root of trust as the cumulative harm grows the more one must recover from exploit.

KERI enables a scalable distributed hierarchical dissemination of trust out to the leaves but with ultimate recovery potential back to the extremely well protected root. Multi-level threshold structures provide the strongest possible security mechanisms and may be designed to be arbitrarily secure merely by multiplying the number of attack surfaces that must be simultaneously breached for successful attack. KERI's hierarchical cooperative delegation enables convenient and scalable performance at the outer levels without sacrificing ultimate security provided by increasingly higher levels of security of the nested levels ending in the root level.

How was GLEIF established as the root-of-trust for the vLEI ecosystem?

In order to understand how a hypothetically one would handle the compromise of keys, it is helpful to understand the original process by which the root-of-trust root Autonomic Identifier (AID) GLEIF ID (GID) is established.

The original root identifier GID becomes qualified after its creation by a multi-factor association with GLEIF as the controller. Once this multi-factor association is sufficiently published and recognized that it becomes “common knowledge”. At which point the identifier (KERI AID = GID) may be used as the recognized root-of-trust. The first action of this root of trust will be to delegate other identifiers (GIDs = KERI AIDs). The root level is only used for delegation and only allows rotations. This means that each set of keys is a one-time use key set for a rotation that delegates. Each rotation commits to a new set of pre-rotated keys that have never been exposed. Because the first rotation makes a set of delegations this root set may never need to be used again until such time as recovery or rotation of delegated keys is needed. Thus, the usage is minimized and be performed in the most secure manner possible.

But suppose that somehow despite all these security measures, the root GID set of pre-rotated keys becomes compromised (note this would be a multi-sig compromise of unused keys stored in separate locations), then the original GID would have to be abandoned and a new GID would have to be established via the same multi-factor association process as the original GID.

Recall, that any PKI scheme ultimately must perform such an abandonment and re-association if all keys are compromised. What is unique about KERI is that by using threshold structures of multi-sig, nested delegation, and limiting exposure through one-time pre-rotated keys, the likelihood of such a complete compromise is as small as practically possible using current widely accepted digital signature Public Key Infrastructure (PKI) libraries.

How will the vLEI infrastructure handle revocation of credentials?

GLEIF believes that revocation enforcement is one of the key differentiators to X.509 certificates. GLEIF’s ToIP Ecosystem Governance Framework establishes a decentralized, hierarchical Public Key Infrastructure (PKI) with GLEIF being the root of trust for the associated vLEI ecosystem. Revocation is provided at each level of chained issued vLEI credentials. LEIs will be wrapped as Verifiable Credentials according to the ToIP Authentic Chained Data Container (ACDC). As an ACDC VC, the LEI will be cryptographically bound to the owner of the designated private/public key pair. Each vLEI will get a Decentralized Identifier (DID) assigned Data will be discoverable via respective interfaces. vLEIs will support personal and organizational wallets and the respective protocols. Having this said, governance of revocation does not only have technical but also business implications. vLEIs are subject to revocation if the underlying LEI lapses. Qualified vLEI Issuers are required on a regular basis to check the Entity and Registration status of LEIs for which Legal Entity vLEI Credentials have been issued and QVIs are required to revoke the Legal Entity’s Legal Entity vLEI Credential if these status requirements are not met. This would also help the LEI system and the reinforcement of renewals in order to achieve higher data quality levels.

What is the nested delegation that KERI uses?

There are two types of delegation:

- Delegated AIDs (Autonomic Identifiers (AIDs) which have associated Decentralized Identifier (DIDs). These are primary identifiers. Unless otherwise indicated, whenever the term identifier is used with reference to KERI, the references are to primary identifiers.
- Delegated Authentic Chained Data Container (ACDC) credentials, the credential type used for vLEIs, in which an LEI code is an attribute of a vLEI. A LEI code is a secondary identifier.

Delegated AIDs use cooperative delegation. Delegated ACDCs (vLEIs) do not use cooperative delegation.

An ACDC (vLEI) is issued by a DID (derived from an AID) and issued to a DID (derived from an AID). Thus a delegated vLEI issuance has an issuer and a holder. However, the DIDs for the issuer and holder of this delegated vLEI may or may not be related via a cooperative AID delegation.

Usually cooperative AIDs are based on some formal relationship such as employer-employee or other affiliation like GLEIF and its vLEI Issuers. The cooperation relationship imposes friction but with the reward of enhanced security.

Delegation of ACDCs (vLEIs) are non-cooperative delegations. Thus, they could always be used to provenance any vLEI and any data within the vLEI without imposing a cooperation relationship on the delegation parties.

Nested delegation can store keys in a less secure manner but be protected by the delegator's key management infrastructure. This allows higher performance by delegated infrastructure without sacrificing ultimate security. See the following for a more detailed discussion of the advantages of nested delegation:

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/IdentifierTheory_web.pdf

Glossary of Additional KERI Terms

A subset of KERI terms is provided below for convenience. A full description of all KERI terminology is beyond the scope of this document but may be found in the KERI White Paper here:

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP.web.pdf

Duplicitous Event Log
A Duplicitous Event Log (DEL) is record of inconsistent event messages produced by a given Controller or Witness with respect to a given KERL. The duplicitous events are indexed to the corresponding event in a KERL. A duplicitous event is represented by a set of two or more provably mutually inconsistent event messages with respect to a KERL. Each Watcher keeps a DEL for each Controller and all designated Witnesses with respect to a KERL. Any Validator may confirm duplicity by examining a DEL.

Transferable (Non-transferable) Identifier
A Transferable (Non-transferable) Identifier allows (dis-allows) transfer of its control authority from the current set of controlling keys to a new (next or ensuing) set via a rotation event (see below). An identifier may be declared non-transferable at inception in its derivation code and/or in its inception event (see below). Derivation code declaration is defined only for basic self-certifying identifiers. A rotation event (operation) on a transferable identifier may rotate to a null key thereby irreversibly converting it into a Non-transferable Identifier. Once an identifier becomes non-transferable, no more events are allowed for that identifier. The identifier is effectively abandoned from the standpoint of KERI. By convention, when non-transferability of an identifier is declared in its derivation code then its authoritative (signing) key-pair may be converted to an encryption key-pair to enable a self-contained bootstrap to a secure communications channel using only the exchange of the non-transferable identifier. Although a non-transferable identifier is abandoned from the standpoint of KERI, it does not preclude a given application from employing the identifier. It is just that no more events within KERI are allowed on the identifier (see event definition below). An identifier declared at inception as non-transferable may have one and only one event, that is, the inception event. In this sense, a Non-transferable Identifier at inception is pre-abandoned. These identifiers are typically meant to be used as ephemeral identifiers or identifiers where replacement of the identifier instead of key rotation is the preferred approach when the identifier becomes compromised.

Cooperative Delegation
A delegation or identifier delegation operation is provided by a pair of events. One event is the delegating event and the other event is the delegated event. This pairing of events is a somewhat novel approach to delegation in that the resultant delegation requires cooperation between the delegator and delegate. This is called Cooperative Delegation. In a Cooperative Delegation, a delegating identifier performs an establishment operation (inception or rotation) on a delegated identifier. A delegating event is a type of event that includes in its data payload an event seal of the delegated event that is the target the delegation operation. This delegated event seal includes a digest of the delegated event.

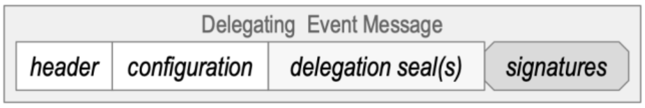

Figure: Delegating Event

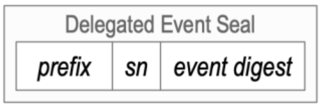

Figure: Delegated Event Seal

Likewise, the targeted delegated event has a delegating event location seal that includes the unique location of the delegating event.

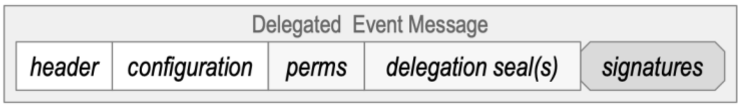

Figure: Delegated Event

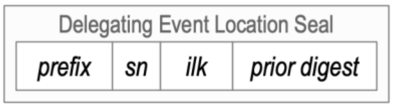

Figure: Delegating Event Location Seal

The pair of seals cross-reference the two events participating in the cooperative delegation operation. In general, we may refer to both delegating and delegated event seals as delegation event seals or delegation seals. A delegation seal is either an event seal or an event location seal. The delegating event seal is an event location seal and a delegated event seal is an event seal. The delegated event seal in the delegating event provides an anchor to the delegated event. Likewise, the delegating event location seal in the delegated event provides an anchor back to the delegating event.

Because the delegating event payload is a list, a single delegating event may perform multiple delegation operations, one per set of delegation seals.

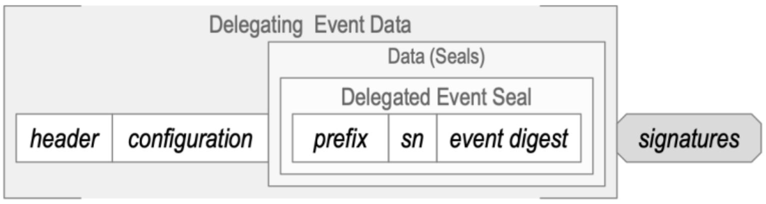

Figure: Delegating Event Data

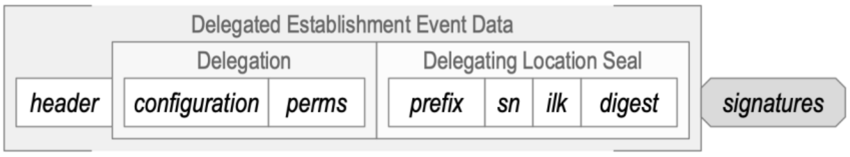

Figure: Delegated Event Data

A delegation operation directly delegates an establishment event, either an inception or rotation. Thus, a delegation operation may either delegate an inception or delegate a rotation that respectively may create and rotate the authoritative keys for delegated self-certifying identifier prefix.

The delegated identifier prefix is a type of self-addressing self-certifying prefix. This binds the delegated identifier to its delegating identifier. The delegating identifier controller retains establishment control authority over the delegated identifier in that the new delegated identifier may only authorize non-establishment events with respect to itself. Delegation therefore authorizes signing authority that can be revoked to some other self-certifying identifier. The delegated identifier may have its own delegated key event sequence where the inception event is a delegated inception and any rotation events are delegated rotation events. Control authority for the delegated identifier therefore requires verification of a given delegated establishment event which in turn requires verification of the delegating identifier's establishment event subsequence.

To reiterate, because the delegation seal in the data payload of the delegating event includes a digest of the full delegated event, it thereby provides a forward cryptographic commitment to the delegated identifier as well as any permissions and other configuration data in its associated event. The delegation seal included in the delegated event provides a backward reference to the delegating event's unique location. This uniquely establishes which event in the delegating event log holds the corresponding seal. This provides a type of cross reference that enables a verifier to look up the delegating event and verify the existence of the delegation seal in the list of seals in that delegating event and then verify that the event seal digest is a digest of the delegated event.

A common use case of delegation would be to delegate signing authority to a new identifier prefix. The signing authority may be exercised by a sequence of signing keys that are able to be revoked distinct from the keys used for the root identifier. This enables horizontal scalability of signing operations. The other major benefit of a cooperative delegation is that any exploiter that compromises only the delegate's authoritative keys may not thereby capture control authority of the delegate. A successful exploiter must also compromise the delegator's authoritative keys. Any exploit of the delegate is recoverable by the delegator. Conversely, merely compromising the delegator's signing keys may not enable a delegated rotation without also compromising the delegates pre-rotated keys. Both sets of keys must be compromised simultaneously. This joint compromise requirement is a distinctive security feature of cooperative delegation. Likewise as explained later, this cooperative feature also enables recovery of a joint compromise of a delegation at any set of delegation levels by a recovery at the next higher delegation level.

One reason to use rotations for delegation is for enhanced security. A rotation event is a first time use of the pre-rotated keys to sign an event. The distinction between an interaction event rotating a delegated identifier's keys and a rotation event rotating a delegated identifier's keys is that the latter enables nested recovery of a compromise of the delegate's keys, even its pre-rotated keys. A rotation event may be used to supersede an interaction event. When this happens the key event log forks at the superseding rotation event. With delegated events this means recovery is enabled even in the event of the joint compromise of a delegating identifier's signing keys and the delegated identifiers pre-rotated keys. The delegating identifier merely needs to perform a rotation event that provides a superseding rotation of the interaction event used to delegate a rotation of the delegate. This superseding rotation also performs a superseding rotation of the delegates rotation. This nested recovery may be applied to multiple levels of delegation. A rotation at the next higher level of delegation may be used to recover from key compromise across any set of lower levels of delegation.

Seal
A seal is a cryptographic commitment in the form of a cryptographic digest or hash tree root (Merkle root) that anchors arbitrary data or a tree of hashes of arbitrary data to a particular event in the key event sequence. According to the dictionary, a seal provides evidence of authenticity. A key event sequence provides a verifiable proof of current control authority at the location of each event in the key event sequence. In this sense therefore, a seal included in an event provides proof of current control authority, i.e., authenticity of the data anchored at the location of the seal in the event sequence. A seal is an ordered self-describing data structure. Abstractly, this means each element of the seal has a tag or label that describes the associated element's value. So far there are four normative types of seals - these are digest, root, event, and location seals.

A digest seal includes a digest of external data. This minimal seal has an element whose label indicates that the value is a digest. The value is fully qualified Base64 with a prepended derivation code that indicates the type of hash algorithm used to create the digest.

Digest Seal
digest
Figure: KERI Digest Seal

A root seal provides the hash tree root of external data. This minimal seal has an element whose label indicates that the value is the root of a hash tree. The value is fully qualified Base64 with a prepended derivation code that indicates the type of hash algorithm used to create the hash root. In order to preclude second pre-image attacks, hash trees used for hash trees roots in KERI seals must be sparse and of known depth similar to certificate transparency. One simple way to indicate depth is that internal nodes in a sparse tree include a depth prefix that decrements with each level and must remain non-negative at a leaf.

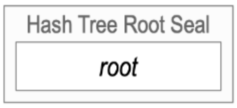

Figure: KERI Root Seal

An event seal includes the identifier prefix, sequence number, and digest of an event in a key event log. The prefix, sequence number, and digest allow locating the event in an event log database. The digest also allows confirmation of the anchored event contents. An event seal anchors one event to another event. The two events may be either in the same key event sequence in two different key event sequences with different identifier prefixes. Thus, a seal may provide a cryptographic commitment to some key event from some other key event.

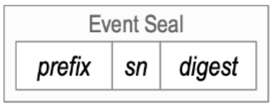

Figure: KERI Event Seal

An event location seal is similar to an event seal. A location seal includes the prefix, sequence number, ilk and prior digest from an event. These four values together uniquely identify the location of an event in a Key Event Log. A location event is useful when two seals in two different events are cross-anchoring each other. This provides a cross reference of one event to another where the other event's digest must include the seal in the event contents so it cannot contain the first event's digest but the digest of the preceding event.

To clarify, digest creation means that only one of the cross anchors can include a complete digest of the other event. The other cross anchor must use a unique subset of data such as the unique location of the event. The *ilk* is required in the location because of the special case of recovery where a rotation event supersedes an interaction event. This is described in detail in the white paper under recovery. Location seals are also useful in external data that is anchored to an event log. The location seal allows the external data to include a reference to the event that is anchoring the external data's contents. Because the anchoring event includes a seal with the digest of the external data, it is another form of cross anchor.

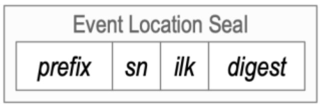

Figure: KERI Event Location Seal

The data structure that provides the elements of a seal must have a canonical order so that it may be reproduced in a digest of elements of an event. Different types of serialization encodings may provide different types of ordered mapping data structures. One universal canonical ordering data structure is a list of lists (array or arrays) of (label, value) pairs. The order of appearance in each list of each (label, value) pair is standardized and may be used to produce a serialization of the associated values.

The interpretation of the data associated with the digest or hash tree root in the seal is independent of KERI. This allows KERI to be agnostic about anchored data semantics. Another way of saying this is that seals are data agnostic; they do not care about the semantics of the associated data. This better preserve privacy because the seal itself does not leak any information about the purpose or specific content of the associated data. Furthermore, because digests are a type of content address, they are self-discoverable. This means there is no need to provide any sort of context or content specific tag or label for the digests. Applications that use KERI may provide discovery of a digest via a hash table (mapping) whose indexes (hash keys) are the digests and the values in the table are the location of the digest in a specific event. To restate, the semantics of the digested data are not needed for discovery of the digest within a key event sequence.

To elaborate, the provider of the data understands the purpose and semantics and may disclose those as necessary, but the act of verifying authoritative control does not depend on the data semantics, merely the inclusion of the seal in an event. It is up to the provider of the data to declare or disclose the semantics when used in an application. This may happen independently of verifying the authenticity of the data via the seal. This declaration may be provided by some external Application Program Interface (API) that uses KERI. In this way, KERI provides support to applications that satisfies the spanning layer maxim of minimally sufficient means. Seals merely provide evidence of authenticity of the associated (anchored) data whatever that may be.

This approach follows the design principle of context independent extensibility. Because the seals are context agnostic, the context is external to KERI. Therefore, the context extensibility is external to, and hence, independent of KERI. This is in contrast to context dependent extensibility or even independently extensible contexts that use extensible context mechanisms such as linked data or tag registries. Context independent extensibility means that KERI itself is not a locus of coordination between contexts for anchored data. This maximizes decentralization and portability.

Extensibility is provided instead at the application layer above KERI though context specific external APIs that reference KERI seals in order to establish control authority, and hence, authenticity of the anchored (digested) data. Each API provides the context, not KERI. This means that interoperability within KERI is focused solely on interoperability of control establishment. But that interoperability is total and complete and is not dependent on anchored data context.