

Key Event Receipt Infrastructure (KERI)

Model for a Universal DKMI

Samuel M. Smith Ph.D.
<https://github.com/SmithSamuelM/Papers>
sam@samuelsmith.org

Background Reading

Girault, M., “Self-certified public keys,” EUROCRYPT 1991: Advances in Cryptology, pp. 490-497, 1991

https://link.springer.com/content/pdf/10.1007%2F3-540-46416-6_42.pdf

Kaminsky, M. and Banks, E., “SFS-HTTP: Securing the Web with Self-Certifying URLs,” MIT, 1999

<https://pdos.csail.mit.edu/~kaminsky/sfs-http.ps>

Mazieres, D. and Kaashoek, M. F., “Escaping the Evils of Centralized Control with self-certifying pathnames,” MIT Laboratory for Computer Science, 2000

<http://www.sigops.org/ew-history/1998/papers/mazieres.ps>

Mazieres, D., “Self-certifying File System,” MIT Ph.D. Dissertation, 2000/06/01

<https://pdos.csail.mit.edu/~ericp/doc/sfs-thesis.ps>

Smith, S. M., “Open Reputation Framework,” vol. Version 1.2, 2015/05/13

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/open-reputation-low-level-whitepaper.pdf>

Smith, S. M. and Khovratovich, D., “Identity System Essentials,” 2016/03/29

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/Identity-System-Essentials.pdf>

Smith, S. M., “Decentralized Autonomic Data (DAD) and the three R’s of Key Management,” Rebooting the Web of Trust RWOT 6, Spring 2018

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/DecentralizedAutonomicData.pdf>

Conway, S., Hughes, A., Ma, M. et al., “A DID for Everything,” Rebooting the Web of Trust RWOT 7, 2018/09/26

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

Smith, S. M., “**Key Event Receipt Infrastructure (KERI) Design and Build**”, arXiv, 2019/07/03

<https://arxiv.org/abs/1907.02143>

Human Basis of Trust

I know you

or

I know of you

therefore

I trust you

On the internet I can't really know who you are.

therefore

I can't trust you

Decentralized Control Authority

Asymmetric PKI: **Public-Private Key Pairs for Digital Signatures**

CSPRNG Cryptographic Strength Pseudo Random Number Generator

Collision Resistant Random Seed (Entropy) Available to Anyone

Seed -> Private Key -> One Way Function -> Public Key

Authority comes from collision resistance

Inherently decentralized via sole-sovereignty over random seed

Controller of private key is the only one who can make verifiable signed attestations associated with the Public Key (Identifier)

Derived Self-Certifying Content Addressable Identifiers

Content Hash:

Seed -> Private Key -> One-way Function (sign scheme) -> Public Key -> +Content -> One-way Function (hash scheme) -> Derived Identifier

Seed -> Private Key -> One-way Function (sign scheme) -> Public Key -> [+Content -> One-way Function (hash scheme) -> Derived Identifier] ...

Cryptographic Root-of-Trust

Trust *who said it* not *what was said*

Consistent attribution is the root-of-trust (integral non-repudiable statements)

Duplicity detection

I trust that controller of private key made a set of statements

Build trust in *what was said* via consistently attributable (*who*) histories of consistent statements (*what*). (i.e. reputation)

Self-Certifying Identifier/Namespace

Decentralized Cryptographic Root-of-Trust

Use public key in identifier or as prefix in namespace

Signed statements that include self-certifying identifier

Self-Certifying namespace

End-wise Verifiable to Primary Root-of-Trust

Other roots of trust may add to but not replace self-certification

All decentralized infrastructure has self-certifying identifiers as primary root-of-trust

Decentralized key management is therefore essential to protecting infrastructure

Enables provenanced chain of transformations with verifiable control over transformations.

Authoritative control via signing not encryption keys.

AUTONOMIC IDENTIFIER (AID)

Self-Governing

Self-Certifying

Self-Sovereign

Self-Managing

Self-Administering

AUTONOMOMIC NAMESPACE (AN)

Prefixed with AID

Extensible

Cryptographically Verifiable Identifier Derivation

KEY MANAGEMENT

Rotation = Revoke and Replace

Reproduction = Creation, Derivation

Recovery

BEST PRACTICE

One-use = One-time, One-place, One-way

Key Rotation

Key rotation is useful when the controller of a self-certifying identifier needs to maintain persistent control over that identifier indefinitely despite exploits of the private key(s).

Otherwise in the event of exploit, the controller could just abandon the exploited identifier and create a new identifier with a new (public, private) key pair.

Rotatable Authoritative Signing Key(s)

vs.

Non-rotatable Authoritative Signing Key(s)

Key Rotation More

After rotation the public key associated with the identifier is not changed, but merely the private key(s) that is authoritative is changed. Otherwise the identifier loses its value as an identifier.

After rotation the original private key has been revoked and replaced with the new private key(s) specified in the rotation operation.

The initial rotation operation must be signed at the very least with the original private key.

The new public key is then included in the identifier's key rotation history.

The latest rotation entry provides the current authoritative key pair.

In order to verify an attestation belonging to a self-certifying identifier the verifier must lookup and validate the key rotation history for that identifier.

Pre-rotation

Inception -> Rotation -> Rotation -> ...

Inception event:

- 1) Establish original self-certifying key
- 2) Commit to next set of key(s)

Each rotation event:

- 1) Execute rotation operation to new key(s)
- 2) Commit to next set of key(s)

Rotation signed by both previous and new set of key(s)

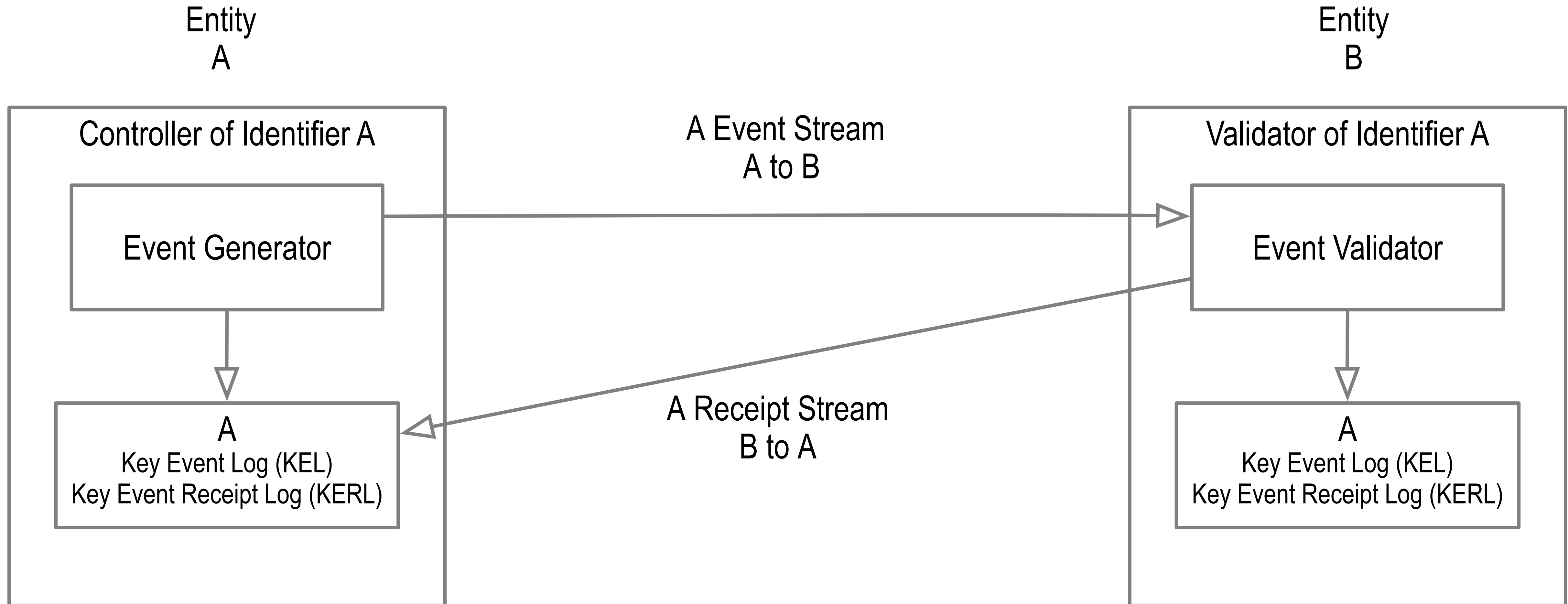
Rotation exploit requires exploit of two sets key(s) one of which has not been exposed.

Two Use Cases

Online haltable establishment (one-to-one, pair-wise)

Offline non-haltable establishment (one-to-many, any-wise, N-wise)

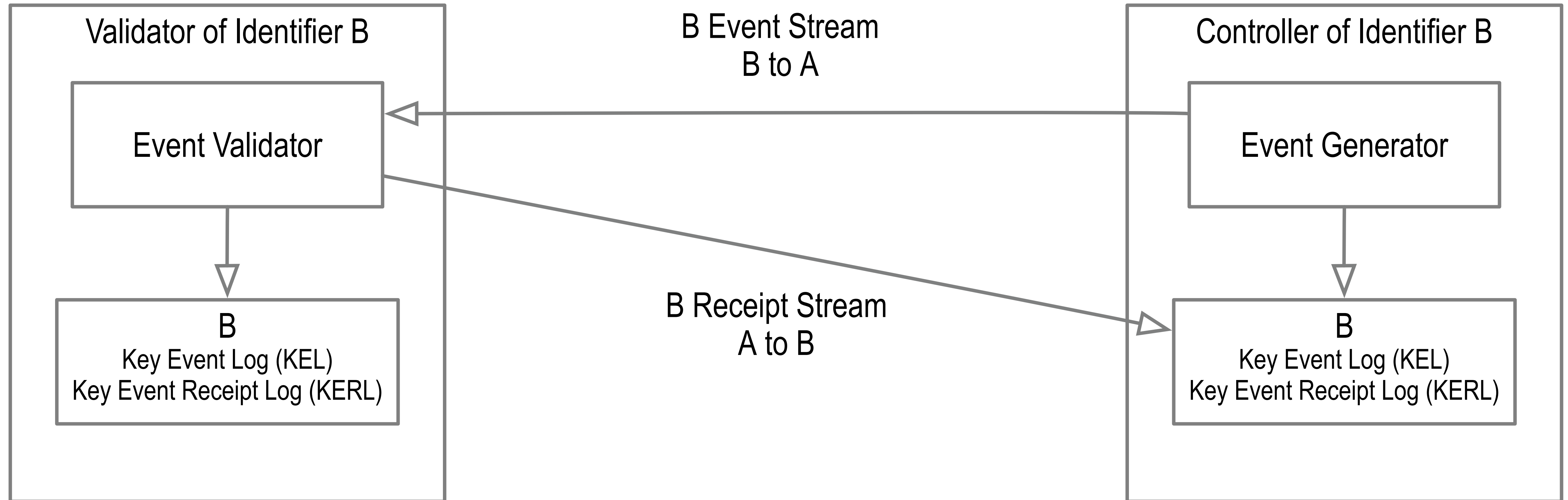
Online or One-to-one or Pair-wise: A to B



Online or One-to-one or Pair-wise: B to A

Entity
A

Entity
B



Offline Trustworthy Service

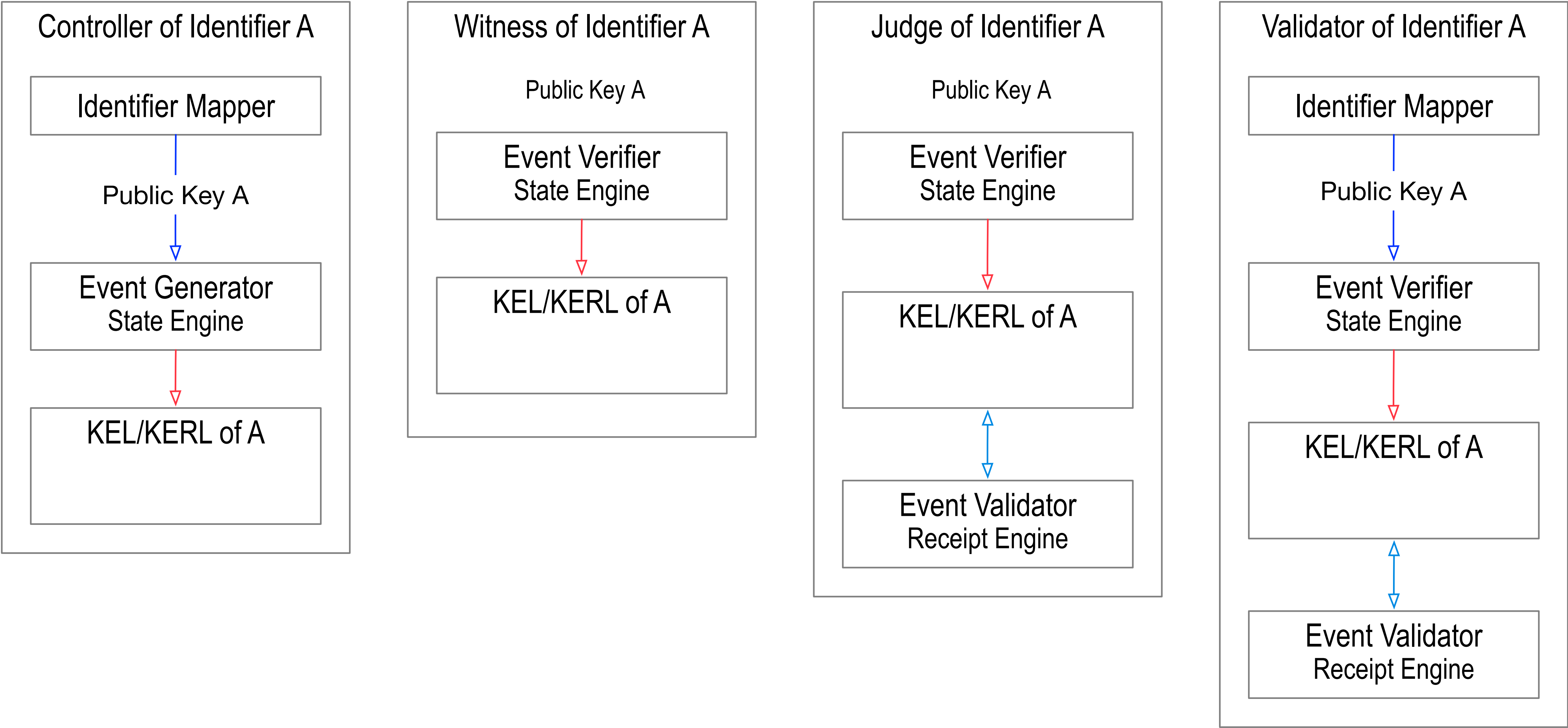
Service provided by N designated *witnesses* and at least one *judge*.

Service provides a *correct* or at least a *complete* KERL to any validator that requests it.

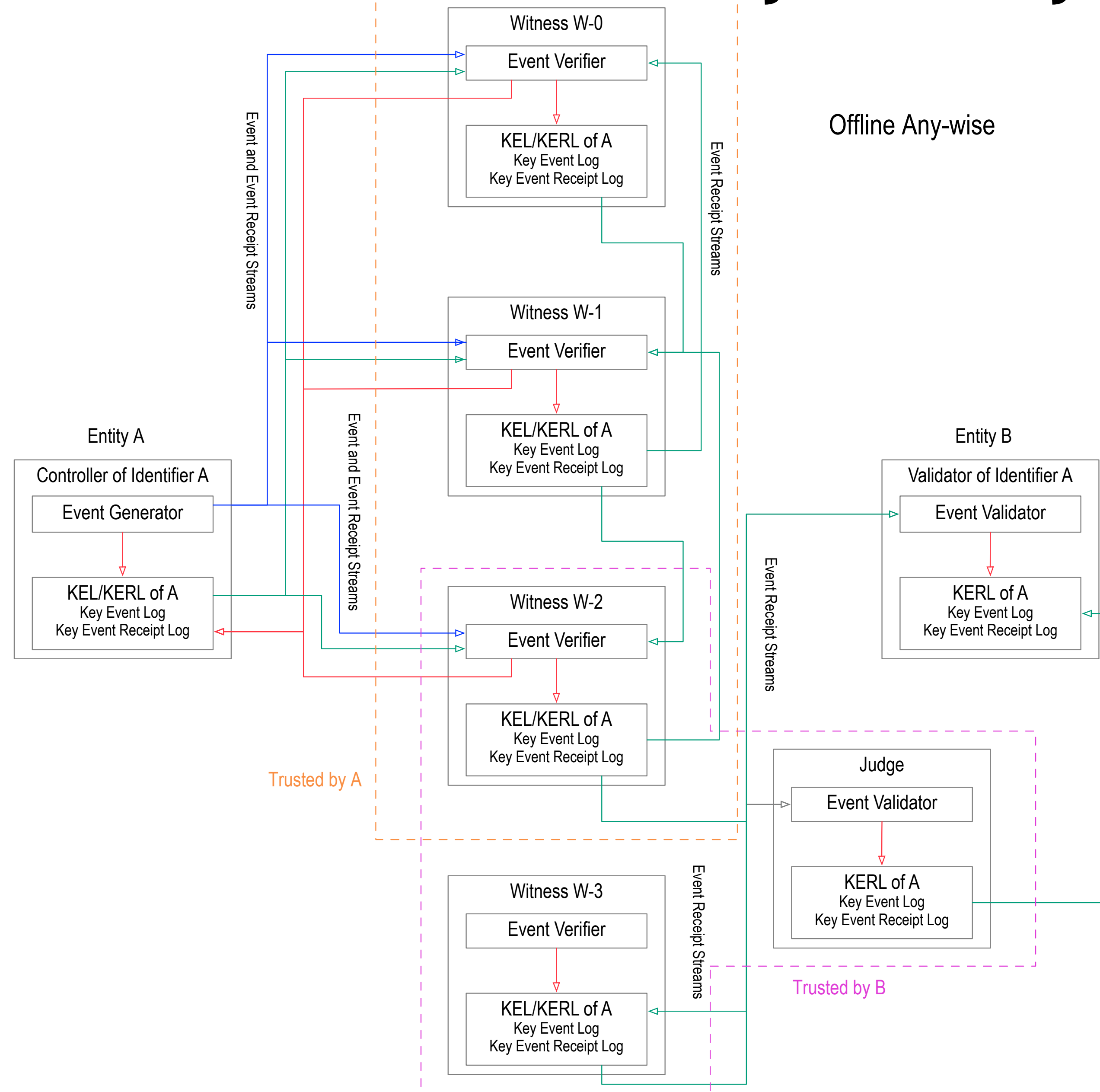
Service availability means at least one *correct* or *complete* KERL is extant

key state validation mechanism does not use total ordered distributed consensus

Offline Components



Offline or One-to-many or Any-wise



Complete or Correct

Witnesses only signs and stores first received and verified version of an event at a given location in full event sequence.

Witnesses provide an immutable KERL of all receipts from any witnesses for the version of each event in its log.

M of N Designated Witnesses

Complete if KERL has $M+1$ receipts for event

Correct if M is a majority of N

Exploit Recovery

Pre-rotation protects against live exploit

Recover by rotation to new authoritative key(s)

Witnessed event log protects against dead exploit

Single honest witness enables duplicity detection

Historical receipts prove accepted history

Variants

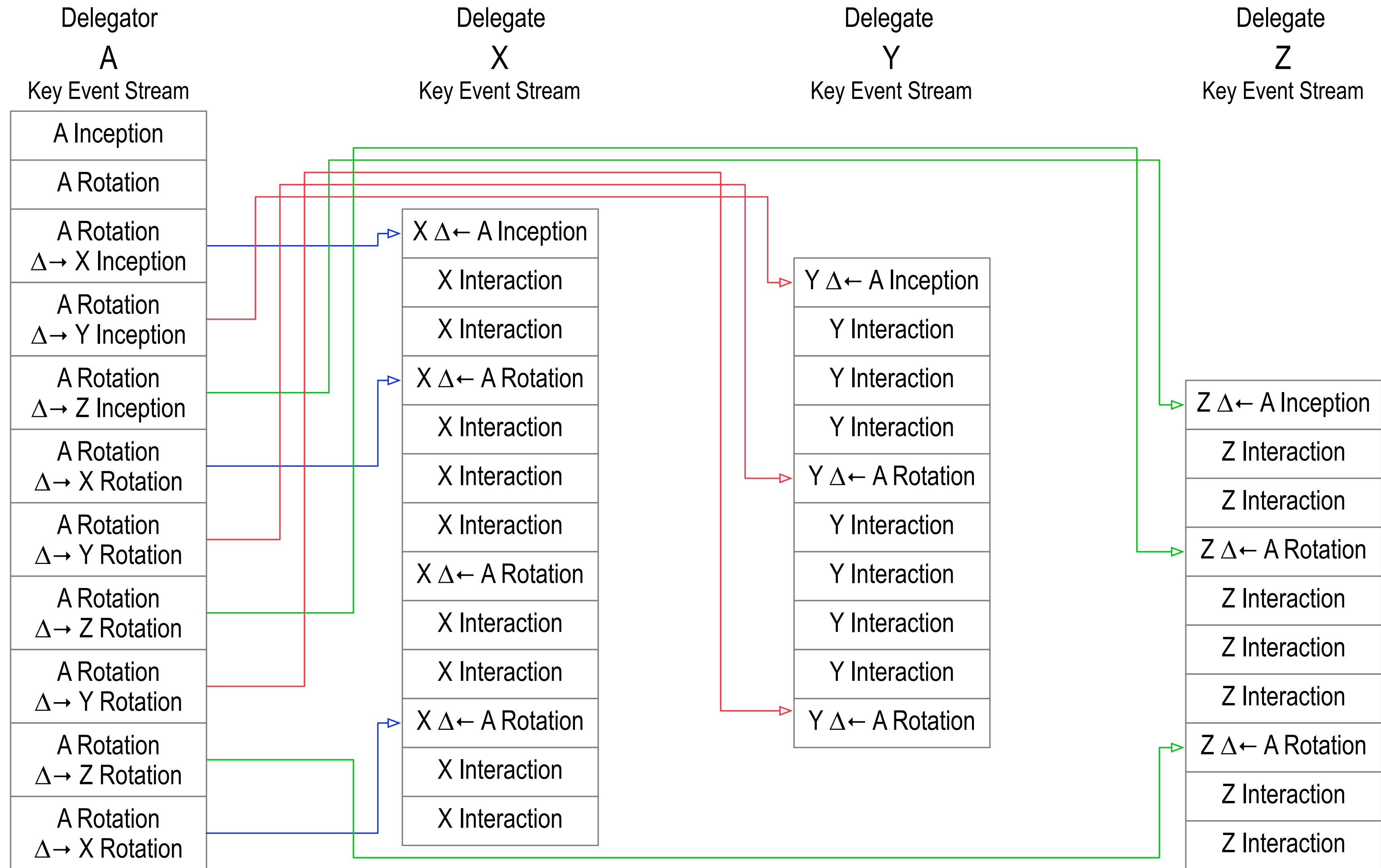
Single Key Sequence for Rotation and Signing

Single Interleaved (Doubled) Key Sequence for Rotation and Signing

Single Key Sequence for Rotation with multiple Delegated Key
Sequence(s) for Signing

Dual Key Sequence for Separate Rotation and Signing

Delegation



$\Delta \rightarrow$ X : Delegation to X

$\Delta \leftarrow$ A : Delegation from A

KEY Event Based Provenance of Identifiers

KERI events enable cryptographic Proof-of-Authority or Proof-of-Provenance for every identifier.

Proof is copy of key event receipt log (KERL)

KEY (Identifier) Role Support

Controlling Key (Identifier) Roles via Authoritative Delegation:

- Signing Keys

- Rotatable (optional)

- Fully verifiable to root-of-trust using KERI

Non-Controlling Key (Identifier) Roles via Authoritative Attestation:

- Encryption, Sign-cryption, etc

- Always Non-Rotatable

- Fully verifiable to authoritative controlling key(s) at point of attestation

- Granular role establishment via events

Generalized Discovery Support

Non-Controlling Identifier Discovery Roles via Authoritative
Attestation:

Service Endpoints, Documentation, Etc.

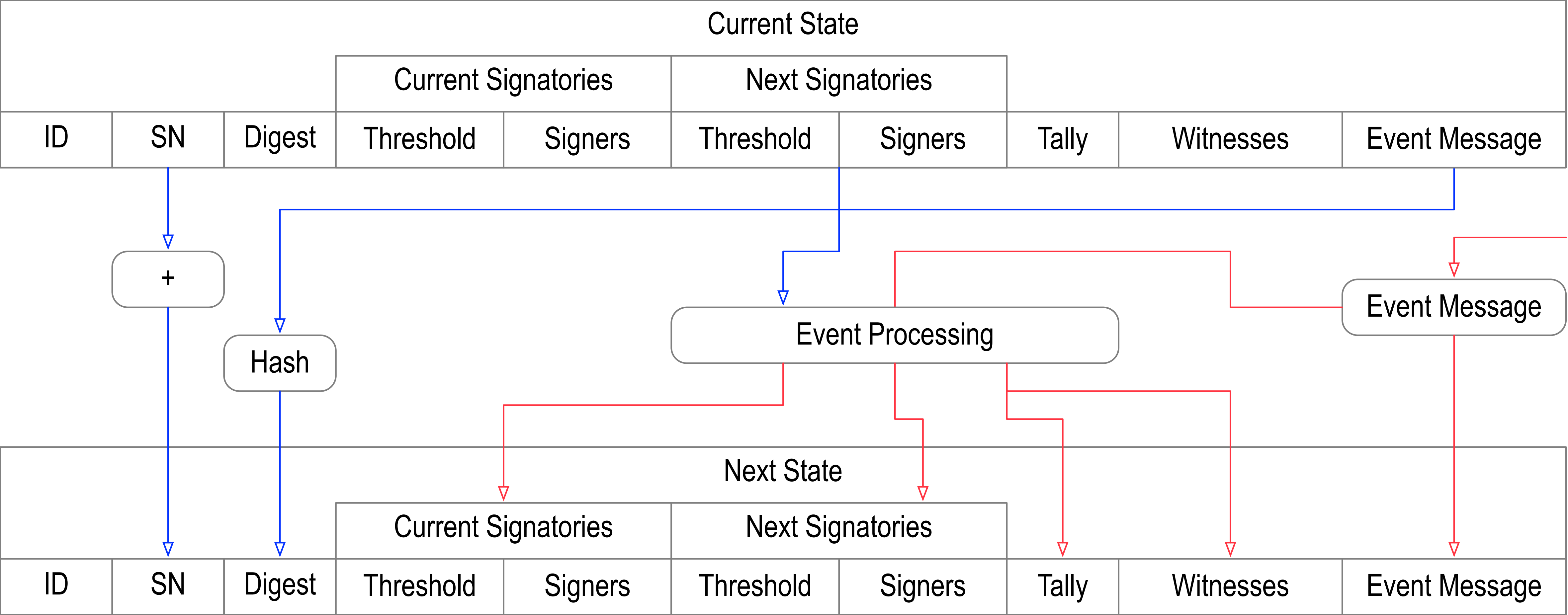
Always Non-Rotatable

Fully verifiable to authoritative controlling key(s) at point of
attestation

Granular role establishment via events

State Verifier Engine

KERI Core — State Verifier Engine



Generic Inception

$$\varepsilon_0 = \left\langle C, t_0, \texttt{icp}, C^0, K_1, \hat{C}_1, M_0, \hat{W}_0 \right\rangle \sigma_{C^0}$$

$$\hat{C}_1 = \left[C^1, \dots, C^{L_1} \right]_1$$

$$\hat{W}_0 = \left[W_0, \dots, W_{N-1} \right]_0$$

Generic Rotation

$$\varepsilon_k = \left\langle C, t_k, \eta(\varepsilon_{k-1}), \text{rot}, K_{l+1}, \hat{C}_{l+1}, M_l, \hat{X}_l, \hat{Y}_l, \{data\}, \hat{e}_l, \hat{s}_l \right\rangle \hat{\sigma}_l$$

$$\hat{C}_{l+1} = \left[C^{r_{l+1}}, \dots, C^{r_{l+1}+L_{l+1}-1} \right]_{l+1}$$

$$\hat{X}_l = \left[X_0, \dots, X_{O_l-1} \right]_l$$

$$\hat{Y}_l = \left[Y_0, \dots, Y_{P_l-1} \right]_l$$

$$\hat{e}_l = \left[e_0, \dots, e_{E_l-1} \right]_l$$

$$\hat{s}_l = \left[s_0, \dots, s_{S_l-1} \right]_l$$

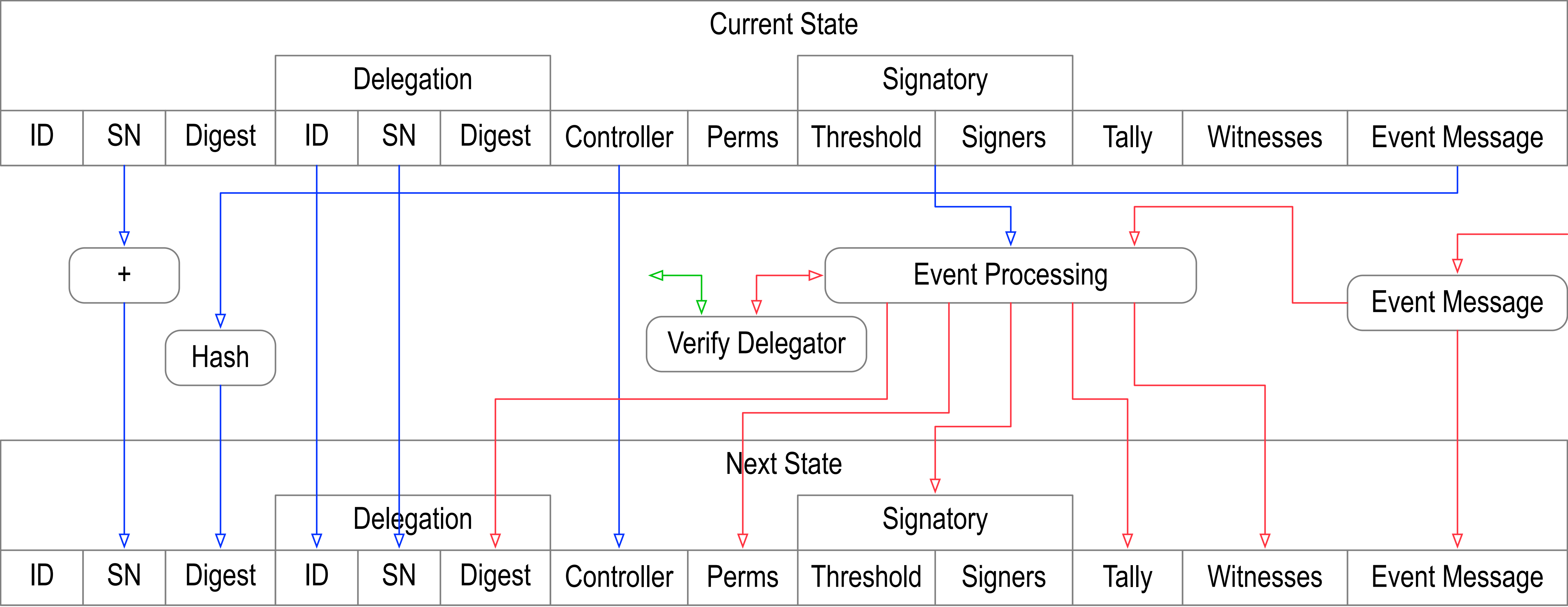
$$\hat{\sigma}_l = \sigma_{C^{e_0+\eta-1}} \dots \sigma_{C^{e_{E_l-1}+\eta-1}} \sigma_{C^{s_0+\eta}} \dots \sigma_{C^{s_{S_l-1}+\eta}}$$

Interaction Event

$$\boldsymbol{\varepsilon}_k = C_{\boldsymbol{\varepsilon}_k} = \left\langle C, t_k, \eta(\boldsymbol{\varepsilon}_{k-1}), \texttt{itc}, \{data\}, \hat{s}_k \right\rangle \hat{\sigma}_k$$

Delegated State Verifier Engine

KERI Delegated Core — State Verifier Engine



Delegating Inception

$$\varepsilon_k = \left\langle C, t_k, \eta(\varepsilon_{k-1}), \text{rot}, K_{l+1}, \hat{C}_{l+1}, M_l, \hat{X}_l, \hat{Y}_l, \hat{\Delta}_0^D, \hat{e}_l, \hat{s}_l \right\rangle \hat{\sigma}_l$$

$$\hat{\Delta}_0^D = \{D, \text{dip}, D^0, perms, K_0^D, \hat{D}_0^D, M_0^D, \hat{W}_0^D\}$$

$$\varepsilon_0 = D_{\varepsilon_0} = \left\langle D, t_0, \text{dip}, \hat{\Delta}_k^C, D^0, perms, K_0^D, \hat{D}_0^D, M_0^D, \hat{W}_0^D \right\rangle \sigma_{D^0}$$

$$\hat{\Delta}_k^C = \{C, t_k^C, \eta(\varepsilon_k^C)\}$$

Delegating Rotation

$$\varepsilon_k = \left\langle C, t_k, \eta(\varepsilon_{k-1}), \mathbf{rot}, K_{l+1}, \hat{C}_{l+1}, M_l, \hat{X}_l, \hat{Y}_l, \hat{\Delta}_e^D, \hat{e}_l, \hat{s}_l \right\rangle \hat{\sigma}_l$$

$$\hat{\Delta}_e^D = \left\{ D, \mathbf{drt}, perms, K_e^D, \hat{D}_e^D, M_e^D, \hat{X}_e^D, \hat{Y}_e^D \right\}$$

$$\varepsilon_d = D_{\varepsilon_d} = \left\langle D, t_d, \eta(\varepsilon_{d-1}), \mathbf{drt}, \hat{\Delta}_k^C, perms, K_e^D, \hat{D}_e^D, M_e^D, \hat{X}_e^D, \hat{Y}_e^D \right\rangle \sigma_{D^0}$$

$$\hat{\Delta}_k^C = \left\{ C, t_k^C, \eta(\varepsilon_k^C) \right\}$$

Delegated Interaction

$$\boldsymbol{\varepsilon}_d = D_{\boldsymbol{\varepsilon}_d} = \left\langle D, t_d, \eta(\boldsymbol{\varepsilon}_{d-1}), \texttt{itc}, data, \widehat{s}_d^D \right\rangle \widehat{\boldsymbol{\sigma}}_d$$

KERI Nomenclature

self-certifying identifier: includes public key

digital signature: unique non-repudiable (cypher suite known)

digest: collision resistant hash of content

signed digest: commitment to content

controller: controlling entity of identifier

message: serialized data structure

event: actionable message

key event: key management operation

More KERI Nomenclature

inception event: unique self-signed event that creates identifier and controlling key(s)

rotation event: self-signed uniquely ordered event from a sequence that changes the set of controlling keys

verifier: cryptographically verifies signature(s) on an event message.

witness: entity that may receive, verify, and store key events for an identifier. Each witness controls its own identifier used to sign key event messages, *controller* is a special case of witness.

receipt: event message or reference with one or more witness signatures

Even More KERI Nomenclature

key event log: ordered record of all self-signed key event messages

key event receipt log: ordered record of all key event receipts for a given set of witnesses

validator: determines current authoritative key set for identifier from at least one key event (receipt) log.

judge: determines current authoritative key set for identifier from the key event receipt logs from a set of witnesses.

pre-rotation: commitment to next rotated key set in previous rotation or inception event

BACKGROUND