Key Event Receipt Infrastructure

A Secure Identifier Overlay for the Internet

Samuel M. Smith Ph.D.
sam@keri.one
https://keri.one
version 2.60
2021/04/23

Resources

DIF Identity and Discovery WG:
Documentation:
https://github.com/decentralized-identity/keri

Meetings:
Tuesdays:
Specification mtg. 2 pm UTC (7 am MT)
Developer mtg. 3 pm UTC (8 am MT)
Zoom: https://us02web.zoom.us/j/87321306589?pwd=MSs5dlJYR0hOYjBCbWJOSmR3TDQwdz09
Agenda: https://github.com/decentralized-identity/keri/blob/master/agenda.md

Implementations:
Python: https://github.com/decentralized-identity/keripy
Rust: https://github.com/decentralized-identity/keriox
Go: https://github.com/decentralized-identity/kerigo
JavaScript: https://github.com/decentralized-identity/kerijs
Java: coming

Other
https://keri.one
https://arxiv.org/abs/1907.02143
https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP.web.pdf
https://github.com/SmithSamuelM/Papers/blob/master/presentations/KERI_Overview.web.pdf
TOIP ACDC (Authentic Chained Data Containers) TaskForce:
https://wiki.trustoverip.org/display/HOME/ACDC+%28Authentic+Chained+Data+Container%29+Task+Force

Background References

Self-Certifying Identifiers:

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

Mazieres, D. and Kaashoek, M. F., "Escaping the Evils of Centralized Control with self-certifying pathnames," MIT Laboratory for Computer Science,
http://www.sigops.org/ew-history/1998/papers/mazieres.ps

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., "Self-certifying File System," MIT Ph.D. Dissertation, 2000/06/01
https://pdos.csail.mit.edu/~ericp/doc/sfs-thesis.ps

TCG, "Implicit Identity Based Device Attestation," Trusted Computing Group, vol. Version 1.0, 2018/03/05
https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

Autonomic Identifiers:

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

Smith, S. M., "Key Event Receipt Infrastructure (KERI) Design and Build", arXiv, 2019/07/03 revised 2020/04/23
https://arxiv.org/abs/1907.02143

Smith, S. M., "Key Event Receipt Infrastructure (KERI) Design", 2020/04/22
https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf

Stocker, C., Smith, S. and Caballero, J., "Quantum Secure DIDs," RWOT10, 2020/07/09
https://github.com/WebOfTrustInfo/rwot10-buenosaires/blob/master/final-documents/quantum-secure-dids.pdf

Certificate Transparency:

Laurie, B., "Certificate Transparency: Public, verifiable, append-only logs," ACMQueue, vol. Vol 12, Issue 9, 2014/09/08
https://queue.acm.org/detail.cfm?id=2668154

Google, "Certificate Transparency,"
http://www.certificate-transparency.org/home

Laurie, B. and Kasper, E., "Revocation Transparency,"
https://www.links.org/files/RevocationTransparency.pdf

Human *Basis-of-Trust*

“in person”

I can know you – therefore I can trust you

“on the internet”

I can’t really know you – therefore I can’t really trust you

Secure Attribution Problem

Secure attribution of any communication to its source

Authentic communication

Authentic interactions based an secure attribution of all statements by participants

Verifiable authenticity of data

Data Provenance

Authentic data economy

Replace human *basis-of-trust* with cryptographic *root-of-trust.*

With verifiable digital signatures from asymmetric key crypto –
we may not trust in "*what*" was said, but we may trust in "*who*" said it.

We may *verify* that the *controller* of a private key, (*the who*), made a statement
but not the validity of the statement itself.

The *root-of-trust* is *consistent attribution* via verifiable integral non-repudiable statements

We may build trust over time in *what* was said via histories
of verifiably attributable (to *whom*) consistent statements i.e. *reputation*.

KERI is the Platypus of the identity space

Platypus has a unique combination of features

Mammal (milk), Reptile (eggs), Bird (bill), Ray (electro-location), Snake (venom)

KERI is a unique combination of features that solve the secure attribution problem.

KERI has elements from DNS/CA, PGP, Blockchain, etc.

Closest comparison is the DNS/CA system.

Provide perspective and set expectations about KERI

Secure Attribution System Design Problem

Mission (security etc) Survivability = …

Susceptibility, Vulnerability, and Recoverability

Failure is inevitable. Preventing all failures is futile.

Instead ….

Never fail to detect failure

High fidelity failure detection limits exposure to failure = protection

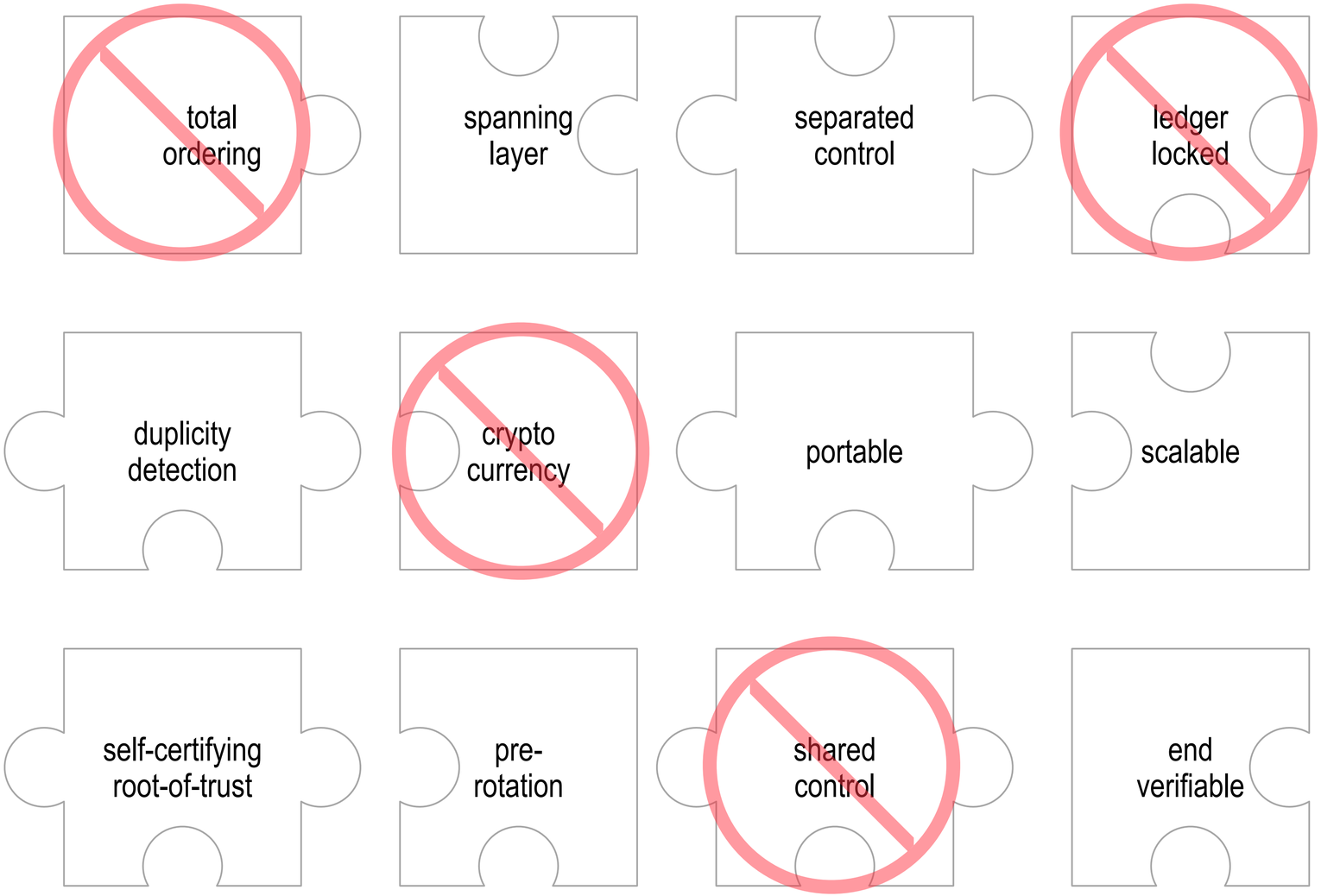
Three principles:

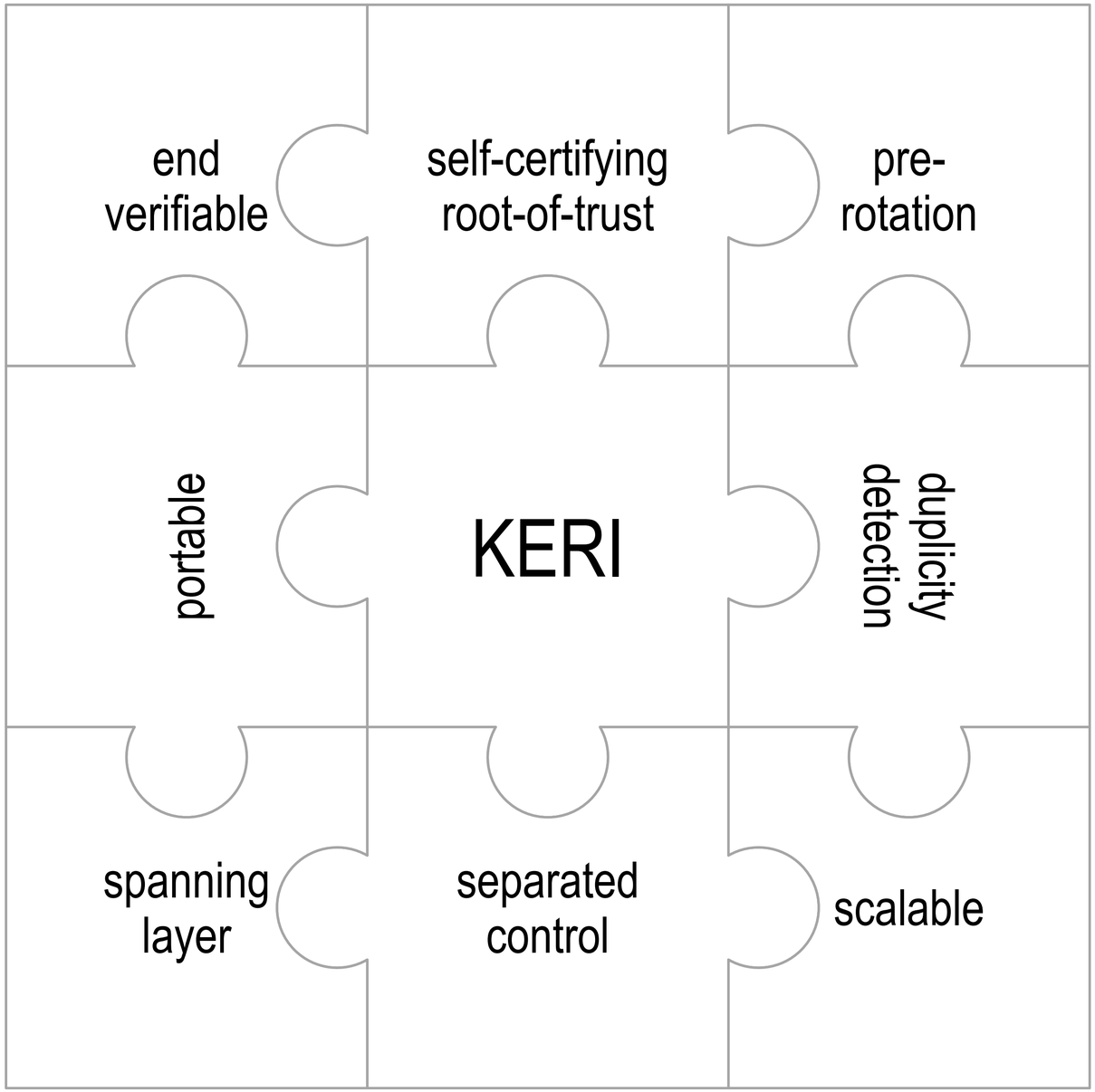
1) Make authentic sources of statement responsible for any failure in the secure attribution of their statements.

2) Enable the secure attribution of any failure to the source of that failure.

3) Failure detection via duplicity detection

System Design Trade Space




end verifiable
self-certifying root-of-trust
pre-rotation
portable
KERI
duplicity detection
spanning layer
separated control
scalable

KERI for Short

Open Source: Apache 2:

Scalability:

Designed for performance, asynchronous, bare metal TCP, UDP, non-enveloped.

Self-Certifying Identifiers:

Root-of-trust in cryptographic derivation from entropy.

Truly Decentralized Identity:

Control over identifiers is not dependent on shared control of resources.

End Verifiable Authenticity:

Zero-trust, key event logs as proofs, no trusted entities, no trust in intervening infrastructure, end state is ambient verifiability.

Decentralized Key Management Infrastructure:

Pre-rotation for key rotation for compromise recovery, post-quantum secure, built-in multi-sig, weighted multi-sig.

Delegated Identifiers for Enterprise Key Management:

Multi-valent key management infrastructure for scalability and more secure compromise recovery.

Supports GDPR Compliance:

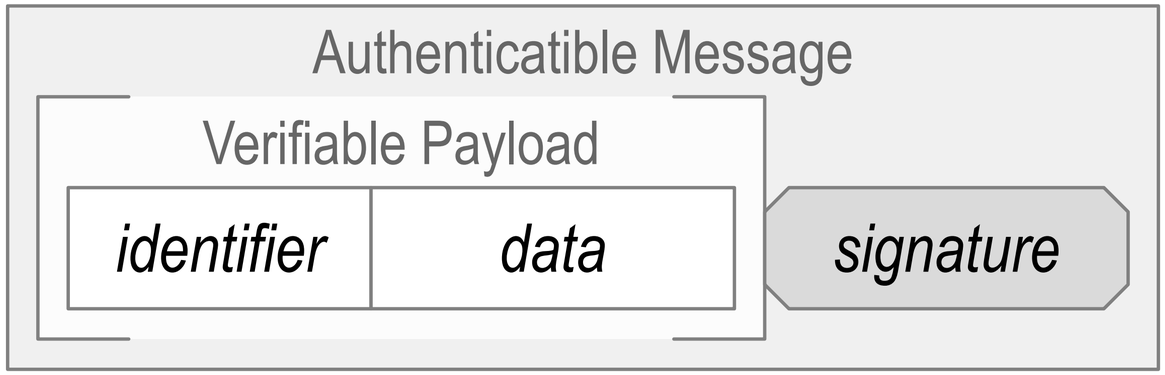
Separable identifier trust bases. Non-intertwined KELs.

Supports Electronic Digital Signature Compliance (EiDAS, UETA, E-SIGN):

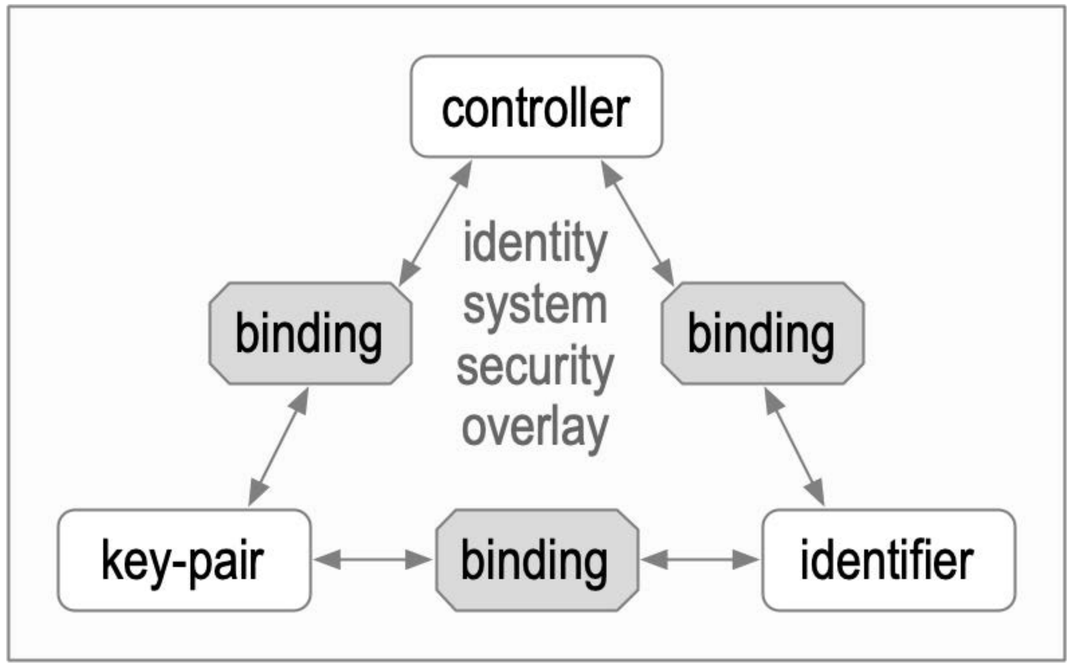
Non-repudiable digital signatures under sole control of identifier controller satisfies core condition for legal liability.

Identity System Security Overlay

Establish authenticity of IP packet's message payload.

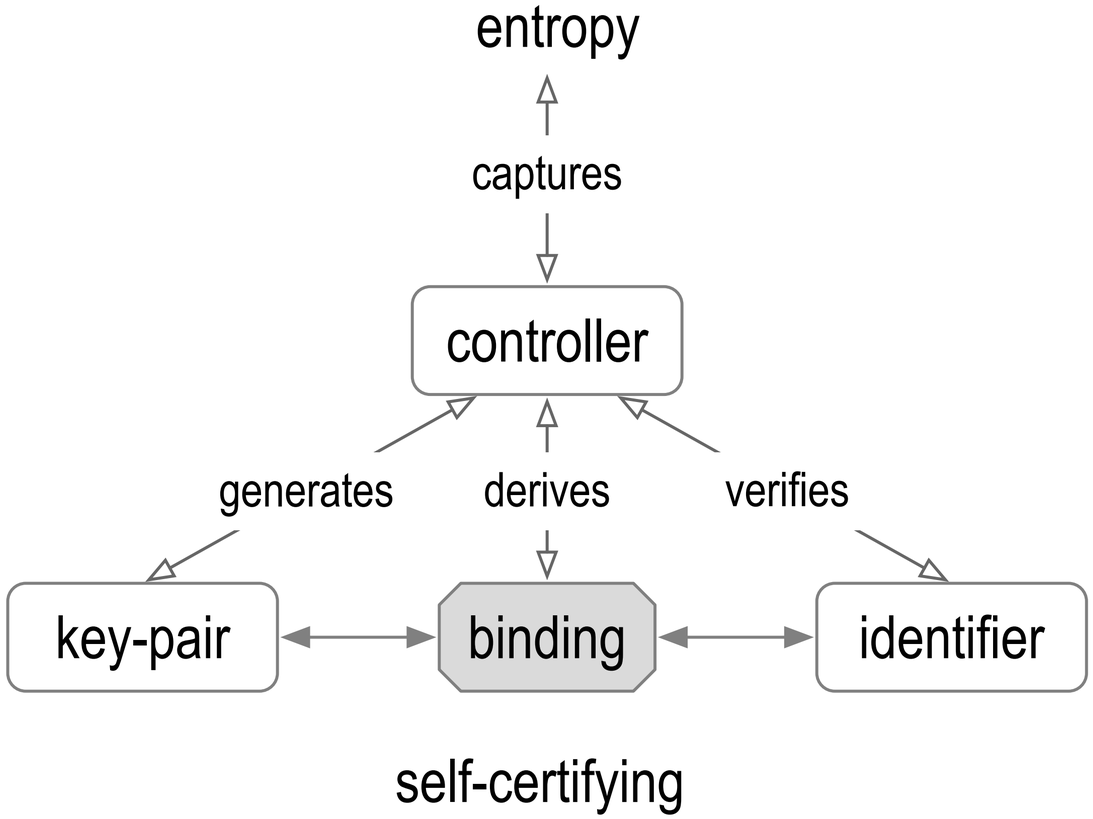


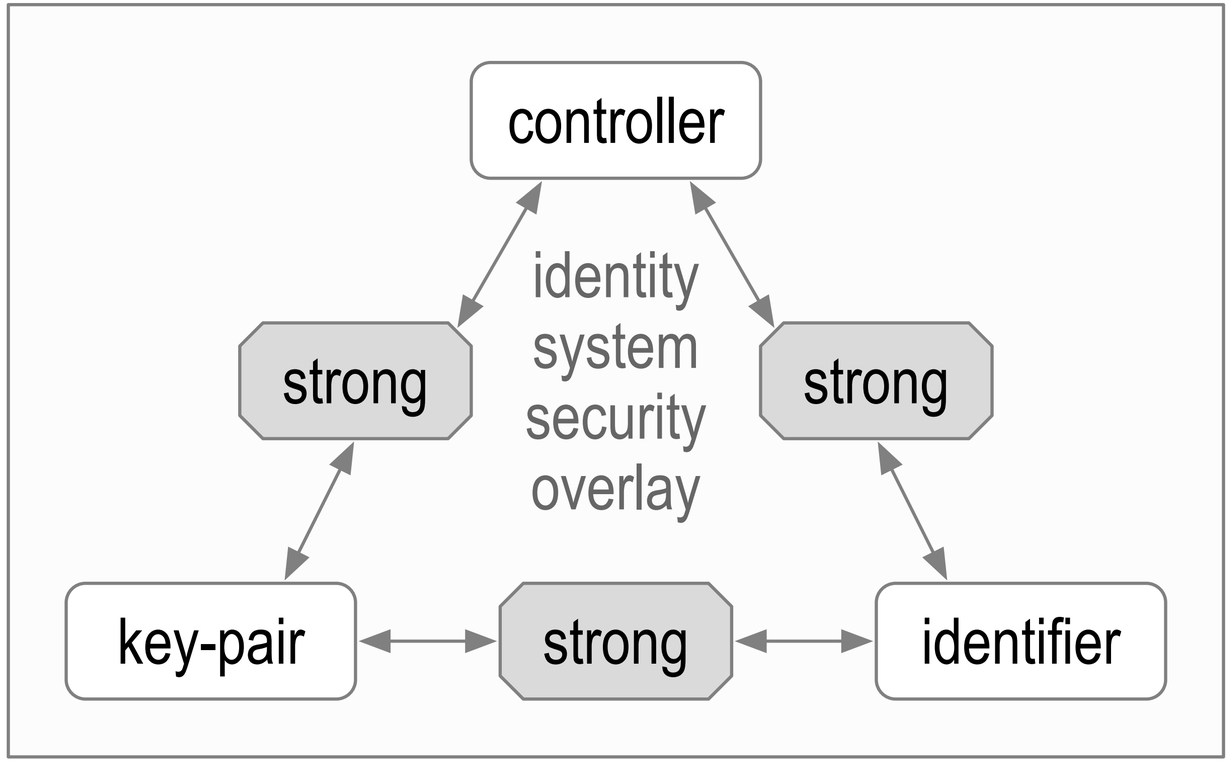
The overlay's security is contingent on the mapping's security.



Identifier Issuance

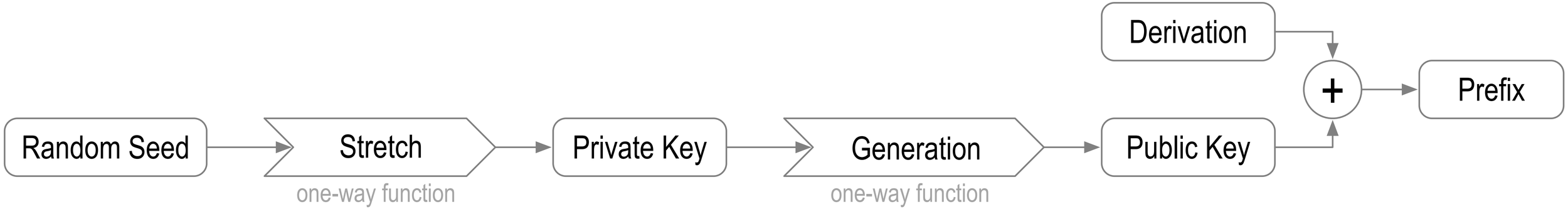
Self-Certifying Identifier Issuance and Binding

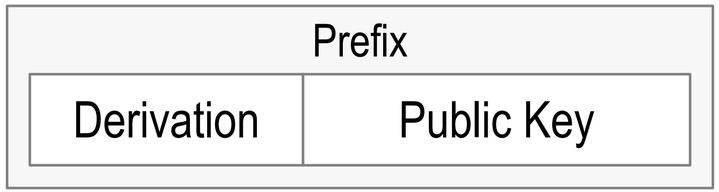


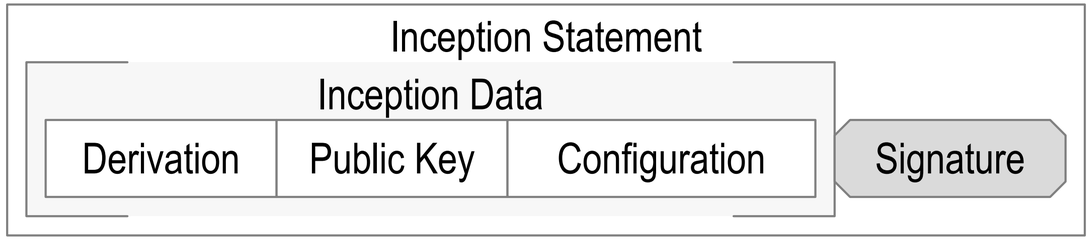


Self-Certifying Identifier Issuance

Basic SCID



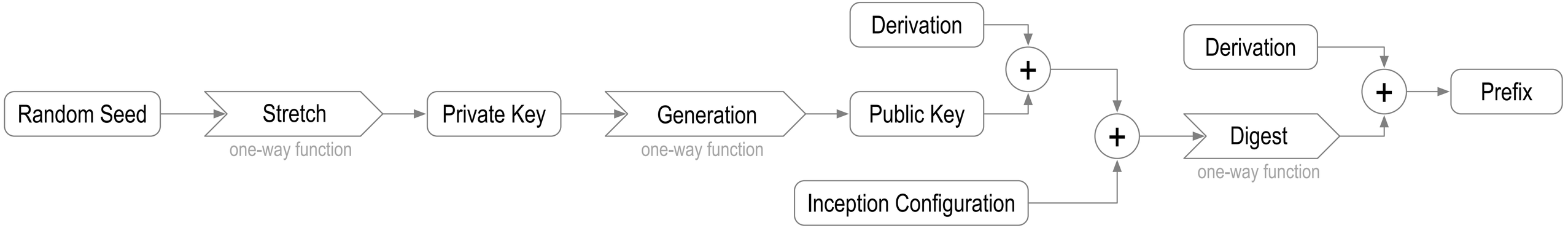


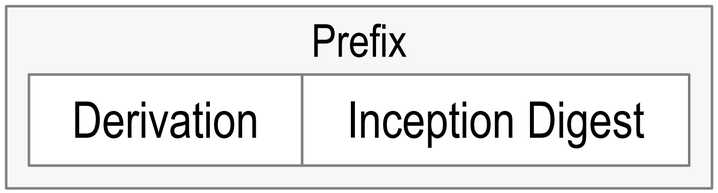


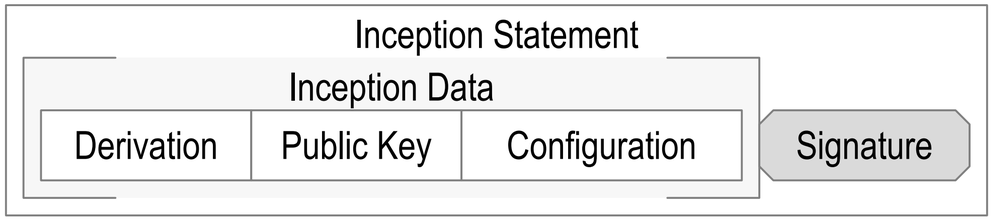
`BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0`

`did:un:BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0/path/to/resource?name=secure#really`

Self-Addressing SCID



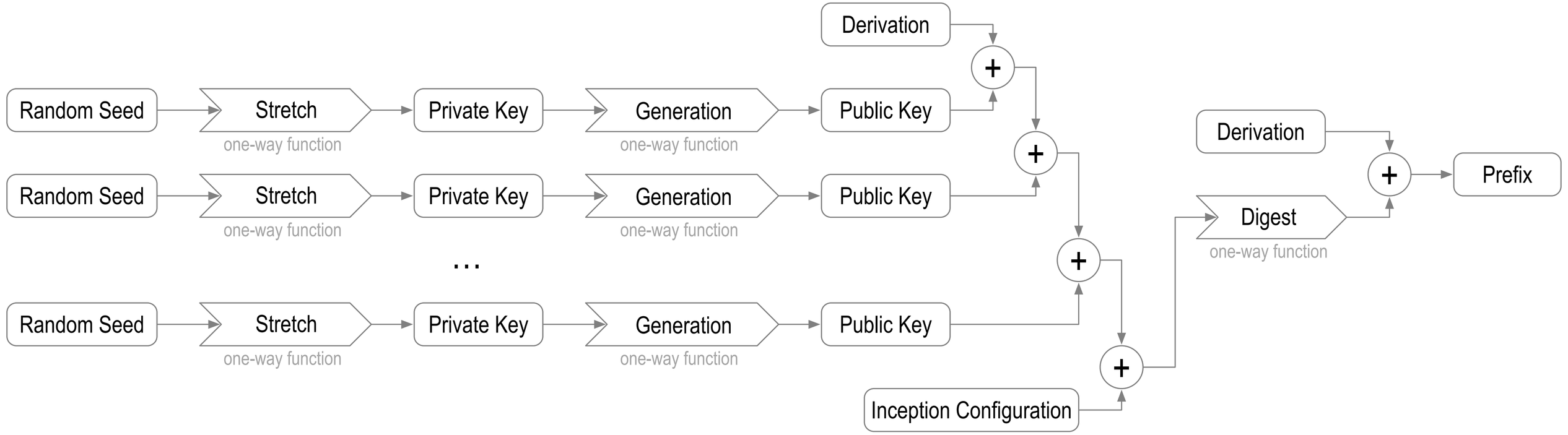


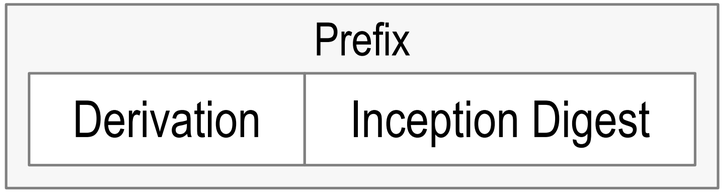


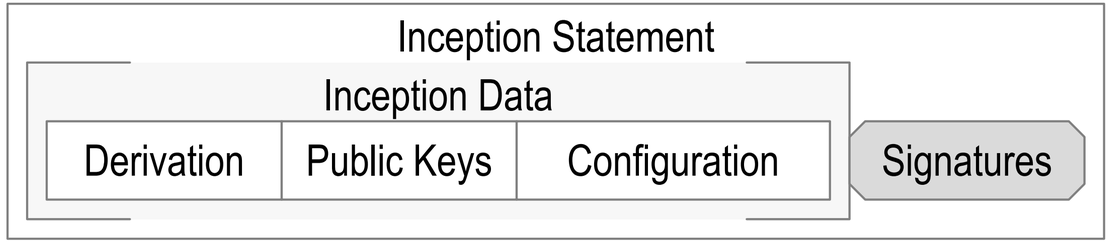
`EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148`

`did:keri:EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148/path/to/resource?name=secure#this`

Multi-Sig Self-Addressing SCID



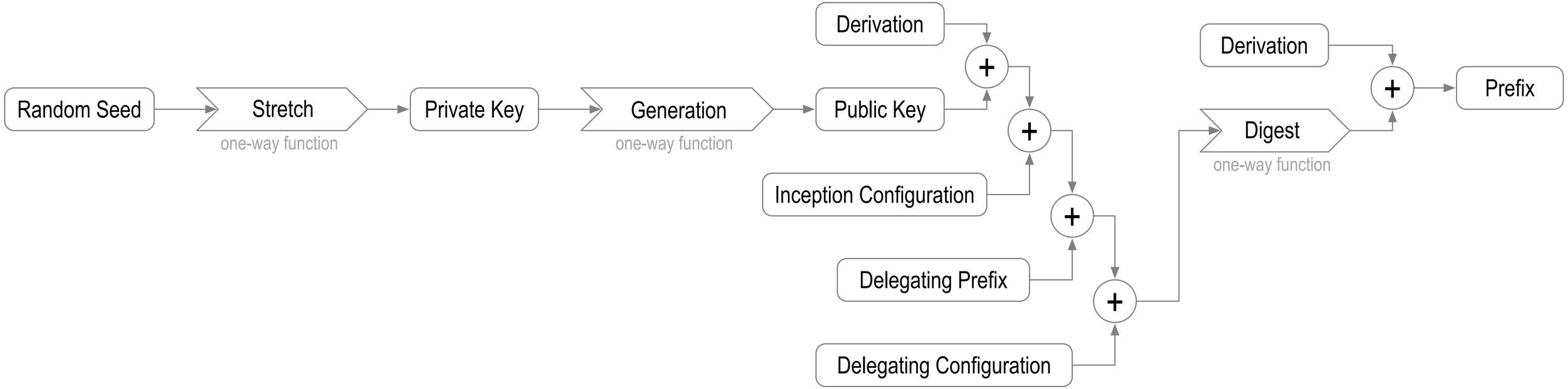


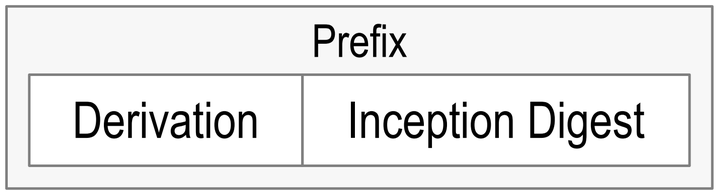


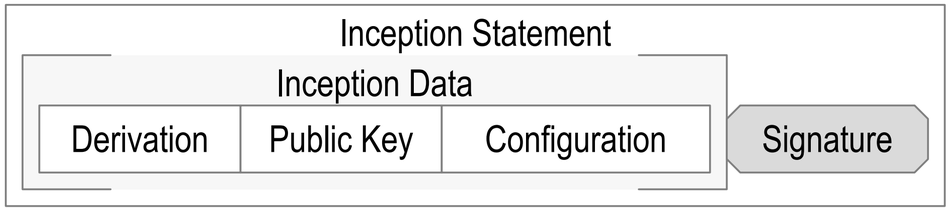
EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

did:un:EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148/path/to/resource?name=secure#really

Delegated Self-Addressing SCID



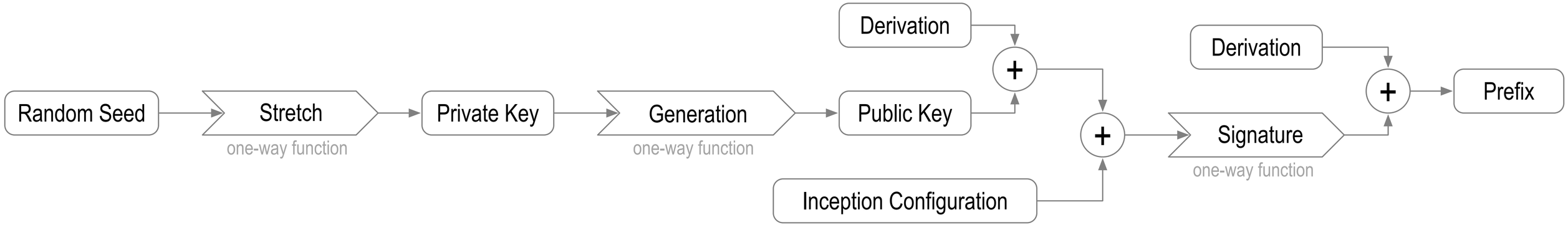


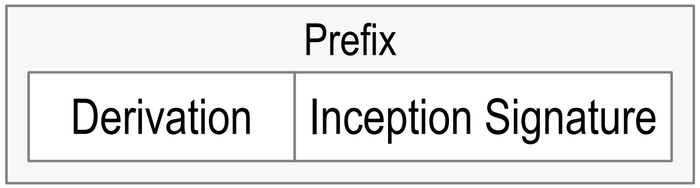


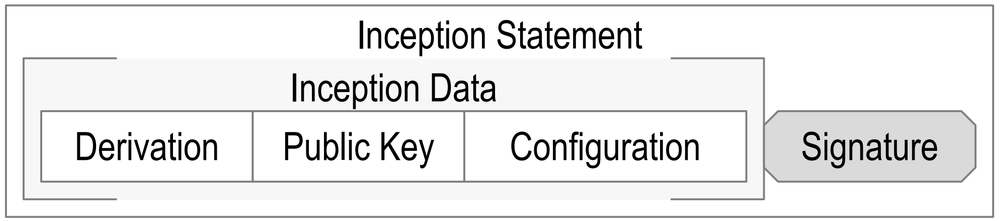
EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

did:un:EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148/path/to/resource?name=secure#really

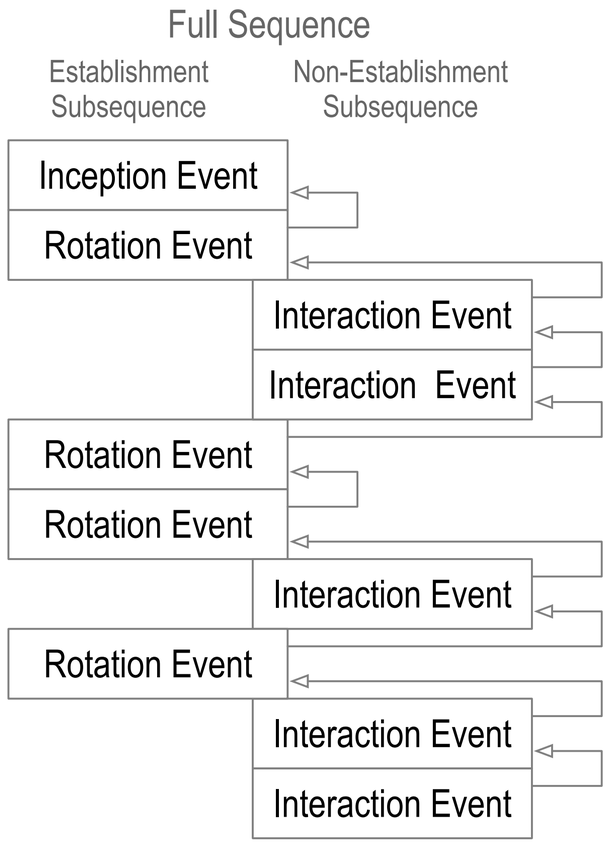
Self-Signing SCID







Inconsistency and Duplicity



inconsistency: lacking agreement, as two or more things in relation to each other

duplicity: acting in two different ways to different people concerning the same matter

Internal vs. External Inconsistency

Internally inconsistent log = not verifiable.

Log verification from self-certifying root-of-trust protects against internal inconsistency.

Externally inconsistent log with a purported copy of log but both verifiable = duplicitous.

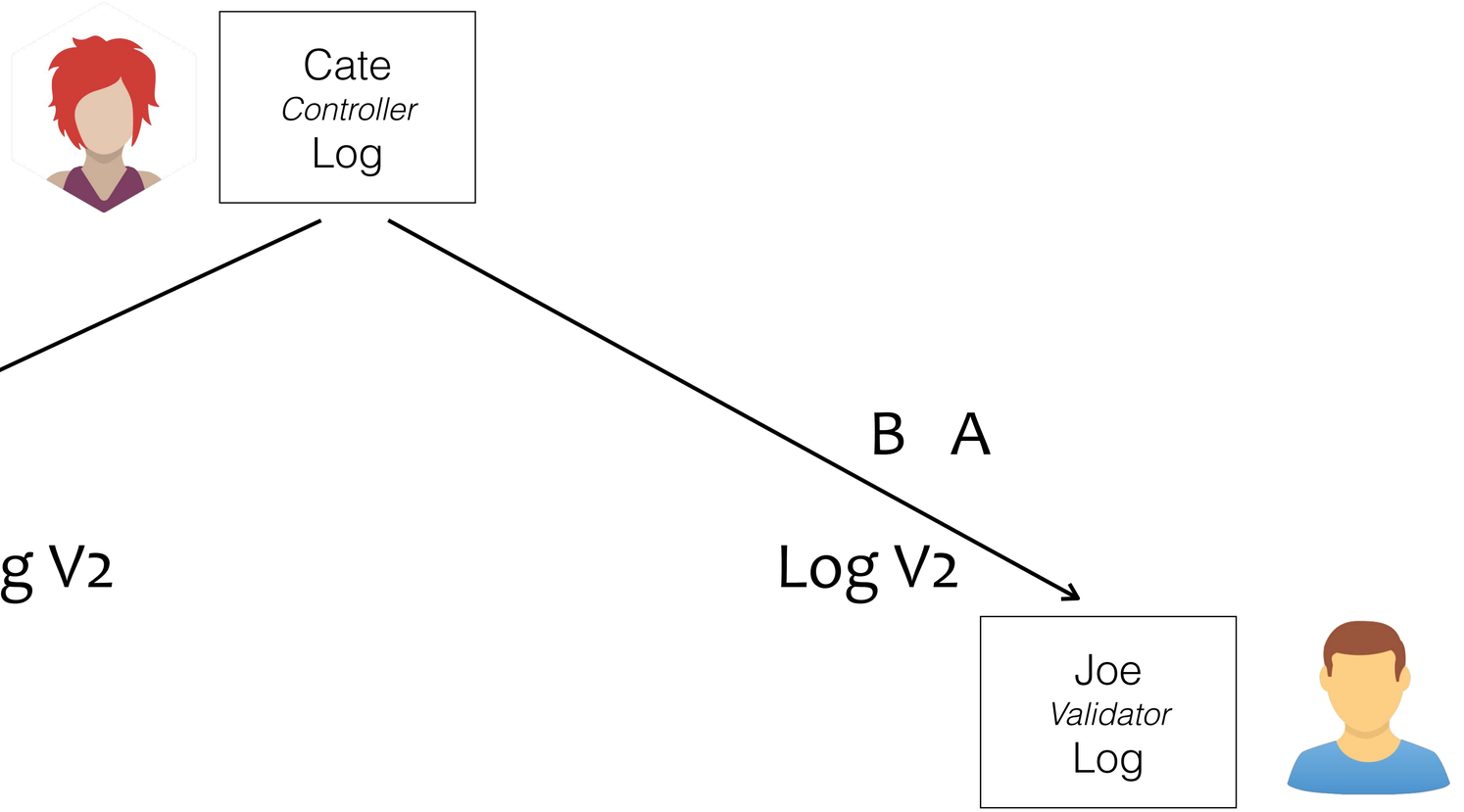
Duplicity detection protects against external inconsistency.

Duplicity Game

Cate promises to provide a consistent pair-wise log.

Local Consistency Guarantee

How may Cate be duplicitous and not get caught?



private (one-to-one) interactions

Duplicity Game

Service promises to provide a consistent log to anyone.

Local Consistency Guarantee

How may Cate/Service/Agent be duplicitous and not get caught?

Cate
Controller
Log

Truncate Log

Delete Log

Service/Agent
Controlled by Cate
Log

A

B A

Log V1 Log V2

Log V2

Eve
Validator
Log

Joe
Validator
Log

highly available, private (one-to-one) interactions

Duplicity Game

Service promises to provide exact same log to everyone.

Global Consistency Guarantee

How may Cate and/or service be duplicitous and not get caught?

Cate
Controller
Log

Breaking the promise of global consistency is a provable liability.

Global consistency may only matter after Eve and Joe need to interact not before.

Service
Controlled by Cate
Log

Truncate Log

?

isolate network

isolate network

A

Delete Log

A

log V1

log V2

Doug
Global Duplicity
Log

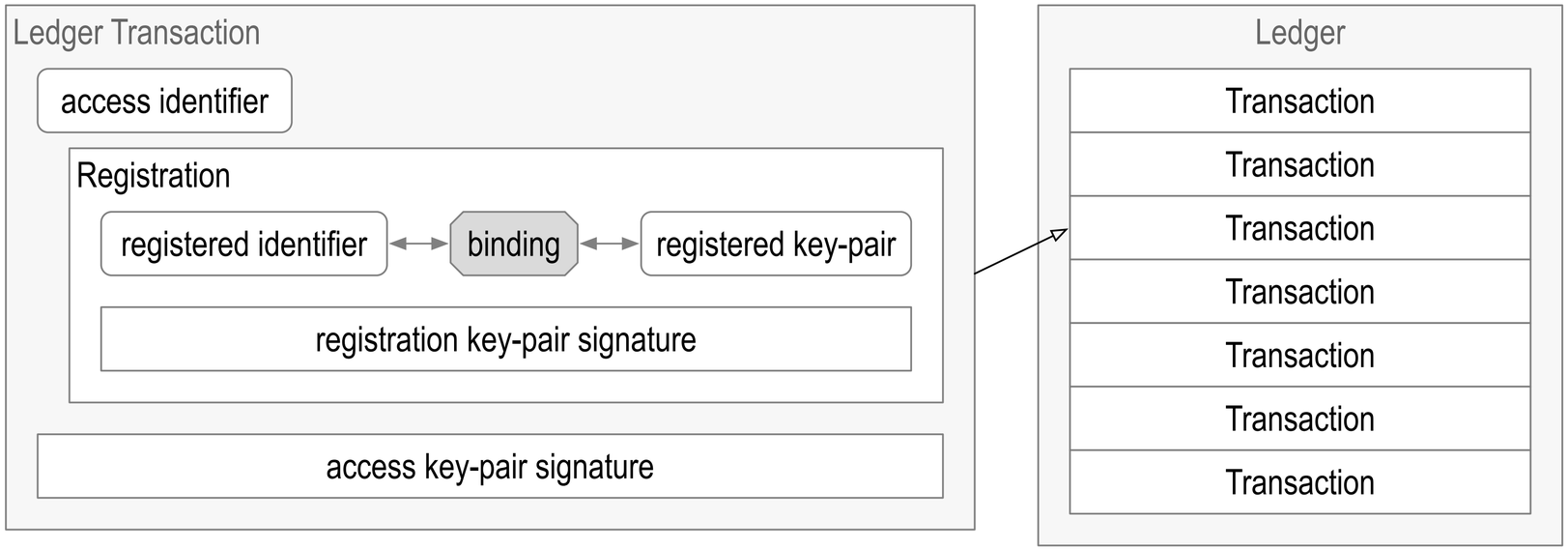
Eve
Validator
Log

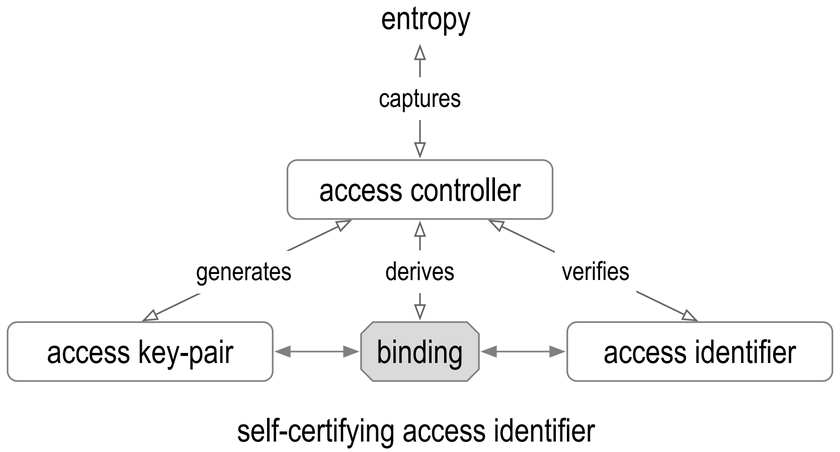
Ambient Duplicity Detection

Joe
Validator
Log

global consistent, highly available, and public (one-to-any) interactions

Ledger Registration





The access identifier may have a self-certifying primary root-of-trust, but the registered identifier does not, even if its format appears to be self-certifying.

Autonomic Identifier (AID) and Namespace (AN)

auto nomos = self rule

autonomic = self-governing, self-controlling, etc.

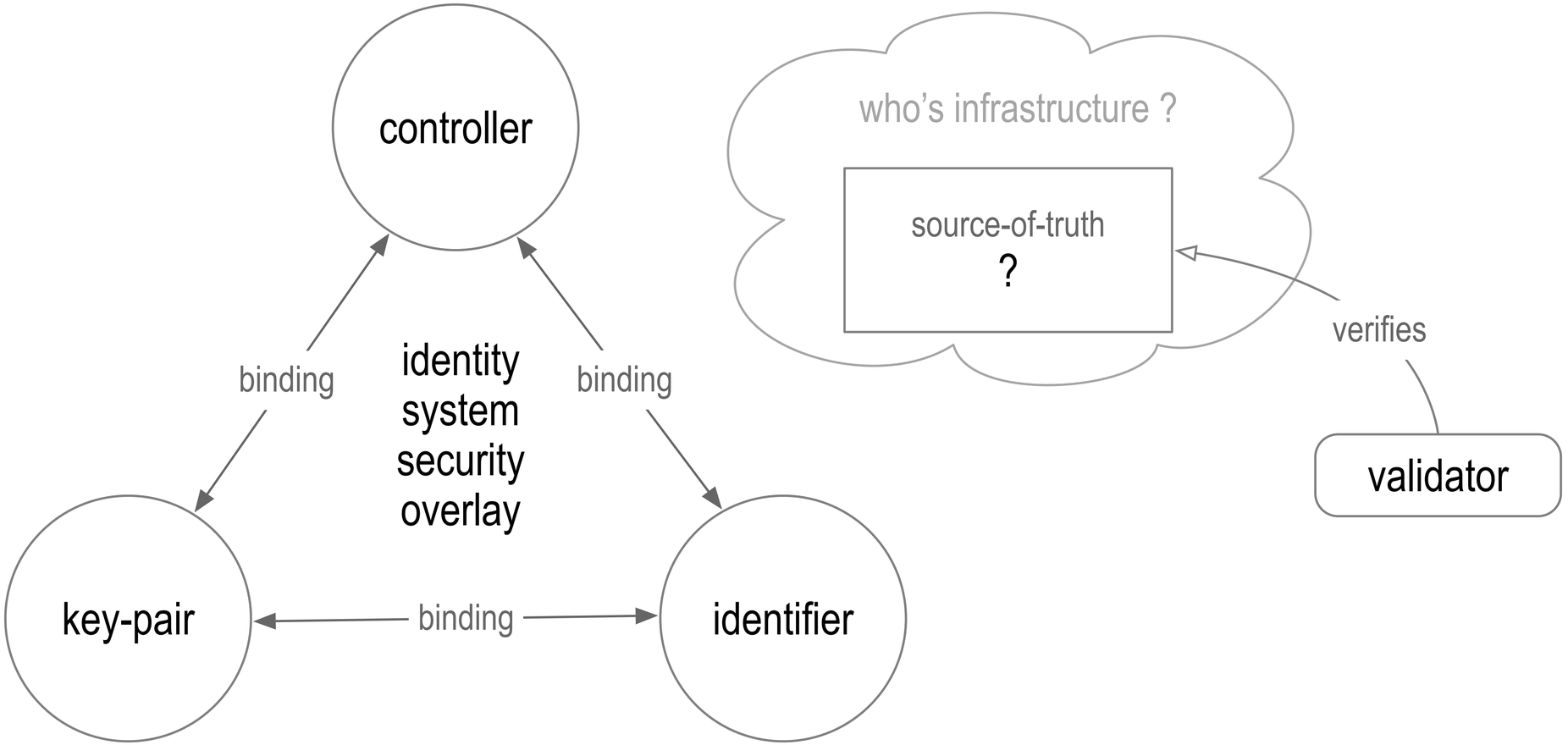
An *autonomic* namespace is

self-certifying and hence *self-administrating.*

AIDs and ANs are *portable* = truly self-sovereign.

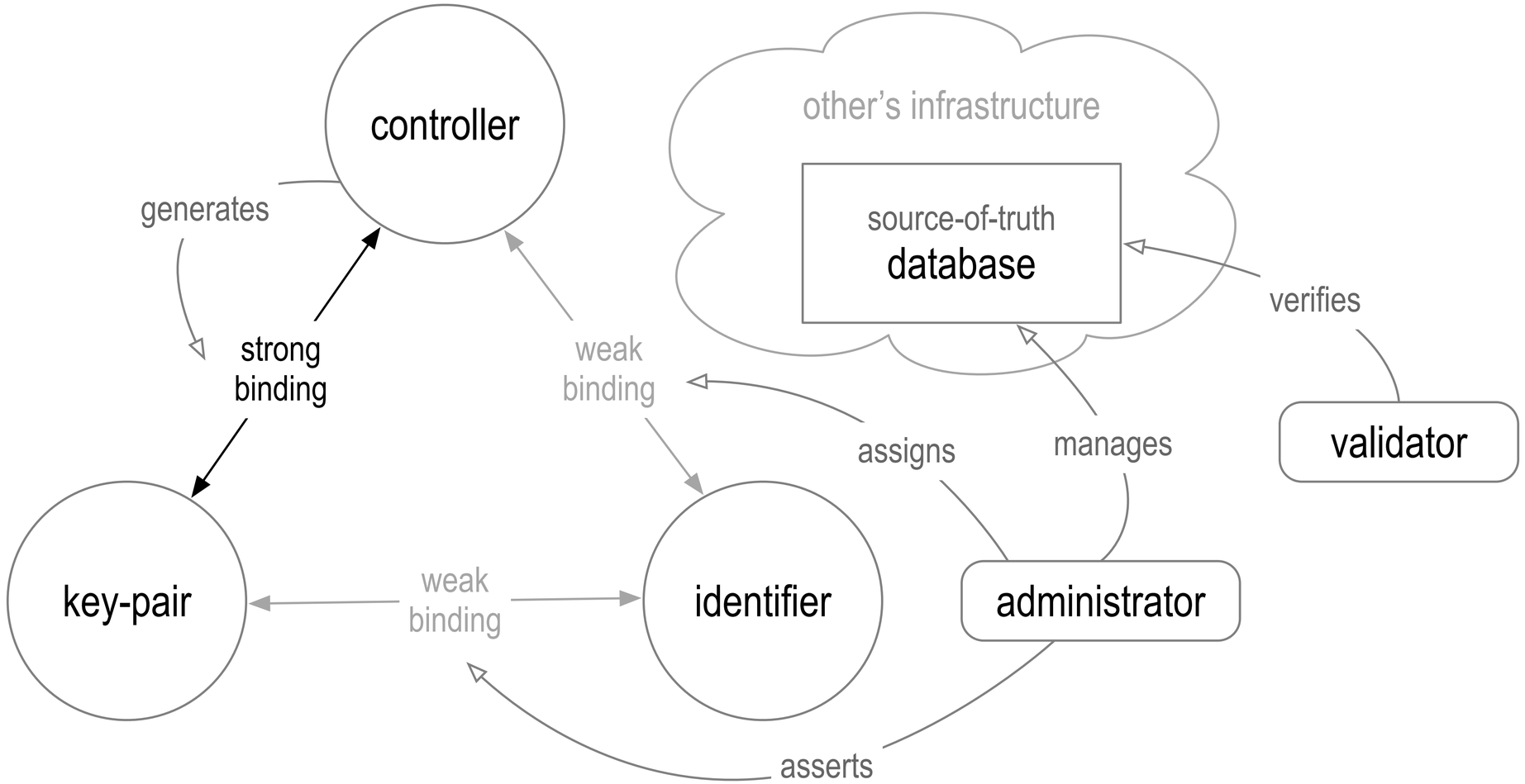
autonomic prefix = self-cert + UUID + URL = universal identifier

Trust Basis


controller
who's infrastructure ?
source-of-truth
?
verifies
validator
binding
identity
system
security
overlay
binding
key-pair
binding
identifier

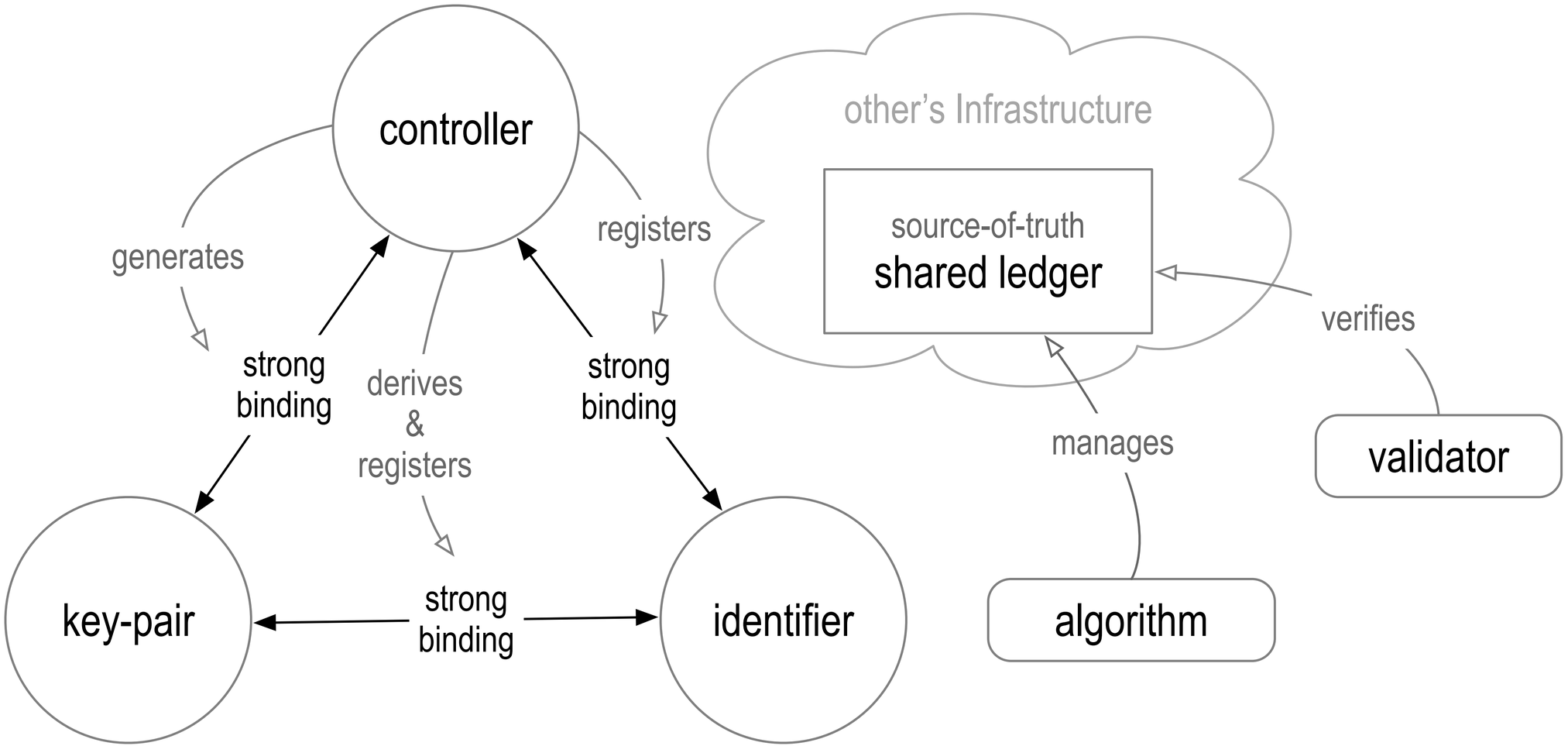
Administrative Trust Basis

DNS/Certificate Authorities



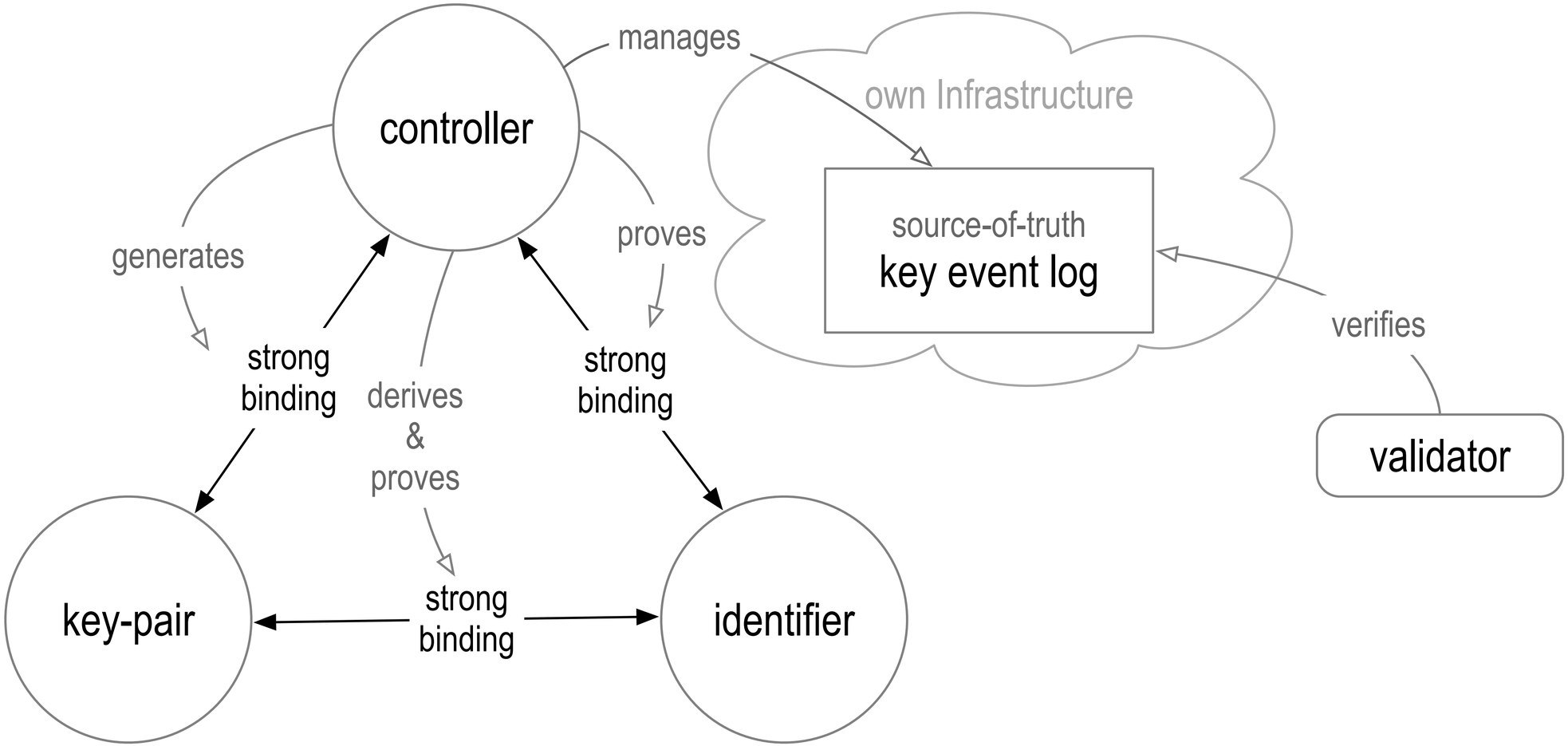
Algorithmic Trust Basis

Shared Distributed Ledgers



Autonomic Trust Basis

Cryptographic Proofs



KEY Event Based Provenance of Identifiers

KERI enables cryptographic *proof-of-control-authority* (*provenance*) for each identifier.

A *proof* is in the form of an identifier's *key event receipt log* (KERL).

KERLs are *End Verifiable*:

End user alone may verify. Zero trust in intervening infrastructure.

KERLs may be *Ambient Verifiable*:

Anyone may verify *any-log*, *anywhere*, at *anytime*.

KERI = self-cert root-of-trust + certificate transparency + KA^{2}CE + recoverable + post-quantum.

KERI for the *DID*ified

KERI non-transferable ephemeral with derivation code ~ did:key

KERI private direct mode (one-to-one) ~ did:peer

KERI public persistent indirect mode (one-to-any) ~ Indy interop, did:sov etc

KERI = did:keri (did:uni, did:un) (all of the above in one method)

`did:keri:`*`prefix`*`[:`*`options`*`][/`*`path`*`][?`*`query`*`][#`*`fragment`*`]`

```
BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0
did:keri:BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0/path/to/resource?name=secure#really
```

KERI Agnosticism and Interop

KERI itself is completely agnostic about anything but the *prefix* !

`???:prefix[:options][/path][?query][#fragment]`

The KERI layer establishes control authority over a *prefix*

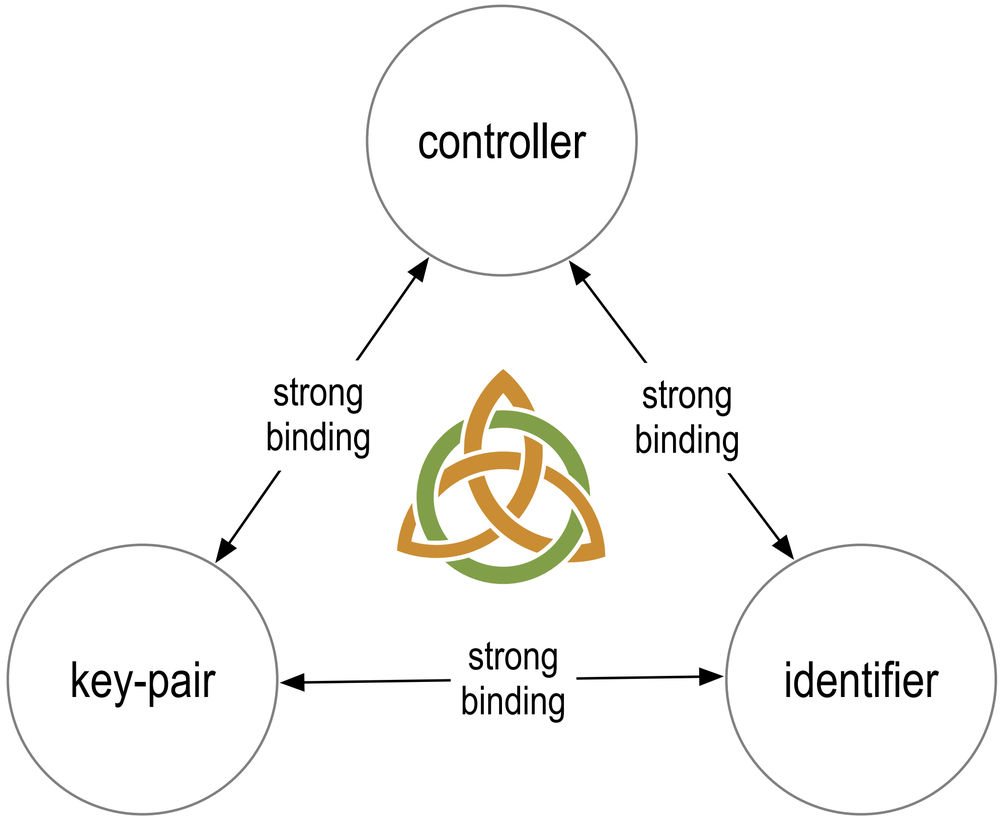
Any and *All* namespaces that share the same *prefix* may share the same KERI trust basis for control establishment over that *prefix* and hence that namespace.

Interop happens in a layer above the KERI layer

All we need for bootstrapping interop is some indication that the *prefix* inside identifier is KERI based (KERI trust basis).

`BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0`
`did:indy:sov:keri:BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUtlDdhh0`

Autonomic Identifier System



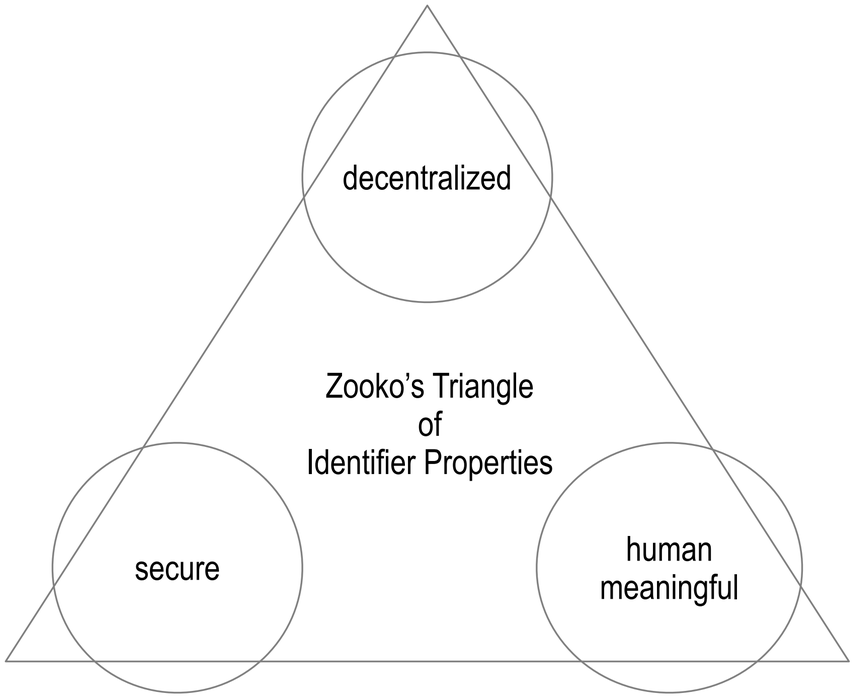
KERI

Zooko's Trilemma

Desirable identifier properties: secure, decentralized, human meaningful

Trilemma: May have any two of the three properties but not all three.

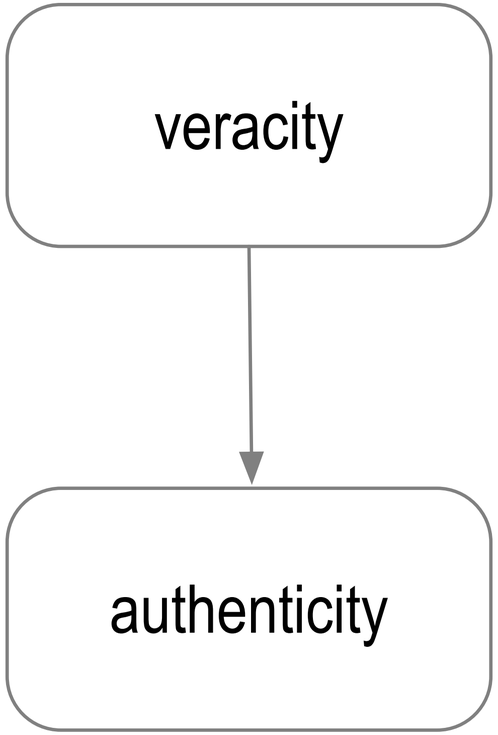
One way to sort of solve the trilemma is to uniquely register a human meaningful identifier on a ledger controlled by a different identifier that is secure and decentralized but not human meaningful.



Trust Balance

Reputational Trust

Cryptographic Trust



Unified Identifier Model

AID: Autonomic Identifier (primary)

self-managing self-certifying identifier with cryptographic root of trust

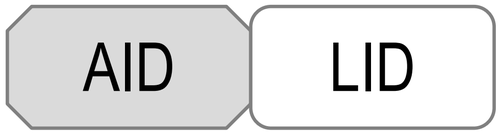
secure, decentralized, portable, universally unique

L*ID*: Legitimized Human Meaningful Identifier (secondary)

legitimized within trust domain of given AID by a verifiable authorization from AID controller

authorization is verifiable to the root-of-trust of AID

Forms *AID*|*LID* couplet within trust domain of AID

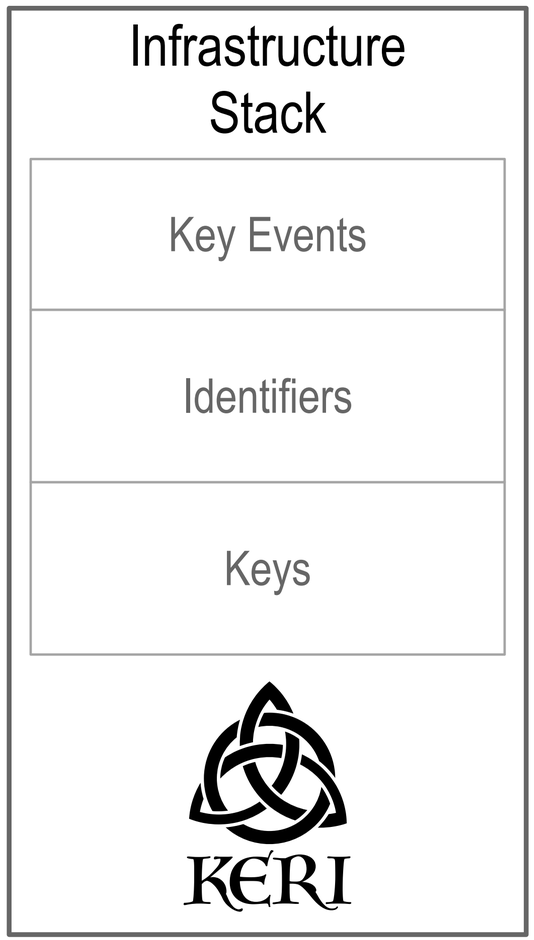


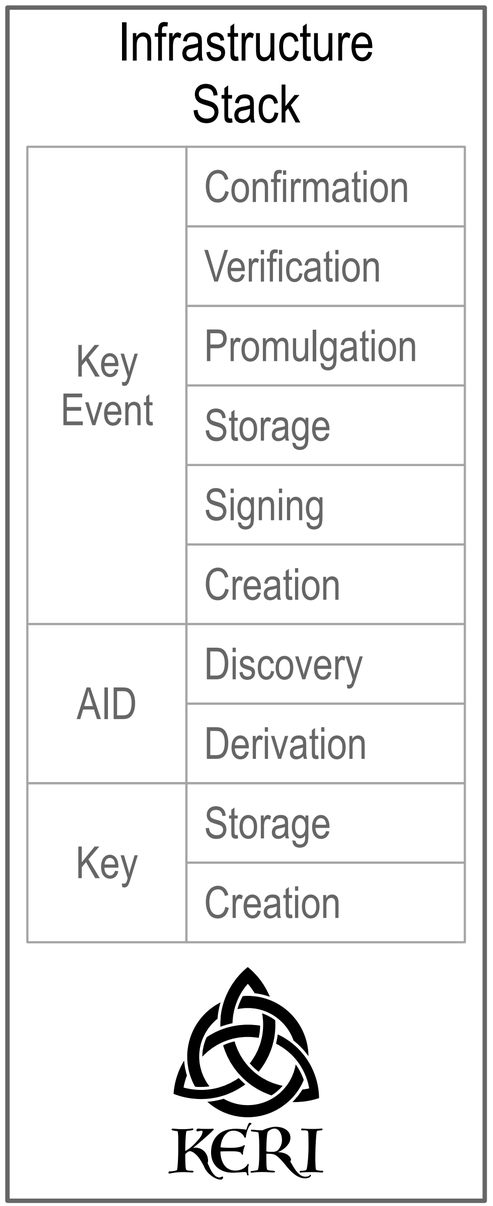
AID|LID Couplet

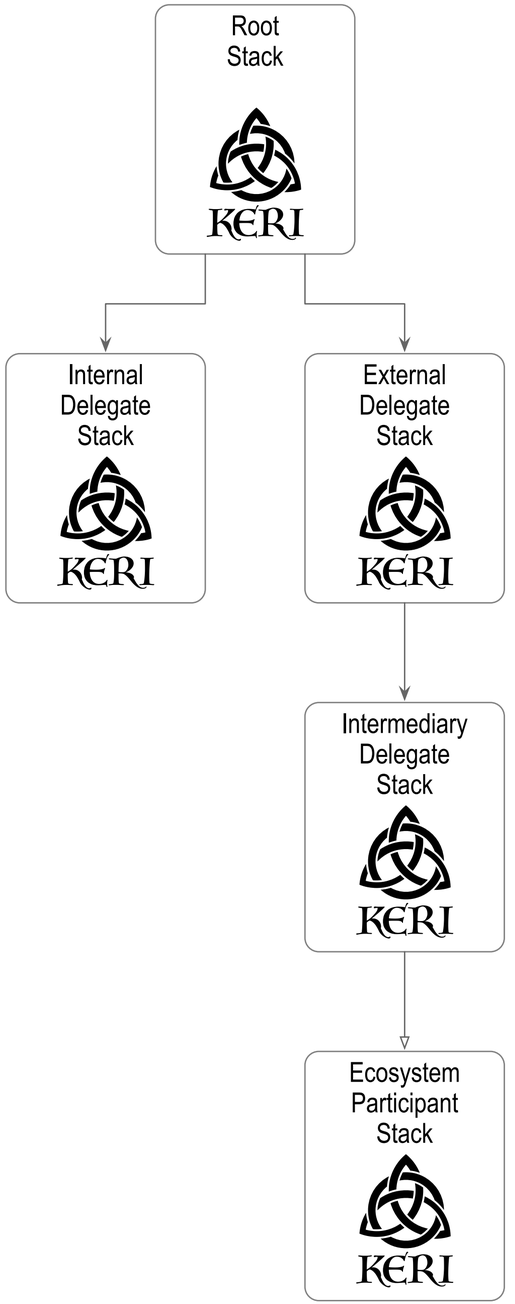
625.127C125r

EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148|625.127C125r

KERI Stack


Infrastructure Stack
Key Events
Identifiers
Keys
KERI


Infrastructure Stack
Key Event
Confirmation
Verification
Promulgation
Storage
Signing
Creation
AID
Discovery
Derivation
Key
Storage
Creation
KERI


Root Stack
KERI
Internal Delegate Stack
KERI
External Delegate Stack
KERI
Intermediary Delegate Stack
KERI
Ecosystem Participant Stack
KERI

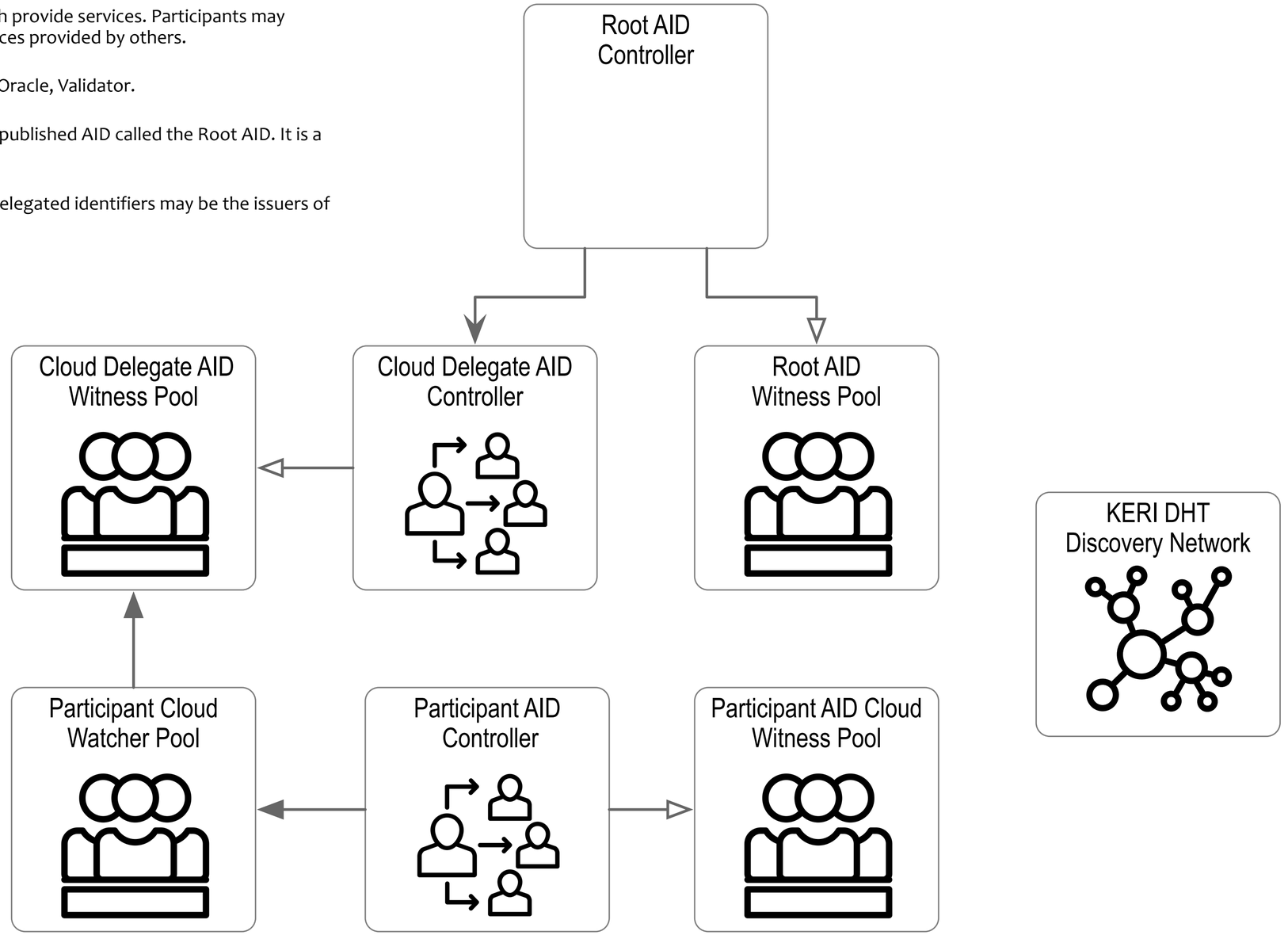
Basic KERI Stack

KERI employs a modular architecture with modular components that each provide services. Participants may configure their stacks to provide some of all of the services or share services provided by others.

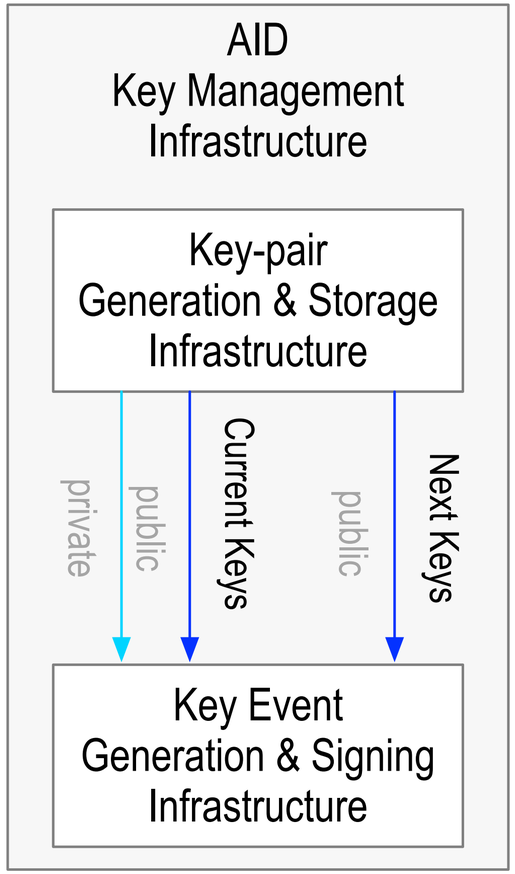
The component services include Controller, Witness, Watcher, Delegate, Oracle, Validator.

The root-of-trust for the GLEIF ecosystem is provided by a single globally published AID called the Root AID. It is a KERI DID.

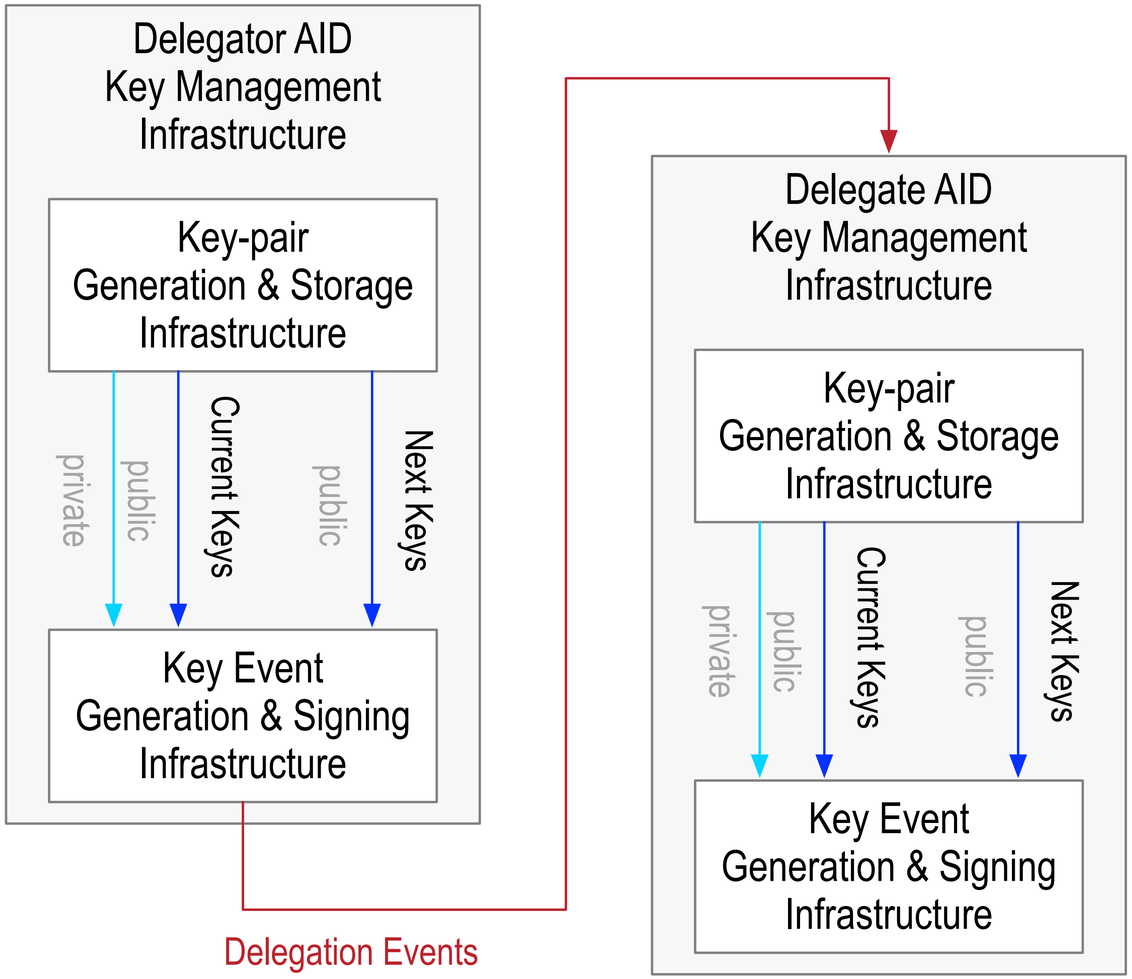
This Root AID is the issuer of delegations to other KERI AID DIDs. These delegated identifiers may be the issuers of VCs.



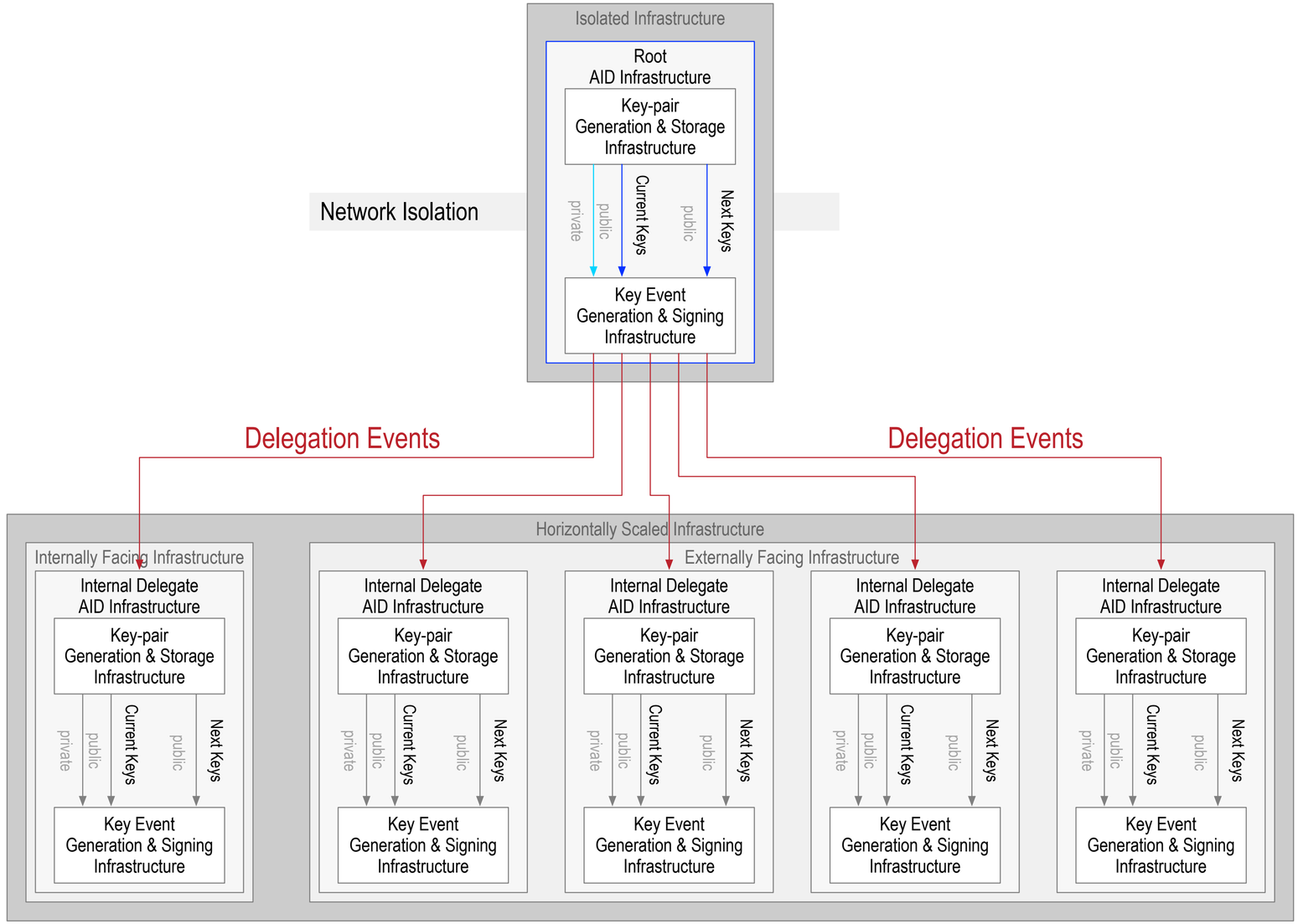
Decentralized Key Management Infrastructure (Univalent DKMI)



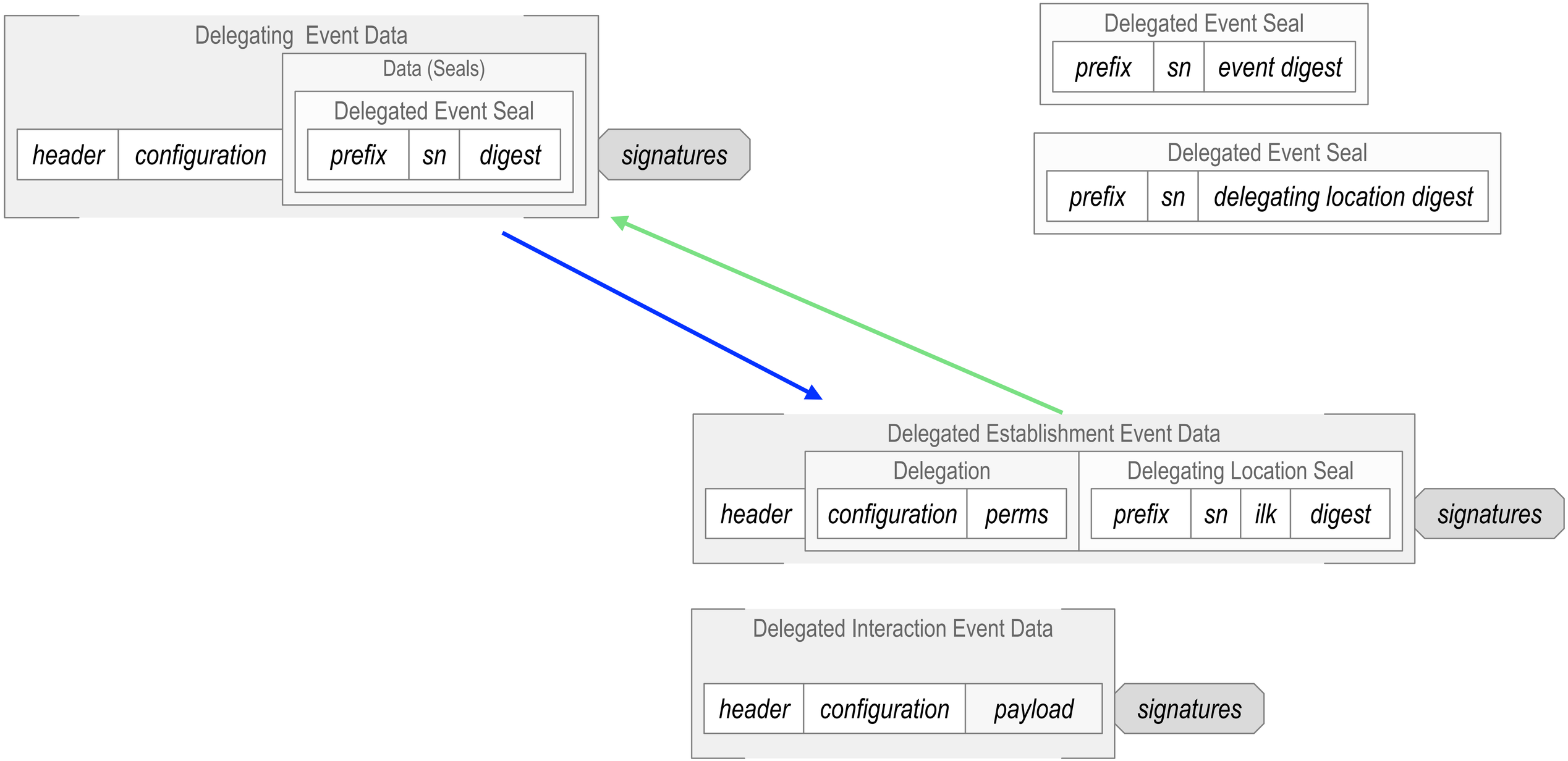
Hierarchical DKMI: Bivalent DKMI



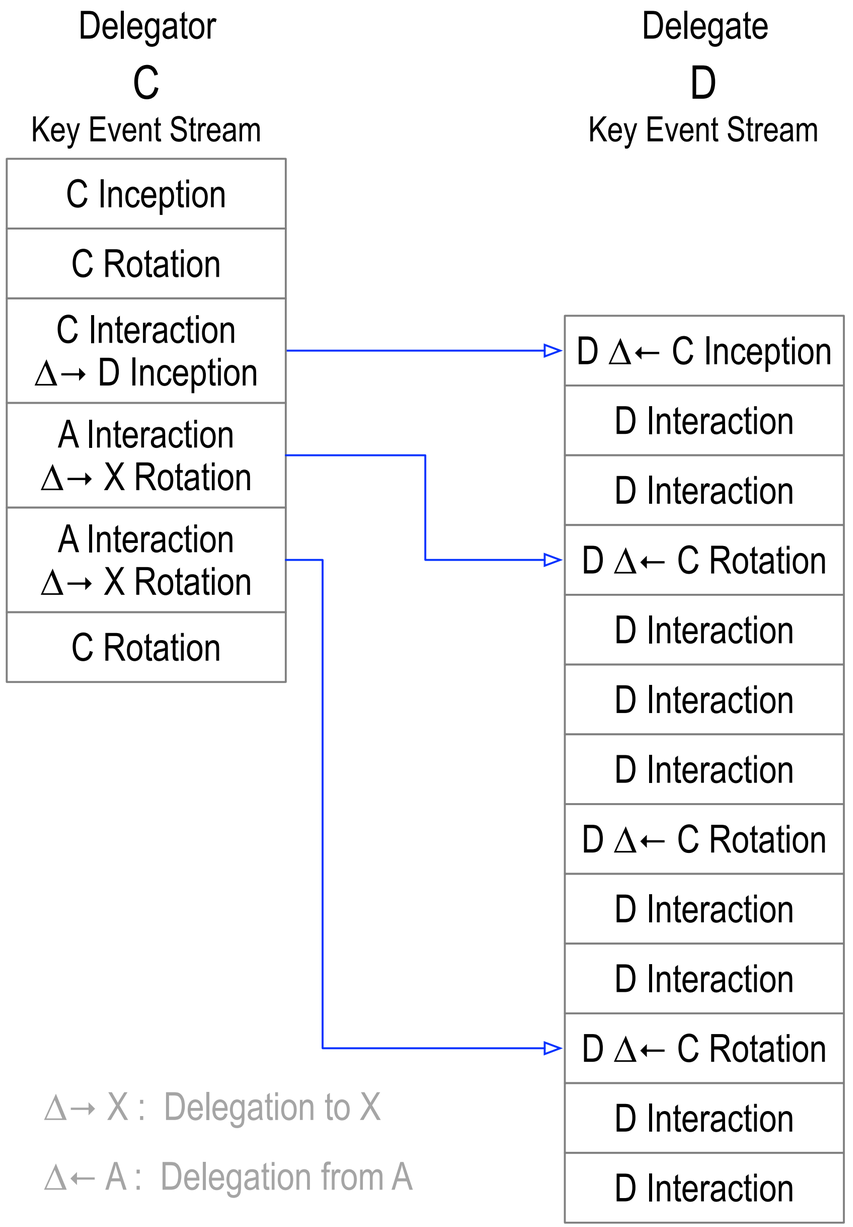
MultiValent Delegation



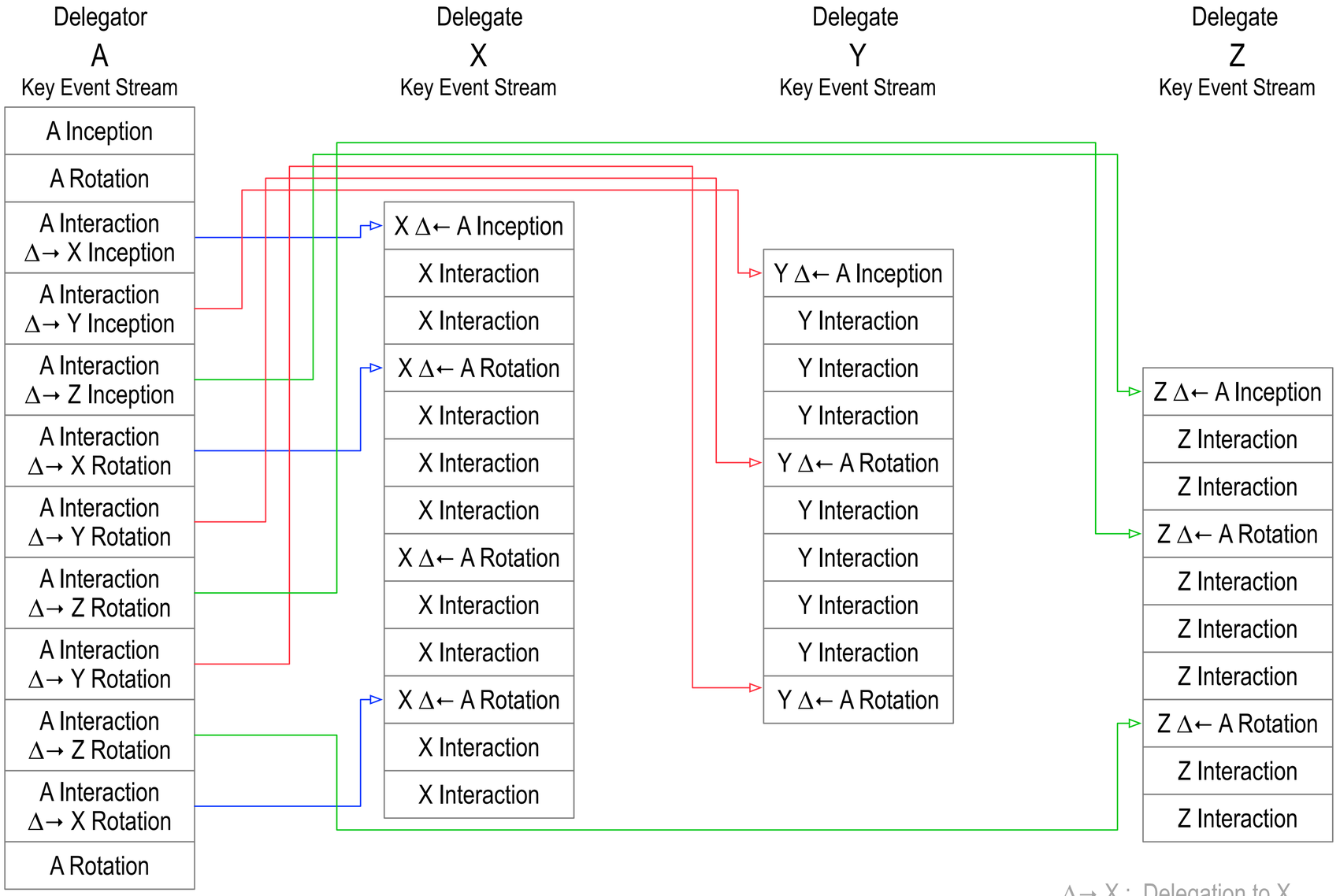
Delegation (Cross Anchor)



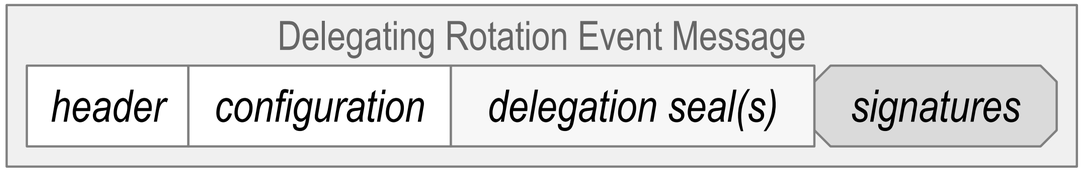
Interaction Delegation

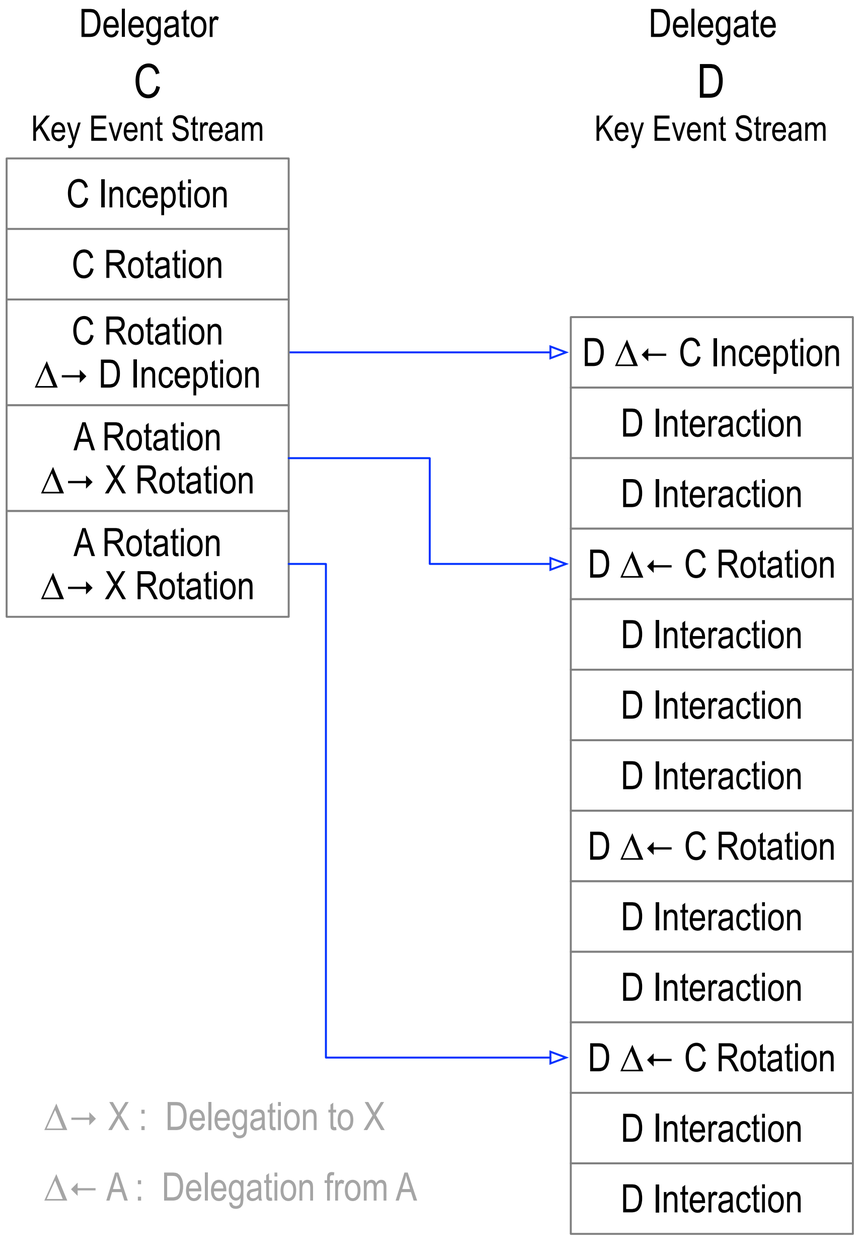


Scaling Delegation via Interaction

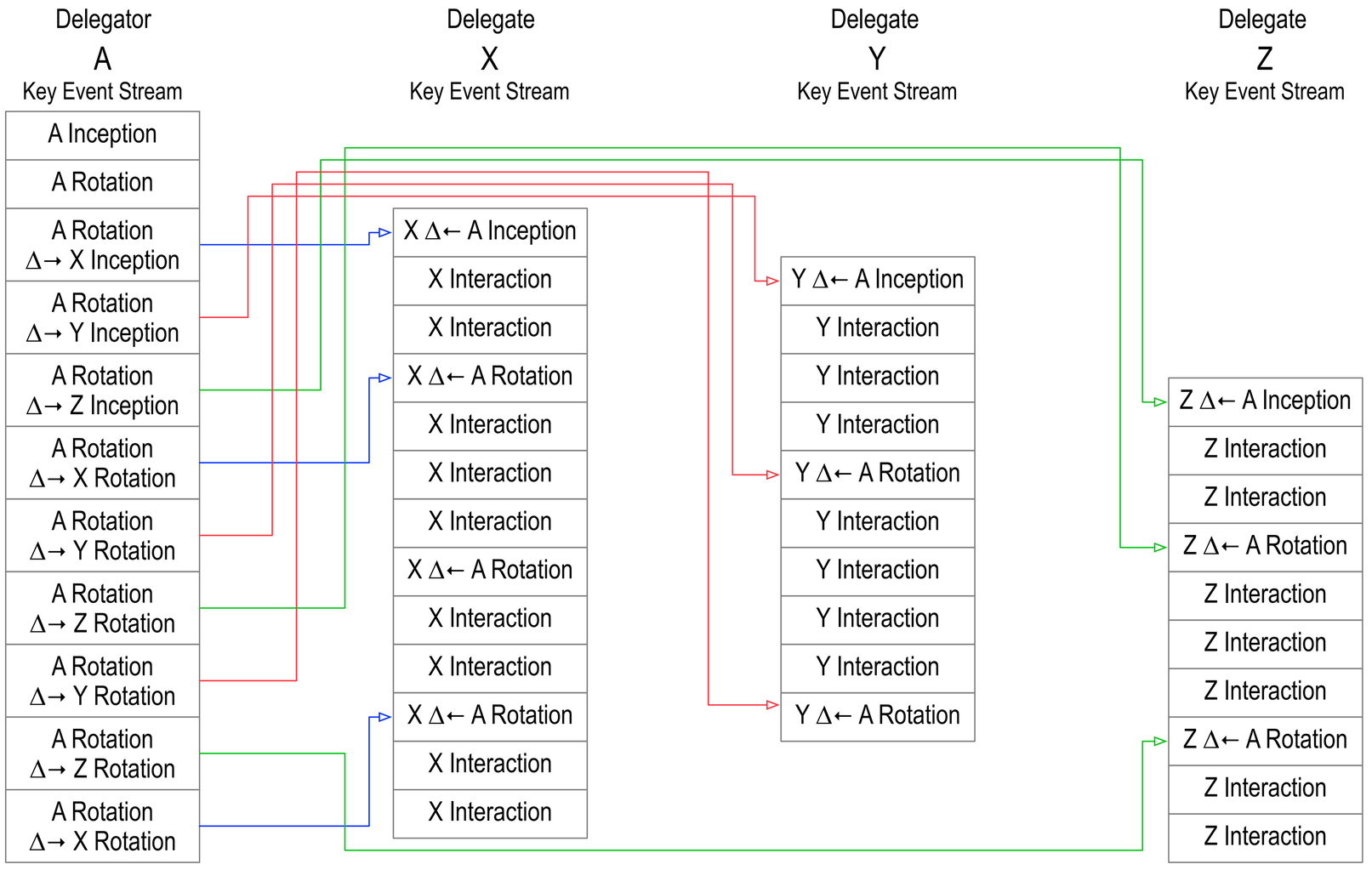


Rotation Delegation





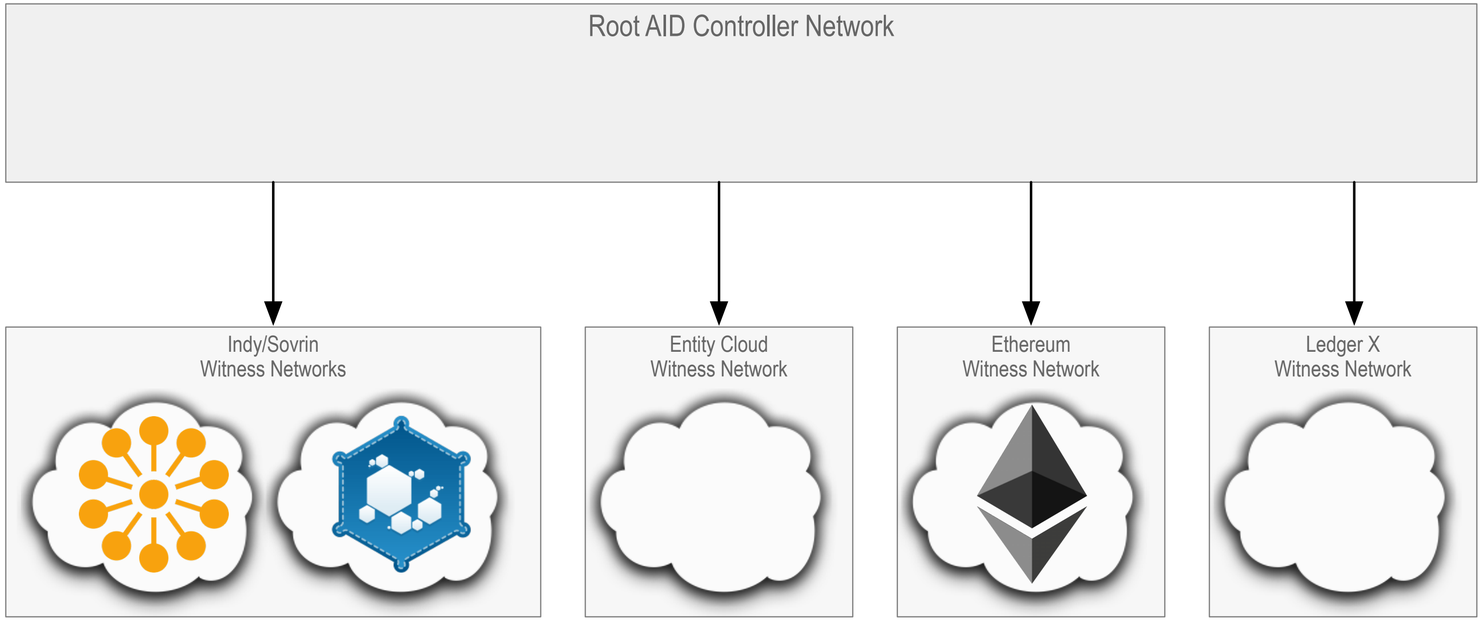
Scaling Delegation via Rotation

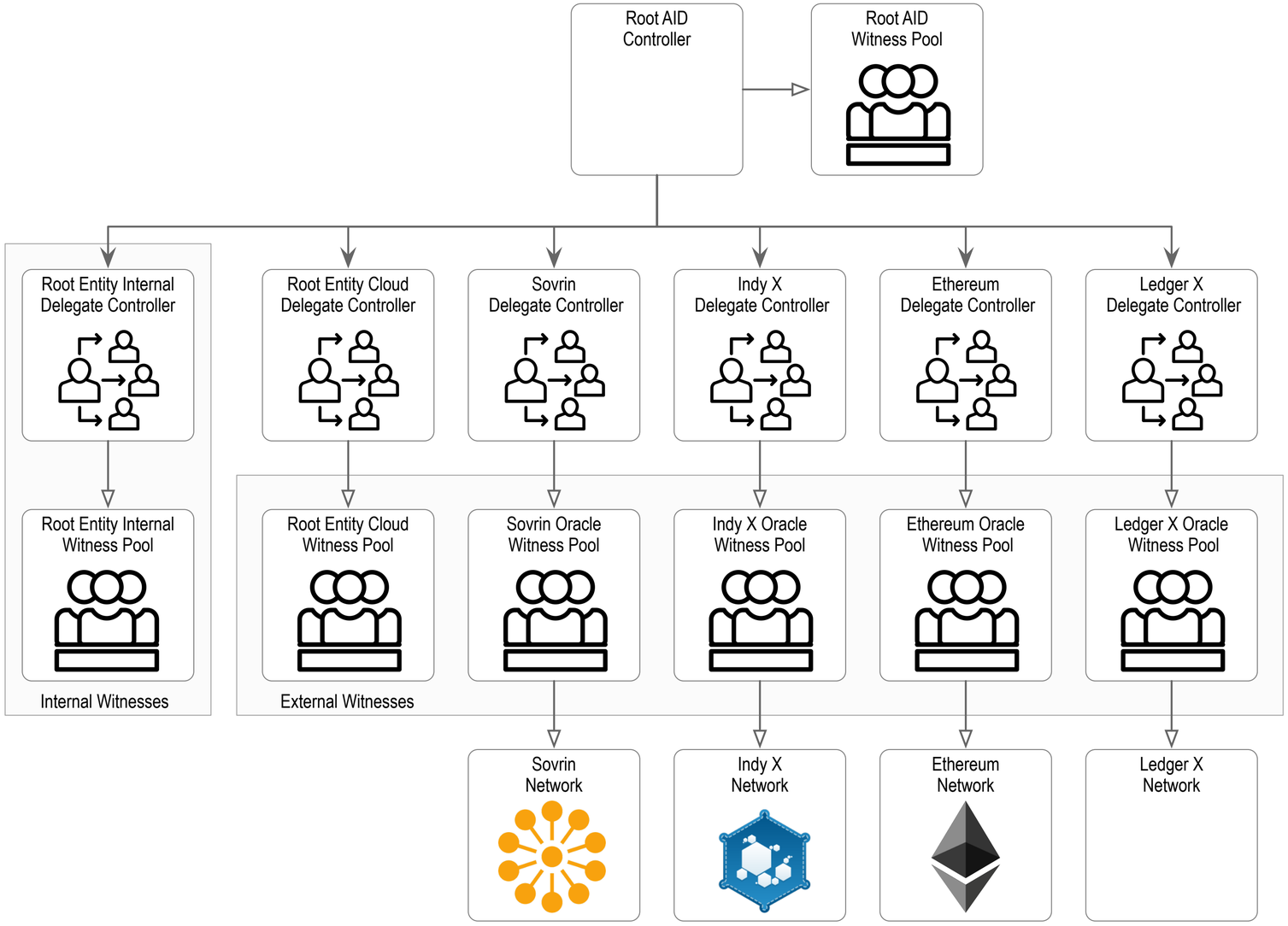


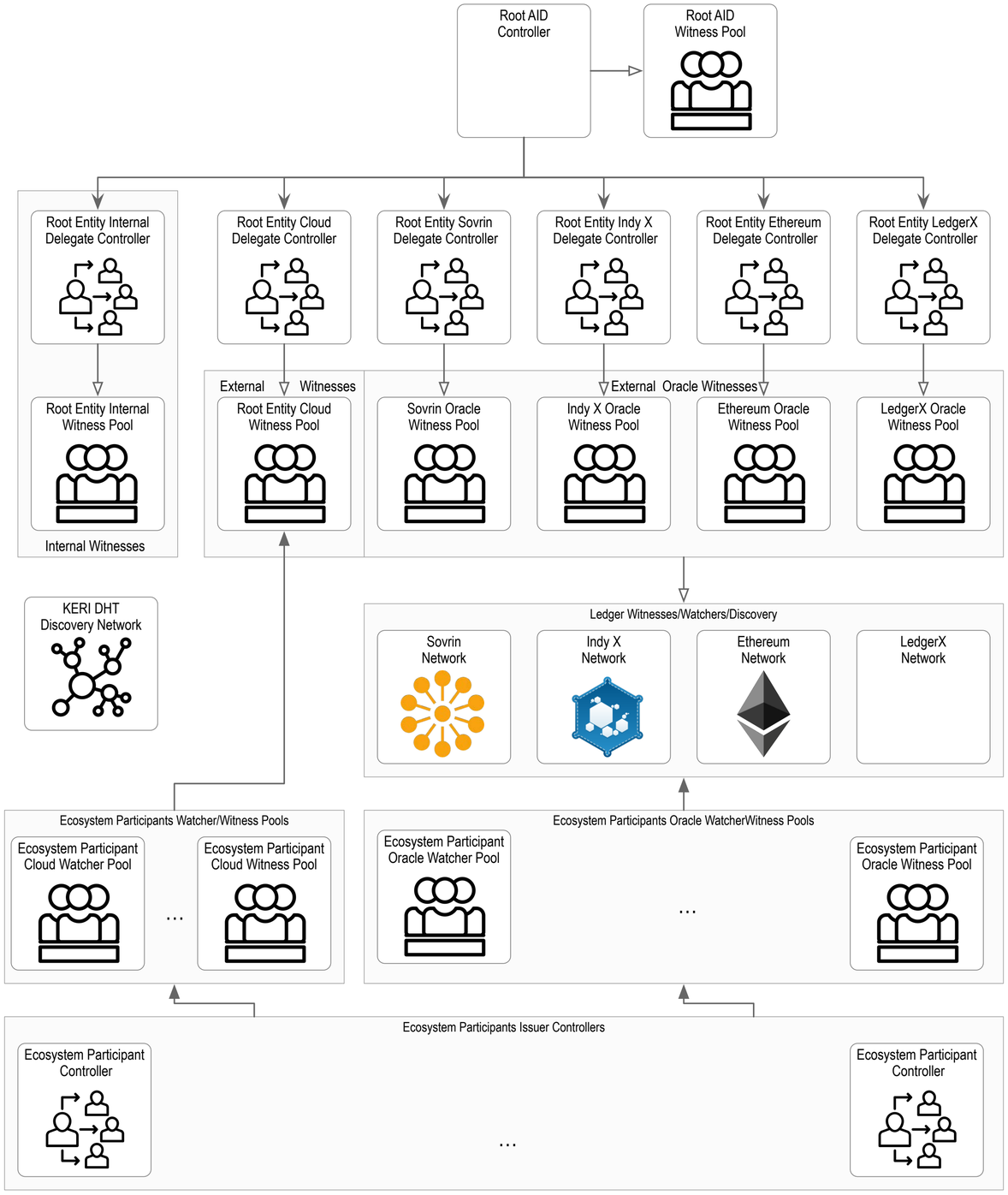
Δ→ X : Delegation to X

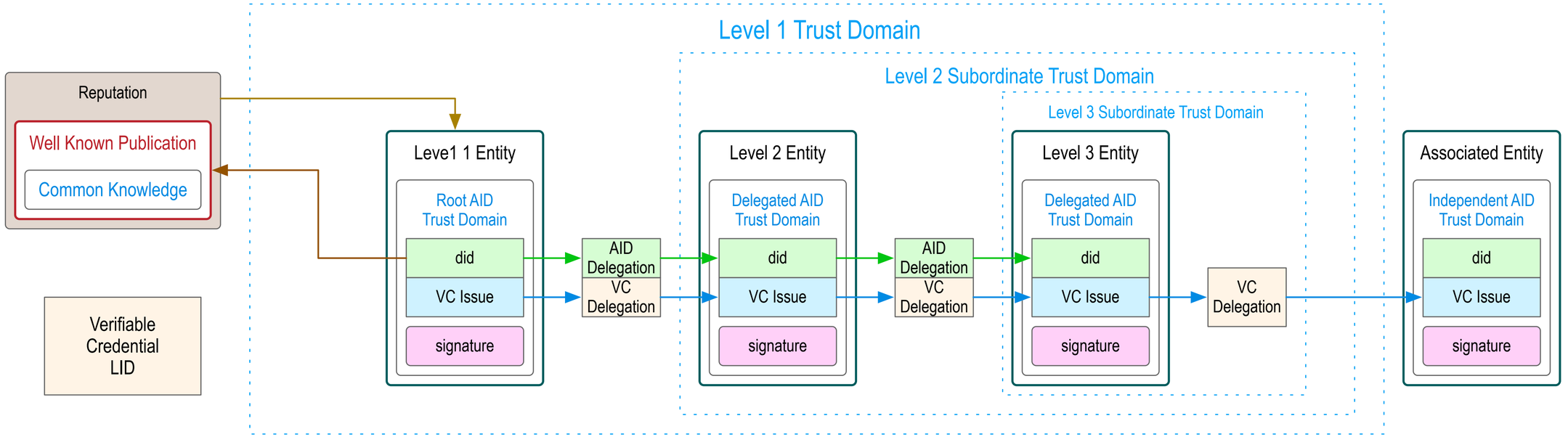
Δ← A : Delegation from A

Witness
Network-of-Networks




Root AID Controller
Root AID Witness Pool
Root Entity Internal Delegate Controller
Root Entity Cloud Delegate Controller
Sovrin Delegate Controller
Indy X Delegate Controller
Ethereum Delegate Controller
Ledger X Delegate Controller
Root Entity Internal Witness Pool
Root Entity Cloud Witness Pool
Sovrin Oracle Witness Pool
Indy X Oracle Witness Pool
Ethereum Oracle Witness Pool
Ledger X Oracle Witness Pool
Internal Witnesses
External Witnesses
Sovrin Network
Indy X Network
Ethereum Network
Ledger X Network

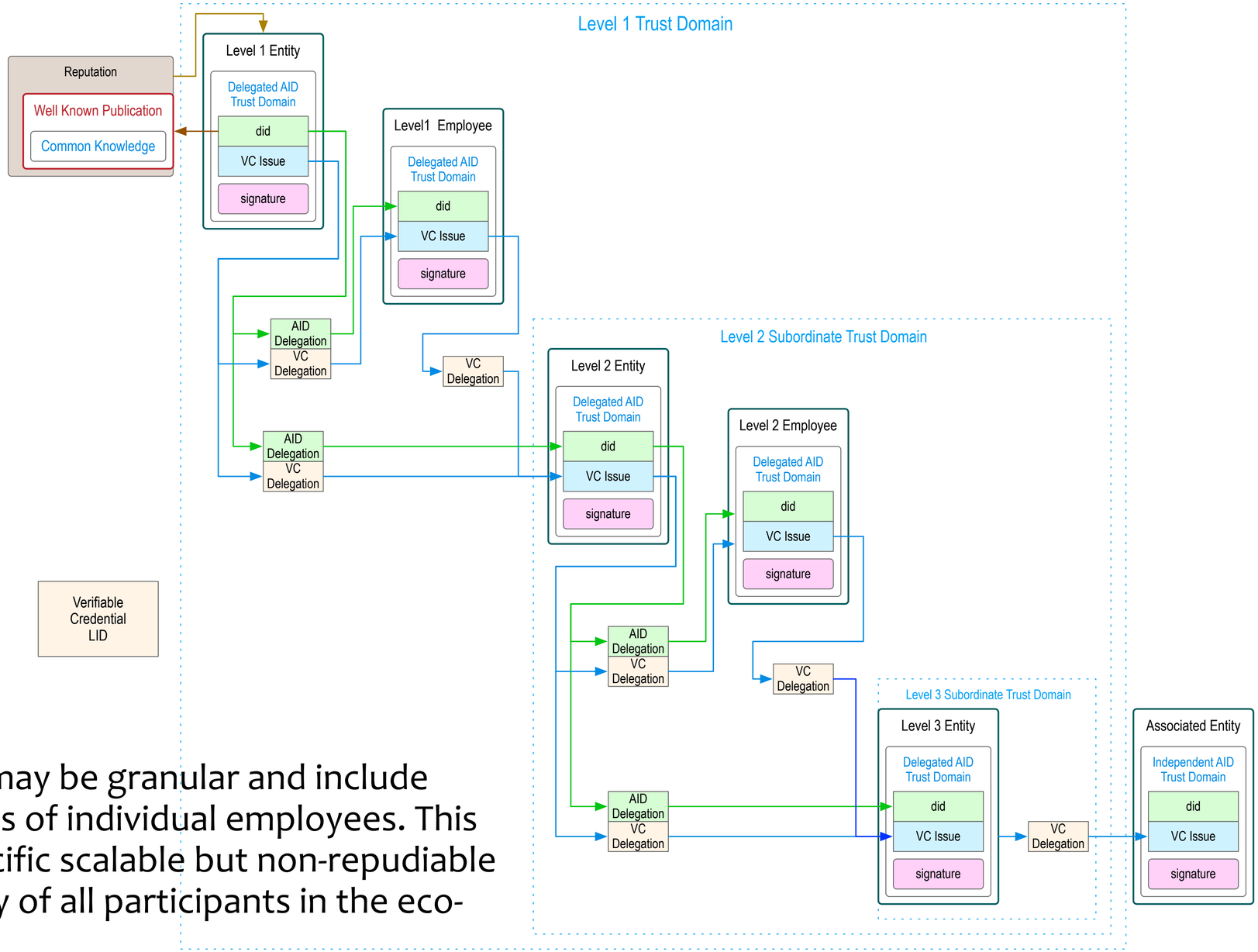

Root AID Controller
Root AID Witness Pool
Root Entity Internal Delegate Controller
Root Entity Cloud Delegate Controller
Root Entity Sovrin Delegate Controller
Root Entity Indy X Delegate Controller
Root Entity Ethereum Delegate Controller
Root Entity LedgerX Delegate Controller
Root Entity Internal Witness Pool
Internal Witnesses
External Witnesses
Root Entity Cloud Witness Pool
External Oracle Witnesses
Sovrin Oracle Witness Pool
Indy X Oracle Witness Pool
Ethereum Oracle Witness Pool
LedgerX Oracle Witness Pool
KERI DHT Discovery Network
Ledger Witnesses/Watchers/Discovery
Sovrin Network
Indy X Network
Ethereum Network
LedgerX Network
Ecosystem Participants Watcher/Witness Pools
Ecosystem Participant Cloud Watcher Pool
Ecosystem Participant Cloud Witness Pool
Ecosystem Participants Oracle WatcherWitness Pools
Ecosystem Participant Oracle Watcher Pool
Ecosystem Participant Oracle Witness Pool
Ecosystem Participants Issuer Controllers
Ecosystem Participant Controller
Ecosystem Participant Controller



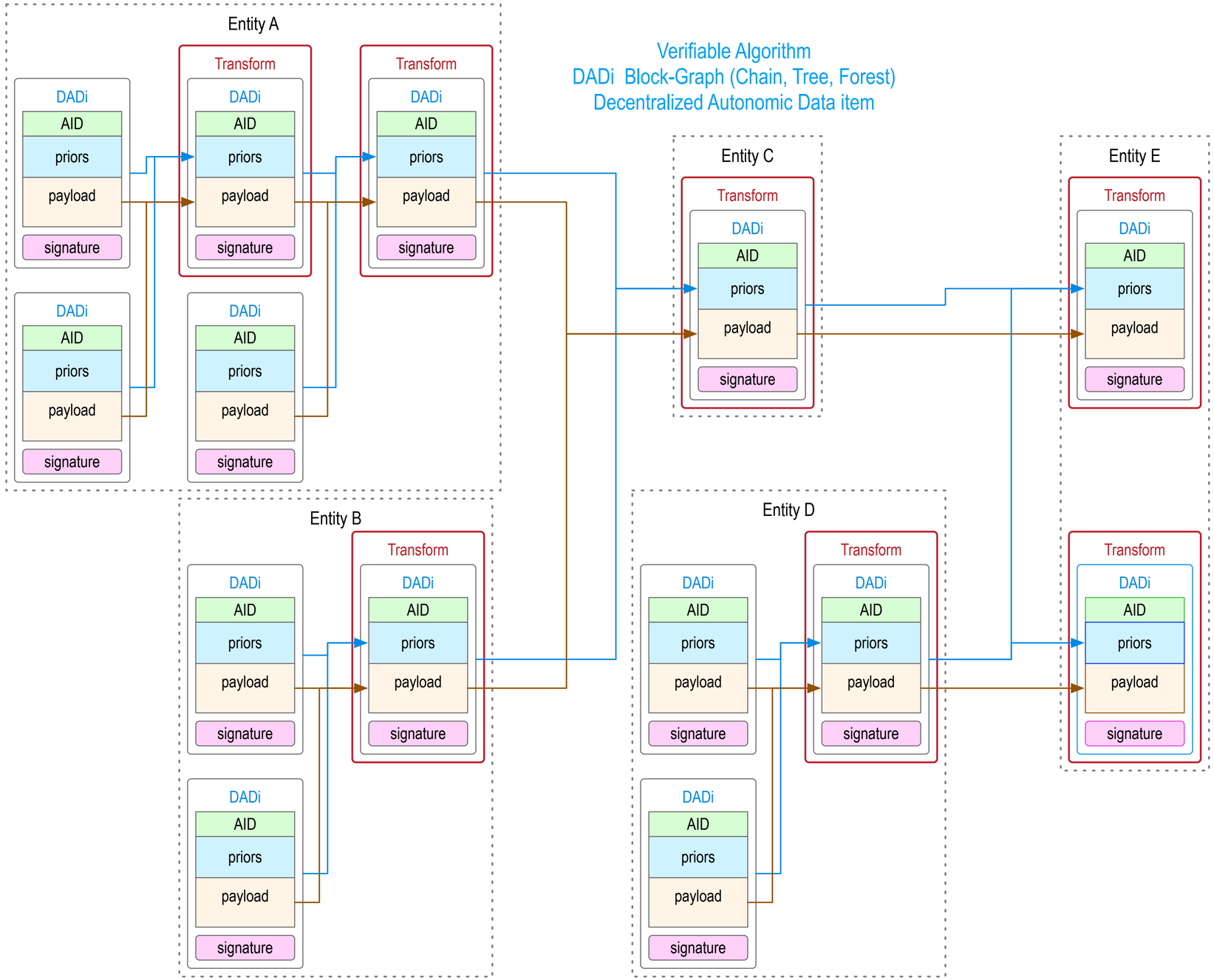
Each level of delegation forms a nested trust domain that is protected by the level above.
This increases ultimate security while enabling higher performance event issuance in lower layers.

The Level 1 entity AID provides the root-of-trust for the whole ecosystem. This enables secure decentralized interoperability.

Each trust domain may make delegations of both identifiers and verifiable credentials to a subordinate trust domain. These delegations provide revocable authorizations.



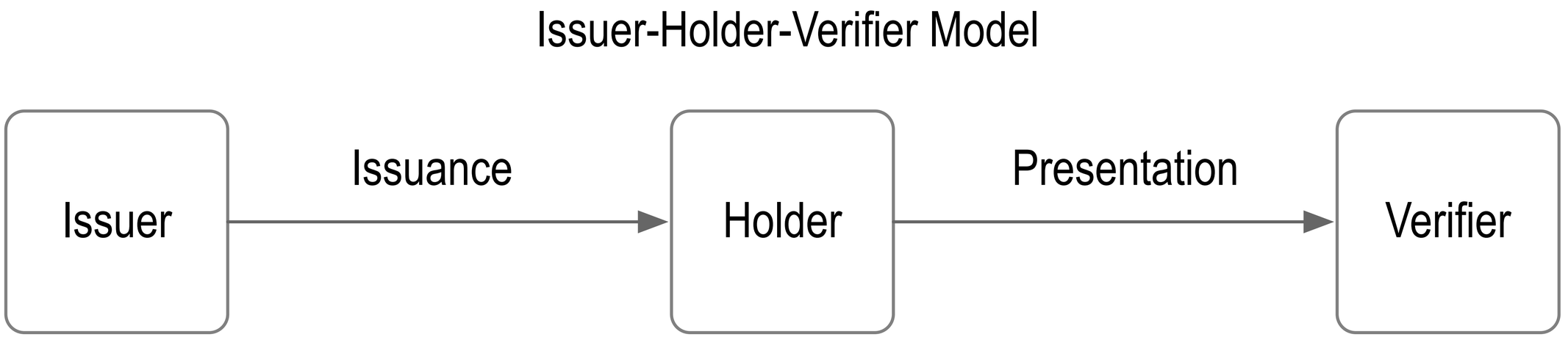
Delegations may be granular and include authorizations of individual employees. This provides specific scalable but non-repudiable accountability of all participants in the eco-system.


Verifiable Algorithm
DADi Block-Graph (Chain, Tree, Forest)
Decentralized Autonomic Data item
Entity A
Entity B
Entity C
Entity D
Entity E
Transform
DADi
AID
priors
payload
signature

Tripartite Authentic Data (VC) Model

Issuer: Source of the VC. Creates (issues) and signs VC
Holder: Usually the target of the VC. The holder is the "*issuee*" that receives the VC and holds it for its own use.
Verifier: Verifies the signatures on the VC and authenticates the holder at the time of presentation

The issuer and target each have a DID (decentralized identifier).
The DIDs are used to look-up the public key(s) needed to verify signatures.

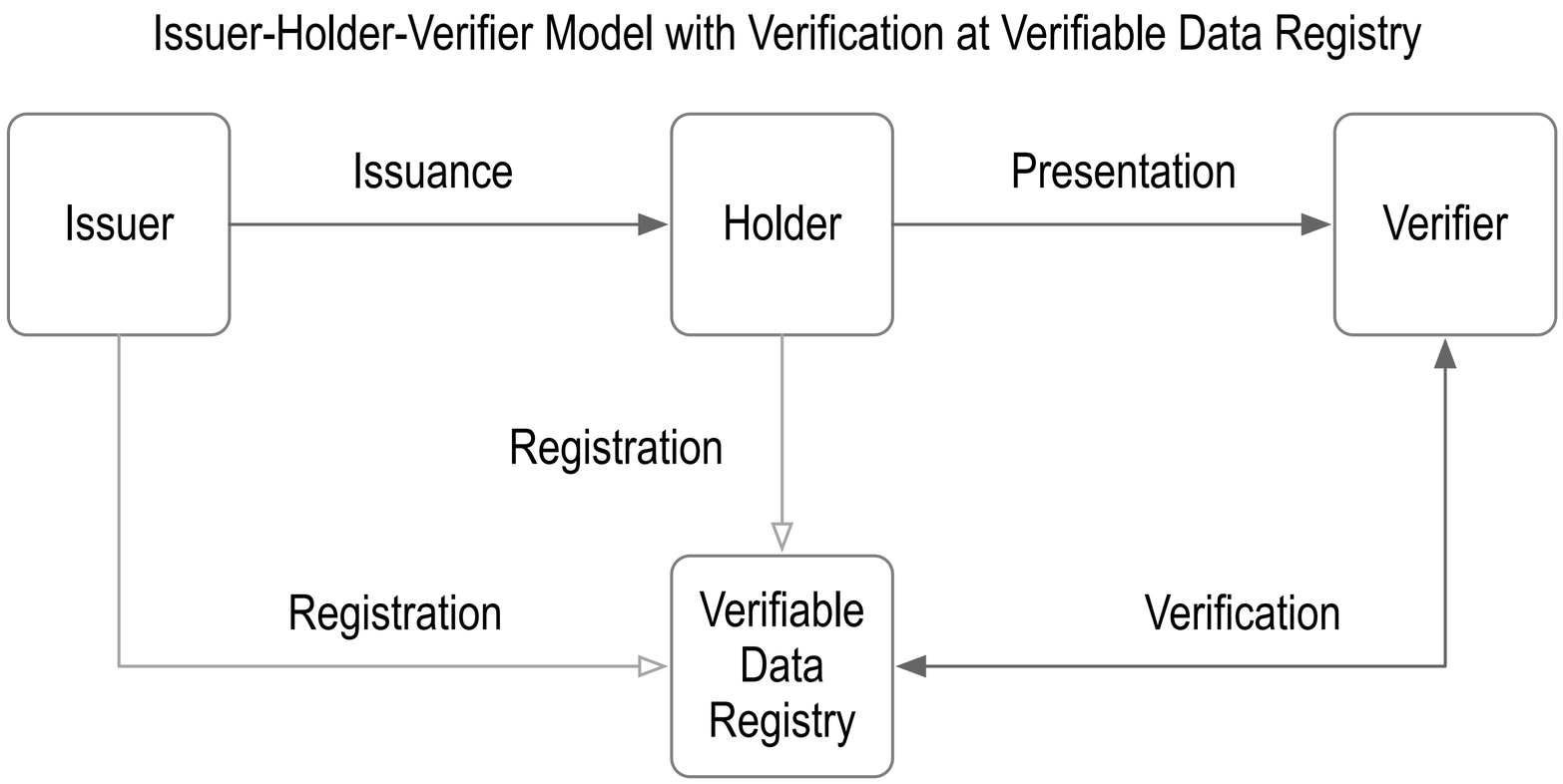


Tripartite Authentic Data (VC) Model with VDR

Verifiable Data Registry (VDR) enables decentralized but interoperable discovery and verification of authoritative key pairs for DIDs in order to verify the signatures on VCs. A VDR may also provide other information such as data schema or revocation state of a VC.

Each controller of a DID registers that DID on a VDR so that a verifier can determine the authoritative key pairs for any signatures.

We call this determination, *establishment of control authority* over a DID.

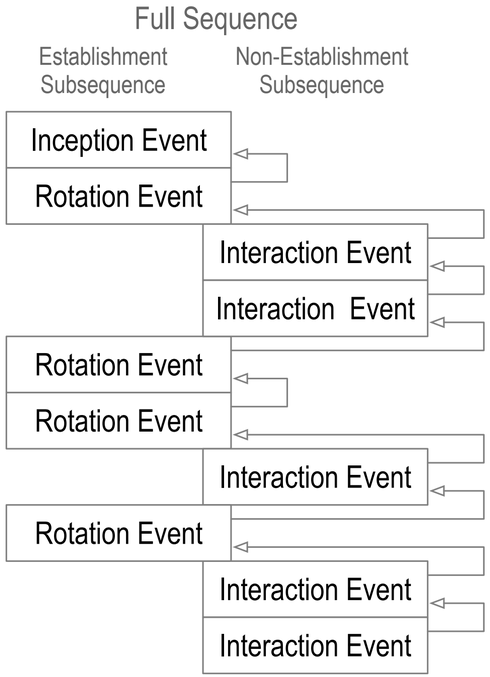


KERI VDRs vs. Shared Ledger VDRs

Most DID methods use a shared ledger (commonly referred to as a *blockchain*) for their VDR. Typically, in order to interoperate all participants must use the same shared ledger or support multiple different DID methods. There are currently over 70 DID methods. Instead GLEIF has chosen to use KERI based DID methods. KERI stands for Key Event Receipt Infrastructure. KERI based VDRs are ledger independent, i.e. not locked to a given ledger. This provides a path for greater interoperability without forcing participants in the vLEI ecosystem to use the same shared ledger.

A KERI VDR is called a key event log (KEL). It is a cryptographically verifiable signed hash chained data structure, a special class of verifiable data structure. Each KERI based identifier has its own dedicated KEL. The purpose of the KEL is to provide proof of the establishment of control authority over an identifier. This provides cryptographically verifiable proof of the current set of authoritative keys for the identifier. KERI identifiers are long cryptographic pseudo random strings of characters. They are self-certifying and self-managing.

A KERI identifier is abstractly called an Autonomic Identifier (AID) because it is self-certifying and self-managing. A KERI DID is one concrete implementation of a KERI AID. The same KERI prefix may control multiple different DIDs as long as they share the same prefix.



`did:keri:`*`prefix`*`[:`*`options`*`][/`*`path`*`][?`*`query`*`][#`*`fragment`*`]`

`did:keri:ENqFtH6_cfDg8riLZ-GDvDaCKVn6clOJa7ZXXVXSWpRY`

KERI Identifier KEL VDR *Controls* Verifiable Credential Registry TEL VDR

A KERI KEL for a given identifier provides proof of authoritative key state at each event. The events are ordered. This ordering may be used to order transactions on some other VDR such as a Verifiable Credential Registry by attaching anchoring seals to KEL events.

Seals include cryptographic digest of external transaction data.

A seal binds the key-state of the anchoring event to the transaction event data anchored by the seal.

The set of transaction events that determine the external registry state form a log called a Transaction Event Log (TEL).

Transactions are signed with the authoritative keys determined by the key state in the KEL with the transaction seal.

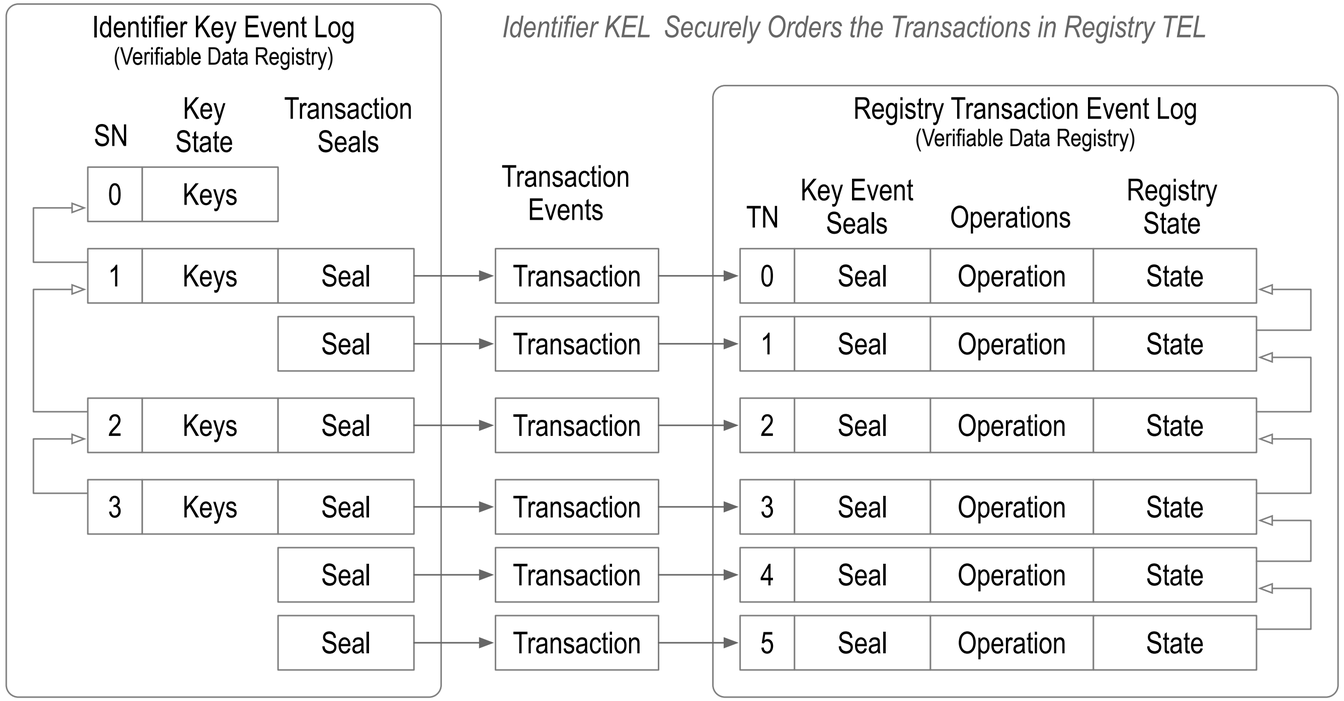
The transactions likewise contain a reference seal back to the key event authorizing the transaction.

This setup enables a KEL to control a TEL for any purpose. This includes what are commonly called “smart contracts”.

The TEL provides a cryptographic proof of registry state by reference to the corresponding controlling KEL.

Any validator may therefore cryptographically verify the authoritative state of the registry.

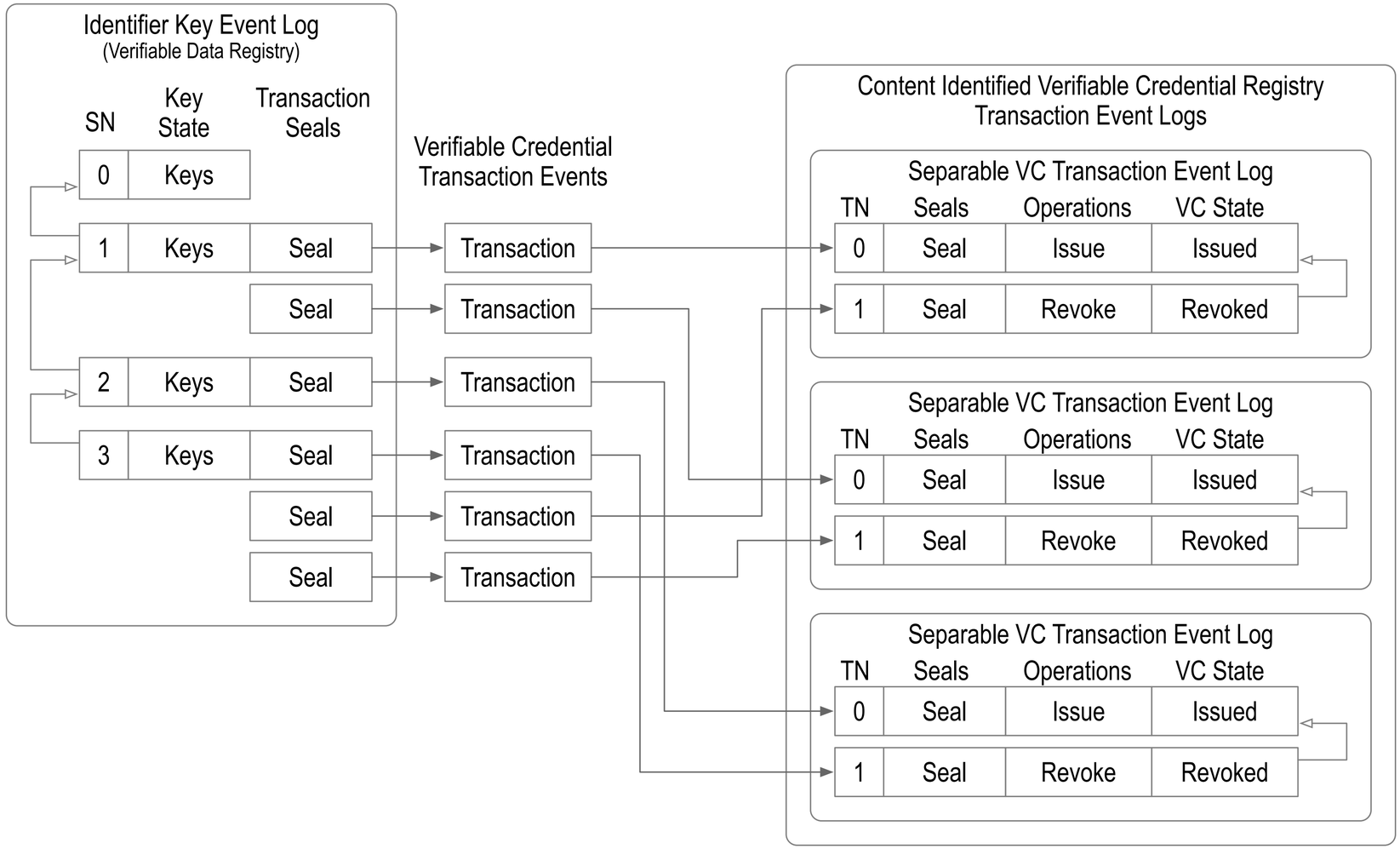
In the case of the vLEI the associated TEL controls a vLEI issuance and revocation registry.



seal = proof of authenticity

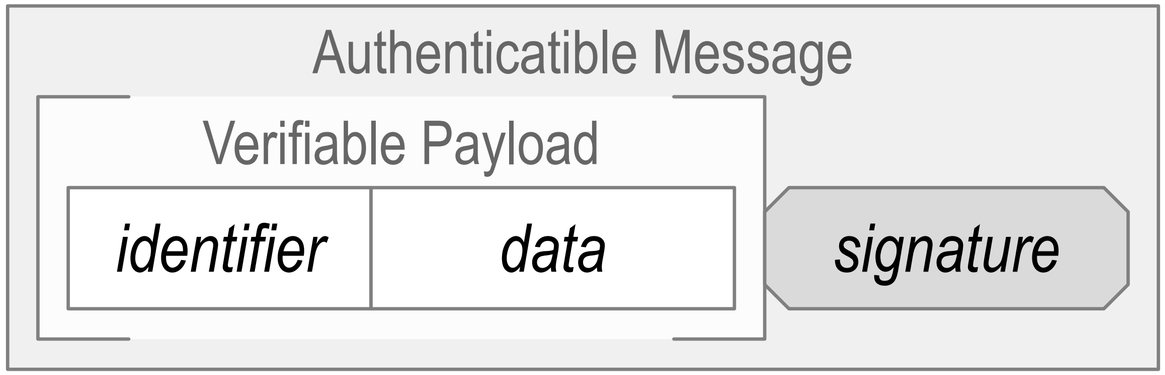
Registry with Separable VC Issuance-Revocation TELs

Each VC also has a uniquely identified issuer using a KERI AID.
Each VC may be uniquely identified with a content digest.
A full identifier for the VC may include its content digest but also be in the namespace of its issuer.
These may be used as database keys to lookup a VC and verify the content of a given VC.
This combination enables a separable registry of VC issuance-revocation state.
The state may employ a cryptographic accumulator for enhanced privacy

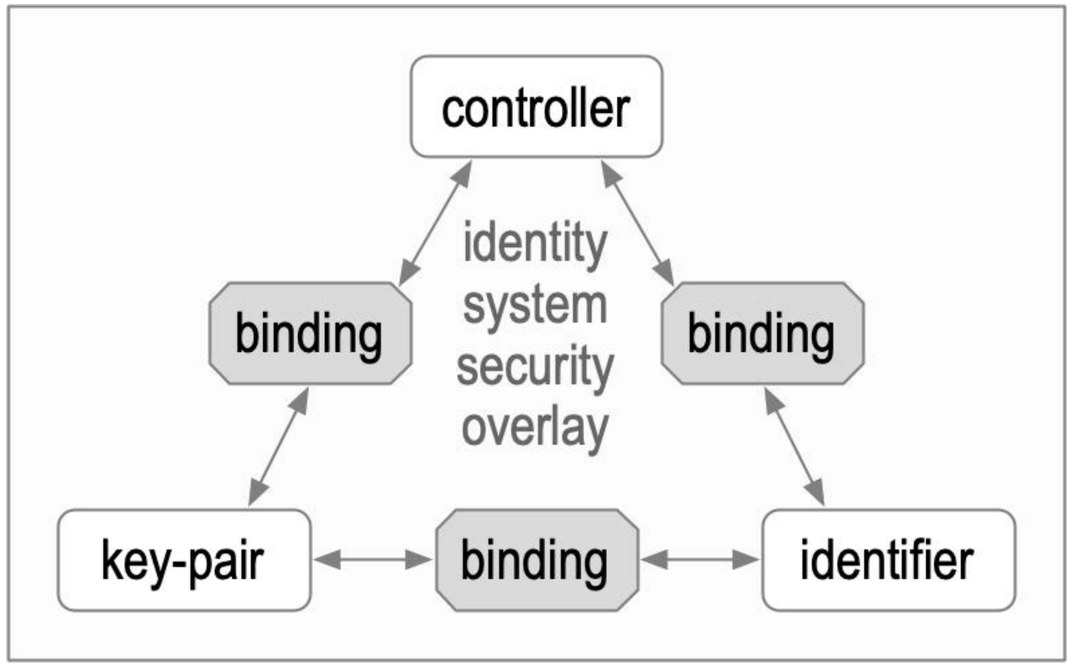


Identity System Security Overlay

Establish authenticity of IP packet's message payload.



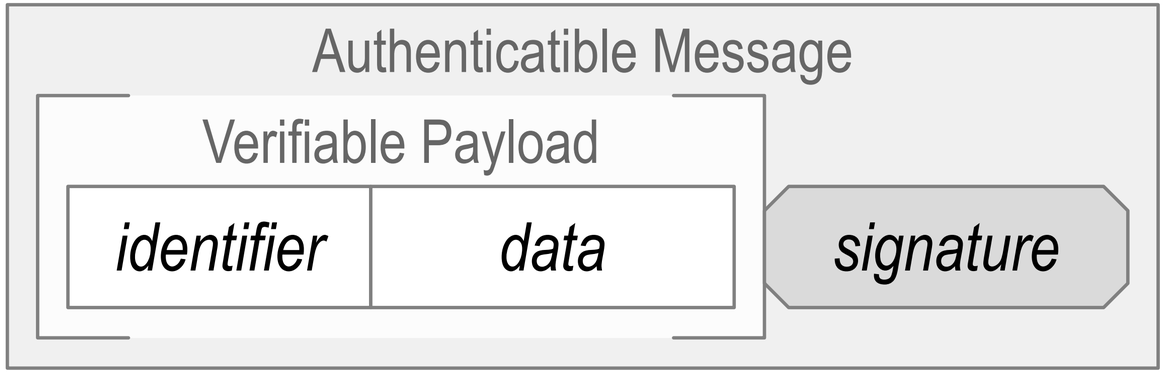
The overlay's security is contingent on the mapping's security.

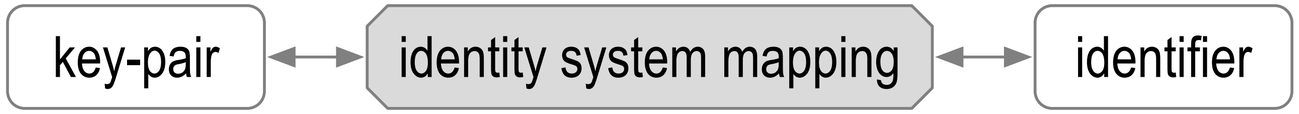


Identifier Issuance

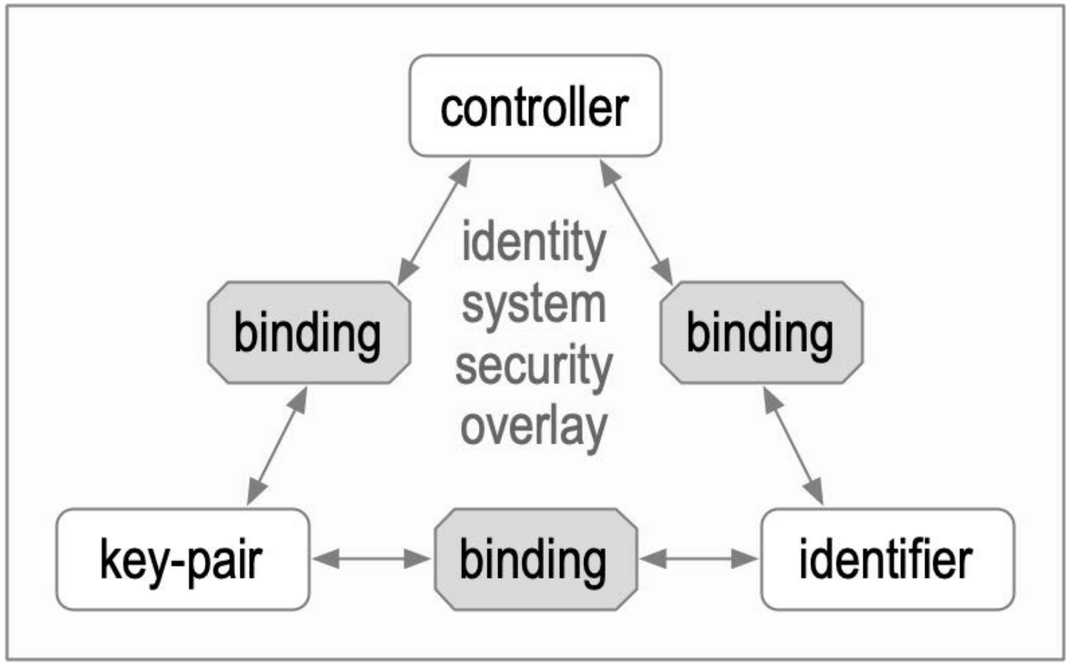
Identifier System Security

Authentic transmission of data may be verified using an identity system security overlay.
This overlay maps cryptographic key-pairs to identifiers.
When those identifiers are self-certifying they are derived via cryptographic one-way functions from the key pairs.
This provides a self-certifying identifier with a cryptographic root-of-trust.
A key event log (KEL) provide support for secure key rotation without changing the identifier.
Message authenticity is provided by verifying signatures to the authoritative keys pairs for the identifier included in the message.



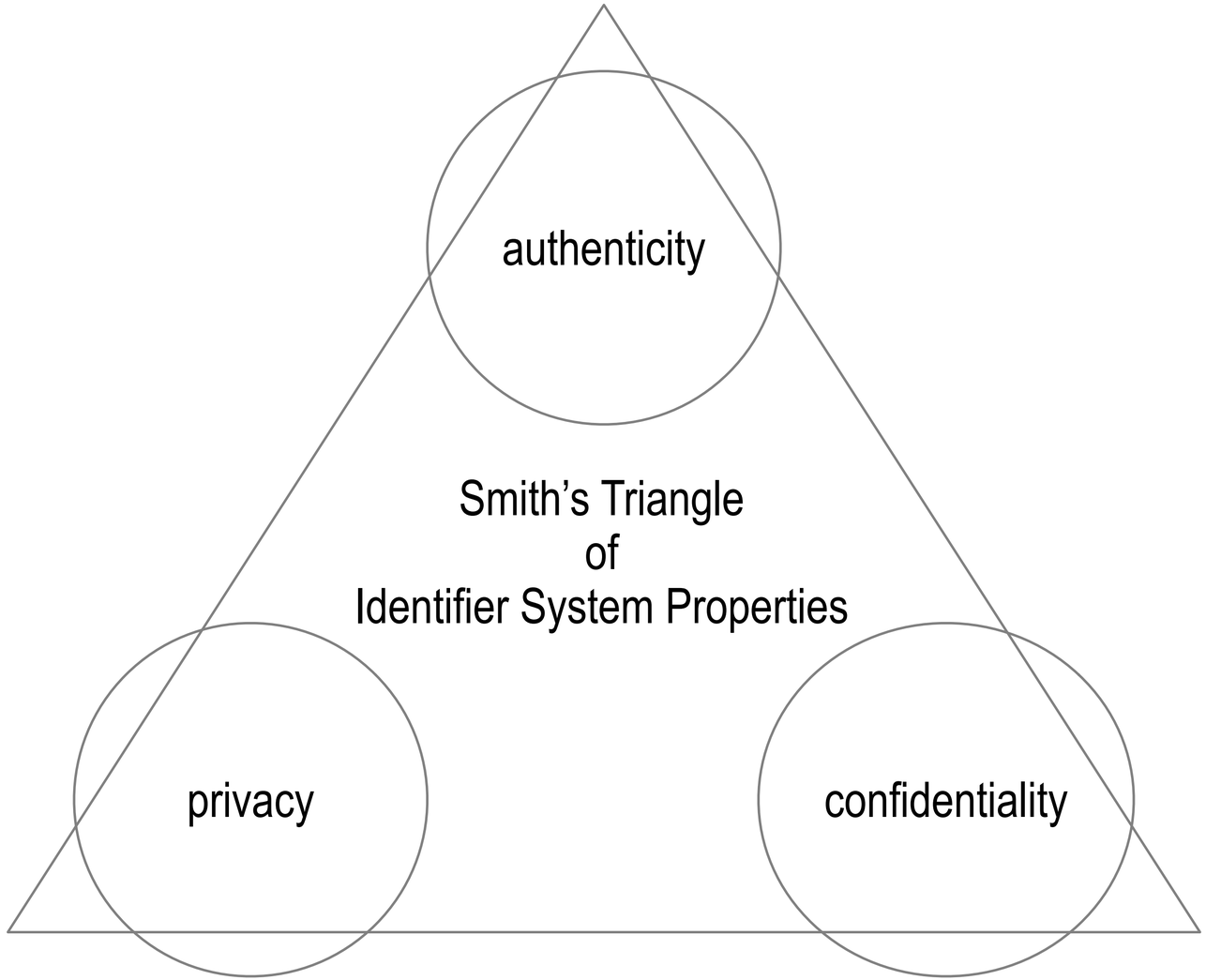


The overlay's security is contingent on the mapping's security.



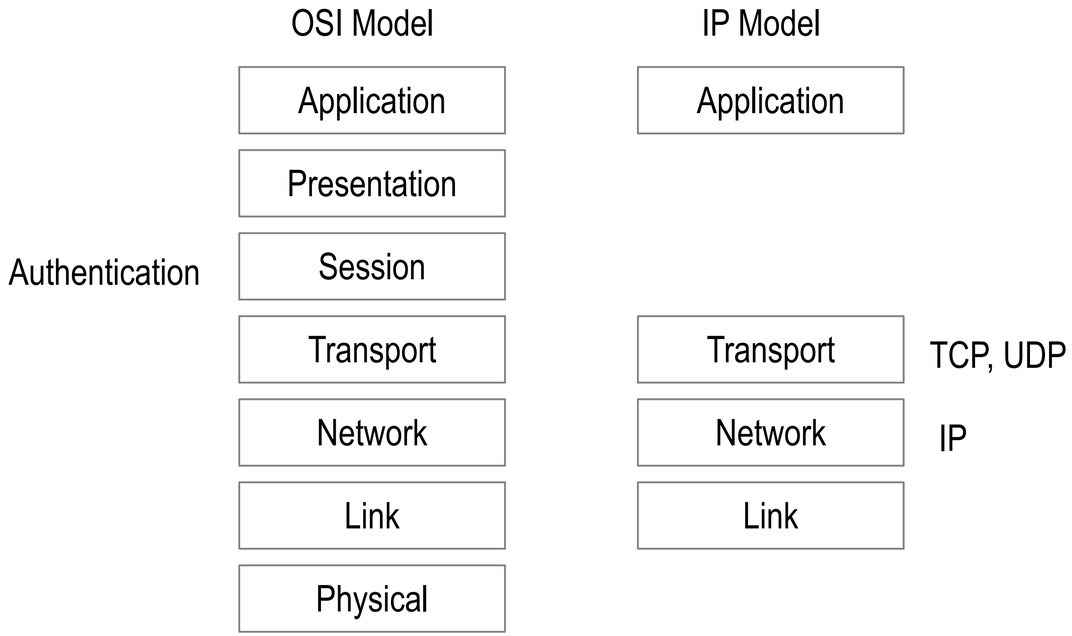
Identifier Issuance

Smith’s Identifier System Properties Triangle



May exhibit any two at the highest level but not all three at the highest level

The Internet Protocol (IP) is *bro-ken* because it has no *security* layer.

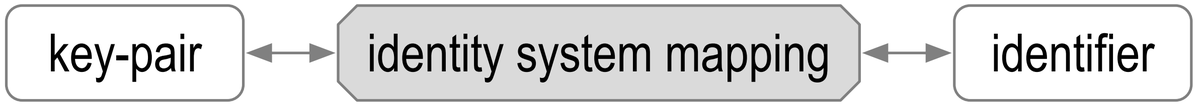


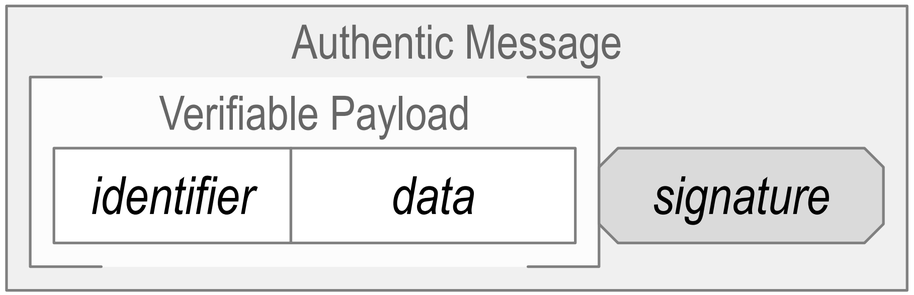
Instead …

We use *bolt-on* identity system security overlays. (DNS-CA …)

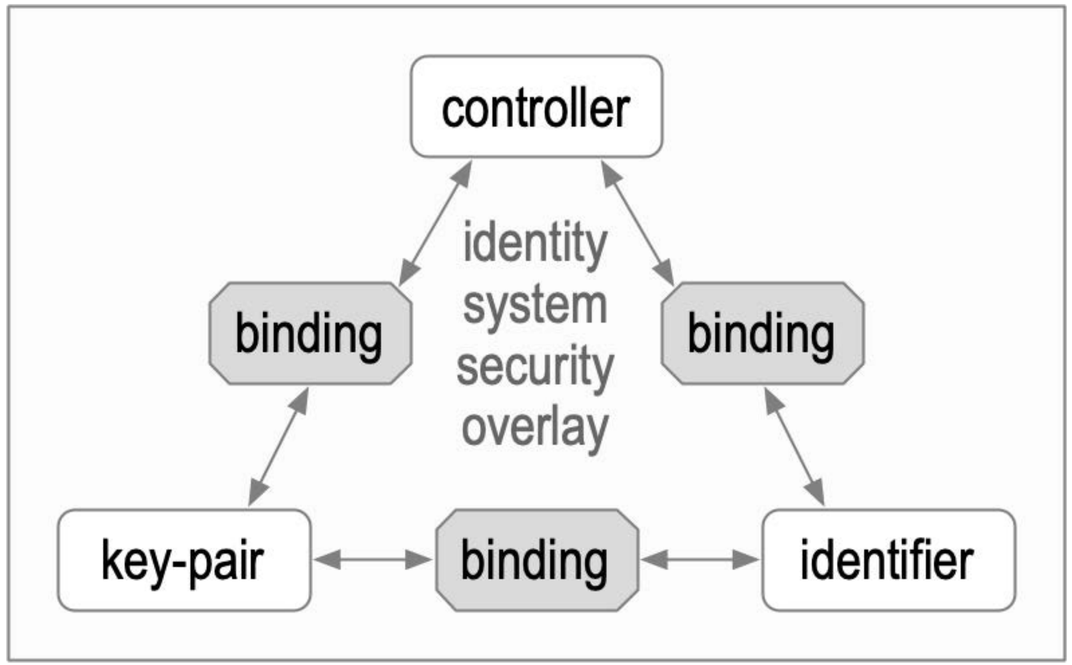
Identity System Security Overlay

Establish authenticity of IP packet's message payload.



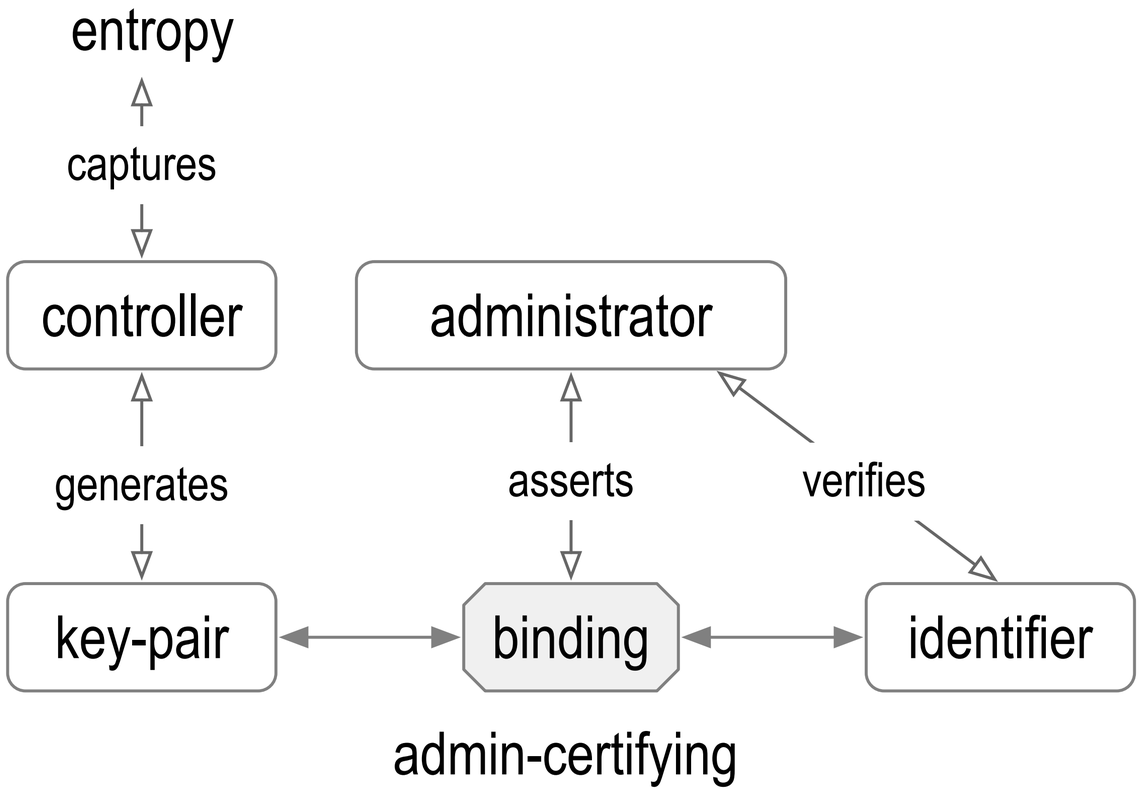


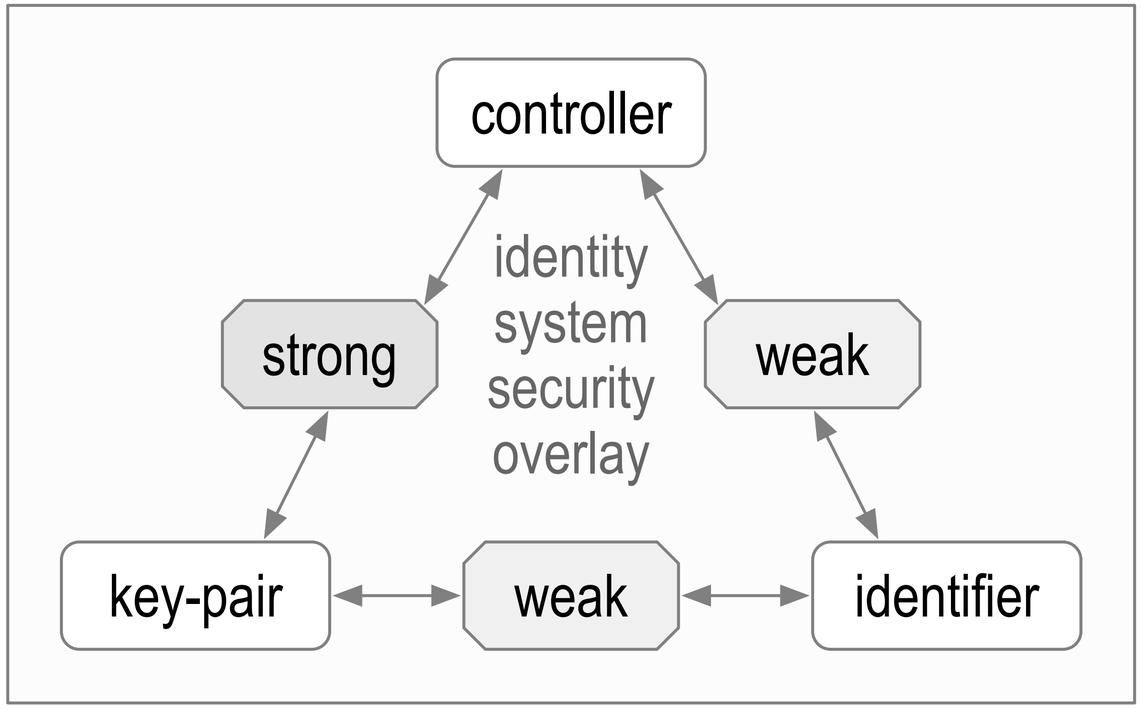
The overlay's security is contingent on the mapping's security.



Identifier Issuance

Administrative Identifier Issuance and Binding



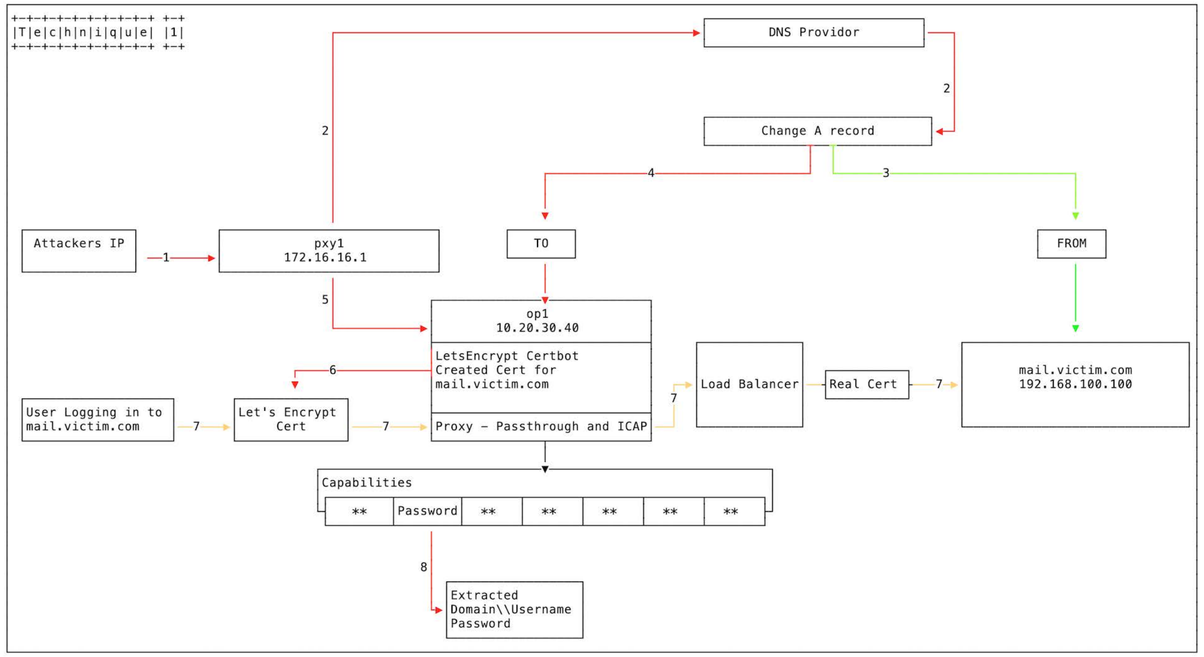


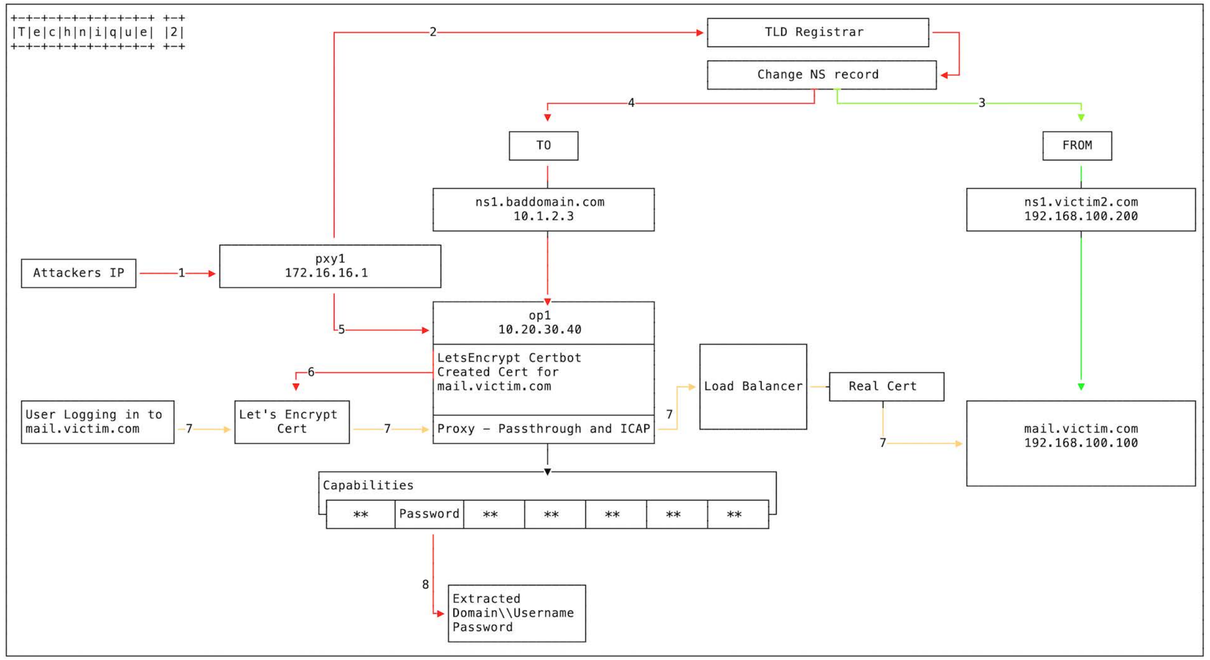
Admin-Certifying Identifier Issuance

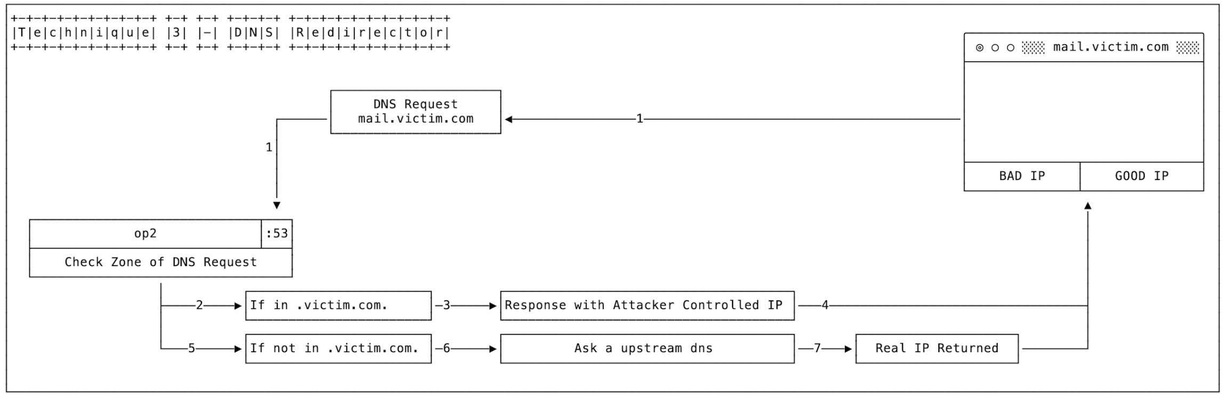
DNS Hijacking

A DNS hijacking wave is targeting companies at an almost unprecedented scale. Clever trick allows attackers to obtain valid TLS certificate for hijacked domains.

https://arstechnica.com/information-technology/2019/01/a-dns-hijacking-wave-is-targeting-companies-at-an-almost-unprecedented-scale/







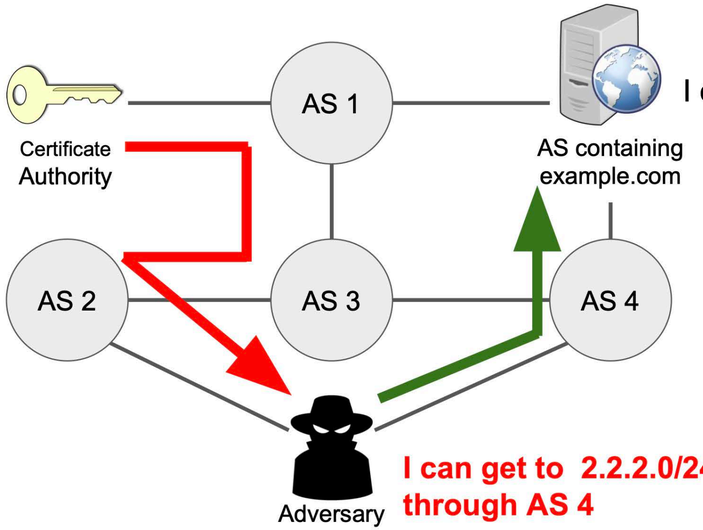
BGP Hijacking: AS Path Poisoning

Spoof domain verification process from CA. Allows attackers to obtain valid TLS certificate for hijacked domains.

Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J. and Mittal, P., "Bamboozling certificate authorities with {BGP}," vol. 27th {USENIX} Security Symposium, no. {USENIX} Security 18, pp. 833-849, 2018 https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee

Gavrichenkov, A., "Breaking HTTPS with BGP Hijacking," BlackHat, 2015 https://www.blackhat.com/docs/us-15/materials/us-15-Gavrichenkov-Breaking-HTTPS-With-BGP-Hijacking-wp.pdf

AS path poisoning

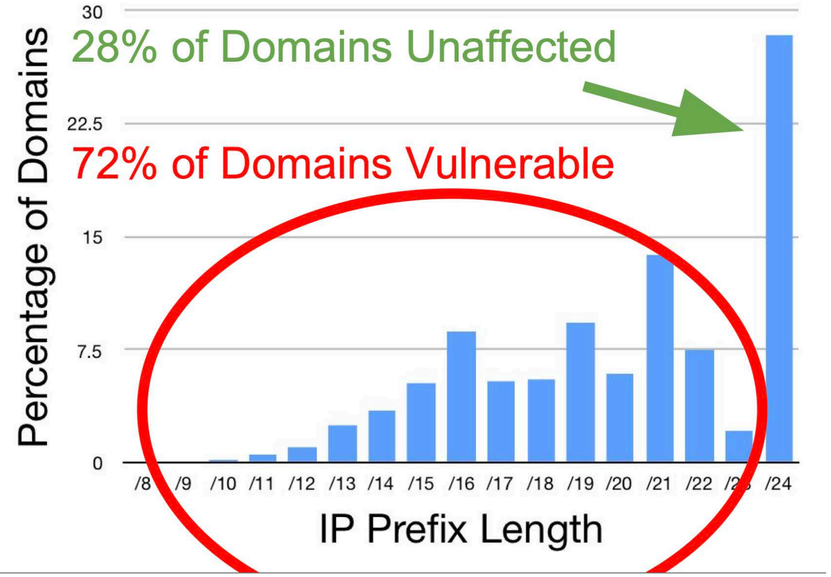


I own 2.2.2.0/23

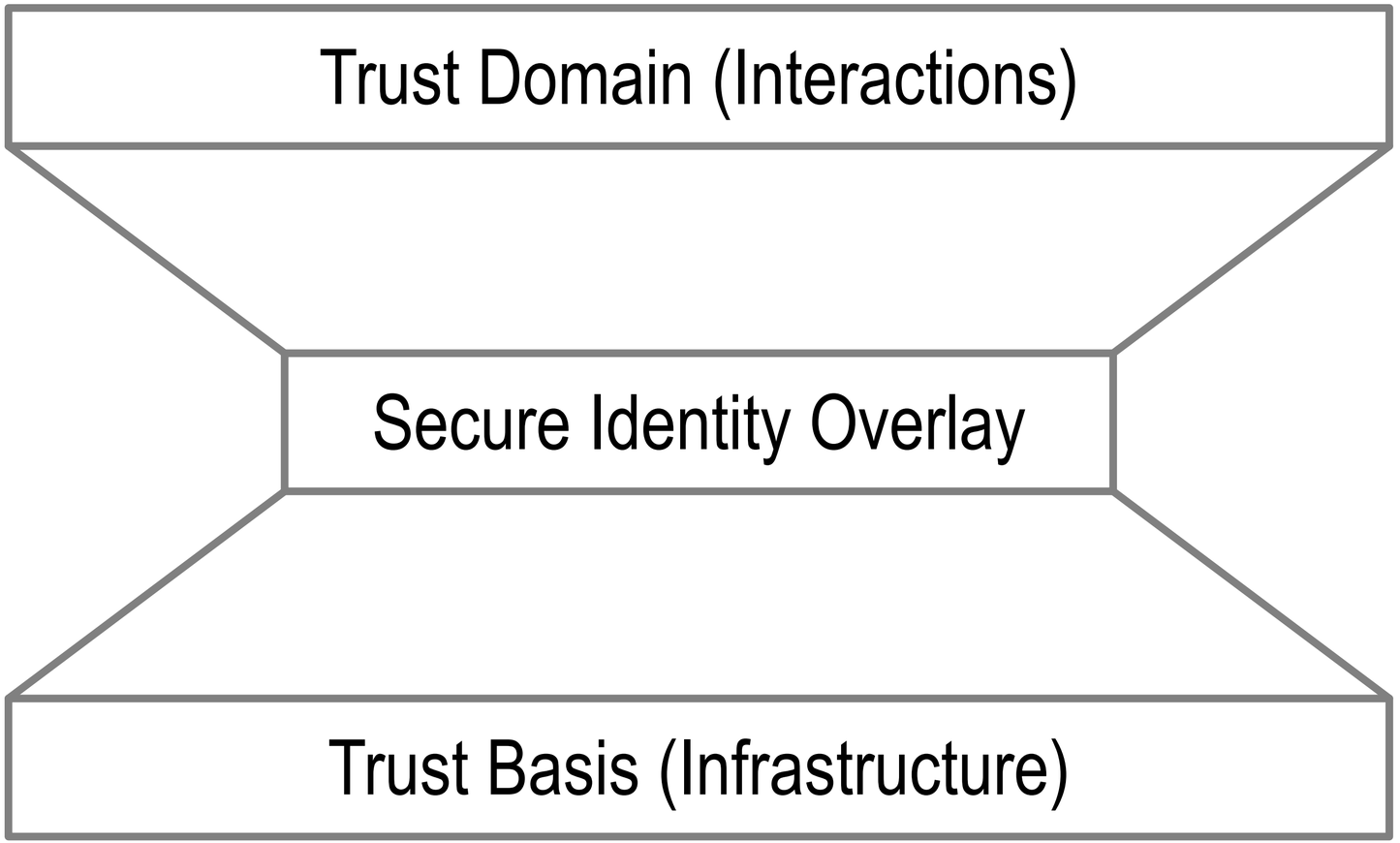
- Everyone sees announcement but looks less suspicious
- Connectivity preserved
- Almost any AS can perform
- Very stealthy
- Perfect setup to intercept traffic with certificate

Vulnerability of domains: sub-prefix attacks

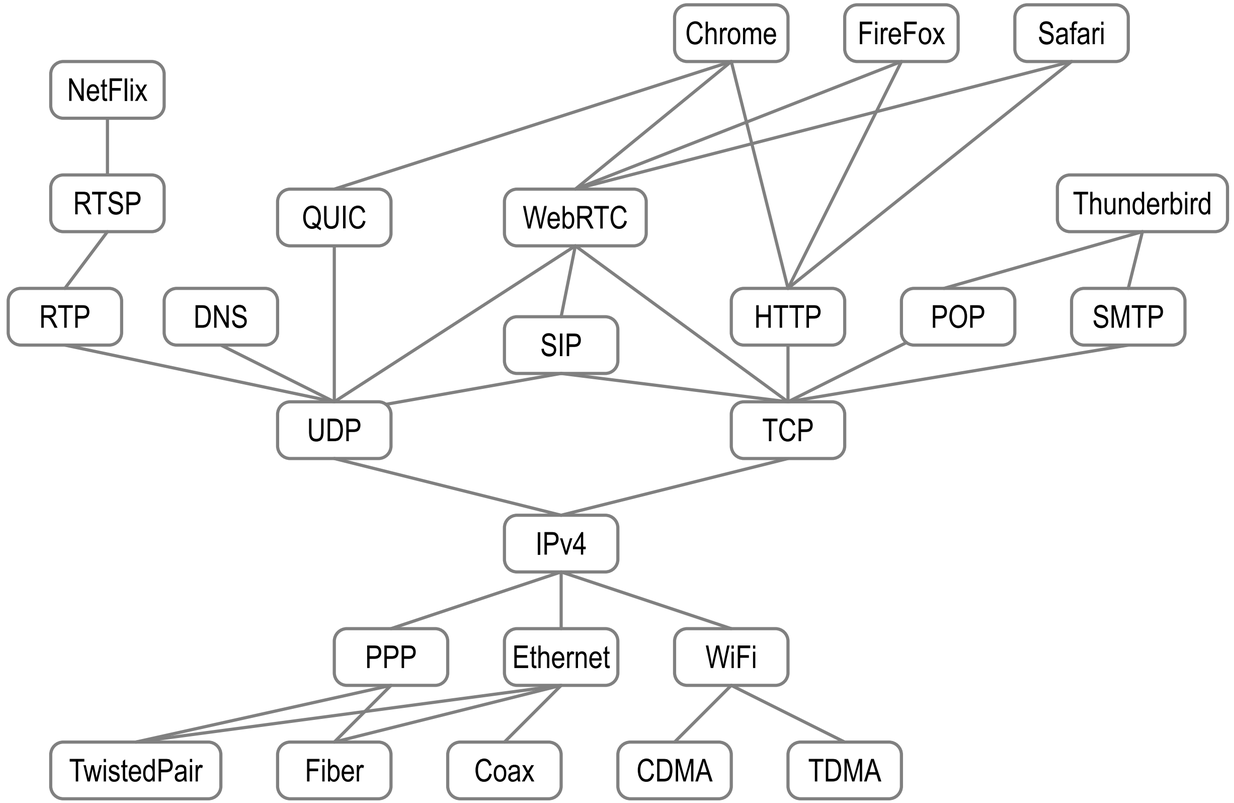
- Any AS can launch
- Only prefix lengths less than /24 vulnerable (filtering)

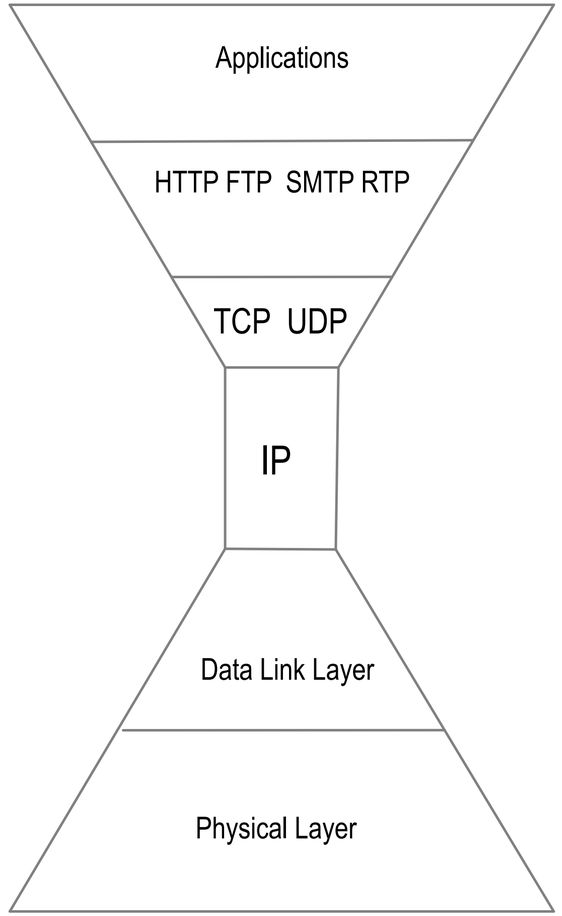


Identity System Security Overlay

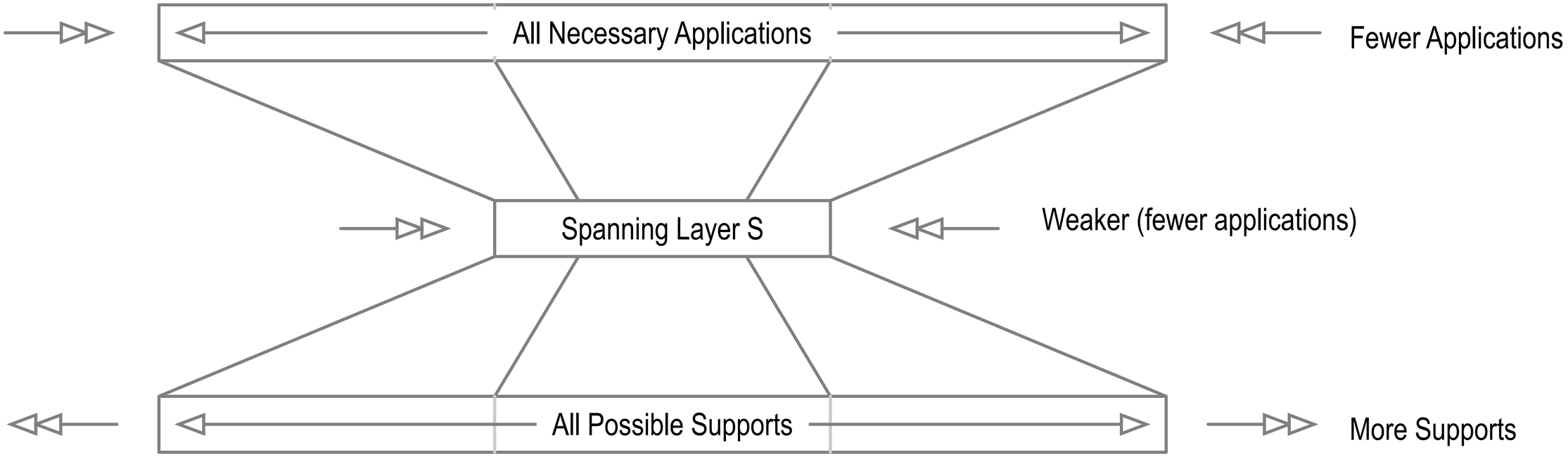


Spanning Layer

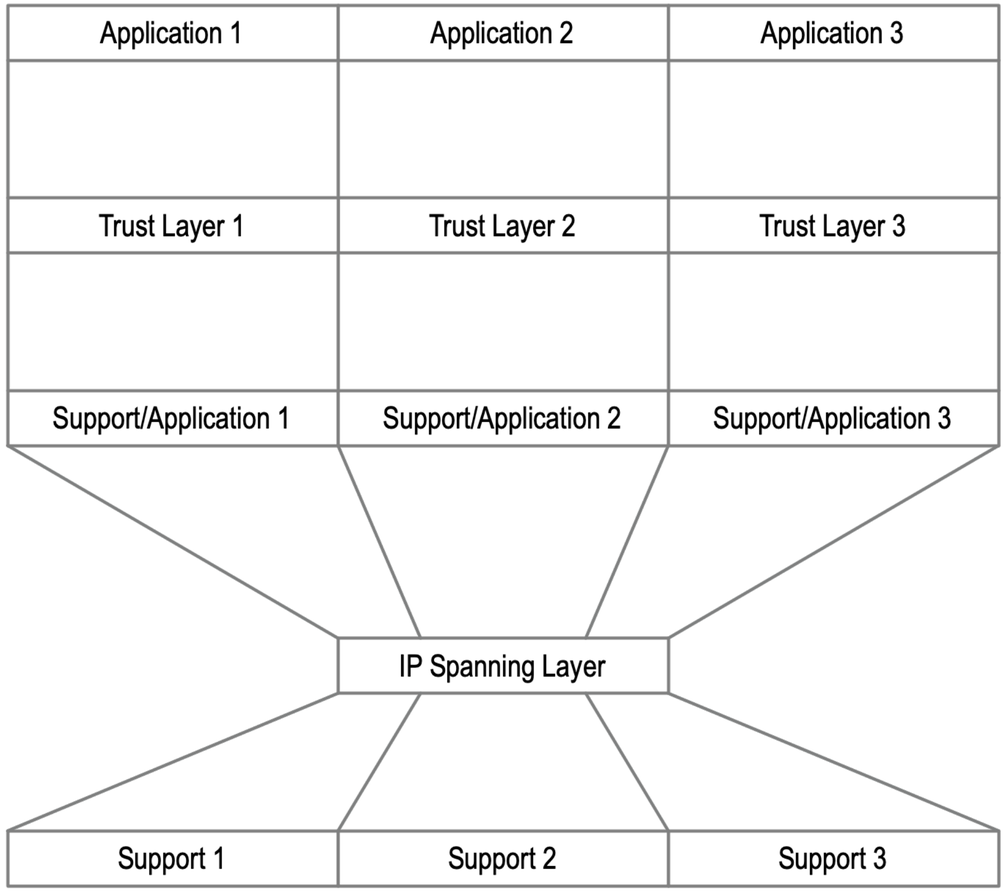




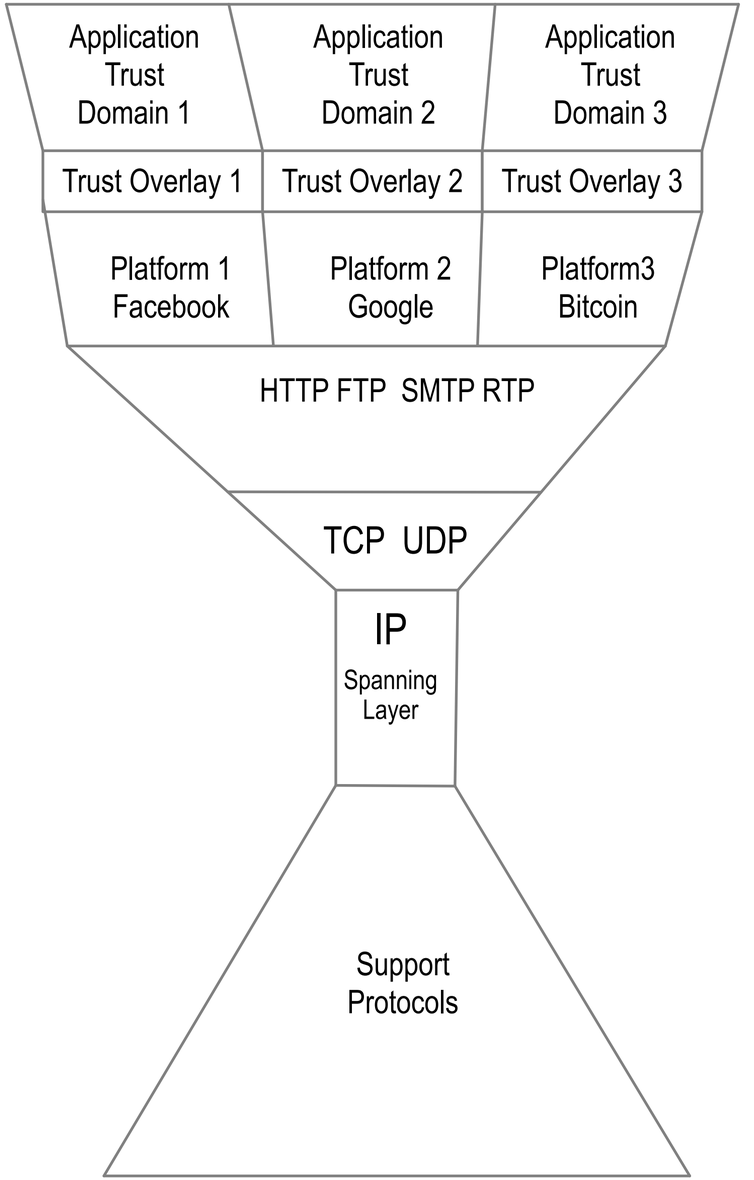
Hourglass


All Necessary Applications
Fewer Applications
Spanning Layer S
Weaker (fewer applications)
All Possible Supports
More Supports

Platform Locked Trust



Trust Domain Based Segmentation

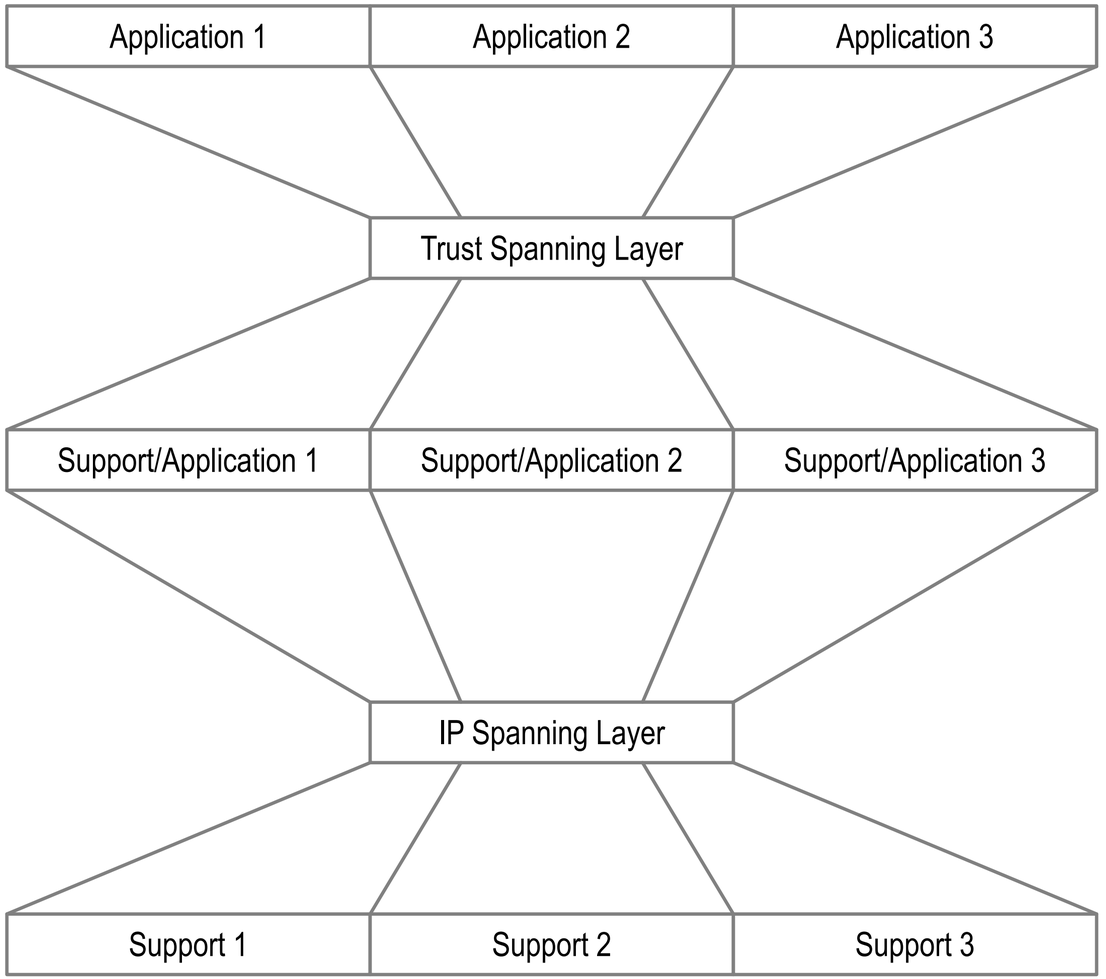


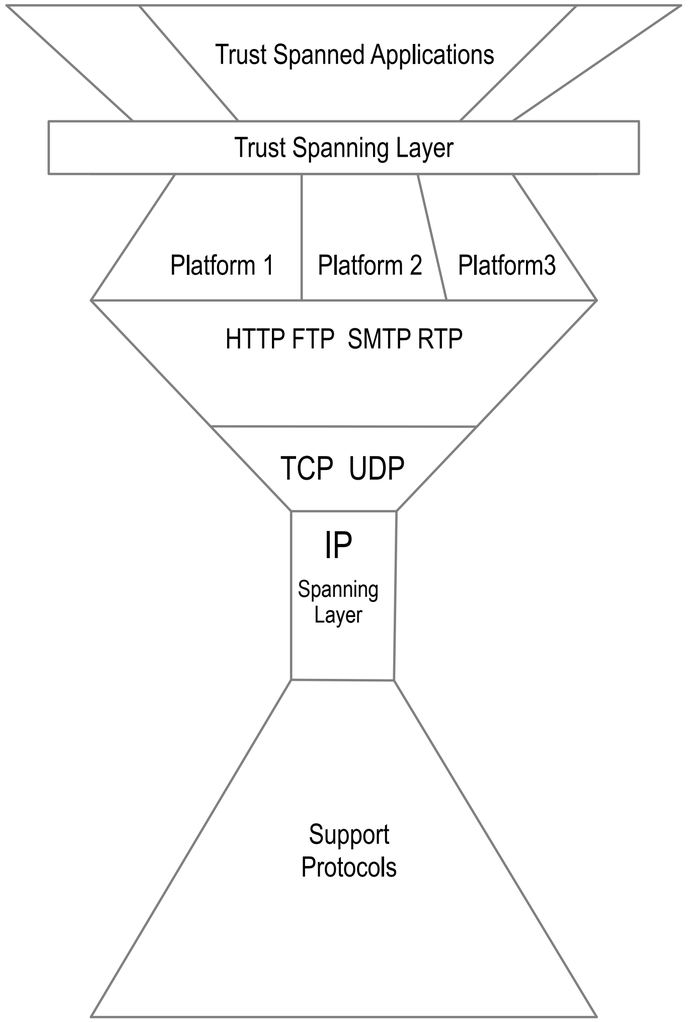
Each trust layer only spans platform specific applications

Bifurcates the internet trust map

No spanning trust layer

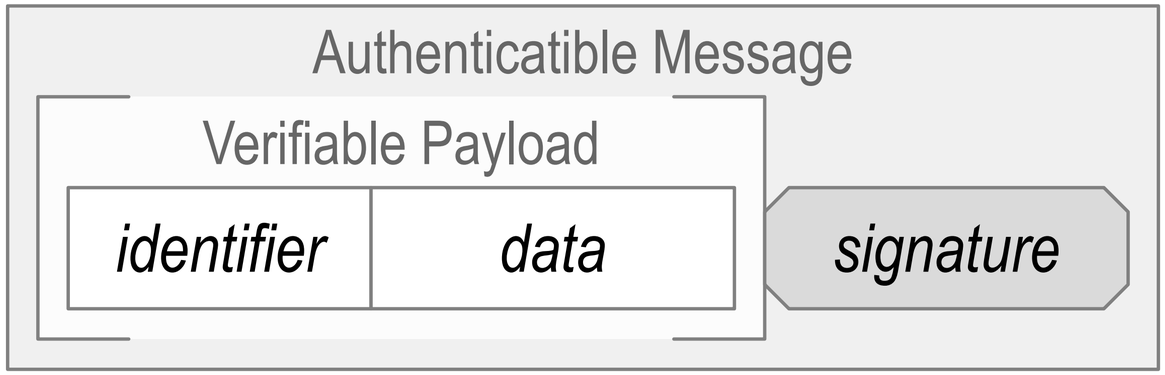
Waist and Neck



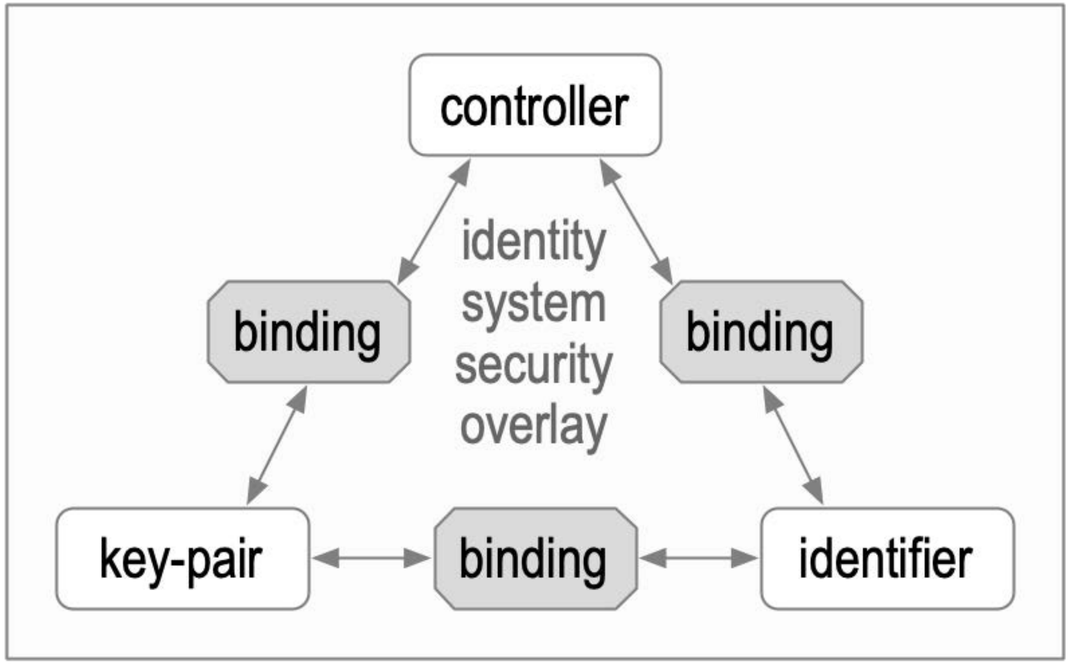


Identity System Security Overlay

Establish authenticity of IP packet's message payload.

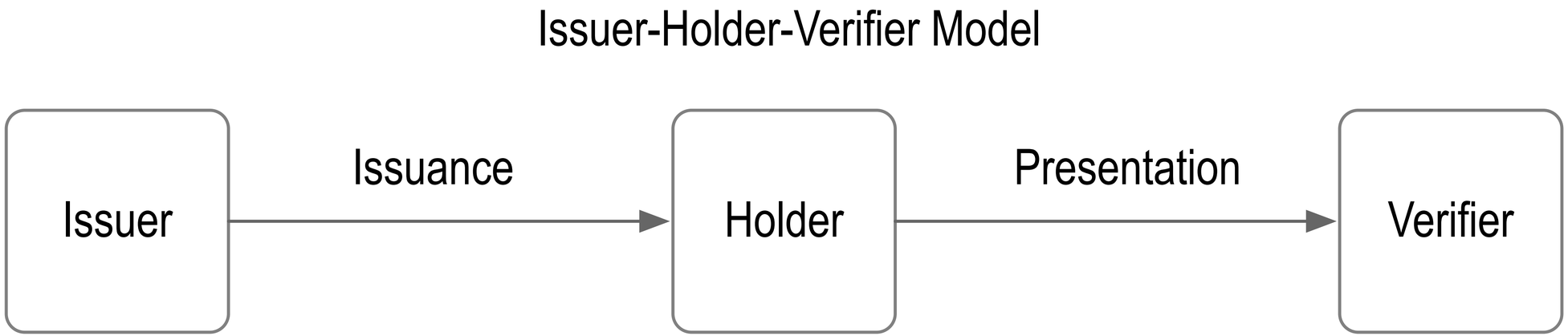


The overlay's security is contingent on the mapping's security.

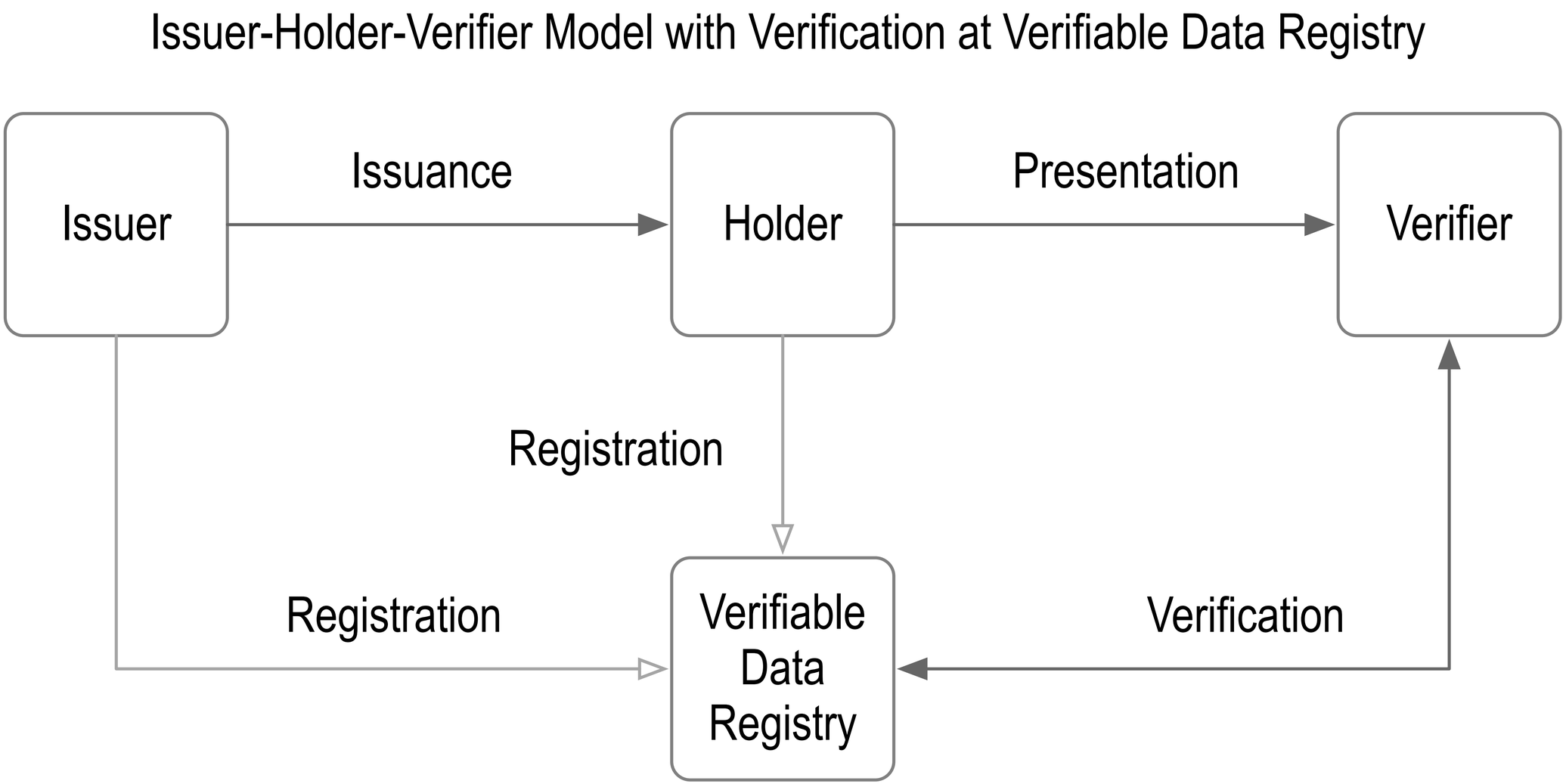


Identifier Issuance

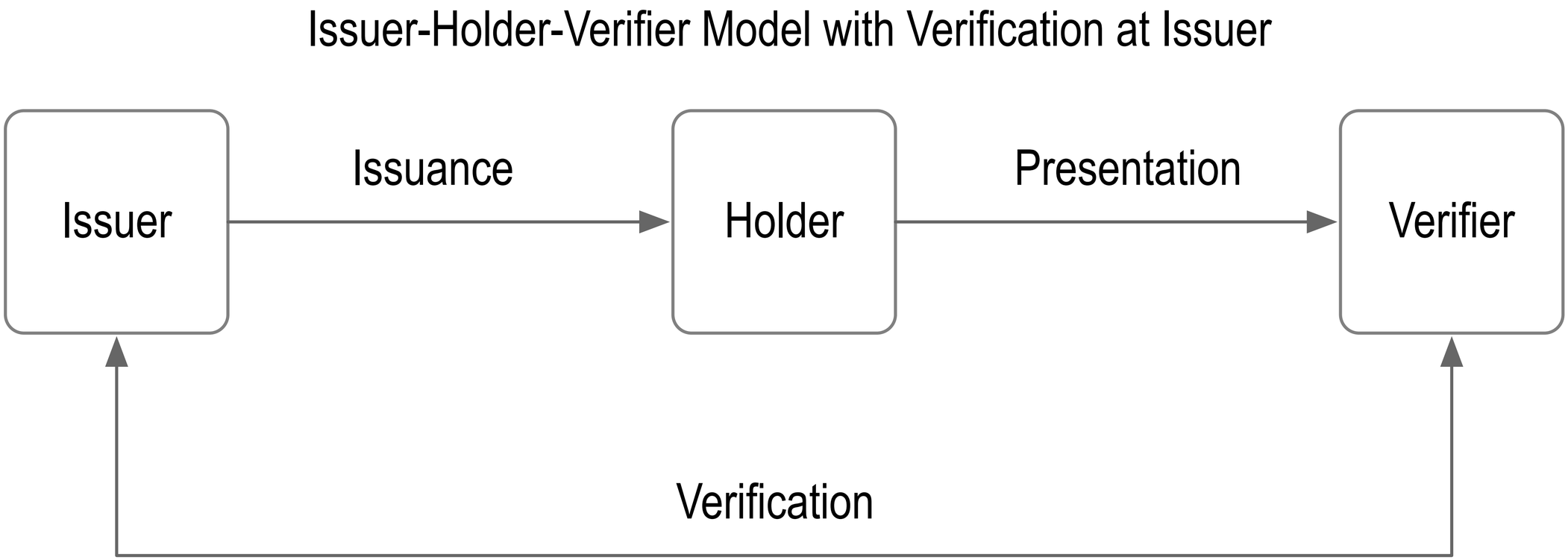
Tripartite Authentic Data Model



Tripartite with VDR

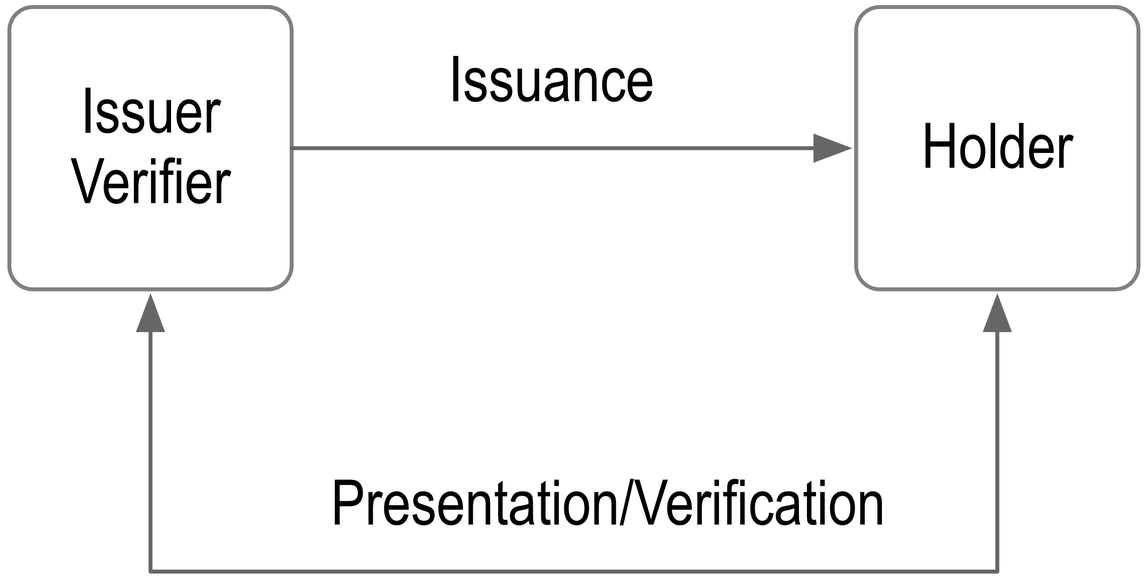


Tripartite without VDR

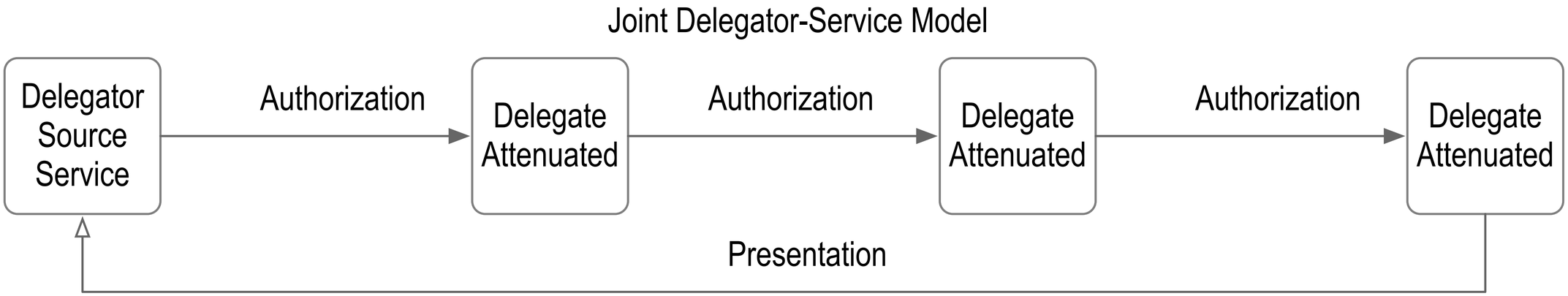


Bipartite Model

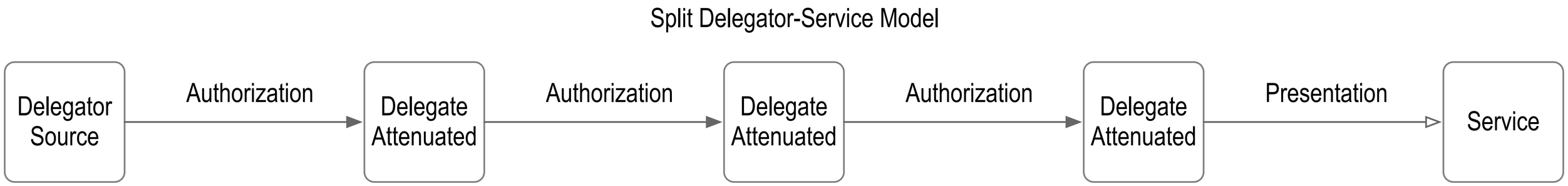
Issuer-Holder Model with Verification at Issuer



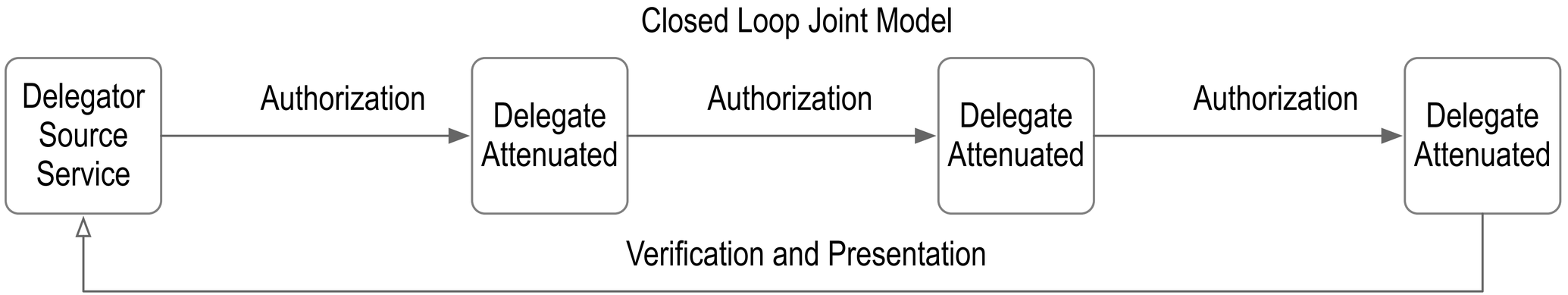
Joint Delegator-Service Model



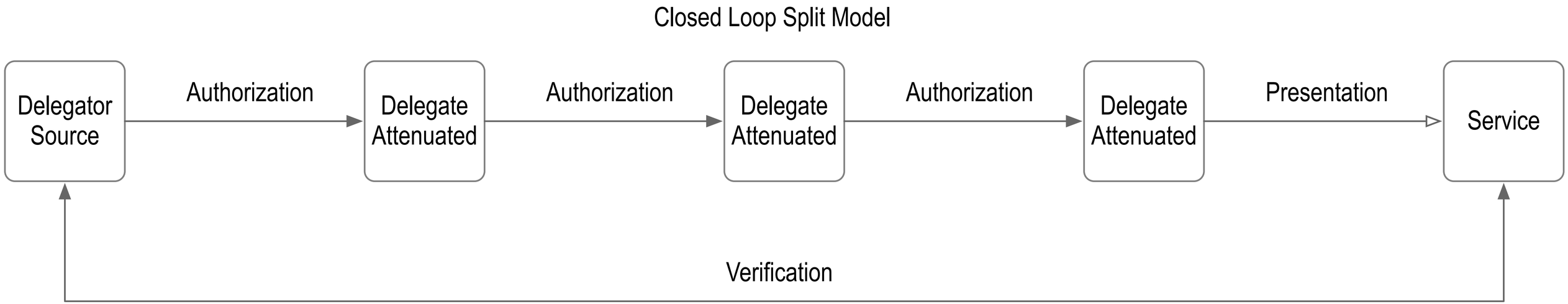
Split Delegator-Service Model



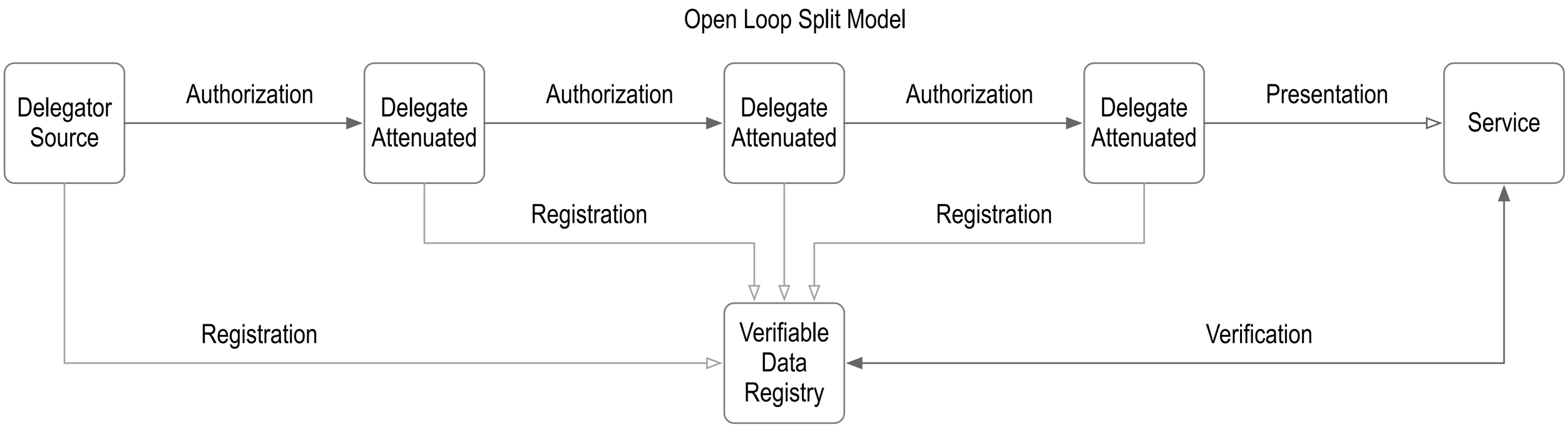
Closed Loop Joint Model



Closed Loop Split Model



Open Loop Split Model



Autonomic Identity System

why, *how* – *who* controls *what*, *when*, and *how*?

Root-of-Trust

cryptographic autonomic identifier = *why*, *how*

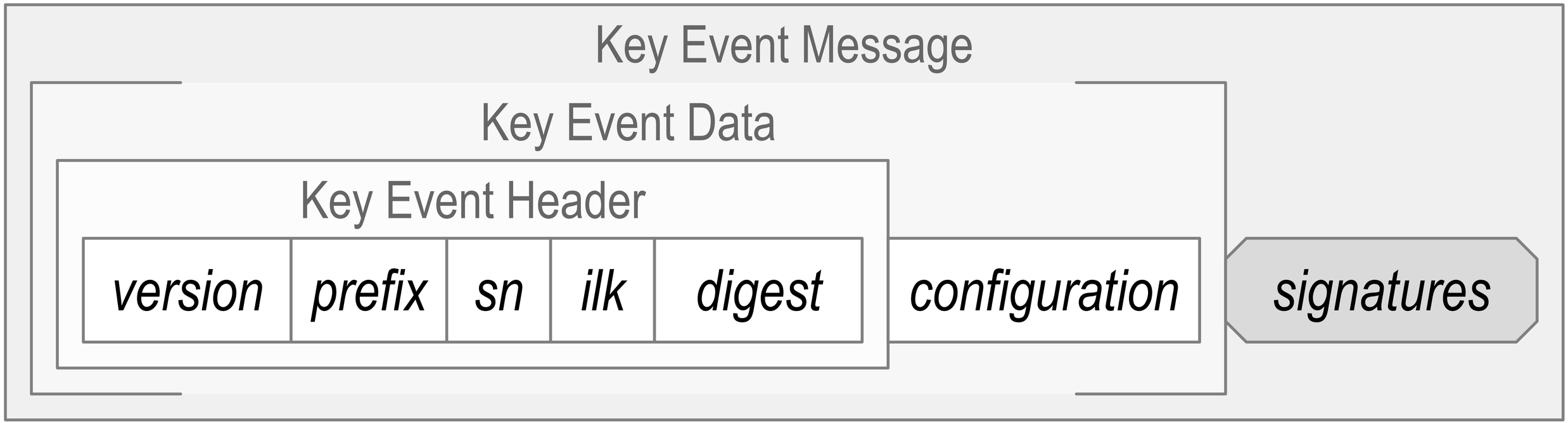
Source-of-Truth

controller of the private key = *who*

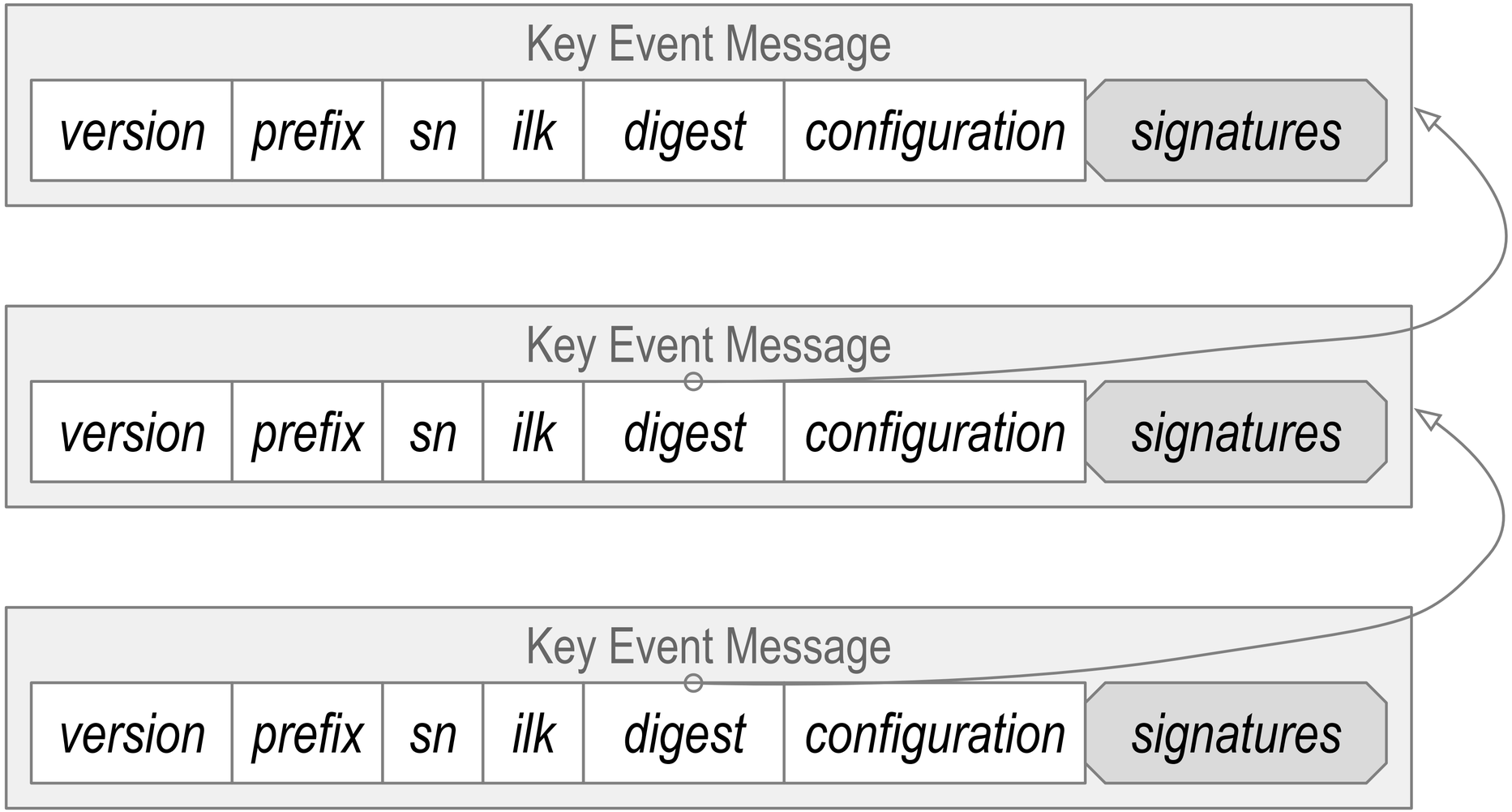
Loci-of-Control

authoritative operation = *what*, *when*, *how*

Key Event Message



Event Chaining

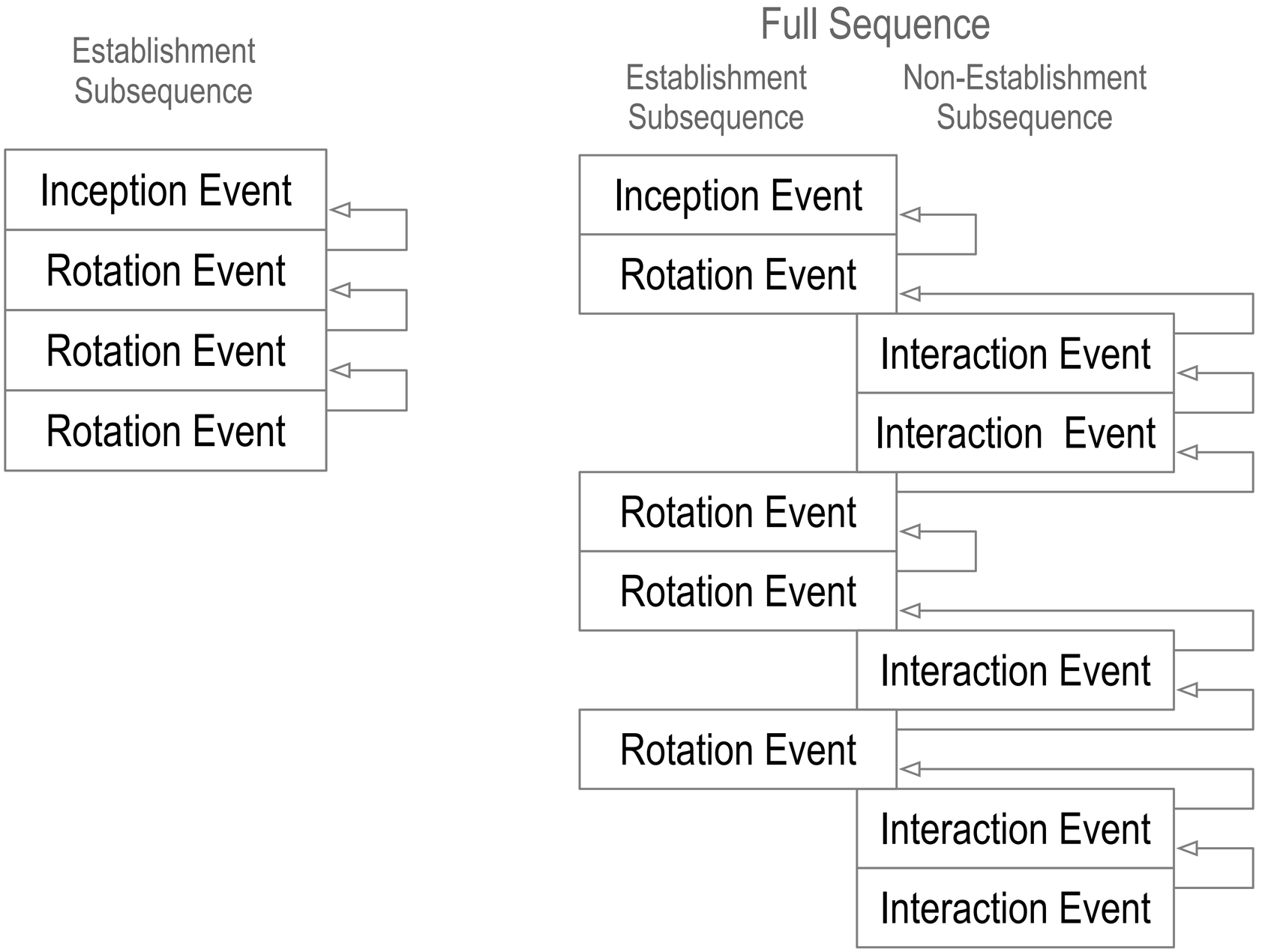


Self-Certifying Identifier Prefixes

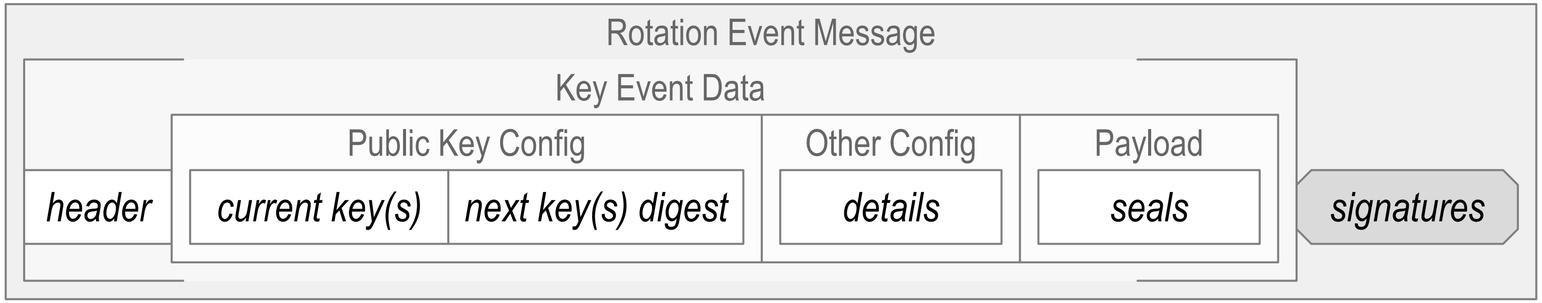
All crypto material appears in KERI in a fully qualified representation that includes a derivation code prepended to the crypto-material.

Identifier prefixes are fully qualified crypto-material.

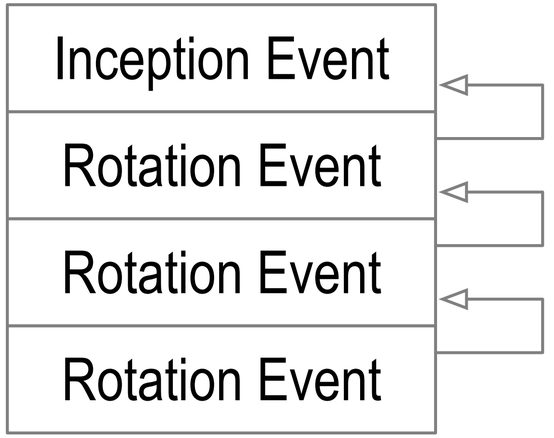
Event Sequencing



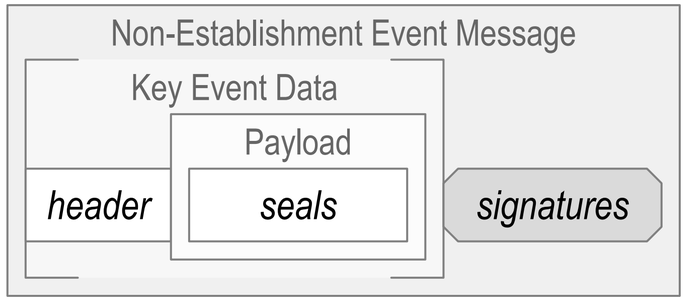
Establishment Events

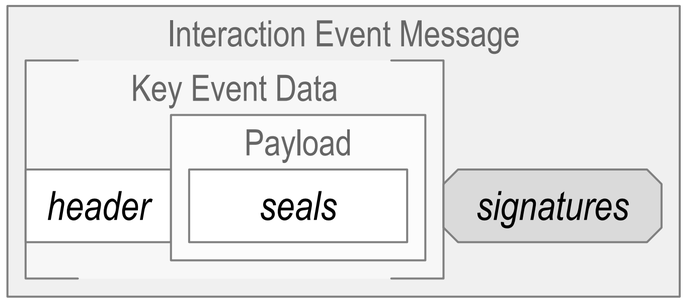


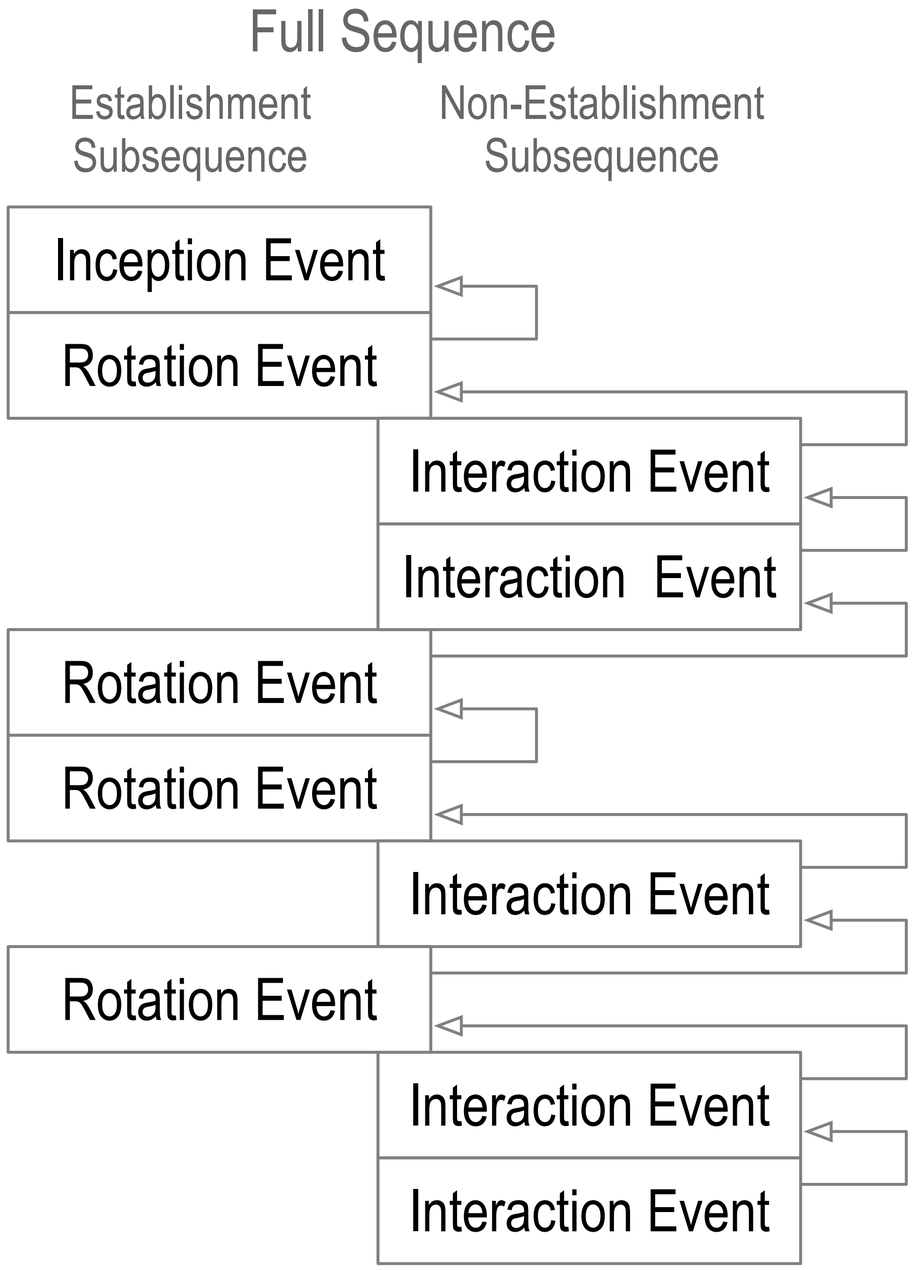
Establishment Subsequence



Non-Establishment Events

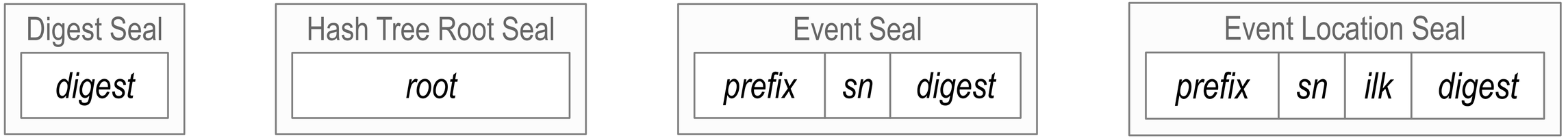






Seal (Anchor)

seal provides evidence of authenticity



A *seal* anchors arbitrary data to an event in the key event sequence thereby providing proof of control authority for that data at the location of the anchoring event.

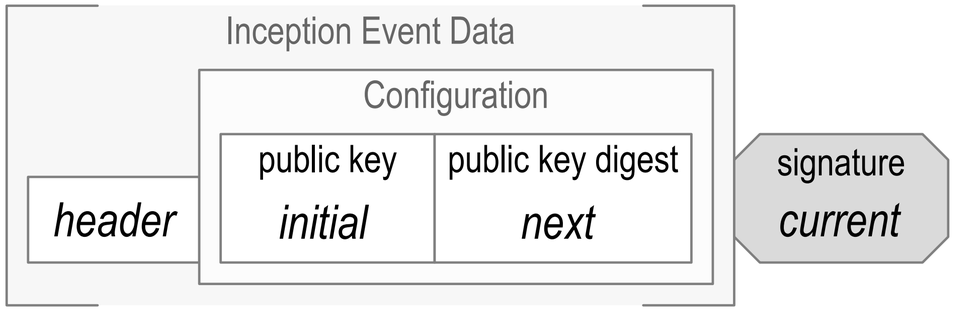
Seals make KERI both privacy preserving and *data semantic agnostic.*

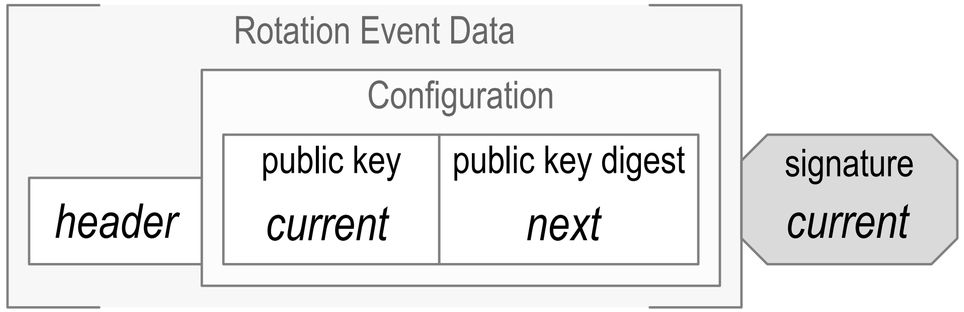
Context independent extensibility via externally layered APIs for anchored data instead of context dependent extensibility via internal linked data or tag registries.

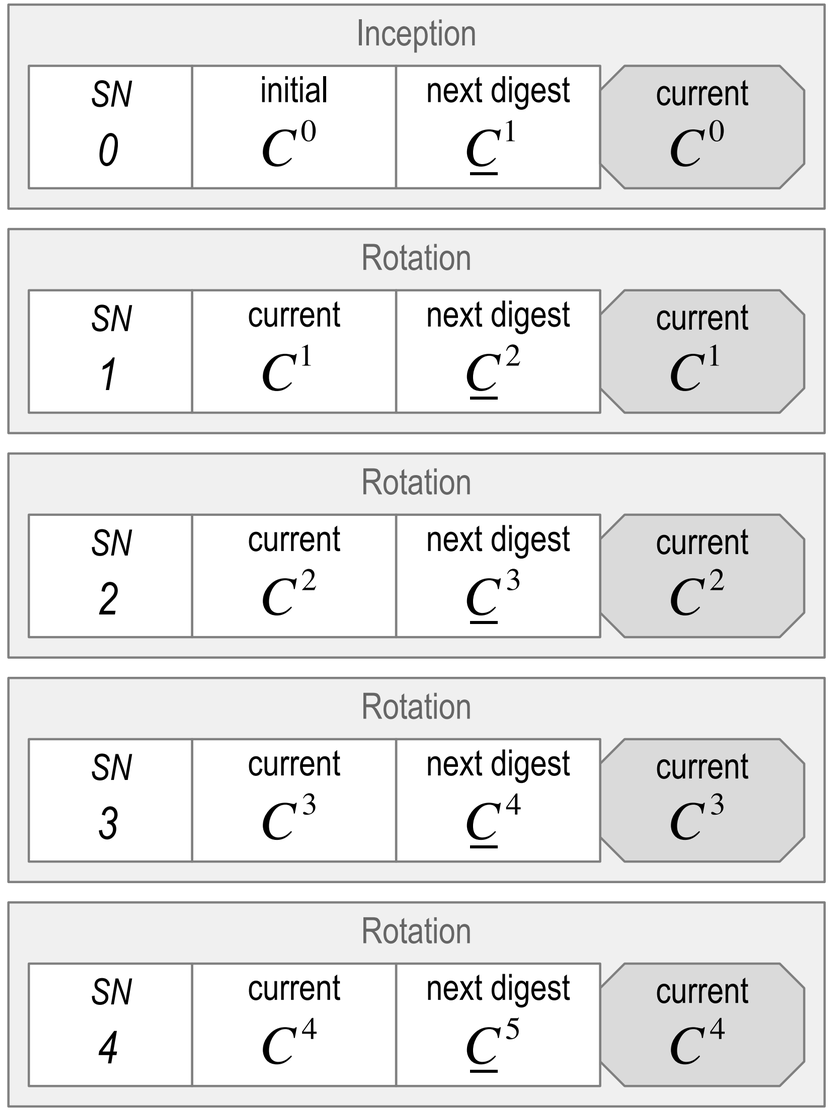
Interoperability is total w.r.t. establishment of control authority.

Minimally sufficient means.

Pre-Rotation

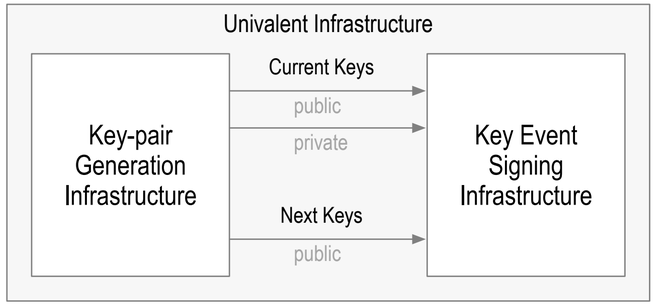


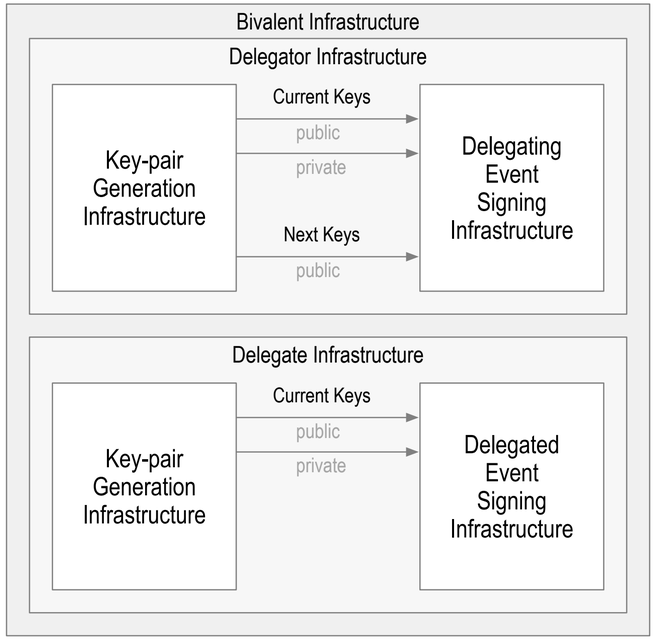


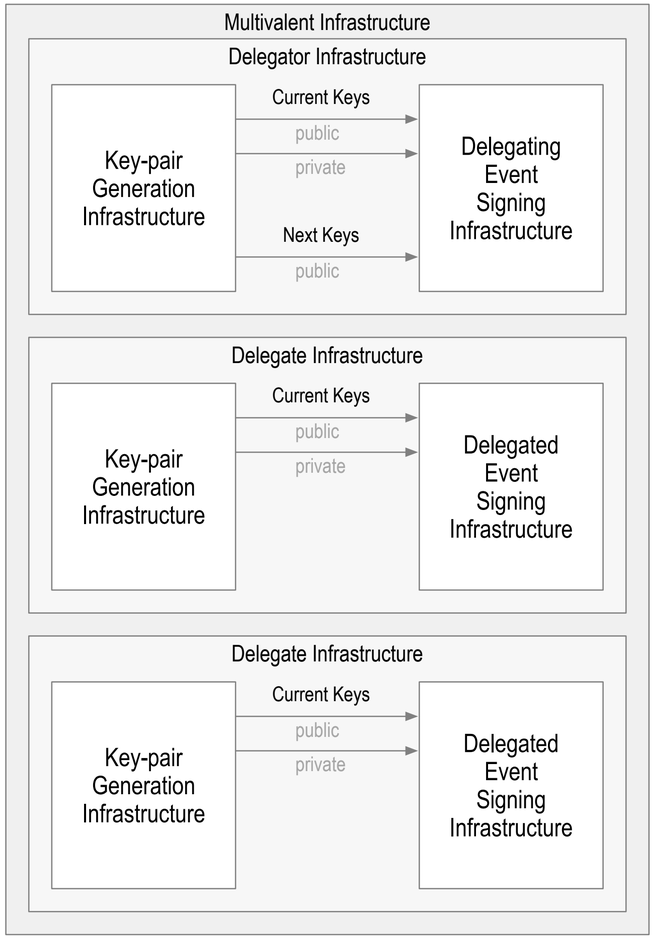


Digest of *next* key(s) makes pre-rotation post-quantum secure

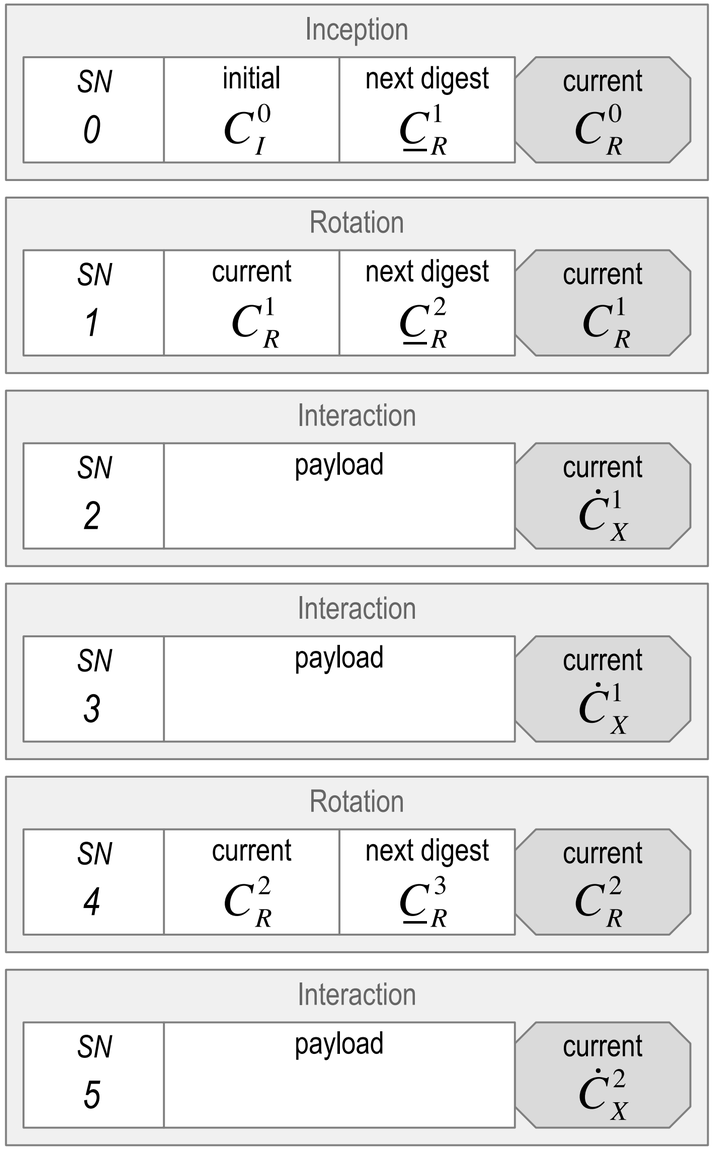
Key Infrastructure Valence

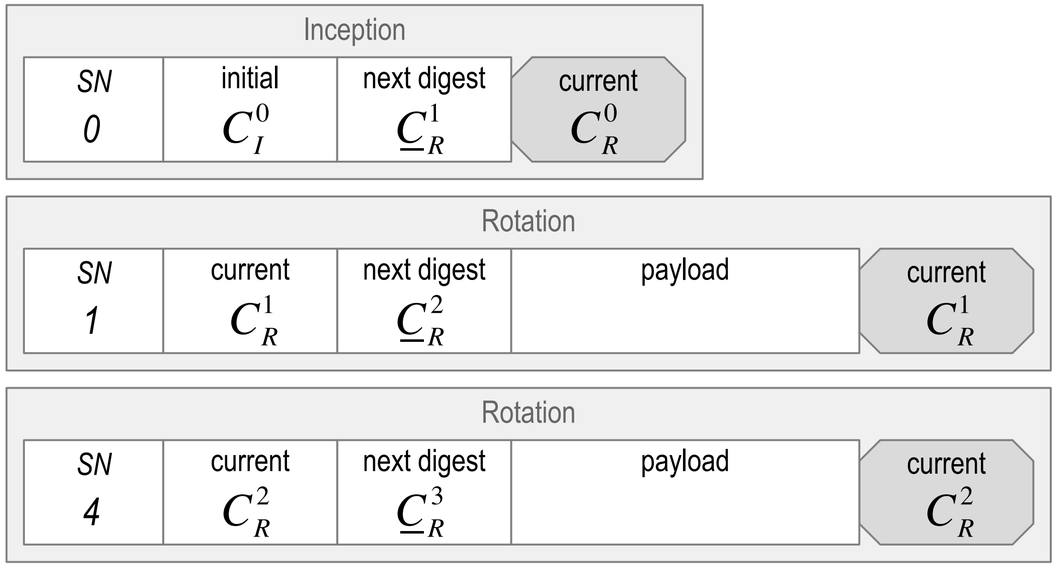






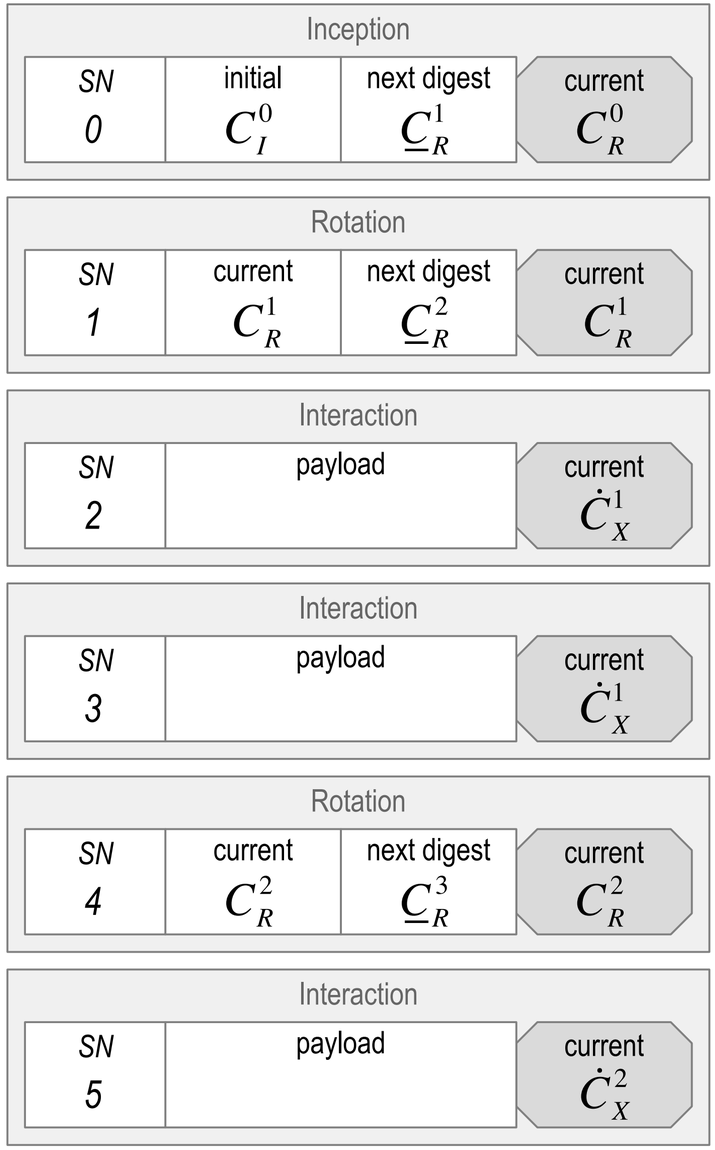
Repurposed Keys



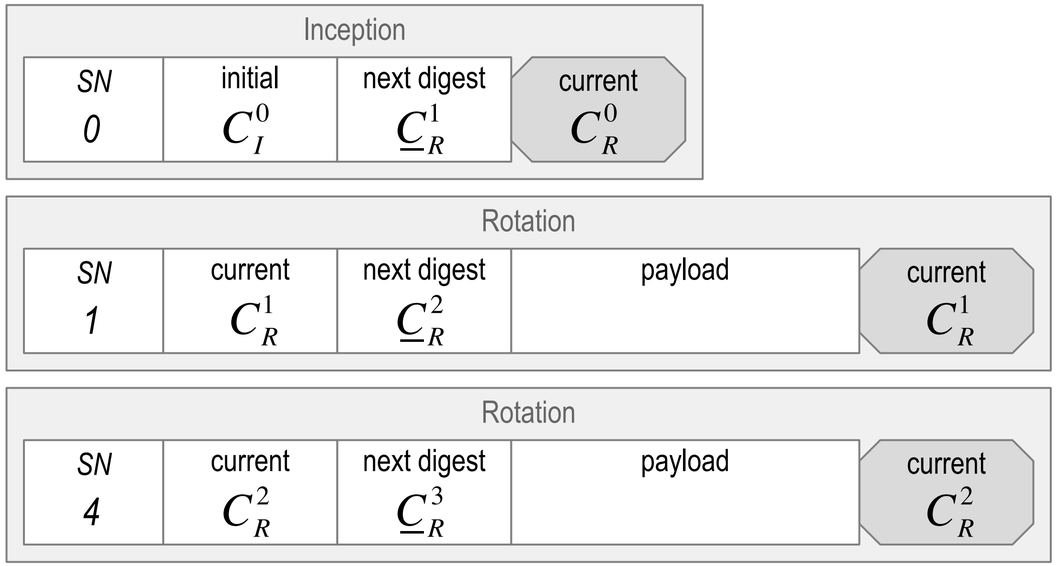


Univalent Key Roles

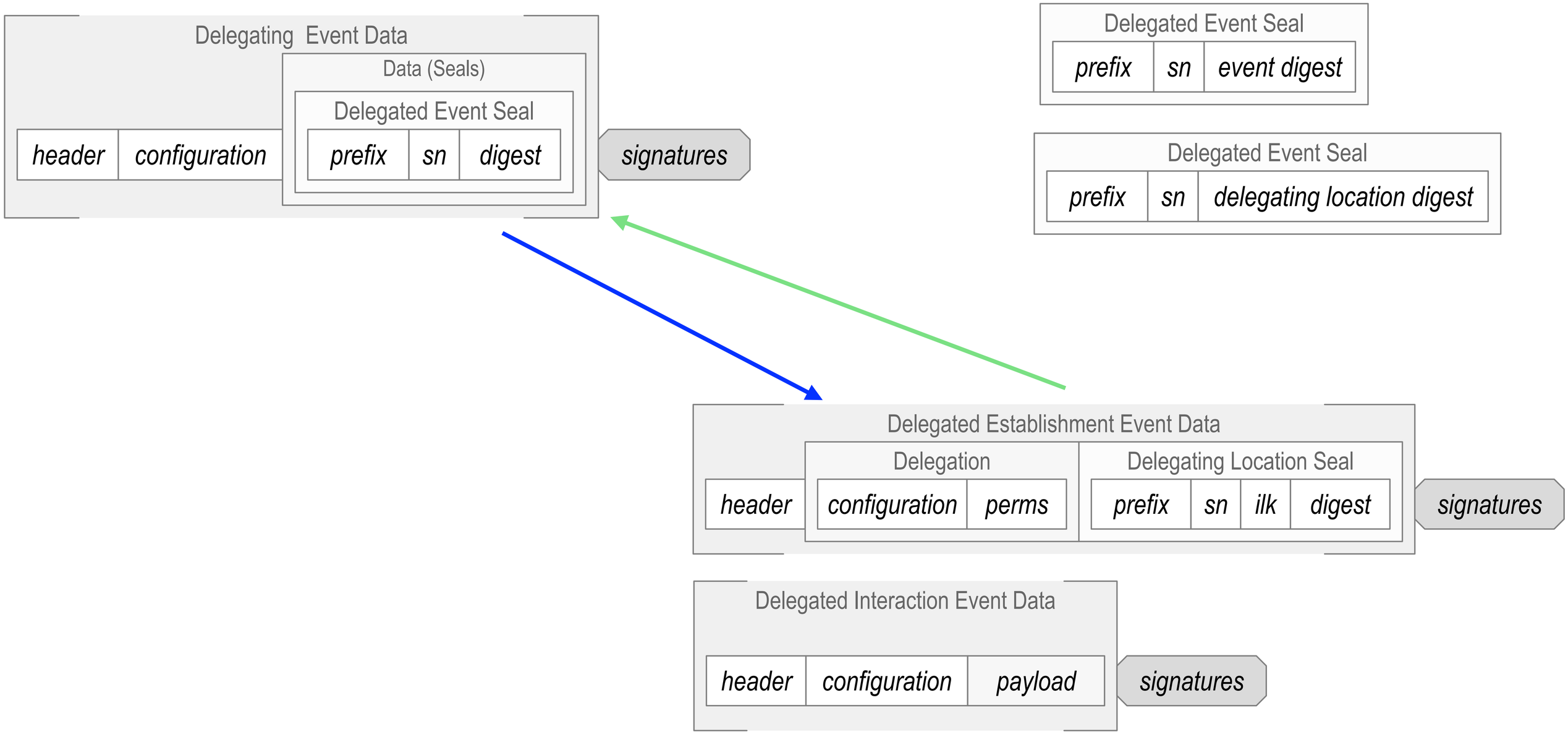
Repurposed Rotation to Interaction



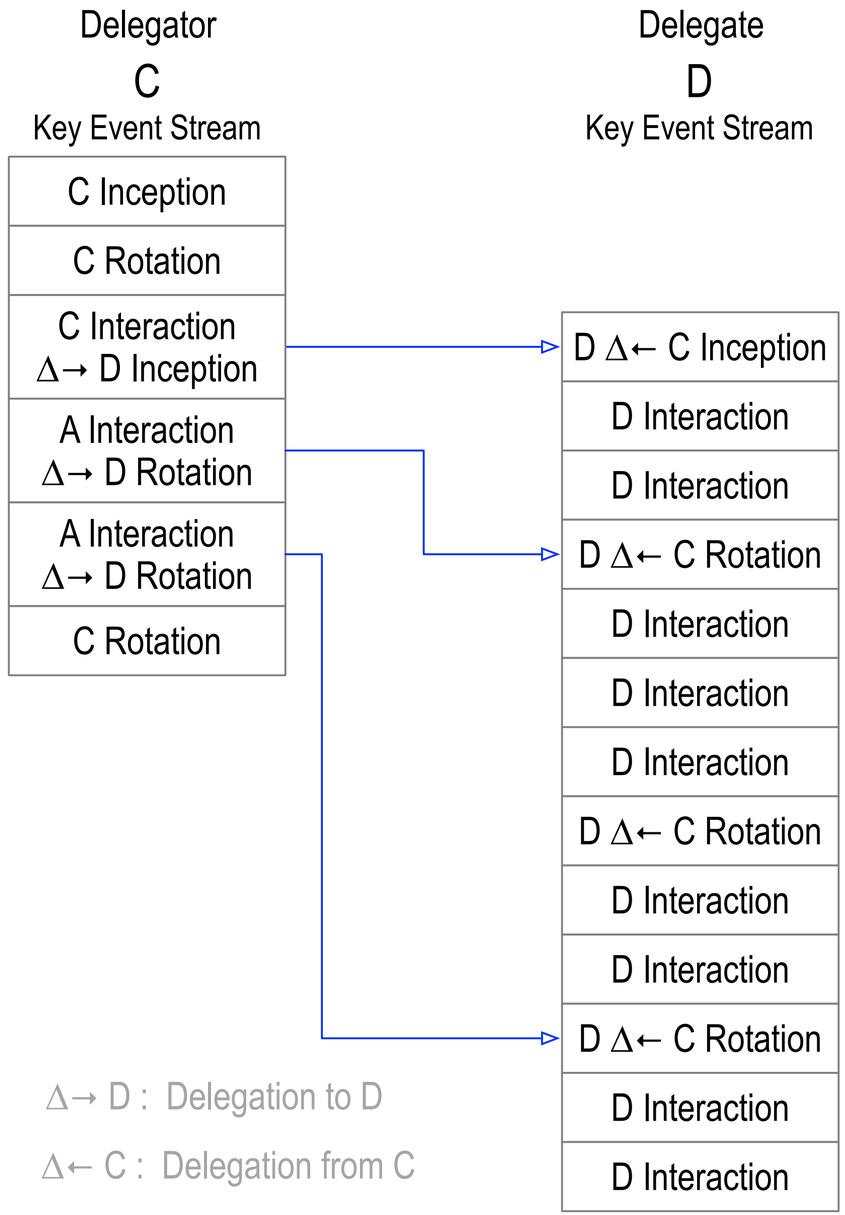
Rotation Only



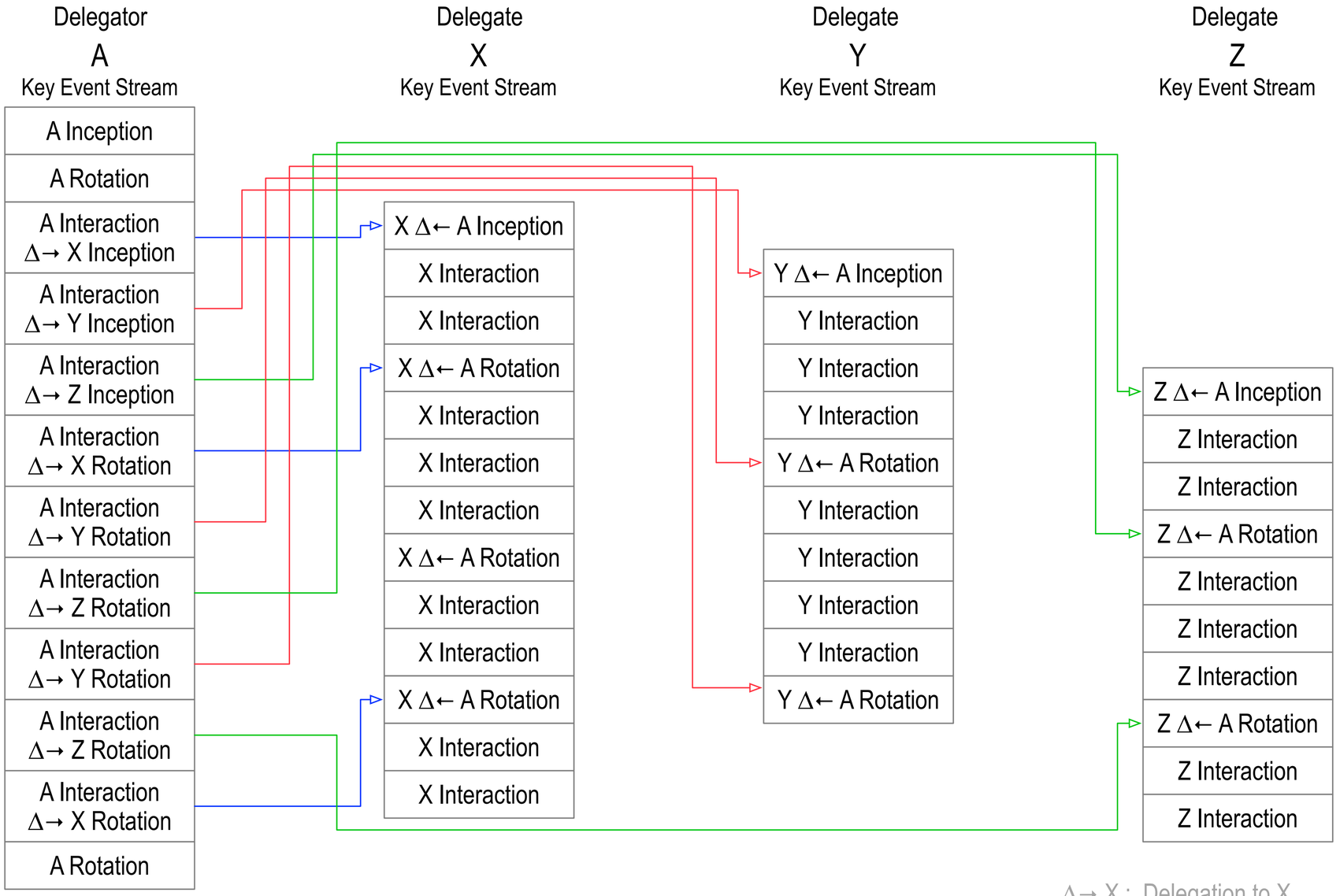
Delegation (Cross Anchor)



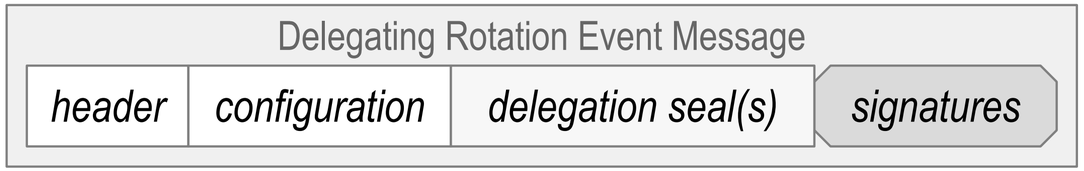
Interaction Delegation

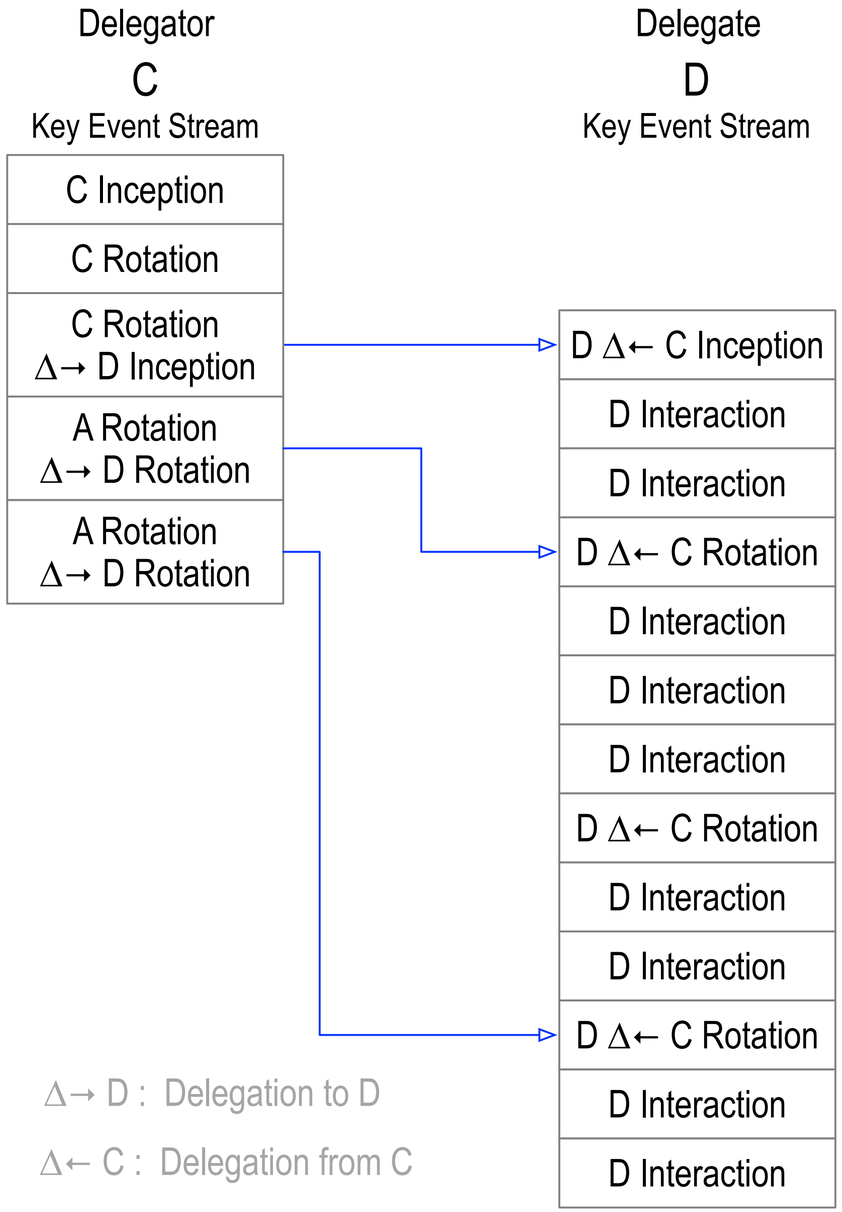


Scaling Delegation via Interaction

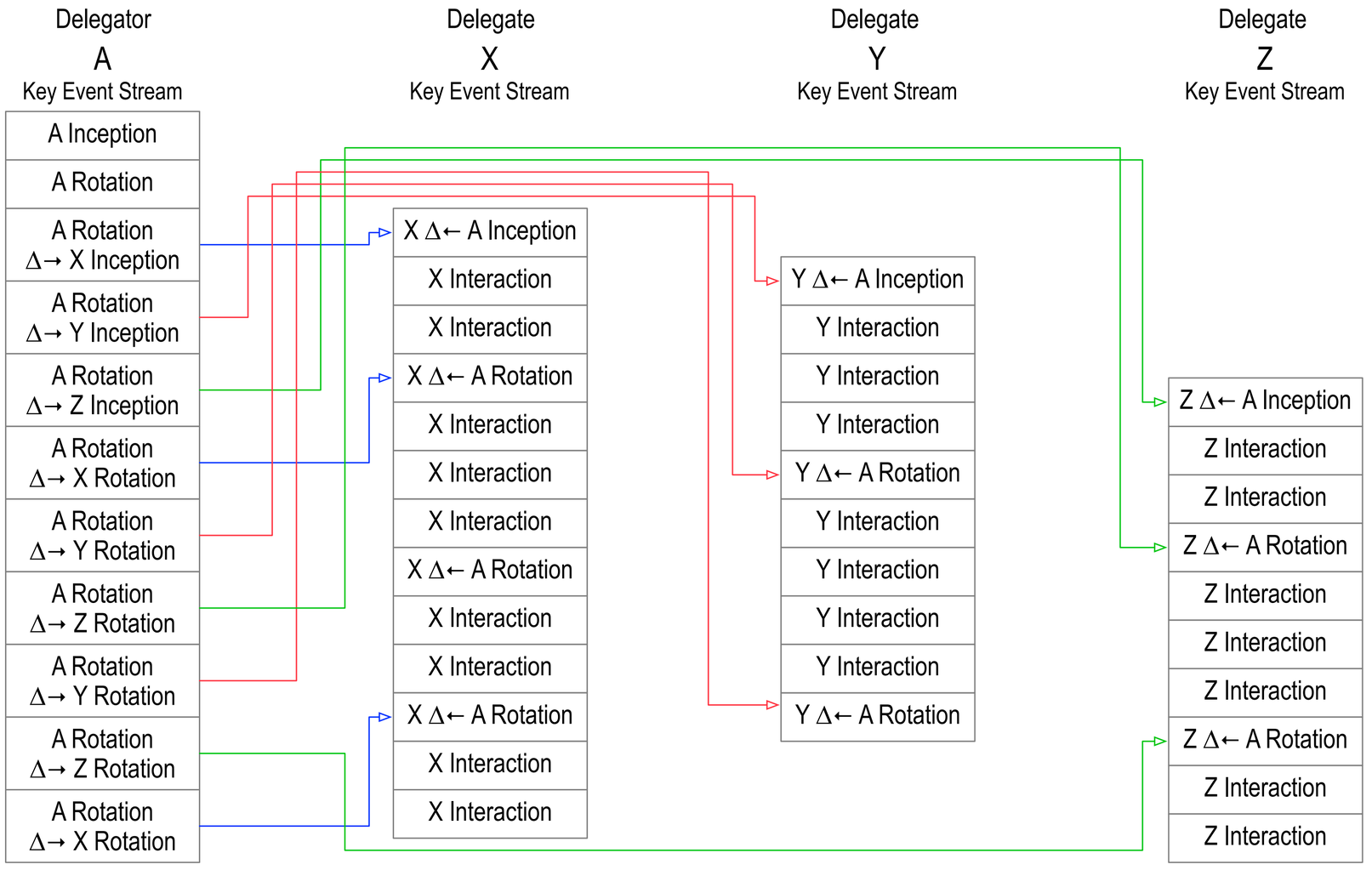


Rotation Delegation





Scaling Delegation via Rotation



Δ→ X : Delegation to X

Δ← A : Delegation from A

Security Concepts

Availability, Consistency, and Duplicity

Harm to controller: Unavailability, loss of control authority, externally forced duplicity

Harm to validator: Inadvertent acceptance of verifiable but forged or duplicitous events

Local vs. Global Duplicity Guarantees

Direct Mode vs. Indirect Mode Operation

Malicious Controller vs. Malicious Third Party

Live Exploit vs. Dead Exploit

Controller Protection vs. Validator Protection

Protection to controller: key management, promulgation consensus, redundancy.

Protection to validator: verifiable logs, verification consensus, duplicity detection

Ledger Attack Vectors

TABLE I

ATTACK VECTORS RELATED TO THE ATTACK CLASS IN BLOCKCHAIN SYSTEMS. WE ALSO SHOW, BY REFERENCING TO THE PRIOR WORK, HOW EACH ATTACK AFFECTS THE ENTITIES INVOLVED WITH BLOCKCHAIN SYSTEMS. FOR INSTANCE, ORPHANED BLOCKS AFFECT THE BLOCKCHAIN, THE MINERS, AND THE MINING POOLS.

	Attacks	Blockchain	Miners	Mining Pools	Exchanges	Application	Users
Blockchain Structure	Forks [26]	✓					
	Orphaned blocks [27]	✓	✓	✓			
Peer-to-Peer System	DNS hijacks [28]		✓	✓	✓		✓
	BGP hijacks [29]		✓	✓			✓
	Eclipse attack [30]		✓				✓
	Majority attack [31]	✓	✓			✓	
	Selfish mining [32]	✓	✓	✓			
	DDoS attacks [33]	✓	✓	✓			
	Consensus Delay [34]		✓	✓			✓
	Block Withholding [34]		✓	✓			
	Timejacking attacks [35]		✓	✓		✓	
	Finney attacks [36]		✓	✓			✓
Blockchain Application	Blockchain Ingestion [37]	✓					
	Wallet theft [38]				✓	✓	✓
	Double-spending [39]	✓					✓
	Cryptojacking [40]					✓	✓
	Smart contract DoS [41]	✓				✓	✓
	≈ Reentracy attacks [42]					✓	✓
	≈ Overflow attacks [42]					✓	✓
	≈ Replay attacks [41]	✓		✓		✓	✓
	≈ Short address attacks [42]					✓	
	≈ Balance attacks [41]					✓	✓

Exploring the Attack Surface of Blockchain: A Systematic Overview https://arxiv.org/pdf/1904.03487.pdf

Ledger Attack Effects

TABLE II

IMPLICATIONS OF EACH ATTACK ON THE BLOCKCHAIN SYSTEM IN THE LIGHT OF THE PRIOR WORK. FOR INSTANCE, FORKS CAN LEAD TO CHAIN SPLITTING AND REVENUE LOSS. AS A RESULT OF A FORK, ONE AMONG THE CANDIDATE CHAINS IS SELECTED BY THE NETWORK WHILE THE OTHERS ARE INVALIDATED. THIS LEADS TO INVALIDATION OF TRANSACTION AND REVENUE LOSS TO MINERS.

	Attacks	Chain Splitting	Revenue Loss	Partitioning	Malicious Mining	Delay	Info Loss	Theft
Blockchain Splitting	Forks [26]	✓	✓					
	Orphaned Blocks [27]		✓					
P2P System	DNS hijacks [28]		✓	✓				✓
	BGP hijacks [29]		✓	✓				✓
	Eclipse attacks [30]			✓				
	Majority attacks [31]	✓	✓		✓			
	Selfish mining [32]		✓		✓			
	DDoS attacks [33]				✓			✓
	Consensus Delay [34]					✓	✓	
	Block Withholding [34]		✓		✓			
	Timejacking attacks [35]	✓	✓		✓	✓		
	Finney attacks [36]		✓					
Blockchain Application	Blockchain Ingestion [37]						✓	
	Wallet theft [38]		✓					✓
	Double-spending [39]							
	Cryptojacking [40]	✓			✓			✓
	Smart contract DoS [41]		✓			✓		✓
	≈ Reentracy attacks [42]		✓					✓
	≈ Overflow attacks [42]							✓
	≈ Replay attacks [41]		✓				✓	
	≈ Short address attacks [42]		✓					✓
	≈ Balance attacks [41]		✓					✓

Exploring the Attack Surface of Blockchain: A Systematic Overview https://arxiv.org/pdf/1904.03487.pdf

Apples-to-Apples

Ledger:

Network paid by transaction fees

(more or less competitive within the network)

Successful exploits without compromised keys

Controller:

Highly available nodes of other's choosing

Must trust that a majority are honest

No recovery if keys compromised

Validator;

Need full copy of ledger (big)

Need full access to network

Must trust that a majority are honest

KERI:

Networks paid by transaction fees

(competitive across all networks)

Controller:

Highly available nodes of own choosing

Must "*trust*" that a majority are honest

Successful exploits must compromise keys

Recovery if keys compromised

Validator:

Need full copy of KEL (small)

Need full access to network of own choosing.

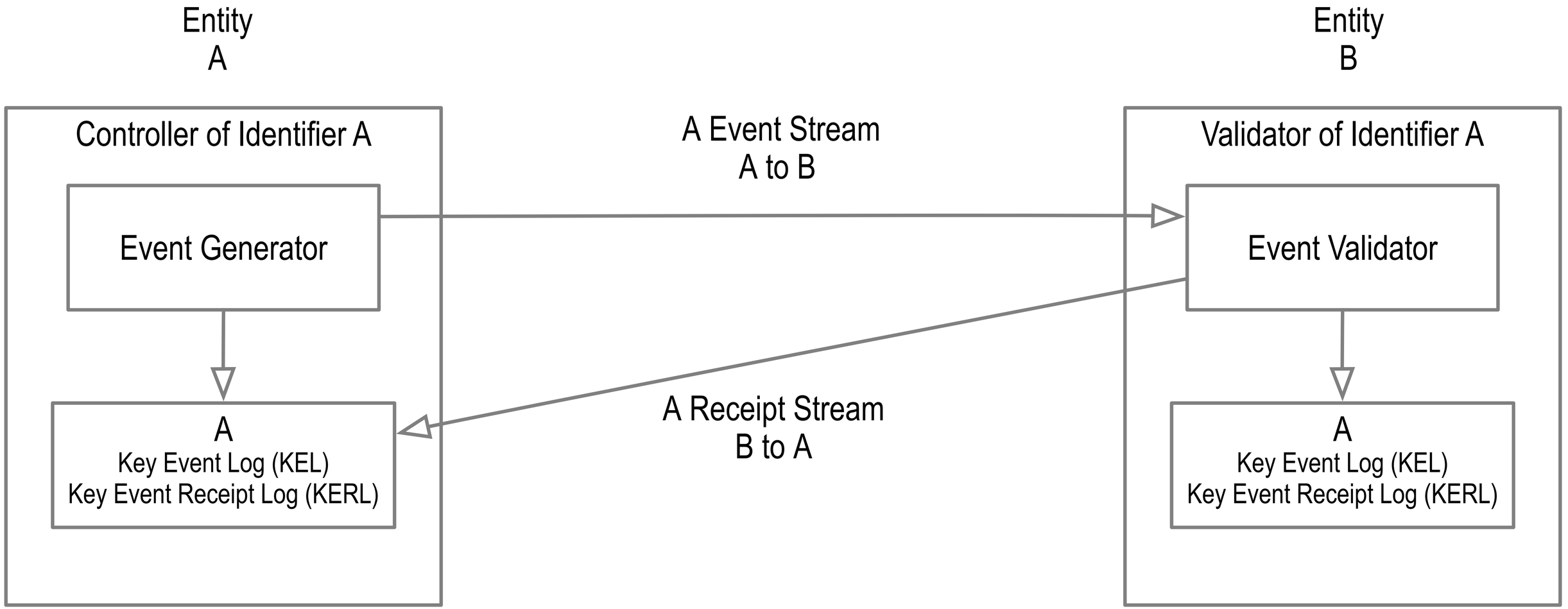
Must "*trust*" that a majority are honest

Protocol Operational Modes

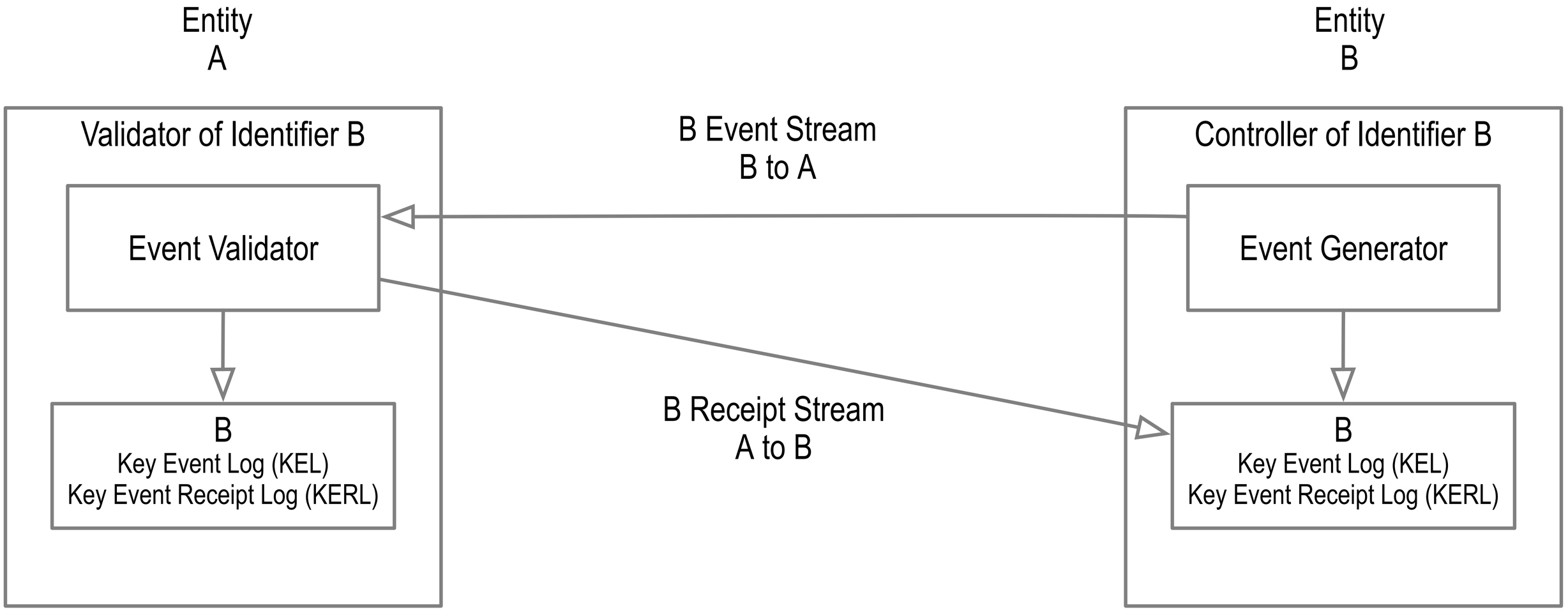
Direct Event Replay Mode (one-to-one)

Indirect Event Replay Mode (one-to-any)

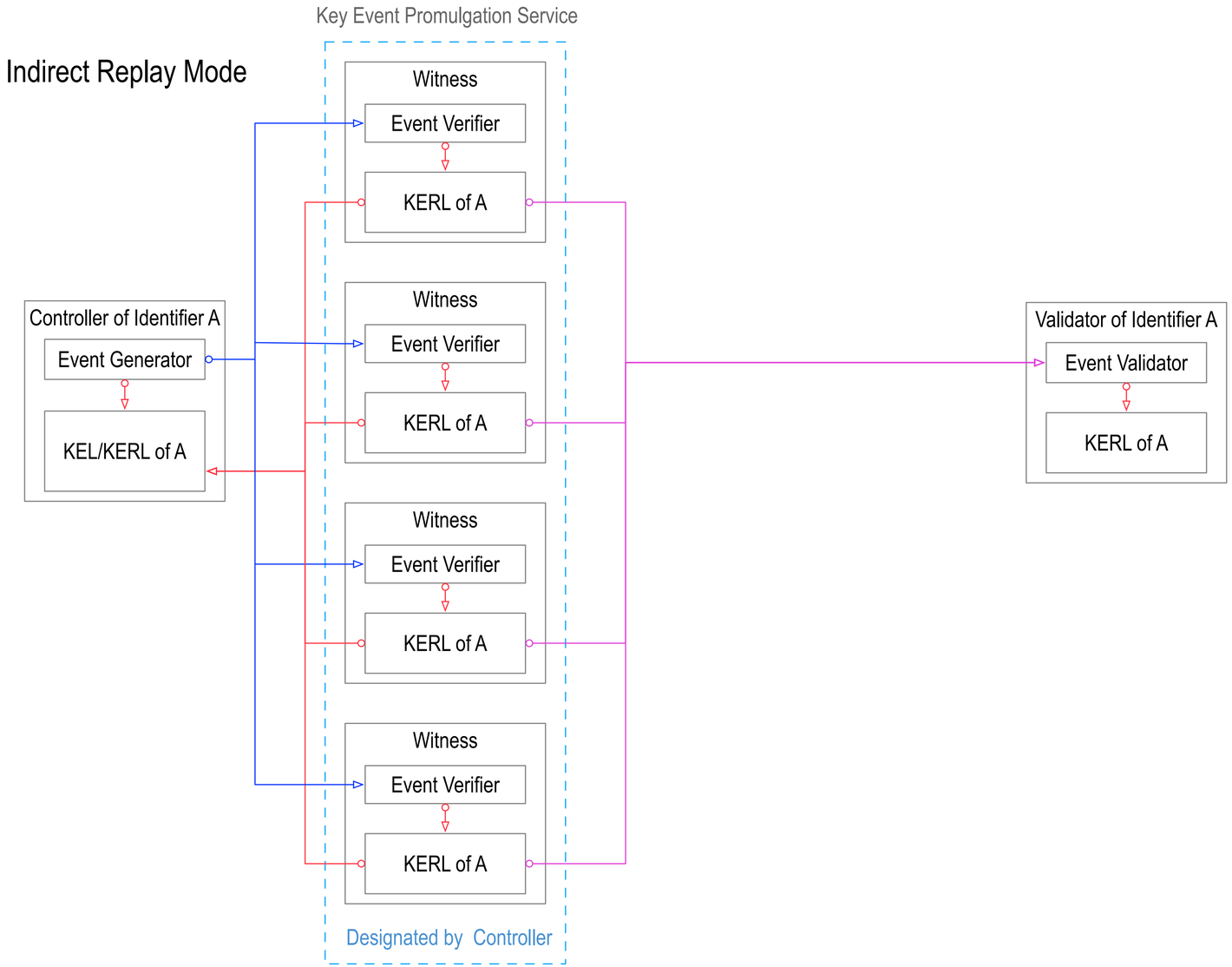
Direct Mode: A to B



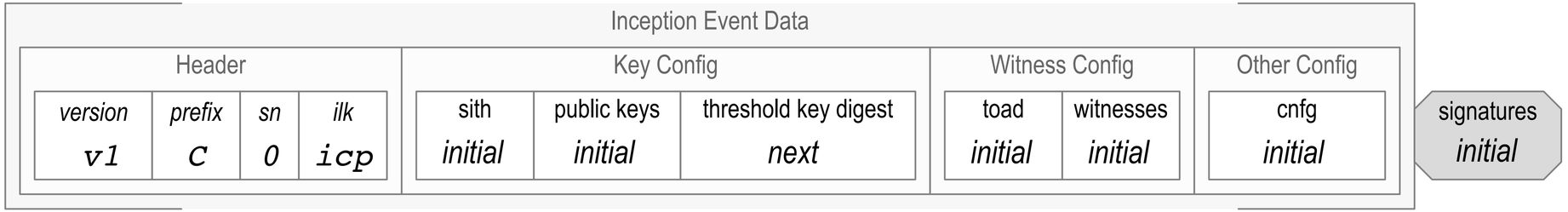
Direct Mode: B to A



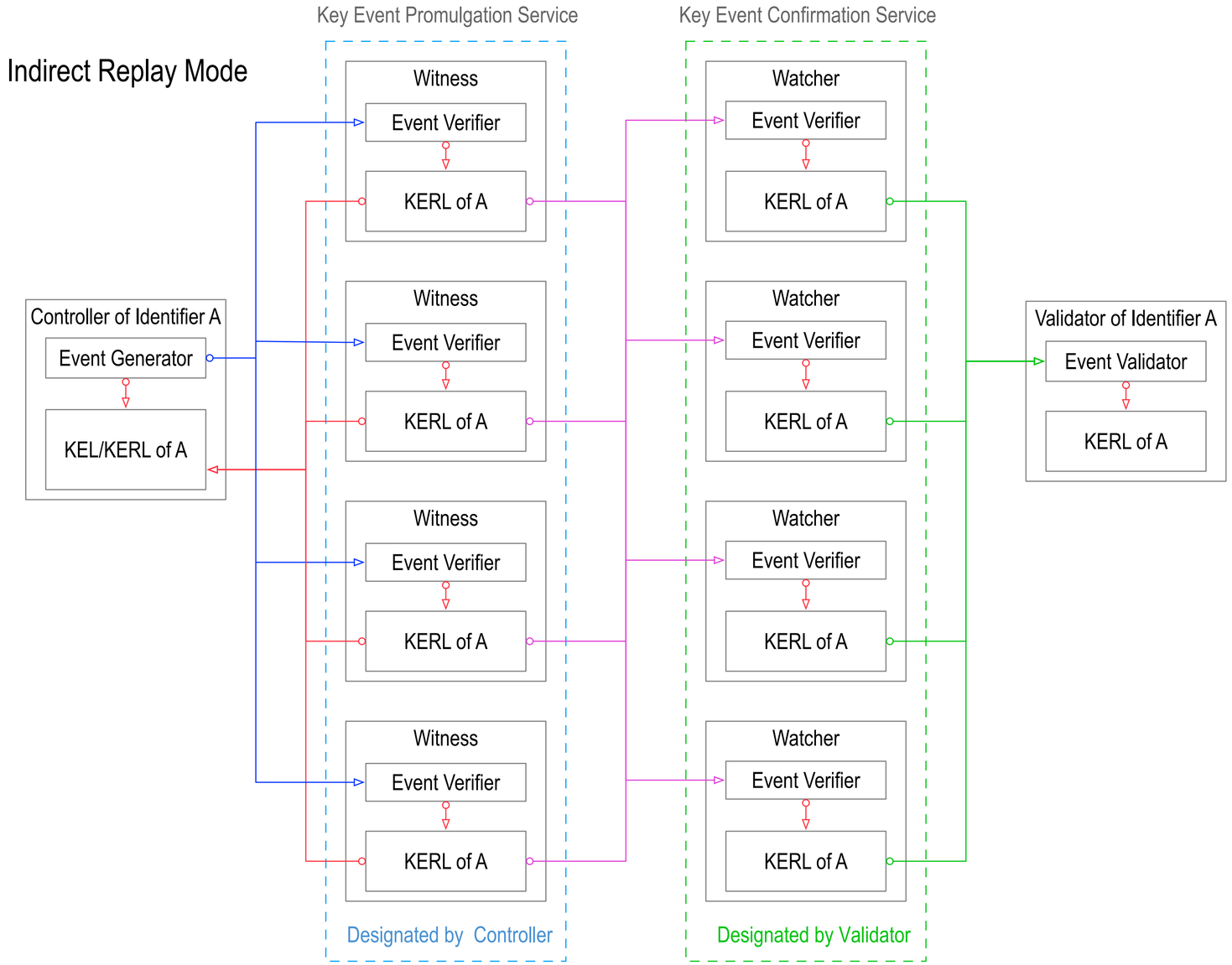
Indirect Mode Promulgation Service



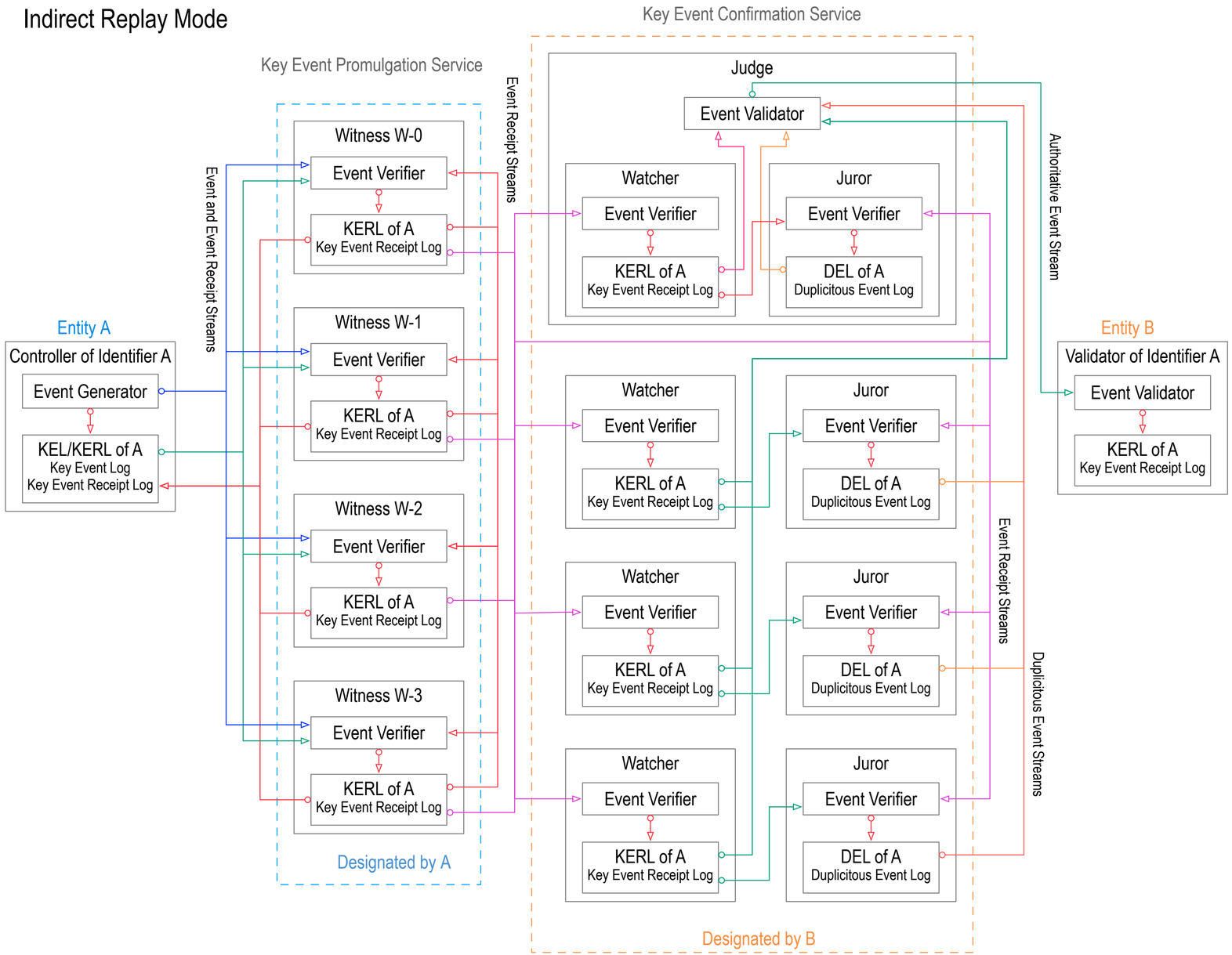
Establishment Events



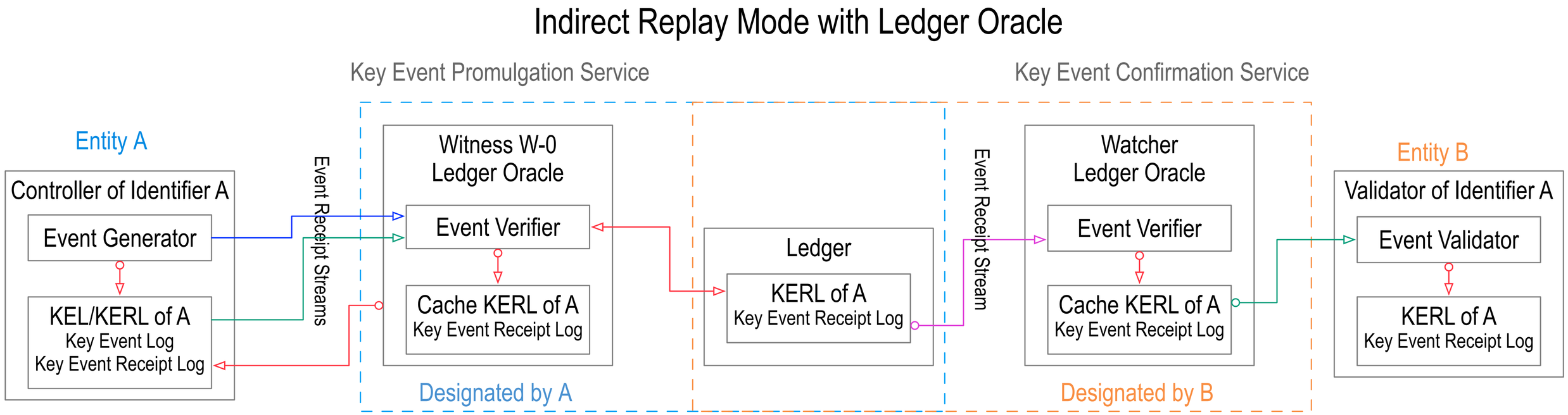
Indirect Mode
Promulgation and Confirmation Services



Indirect Mode Full



Indirect Mode with Ledger Oracles



Separation of Control

Shared ledger = *shared control* over *shared data*.

Shared *data* = *good*, shared *control* = *bad*.

Shared control between controllers and validators may be problematic for governance, scalability, and performance.

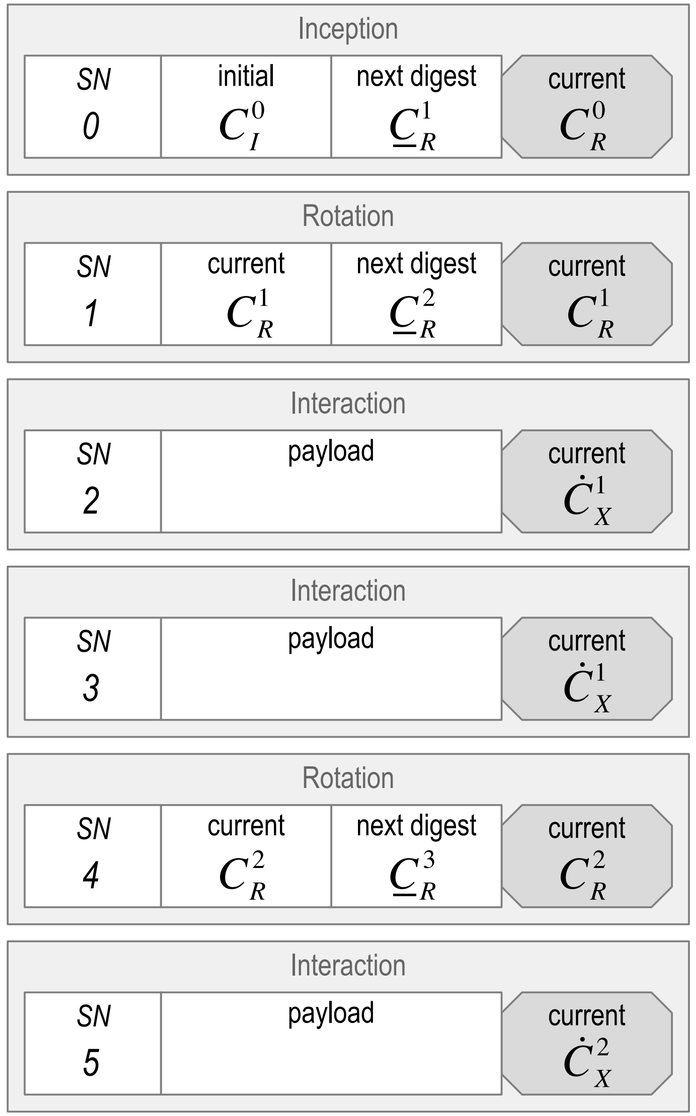
KERI = *separated control* over *shared data*.

Separated control between controllers and validators may provide better decentralization, more flexibility, better scalability, lower cost, higher performance, and more privacy at comparable security.

Live Exploit (current signing keys)

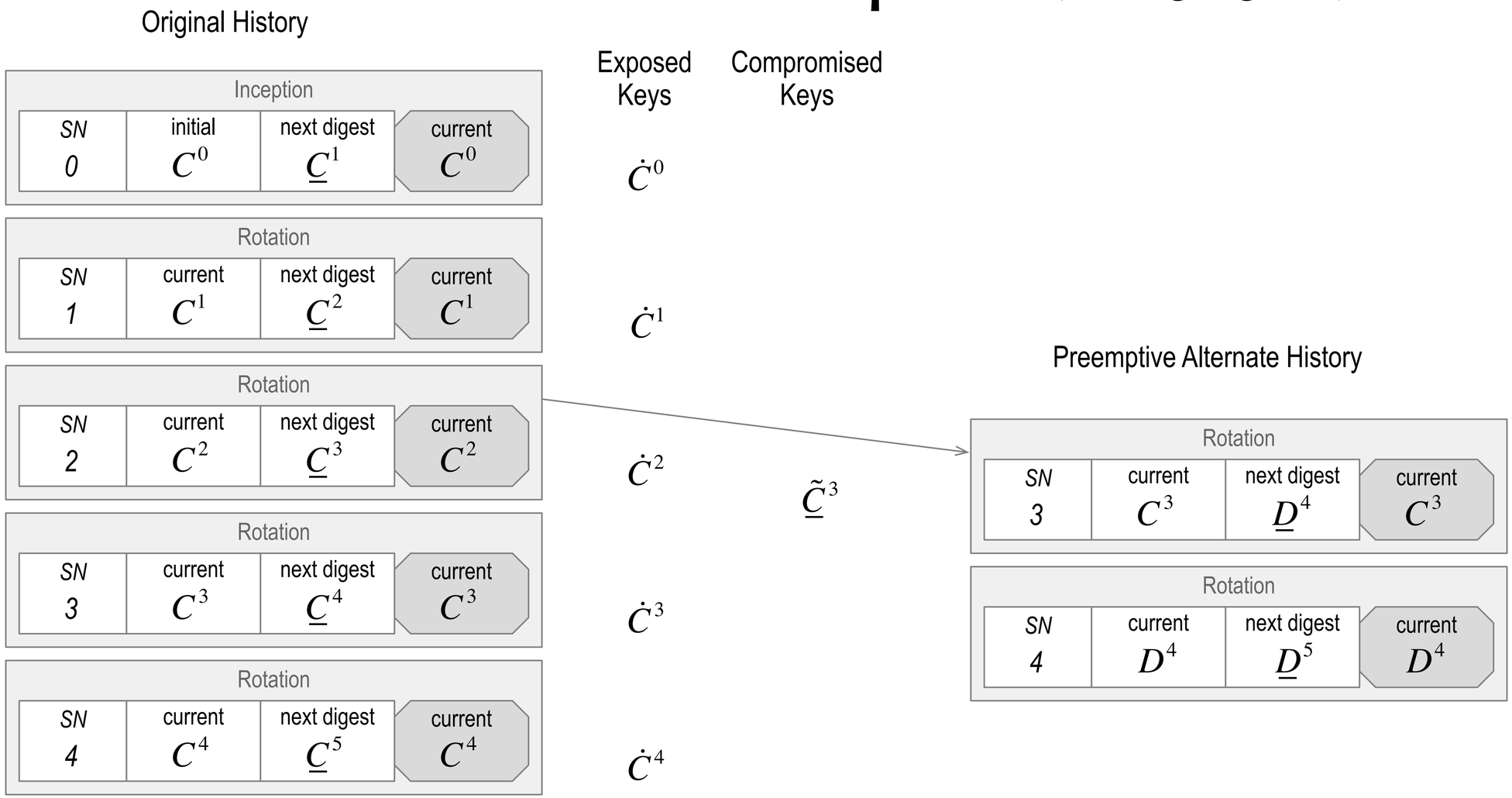
Hard Problem:

Recovery from Live Exploit of Current Signing Keys



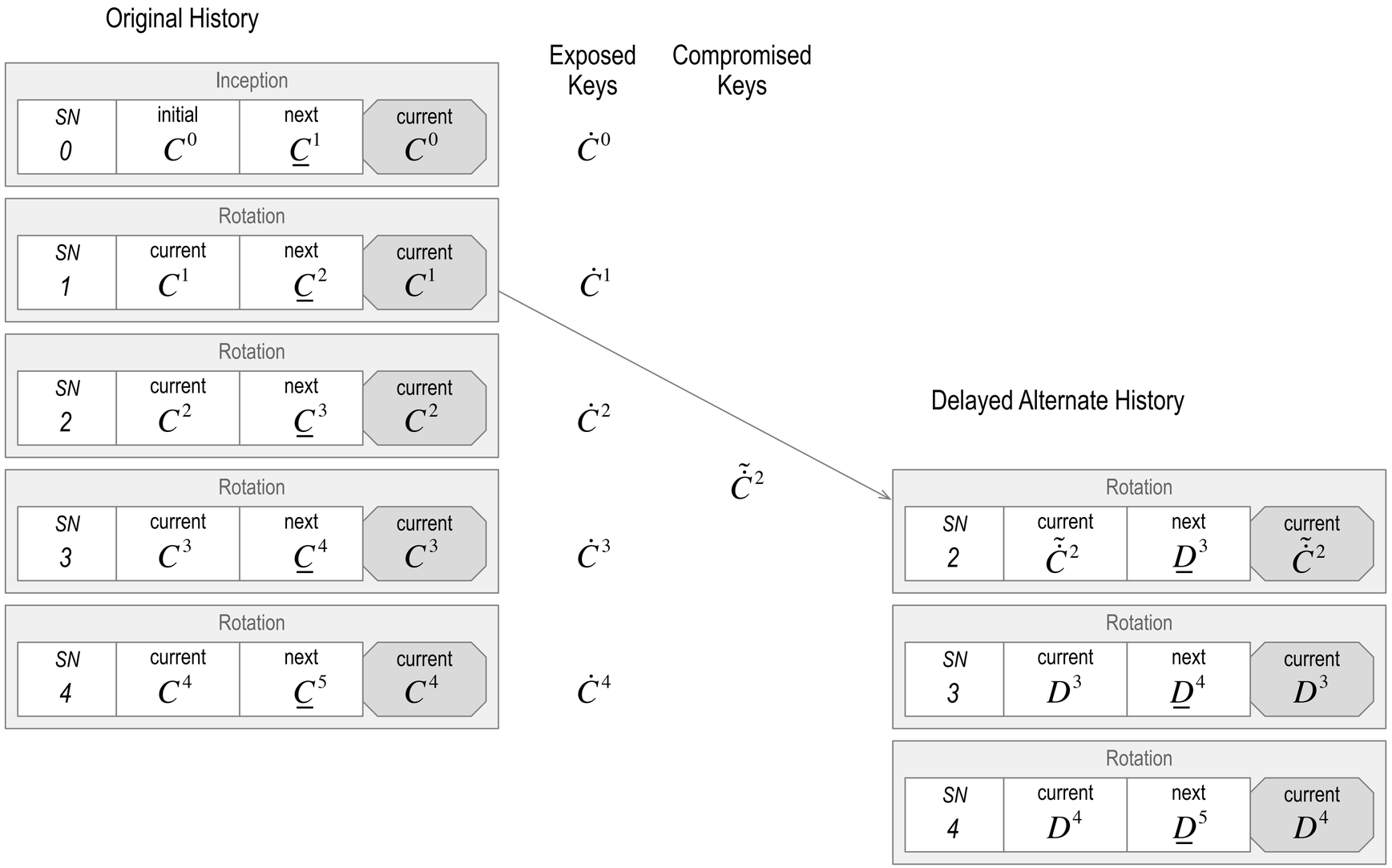
Pre-rotation provides protection from successful *live* exploit of current signing keys.

Live Exploit (next signing keys)



Difficulty of inverting *next* key(s) protects against successful *live* exploit.

Dead Exploit (stale next signing keys)



Any copy of original history protects against successful *dead* exploit: First Seen Wins

Witnessing Rules

An honest witness will only *witness*, (i.e. create, store, and promulgate a receipt for), at most *one and only one version* of any event.
That event version must first be verified.
A verified event version must be signed by the controller's authoritative keys as determined by prior events.
A verified event version must be consistent with all prior events.

Agreement

A *state of agreement* about a version of an event is defined with respect to *set* of witnesses in agreement:
Each witness in that *set* has witnessed the same version of that event and each receipt in that set has been promulgated to every other witness in that *set*.
The size of an agreement is the number of contributing witnesses in the *set*.
The associated *agreement* include a receipt from each witness in the *set*.
This state of agreement is provable to any validator, watcher, juror, or judge via a verifiable fully receipted copy of the event i.e the *agreement*.
This copy provides *proof of agreement*.
Such a proof may be obtained via any verifiable KERL that includes that version of that event.

Definitions

$N =$ number of witness

$M =$ size of agreement

$F =$ faulty witnesses

V *Validator*

C *Controller*

Threshold of Accountable Duplicity

TOAD

$\boldsymbol{M}_C$

$\boldsymbol{M}_V$

Sufficient Agreement

Controller's Guarantee

$$\boldsymbol{M} \geq \boldsymbol{M}_C$$

Validator's Choice

$$\boldsymbol{M}_V \geq \boldsymbol{M}_C$$

Algorithm Objectives

Any pre-existing copy or digest of original KERL available to Validator protects Validator from future dead exploits.

KAACE provides fault tolerance from live exploit.

A successful but recoverable live exploit is a compromise of the controller's current signing keys and/or a compromise of the witnesses' signing keys.

A) WRT Controller, a successful live exploit of the witnesses' would prevent sufficient agreement thereby inducing a recovery operation.

B) WRT Validator, a successful live exploit would produce undetectably duplicitous but sufficient agreement about current events.

(KAACE immune agreement prevents this, i.e. Validator is immune)

Detectable vs Undetectable Duplicity

Witness Duplicity

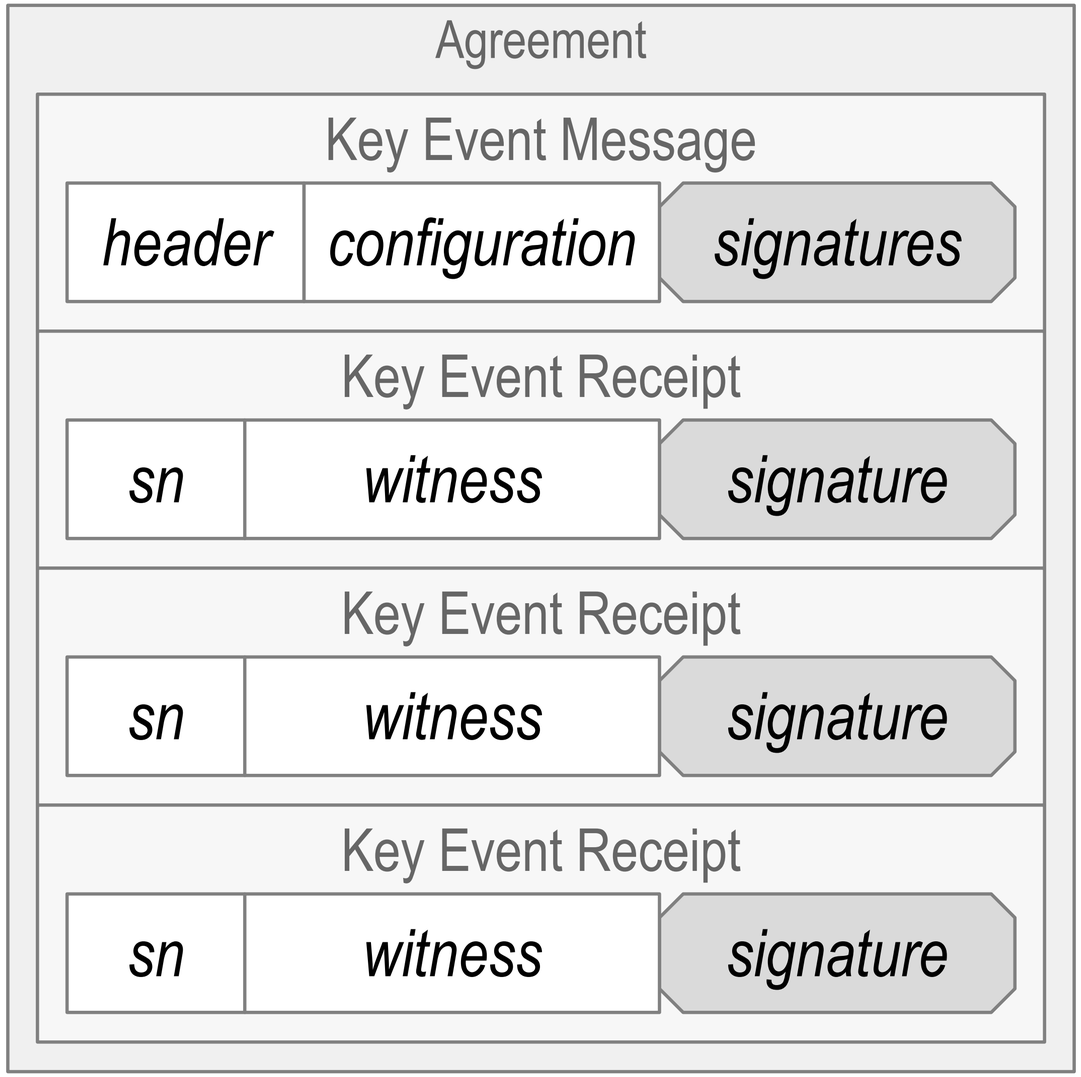
Witness Duplicity is Detectable.

Controller Duplicity

Controller Duplicity wrt witnesses is undetectable if a sufficient number of witnesses are not duplicitous and sufficient agreement is small enough.

(KA²CE)
Keri's Agreement Algorithm for Control Establishment

Produce Witnessed Agreements with Guarantees



Agreement Constraints

N = number of witness

M = size of agreement

F = faulty witnesses

V *Validator*

C *Controller*

Proper Agreement

$$F + 1$$

Bounds on Sufficient Agreement

$$M > F$$

$$M \leq N - F$$

$$F < M \leq N - F$$

Intact Agreement

$$N \geq 2F + 1$$

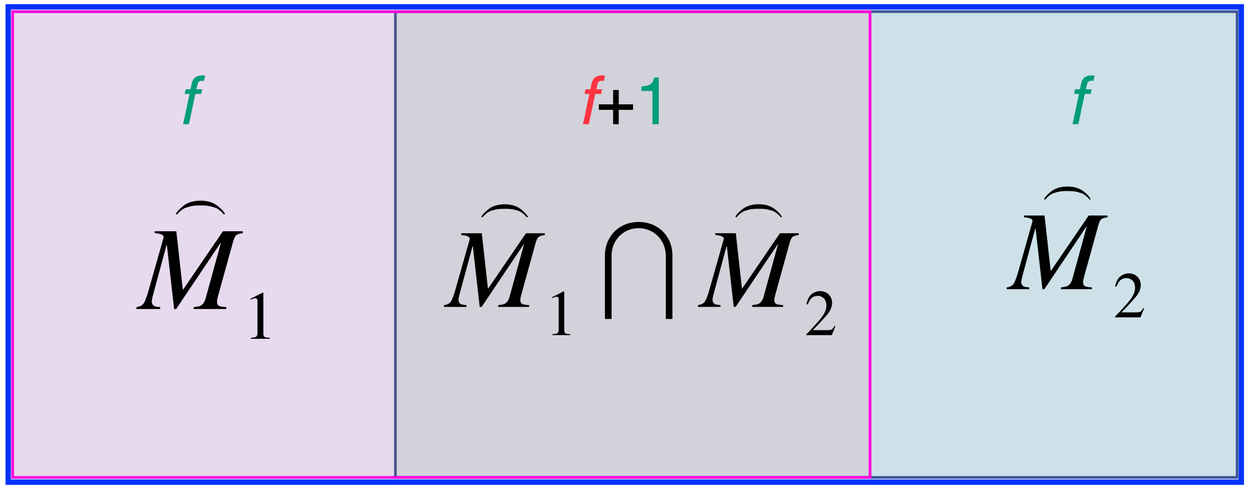
One Agreement or None at All (*Immune*)

first seen rule limits liveness induces recovery rotation

$$\left|\widehat{N}\right| = N \qquad \left|\widehat{M}_1\right| = \left|\widehat{M}_2\right| = M$$

Overlapping Sets

$$\widehat{M}_1 \bigcup \widehat{M}_2 = \widehat{N}$$



One honest witness if:

$$\left|\widehat{M}_1 \bigcap \widehat{M}_2\right| \geq F+1$$

$$\left|\widehat{M}_1 \bigcup \widehat{M}_2\right| = \left|\widehat{N}\right| = N$$

$$\left|\widehat{M}_1\right| + \left|\widehat{M}_2\right| = \left|\widehat{M}_1 \bigcup \widehat{M}_2\right| + \left|\widehat{M}_1 \bigcap \widehat{M}_2\right|$$

$$2M = N + F + 1$$

$$M \geq \left\lceil \frac{N+F+1}{2} \right\rceil$$

$$M \leq N - F$$

Immune Agreement

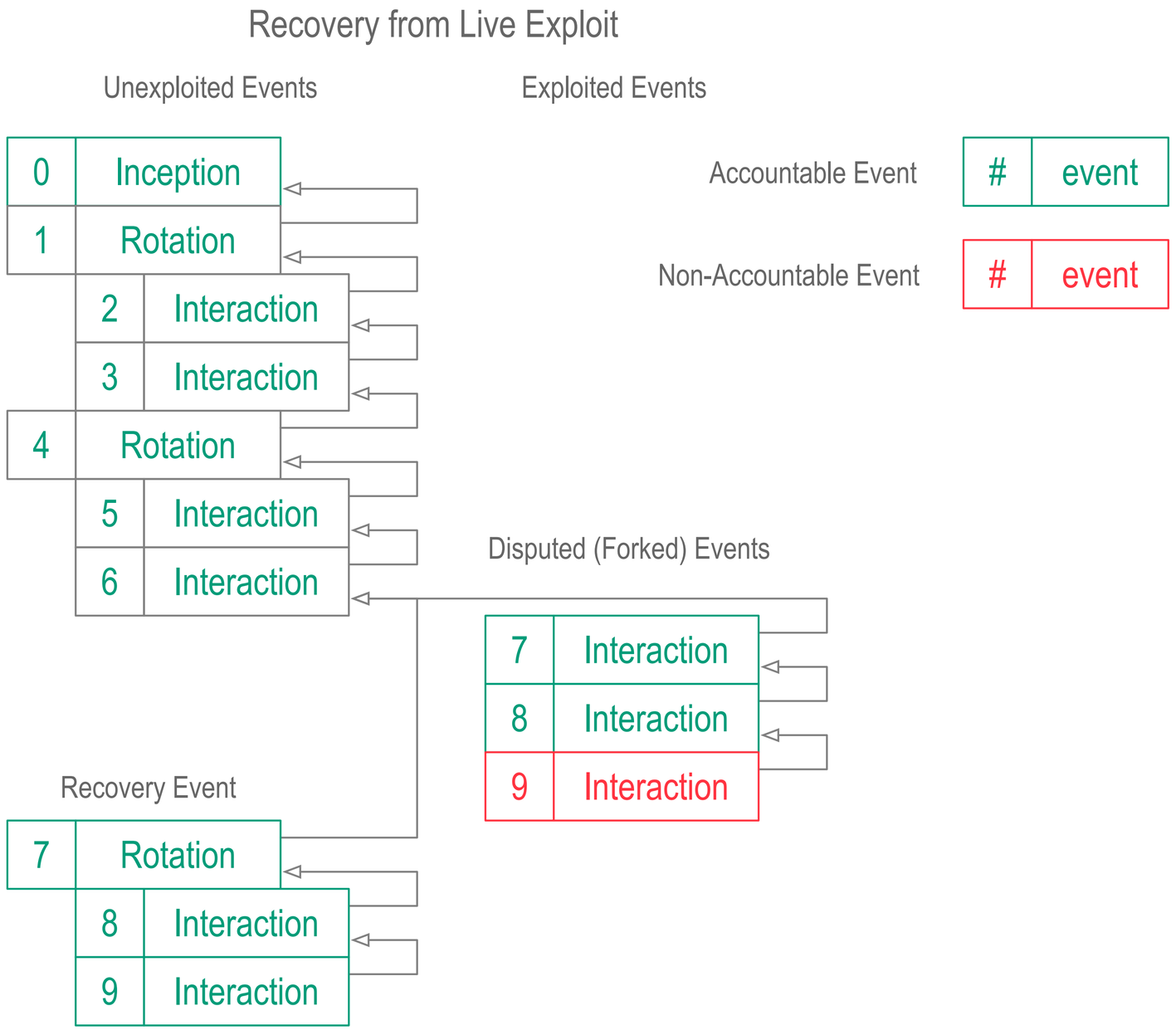
$$\frac{N+F+1}{2} \leq M \leq N - F$$

Example Values

Immunity

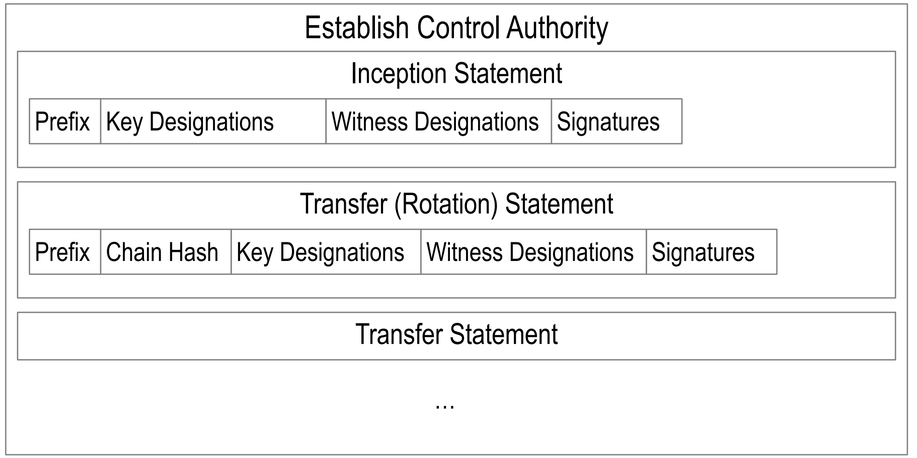
F	N	3F+1	$\left\lceil \frac{N+F+1}{2} \right\rceil$	N-F	M
1	4	4	3	3	3
1	5	4	4	4	4
1	6	4	4	5	4, 5
1	7	4	5	6	5, 6
1	8	4	5	7	5, 6, 7
1	9	4	6	8	6, 7, 8
2	7	7	5	5	5
2	8	7	6	6	6
2	9	7	6	7	6, 7
2	10	7	7	8	7, 8
2	11	7	7	9	7, 8, 9
2	12	7	8	10	8, 9, 10
3	10	10	7	7	7
3	11	10	8	8	8
3	12	10	8	9	8, 9
3	13	10	9	10	9, 10
3	14	10	9	11	9, 10, 11
3	15	10	10	12	10, 11, 12

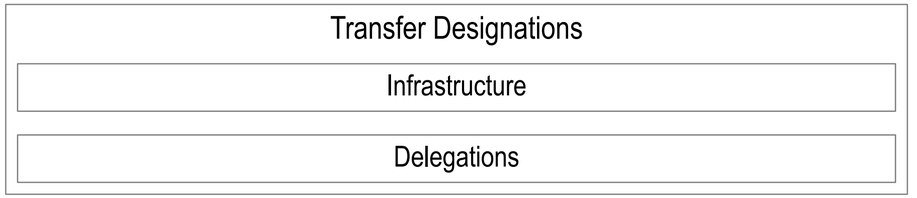
Recovery from Live Exploit Of Current Signing Keys



Function Stack

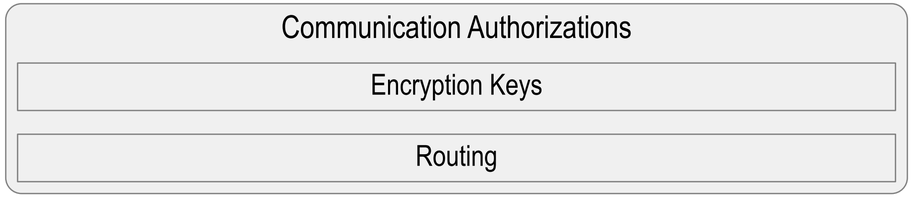
KERI





Authorizations after Establishment

On Top of KERI

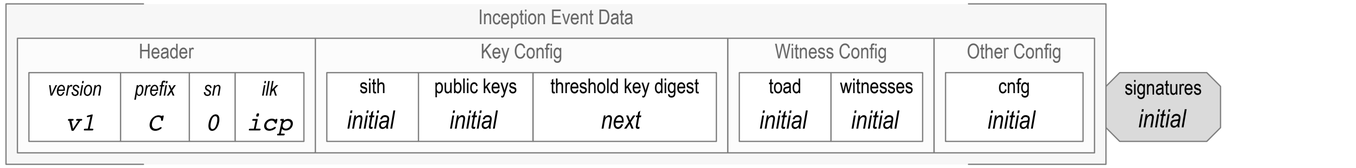


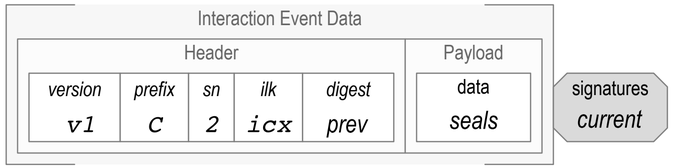
Design follows the *Hourglass Model* of a stack of thin layers

Rotate Prefix vs Rotate Keys

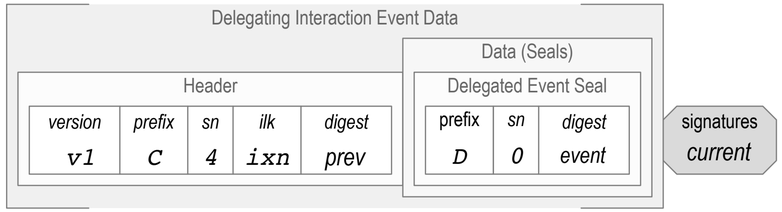
Non-transferable may not rotate keys. May only rotate prefix
Rotate prefix good for bootstrapping. No key event log (KEL) needed.
If prefix has no persistent value outside its function and its function may be marshaled by some other prefix controller then rotating prefix may be preferred.

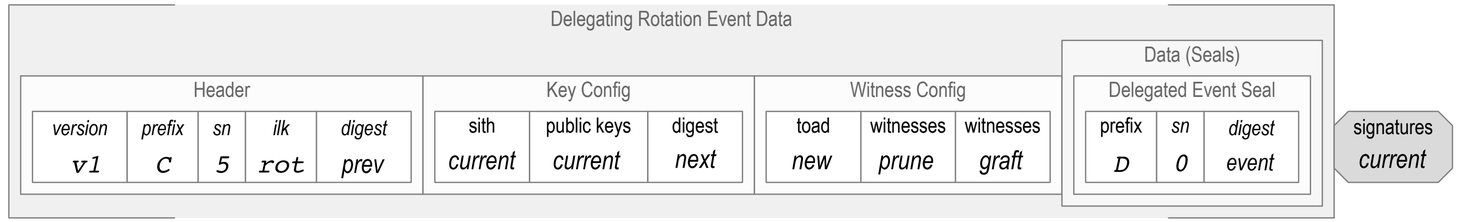
Events



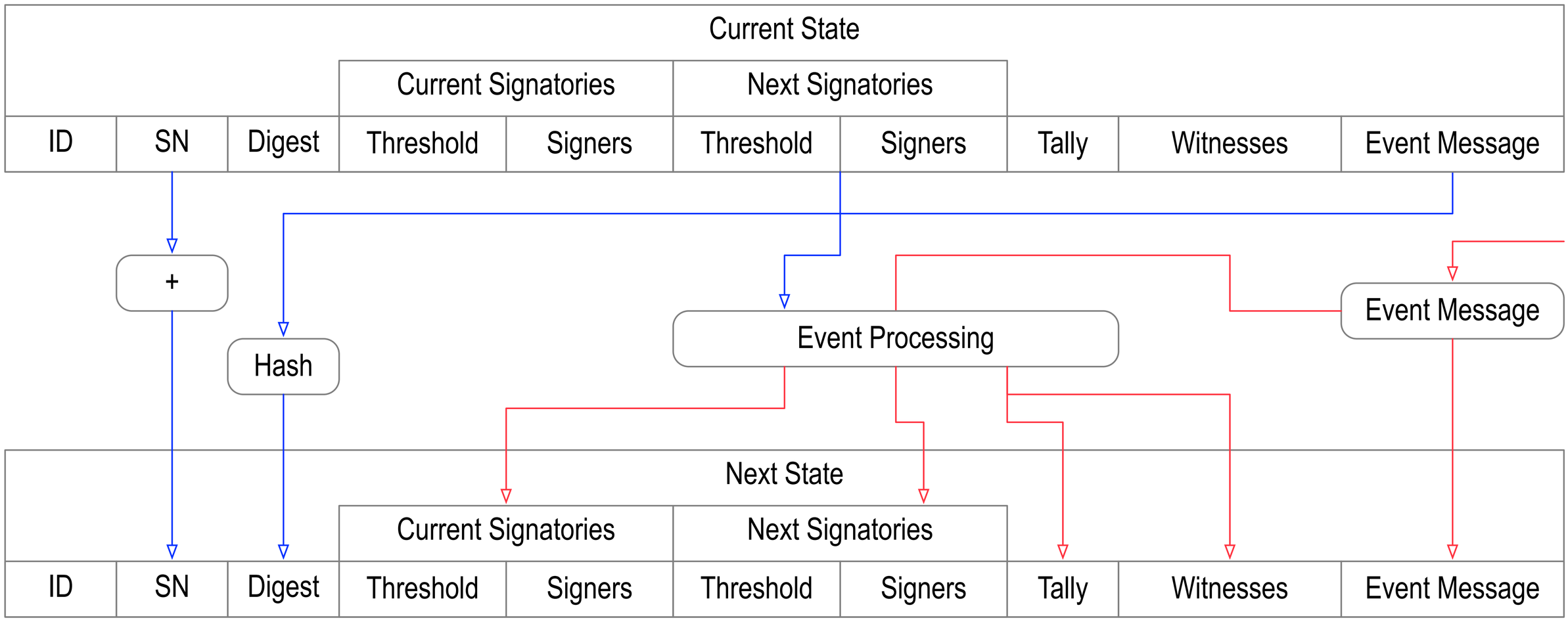


Delegating Events

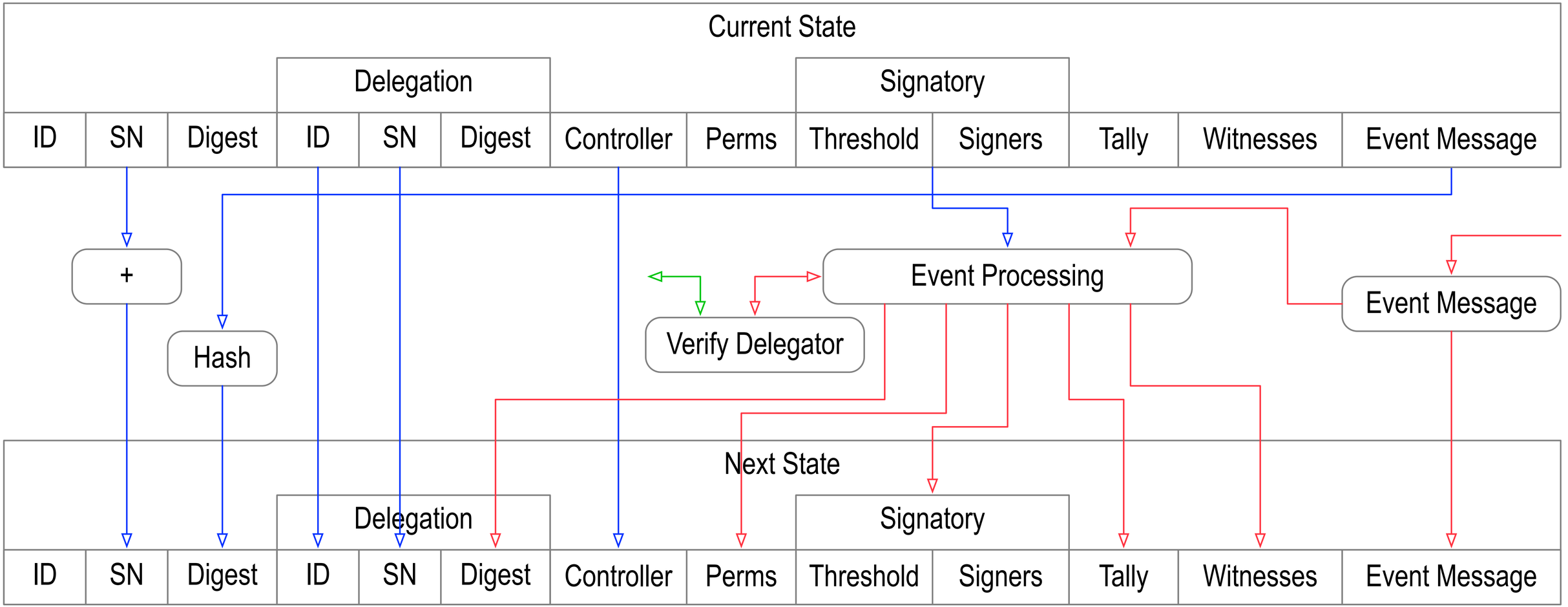




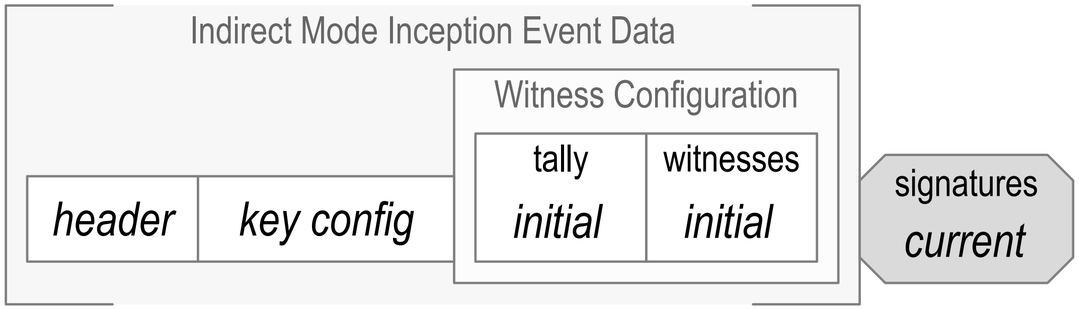
State Verifier Engine

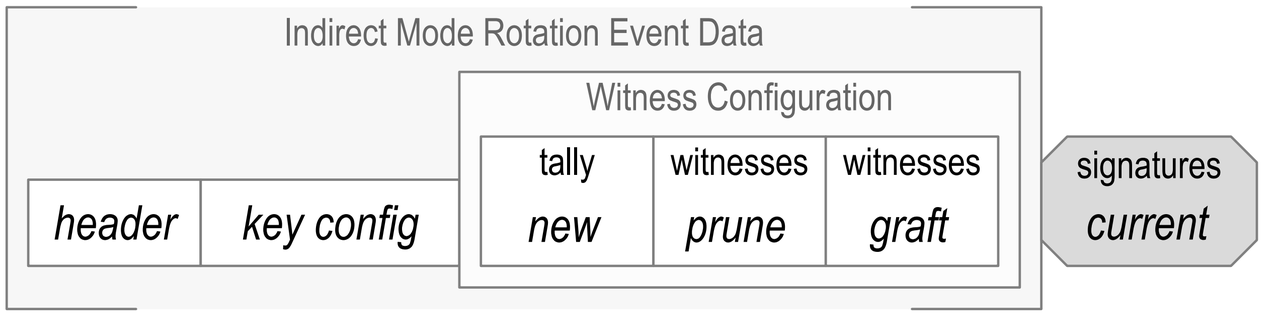


Delegated State Verifier Engine



Witness Designation



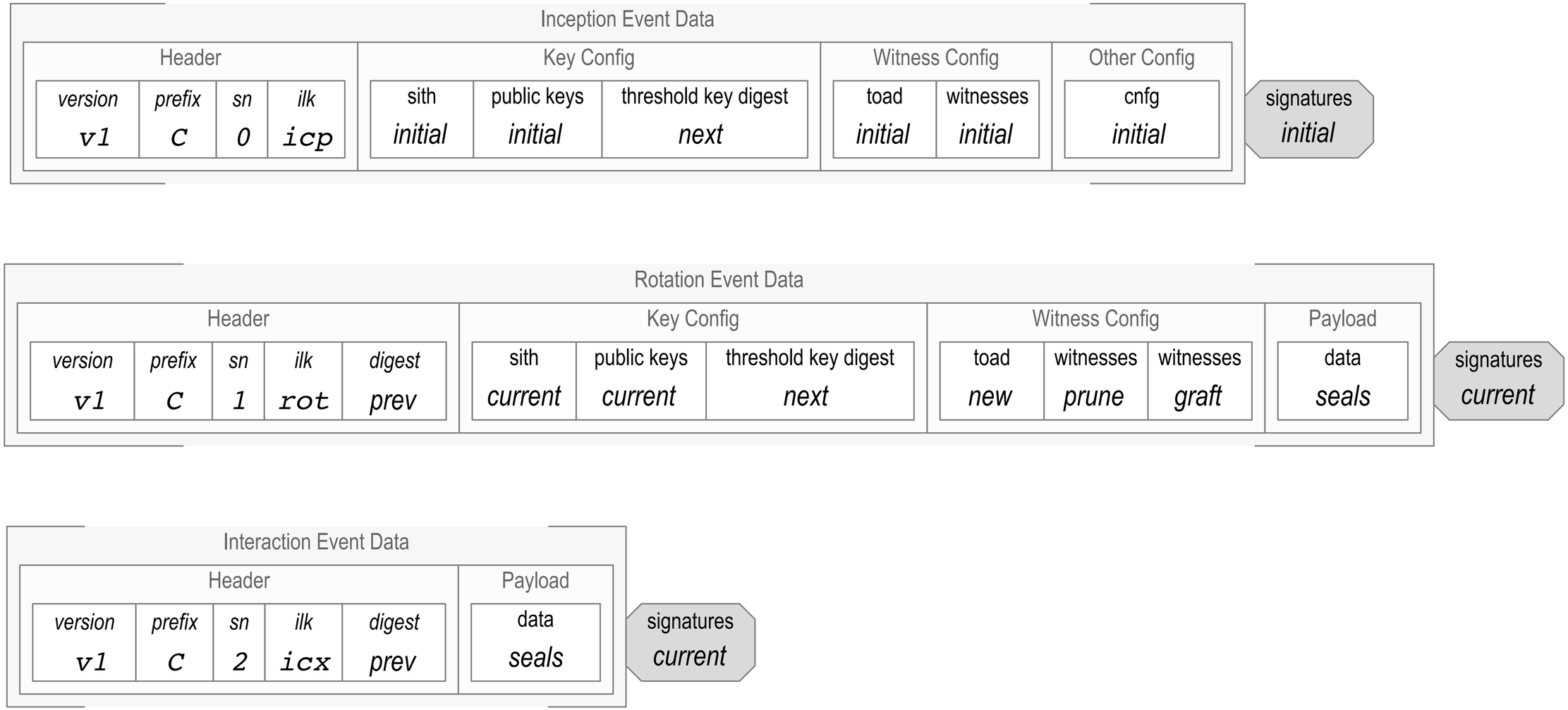


Witnessed Key Event Receipt

$$\left\langle \varepsilon_k^C \right\rangle W_0^C \sigma_{W_0^C}^C$$

$$\left\langle \varepsilon_k^C \right\rangle W_0^C \sigma_{W_0^C}^C W_1^C \sigma_{W_1^C}^C$$

Generic Event Formats



Generic Inception

$$\varepsilon_0^C = \left\langle \nu_0^C, C, t_0^C, \texttt{icp}, K_0^C, \widehat{C}_0^C, \eta_0^C\left(\left\langle K_1^C, \widehat{C}_1^C \right\rangle\right), M_0^C, \widehat{W}_0^C, [cnfg] \right\rangle \widehat{\sigma}_0^C$$

$$\widehat{C}_0^C = \left[C^0, \ldots, C^{L_0^C - 1} \right]_0^C$$

$$\widehat{C}_1^C = \left[C^{r_1}, \ldots, C^{r_1 + L_1^C - 1} \right]_1^C$$

$$\widehat{W}_0^C = \left[W_0^C, \ldots, W_{N_0^C - 1}^C \right]_0^C$$

$$\widehat{\sigma}_0^C = \sigma_{C^{s0}} \ldots \sigma_{C^{s S_0^C - 1}}$$

Generic Rotation

$$\varepsilon_k^C = \left\langle \nu_k^C, C, t_k^C, \eta_k^C\left(\varepsilon_{k-1}^C\right), \texttt{rot}, K_l^C, \widehat{C}_l^C, \eta_l^C\left(\left\langle K_{l+1}^C, \widehat{C}_{l+1}^C \right\rangle\right), M_l^C, \widehat{X}_l^C, \widehat{Y}_l^C, [seals] \right\rangle \widehat{\sigma}_{kl}^C$$

$$\widehat{C}_l^C = \left[C^{r_l^C}, \ldots, C^{r_l^C + L_l^C - 1} \right]_l^C$$

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

$$\widehat{X}_l^C = \left[X_0^C, \ldots, X_{O_l^C - 1}^C \right]_l^C$$

$$\widehat{Y}_l^C = \left[Y_0^C, \ldots, Y_{P_l^C - 1}^C \right]_l^C$$

$$\widehat{\sigma}_{kl}^C = \sigma_{C^{r_l^C + s_0}} \ldots \sigma_{C^{r_l^C + s_{S_{kl}^C - 1}}}$$

Generic Interaction

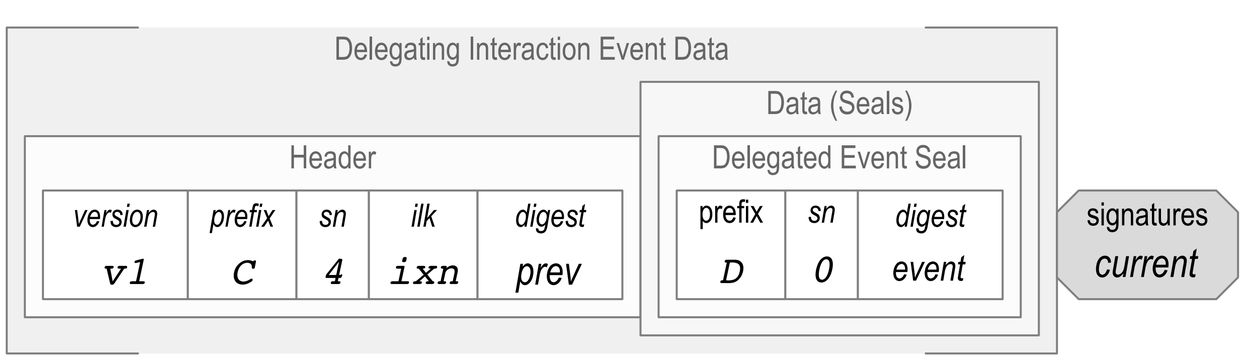
$$\varepsilon_k^C = \left\langle \nu_k^C, C, t_k^C, \eta_k^C\left(\varepsilon_{k-1}^C\right), \texttt{ixn}, [seals] \right\rangle \widehat{\sigma}_{kl}^C$$

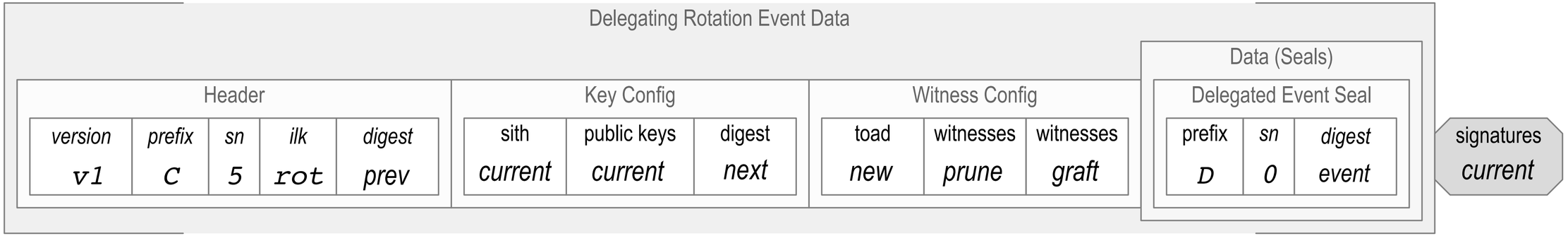
$$K_l^C$$

$$\widehat{C}_l^C = \left[C^{r_l^C}, \ldots, C^{r_l^C + L_l^C - 1} \right]_l^C$$

$$\widehat{\sigma}_{kl}^C = \sigma_{C^{r_l^C + s_0}} \ldots \sigma_{C^{r_l^C + s_{S_{kl}^C - 1}}}$$

Generic Delegating Event Formats





Generic Delegated Event Formats

Delegating Event Location Seal

prefix	sn	ilk	prior digest

Delegated Inception Event Data from Interaction

Header				Key Config			Witness Config		Other Config	Delegating Location Seal				signatures
version	prefix	sn	ilk	sith	public keys	digest	toad	witnesses	cfg	prefix	sn	ilk	digest	
v1	D	0	dip	initial	initial	next	initial	initial	traits	C	5	ixn	prior event	initial

Delegated Rotation Event Data from Interaction

Header					Key Config			Witness Config			Seals Config	Delegating Location Seal				signatures
version	prefix	sn	ilk	digest	sith	public keys	digest	toad	witnesses	witnesses	data	prefix	sn	ilk	digest	
v1	D	1	drt	prev	current	current	next	new	prune	graft	seals	C	5	ixn	prior event	current

Delegated Interaction Event Data

Header					Payload Config	
version	prefix	sn	ilk	digest	data	signatures
v1	D	2	isn	prev	seals	current

Inception Delegation

$$\widehat{\Delta}_0^D = \left\{ D, t_0^D, \eta_k^C \left(\varepsilon_0^D \right) \right\}$$ Delegated Event Seal

$$\varepsilon_0^D = \left\langle \nu_0^D, D, t_0^D, \texttt{dip}, K_0^D, \widehat{D}_0^D, M_0^D, \widehat{W}_0^D, \left[cnfg \right], \widehat{\Delta}_k^C \right\rangle \widehat{\sigma}_0^D$$

$$\widehat{D}_0^D = \left[D^0, \ldots, D^{L_0^D - 1} \right]_0^D$$

$$\widehat{W}_0^C = \left[W_0^C, \ldots, W_{N_0^C - 1}^C \right]_0^C$$

$$\widehat{\Delta}_k^C = \left\{ C, t_k^C, ilk, \eta_k^C \left(\varepsilon_{k-1}^C \right) \right\}$$ Delegating Event Location Seal

$$\widehat{\sigma}_0^D = \sigma_{D^{s0}} \ldots \sigma_{D^{s S_0^D - 1}}$$

Rotation Delegation

$$\widehat{\Delta}_k^D = \left\{ D, t_k^D, \eta_k^C \left(\varepsilon_k^D \right) \right\}$$ Delegated Event Seal

$$\varepsilon_k^D = \left\langle \nu_k^D, D, t_k^D, \eta_k^D \left(\varepsilon_{k-1}^D \right), \mathtt{drt}, K_l^D, \widehat{D}_l^D, M_l^D, \widehat{X}_l^D, \widehat{Y}_l^D, \left[seals \right], \widehat{\Delta}_k^C \right\rangle \widehat{\sigma}_{kl}^D$$

$$\widehat{D}_l^D = \left[D^{r_l^D}, \ldots, D^{r_l^D + L_l^D - 1} \right]_l^D$$

$$\widehat{X}_l^D = \left[X_0^D, \ldots, X_{O_l^D - 1}^D \right]_l^D$$

$$\widehat{Y}_l^D = \left[Y_0^D, \ldots, Y_{P_l^D - 1}^D \right]_l^D$$

$$\widehat{\Delta}_k^C = \left\{ C, t_k^C, ilk, \eta_k^C \left(\varepsilon_{k-1}^C \right) \right\}$$ Delegating Event Location Seal

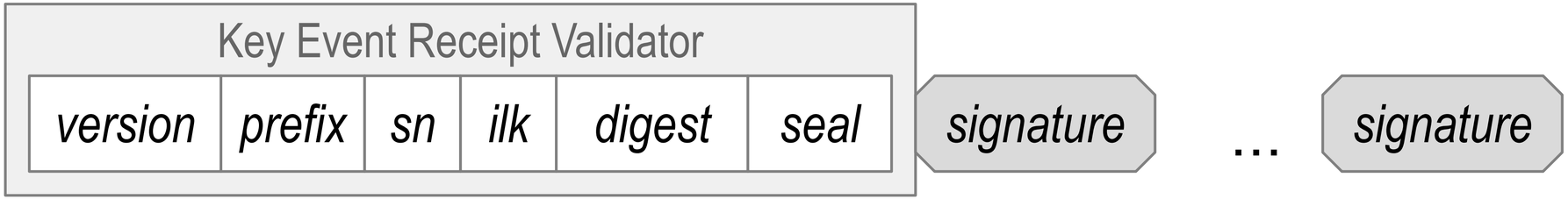
$$\widehat{\sigma}_{kl} = \sigma_{C^{+r_l^D + s_0}} \ldots \sigma_{C^{r_l^D + s_{S_{kl}^D - 1}}}$$

Delegated Interaction

$$\varepsilon_k^D = \left\langle \nu_k^D, D, t_k^D, \eta_k^D\left(\varepsilon_{k-1}^D\right), \texttt{ixn}, \left[seals\right] \right\rangle \widehat{\sigma}_{kl}^D$$

Receipt Messages

$$\rho^{C}_{\tilde{W}^{C}_{ls}}\left(\varepsilon^{C}_{k}\right)=\left\langle \nu^{C}_{k},C,t^{C}_{k},\texttt{rct},\eta^{C}_{k}\left(\varepsilon^{C}_{k}\right)\right\rangle W^{C}_{l0}\sigma^{C}_{W^{C}_{l0}},\ldots,W^{C}_{lN^{C}_{s}-1}\sigma^{C}_{W^{C}_{lN_{s}-1}}$$



$$\rho^{C}_{V}\left(\varepsilon^{C}_{k}\right)=\left\langle \nu^{C}_{k},C,t^{C}_{k},\texttt{vrc},\eta^{C}_{k}\left(\varepsilon^{C}_{k}\right),\widehat{\Delta}^{V}_{k}\right\rangle \widehat{\sigma}^{C}_{V_{l}}$$

$$\widehat{\Delta}^{V}_{k}=\left\{V,\eta^{V}_{k}\left(\varepsilon^{V}_{k}\right)\right\}$$

Witness Rotations

$$\widehat{W}_0 = \left[W_0 , W_1 , \cdots , W_{N-1} \right]$$

$$\widehat{W}_l = \left(\widehat{W}_{l-1} - \widehat{X}_l \right) \bigcap \widehat{Y}_l$$

$$\widehat{X}_l \subseteq \widehat{W}_{l-1} \quad \widehat{Y}_l \not\subset \widehat{W}_{l-1} \quad \widehat{X}_l \not\subset \widehat{W}_l$$

$$N_l = N_{l-1} - O_l + P_l$$

$$M_l \leq N_l$$

$$\left| \widehat{X}_l \right| = O_l \quad \left| \widehat{Y}_l \right| = P_l \quad \left| \widehat{W}_l \right| = N_l$$

$$\widehat{U}_{l-1} \subseteq \widehat{W}_{l-1} \qquad \left| \widehat{U}_{l-1} \right| \geq M_{l-1}$$

$$\widehat{U}_l \subseteq \widehat{W}_l \qquad \left| \widehat{U}_l \right| \geq M_l$$

$$\left| \widehat{U}_{l-1} \bigcup \widehat{U}_l \right| \leq M_{l-1} + M_l$$

Complex Weighted Signing Thresholds

$$\widehat{C}_l = \left[C_l^1, \ldots, C_l^{L_l} \right]_l$$

$$\widehat{K}_l = \left[U_l^1, \ldots, U_l^{L_1} \right]_l$$

$$0 < U_l^j \leq 1$$

$$\widehat{s}_k^l = \left[s_0, \ldots, s_{S_k^l - 1} \right]_k^l$$

$$\bar{U}_l = \sum_{i=s_0}^{s_{S_k-1}} U_l^i \geq 1$$

$$\widehat{C} = \left[C^1, C^2, C^3 \right]$$

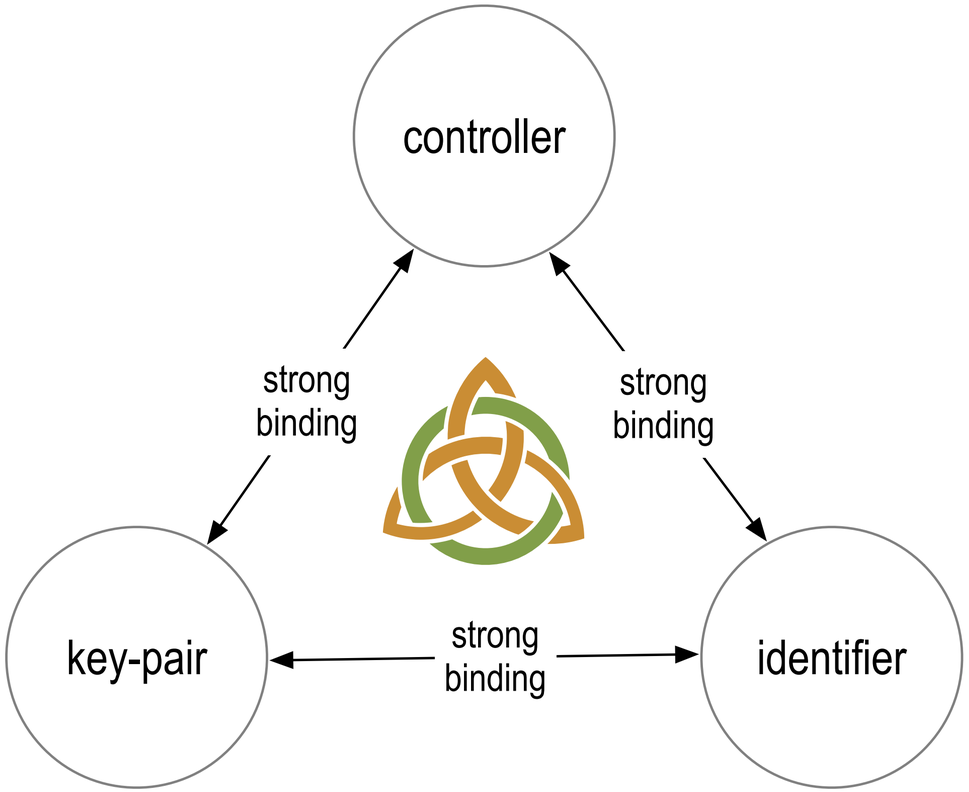
$$U_l^j = 1 / K_l$$

$$\widehat{K} = \left[1/2, 1/2, 1/2 \right]$$

$$\widehat{K}_l = \left[1/2, 1/2, 1/4, 1/4, 1/4, 1/4 \right]_l$$

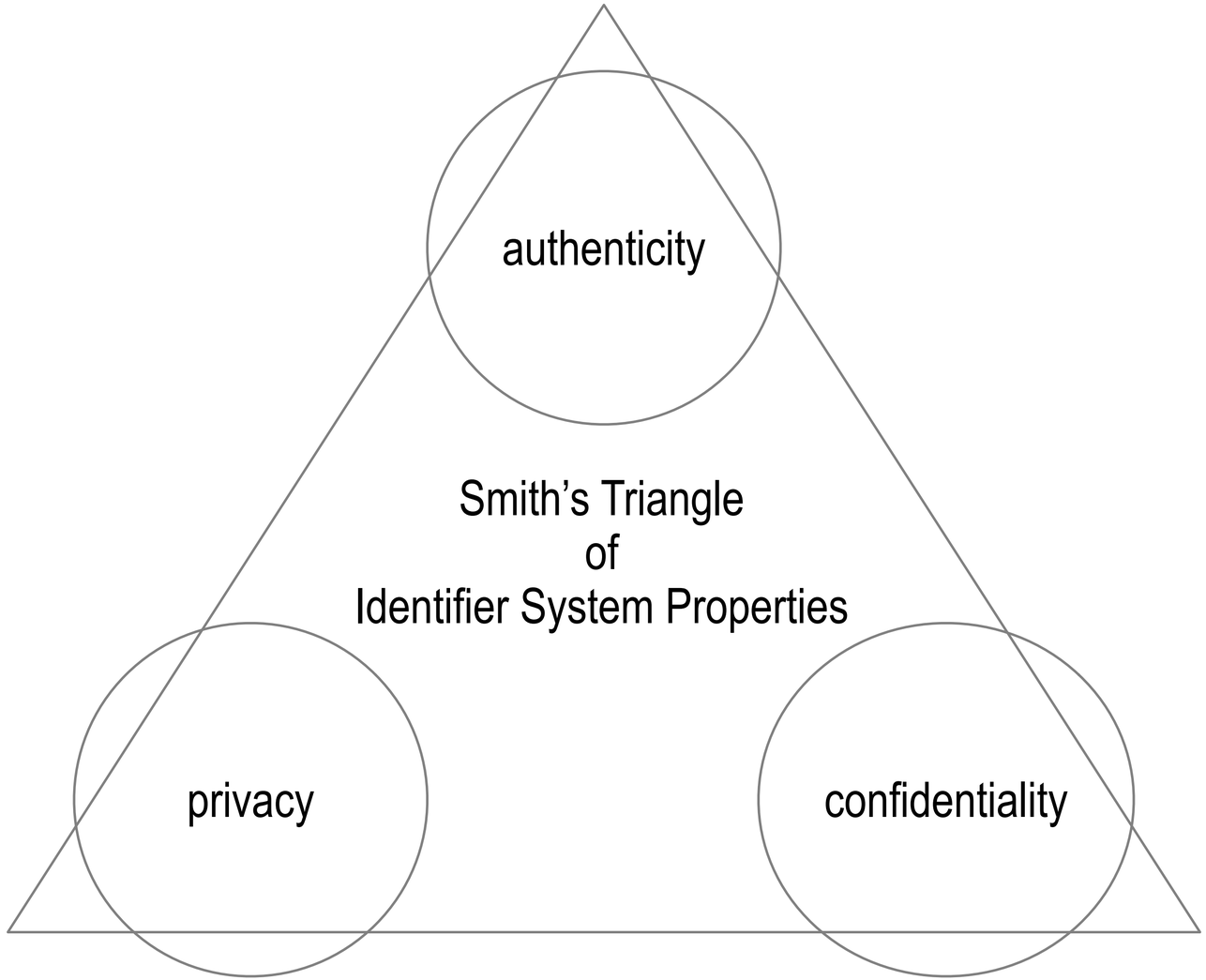
$$\widehat{K}_l = \left[\left[1/2, 1/2, 1/4, 1/4, 1/4, 1/4 \right], \left[1/2, 1/2, 1/2, 1/2 \right], \left[1, 1, 1, 1 \right] \right]$$

BACKGROUND



KERI

Smith’s Identifier System Properties Triangle

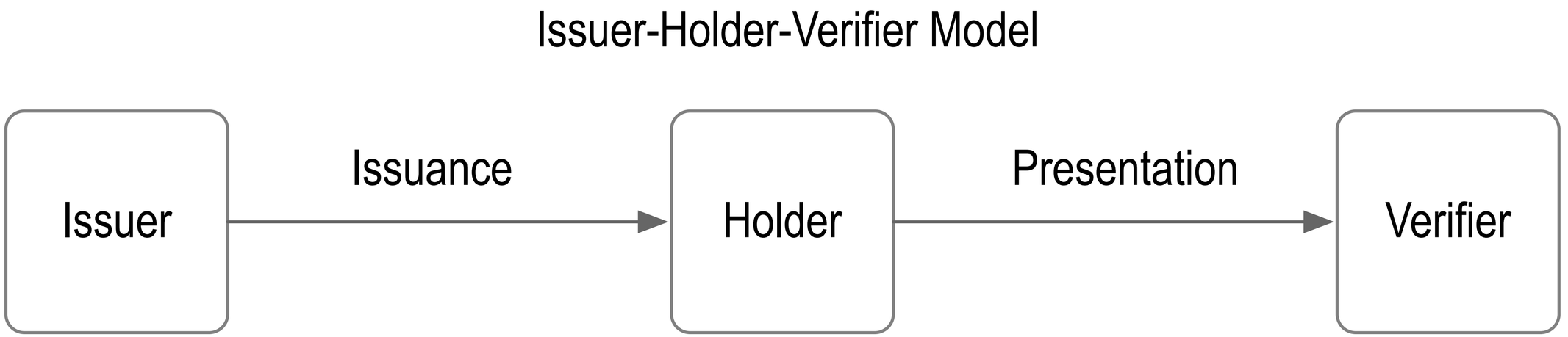


May exhibit any two at the highest level but not all three at the highest level

Tripartite Authentic Data (VC) Model

Issuer: Source of the VC. Creates (issues) and signs VC
Holder: Usually the target of the VC. The holder is the "*issuee*" that receives the VC and holds it for its own use.
Verifier: Verifies the signatures on the VC and authenticates the holder at the time of presentation

The issuer and target each have a DID (decentralized identifier).
The DIDs are used to look-up the public key(s) needed to verify signatures.

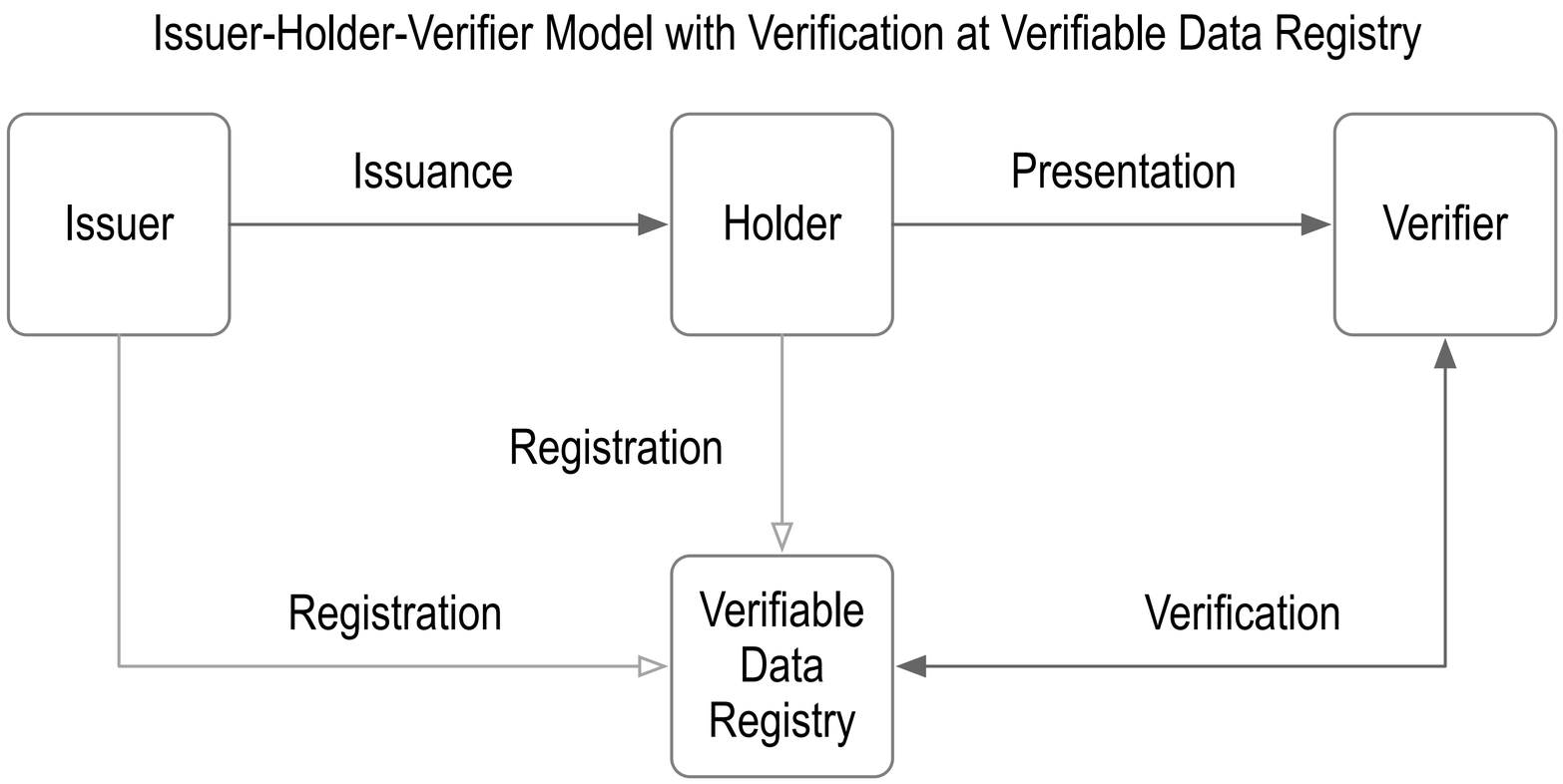


Tripartite Authentic Data (VC) Model with VDR

Verifiable Data Registry (VDR) enables decentralized but interoperable discovery and verification of authoritative key pairs for DIDs in order to verify the signatures on VCs. A VDR may also provide other information such as data schema or revocation state of a VC.

Each controller of a DID registers that DID on a VDR so that a verifier can determine the authoritative key pairs for any signatures.

We call this determination, *establishment of control authority* over a DID.

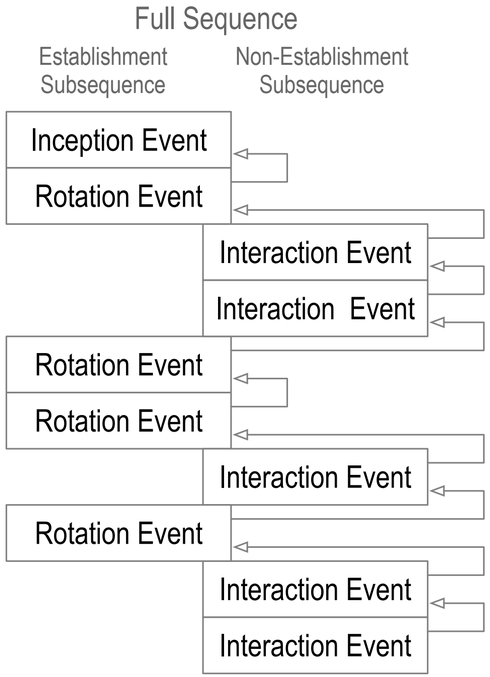


KERI VDRs vs. Shared Ledger VDRs

Most DID methods use a shared ledger (commonly referred to as a *blockchain*) for their VDR. Typically, in order to interoperate all participants must use the same shared ledger or support multiple different DID methods. There are currently over 70 DID methods. Instead GLEIF has chosen to use KERI based DID methods. KERI stands for Key Event Receipt Infrastructure. KERI based VDRs are ledger independent, i.e. not locked to a given ledger. This provides a path for greater interoperability without forcing participants in the vLEI ecosystem to use the same shared ledger.

A KERI VDR is called a key event log (KEL). It is a cryptographically verifiable hash chained data structure. Each KERI based identifier has its own dedicated KEL. The purpose of the KEL is to provide proof of the establishment of control authority over an identifier. This provides cryptographically verifiable proof of the current set of authoritative keys for the identifier. KERI identifiers are long cryptographic pseudo random strings of characters. They are self-certifying and self-managing.

A KERI identifier is abstractly called an Autonomic Identifier (AID) because it is self-certifying and self-managing. A KERI DID is one concrete implementation of a KERI AID. The same KERI prefix may control multiple different DIDs as long as they share the same prefix.



`did:keri:`*`prefix`*`[:`*`options`*`][/`*`path`*`][?`*`query`*`][#`*`fragment`*`]`

`did:keri:ENqFtH6_cfDg8riLZ-GDvDaCKVn6clOJa7ZXXVXSWpRY`

KERI Identifier KEL VDR *Controls* Verifiable Credential Registry TEL VDR

A KERI KEL for a given identifier provides proof of authoritative key state at each event. The events are ordered. This ordering may be used to order transactions on some other VDR such as a Verifiable Credential Registry by attaching anchoring seals to the KEL events.

The set of transactions that determine registry state form a log called a Transaction Event Log (TEL).

Transactions are signed with the authoritative keys determined by the key state in the KEL with the transaction seal.

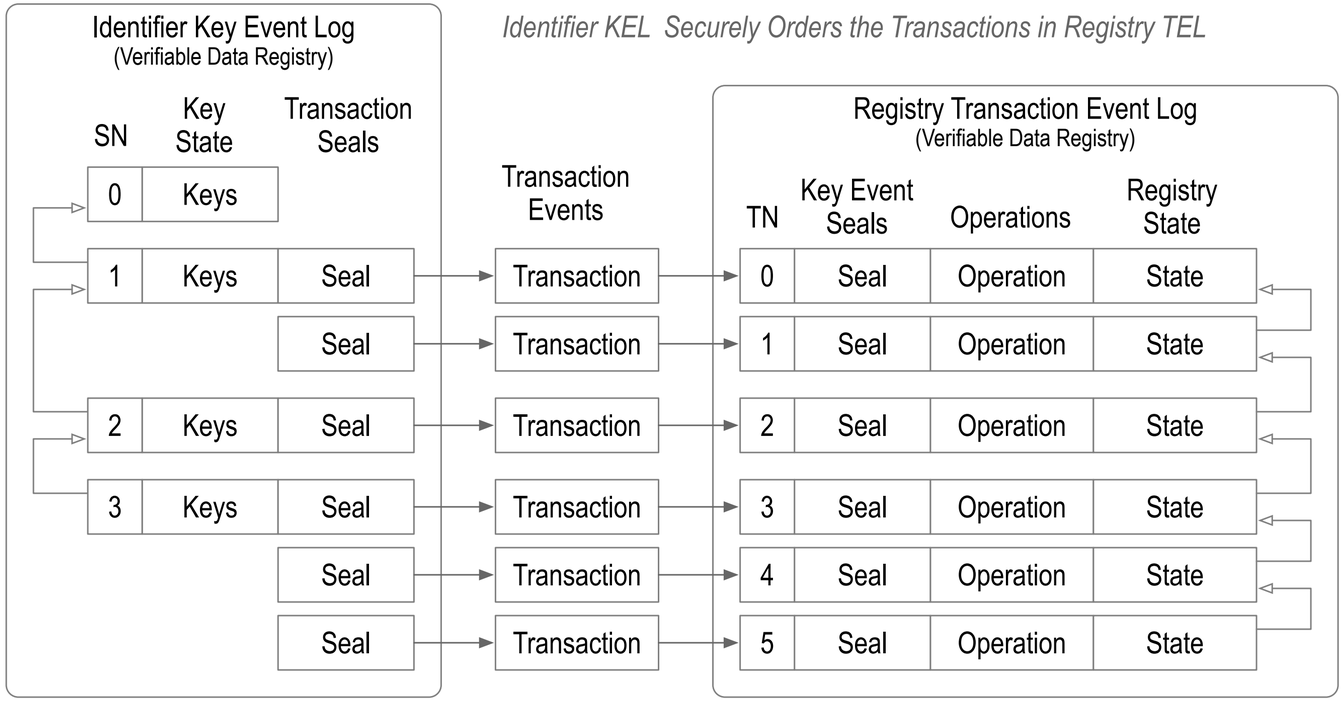
The transactions likewise contain a reference seal back to the key event authorizing the transaction.

This setup enables a KEL to control a TEL for any purpose.

In the case of the vLEI the TEL controls a vLEI issuance and revocation registry.

The TEL provides a cryptographic proof of registry state by reference to the corresponding controlling KEL.

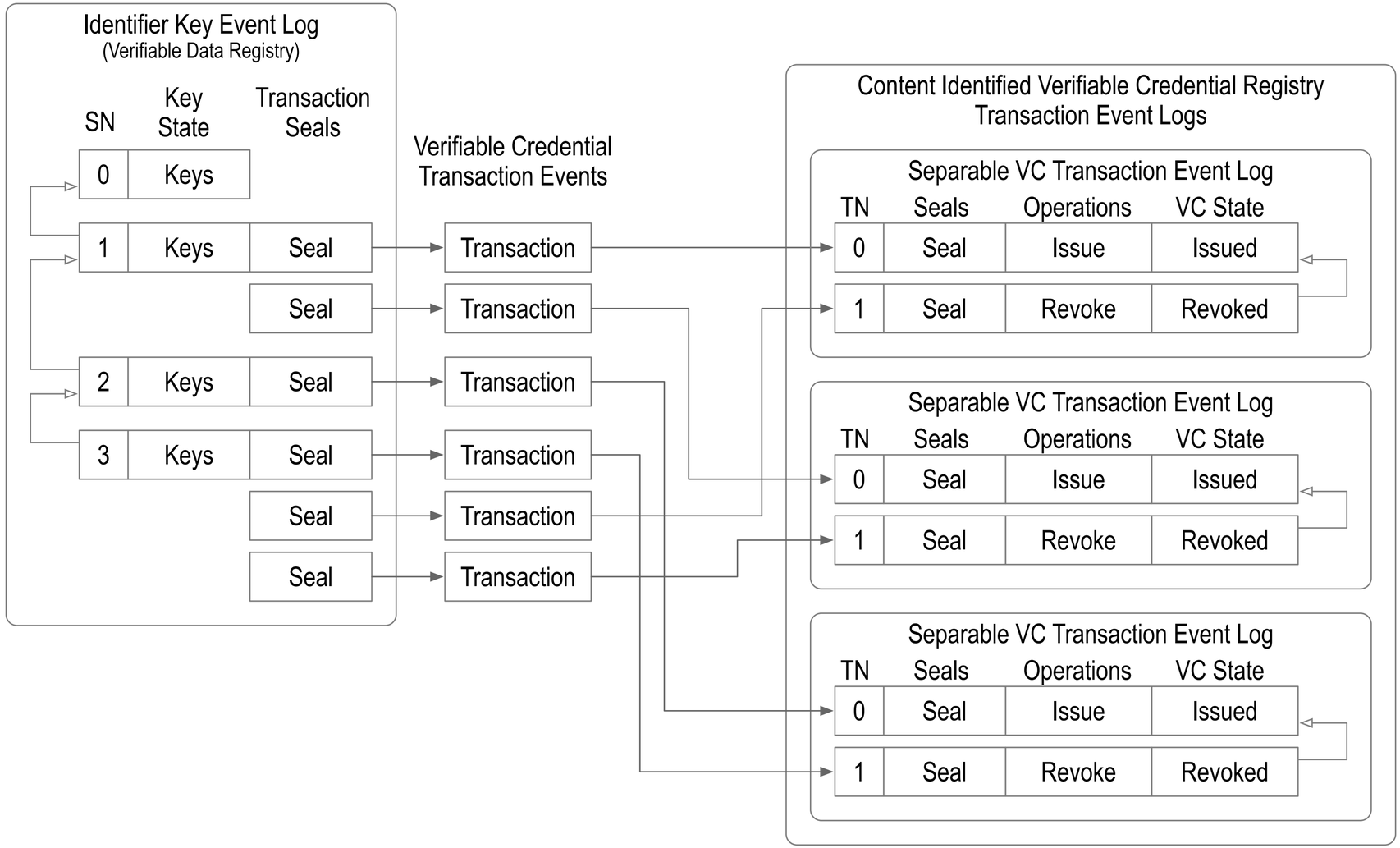
Any validator may therefore cryptographically verify the authoritative state of the registry.



seal = proof of authenticity

Registry with Separable VC Issuance-Revocation TELs

Each VC may be uniquely identified with a content digest.
Each VC also has a uniquely identified issuer using a KERI AID.
This combination enables a separable registry of VC issuance-revocation state.
The state may employ a cryptographic accumulator for enhanced privacy



Identifier System Security

Authentic transmission of data may be verified using an identity system security overlay.
This overlay maps cryptographic key-pairs to identifiers.
When those identifiers are self-certifying they are derived via cryptographic one-way functions from the key pairs.
This provides a self-certifying identifier with a cryptographic root-of-trust.
A key event log (KEL) provide support for secure key rotation without changing the identifier.
Message authenticity is provided by verifying signatures to the authoritative keys pairs for the identifier included in the message.

Cryptographic Material Derivation Code Tables

Length of crypt material determines number of pad characters. One character table for one pad char. Two character table for two pad char.

One Character KERI Base64 Prefix Derivation Code Selector

Derivation Code	Prefix Description
0	Two character derivation code. Use two character table.
1	Four character derivation code. Use four character table.
2	Five character derivation code. Use five character table.
3	Six character derivation code. Use six character table.
4	Eight character derivation code. Use eight character table.
5	Nine character derivation code. Use nine character table.
6	Ten character derivation code. Use ten character table.
-	Count code for attached receipts. Use receipt count code table(s)

Four Character KERI Base64 Count Code for Attached Receipt Couplets

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Count Code Length	Qual Length Base64	Code Length Bytes
-A*XX*	Count of Attached Qualified Base64 Receipt Couplets	0	0	4	4	3
-B*XX*	Count of Attached Qualified Base2 Receipt Couplets	0	0	4	4	3

One Character KERI Base64 Prefix Derivation Code

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Derivation Code Length	Prefix Length Base64	Prefix Length Bytes
A	Non-transferable prefix using Ed25519 public signing verification key. Basic derivation.	32	1	1	44	33
B	X25519 public encryption key. May be converted from Ed25519 public signing verification key.	32	1	1	44	33
C	Ed25519 public signing verification key. Basic derivation.	32	1	1	44	33
D	Blake3-256 Digest. Self-addressing derivation.	32	1	1	44	33
E	Blake2b-256 Digest. Self-addressing derivation.	32	1	1	44	33
F	Blake2s-256 Digest. Self-addressing derivation.	32	1	1	44	33
G	Non-transferable prefix using ECDSA secp256k1 public singing verification key. Basic derivation.	32	1	1	44	33
H	ECDSA secp256k1 public signing verification key. Basic derivation.	32	1	1	44	33
I	SHA3-256 Digest. Self-addressing derivation.	32	1	1	44	33
J	SHA2-256 Digest. Self-addressing derivation.	32	1	1	44	33

Two Character KERI Base64 Prefix Derivation Code

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Derivation Code Length	Prefix Length Base64	Prefix Length Bytes
0A	Ed25519 signature. Self-signing derivation.	64	2	2	88	66
0B	ECDSA secp256k1 signature. Self-signing derivation.	64	2	2	88	66
0C	Blake3-512 Digest. Self-addressing derivation.	64	2	2	88	66
0D	SHA3-512 Digest. Self-addressing derivation.	64	2	2	88	66
0E	Blake2b-512 Digest. Self-addressing derivation.	64	2	2	88	66
0F	SHA2-512 Digest. Self-addressing derivation.	64	2	2	88	66

Attached Signature Derivation Code Tables

Length of crypt material determines number of pad characters. One character table for one pad char. Two character table for two pad char.

Two Character KERI Base64 Attached Signature Selection Code

Derivation Code	Selector Description	Data Length Bytes	Pad Length	Derivation Code Length	Prefix Length Base64	Prefix Length Bytes
0	Four character attached signature code. Use four character table					
1	Five character attached signature code. Use five character table					
2	Six character attached signature code. Use six character table					
-	Count code for attached signatures. Use attached signature count code table(s)					

Two Character KERI Base64 Attached Signature Derivation Code

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Derivation Code Length	Prefix Length Base64	Prefix Length Bytes
A*X*	Ed25519 signature	64	2	2	88	66
B*X*	ECDSA secp256k1 signature	64	2	2	88	66

Four Character KERI Base64 Attached Signature Derivation Code

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Derivation Code Length	Prefix Length Base64	Prefix Length Bytes
0A*XX*	Ed448 signature	114	0	4	156	117
OB*XX*						
0C*XX*						

Four Character KERI Base64 Count Code for Attached Signatures

Derivation Code	Prefix Description	Data Length Bytes	Pad Length	Count Code Length	Qual Length Base64	Code Length Bytes
-A*XX*	Count of Attached Qualified Base64 Signatures	0	0	4	4	3
-B*XX*	Count of Attached Qualified Base2 Signatures	0	0	4	4	3

Base64

Base64 Decode ASCII to Binary

Base64 Binary Decoding from ASCII

ASCII Char	Base64 Index Decimal	Base64 Index Hex	Base64 Index 6 bit Binary	ASCII Char	Base64 Index Decimal	Base64 Index Hex	Base64 Index 6 bit Binary	ASCII Char	Base64 Index Decimal	Base64 Index Hex	Base64 Index 6 bit Binary	ASCII Char	Base64 Index Decimal	Base64 Index Hex	Base64 Index 6 bit Binary
A	0	00	000000	Q	16	10	010000	g	32	20	100000	w	48	30	110000
B	1	01	000001	R	17	11	010001	h	33	21	100001	x	49	31	110001
C	2	02	000010	S	18	12	010010	i	34	22	100010	y	50	32	110010
D	3	03	000011	T	19	13	010011	j	35	23	100011	z	51	33	110011
E	4	04	000100	U	20	14	010100	k	36	24	100100	0	52	34	110100
F	5	05	000101	V	21	15	010101	l	37	25	100101	1	53	35	110101
G	6	06	000110	W	22	16	010110	m	38	26	100110	2	54	36	110110
H	7	07	000111	X	23	17	010111	n	39	27	100111	3	55	37	110111
I	8	08	001000	Y	24	18	011000	o	40	28	101000	4	56	38	111000
J	9	09	001001	Z	25	19	011001	p	41	29	101001	5	57	39	111001
K	10	0A	001010	a	26	1A	011010	q	42	2A	101010	6	58	3A	111010
L	11	0B	001011	b	27	1B	011011	r	43	2B	101011	7	59	3B	111011
M	12	0C	001100	c	28	1C	011100	s	44	2C	101100	8	60	3C	111100
N	13	0D	001101	d	29	1D	011101	t	45	2D	101101	9	61	3D	111101
O	14	0E	001110	e	30	1E	011110	u	46	2E	101110	-	62	3E	111110
P	15	0F	001111	f	31	1F	011111	v	47	2F	101111	_	63	3F	111111

Base64 Encode Binary to ASCII

Base64 Binary Encoding to ASCII

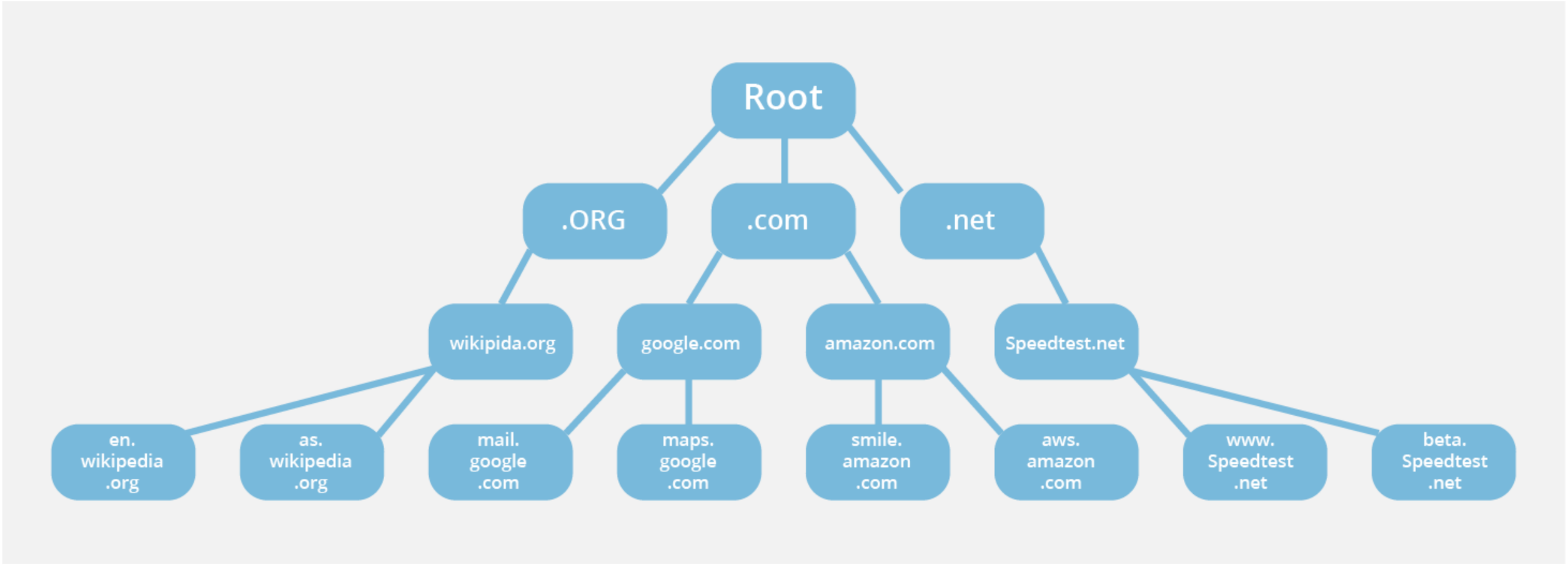
Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary
0	A	65	41	01000001	**16**	Q	81	51	01010001	**32**	g	103	67	01100111	**48**	w	119	77	01110111
1	B	66	42	01000010	**17**	R	82	52	01010010	**33**	h	104	68	01101000	**49**	x	120	78	01111000
2	C	67	43	01000011	**18**	S	83	53	01010011	**34**	i	105	69	01101001	**50**	y	121	79	01111001
3	D	68	44	01000100	**19**	T	84	54	01010100	**35**	j	106	6A	01101010	**51**	z	122	7A	01111010
4	E	69	45	01000101	**20**	U	85	55	01010101	**36**	k	107	6B	01101011	**52**	0	48	30	00110000
5	F	70	46	01000110	**21**	V	86	56	01010110	**37**	l	108	6C	01101100	**53**	1	49	31	00110001
6	G	71	47	01000111	**22**	W	87	57	01010111	**38**	m	109	6D	01101101	**54**	2	50	32	00110010
7	H	72	48	01001000	**23**	X	88	58	01011000	**39**	n	110	6E	01101110	**55**	3	51	33	00110011
8	I	73	49	01001001	**24**	Y	89	59	01011001	**40**	o	111	6F	01101111	**56**	4	52	34	00110100
9	J	74	4A	01001010	**25**	Z	90	5A	01011010	**41**	p	112	70	01110000	**57**	5	53	35	00110101
10	K	75	4B	01001011	**26**	a	97	61	01100001	**42**	q	113	71	01110001	**58**	6	54	36	00110110
11	L	76	4C	01001100	**27**	b	98	62	01100010	**43**	r	114	72	01110010	**59**	7	55	37	00110111
12	M	77	4D	01001101	**28**	c	99	63	01100011	**44**	s	115	73	01110011	**60**	8	56	38	00111000
13	N	78	4E	01001110	**29**	d	100	64	01100100	**45**	t	116	74	01110100	**61**	9	57	39	00111001
14	O	79	4F	01001111	**30**	e	101	65	01100101	**46**	u	117	75	01110101	**62**	-	45	2D	00101101
15	P	80	50	01010000	**31**	f	102	66	01100110	**47**	v	118	76	01110110	**63**	_	95	5F	01011111

Discovery

Ledger Based

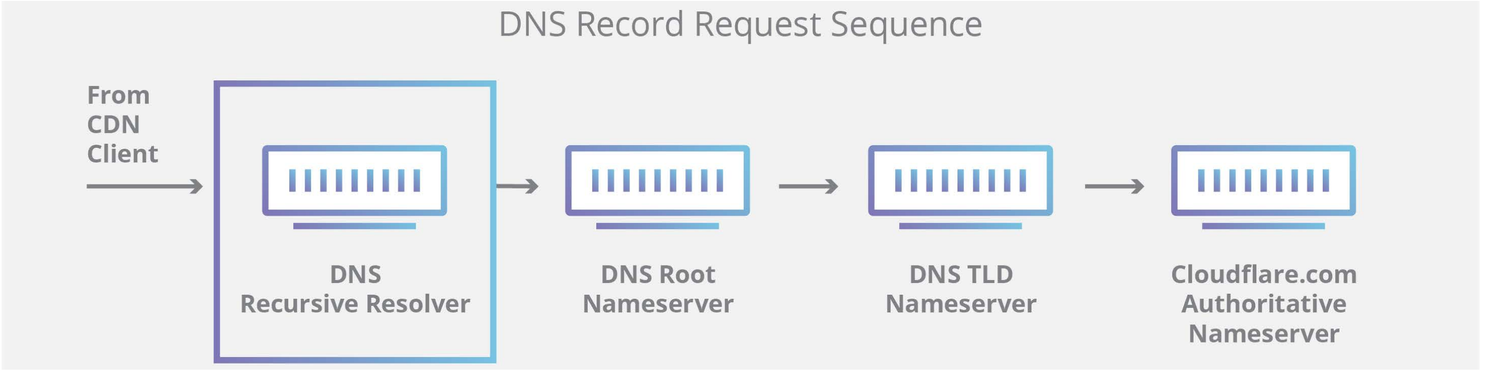
Non-Ledger Based

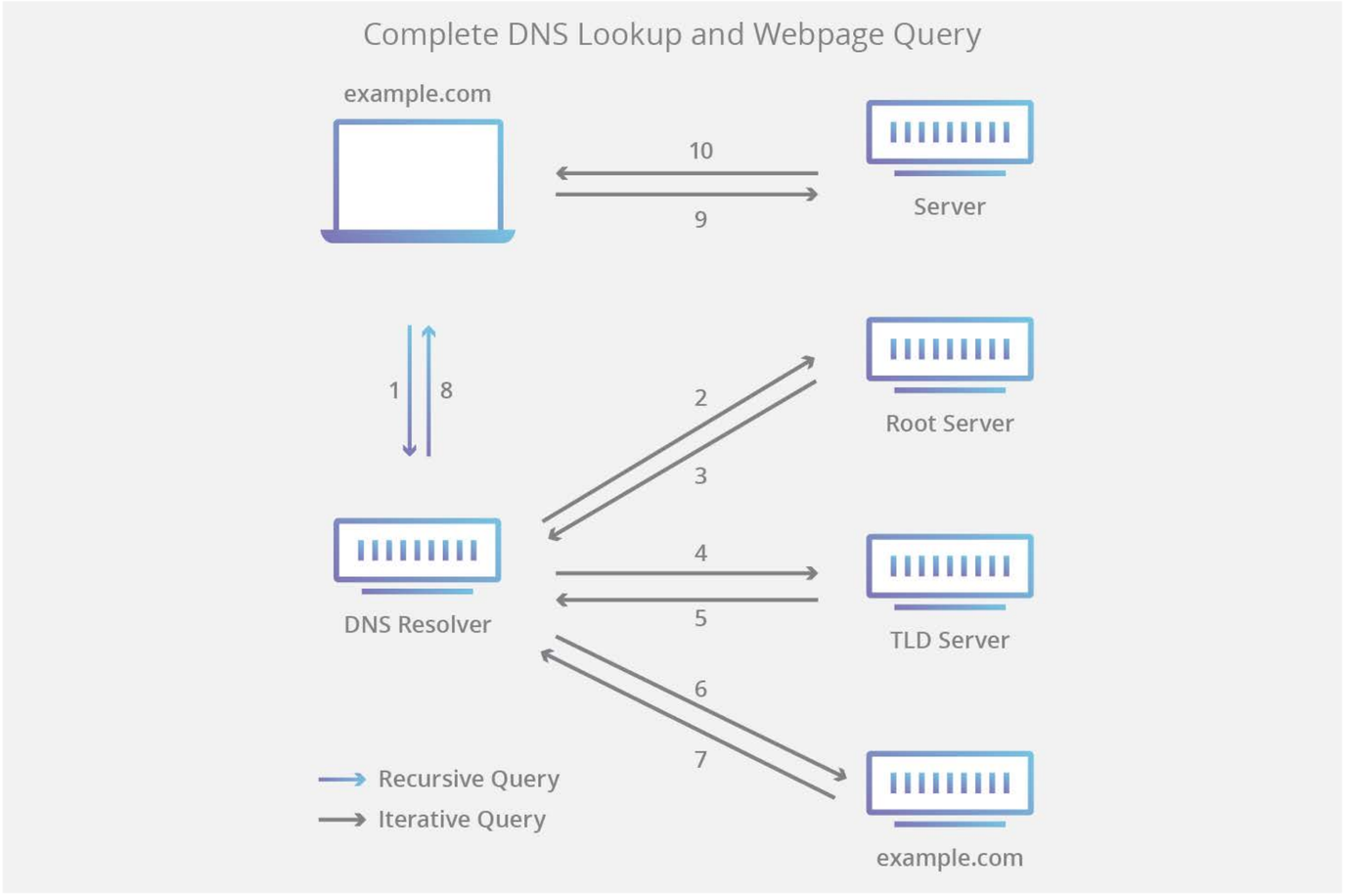
DNS “Hierarchical” Discovery



Diagrams from https://www.cloudflare.com/learning/dns/what-is-dns/

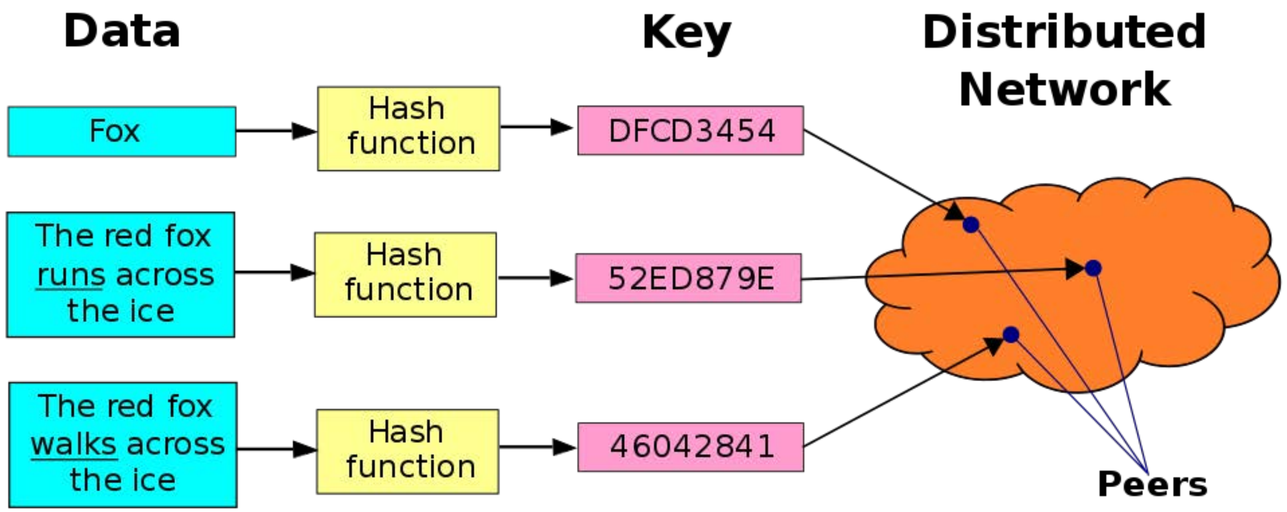
DNS “Hierarchical” Discovery





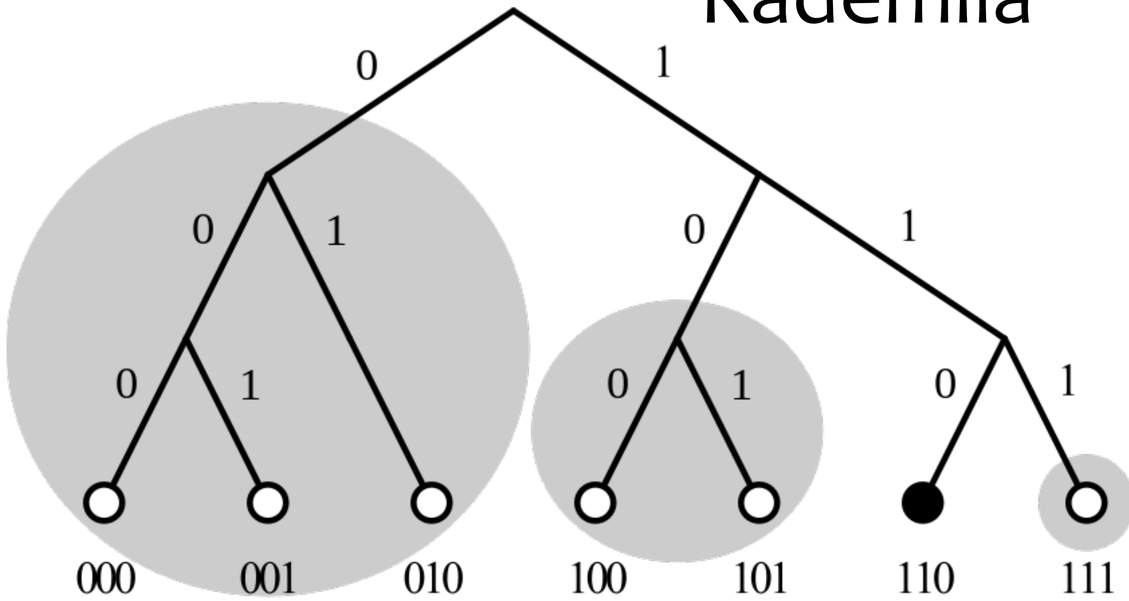

```
$ORIGIN example.com.
                        @               3600 SOA   ns1.p30.oraclecloud.net. (
                        zone-admin.dyndns.com.     ; address of responsible party
                        2016072701              ; serial number
                        3600                    ; refresh period
                        600                     ; retry period
                        604800                  ; expire time
                        1800                    ) ; minimum ttl
                        86400 NS    ns1.p68.dns.oraclecloud.net.
                        86400 NS    ns2.p68.dns.oraclecloud.net.
                        86400 NS    ns3.p68.dns.oraclecloud.net.
                        86400 NS    ns4.p68.dns.oraclecloud.net.
                        3600 MX     10 mail.example.com.
                        3600 MX     20 vpn.example.com.
                        3600 MX     30 mail.example.com.
                        60 A        204.13.248.106
                        3600 TXT    "v=spf1 includespf.oraclecloud.net ~all"
                        mail            14400 A     204.13.248.106
                        vpn               60 A      216.146.45.240
                        webapp            60 A      216.146.46.10
                        webapp            60 A      216.146.46.11
www                     43200 CNAME example.com.
```

DHT “Distributed” Discovery



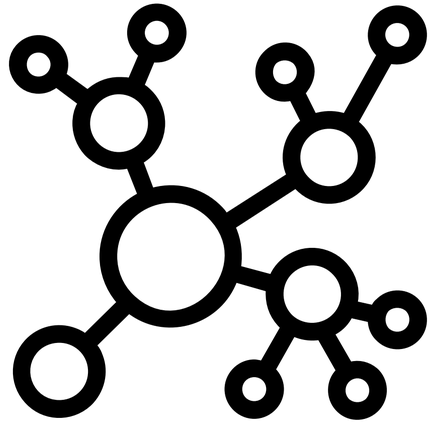
Diagrams from https://en.wikipedia.org/wiki/Distributed_hash_table

Kademlia



Diagrams from https://en.wikipedia.org/wiki/Kademlia

DHT Discovery for KERI



Lookup IP address of any DHT Node Prefix

bootstrap DHT access from any cache (≈ DNS Server cache)

Query DHT for Latest Rotation Event of Controller Prefix

-> Extract Witness Prefixes from Event (≈ CName Record)

Query DHT for IP Address of Witness Prefix (≈ A Record)

Query Witness for KERL of Controller Prefix

Query Watcher Network for Duplicitous KERL of Controller Prefix

Certificate Transparency Problem

“The solution the computer world has relied on for many years is to introduce into the system trusted third parties (CAs) that vouch for the binding between the domain name and the private key. The problem is that we've managed to bless several hundred of these supposedly trusted parties, any of which can vouch for any domain name. Every now and then, one of them gets it wrong, sometimes spectacularly.”

Pinning inadequate
Notaries inadequate
DNSSec inadequate
All require trust in 3rd party compute infrastructure that is inherently vulnerable

Certificate Transparency: (related EFF SSL Observatory)
Public end-verifiable append-only event log with consistency and inclusion proofs
End-verifiable duplicity detection = Ambient verifiability of duplicity
Event log is third party infrastructure but zero trust because it is verifiable.
Sparse Merkle Trees for revocation of certificates

Certificate Transparency Solution

Public end-verifiable append-only event log with consistency and inclusion proofs
End-verifiable duplicity detection = ambient verifiability of duplicity
Event log is third party infrastructure but it is not trusted because logs are verifiable.
Sparse Merkle trees for revocation of certificates
(related EFF SSL Observatory)

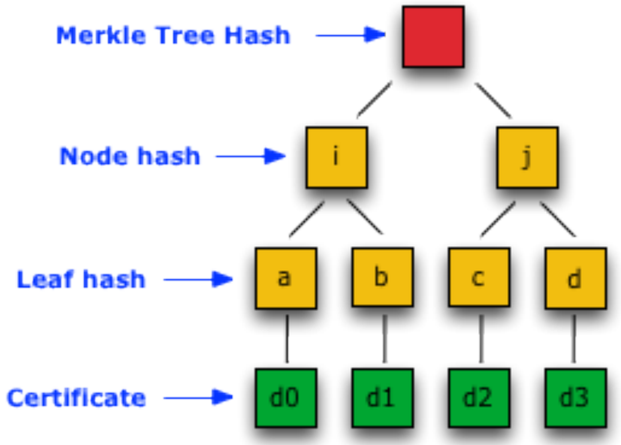


Figure 1

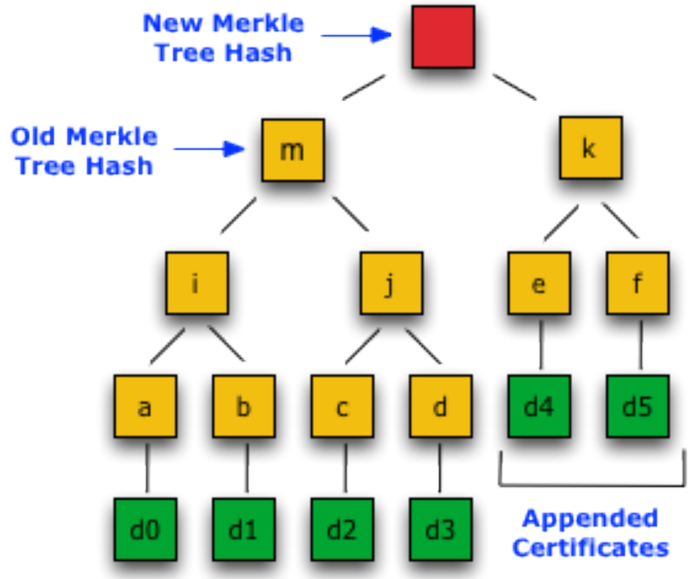


Figure 2

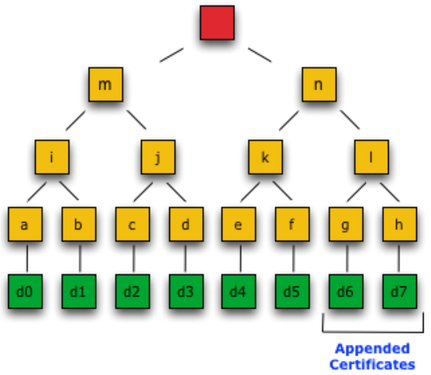


Figure 3

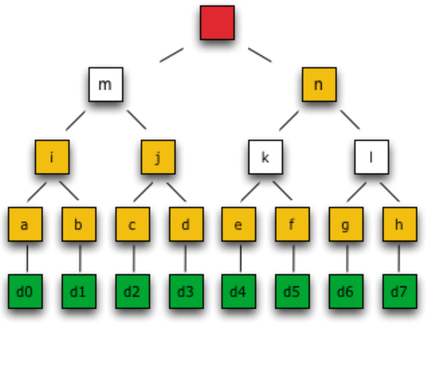


Figure 4

Key Event Receipt Infrastructure

Security Considerations

Samuel M. Smith Ph.D.
sam@samuelsmith.org
https://keri.one
version 2.59
2021/03/09

Macro vs. Micro Security Considerations for Public KERI Identifiers

Macro:

Cryptographic root-of-trust via self-certifying identifiers (SCID) of different types (self-addressing, delegated, etc)

Cryptographically Verifiable Data Structure (VDS) that provides proof of key state for its identifier = Key Event Log (KEL)

Key management embedded in VDS construction including recovery from key compromise

Multi-valent key management infrastructure via delegation

Externally anchored transactions via cryptographic commitments in VDS = Transaction Event Log (TEL)

Separated control of promulgation of KEL (witnesses) vs confirmation of KEL (watchers+)

Threshold structures via pools of promulgation nodes and confirmation nodes

Duplicity Detection

Micro:

Details of self-certifying identifier construction

Details of VDS = KEL construction

Details of key management embedded in VDS construction including recovery from key compromise

Details of multi-valent key management infrastructure via delegation

Details of promulgation and confirmation communication protocols and pool construction

Details of Duplicity Detection algorithms and protocols

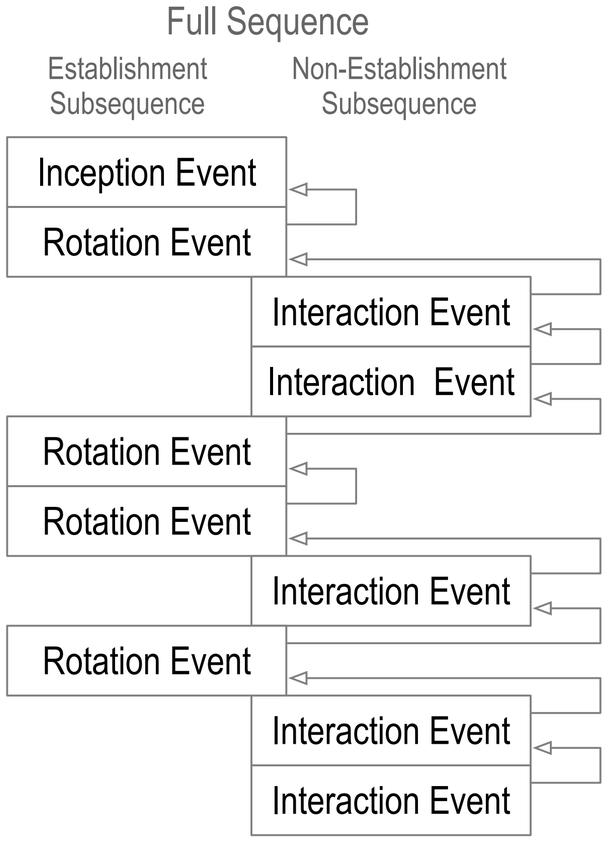
The security of KERI private identifiers in pair-wise or group-wise relationships is trivially established.

Security from Outside In

- Treat Micro considerations as a black box with certain properties
- Examine the macro considerations

KEL
(VDS)
Black Box

Properties:
KEL (VDS) is cryptographically verifiable hash chained non-repudiably signed data structure
Successful attacks on KEL (VDS) and its root-of-trust require key compromise
Best practices KEY management practically limits Key compromise to side channel attacks on exposed signing keys
Recovery from signing key compromise ensured via unexposed post quantum committed one time rotation keys
Multi-valent cooperative delegation = scalable performant tiered-security key management via hierarchies of KELs



Terminology

Controller is entity that controls the authoritative set of asymmetric key-pairs for an identifier (PKI). Non-repudiable signature key pairs.

KEL provides cryptographically verifiable proof of control authority over an identifier and its KEL via authoritative set of key-pairs

KEL authoritative key state is recursively constructed from inviolable entropy generated root-of-trust for the incepting key state.

Future changes to key state are controlled by key-pairs that the KEL itself establishes as authoritative via keys pairs from past key state.

KEL is a cryptographically verifiable state machine that provides proof of key state and hence proof of control authority over identifier.

Validator is entity that validates the key state of a given Controller's identifier.

This validation includes but is not limited to cryptographically verifying the KEL.

A two party interaction between a first party and second party creates a pair-wise relationship of two identifiers/KELs.

Each party acts in turn as *Controller* of own identifier/KEL and *Validator* of the other's identifier/KEL.

Each *Validator's decision to interact* is based on its validation and resultant degree of trust in *Controller's* identifier/KEL used in the interaction.

Macro Consideration #1: Key Compromise

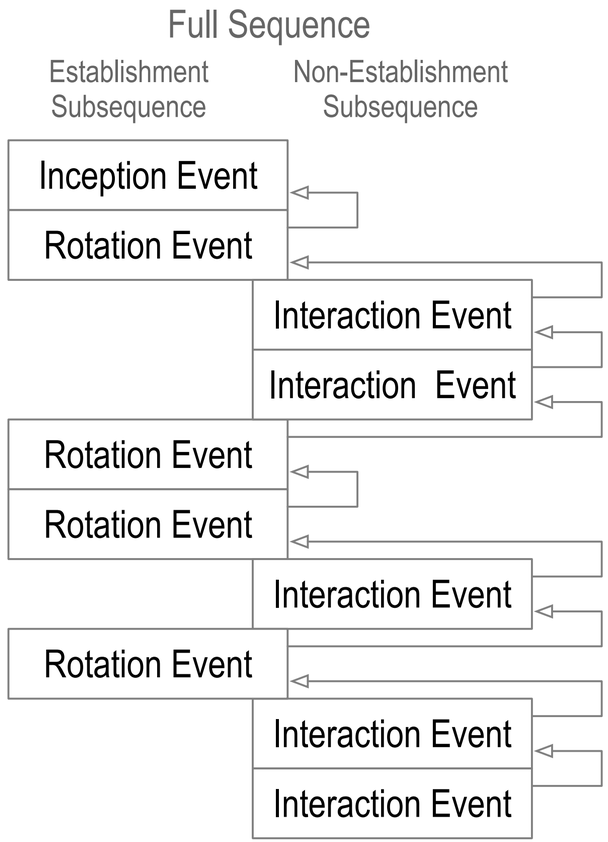
- External entity compromises Controller's exposed signing keys.
- Malicious Controller misuses its own signing keys

Both exhibit as the existence of two or more duplicitous but cryptographically verifiable KELs for a given identifier

Non-repudiable signatures on each KEL make duplicity provable and detectable given any copy of set of duplicitous KELs from any source.

Validator may be protected from such key compromise given it detects duplicity before continuing interaction with Controller.

Inconsistency and Duplicity



inconsistency: lacking agreement, as two or more things in relation to each other

duplicity: acting in two different ways to different people concerning the same matter

Internal vs. External Inconsistency

Internally inconsistent log = not verifiable.

Log verification from self-certifying root-of-trust protects against internal inconsistency.

Externally inconsistent log with a purported copy of log but both verifiable = duplicitous.

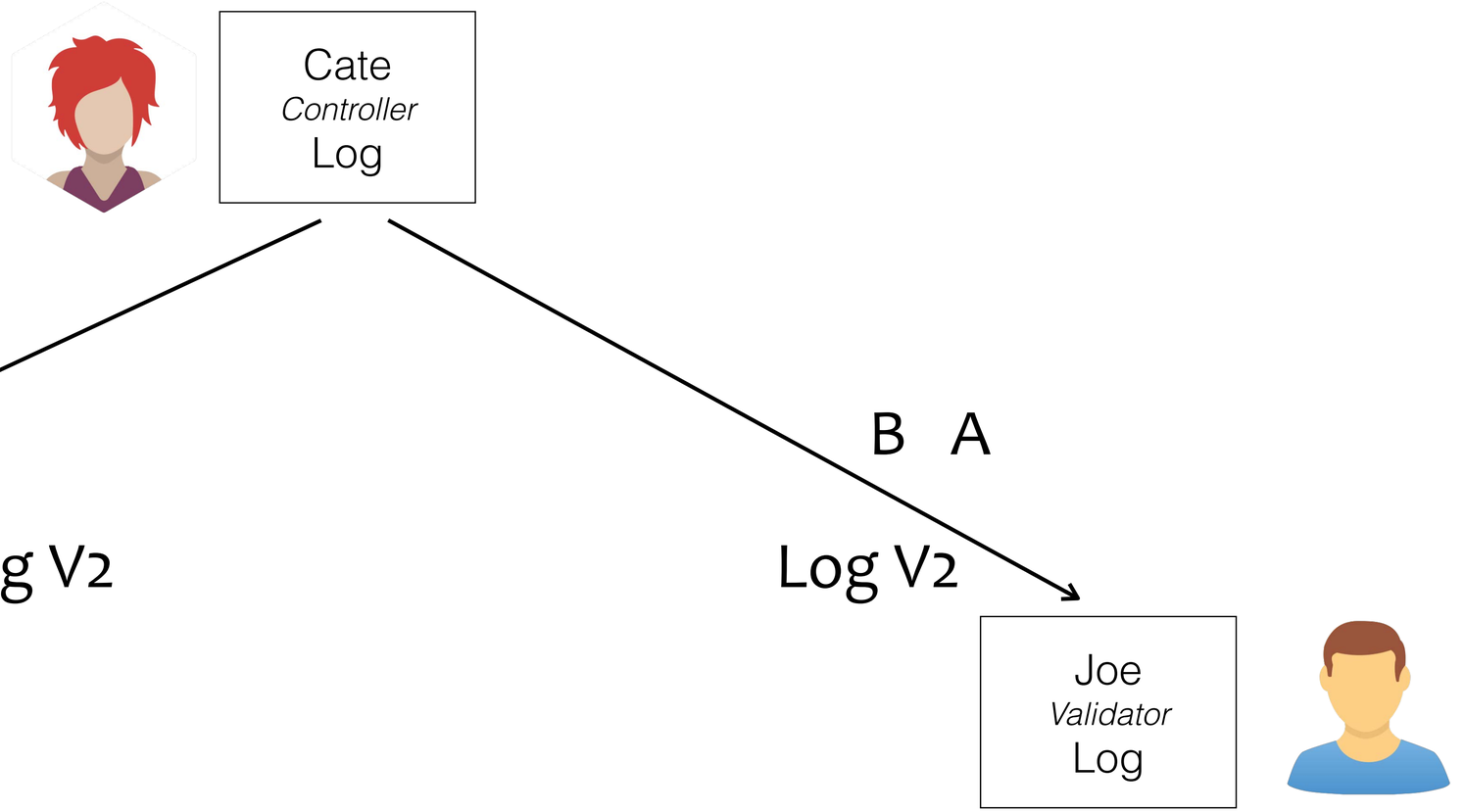
Duplicity detection protects against external inconsistency.

Duplicity Game

Cate promises to provide a consistent pair-wise log.

Local Consistency Guarantee

How may Cate be duplicitous and not get caught?



private (one-to-one) interactions

Duplicity Game

Service promises to provide a consistent log to anyone.

Local Consistency Guarantee

How may Cate/Service/Agent be duplicitous and not get caught?

Cate
Controller
Log

Truncate Log

Delete Log

Service/Agent
Controlled by Cate
Log

A

B A

Log V1 Log V2

Log V2

Eve
Validator
Log

Joe
Validator
Log

highly available, private (one-to-one) interactions

Duplicity Game

Service promises to provide exact same log to everyone.

Global Consistency Guarantee

How may Cate and/or service be duplicitous and not get caught?

Cate
Controller
Log

Breaking the promise of global consistency is a provable liability.

Global consistency may only matter after Eve and Joe need to interact not before.

Service
Controlled by Cate
Log

Truncate Log

?

isolate network

isolate network

A

Delete Log

A

Doug(s)
Global Duplicity Network
Log

log V1

log V2

Eve
Validator
Log

Ambient Duplicity Detection

Joe
Validator
Log

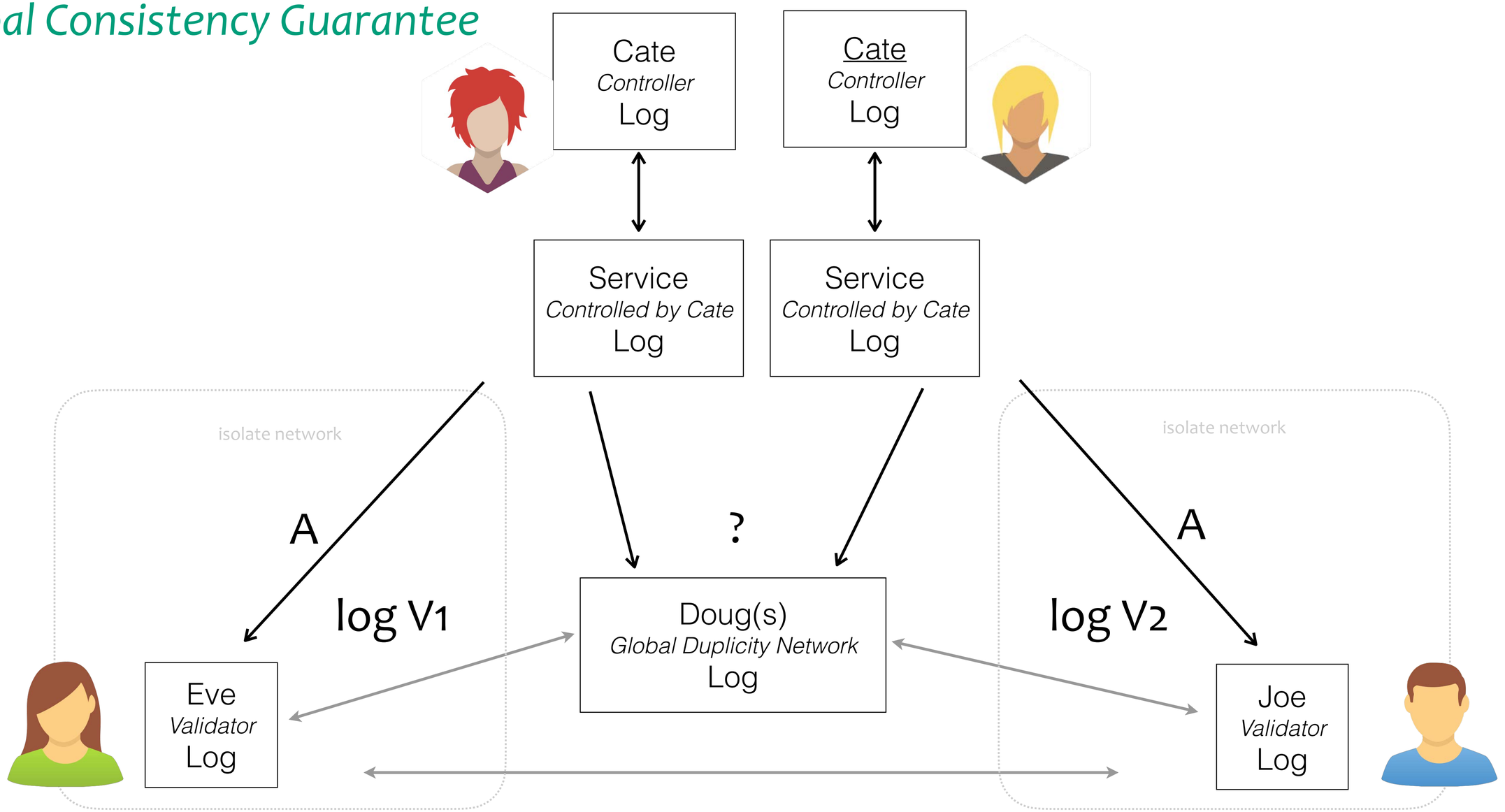
global consistent, highly available, and public (one-to-any) interactions

Duplicity Game

Service promises to provide exact same log to everyone.

Global Consistency Guarantee

How may Cate and/or service be duplicitous and not get caught?



global consistent, highly available, and public (one-to-any) interactions

Macro Consideration #2: First Seen Immutability of Verifiers (Watchers) of KEL

Watcher is observer of KEL that verifies a given KEL using its *first seen* version of that KEL.

First Seen = Always Seen never Unseen. No later key compromise (dead attack) may change first seen state of honest watcher..

Honest watcher's verified KEL is immutable due to its first seen policy.

Watchers are selected by or under the control of Validator.

A pool of watchers may be secure without trusting any given watcher due to threshold structure of pool.

By construction a Validator may trust that a sufficient majority of Watchers in pool are honest at any time.

Watcher pool thus constructed either provides simple secure distributed consensus about KEL duplicity or not at all.

Validator may thereby appraise *authoritative* state of a given KEL in spite of duplicity. One authoritative KEL or none at all.

First Seen consensus in combination with the fact that successful attacks on a KEL require key compromise of exposed keys mean that Controller may have window of opportunity to promulgate key event before key compromise of keys exposed by that same promulgated event.

Once promulgated throughout watcher network (which may happen in milliseconds) the event becomes immutably first seen by consensus majority of watchers. (global majority)

No later compromise of keys and resultant promulgation of alternative event may displace that first seen consensus version.

The authoritative KEL is thus established immutably or not at all.

In the case of side channel attack on live signing keys (live attack), malicious event may be promulgated before Controller's event. A rotation recovery from one time pre-rotated keys enables controller to recover from such a live attack.

This recovery operation may be repeated until Controller detects and mitigates side channel attack.

Main purpose of Witness pool designated by Controller is to help honest Controller ensure that its own promulgation of any event happens before any compromised event.

Macro Consideration #3: Safety and Liveness of Total Ordering

Safety and *Liveness* are typically used to describe properties of Byzantine fault tolerant (BFT) totally ordering distributed consensus algorithms

Such an algorithm depends on a pool of Nodes that each maintain a replicated copy of a global ledger.

Each replicated ledger is a verifiable data structure (VDS) consisting of transactions promulgated to the nodes by any number of remote clients.

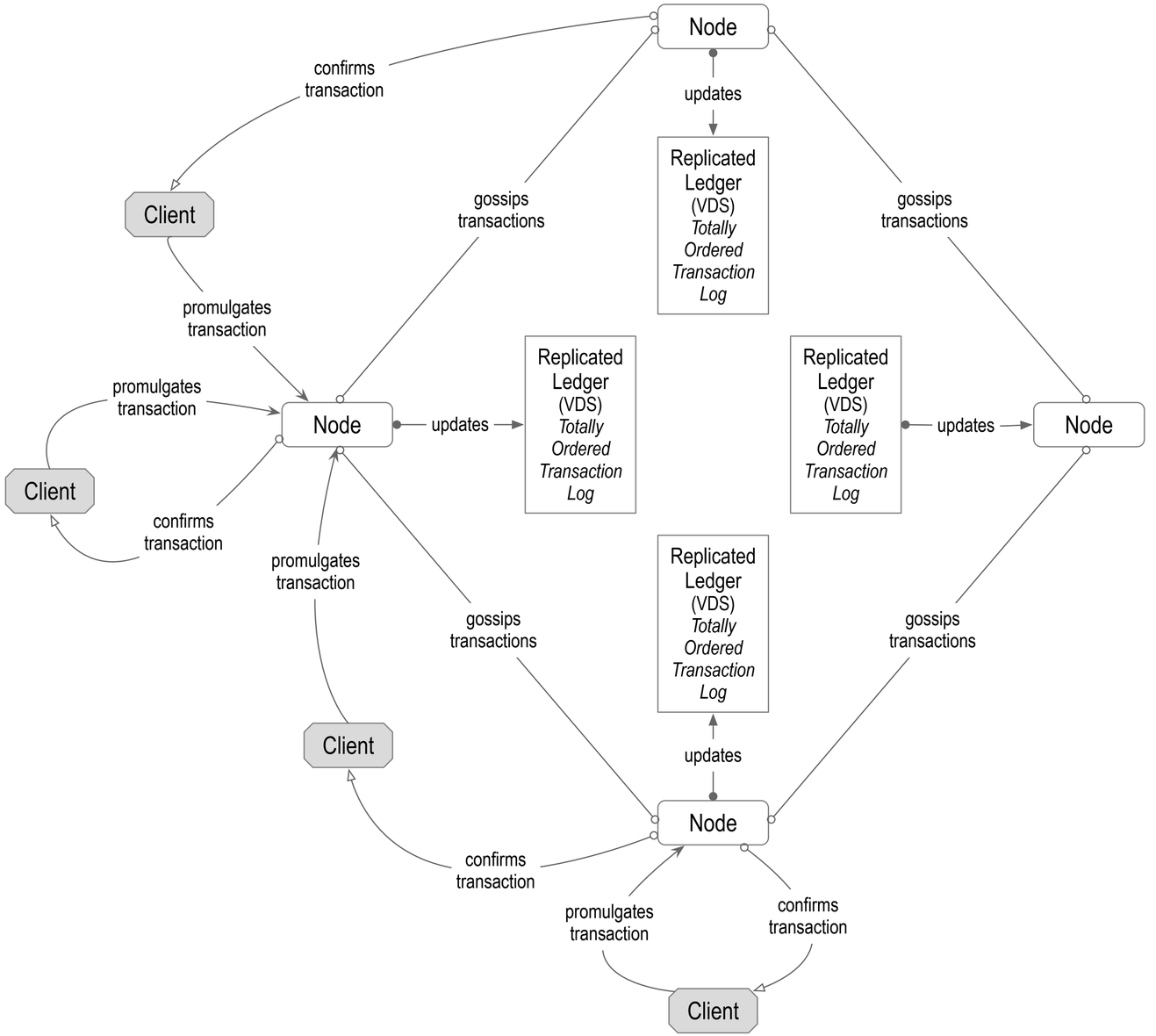
Total ordering means that all the transactions from all clients have the same ordering in every replicated ledger.

Total ordering enables double spend proofing of a both fungible asset balances and non-fungible assets in the ledger.

Double spend proofing is an essential property for crypto-currencies.

Safety means that whenever Node pool comes to consensus as to the ordering of of all client promulgated transactions they will come to the same consensus or none at all.

Liveness means that the Node pool will eventually come to consensus as the total ordering of all client promulgated transactions.



Macro Consideration #3 continued: Hard constraint space = high cost floor

There are four major classes of BFT total ordering distributed consensus algorithms with many variants and hybrids. These are:

Verifiable Random Function (VRF)

Byzantine Agreement (BA)

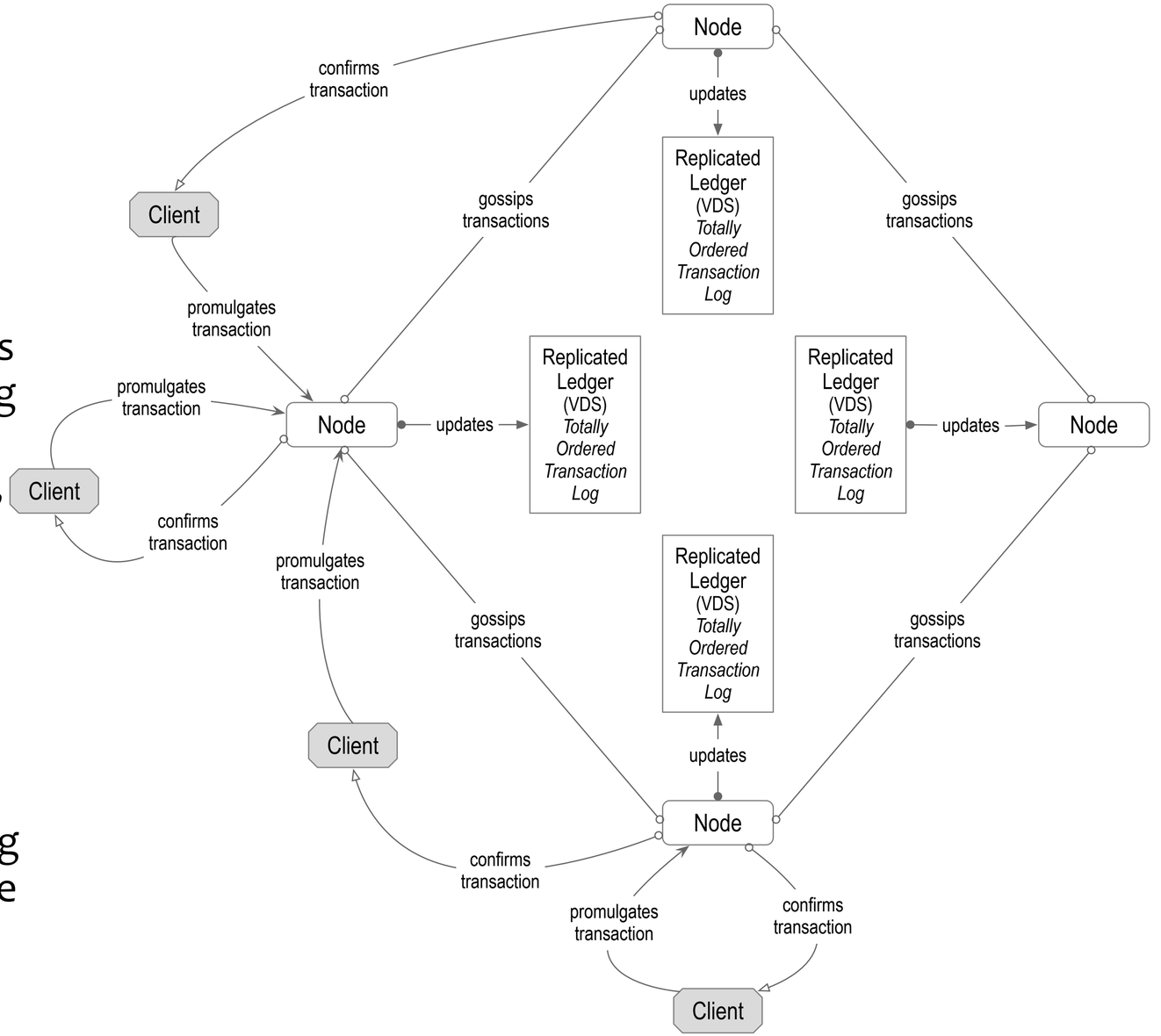
Proof-of-Work (PoW)

Proof-of-Stake (PoS)

Due to all these algorithm's and supporting infrastructure's shared requirements for safety, liveness, and total ordering they operate within hard constraint space that must trade off performance and cost between scalability, throughput, latency and governance to name a few.

This hard constraint space establishes a cost for performance floor that may be orders of magnitudes higher than cloud infrastructure that does have similar requirements.

Thus an identifier system does not absolutely need to exhibit all the properties safety, liveness, and total ordering then there may be compelling reasons to relax one or more of these requirements.



Macro Consideration #3 continued: Ledger as primary root-of-trust is not portable

When the Ledger Algorithm is the primary root-of-trust/ source-of-trust for ordering transactions then key-state is bound to that ledger.

Porting to other ledgers requires additional infrastructure or a VDS that is not the Ledger. In which case the Ledger merely acts as storage , discovery, and/or secondary root-of-trust mechanism.

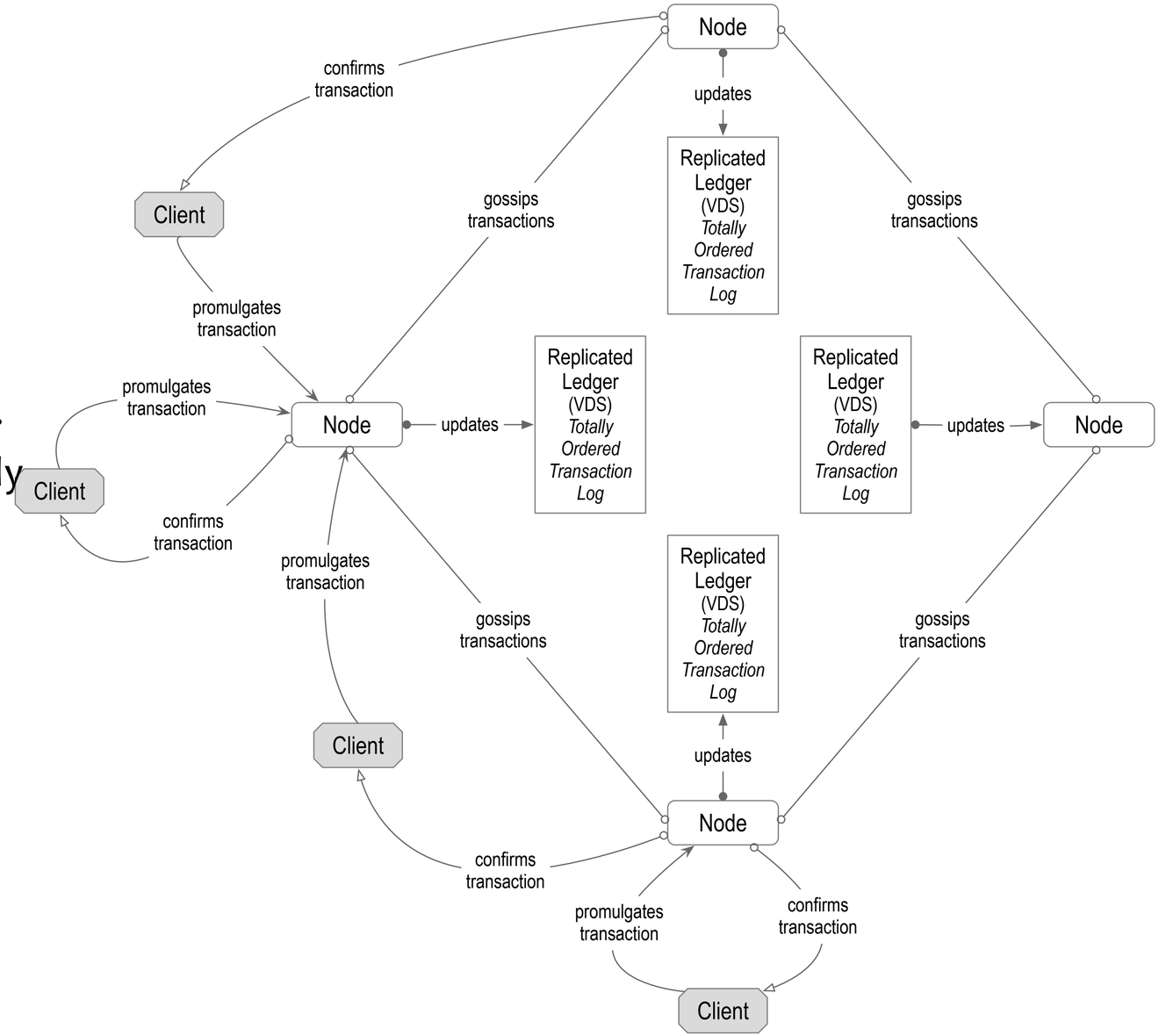
Portable infrastructure requires cryptographic commitments to infrastructure state just like key state.

Ledgers are essentially platforms with shared governance.

Identifier systems that are locked to a ledger are essentially platformed identifiers (identity).

KERI wants to de-platform identity systems by making identifiers truly portable.

Much of DID method proliferation is driven by an attempt to drive market value to a given platform usually a ledger with a platform (ledger) locked (platform specific) identifier (DID method).



Ledger Attack Vectors

TABLE I

ATTACK VECTORS RELATED TO THE ATTACK CLASS IN BLOCKCHAIN SYSTEMS. WE ALSO SHOW, BY REFERENCING TO THE PRIOR WORK, HOW EACH ATTACK AFFECTS THE ENTITIES INVOLVED WITH BLOCKCHAIN SYSTEMS. FOR INSTANCE, ORPHANED BLOCKS AFFECT THE BLOCKCHAIN, THE MINERS, AND THE MINING POOLS.

	Attacks	Blockchain	Miners	Mining Pools	Exchanges	Application	Users
Blockchain Structure	Forks [26]	✓					
	Orphaned blocks [27]	✓	✓	✓			
Peer-to-Peer System	DNS hijacks [28]		✓	✓	✓		✓
	BGP hijacks [29]		✓	✓			✓
	Eclipse attack [30]		✓				✓
	Majority attack [31]	✓	✓			✓	
	Selfish mining [32]	✓	✓	✓			
	DDoS attacks [33]	✓	✓	✓			
	Consensus Delay [34]		✓	✓			✓
	Block Withholding [34]		✓	✓			
	Timejacking attacks [35]		✓	✓		✓	
	Finney attacks [36]		✓	✓			✓
Blockchain Application	Blockchain Ingestion [37]	✓					
	Wallet theft [38]				✓	✓	✓
	Double-spending [39]	✓					✓
	Cryptojacking [40]					✓	✓
	Smart contract DoS [41]	✓				✓	✓
	≈ Reentracy attacks [42]					✓	✓
	≈ Overflow attacks [42]					✓	✓
	≈ Replay attacks [41]	✓		✓		✓	✓
	≈ Short address attacks [42]					✓	
	≈ Balance attacks [41]					✓	✓

Exploring the Attack Surface of Blockchain: A Systematic Overview https://arxiv.org/pdf/1904.03487.pdf

Ledger Attack Effects

TABLE II

IMPLICATIONS OF EACH ATTACK ON THE BLOCKCHAIN SYSTEM IN THE LIGHT OF THE PRIOR WORK. FOR INSTANCE, FORKS CAN LEAD TO CHAIN SPLITTING AND REVENUE LOSS. AS A RESULT OF A FORK, ONE AMONG THE CANDIDATE CHAINS IS SELECTED BY THE NETWORK WHILE THE OTHERS ARE INVALIDATED. THIS LEADS TO INVALIDATION OF TRANSACTION AND REVENUE LOSS TO MINERS.

	Attacks	Chain Splitting	Revenue Loss	Partitioning	Malicious Mining	Delay	Info Loss	Theft
Blockchain Splitting	Forks [26]	✓	✓					
	Orphaned Blocks [27]		✓					
P2P System	DNS hijacks [28]		✓	✓				✓
	BGP hijacks [29]		✓	✓				✓
	Eclipse attacks [30]			✓				
	Majority attacks [31]	✓	✓		✓			
	Selfish mining [32]		✓		✓			
	DDoS attacks [33]				✓			✓
	Consensus Delay [34]					✓	✓	
	Block Withholding [34]		✓		✓			
	Timejacking attacks [35]	✓	✓		✓	✓		
	Finney attacks [36]		✓					
Blockchain Application	Blockchain Ingestion [37]						✓	
	Wallet theft [38]		✓					✓
	Double-spending [39]							
	Cryptojacking [40]	✓			✓			✓
	Smart contract DoS [41]		✓			✓		✓
	≈ Reentracy attacks [42]		✓					✓
	≈ Overflow attacks [42]							✓
	≈ Replay attacks [41]		✓				✓	
	≈ Short address attacks [42]		✓					✓
	≈ Balance attacks [41]		✓					✓

Exploring the Attack Surface of Blockchain: A Systematic Overview https://arxiv.org/pdf/1904.03487.pdf

Macro Consideration #3 continued: Secure Attribution only requires Safety, not liveness nor Total Ordering

An identifier system must solve the secure attribution problem. In other words any statement must be either securely attributed to its source or not at all, i.e. the authenticity of any statement may be established. This protects any user from harm that may result from relying on the authenticity of that statement.

The innovation of KERI is recognizing that the secure attribution problem may be solved by only requiring safety not liveness or total ordering.

In KERI the only entity that may order the events in a KEL is the Controller. Thus total ordering is not required. In KERI a controller is solely responsible for ensuring liveness if at all. The one primarily harmed by lack of liveness is an honest Controller. Each validator is responsible for its own safety with regard a given Controller's KEL. If the validator has irreconcilable evidence of duplicity then that Controller's KEL is effectively dead (no liveness) to that Validator. An honest Controller has every incentive to ensure there is no irreconcilable duplicity with regards its own KEL to ensure liveness.

So a KEL is ordered by its Controller and it may be live and safe but the liveness is up to the controller while safety is up to the Validator.

Better *dead* but *safe* than *live* but *unsafe*.

Macro Consideration #4: Honest Controller ensures Liveness with key management

An honest Controller may ensure liveness of its KEL by employing best-of-breed key management for its authoritative key sets and best practices security for its witnesses.

KERI provides a full suite of operations to facilitate best-of-breed key management with async multi-sig, post quantum pre-rotation, and recovery of compromised signing keys, as well as hierarchical tiered cooperative delegated multi-valent infrastructure with recovery of both compromised signing and compromised rotation keys on nested levels.

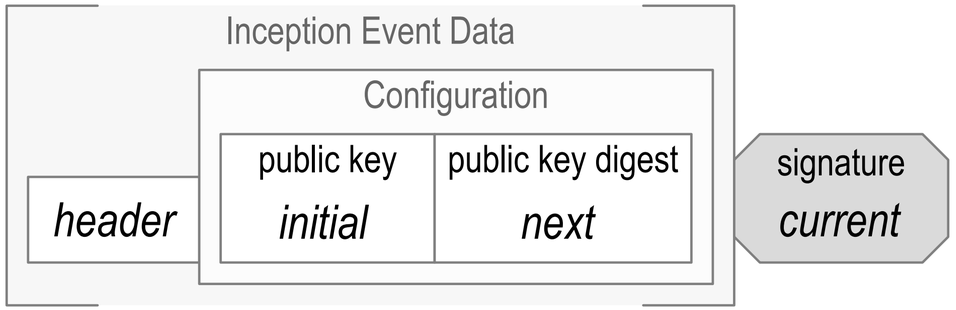
KERI witnesses allow any level of extra security layering (additional access control) by an honest Controller to ensure its own un-compromised events are always promulgated first, resulting in universal immutable first seen status of the global watcher network.

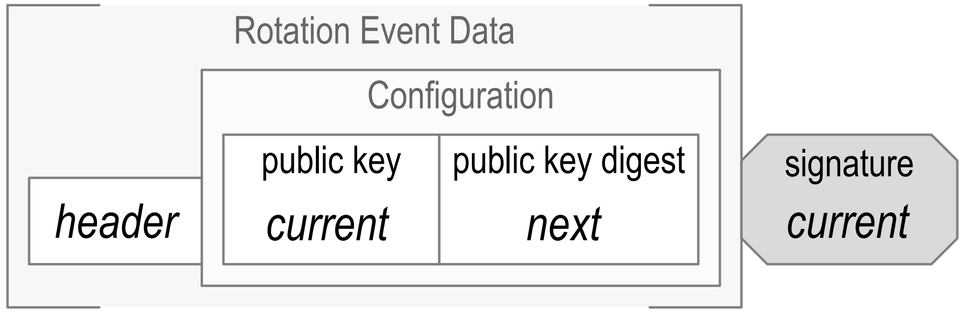
As a special case a Controller may designate as its witnesses, one or more Ledger oracles. This allows applications to leverage needed for beneficial features of a ledger or ledgers.

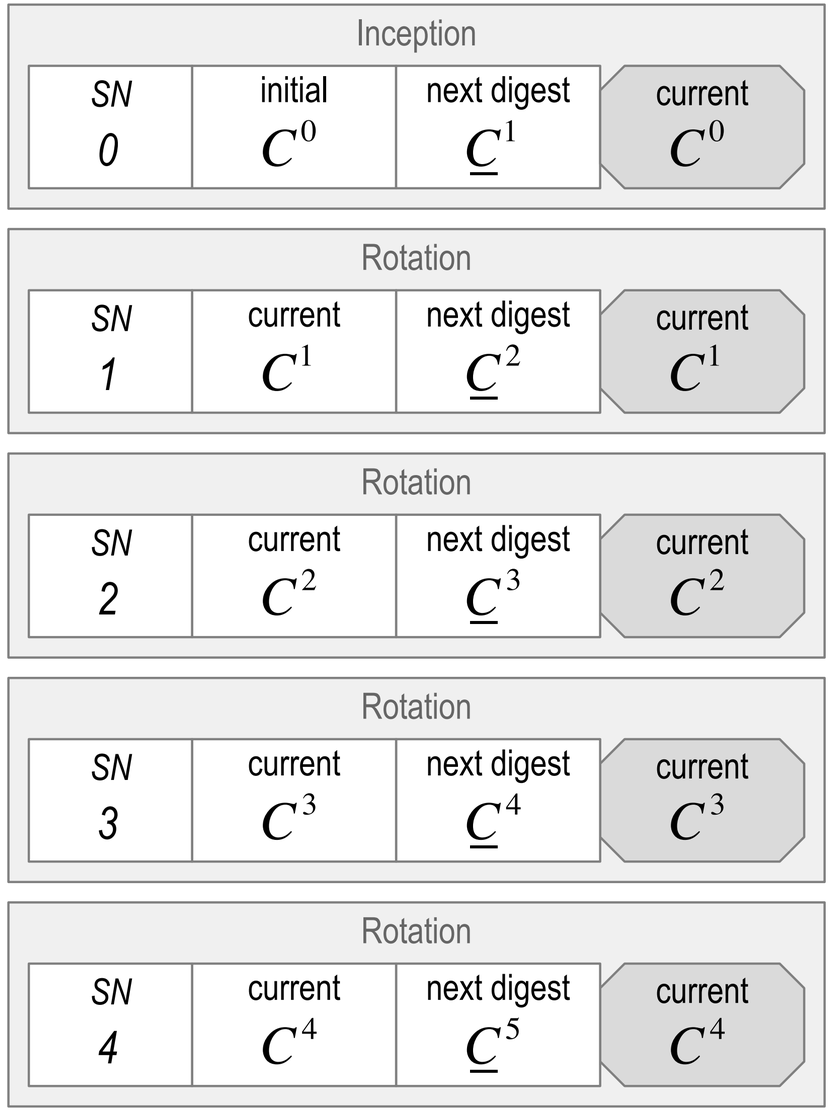
One such application are NFTs where all of safety, liveness, and double spend proofing are required. But because NFTs are non-fungible we can use a KEL with a transferable identifier derived from the digital content of the NFT to provide double spend proofing. We only need the ledger to guarantee liveness. Specifically to ensure double spend proofing and liveness only one ledger at a time may be used but the NFT may be portably moved from one ledger to another by rotating the one witness. So its a one-ledger-at a time solution. All validators must trust the one ledger so there is no duplicity. The authoritative KEL is always the one KEL registered on the one ledger and then moved or registered on a new ledger.

Recall that liveness is the responsibility of the controller. A given NFT could require that each successive controller of the NFT in turn use a minimal level of key management and witness configuration that ensures liveness to a higher margin than a ledger ensures safety. In such a case then the overall security of a non-ledger KERI NFT would be higher than a ledger based KERI NFT. The weak link of the ledger is the access identifier private key protection from key compromise (safety) which may be much less than KERI.

Pre-Rotation

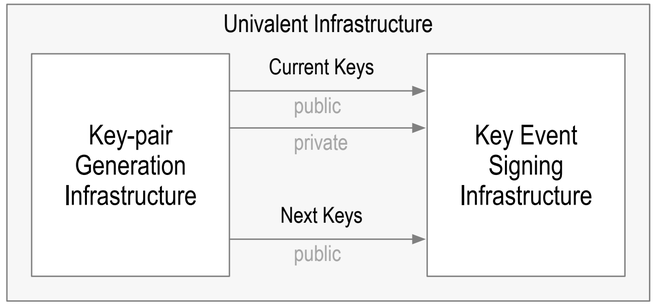


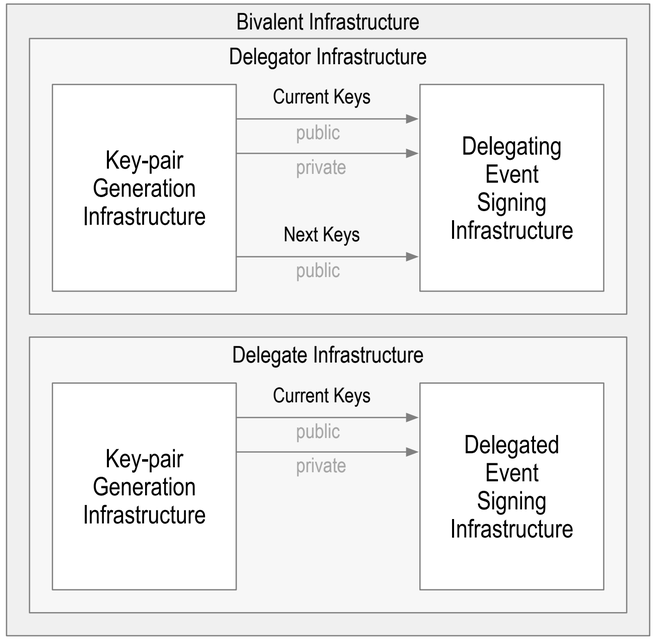


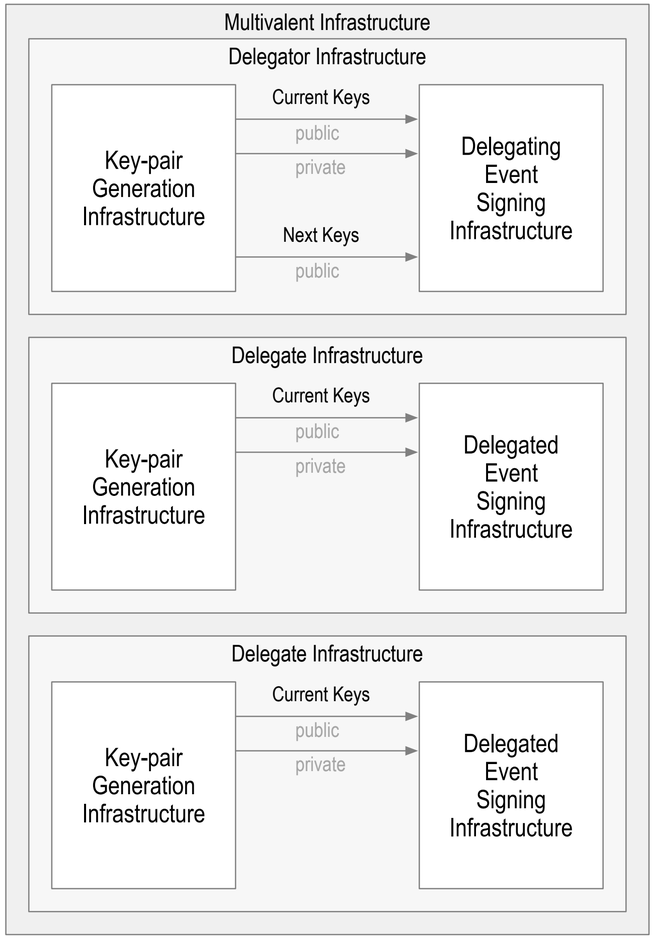


Digest of *next* key(s) makes pre-rotation post-quantum secure

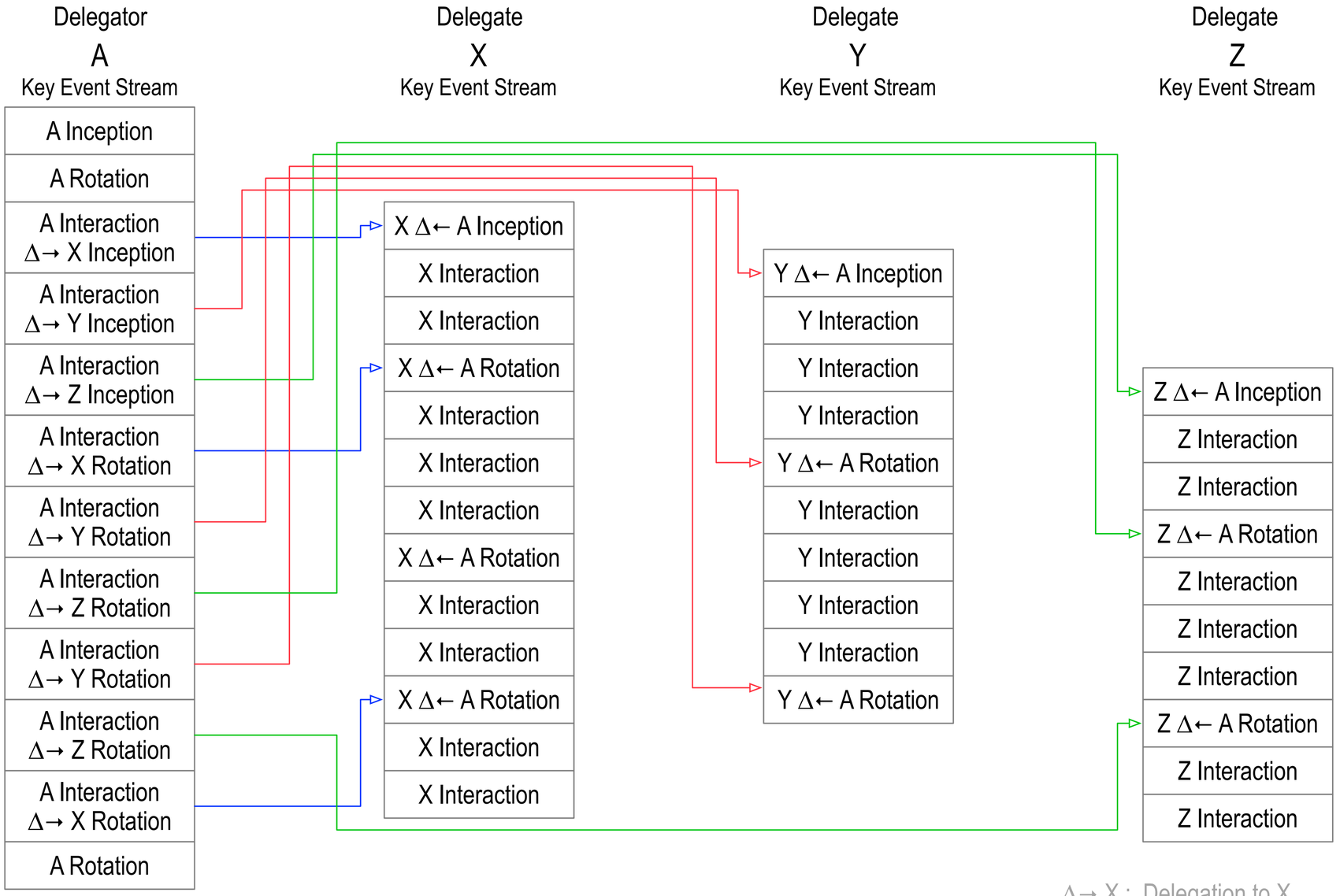
Key Infrastructure Valence







Scaling Delegation via Interaction



Macro Consideration #5: Validator is responsible for ensuring its Safety WRT a given KEL

Each Validator selects Watchers it trusts.

These may include multiple levels of trust from Watchers it controls to reputable watchers controller by other entities.

Together they form a watcher network with a shared incentive to detect duplicitous behavior by Controllers.

Each Validator appraise evidence of duplicity as observed by its Watcher network.

Given no evidence of duplicity then Validator trusts KEL.

Given evidence of duplicity then Validator must reconcile that duplicity or else the KEL is dead to that Validator.

KERI provides key rotation recovery mechanisms to enable duplicity reconciliation of live signing key compromise.

Immutable first seen property of honest watchers in threshold structure enables duplicity reconciliation of dead (stale) key compromise.

Security Concepts

Availability, Consistency, and Duplicity

Harm to controller: Unavailability, loss of control authority, externally forced duplicity

Harm to validator: Inadvertent acceptance of verifiable but forged or duplicitous events

Local vs. Global Duplicity Guarantees

Direct Mode vs. Indirect Mode Operation

Malicious Controller vs. Malicious Third Party

Live Exploit vs. Dead Exploit

Controller Protection vs. Validator Protection

Protection to controller: key management, promulgation consensus, redundancy.

Protection to validator: verifiable logs, verification consensus, duplicity detection

Apples-to-Apples

Ledger:

Network paid by transaction fees

(more or less competitive within the network)

Successful exploits without compromised keys

Controller:

Highly available nodes of other's choosing

Must trust that a majority are honest

No recovery if keys compromised

Validator;

Need full copy of ledger (big)

Need full access to network

Must trust that a majority are honest

KERI:

Networks paid by transaction fees

(competitive across all networks)

Controller:

Highly available nodes of own choosing

Must "*trust*" that a majority are honest

Successful exploits must compromise keys

Recovery if keys compromised

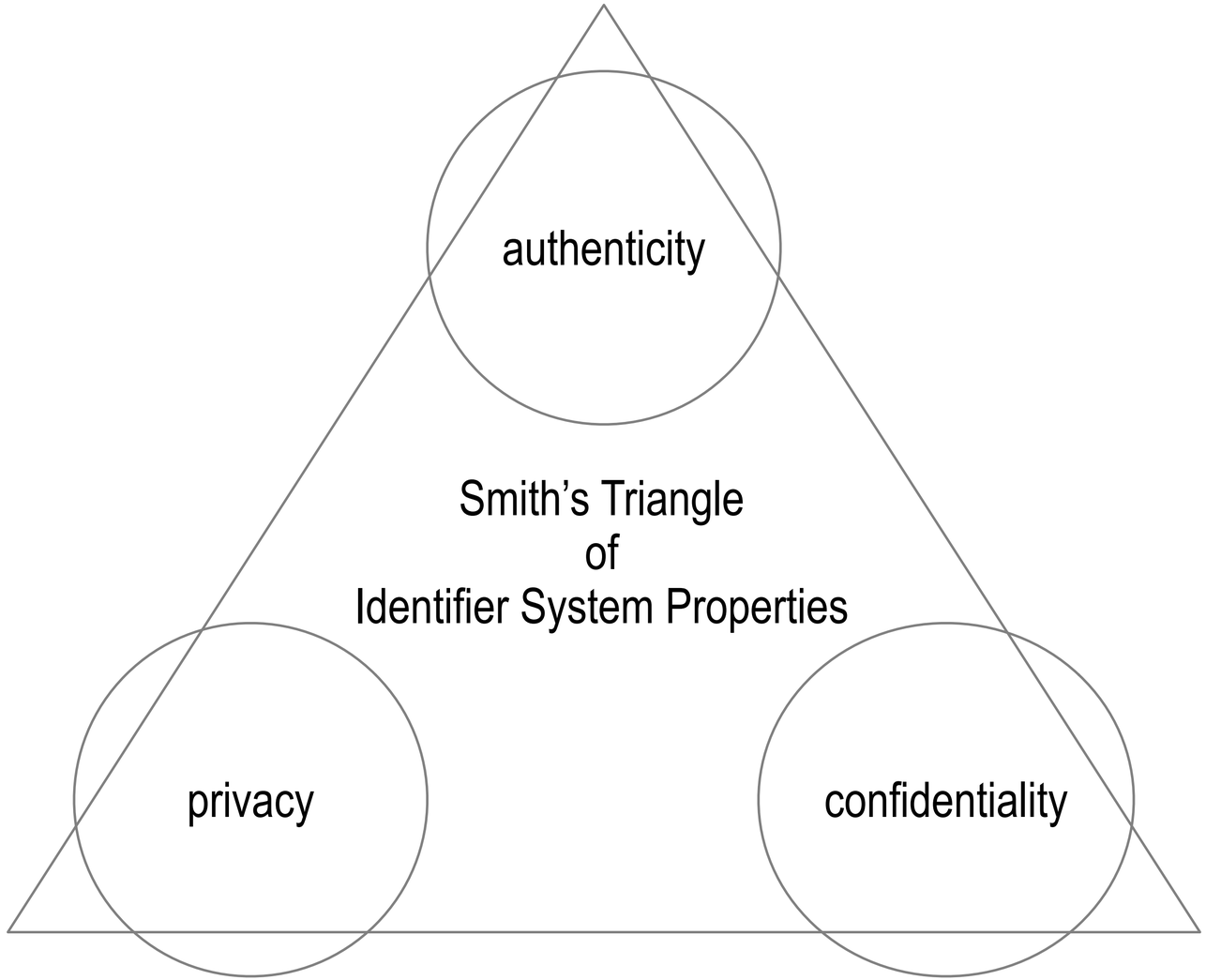
Validator:

Need full copy of KEL (small)

Need full access to network of own choosing.

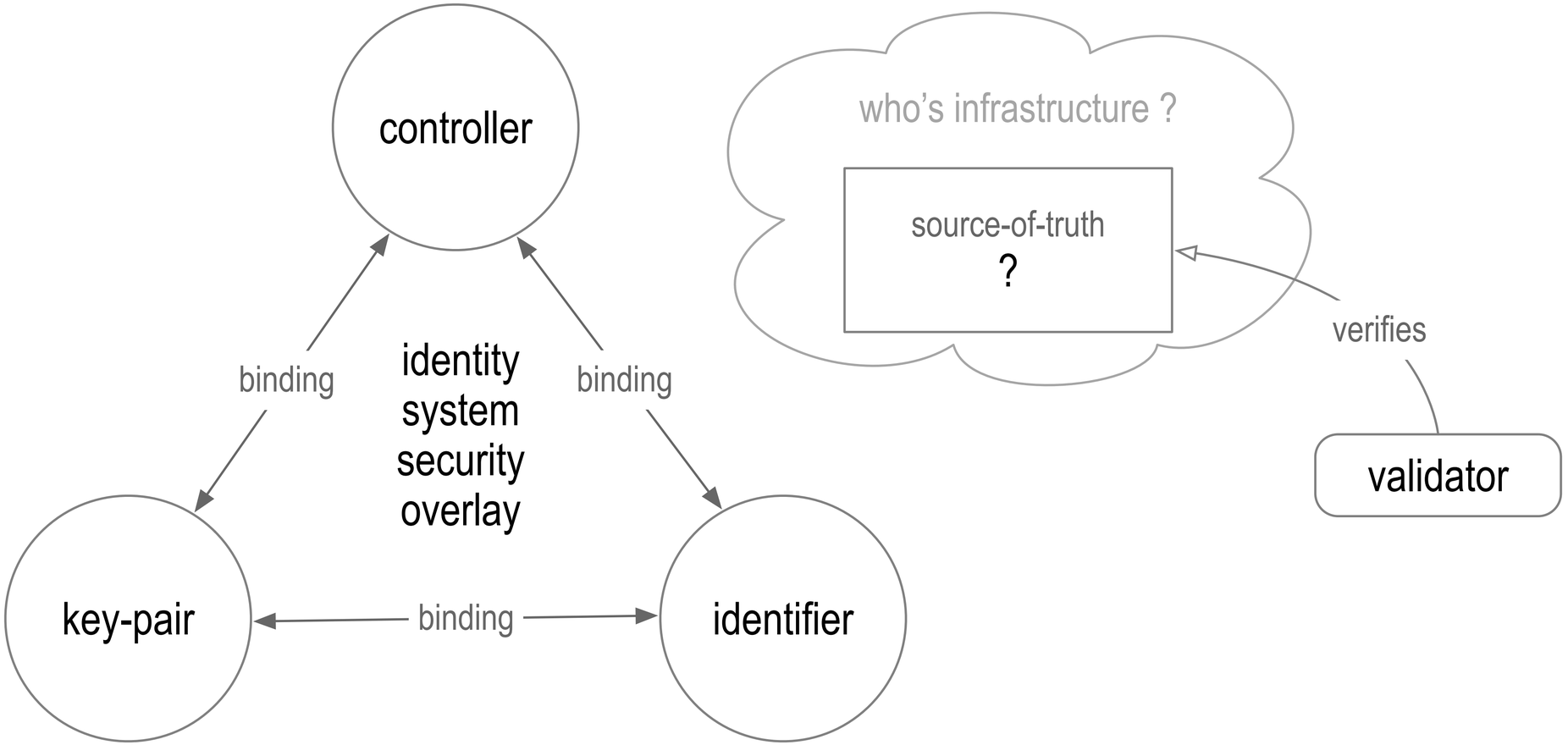
Must "*trust*" that a majority are honest

Smith’s Identifier System Properties Triangle



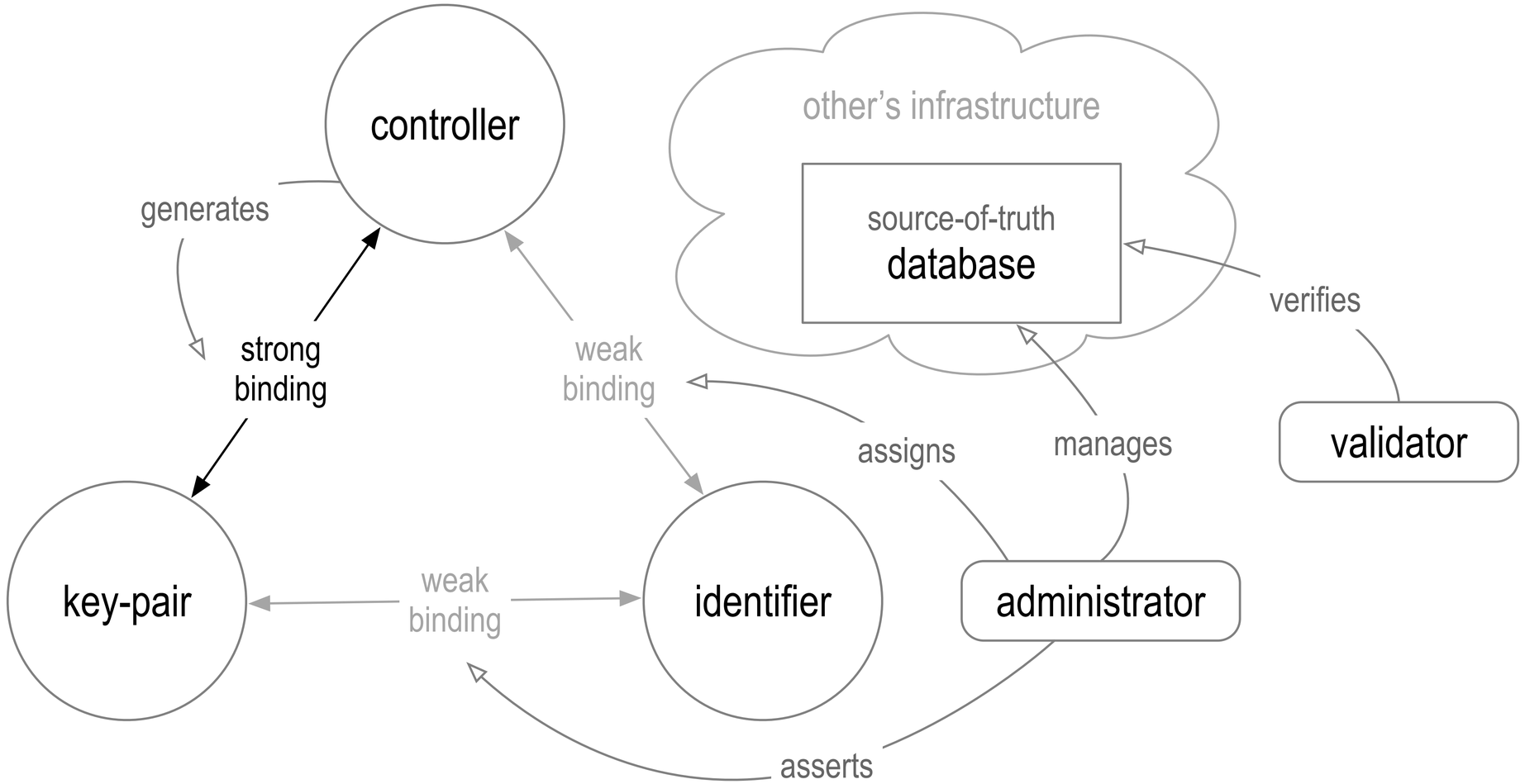
May exhibit any two at the highest level but not all three at the highest level

Trust Basis


controller
who's infrastructure ?
source-of-truth
?
verifies
validator
binding
identity
system
security
overlay
binding
key-pair
binding
identifier

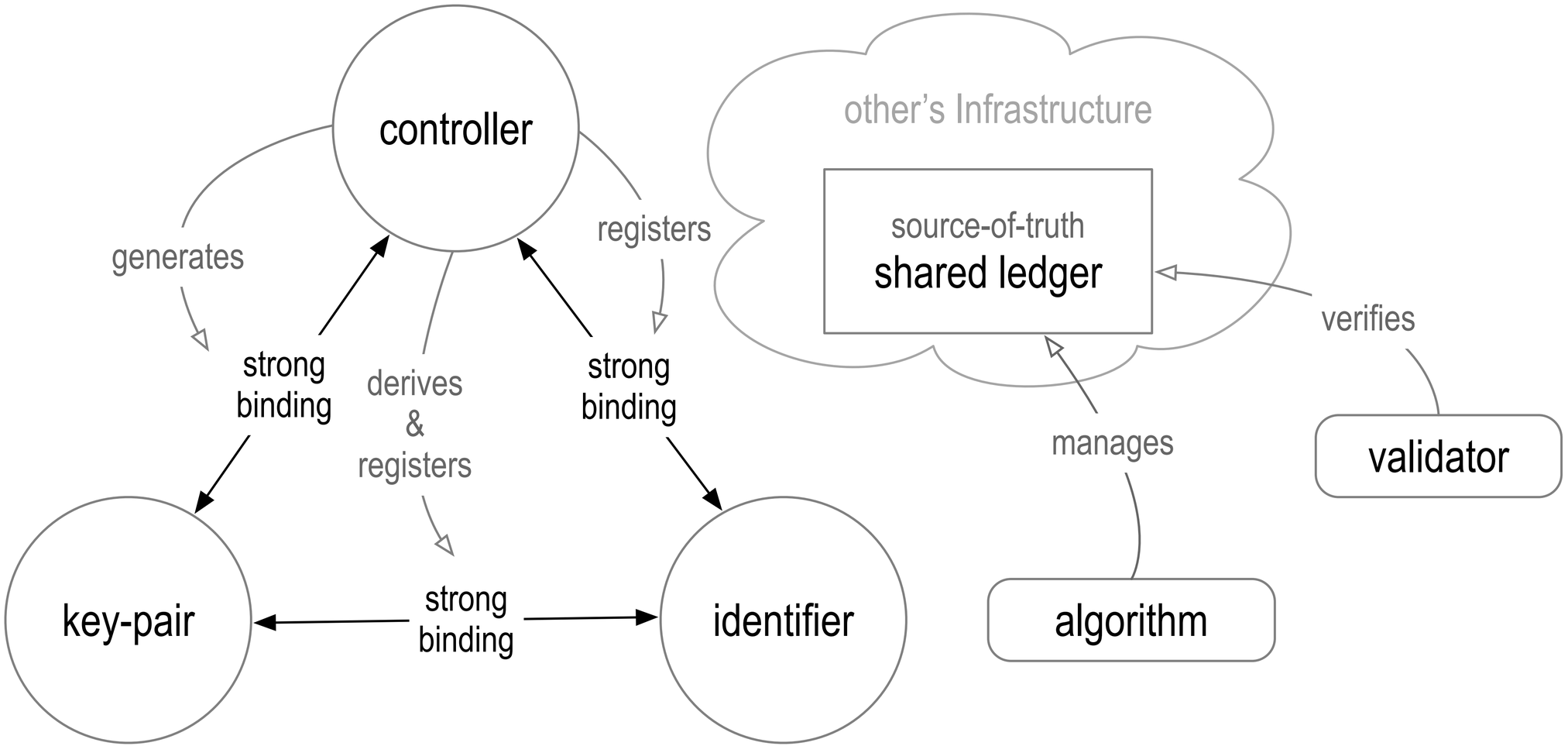
Administrative Trust Basis

DNS/Certificate Authorities



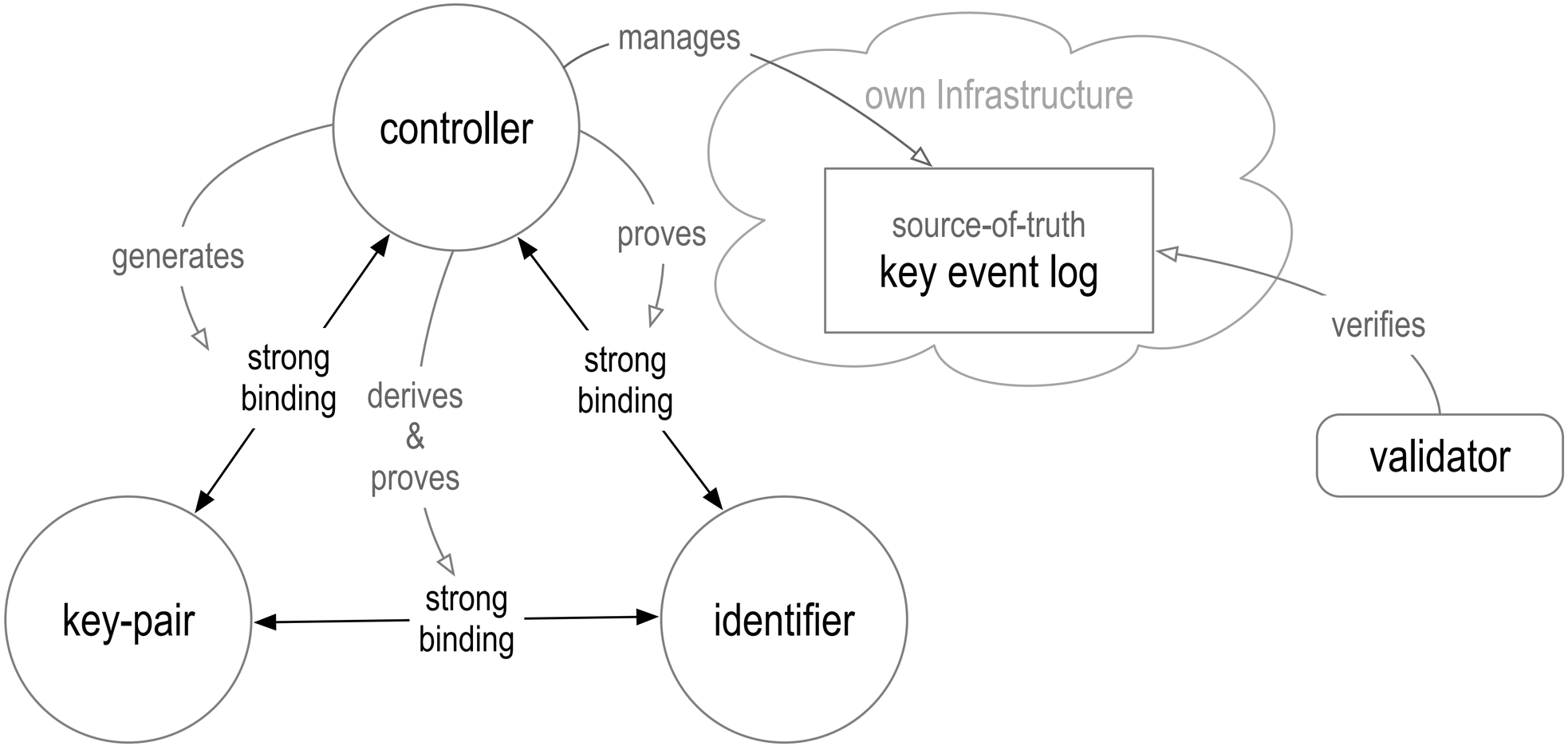
Algorithmic Trust Basis

Shared Distributed Ledgers



Autonomic Trust Basis

Cryptographic Proofs



Key Event Receipt Infrastructure

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

Samuel M. Smith Ph.D.
sam@samuelsmith.org
https://keri.one
version 2.59
2021/03/09

Open Standard Protocol Design History

LonTalk Protocol (1988+)

CEA/EIA/ANSI 709.1 - .4

EN/ISO/IEC 14908.1 - .4

ISO/IEC JTC 1/SC 6.

GB/Z 20177.1-2006

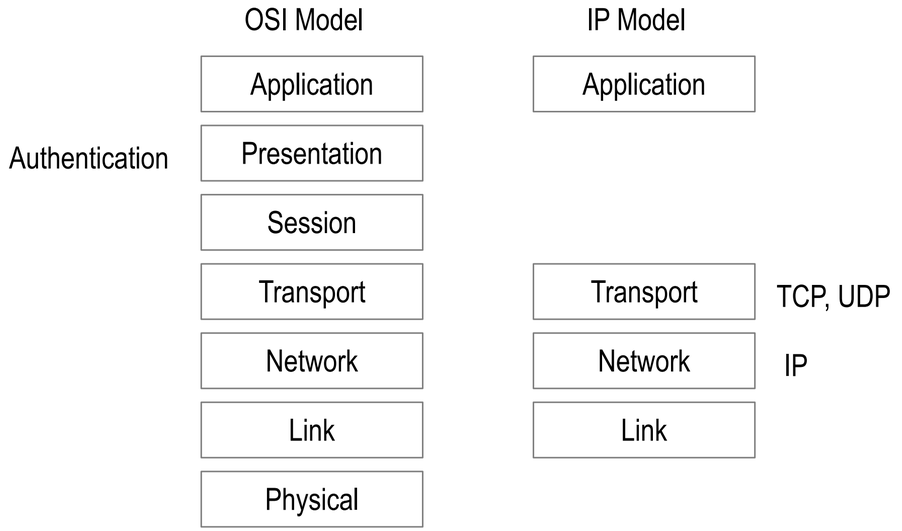
IEEE 1473-L

SEMI E54

IFSF

OSGP

NASA FAA DO-178B 709.1



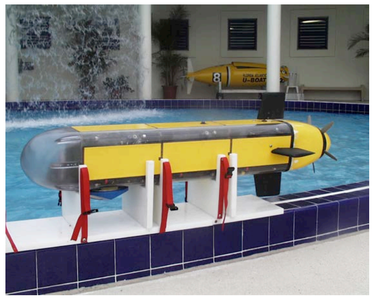
LonTalk over IP (UDP)

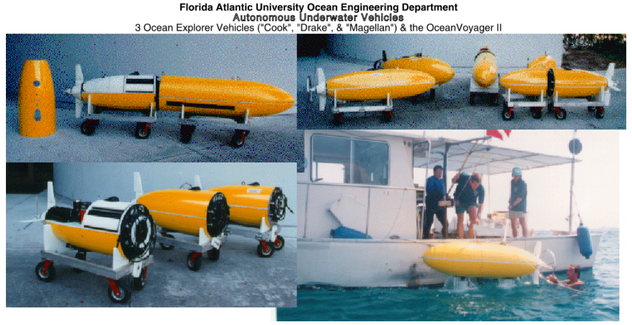
CEA/EIA/ANSI 852 A B C

CEA/EIA/ANSI 852.1

RAET (Reliable Asynchronous Event Transport) Protocol

Reliable secure UDP





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.

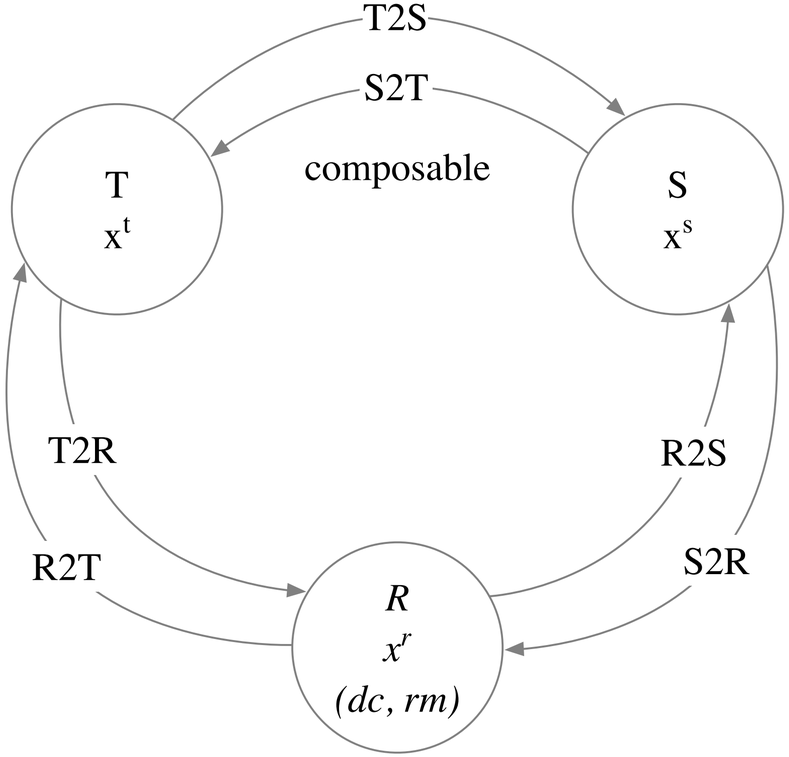
Normally composability requires pad characters (pad bytes) on each Base64 (Base2) primitive.

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.

Basic Primitives

Appear in namespace domain:

Name-spaced identifiers

Ephemeral identifiers

VCs

Documents

Message bodies

Derivation Code	Description	Code Length	Index Length	Total Length	Max Count
	Basic Codes				
	One Char 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	
	Two Char 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	
	Four Char 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 bytes	4		8	
1AAG	DateTime Base64 custom encoded 32 char ISO8601	4		36	

Indexed Codes

Appear in message attachments as stream group:

Compact list is signatures indexed by offset into key or witness list.

KERI Master Base64 Char Derivation Code Table

Derivation Code	Description	Code Length	Index Length	Total Length	Max Count
	Indexed Codes				
	Indexed Two Char Codes				
A#	Ed25519 indexed signature	2	1	88	63
B#	ECDSA secp256k1 indexed signature	2	1	88	63
	Indexed Four Char Codes				
0A##	Ed448 indexed signature	4	2	156	4,095
0B##	Label Base64 char of variable length L=N*4 where N is value of index, total = L+4	4	2	Variable	4,095

Group Count Codes

Appear in message attachments to group and/or pipeline heterogenous attachments and future composable message format:

KERI Master Base64 Char Derivation Code Table

Derivation Code	Description	Code Length	Index Length	Total Length	Max Count
	Counter Codes				
	Four Char Counter Codes				
-A##	Count of attached qualified Base64 indexed controller signatures	4	2	4	4,095
-B##	Count of attached qualified Base64 indexed witness signatures	4	2	4	4,095
-C##	Count of attached qualified Base64 nontransferable identifier receipt couples pre+cig	4	2	4	4,095
-D##	Count of attached qualified Base64 transferable identifier receipt quadruples pre+snu+dig+sig	4	2	4	4,095
-E##	Count of attached qualified Base64 first seen replay couples fnu+dts	4	2	4	4,095
-F##	Count of attached qualified Base64 transferable identifier indexed controller sig groups each triple pre+snu+dig of signer est. event followed by counter -A## followed by indexed controller sigs	4	2	4	4,095
-U##	Count of qualified Base64 groups or primitives in message data	4	2	4	4,095
-V##	Count of total attached grouped material qualified Base64 4 char quadlets	4	2	4	4,095
-W##	Count of total message data grouped material qualified Base64 4 char quadlets	4	2	4	4,095
-X##	Count of total message data plus attachments grouped qualified Base64 4 char quadlets	4	2	4	4,095
-Y##	Count of qualified Base64 groups or primitives in group. (context dependent)	4	2	4	4,095
-Z##	Count of grouped material qualified Base64 4 char quadlets (context dependent)	4	2	4	4,095
-a##	Count of anchor seal groups in list (anchor seal list) (a)	4	2	4	4,095
-c##	Count of config traits (each trait is 4 char quadlet (configuration trait list) (c)	4	2	4	4,095
-d##	Count of digest seal Base64 4 char quadlets in digest (digest seal (d)	4	2	4	4,095
-e##	Count of event seal Base64 4 char quadlets in seal triple of (event seal) (i, s, d)	4	2	4	4,095
-k##	Count of keys in list (key list) (k)	4	2	4	4,095
-l##	Count of locations seal Base64 4 char quadlets in seal quadruple of (location seal) (i, s, t, p)	4	2	4	4,095
-r##	Count of root digest seal Base64 4 char quadlets in root digest (root digest) (rd)	4	2	4	4,095
-w##	Count of witnesses in list (witness list or witness remove list or witness add list) (w, wr, wa)	4	2	4	4,095
	Eight Char Counter Codes				
-0U#### #	Count of qualified Base64 groups or primitives in message data	8	5	8	3,741,823
-0V#####	Count of total attached grouped material qualified Base64 4 char quadlets	8	5	8	3,741,823
-0W#### #	Count of total message data grouped material qualified Base64 4 char quadlets	8	5	8	3,741,823
-0X#####	Count of total group message data plus attachments qualified Base64 4 char quadlets	8	5	8	3,741,823
-0Y#####	Count of qualified Base64 groups or primitives in group (context dependent)	8	5	8	3,741,823
-0Z#####	Count of grouped material qualified Base64 4 char quadlets (context dependent)	8	5	8	3,741,823
-0a#####	Count of anchor seals (seal groups in list)	8	5	8	3,741,823

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.

Annotated Example

```
-FAB                                                # Trans Indexed Sig Groups counter code with 1 following group
E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y    # trans prefix of signer for sigs
-EAB0AAAAAAAAAAAAAAAAAAAAAAB                       # 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 with 3 following sigs
AA5267UlFg1jHee4Dauht77SzGl8WUC_0oimYG5If3SdIOSzWM8Qs9SFajAilQcozXJVnbkY5stG_K4NbKdNB4AQ  # sig 0
ABBgeqntZW3Gu4HL0h3odYz6LaZ_SMfmITL-Btoq_7OZFe3L16jmOe49Ur108wH7mnBaq2E_0U0N0c5vgrJtDpAQ  # sig 1
ACTD7NDX93ZGTkZBBuSeSGsAQ7u0hngpNTZTK_Um7rUZGnLRNJvo5oOnnC1J2iBQHuxoq8PyjdT3BHS2LiPrs2Cg  # sig 2
```

Partially Annotated Example

{"v":"KERI10JSON000154_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"0","t":"icp","kt":["1/2","1/2","1/2"],"k":["DaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg","Duzj-
Z2lR2DqB0cI042loSMUVWOrN5axojx8g9fSx3PM","DRXPAmNVVqafWvQiN5qQmWUDvVupF2w8xFNGg1Gays9Y"],"n":"EO5f_IQjtBoeN_-OyzfVJx1_WqBFUL-Ely4x-xmUtOW8","wt":"0","w":[],"c":[]}

-VEM

-AAD

AA6Z50BRlXby_uSdkqbybLXds-5OMwQil4miux1sRxJkiD3kRS4HuCpv5m-wwsPHWwn_Ku5xB2P—NJ1pl7KXjAQ

ABDjMdRtemkn9oykLFo9MBwZsS85hGd1yaMMdFb_P1FY8_PZcHBVTc2iF5Bd6T2rGorwS-ChRa24bxUrkemWD1DA

ACpxUYq2zrFAlMdWuxdaYTqvh12pgk4Ba-vllsaZP5ct5HcOtJw47B6cVLcEePwEHk6jHlSoDGgH2YiyOwPbgSBQ

-DAD

E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y

0AAAAAAAAAAAAAAAAAAAAAAA

EFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRY

AA6Z50BRlXby_uSdkqbybLXds-5OMwQil4miux1sRxJkiD3kRS4HuCpv5m-wwsPHWwn_Ku5xB2P—NJ1pl7KXjAQ

E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y

0AAAAAAAAAAAAAAAAAAAAAAA

EFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRY

ABDjMdRtemkn9oykLFo9MBwZsS85hGd1yaMMdFb_P1FY8_PZcHBVTc2iF5Bd6T2rGorwS-ChRa24bxUrkemWD1DA

E_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y

0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACpxUYq2zrFAlMdWuxdaYTqvh12pgk4Ba-vllsaZP5ct5HcOtJw47B6cVLcEePwEHk6jHlSoDGgH2YiyOwPbgSBQ-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0B6Z50BRlXby_uSdkqbybLXds-5OMwQil4miux1sRxJkiD3kRS4HuCpv5m-wwsPHWwn_Ku5xB2P—NJ1pl7KXjAQ-
EAB0AAAAAAAAAAAAAAAAAAAAAA1AAG2021-04-22T17c15c03d117401p00c00{"v":"KERI10JSON000098_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"1","t":"ixn","p":"Egd_fi2QDtfjjdB5p9tT6QCHuBsSWUwQP6AbarcjLgw0","a":[]}-VEM-
AADAAPLMNHELcDDuPT1gyI9_TEBM6FRji2xmc0iBfNBwoKJttbJfeQhH41y-ayubtyhyMzHaqrq-WXaNQkpnzTTOPBAABUawpt1Nd7GR9rTwPD4ucT-M7Vy1xuxGlgRf9pgkOcXBBbhomjjEpz3aid9PP2vWeJ_rvw7W5rgrTJ38Q2v8bDwACoHNjlZ-IZ1K9opgeu33TNIFBd3rNW_gKO_bFa-
t2GYwOzlWoDlzF7kSRQnVKlXMeVrLBe3uwO6PjYjeZdUSlDg-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAAPLMNHELcDDuPT1gyI9_TEBM6FRji2xmc0iBfNBwoKJttbJfeQhH41y-ayubtyhyMzHaqrq-
WXaNQkpnzTTOPBAE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYABUawpt1Nd7GR9rTwPD4ucT-
M7Vy1xuxGlgRf9pgkOcXBBbhomjjEpz3aid9PP2vWeJ_rvw7W5rgrTJ38Q2v8bDwE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACoHNjlZ-IZ1K9opgeu33TNIFBd3rNW_gKO_bFa-
t2GYwOzlWoDlzF7kSRQnVKlXMeVrLBe3uwO6PjYjeZdUSlDg-CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0BPLMNHELcDDuPT1gyI9_TEBM6FRji2xmc0iBfNBwoKJttbJfeQhH41y-ayubtyhyMzHaqrq-WXaNQkpnzTTOPBA-
EAB0AAAAAAAAAAAAAAAAAAAAAAQ1AAG2021-04-22T17c15c03d143279p00c00

{"v":"KERI10JSON000190_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"2","t":"rot","p":"E8MU3qwR6gzbMUqEXh0CgG4k3k4WKkk9hM0iaVeCmG7E","kt":["1/2","1/2","1/2"],"k":
["DIsi8qYso1KMmpLOGYty3NC2BAojZmUCfzR5_5oQRiso","DkdClpaWCAoCPBYgUmqP9gwAtsGq81yyPhGQKQ6-W_F0","DKDyq4QQYKnx9ircxeCvEcraI4HUSr_ytWPelDHAM98w"],"n":"EloOvJmwenmC4uHjX7qB40LGVbeZY5rYQeZ6IK5zmdmM","wt":"0","wr":[],"wa":[],"a":[]}

-VEM

-AAD

AAr5HeTAGJ_WfIMO82lbEnpAMkuqZ0iJO0yYhjwvLElPYltF_jSOApKPWxepare2O7XMMOvtgxjXj9pvvqpW8WDgABKHoueBd4JgakpVydJYADwh5wMSNyHNMKXwhYMGrgApl_EvsTmEt8uS94PmrfCtRjLRbZdzLRZVkX7Yx4jlNNCgACjKJlODGhL_a0S3-oDRJhOUG0sul4SCJd21Qp-
KSFSfGavACAwQdEYQL43jko91FDuhwKDt1BD8kAoy3T-tdoAw-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAA7JJAxJL3nhVur7YboCK2zPSmx_AaYDYeN7UsvoKcZKrYbuScUje_qfx_e9z1SM4tm8bUbYJnLXTz8dOta9ZiDwE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYABi7dsjnldn7E-L56Rlz4ZWp8XC5y8v7h4XRoZp2sO69H84dhyRM27UE9_egCWQZJ_MHJKVA5g2s0hXmXvjSKrAQE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACo0JcZmUhiNBfb_3bBwgX7KfN52vmazAzEFgJlr8wNfXSvF6rA5lED4J1EWuEnhA00vUHQqPrjk78nnRBBZlVAA-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0B7JJAxJL3nhVur7YboCK2zPSmx_AaYDYeN7UsvoKcZKrYbuScUje_qfx_e9z1SM4tm8bUbYJnLXTz8dOta9ZiDw-
EAB0AAAAAAAAAAAAAAAAAAAAAg1AAG2021-04-22T17c15c03d150633p00c00{"v":"KERI10JSON000098_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"3","t":"ixn","p":"EO2hh7xg29y3i7uywQ_n0g7vk0W1oGiErUY9QpGjSUhc","a":[]}-VEM-
AADAA5Iox67c4HL78UrYqpSNH-UkHZRpR7X0fPQ0GEYJG8OGqCHBvPJica_yohOQP8GNOFQ9UsmBa0TDji6EAaXivBwAB6BgG2CQ-Ukw8CchtCHf9L5kVsmg1Tu2OuLkcy9Sb9uVm23yLx-8I4pc6KHmZke8KCvpXjcdV65gOOE-
VUIMOBwACXtTZoFqJHFhoMZABunXETksrK1nNiP9xzXx13gl4uqoVZkqfwqUTL3C7q0RcxYwaz5sYSNQA8zblA8YxVyFuCQ-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAAG1L04T2jREp2pizW-
jQ0tglZ8I4CDNoKx4bN2K0ztuf_0ywQ29p2kFkBVZaRPwljOZlUzJqlPU6P2R-IVORJBQE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAB2ss-isfVr2WpdCWWNxO_9N75eJK-2CZp1J-DicWd8FqziZIc-
kAmxNBD9TjxfuYn7pQmXnaWF7g4RhCLJGBuDAE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACrxgx3QlrBs-g369i807ntd8rGWGC4WGrrdy60cPy9hjrP10qjDtSTwa2UZPNVEUZolM-lHsFqoNhjeaHmg_mDA-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0BG1L04T2jREp2pizW-jQ0tglZ8I4CDNoKx4bN2K0ztuf_0ywQ29p2kFkBVZaRPwljOZlUzJqlPU6P2R-IVORJBQ-
EAB0AAAAAAAAAAAAAAAAAAAAAAw1AAG2021-04-22T17c15c03d154307p00c00{"v":"KERI10JSON000098_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"4","t":"ixn","p":"EQI0EXdK6WvQae17PBWDUkMOdOiTPpx48oMSYTUYsCl0","a":[]}-VEM-
AADAAbnPY1i0cpo6q0cmvQr2bZOcipzl7LYY2h-3ixndlzB3f-4VFLzSnIUtB_qwp1H2NI_DNGqXWGACywJoxkFccAQABHDicUliz3Bl6y1T7-sQteMKxoDYZ4A8hVx3p3EjztyO8UnA6PkaV2b7AFwAfk4UbBWKMGjTtpZ88S7P9EsXLBAACNFFh6nDIWNG1ZbEsqqlCG2aKLgnpHmR6cJr1dq1F4pylAF1e3-
on2aasDMYk3c2fj-AWErRqbsf8ejnJE3YvDg-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAAh0E0mltmkUz1_TXMirWFa67IGAaK7fThhrJ8TQyuhY7usunzf8VtWfaaLBQSpofhgppsMlf3zZxDS1g6t-7PCgE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYABECiScuPby_LbGw5s6qNTJQm2m6Dqbsig7sRdk841XWU6hV3MlD-k_SriiPEJWMAWDmY74lM-UiNDvnmN4OAJCAE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACSc48sfSvNtYByMlUQsMPdEsDw5Z6oDX4jlZ9F5eCMcRvYWWApAD-OOi85JTIiW3y3nSdbfyt4vS6YvroA68MAQ-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0Bh0E0mltmkUz1_TXMirWFa67IGAaK7fThhrJ8TQyuhY7usunzf8VtWfaaLBQSpofhgppsMlf3zZxDS1g6t-7PCg-
EAB0AAAAAAAAAAAAAAAAAAAABA1AAG2021-04-22T17c15c03d158167p00c00{"v":"KERI10JSON000098_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"5","t":"ixn","p":"EvrAC5XVQyu01ZuKfq1wiR0kXF2j8TjrCg4QyA0LVjKk","a":[]}-VEM-
AADAA1OJn3UHjLcI333fduqTj6nAJY27VtkQqW_lHalnJKtgmb0tklV5xUCVzpal14xWDuyCdImhFzTk0sRgW4MYDQABOR8ay9qQYR3ieant4ujM_FX0Nm_mUHcVVo4pCqDy8jLaM3EBNmkOKUIfxgZC-8k6OpYcy33gC-
qgUpc6C2_PDwACSoZSibaYcin32vY4ANzflFpJh_EF7mcGbTWSFrNLnwFrrOfhXL3i1Pf39Sk079ApSI87Nt-CvHpRRdows3TABQ-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAAgXtG2I3AxvU5yHKzfucOKOvxeKWwChKQvEQJtJnz9iIpsXqyyrgRfOyoyjhk73D-E3FbDg_3k1XK_3i-yDWeAQE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAByUVjq4Y_sMWi9iqqWXTo2ES5pBMlBgJbAY3h61aJElQdCIxr2ldx_BSq4vA-FlELEBUkSbeHnHGXeFfVi6AjCwE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAC6GmjxPFClVsY7smEcpmptQnZgET9LUO606SzhkCaGCe1jR2KZ3vNsgitA_7OQ_VDipLwoWGv_Kz2YnUkjKFsCw-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0BgXtG2I3AxvU5yHKzfucOKOvxeKWwChKQvEQJtJnz9iIpsXqyyrgRfOyoyjhk73D-E3FbDg_3k1XK_3i-yDWeAQ-
EAB0AAAAAAAAAAAAAAAAAAAABQ1AAG2021-04-22T17c15c03d161436p00c00{"v":"KERI10JSON000098_","i":"E4ReNhXtuh4DAKe4_qcX__uF70MnOvW5Wapj3LcQ8CT4","s":"6","t":"ixn","p":"EwmQtlcszNoEIDfqD-Zih3N6o5B3humRKvBBln2juTEM","a":[]}-VEM-
AADAAvYMCRmJgjFM7EG7rWng7Q3WRfwcd908UdKL-7ZfGw4igpF9DcA-yxwliba59D4pkmhIcrW_Ax76iuaD6yD03BwAB9Wp-awBUfw2jnDRjvEU3xpFlLDHwiFLRKpom8Wnx7qDD4aEv6ERZh-
H8yP3eL4sNEFjP5HcRrb5MpFwOp0VyAwACdedbq9E2Exs1NobGwSNQpNxKlgDPiNDE8nOeOqgXtIrAj8SAh8gX2pOgEFj3g3UB69dNGw2M-bEZ557-p9G-Aw-DADE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYAA9U_Kq0GNM1fFq1Vgp937kHkwxSBn4nT8UciTepjjOdOAR-hvsLCxQx2V2pbyQo3fubs6mPd6TQ4ZUmXNrtxmBwE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYABuFO678xi0JuYyQWnSOtOVXABknvRo6-0EWFCv7hxucmqgE6Je2R4120G3nFsJ_ImTjkDibQU8m7CYBGcFh-hAQE_T2_p83_gRSuAYvGhqV3S0JzYEF2dIa-
OCPLbIhBO7Y0AAAAAAAAAAAAAAAAAAAAAAAAAAEFSbLZkTmOMfRCyEYLgz53ARZougmEu_edeW-0j2DVRYACBUqcpzMYX373ePKsfKBjt9aXO2vkl9jAb5vBHFYc0h5r-pGL2TIgoyfMPMAf0zFrsKnDdmN0HmSaE1OsP2hmDA-
CABBaYh8uaASuDjMUd8_BoNyQs3GwupzmJL8_RBsuNtZHQg0B9U_Kq0GNM1fFq1Vgp937kHkwxSBn4nT8UciTepjjOdOAR-hvsLCxQx2V2pbyQo3fubs6mPd6TQ4ZUmXNrtxmBw-EAB0AAAAAAAAAAAAAAAAAAAABg1AAG2021-04-22T17c15c03d164449p00c00'

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.

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.