



Key Event Receipt Infrastructure

A Secure Identifier Overlay for the Internet

Samuel M. Smith Ph.D.

sam@keri.one

<https://keri.one>

version 2.49

2020/09/25

Resources

sam@prosapien.com

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

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf

https://github.com/SmithSamuelM/Papers/blob/master/presentations/KERI2_Overview.web.pdf

https://github.com/SmithSamuelM/Papers/blob/master/presentations/DuplicityGame_IIW_2020_A.pdf

<https://github.com/SmithSamuelM/keri>

<https://github.com/SmithSamuelM/keripy>

DIF

Identity and Discovery WG

<https://github.com/decentralized-identity/keri>

<https://github.com/decentralized-identity/keripy>

SSI Meetup

<https://ssimeetup.org/key-event-receipt-infrastructure-keri-secure-identifier-overlay-internet-sam-smith-webinar-58/>

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

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

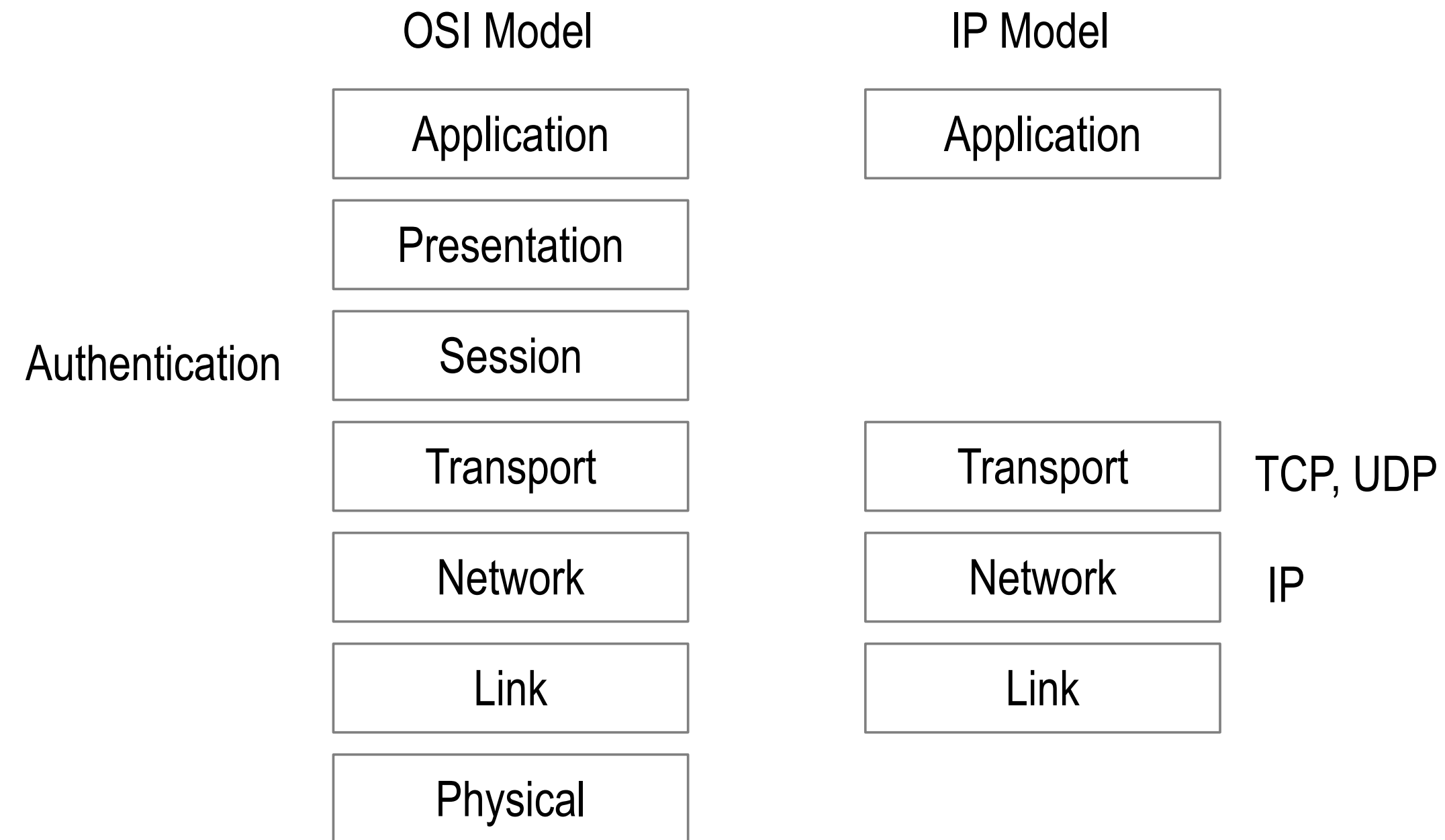
With verifiable digital signatures from asymmetric key crypto –
we may not trust in “*what*” was said, but we may trust in “*who*” said it.

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*.

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

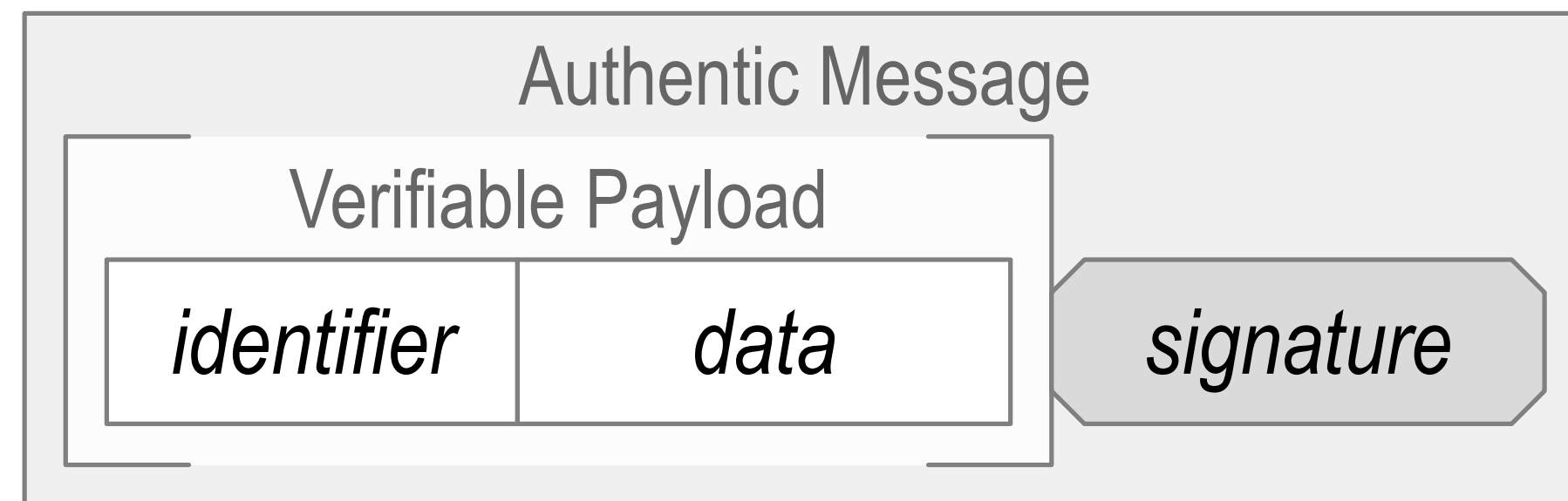
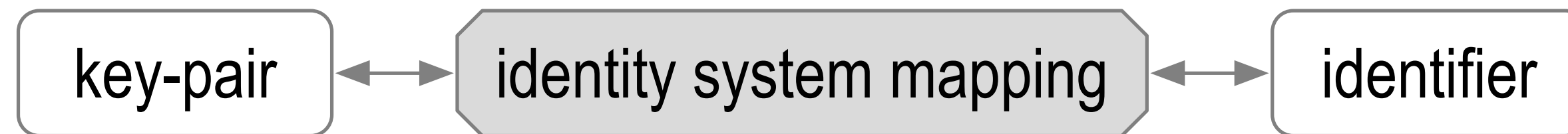


Instead ...

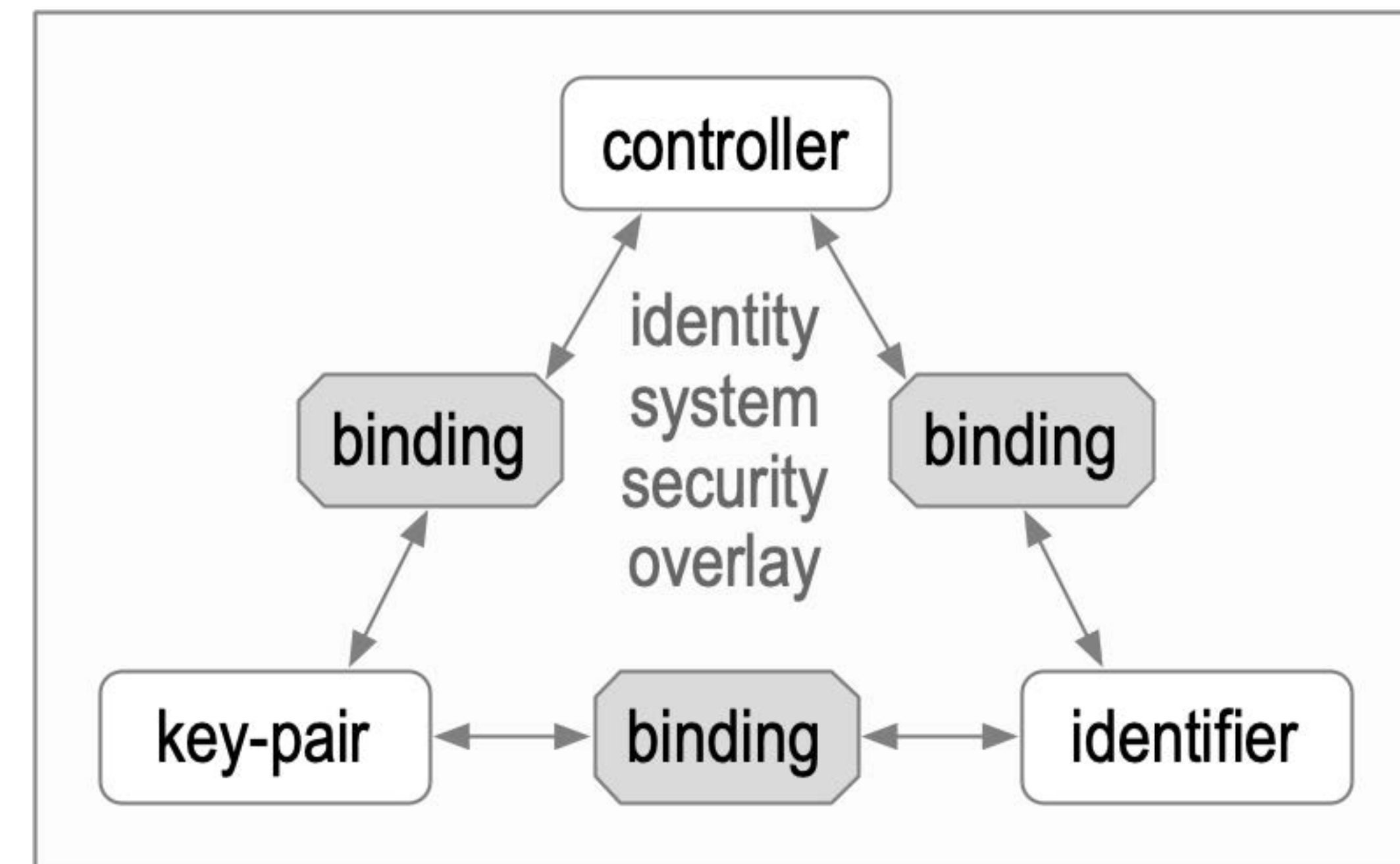
We use *bolt-on* identity system security overlays.
(DNS-CA ...)

Identity System Security Overlay

Establish authenticity of IP packet's message payload.

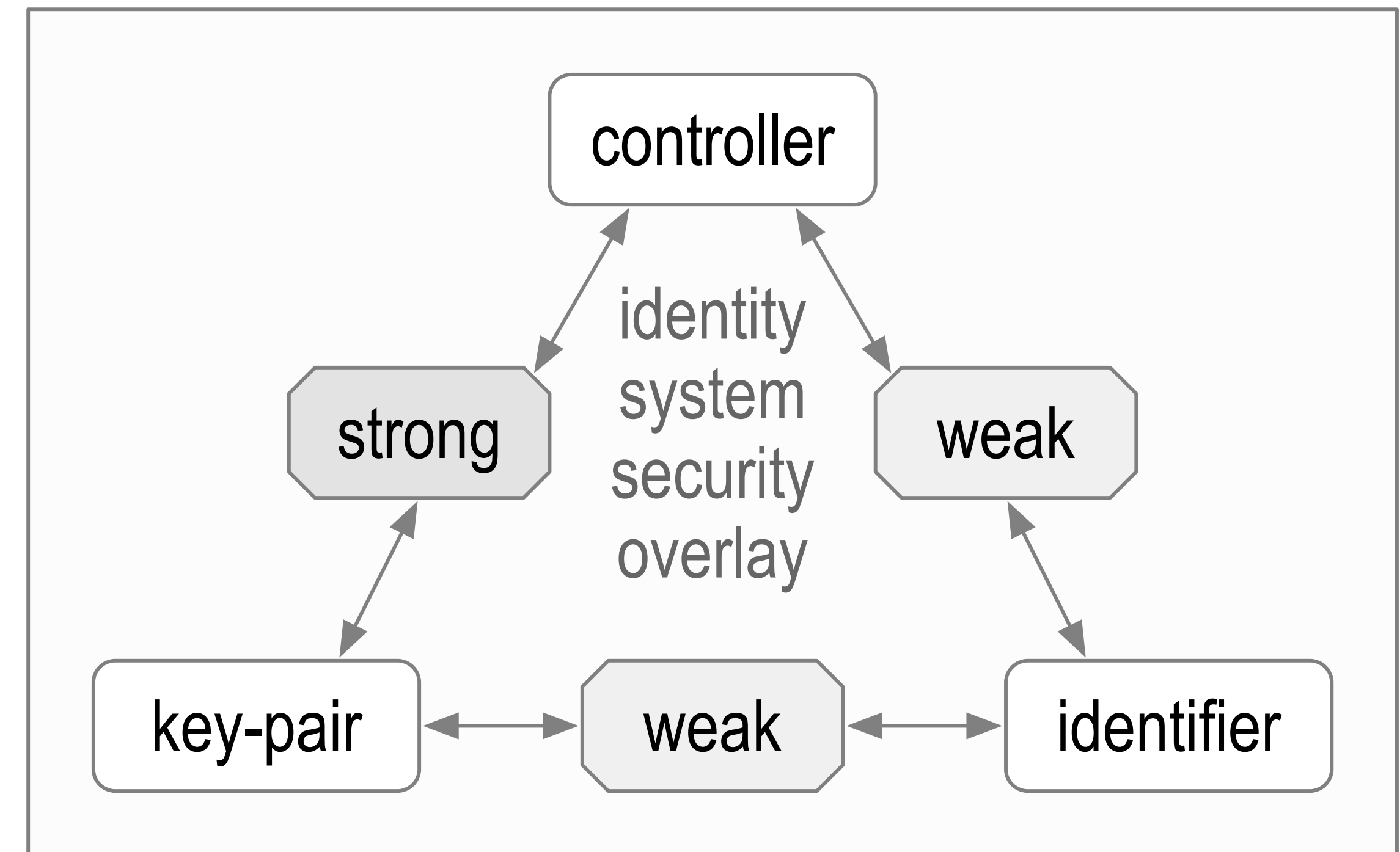
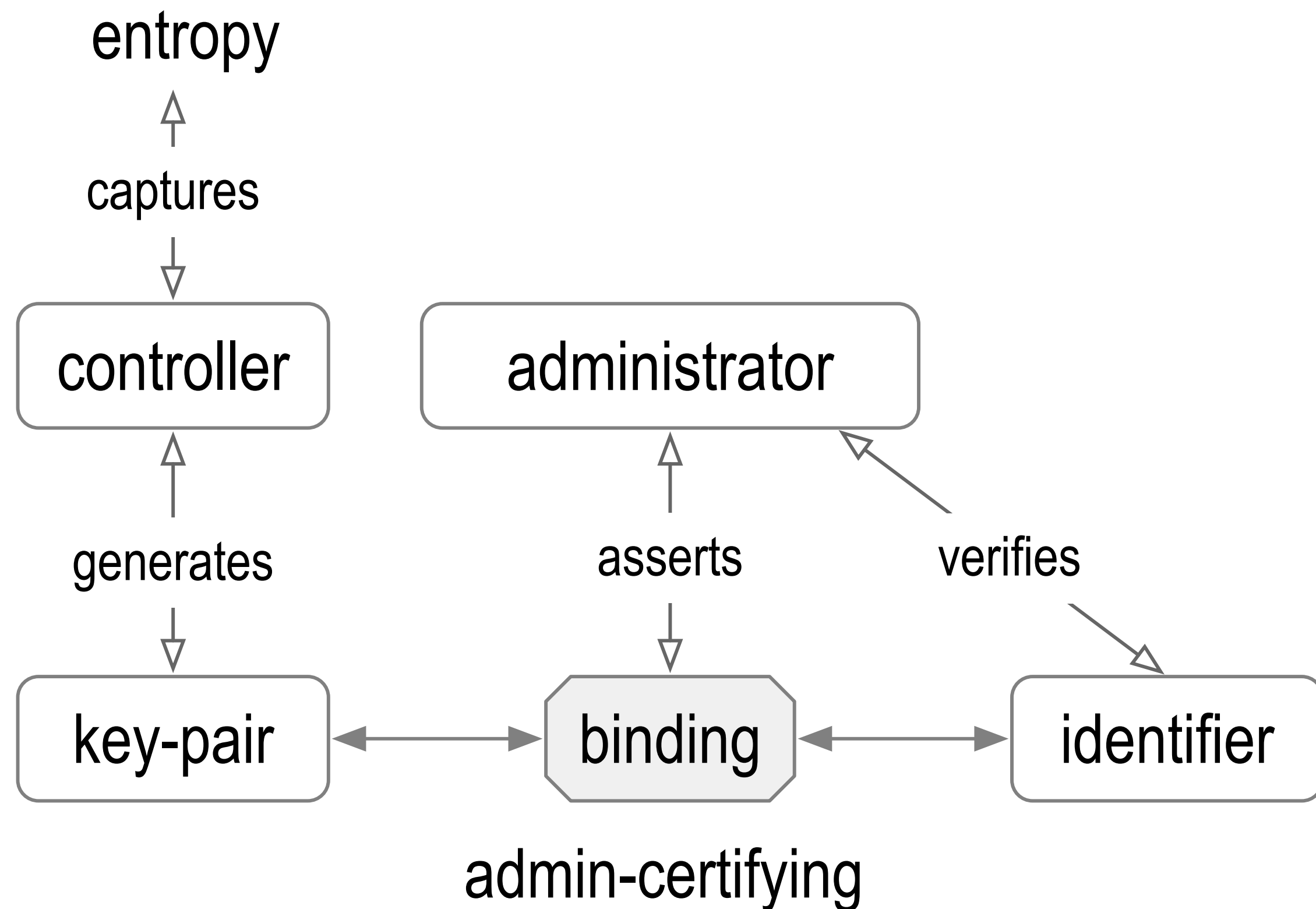


The overlay's security is contingent on the mapping's security.



Identifier Issuance

Administrative Identifier Issuance and Binding

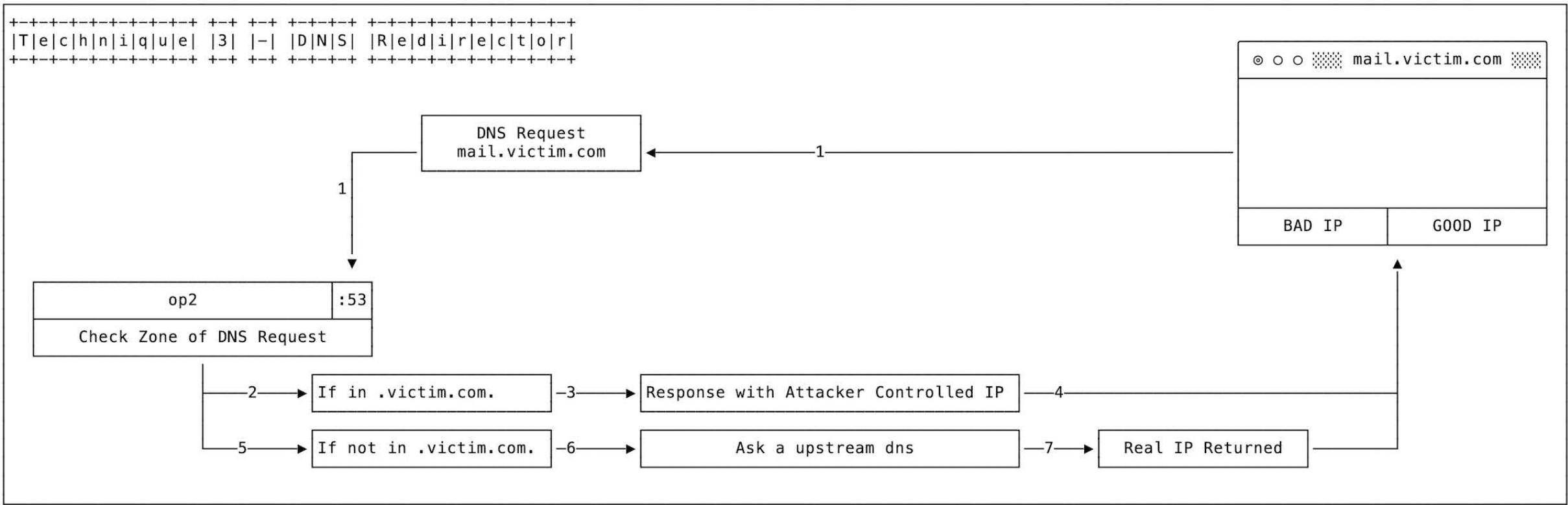
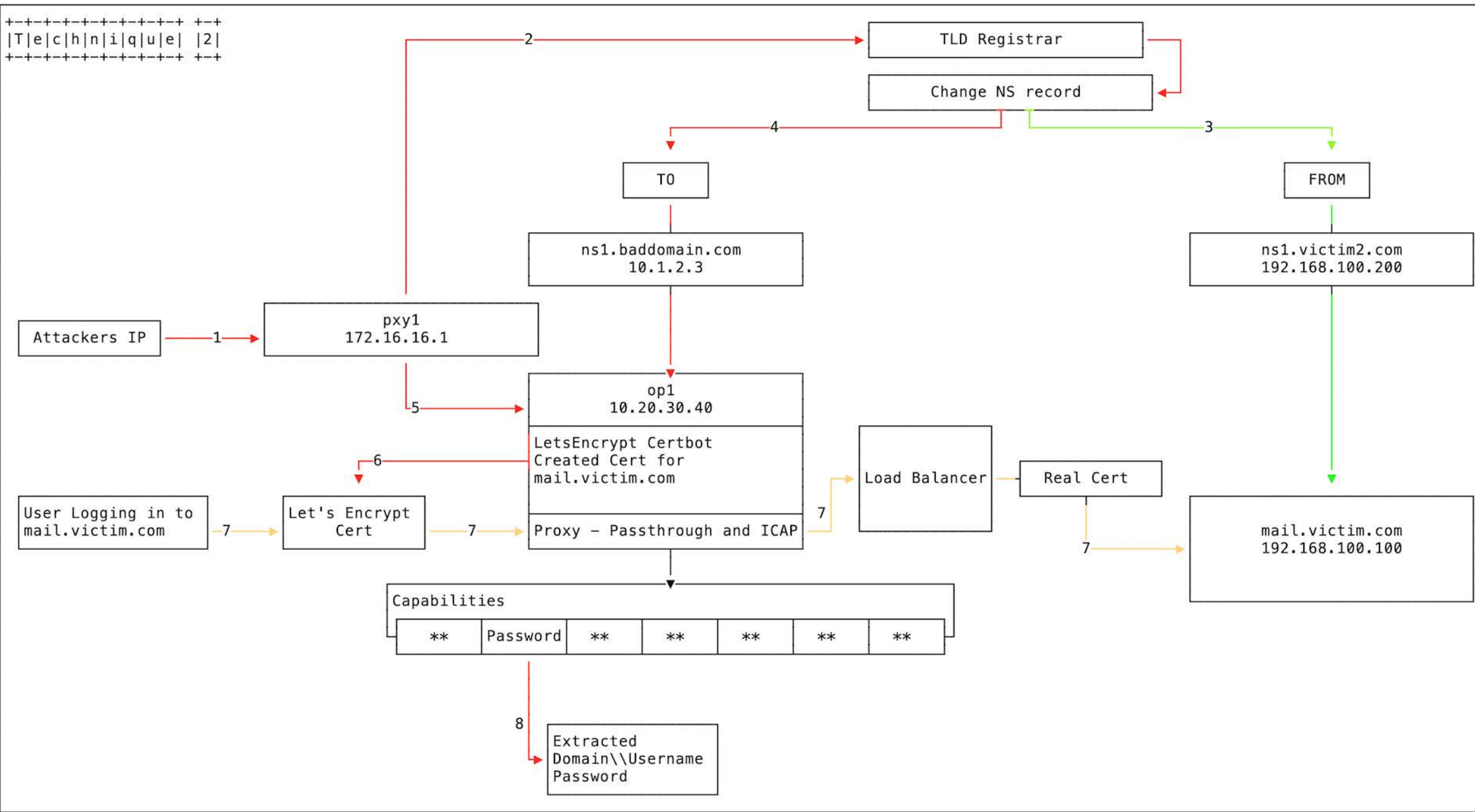
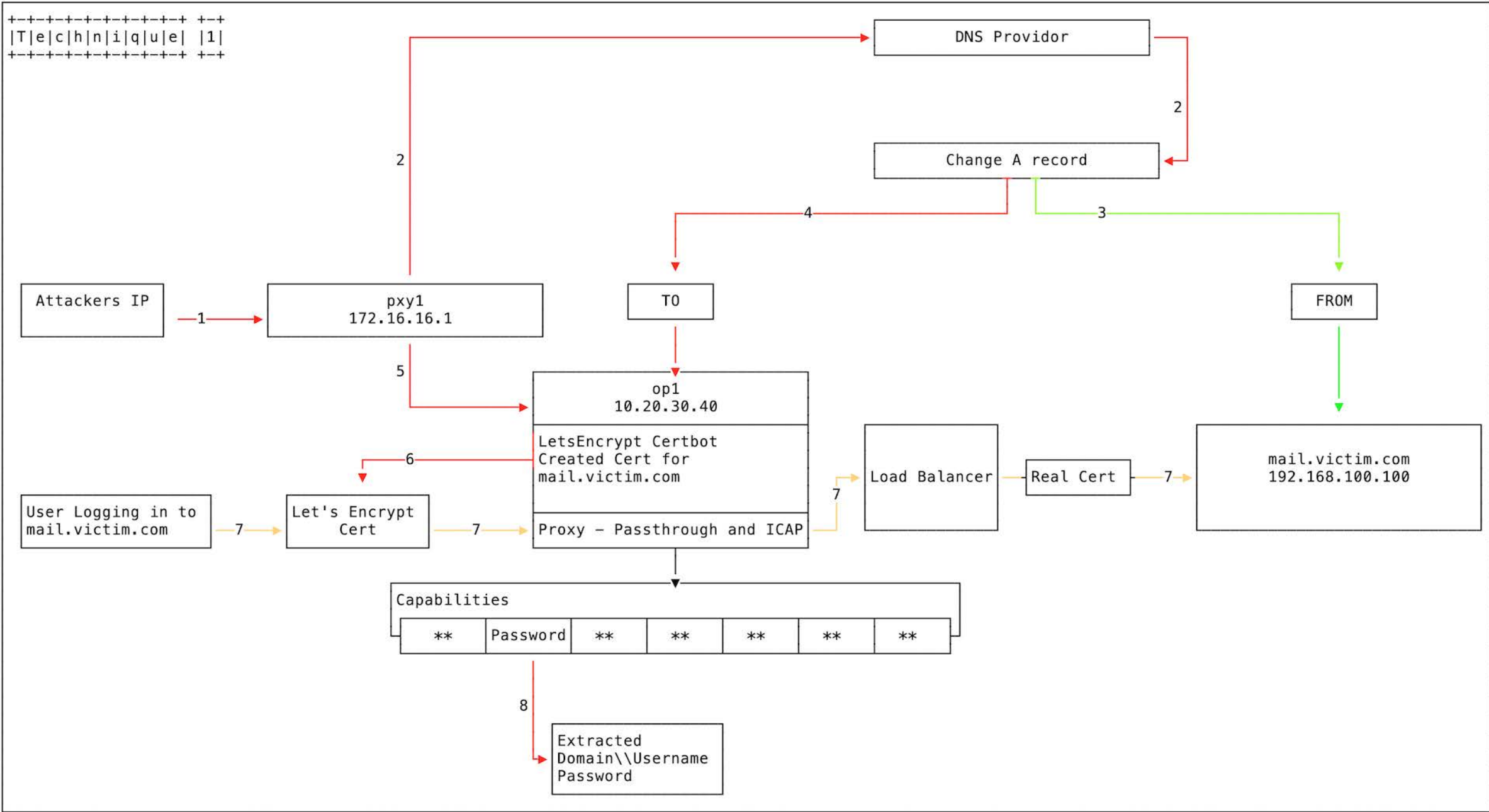


Admin-Certifying Identifier Issuance

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/>



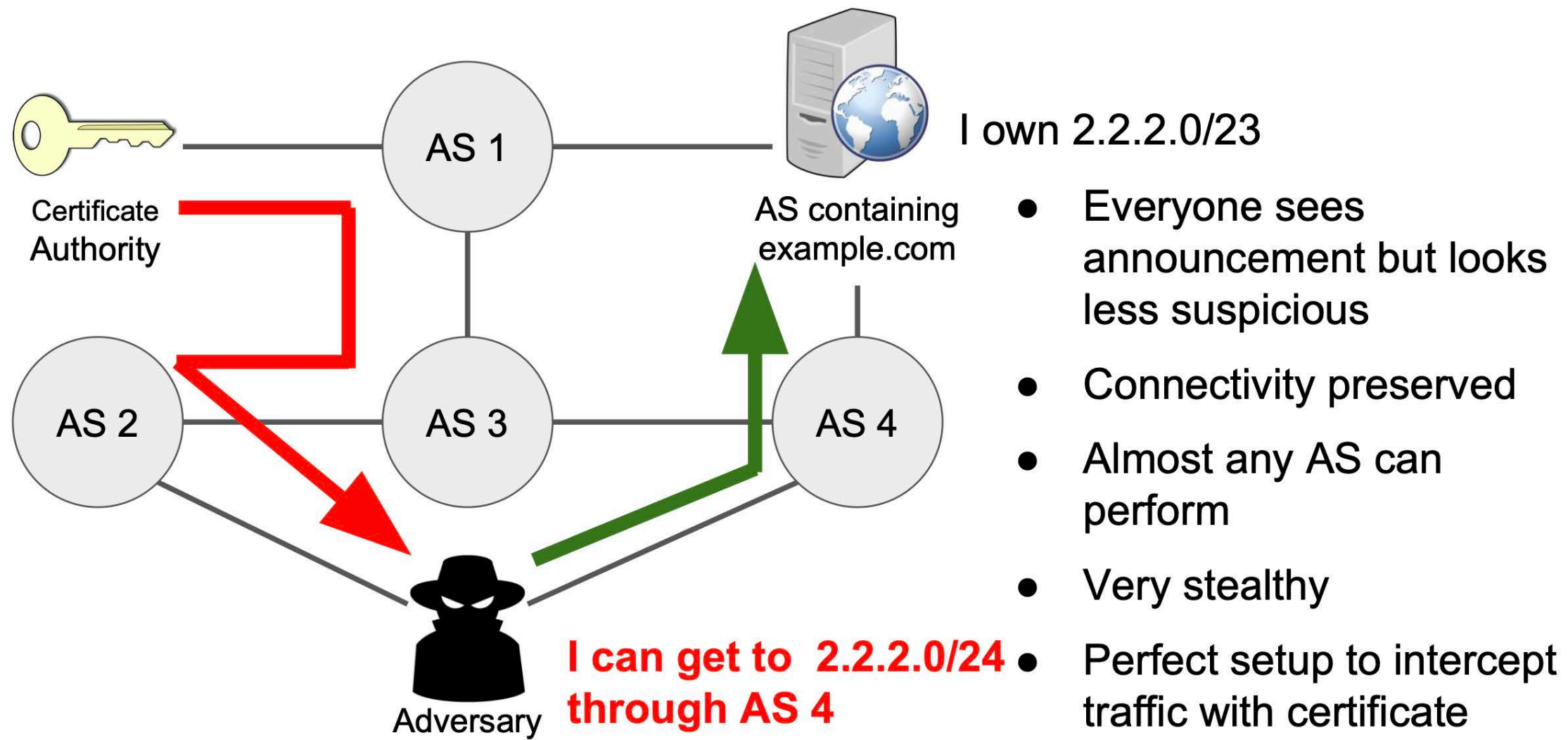
BGP Hijacking: AS Path Poisoning

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>

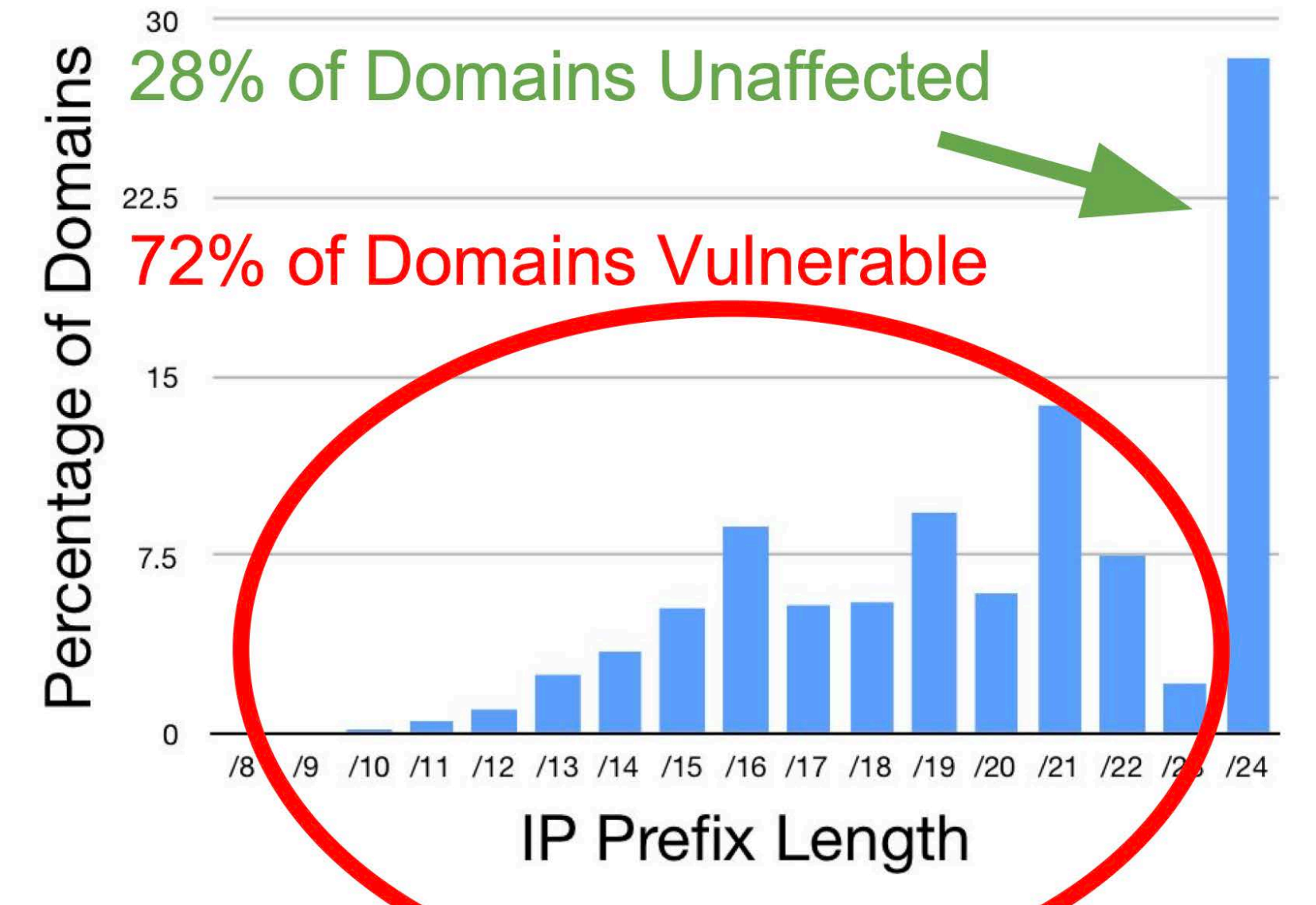
Gavrichenkov, A., “Breaking HTTPS with BGP Hijacking,” BlackHat, 2015 <https://www.blackhat.com/docs/us-15/materials/us-15-Gavrichenkov-Breaking-HTTPS-With-BGP-Hijacking-wp.pdf>

AS path poisoning

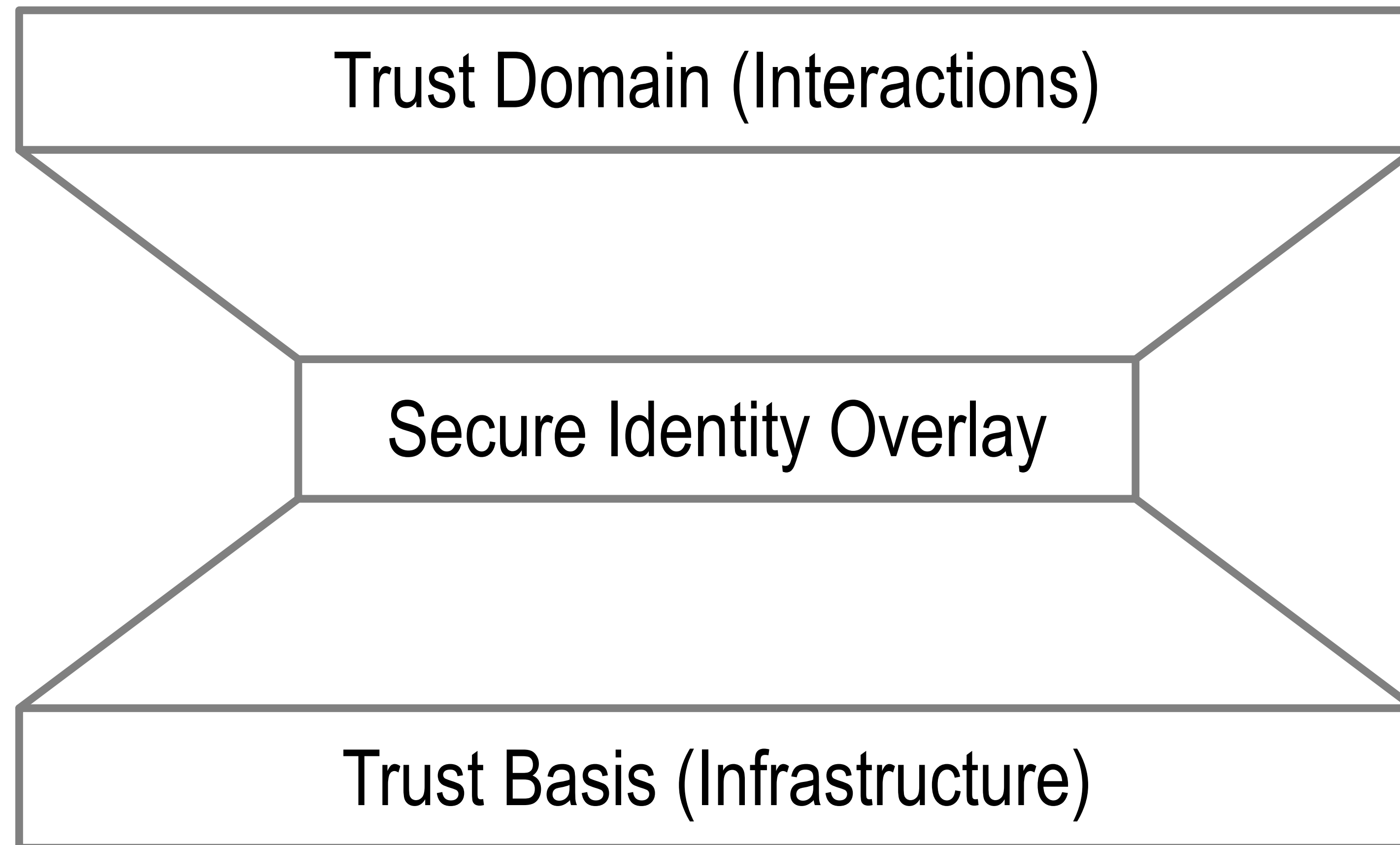


Vulnerability of domains: sub-prefix attacks

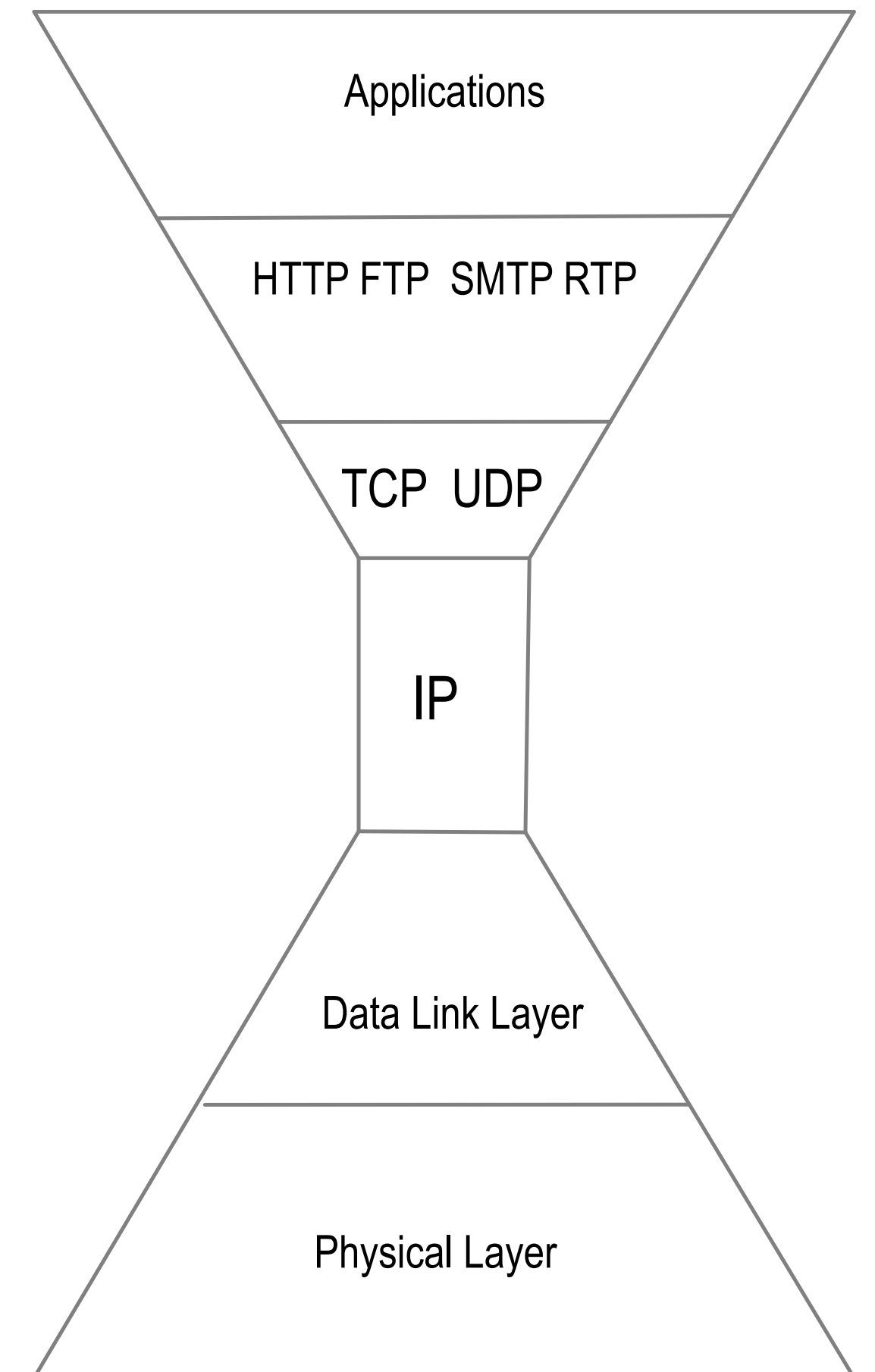
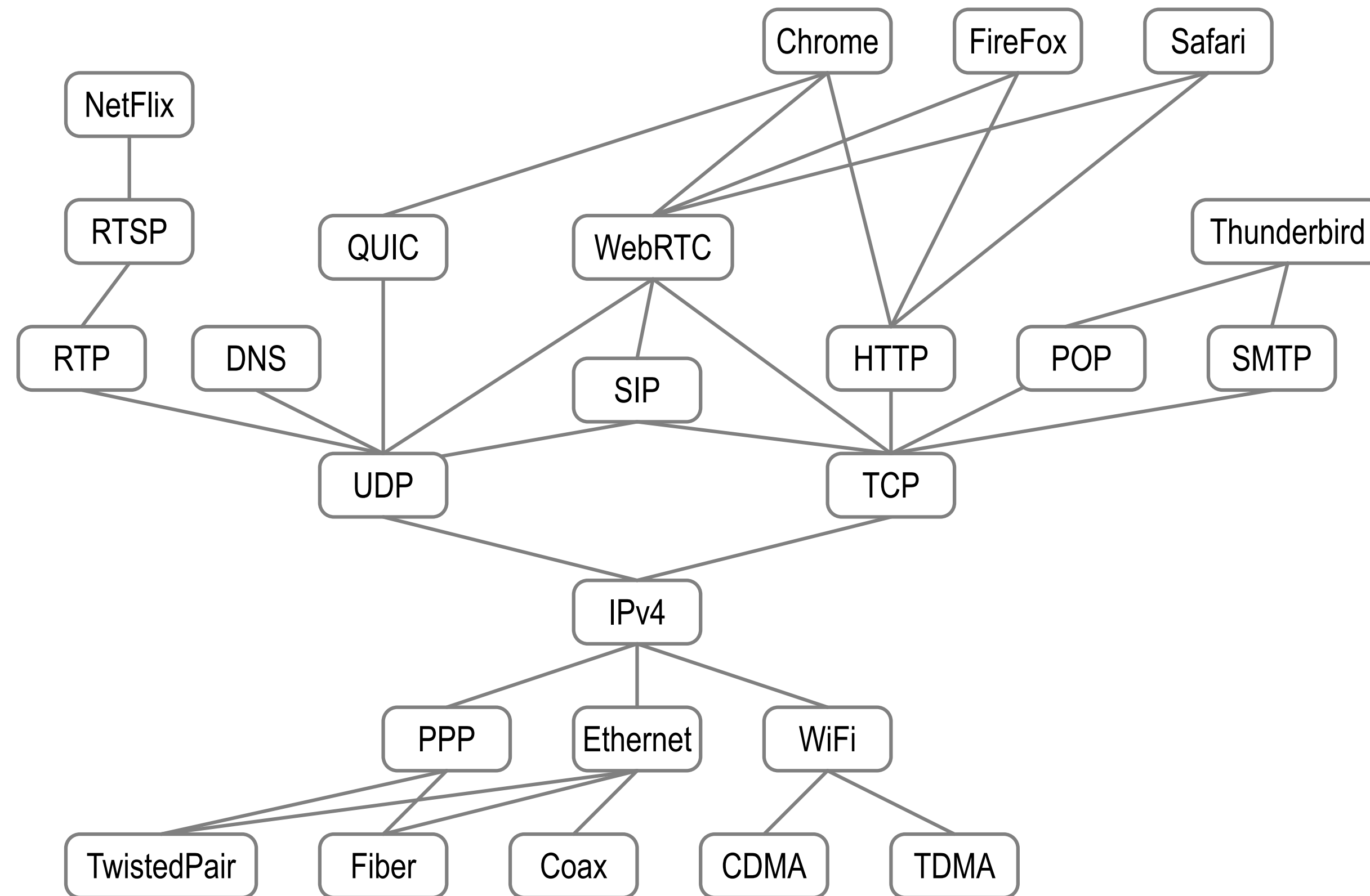
- Any AS can launch
- Only prefix lengths less than /24 vulnerable (filtering)



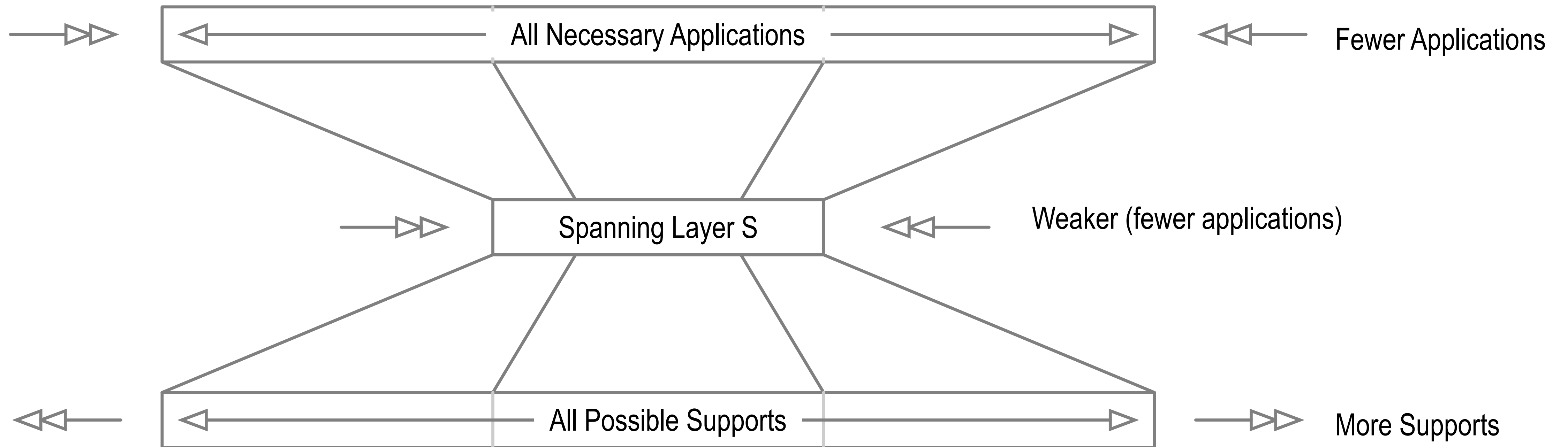
Identity System Security Overlay



Spanning Layer

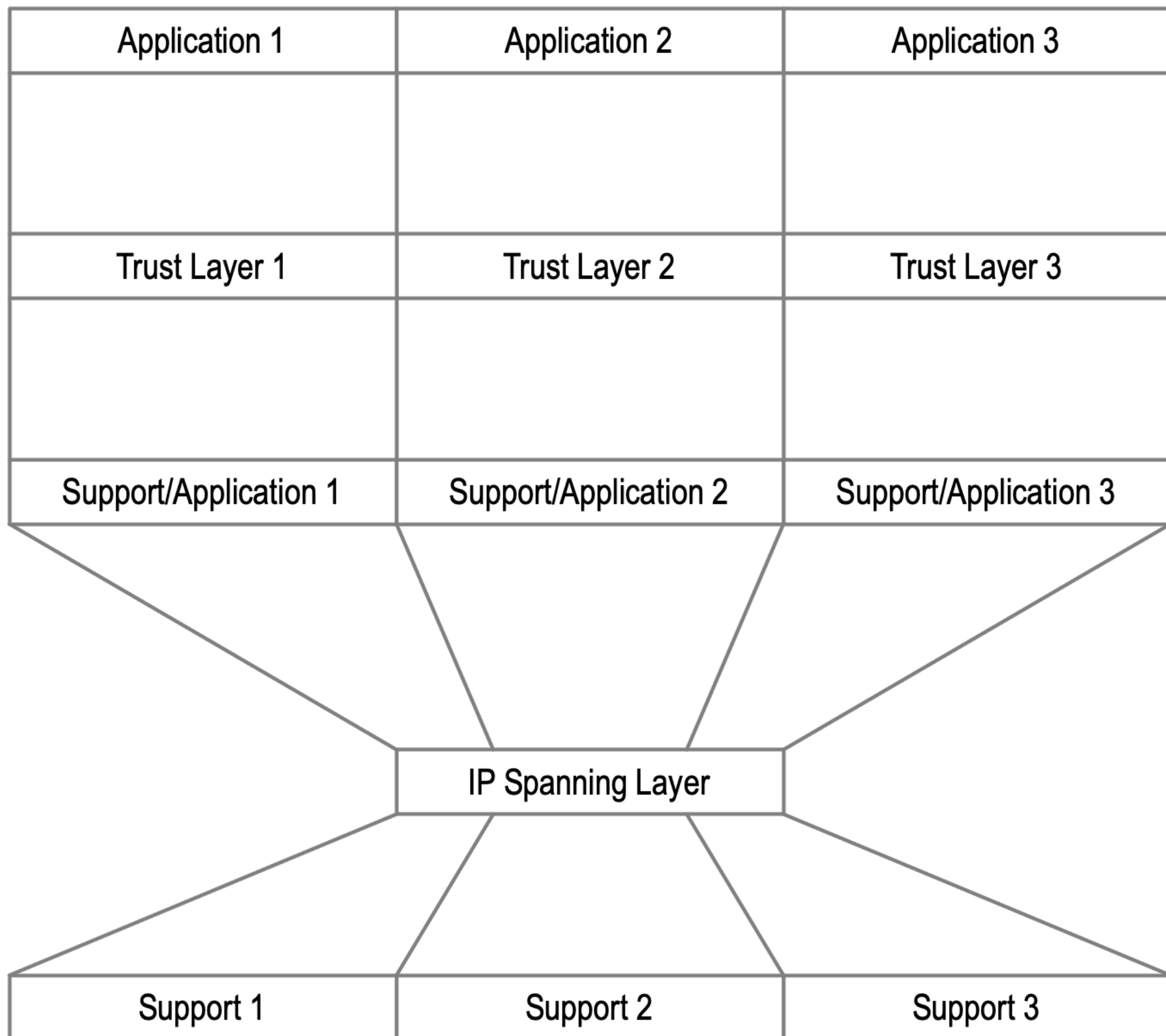


Hourglass

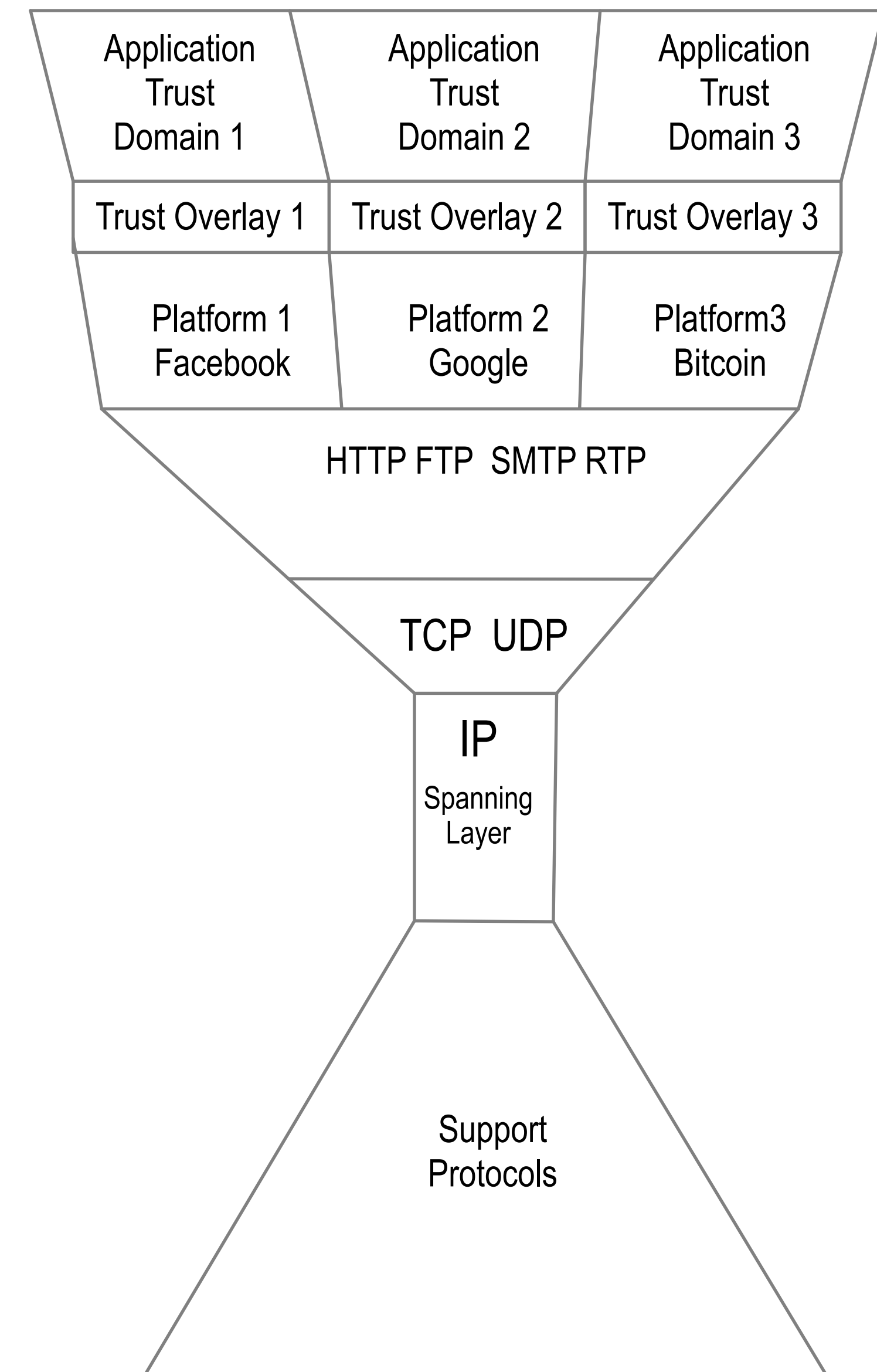


Platform **Locked** Trust

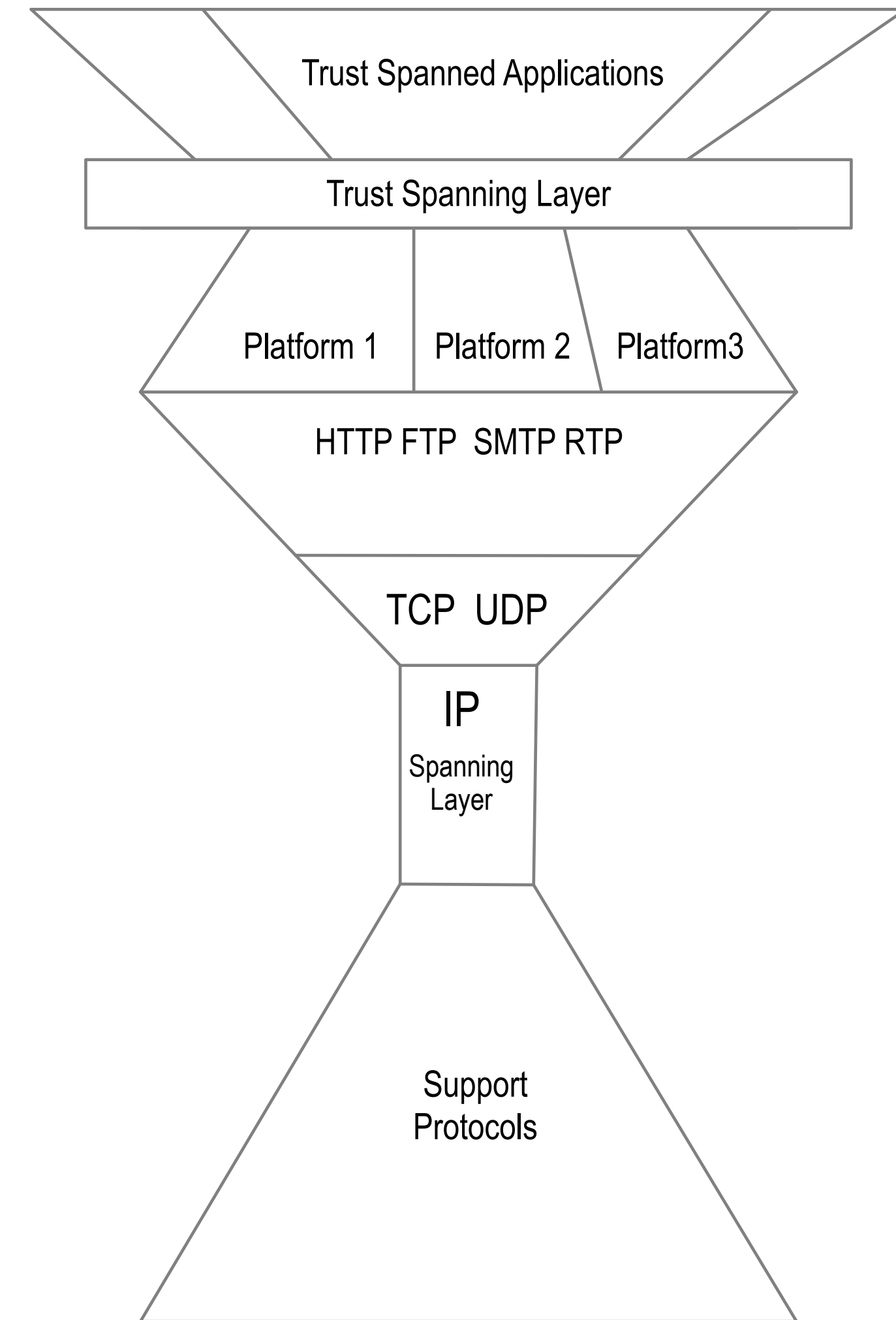
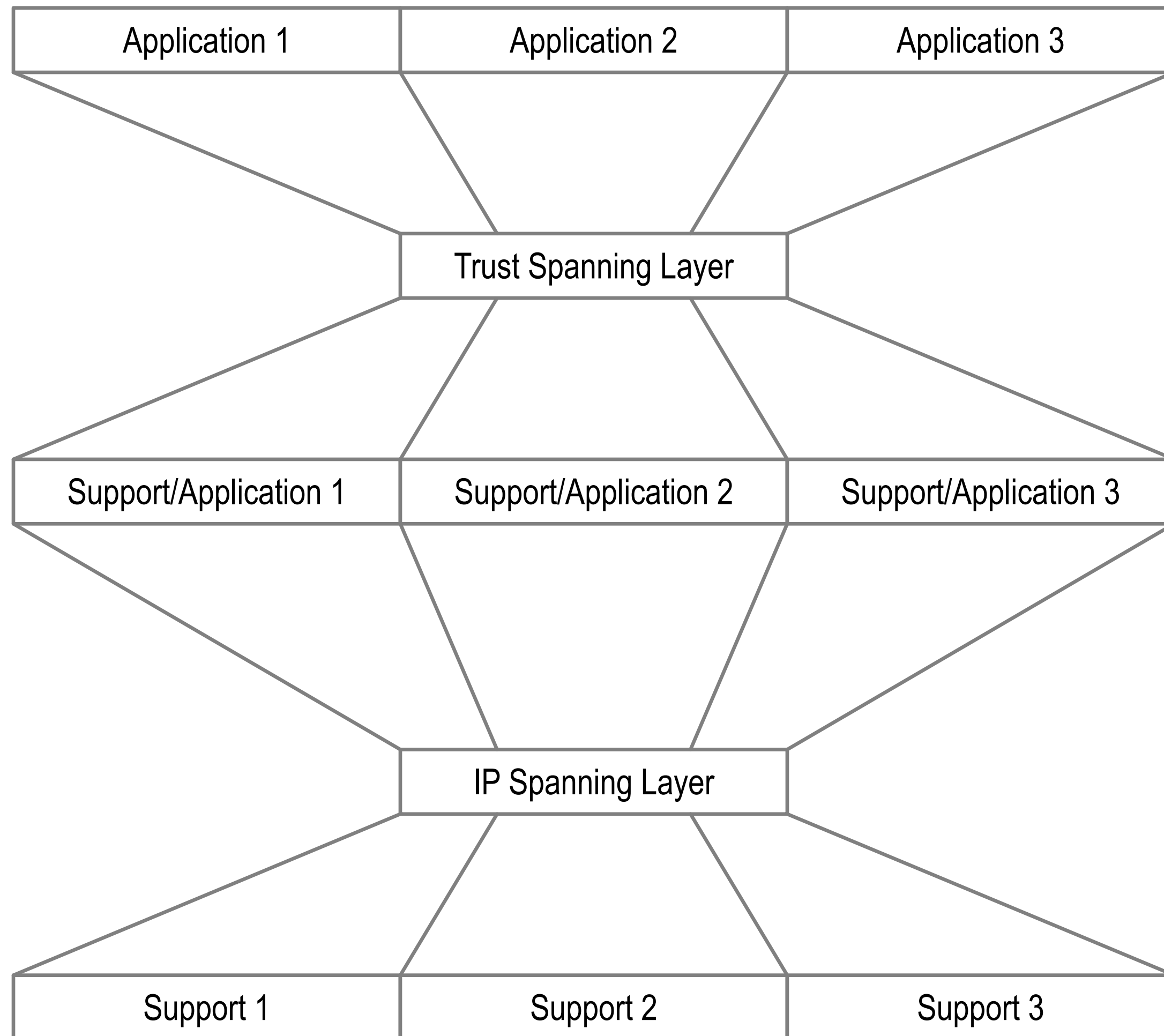
Trust Domain Based Segmentation



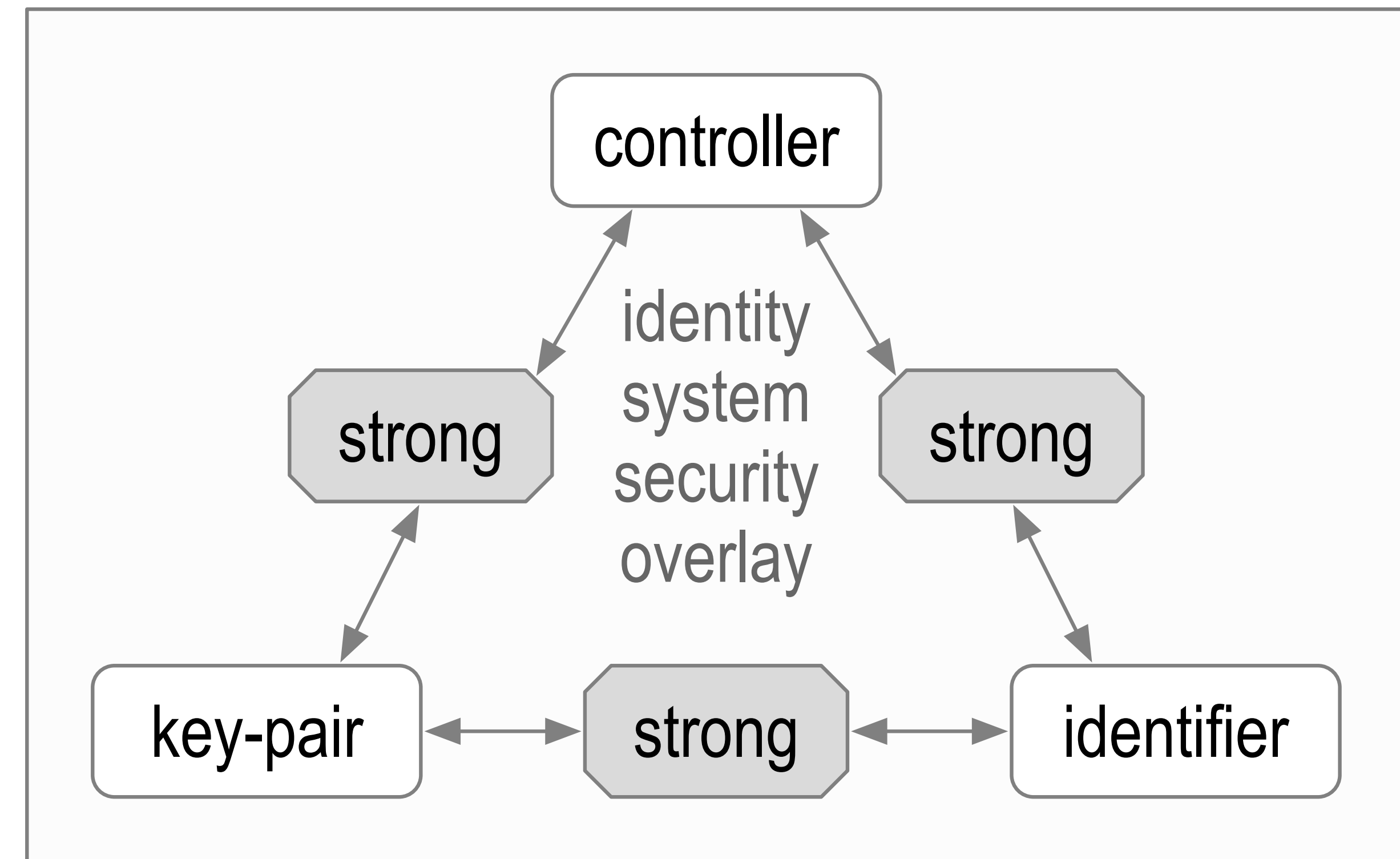
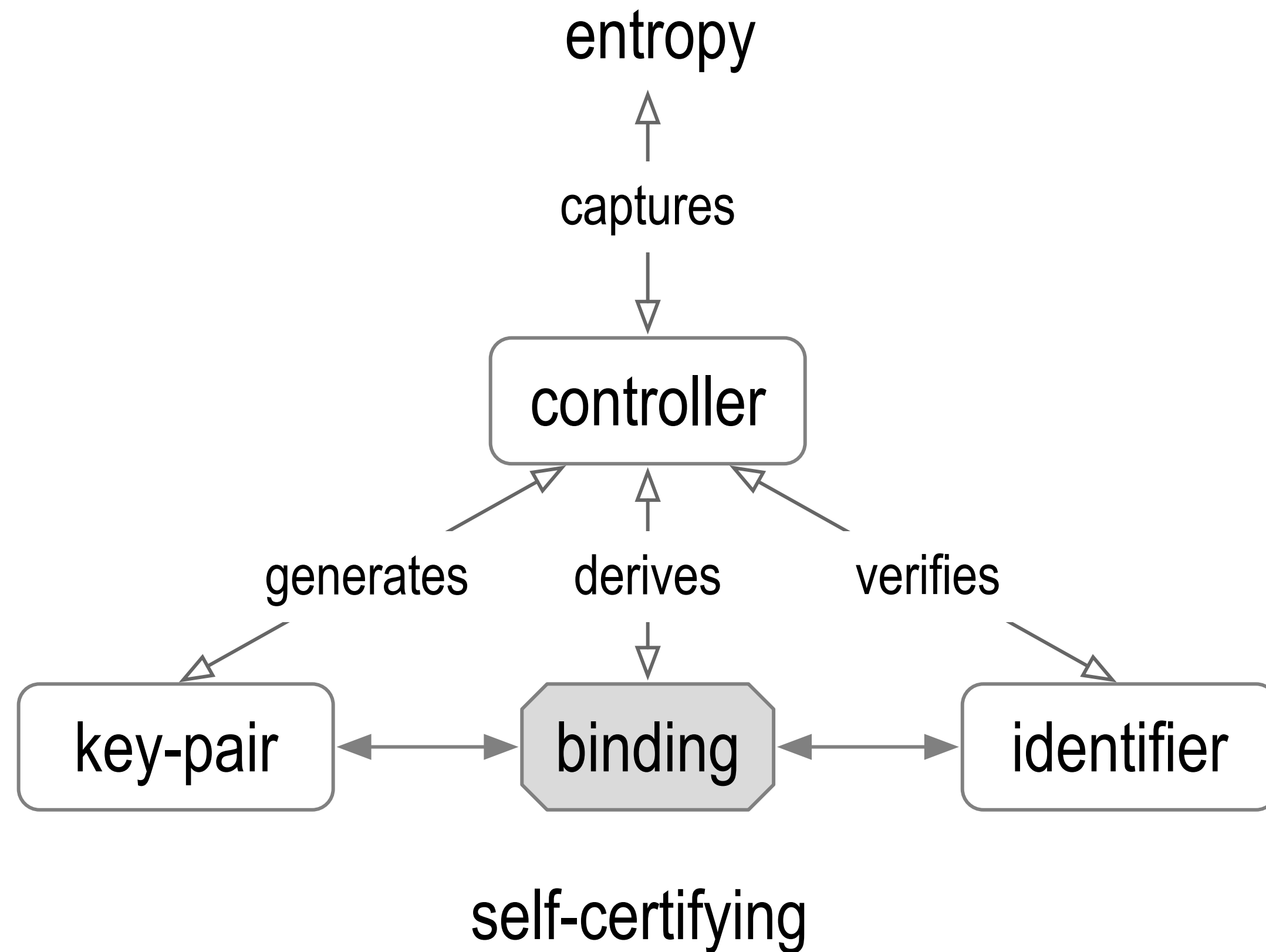
Each trust layer only spans platform specific applications
Bifurcates the internet trust map
No spanning trust layer



Waist and Neck

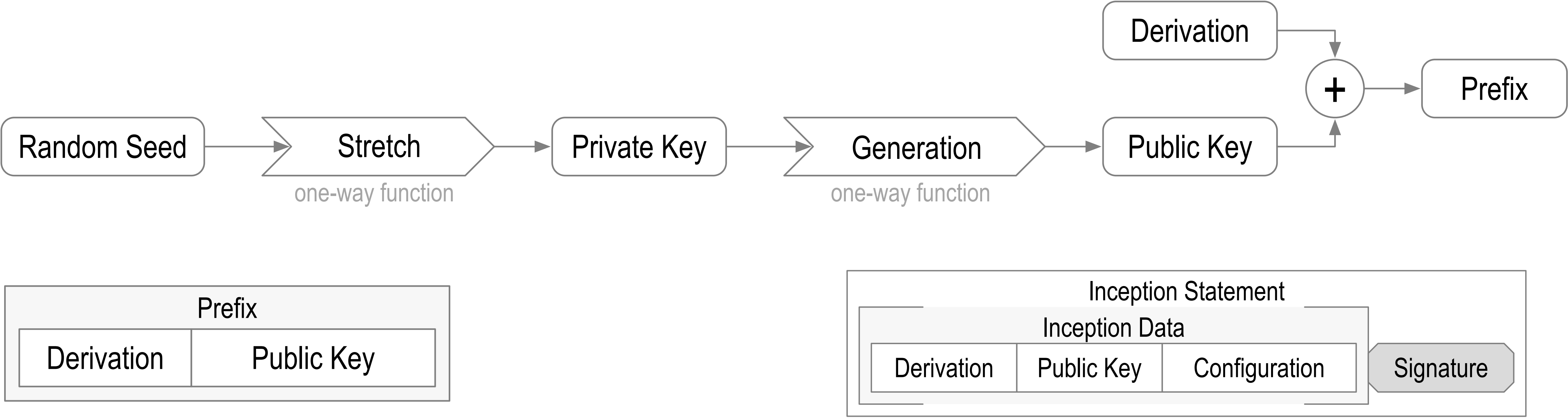


Self-Certifying Identifier Issuance and Binding



Self-Certifying Identifier Issuance

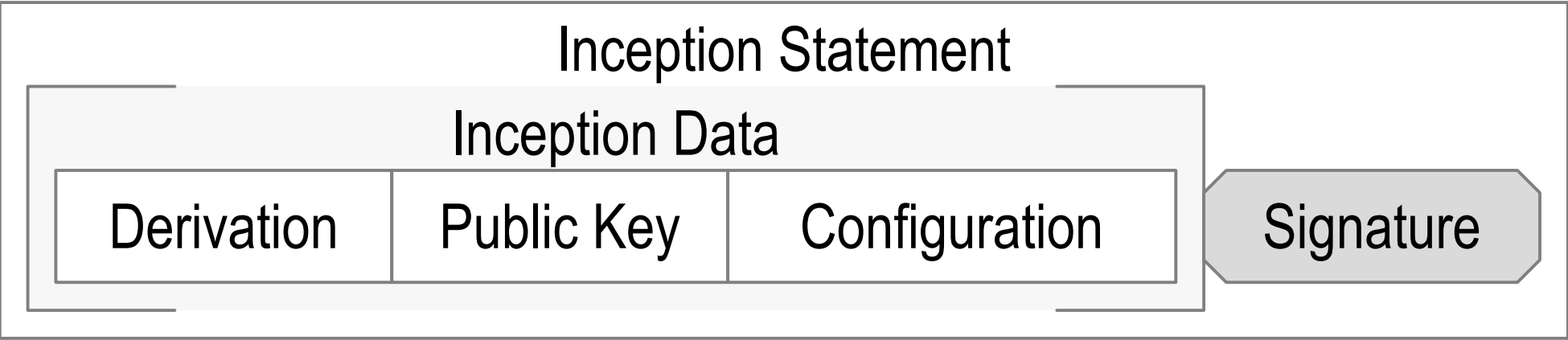
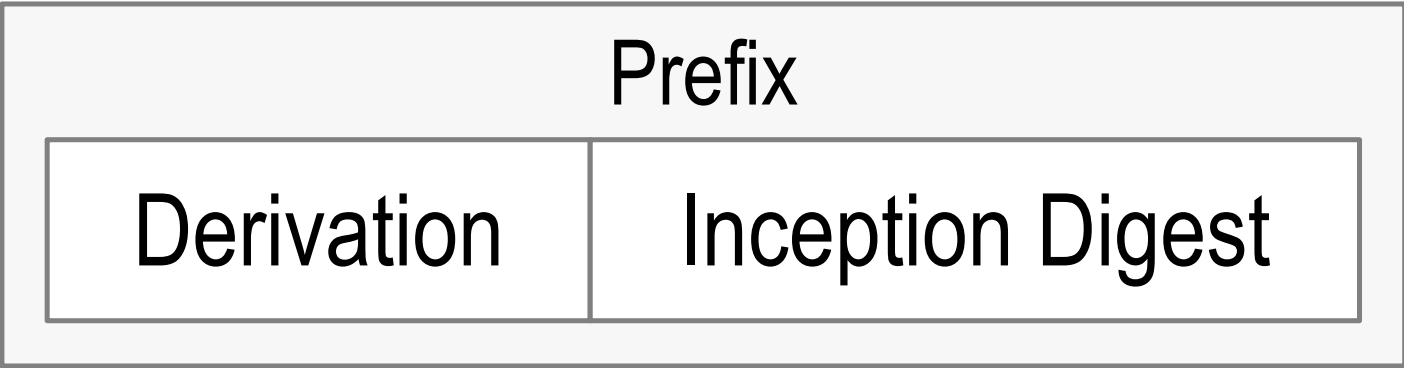
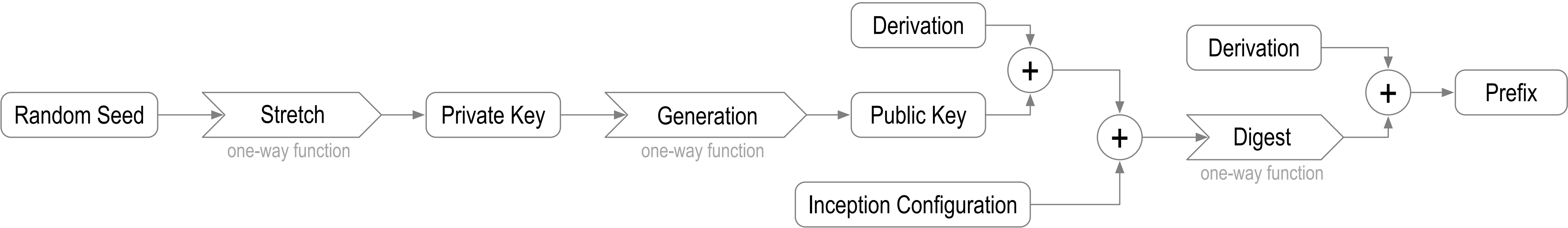
Basic SCID



BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUt1Ddhh0

did:un:BDKrJxkcR9m5u1xs33F5pxRJP6T7hJEbhpHrUt1Ddhh0/path/to/resource?name=secure#really

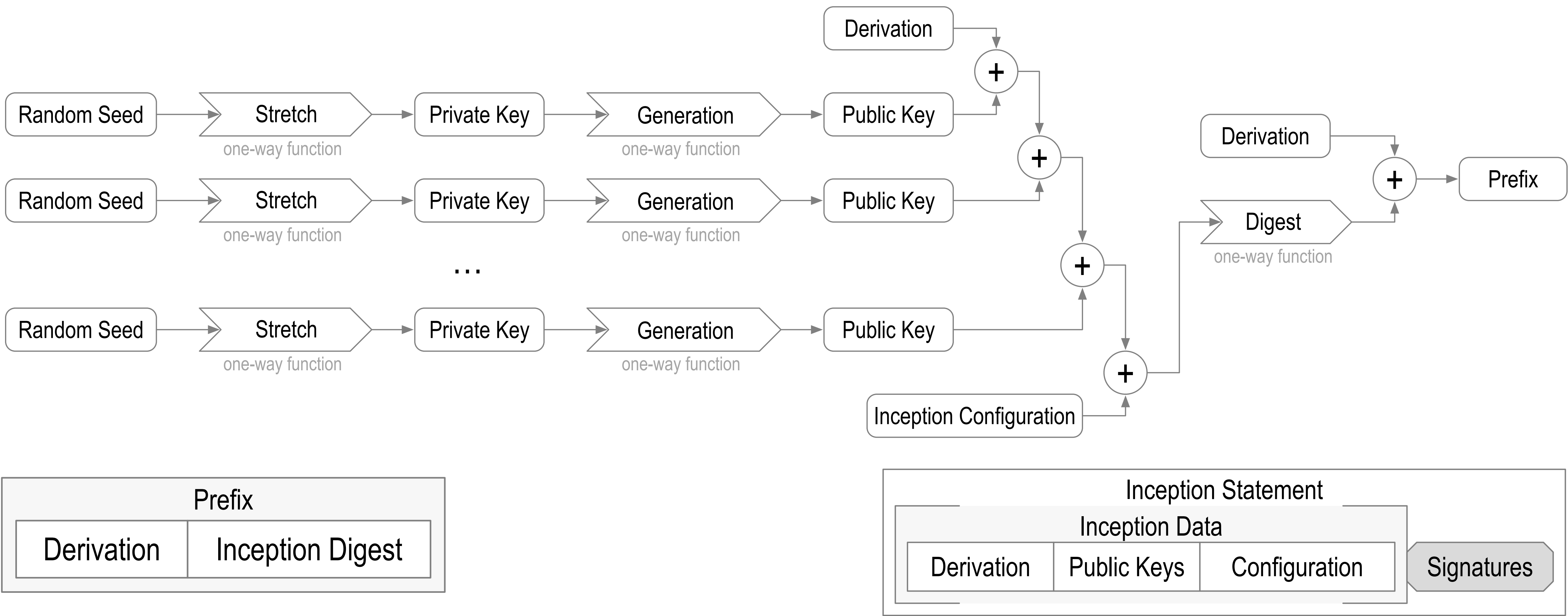
Self-Addressing SCID



EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

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

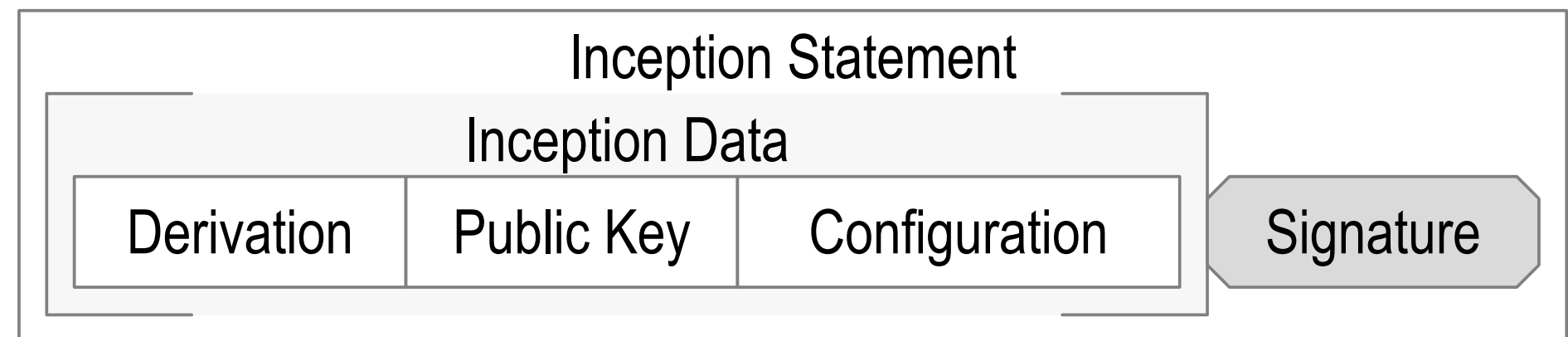
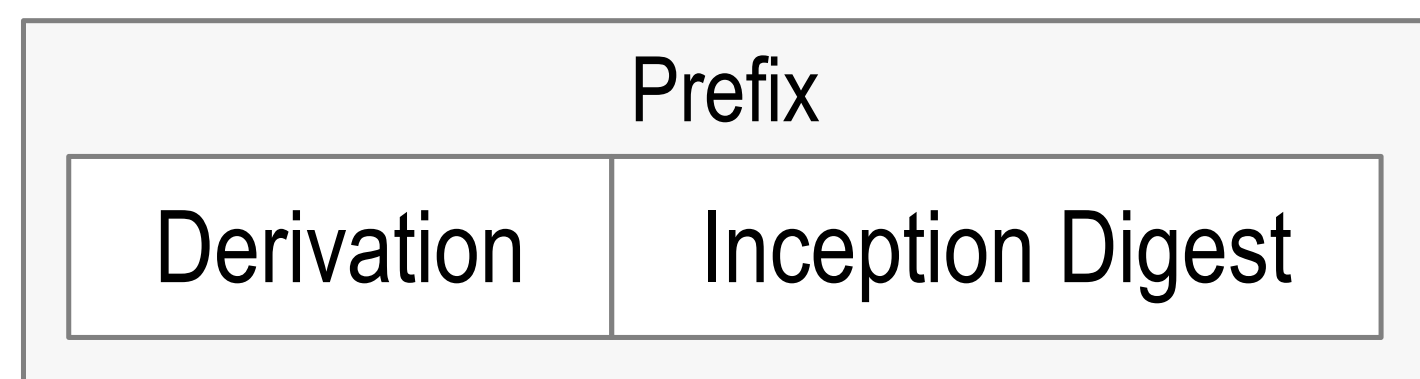
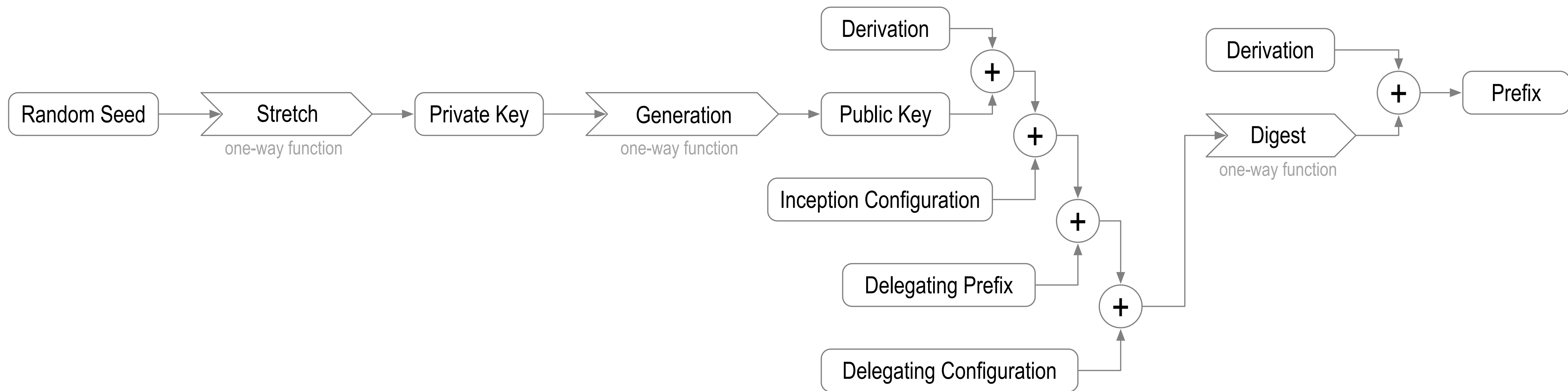
Multi-Sig Self-Addressing SCID



EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

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

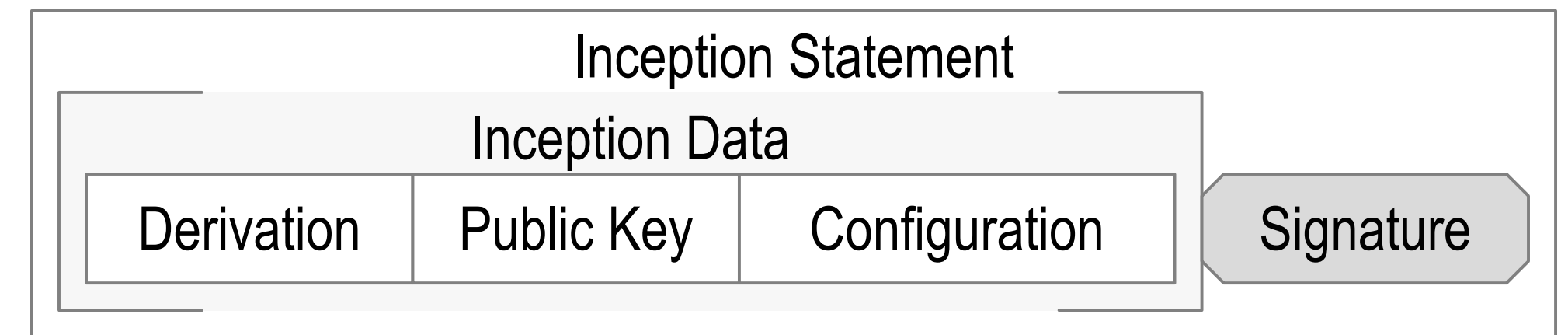
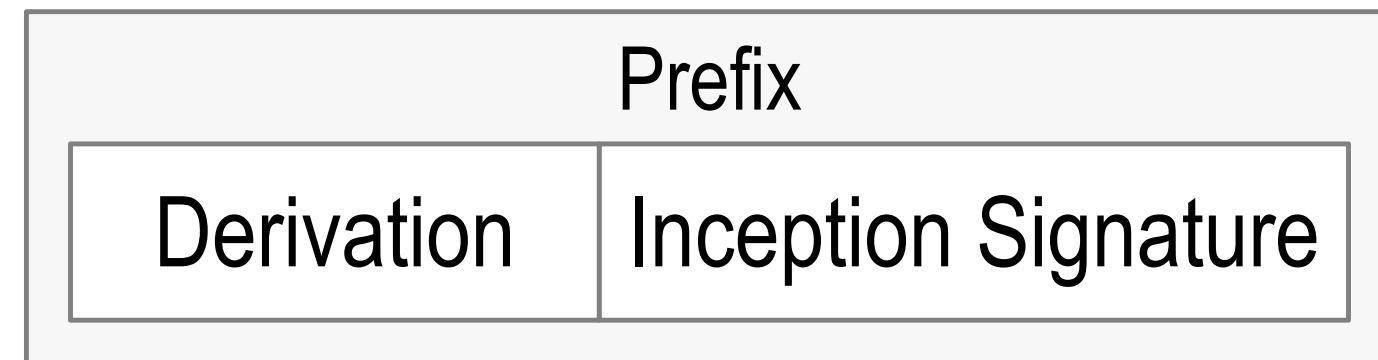
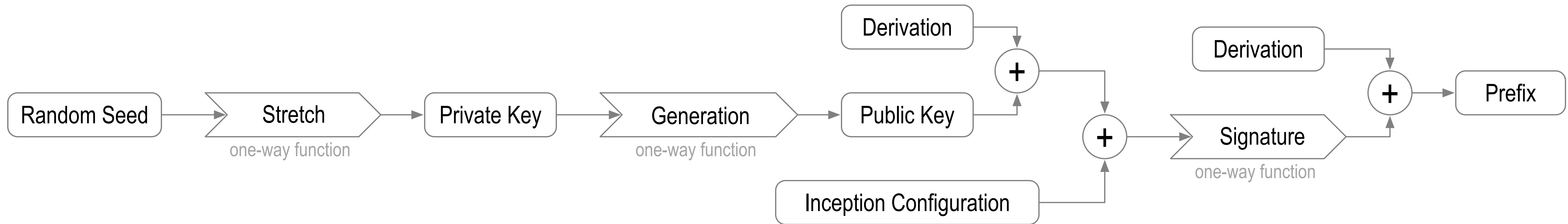
Delegated Self-Addressing SCID



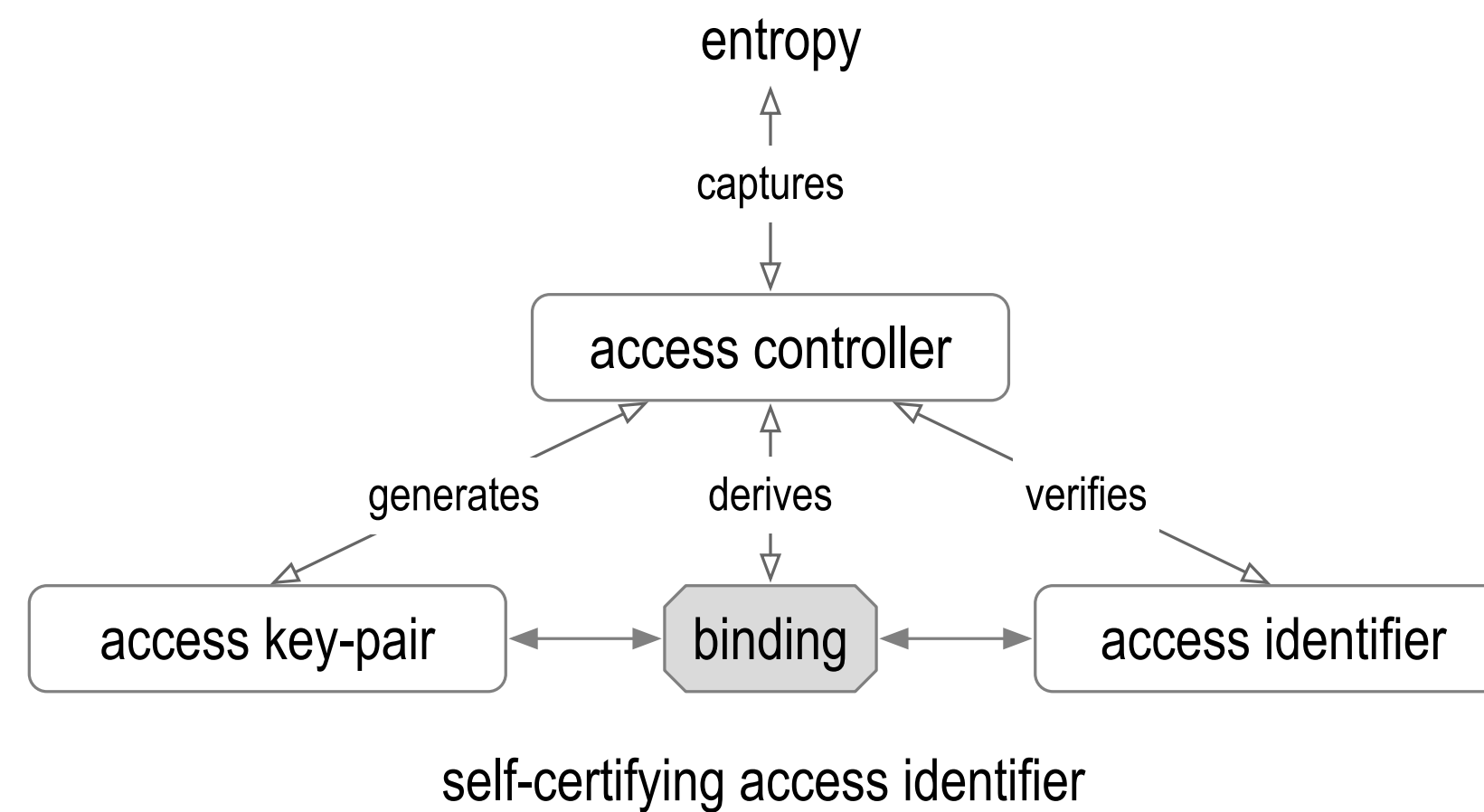
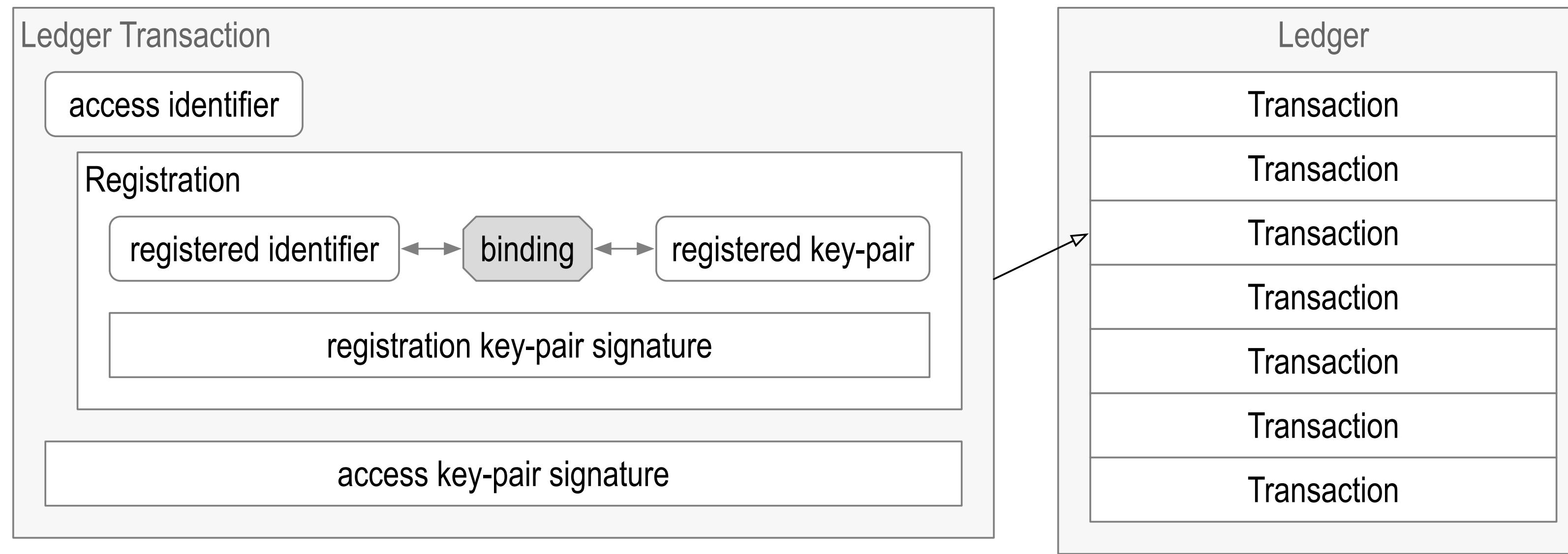
EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

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

Self-Signing SCID



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

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.

Unified Identifier Model

AID: Autonomic Identifier (primary)

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

secure, decentralized, portable, universally unique

HID: Authorized Human Meaningful Identifier (secondary) from `aid|hid` couplet

hid = authorized human meaningful identifier

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

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



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 *anylog*, *anywhere*, at *anytime*.

KERI = self-cert root-of-trust + certificate transparency + KA²CE + 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:un (did:uni, did:u) (all of the above in one method)

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

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).

Autonomic Identity System

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

Root-of-Trust

cryptographic autonomic identifier = *why, how*

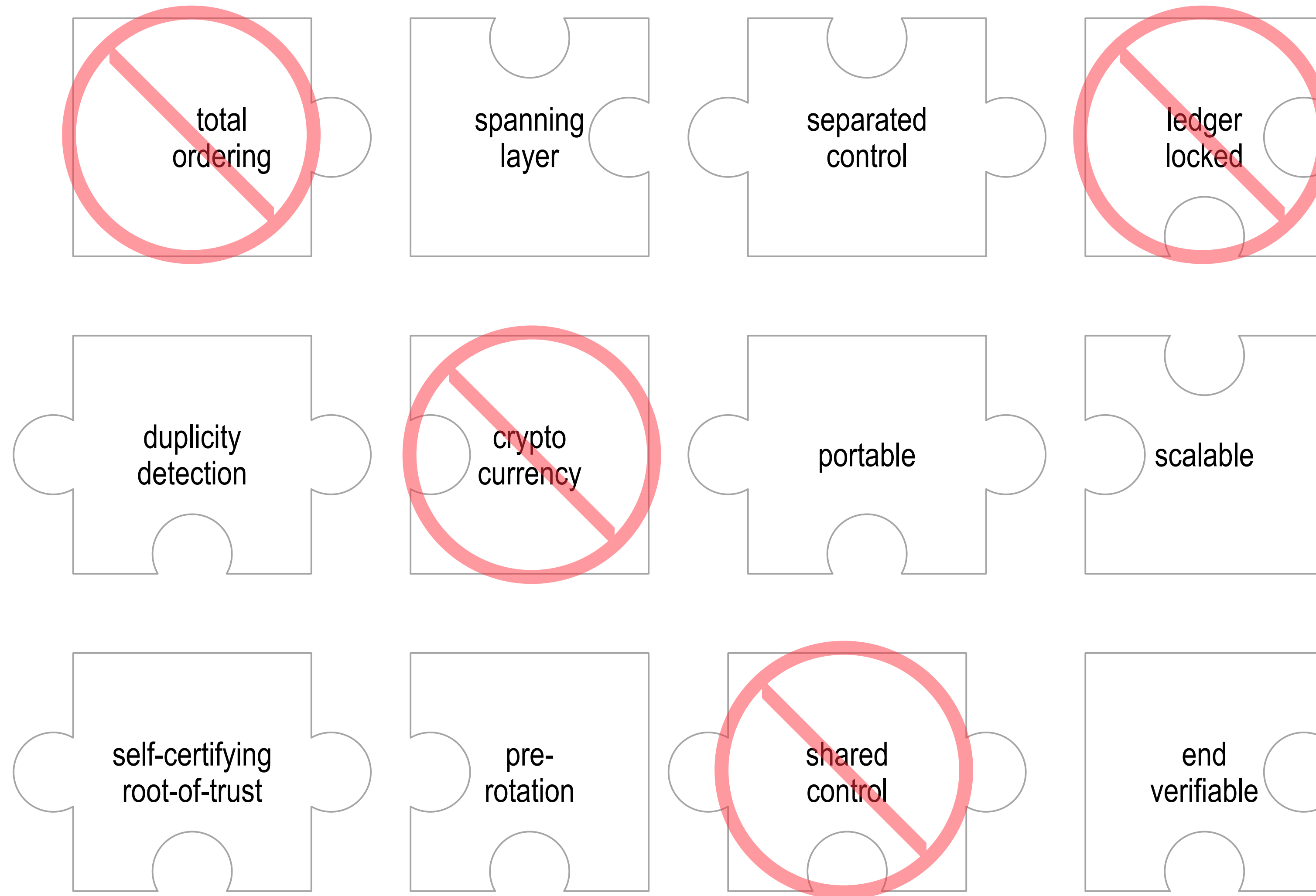
Source-of-Truth

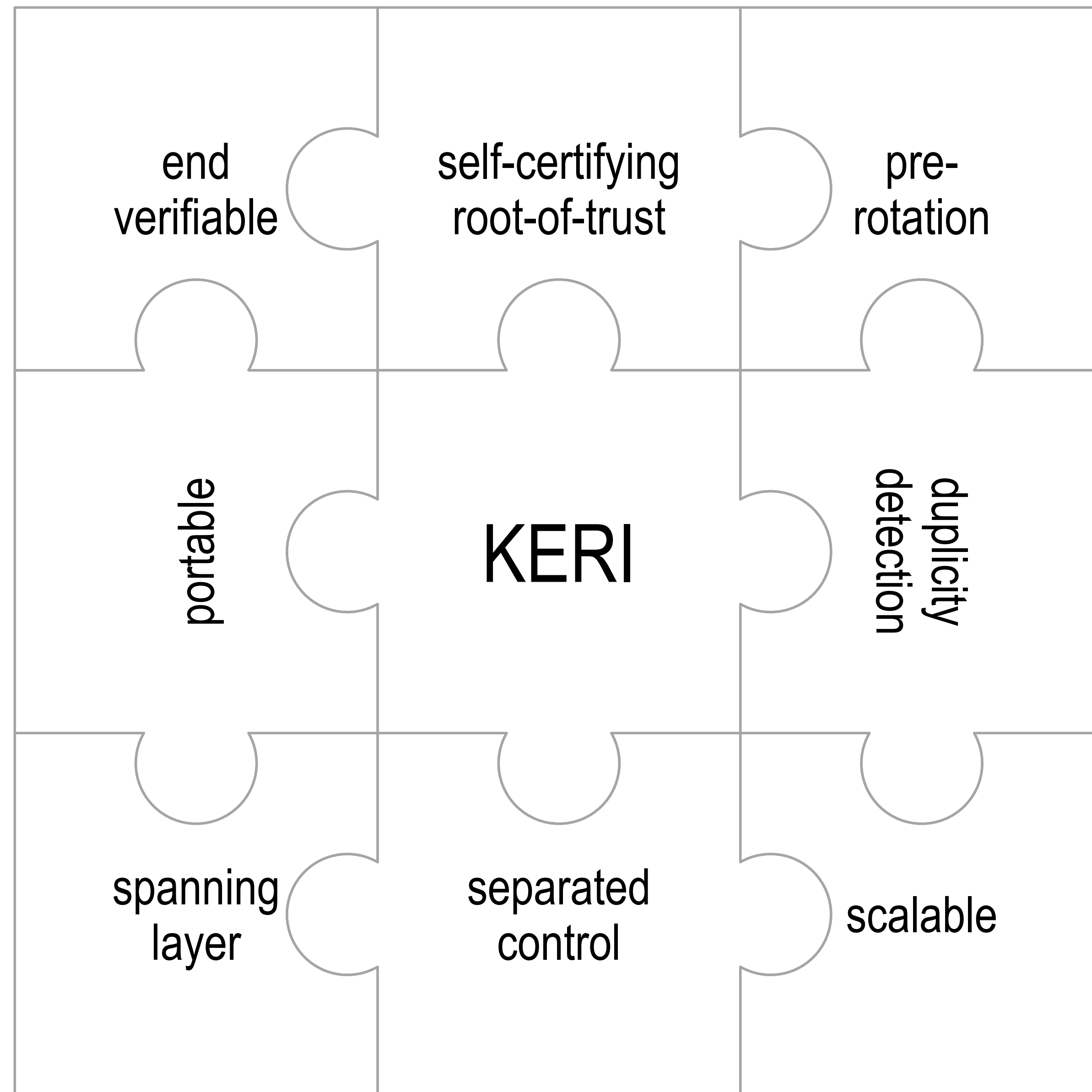
controller of the private key = *who*

Loci-of-Control

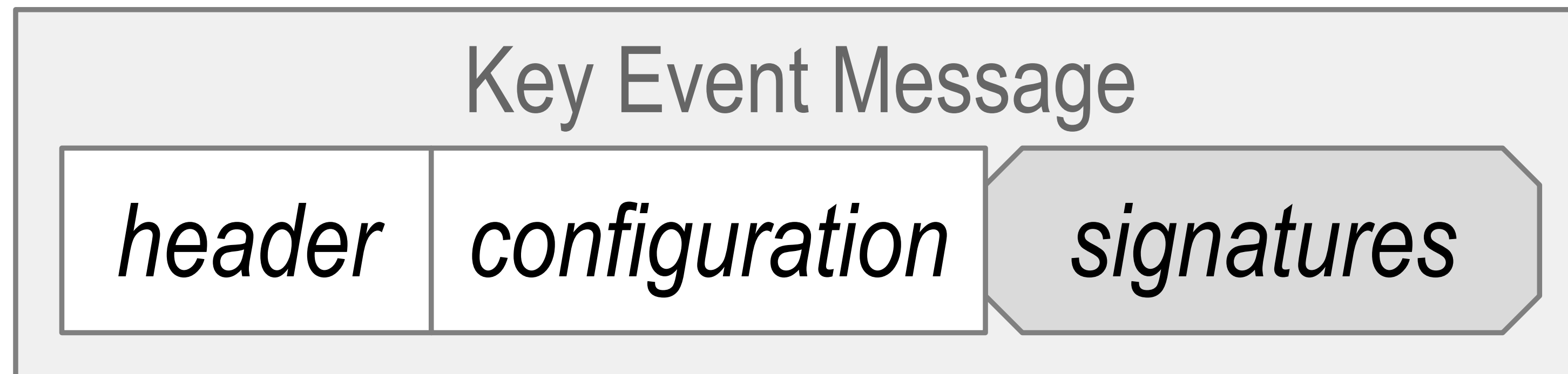
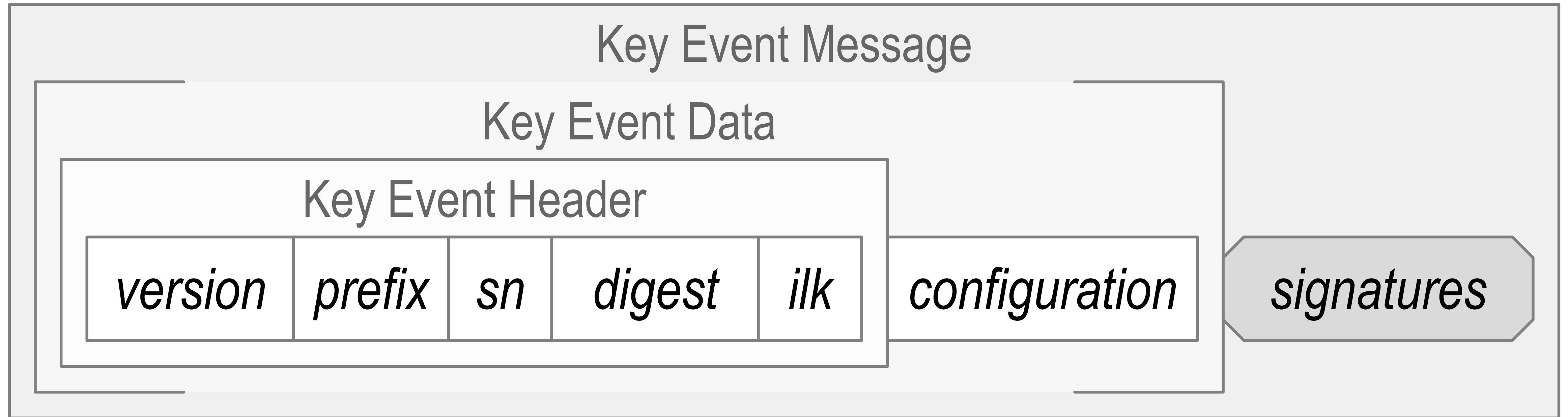
authoritative operation = *what, when, how*

System Design Trade Space

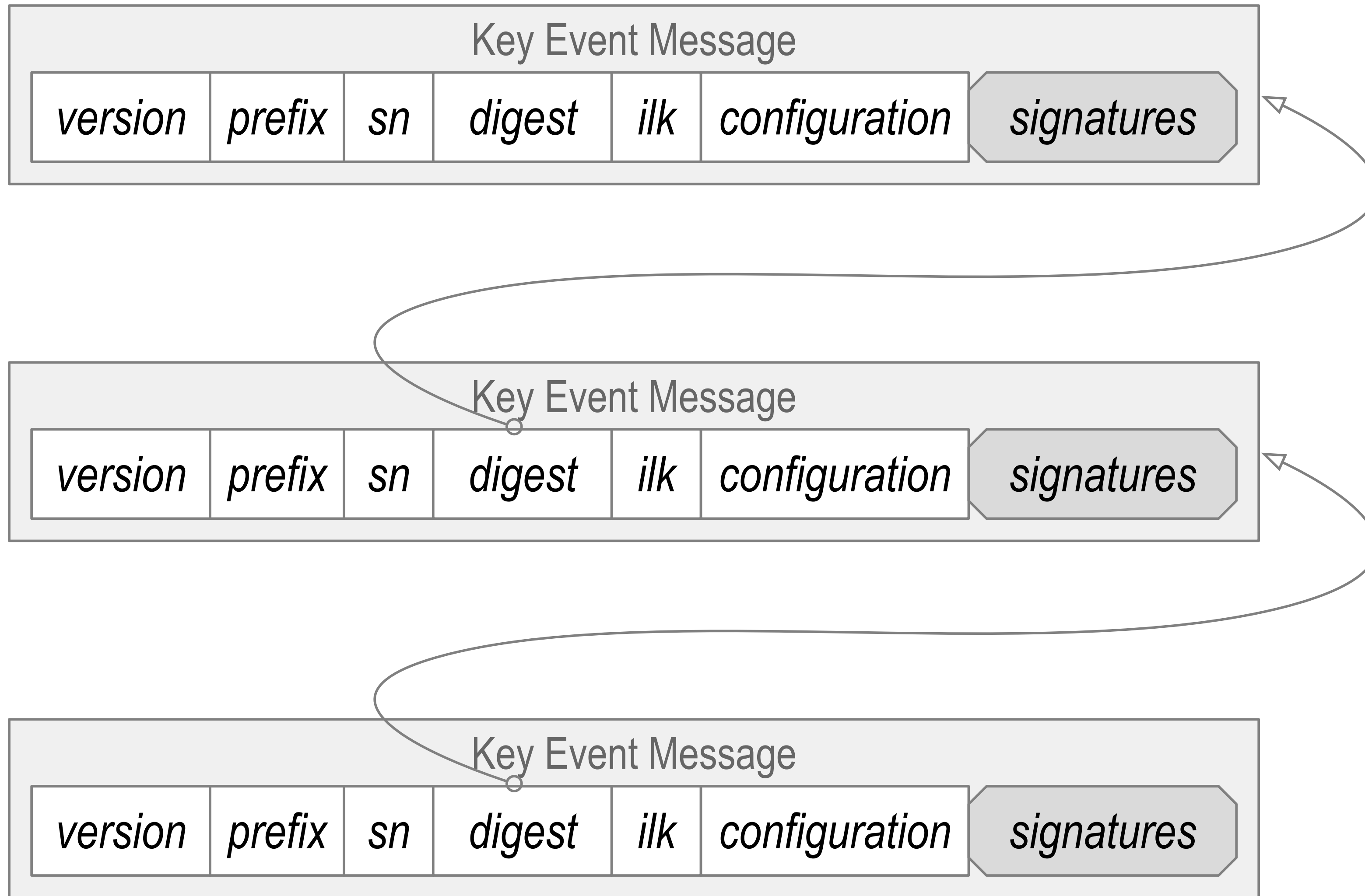




Key Event Message



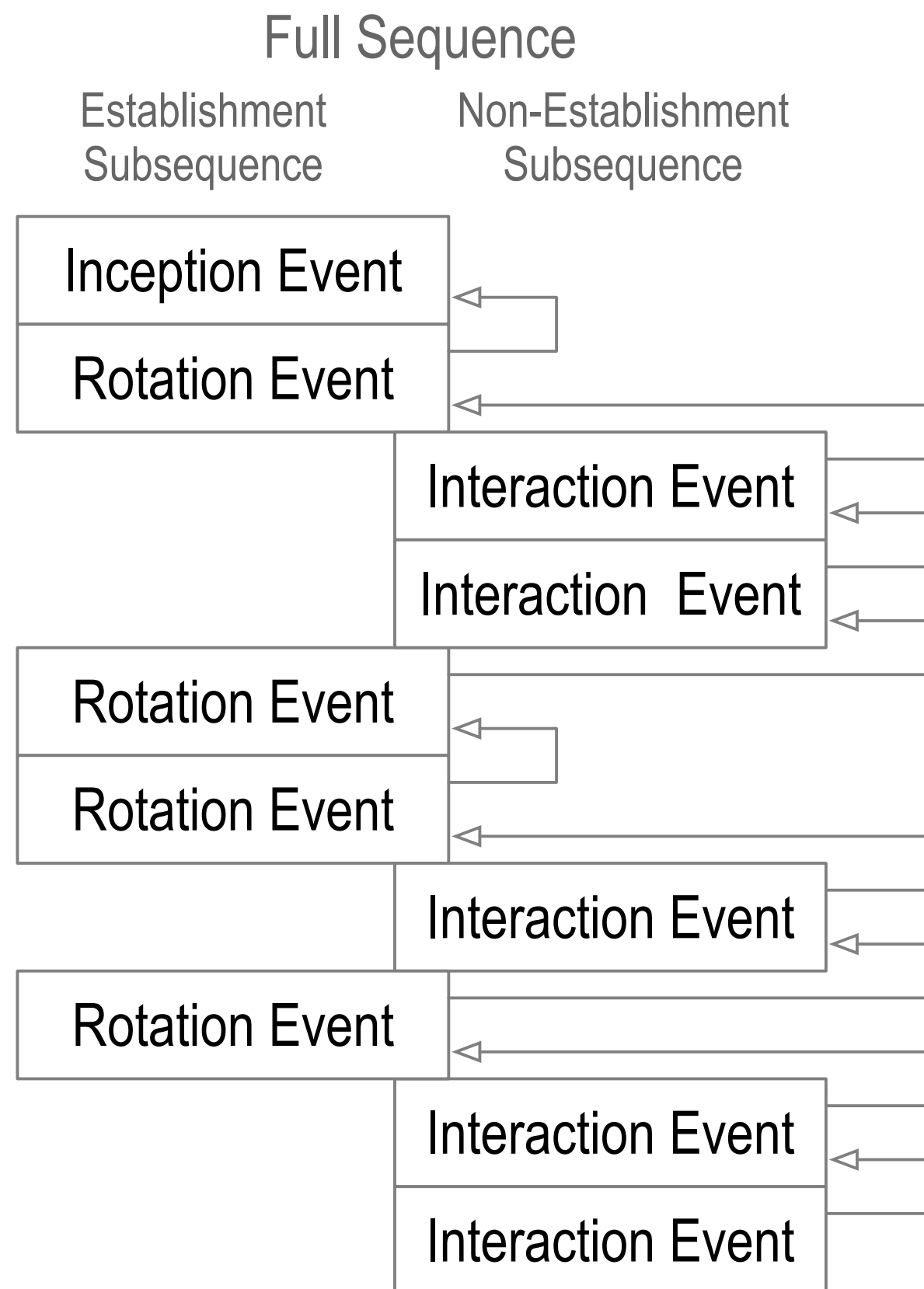
Event Chaining



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**.

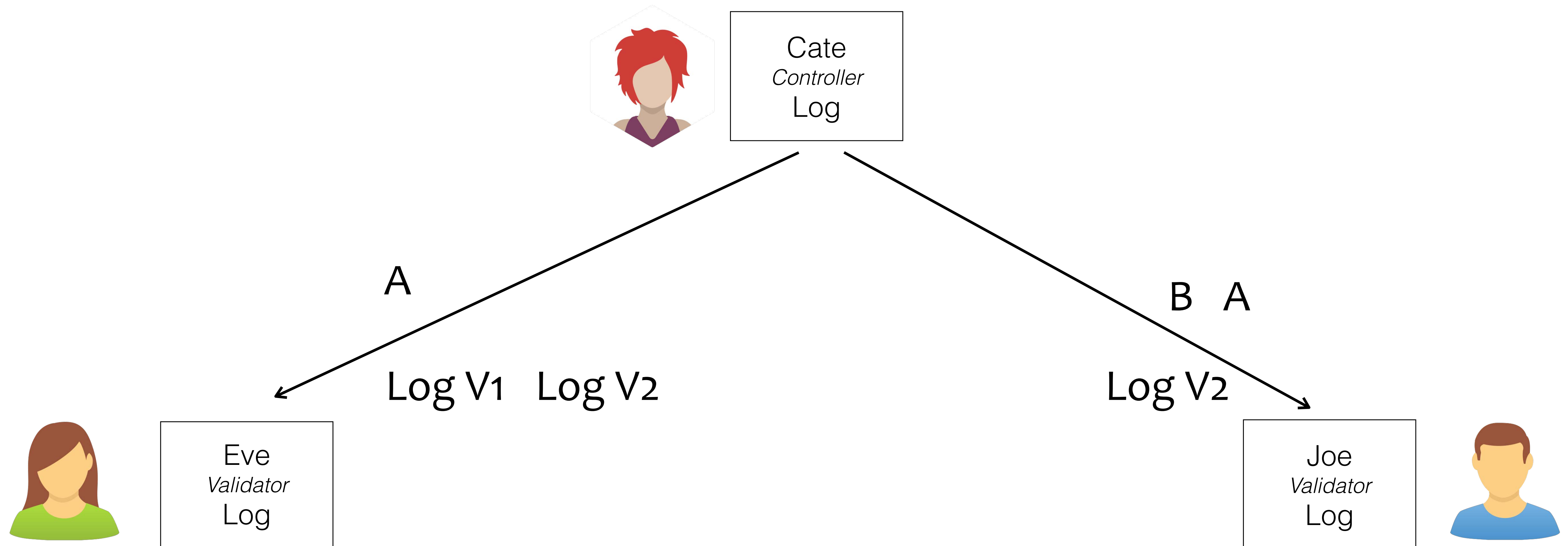
Duplicity detection protects against **external inconsistency**.

Duplicity Game

Cate promises to provide a
consistent pair-wise log.

Local Consistency Guarantee

How may Cate be *duplicitous*
and not get caught?



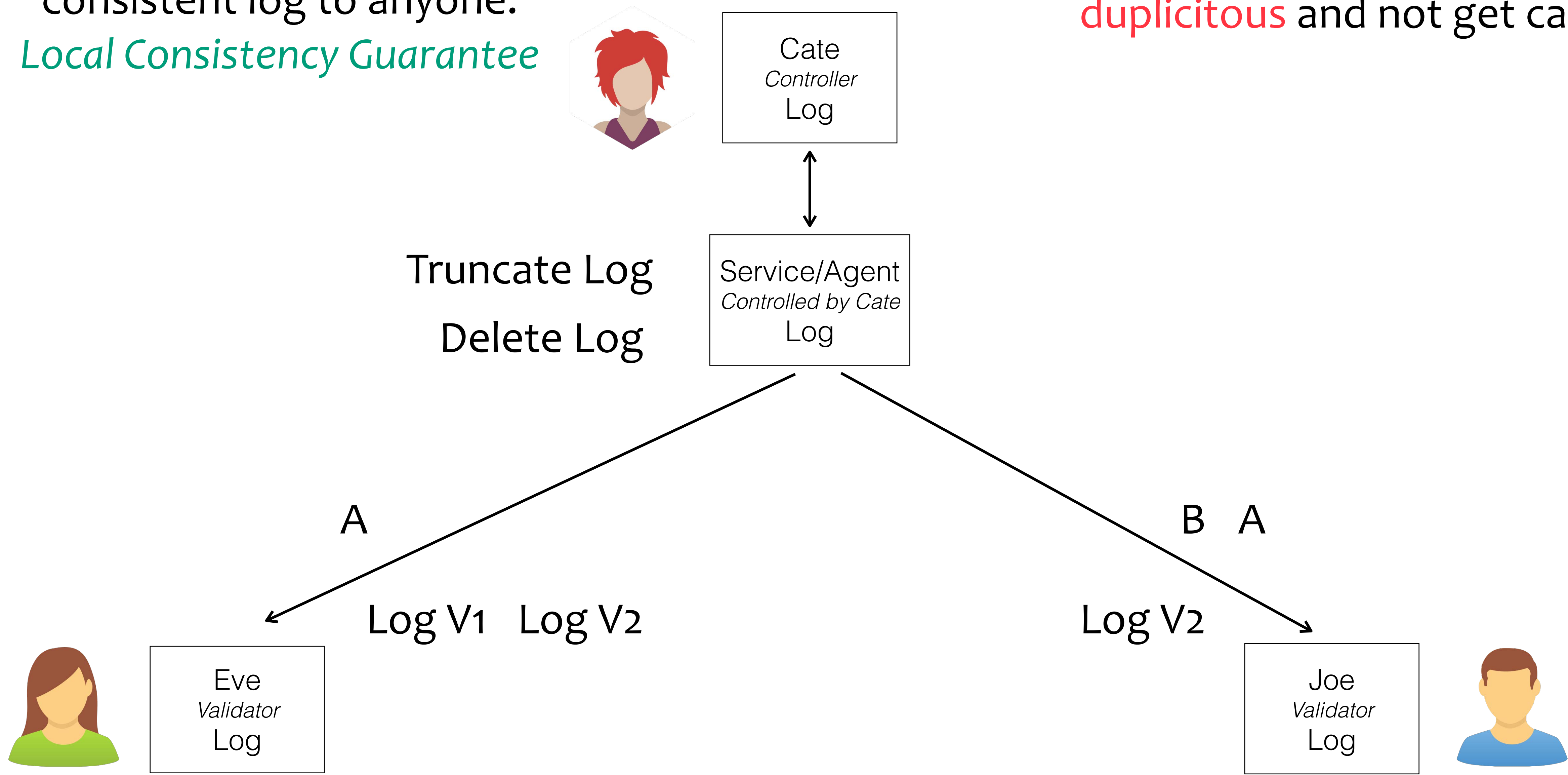
private (one-to-one) interactions

Service promises to provide a consistent log to anyone.

Local Consistency Guarantee

Duplicity Game

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



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

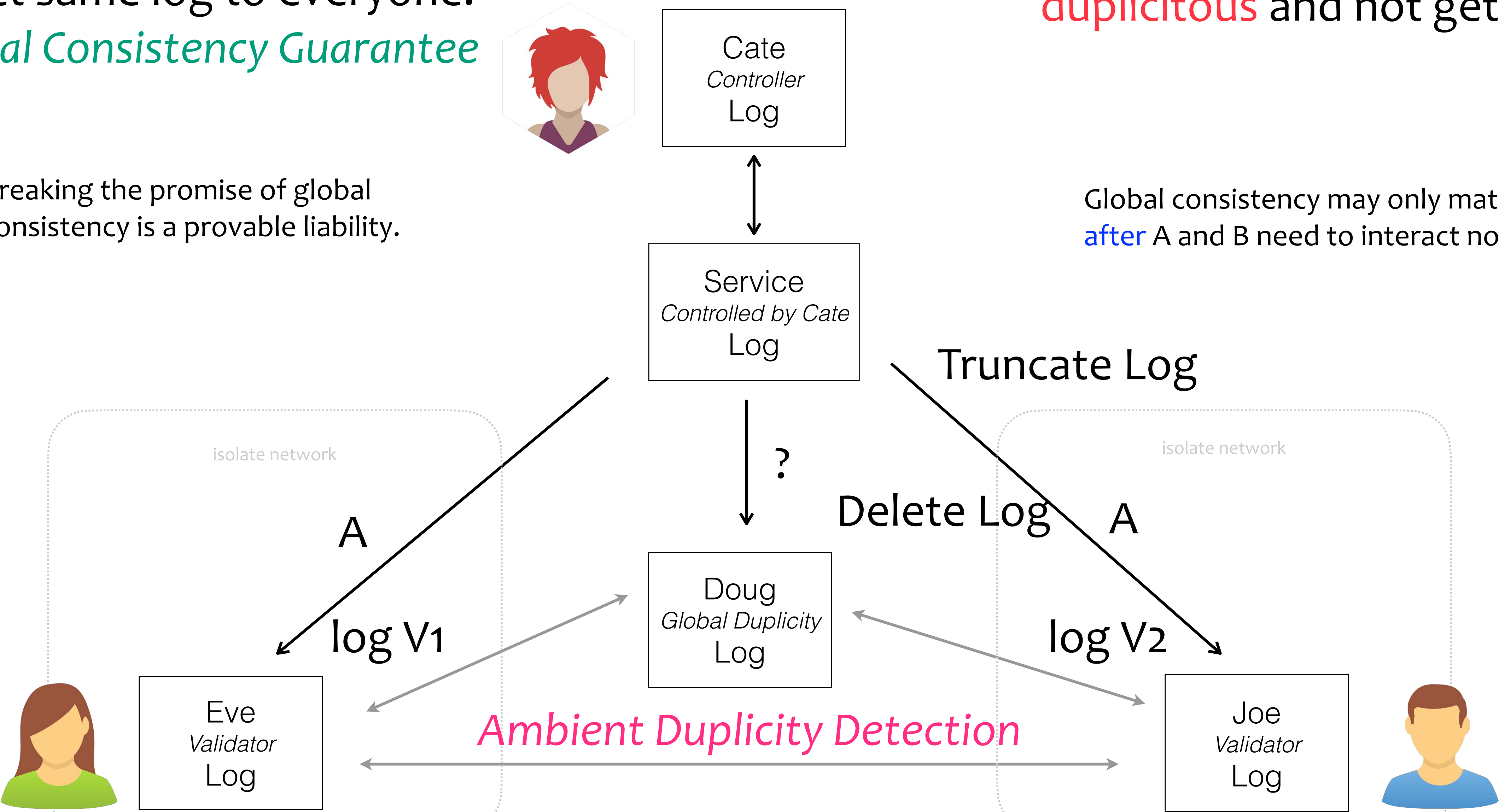
Service promises to provide exact same log to everyone.
Global Consistency Guarantee

Duplicity Game

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

Breaking the promise of global consistency is a provable liability.

Global consistency may only matter **after** A and B need to interact not before.



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

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 *anylog*, *anywhere*, at *anytime*.

KERI = self-cert root-of-trust + certificate transparency + KA²CE + 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:un (did:uni, did:u) (all of the above in one method)

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

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

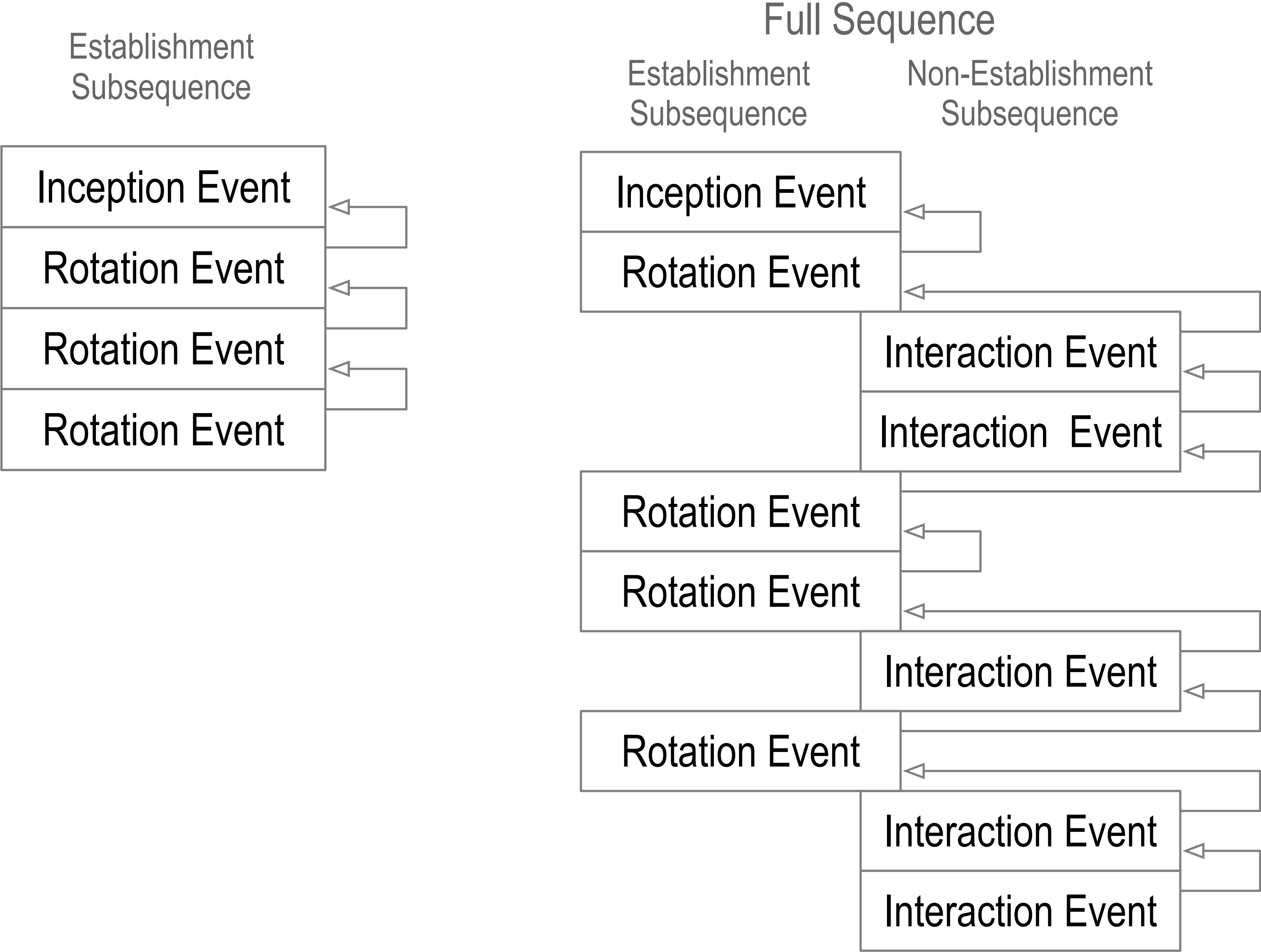
All we need for bootstrapping *interop* is some indication that the *prefix* inside identifier is KERI based (KERI trust basis).

Self-Certifying Identifier Prefixes

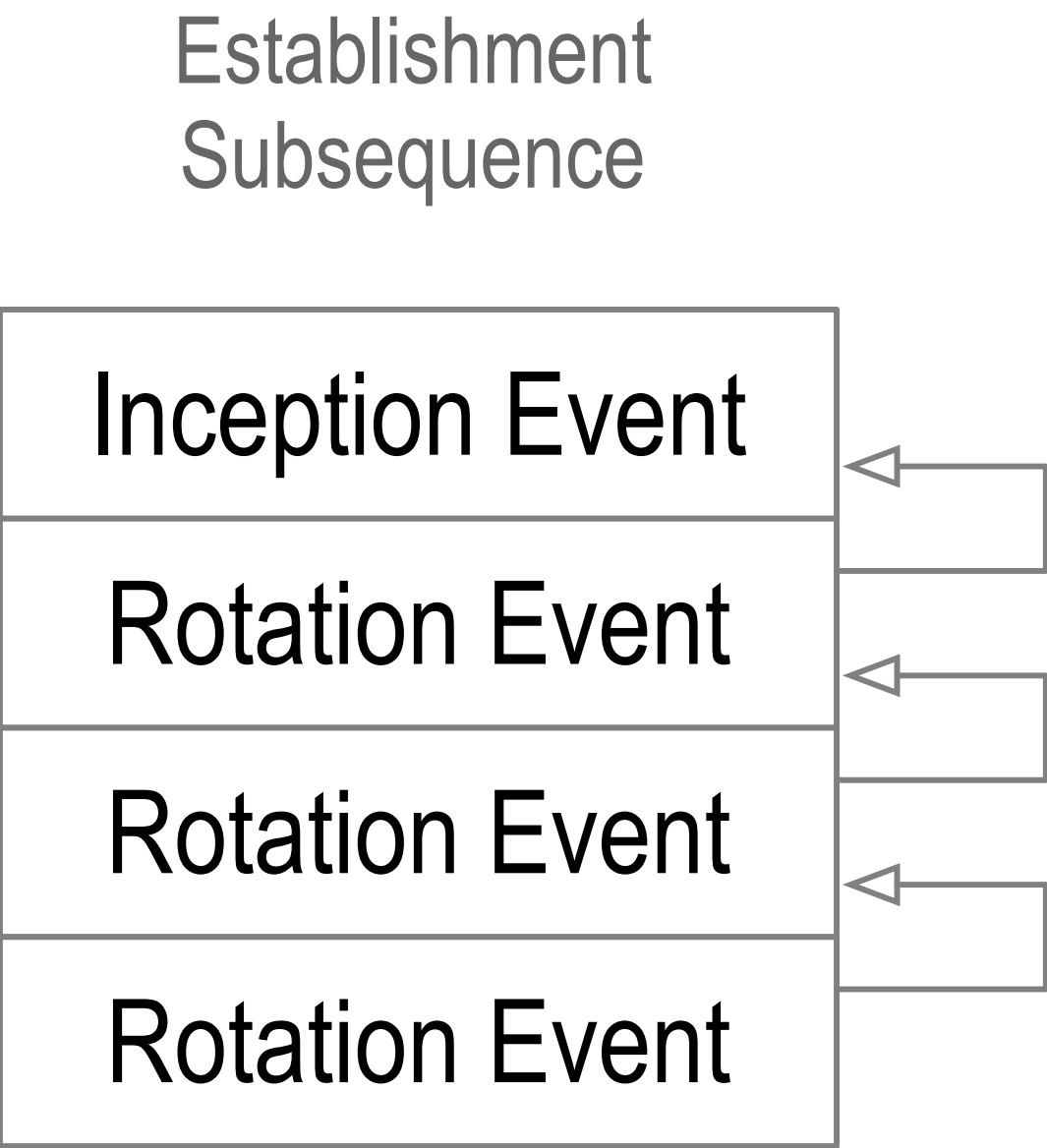
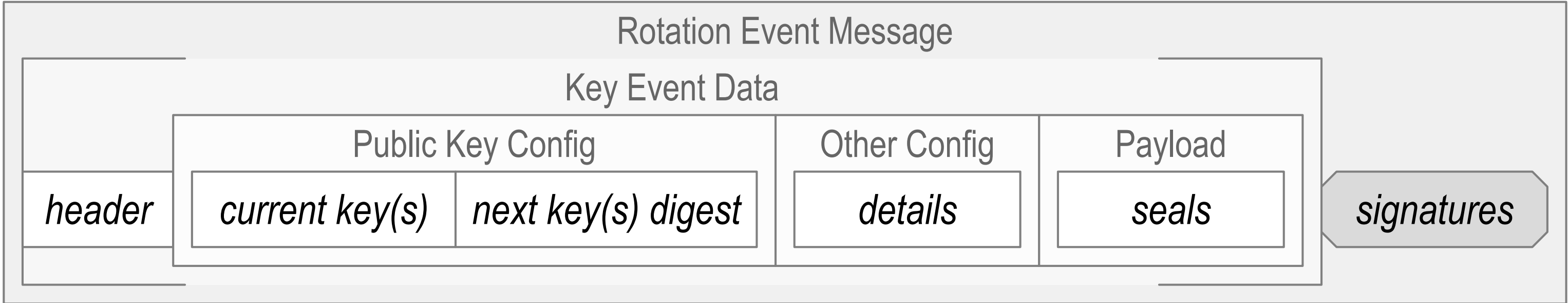
All crypto material appears in KERI in a fully qualified representation that includes a derivation code prepended to the crypto-material.

Identifier prefixes are fully qualified crypto-material.

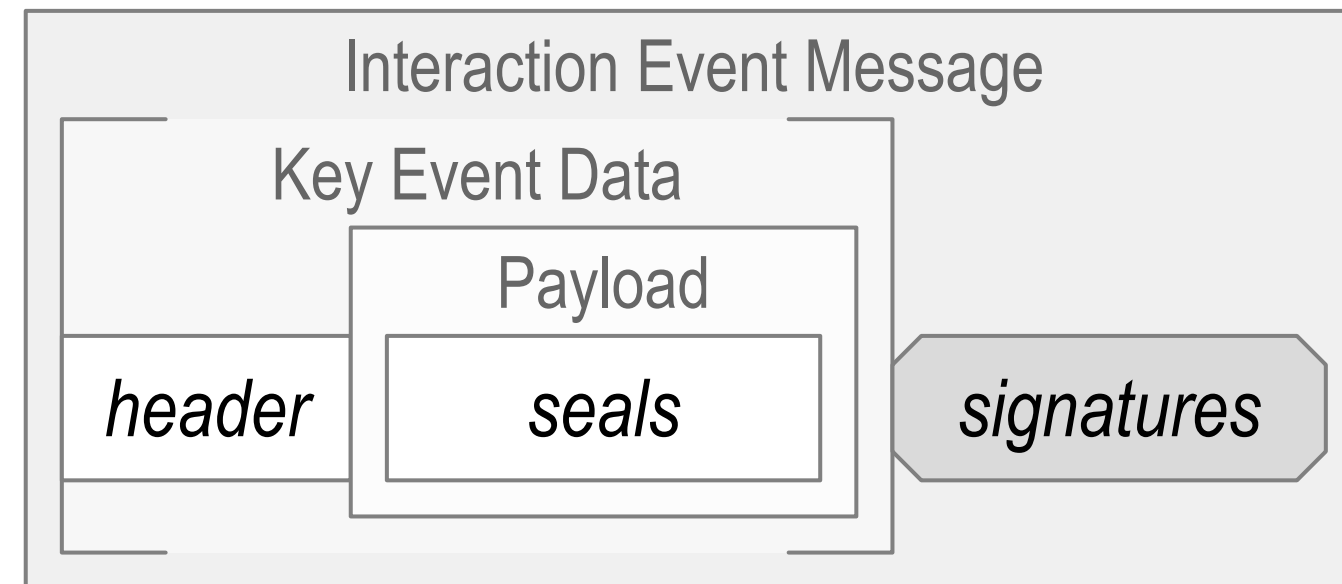
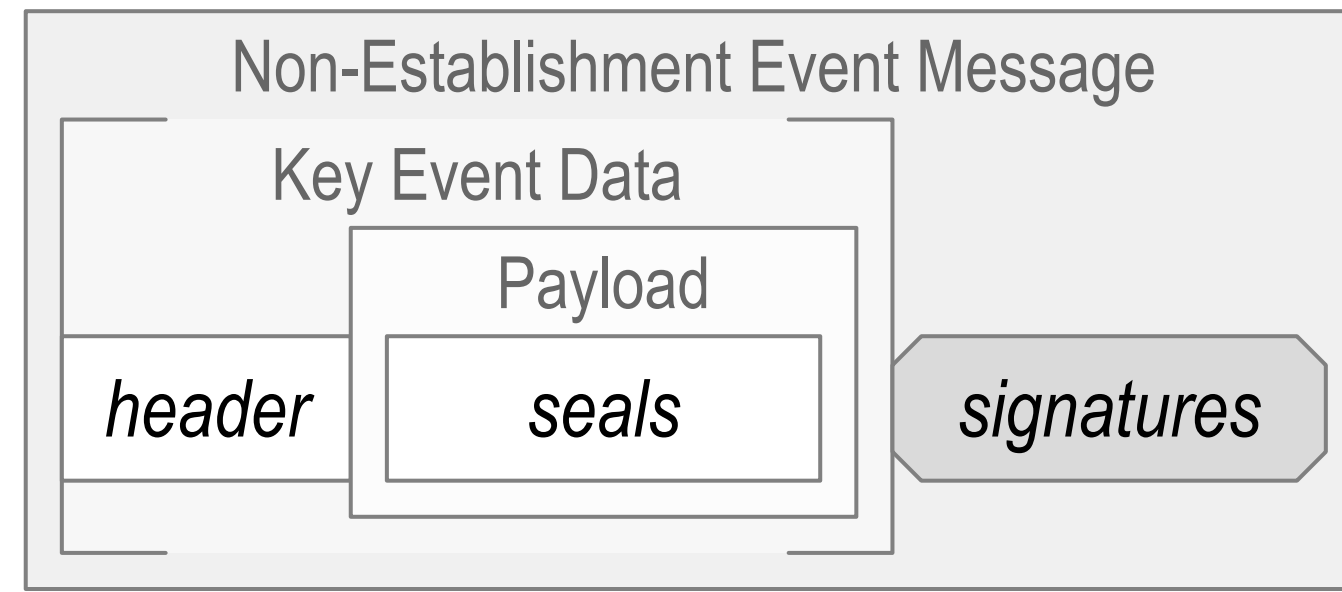
Event Sequencing



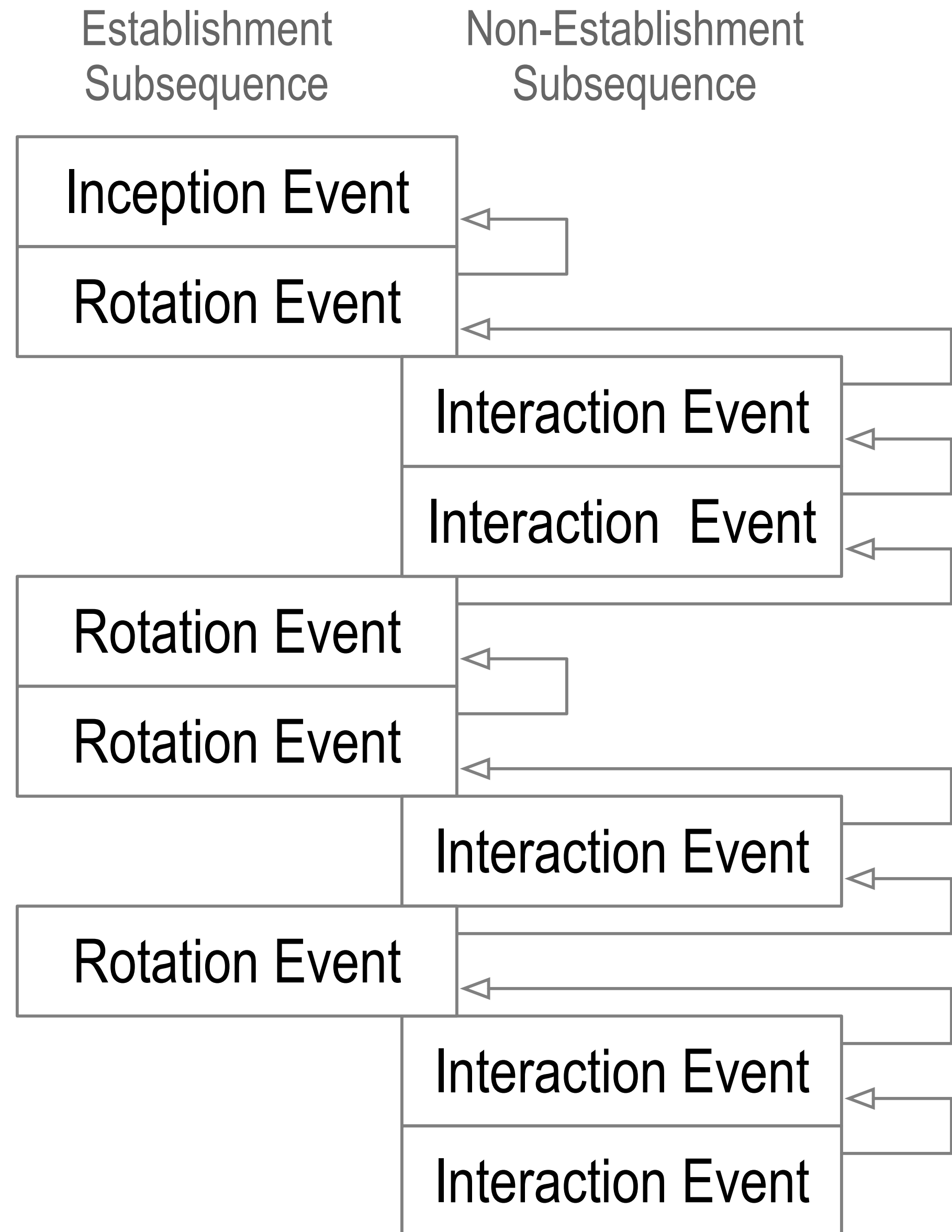
Establishment Events



Non-Establishment Events

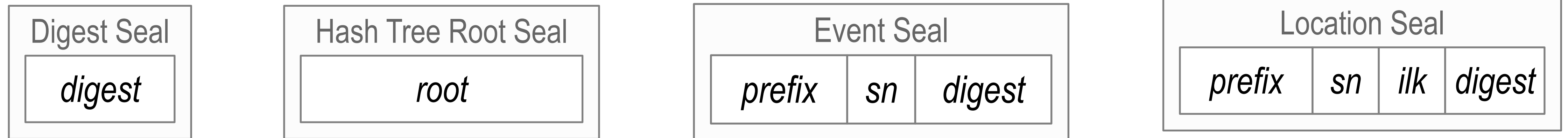


Full Sequence



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.

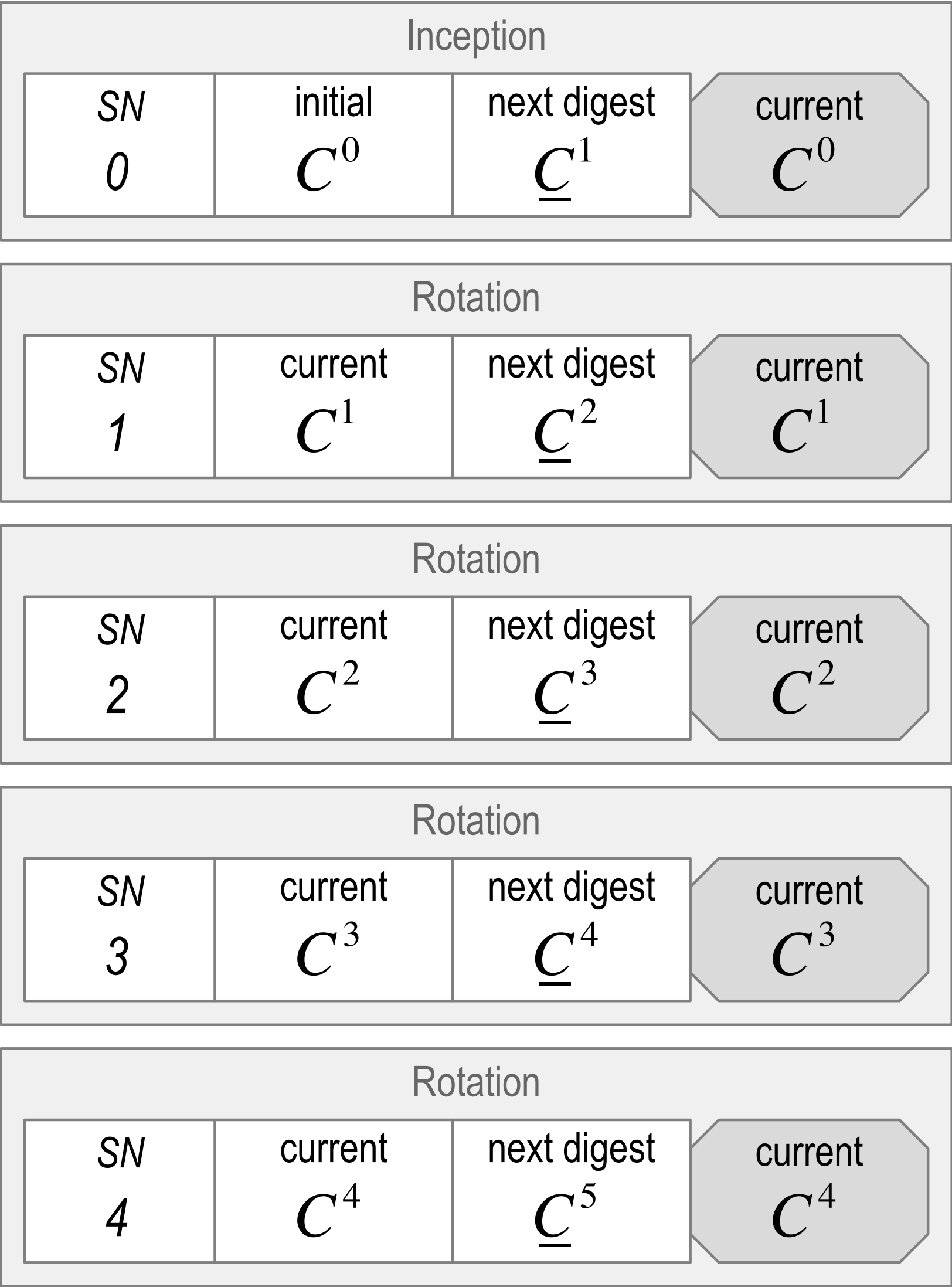
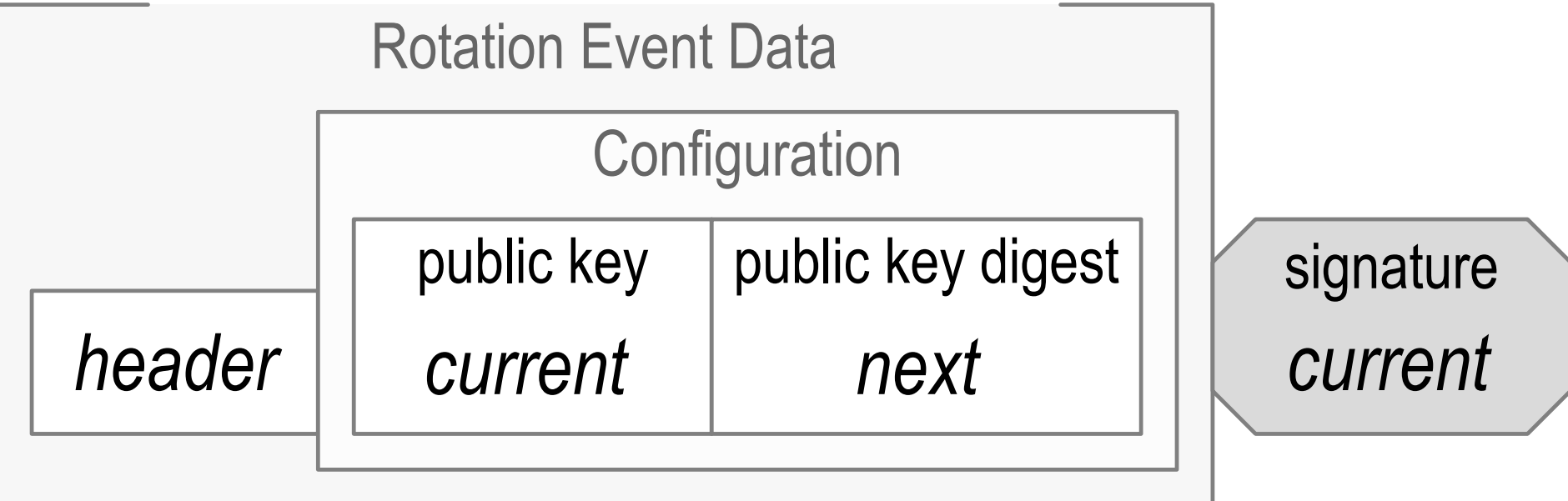
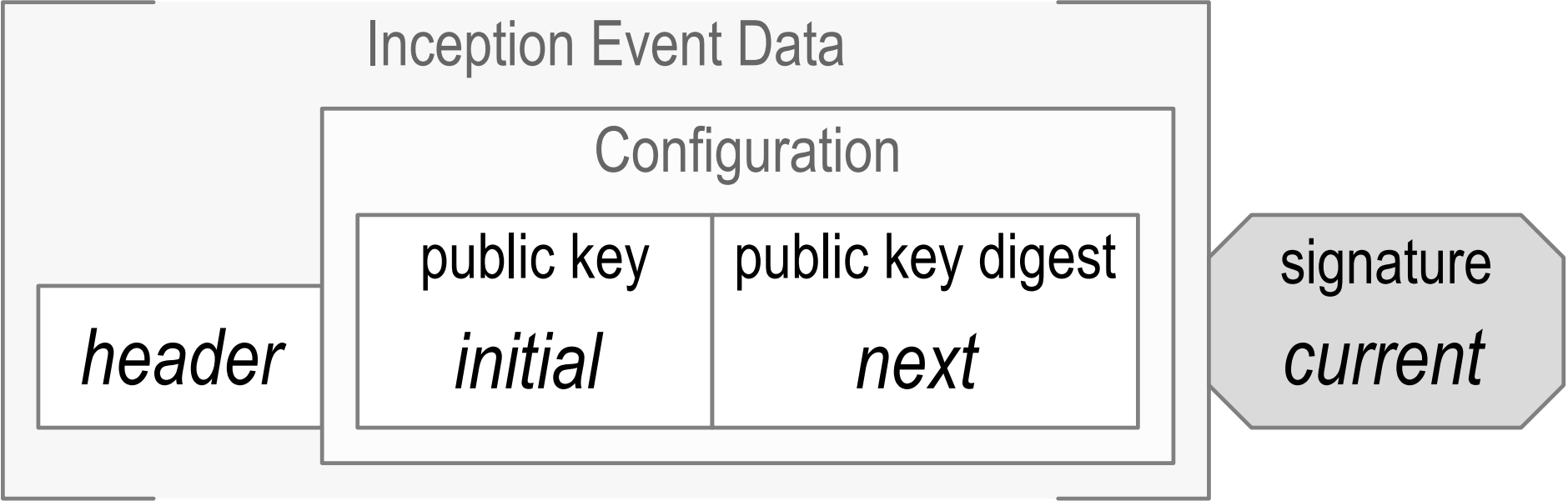
Seals make KERI both privacy preserving and *data semantic agnostic*.

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.

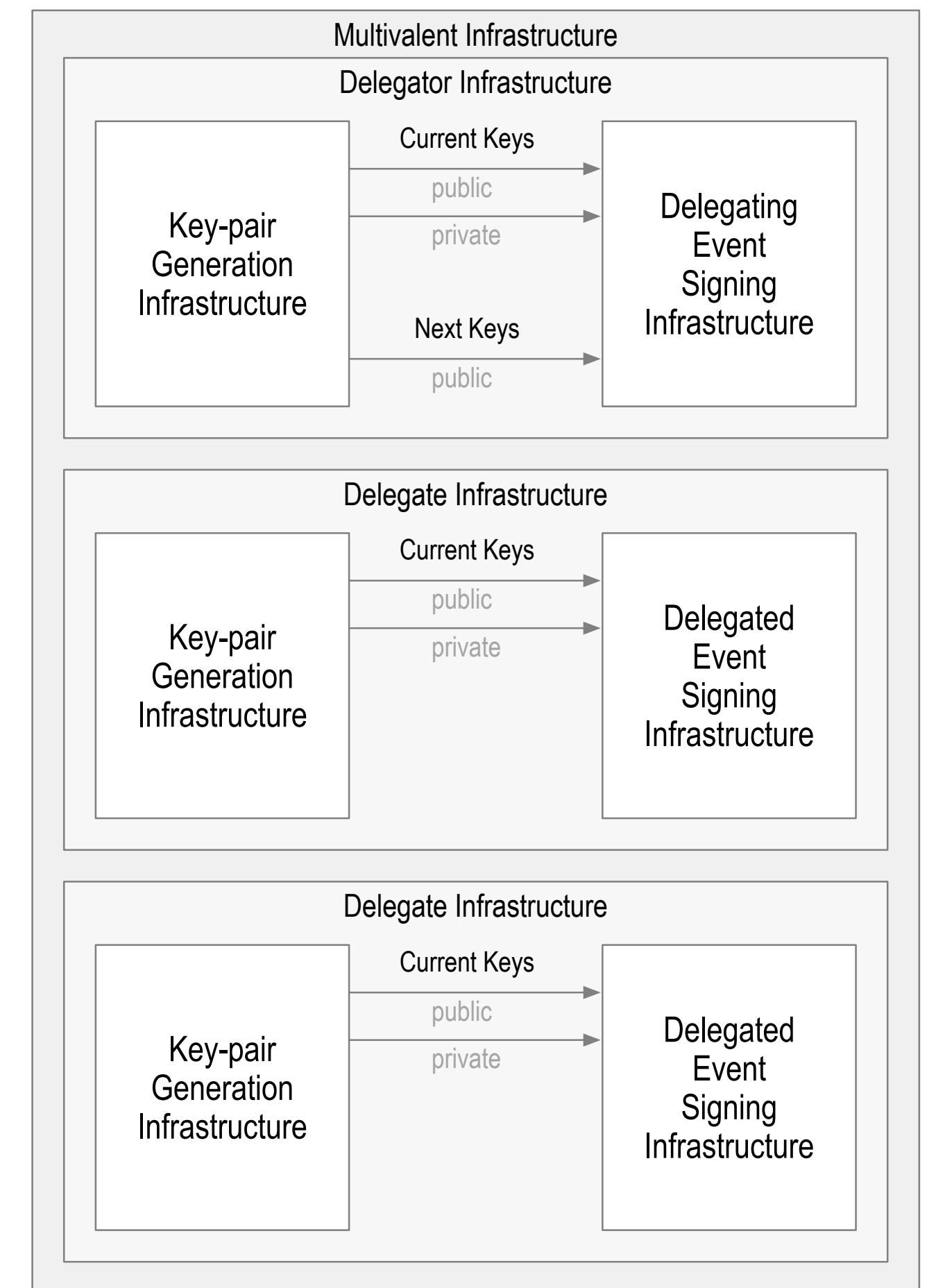
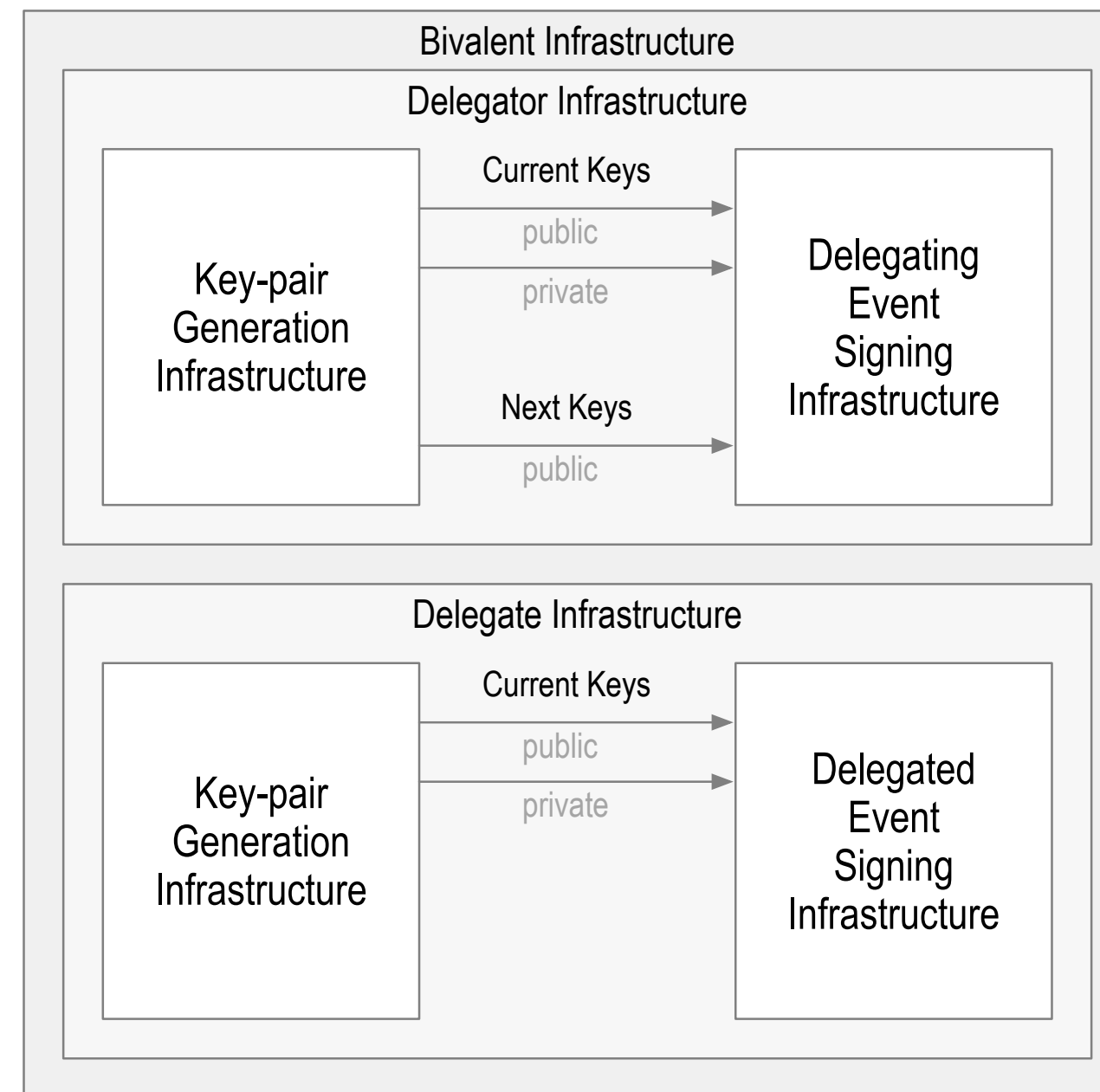
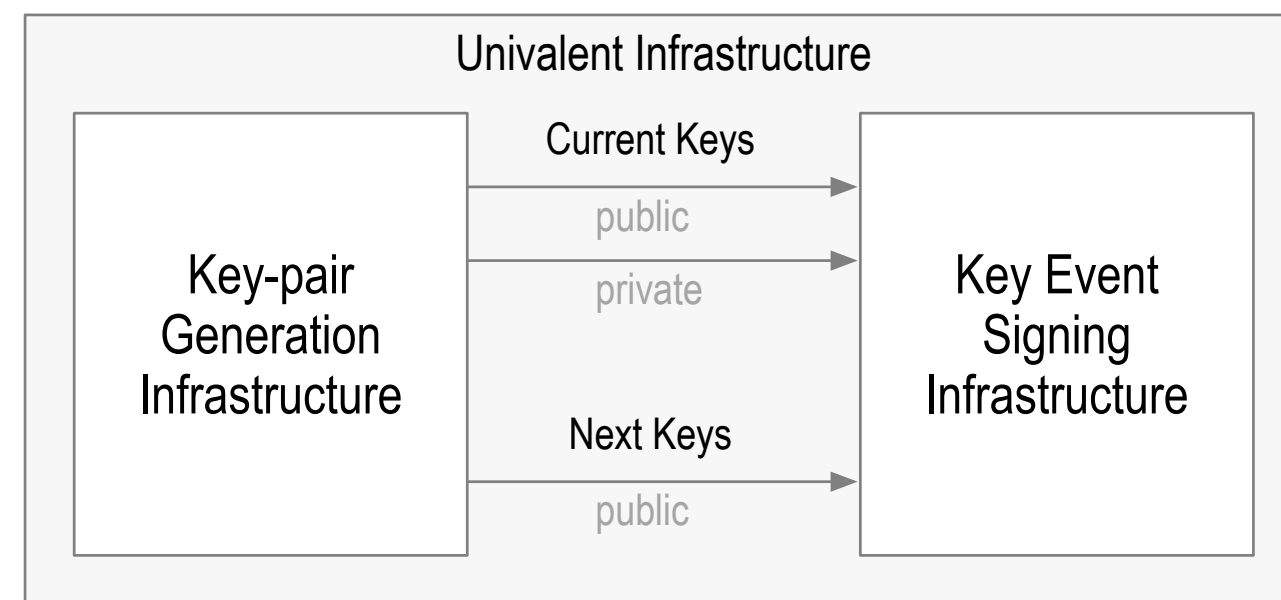
Minimally sufficient means.

Pre-Rotation

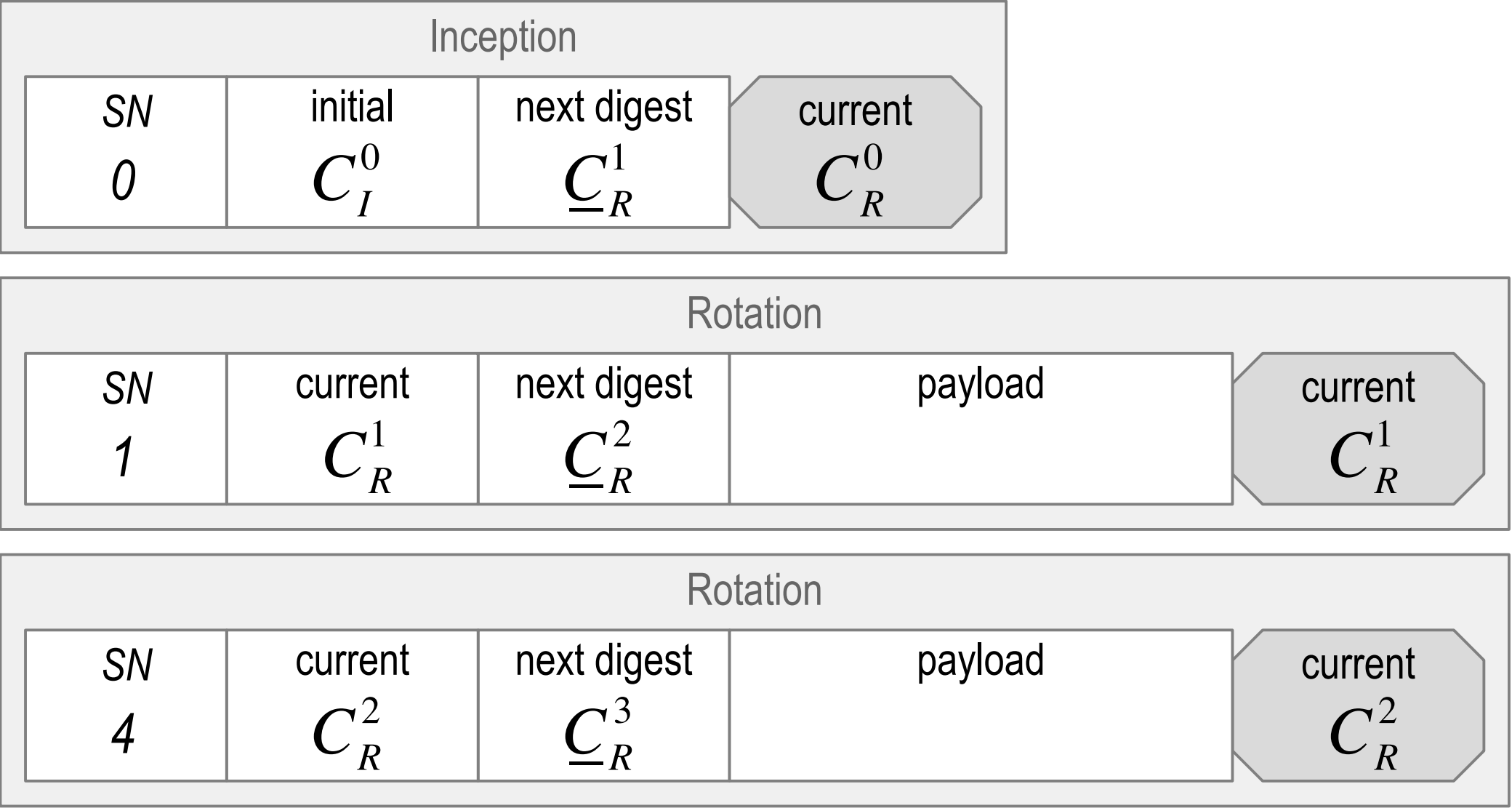
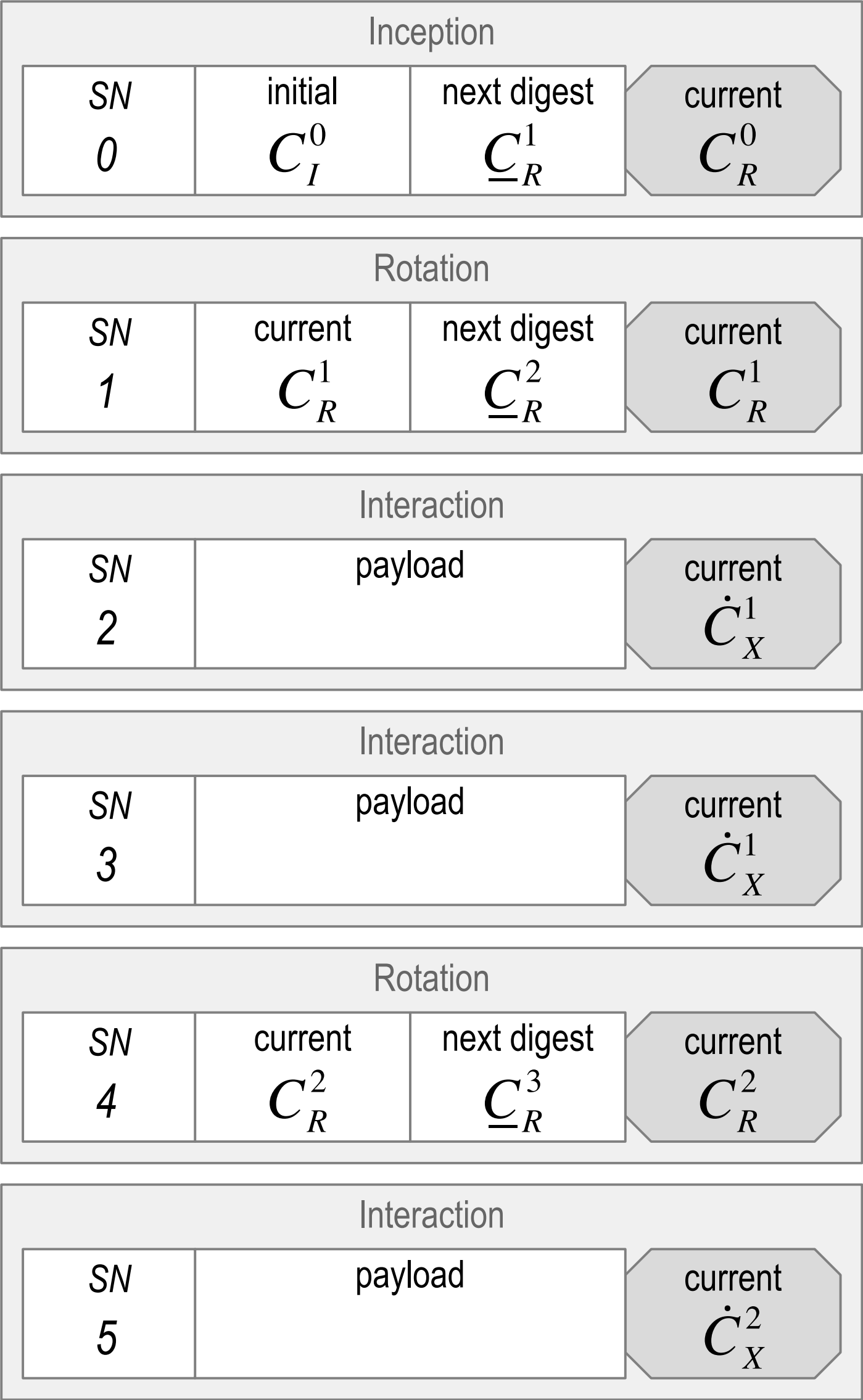


Digest of *next* key(s) makes pre-rotation post-quantum secure

Key Infrastructure Valence

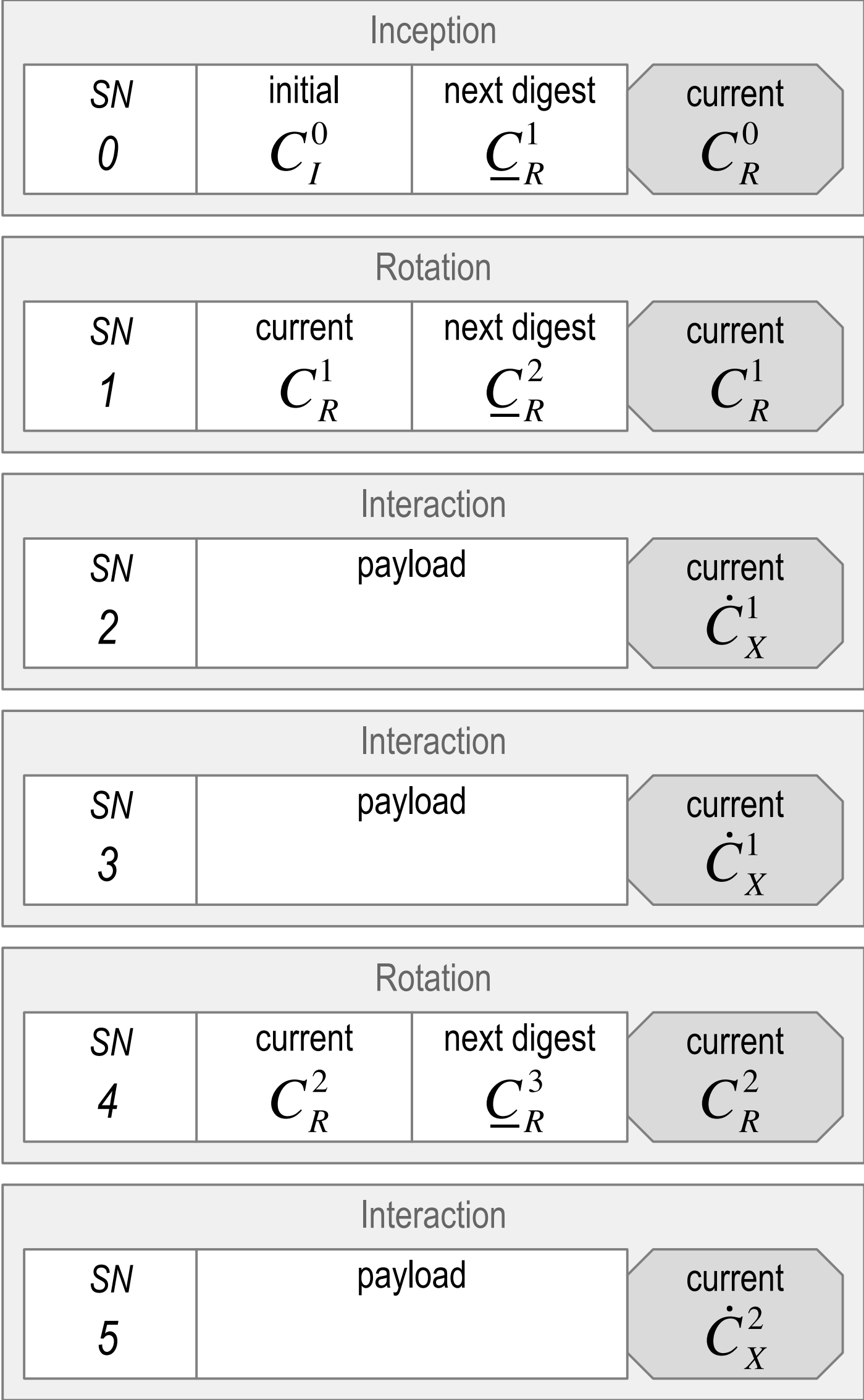


Repurposed Keys

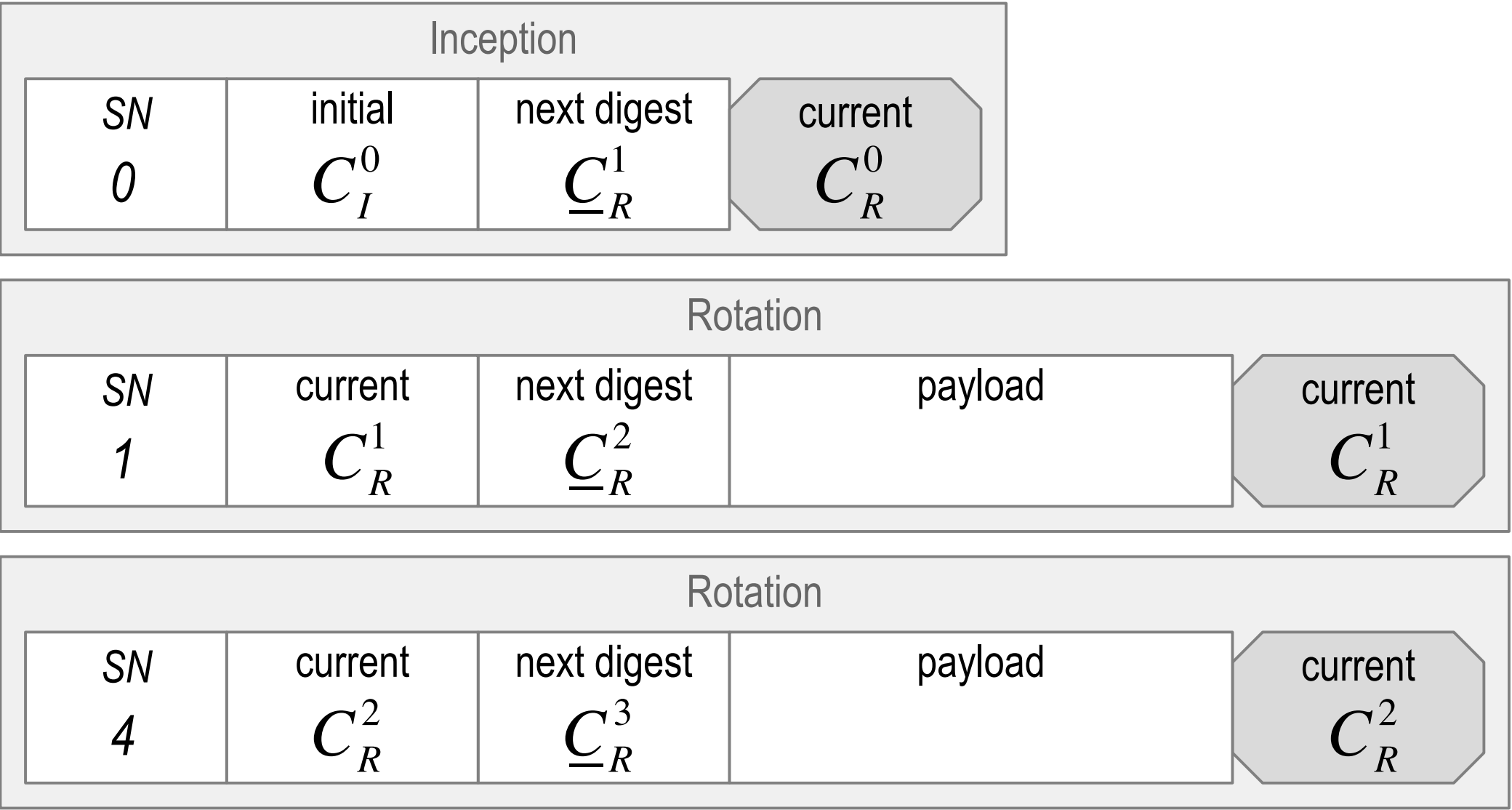


Univalent Key Roles

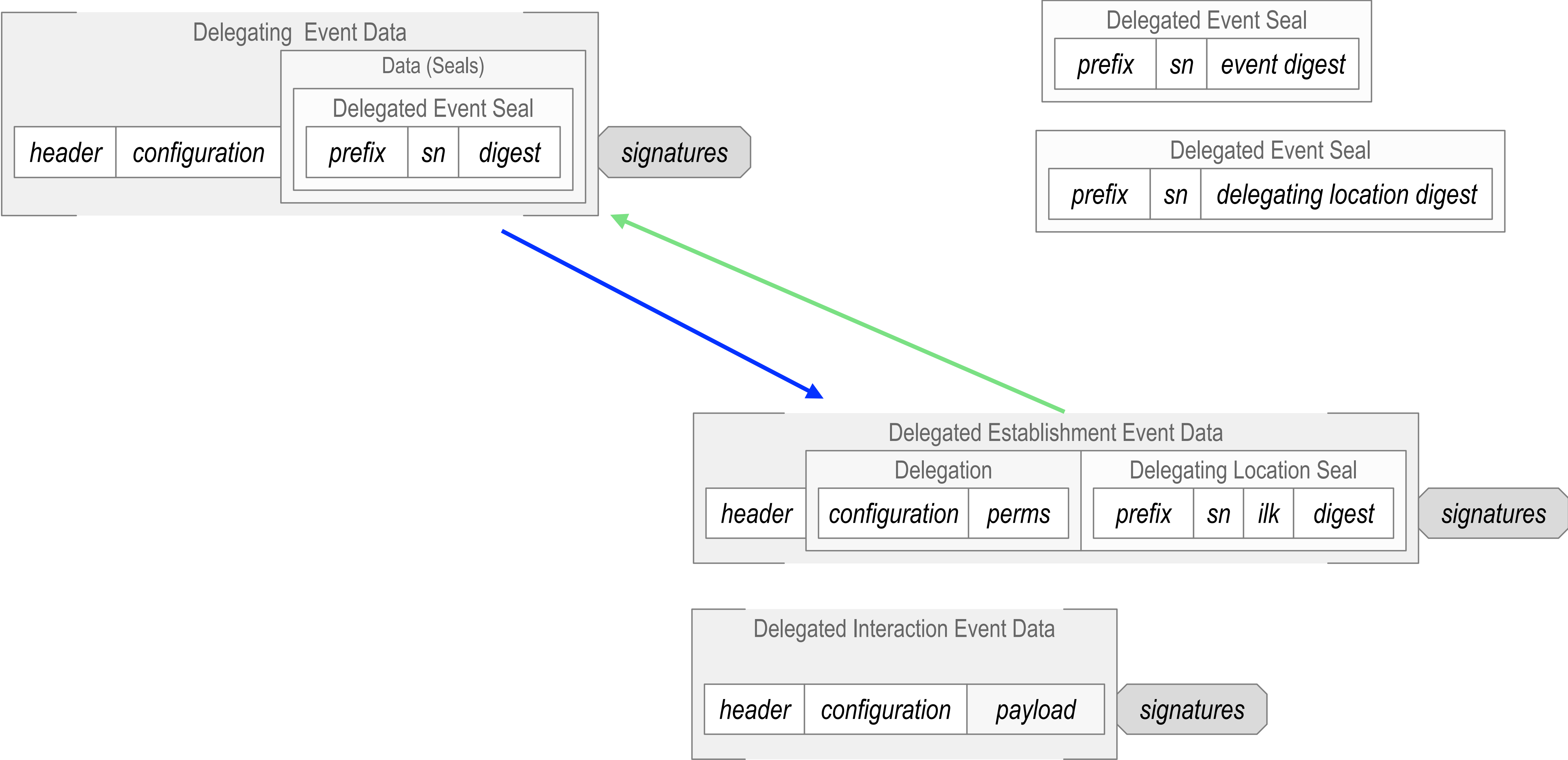
Repurposed Rotation to Interaction



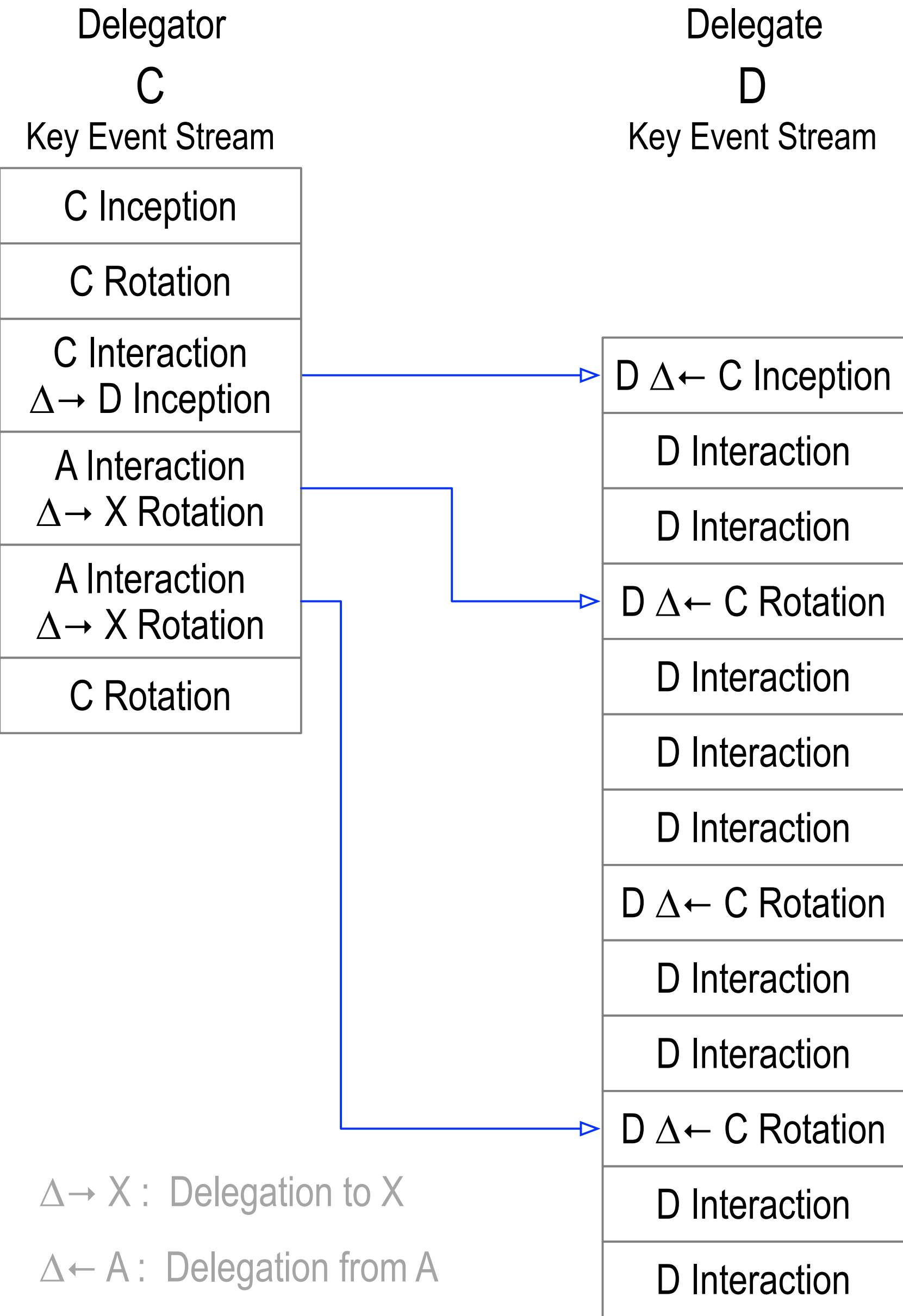
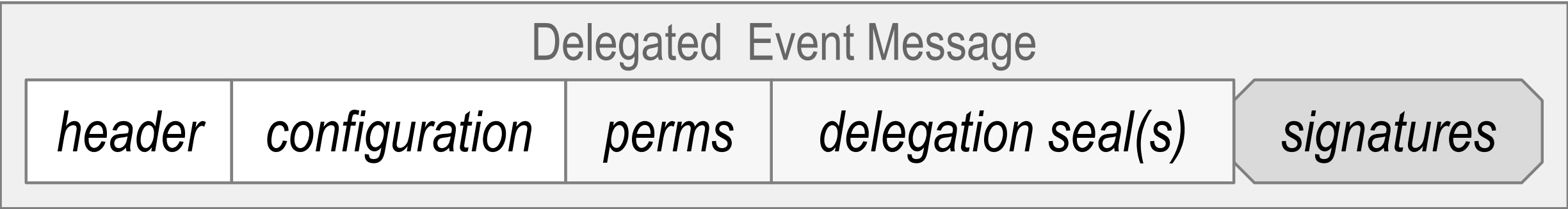
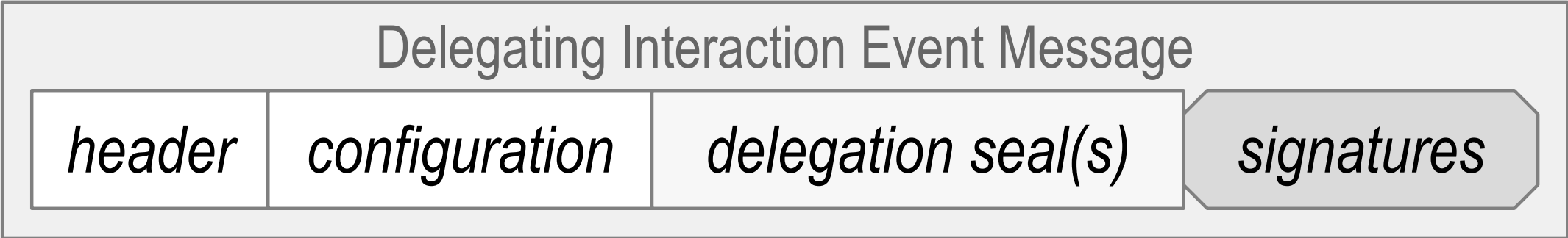
Rotation Only



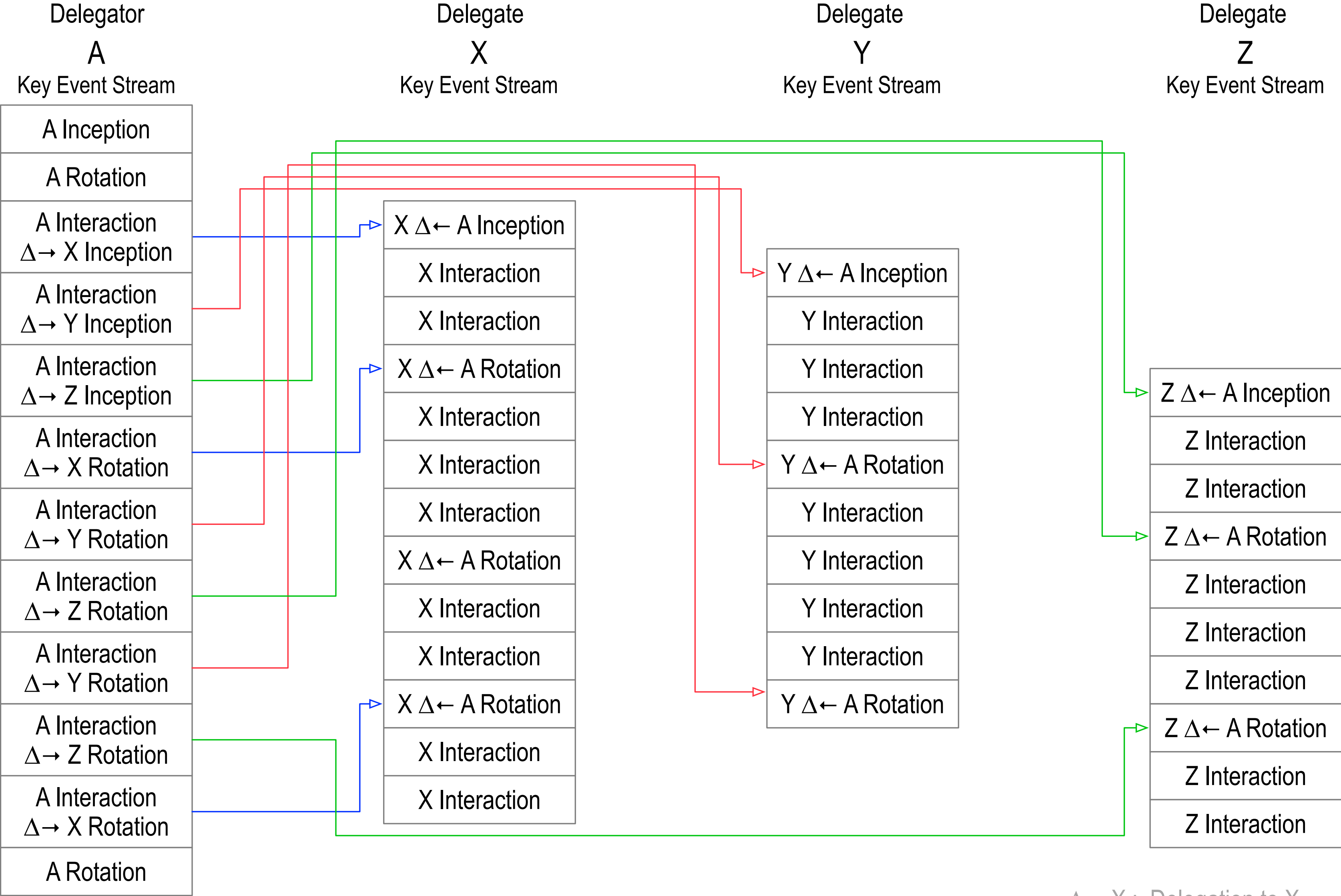
Delegation (Cross Anchor)



Interaction Delegation

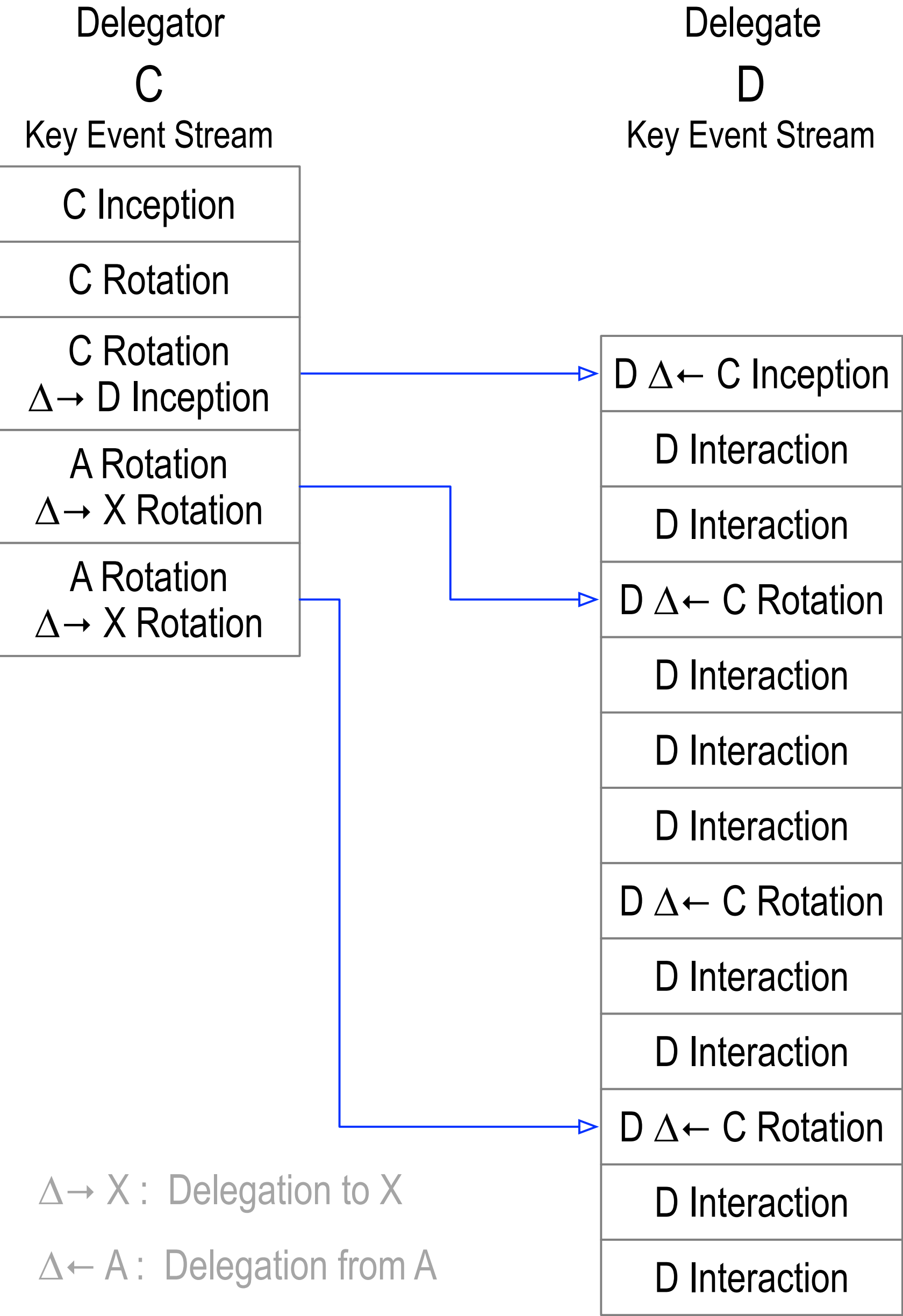
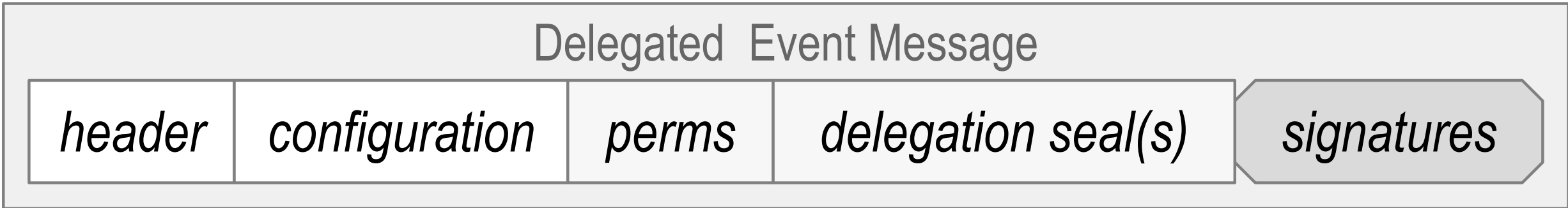
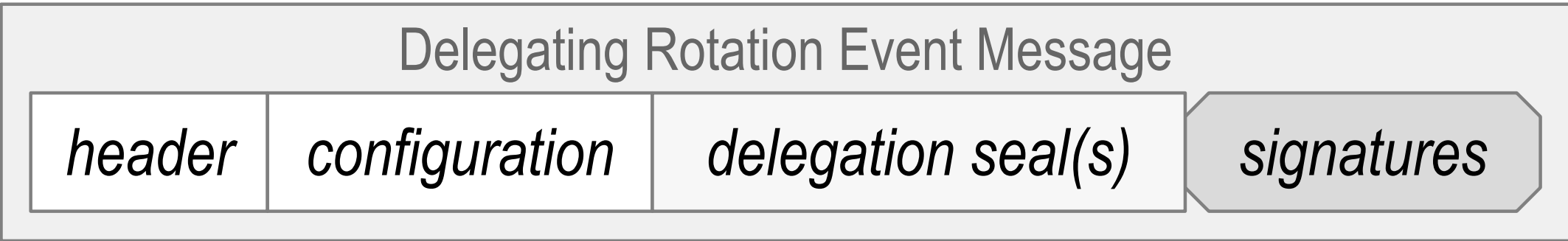


Scaling Delegation via Interaction

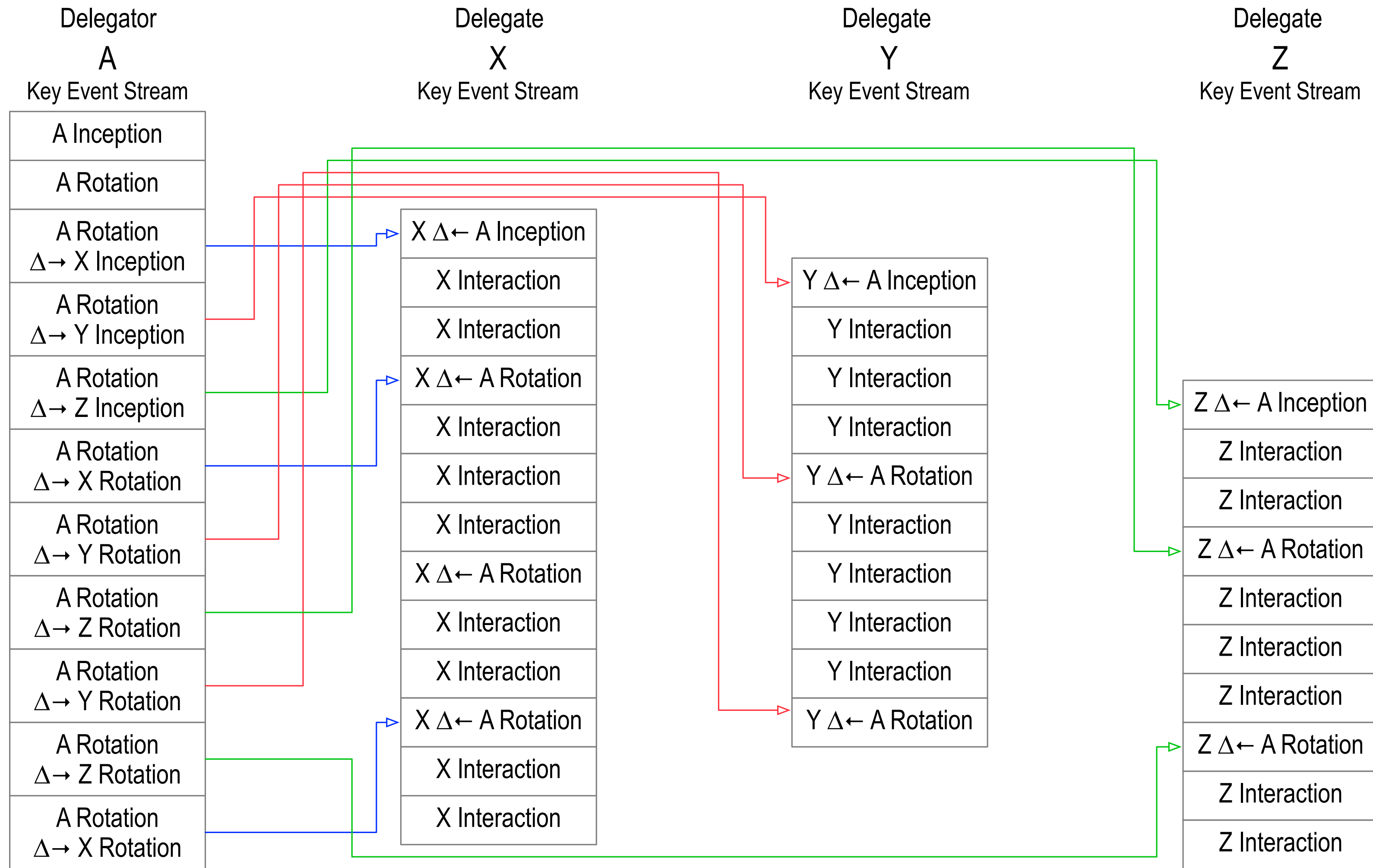


$\Delta \rightarrow X$: Delegation to X
 $\Delta \leftarrow A$: Delegation from A

Rotation Delegation

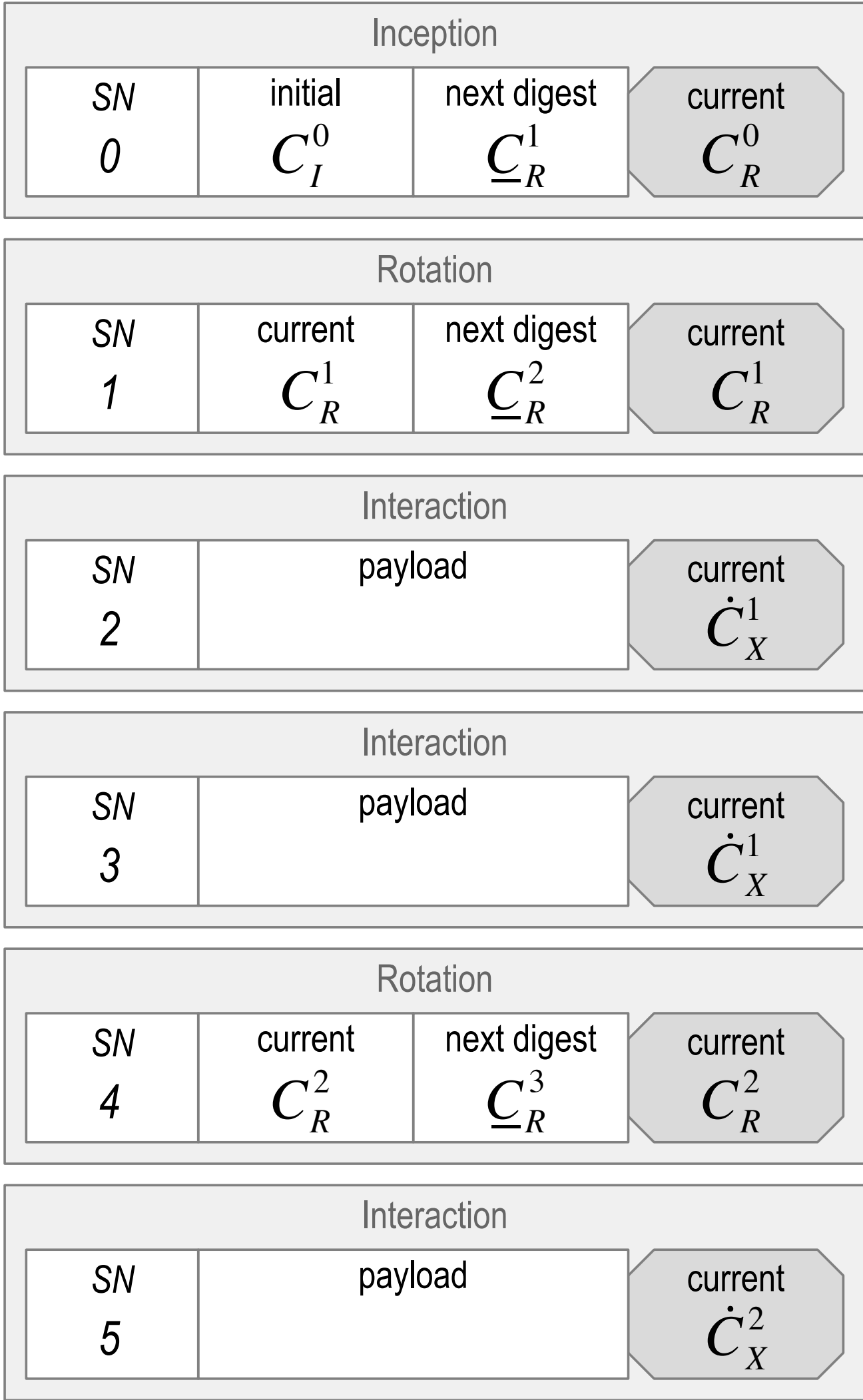


Scaling Delegation via Rotation



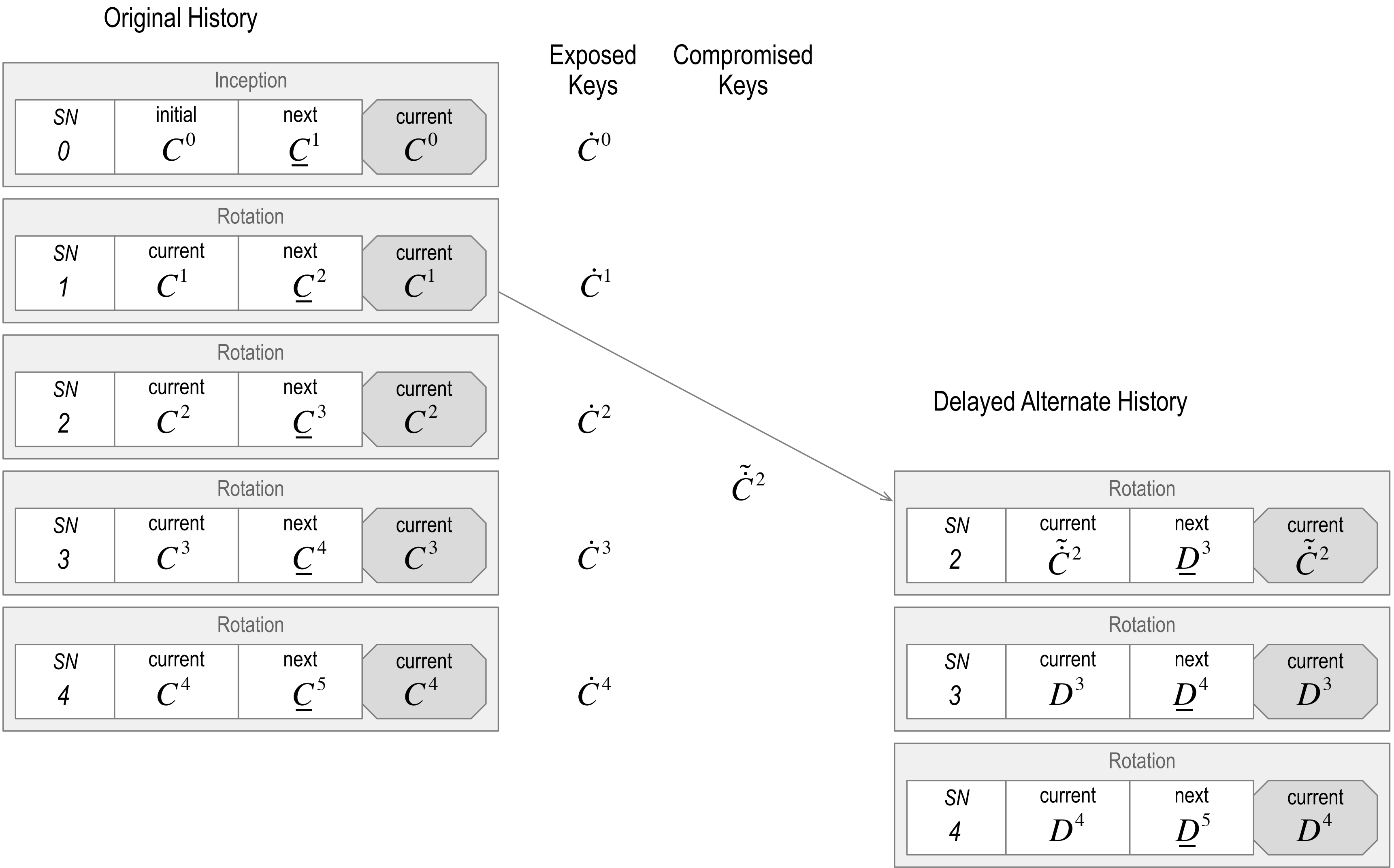
$\Delta \rightarrow X$: Delegation to X
 $\Delta \leftarrow A$: Delegation from A

Live Exploit (current signing keys)



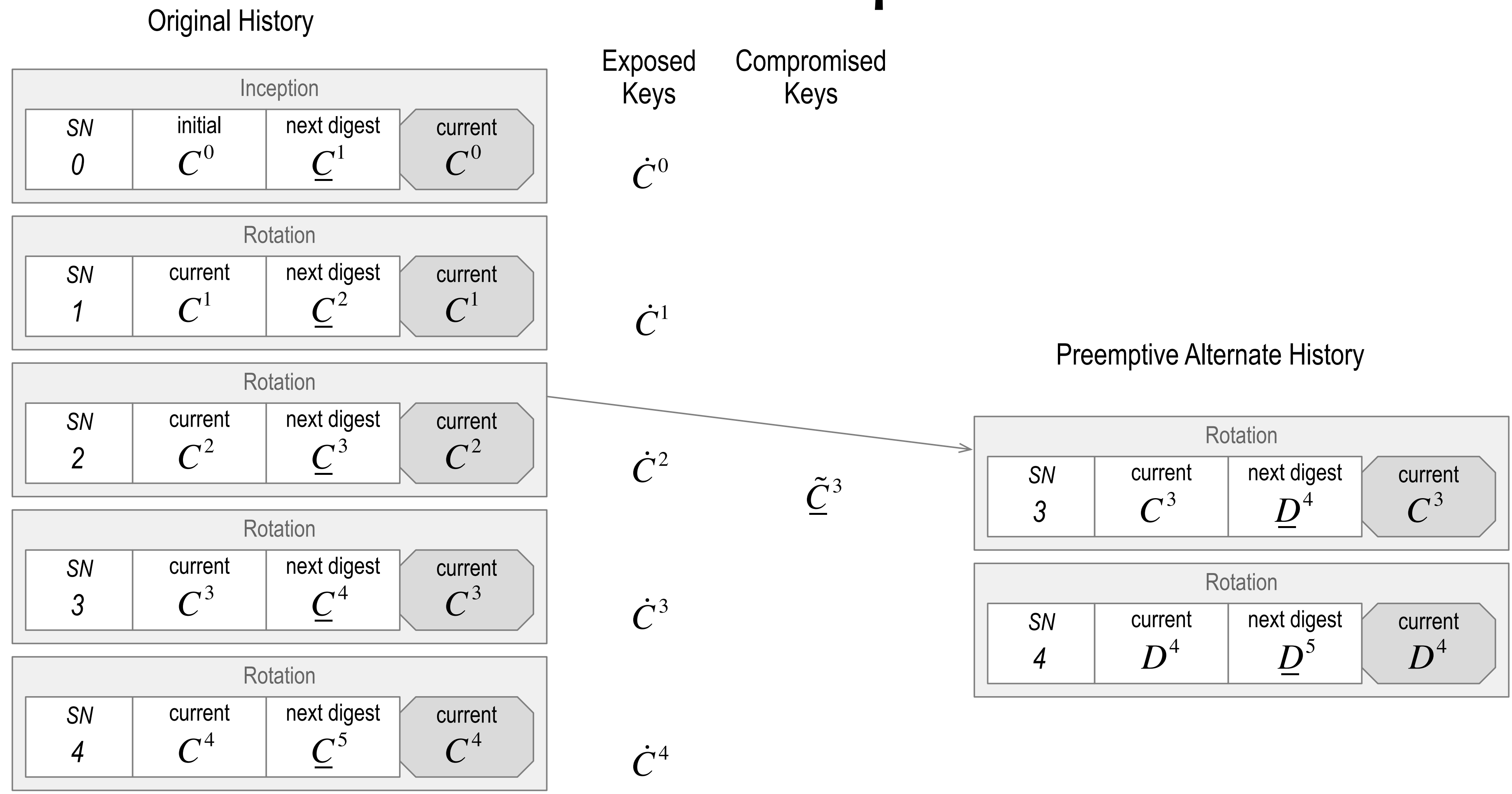
Pre-rotation provides protection from successful *live* exploit of current signing keys.

Dead Exploit (stale next signing keys)



Any copy of original history protects against successful *dead* exploit

Live Exploit (next signing keys)



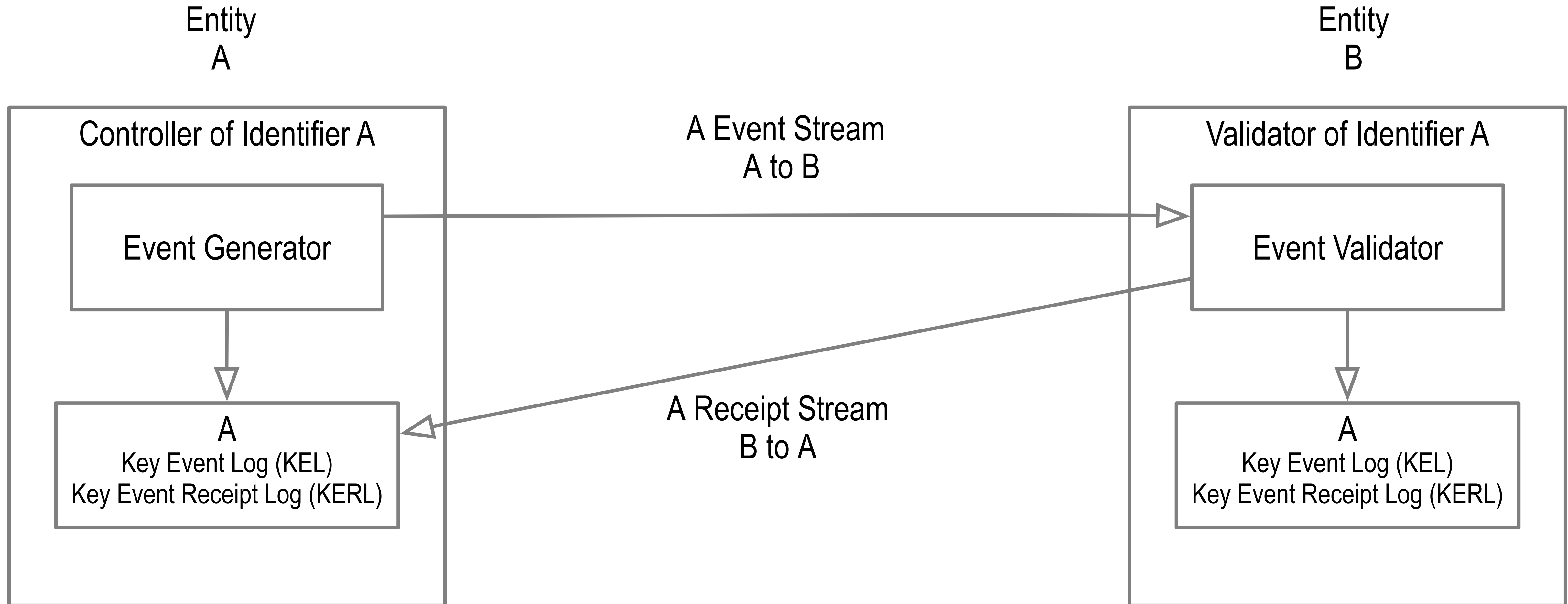
Difficulty of inverting next key(s) protects against successful *live* exploit.

Protocol Operational Modes

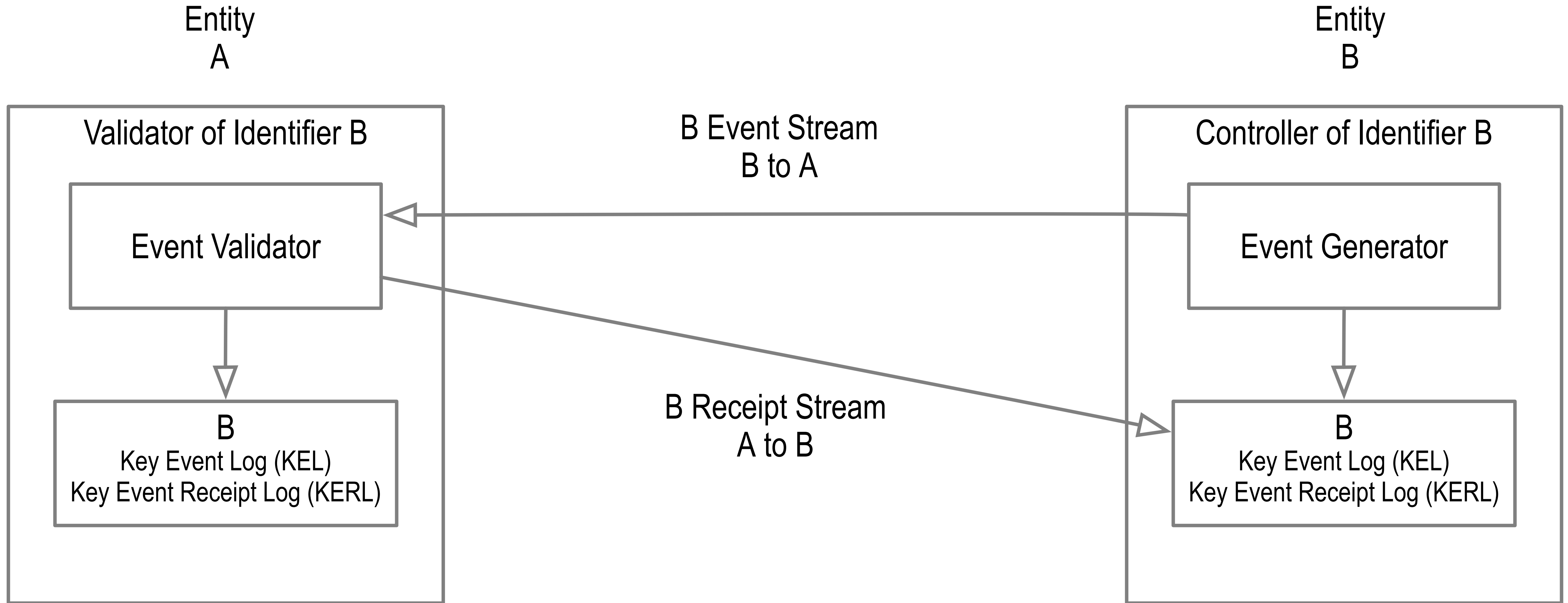
Direct Event Replay Mode (one-to-one)

Indirect Event Replay Mode (one-to-any)

Direct Mode: A to B

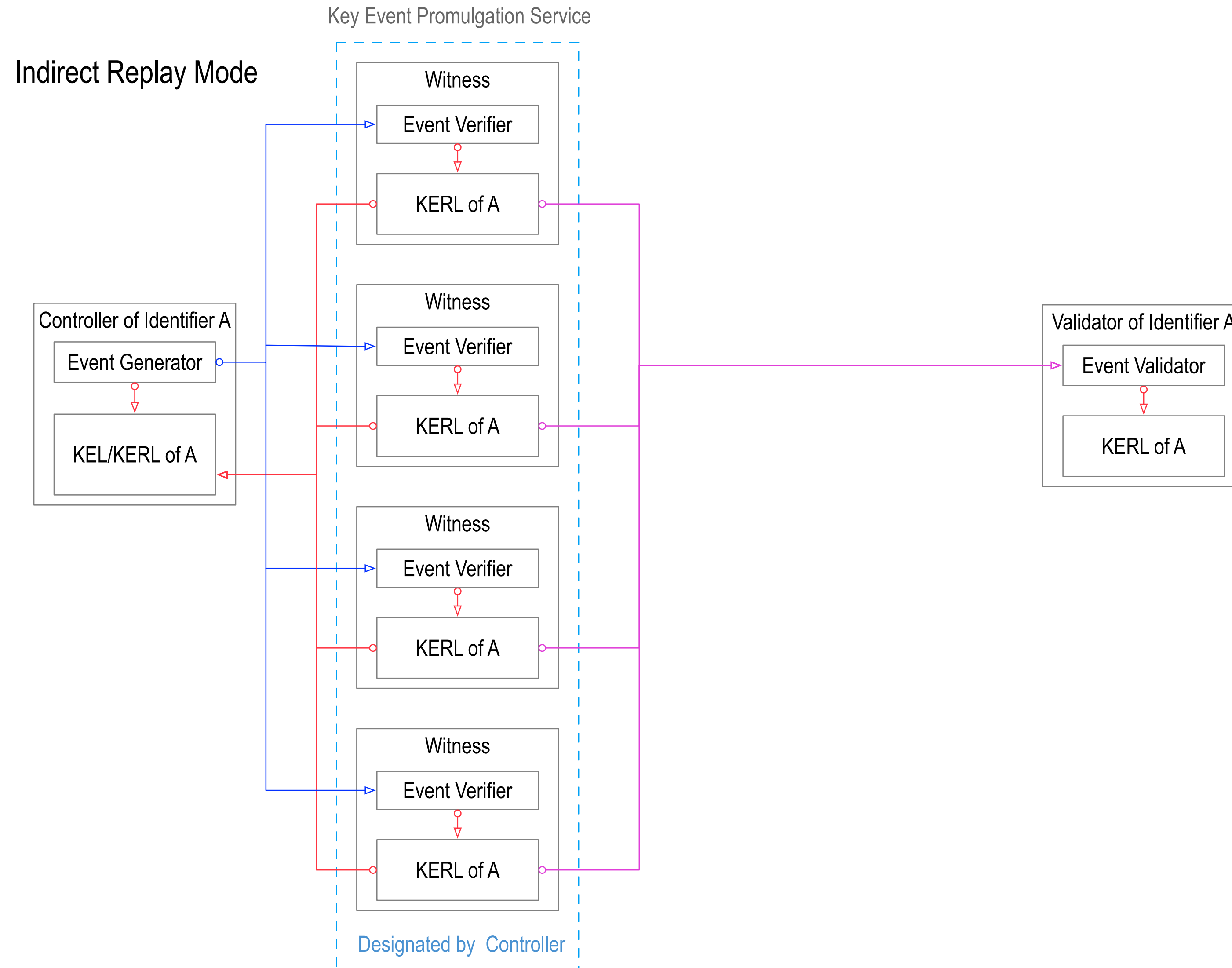


Direct Mode: B to A



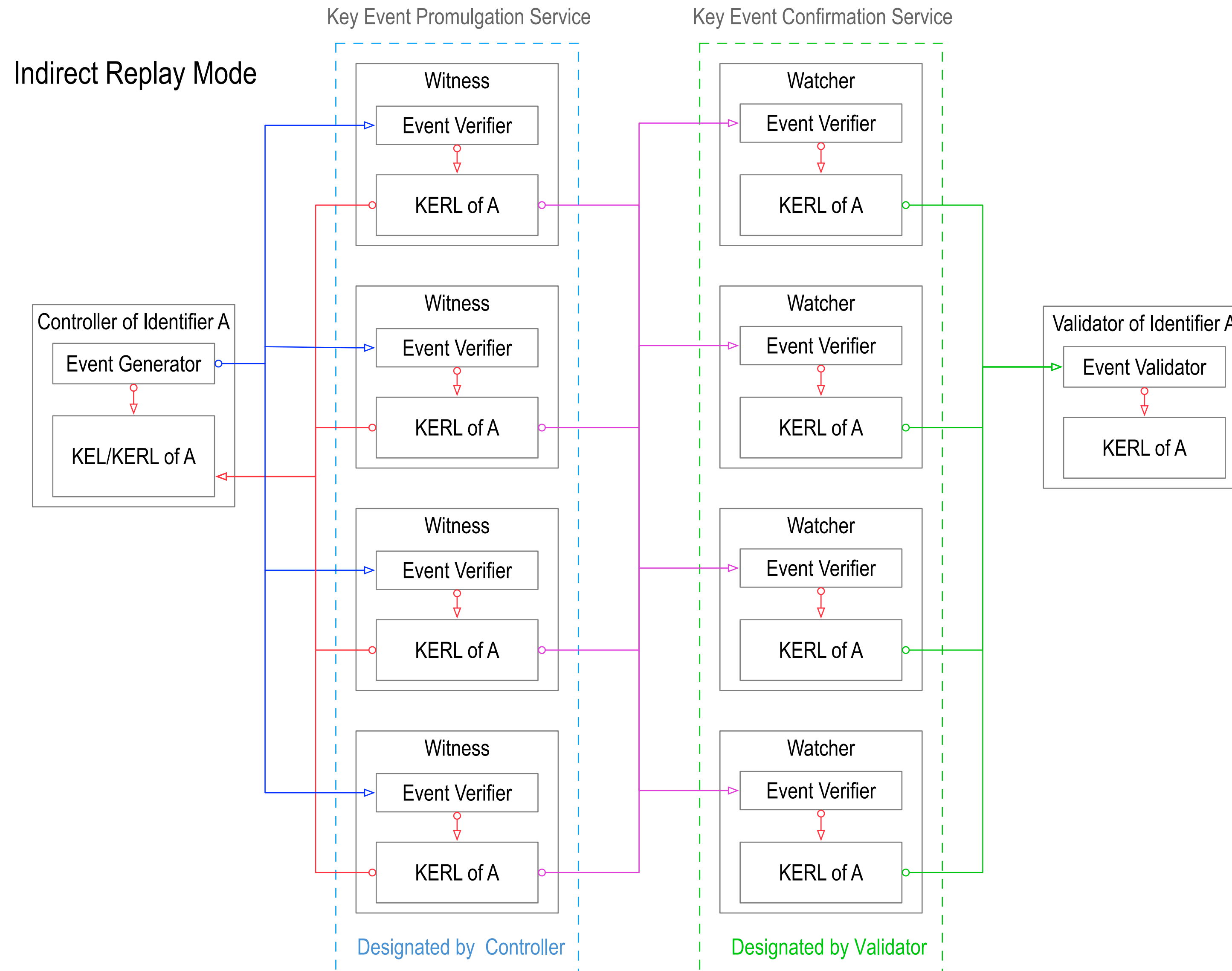
Indirect Mode

Promulgation Service



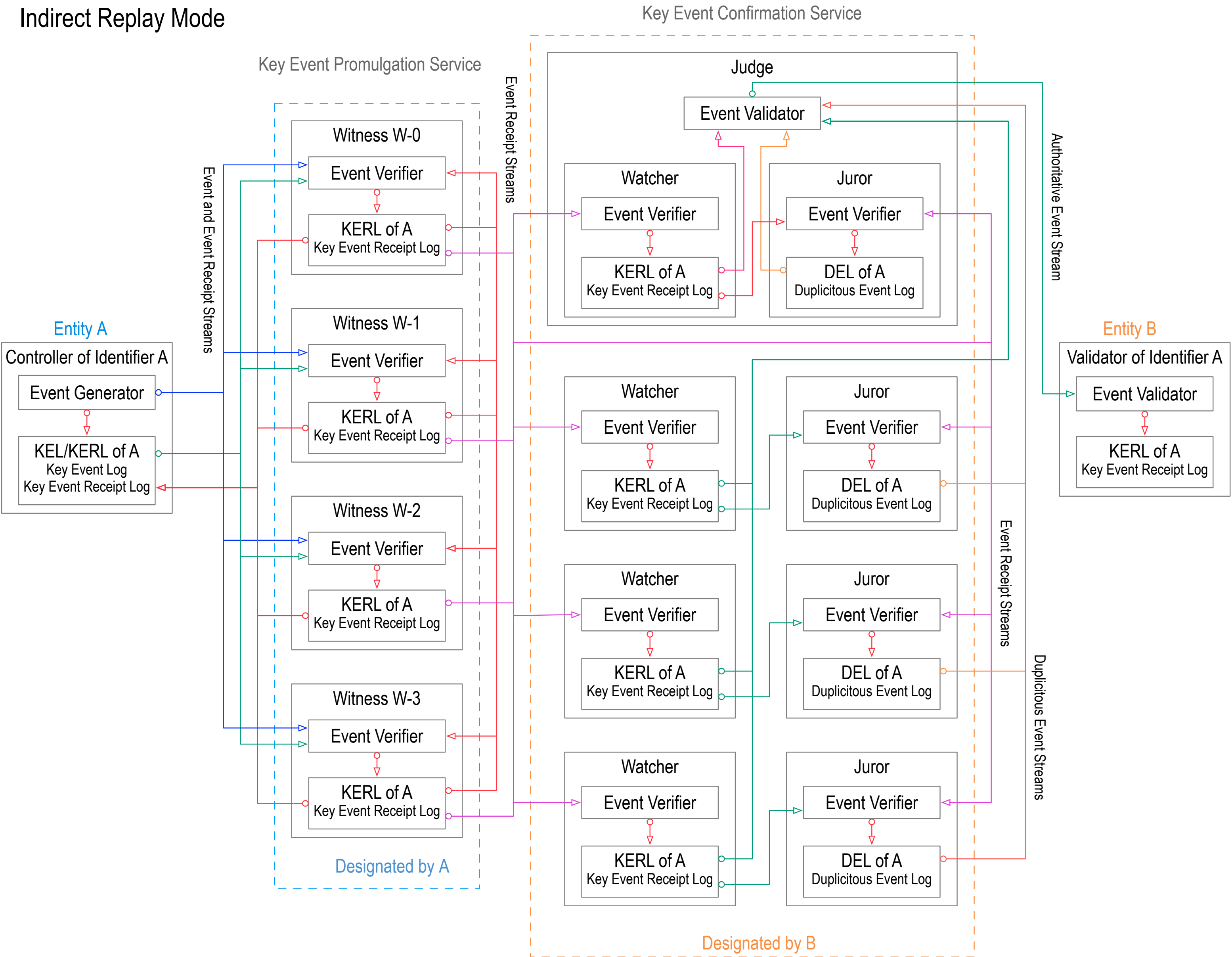
Indirect Mode

Promulgation and Confirmation Services



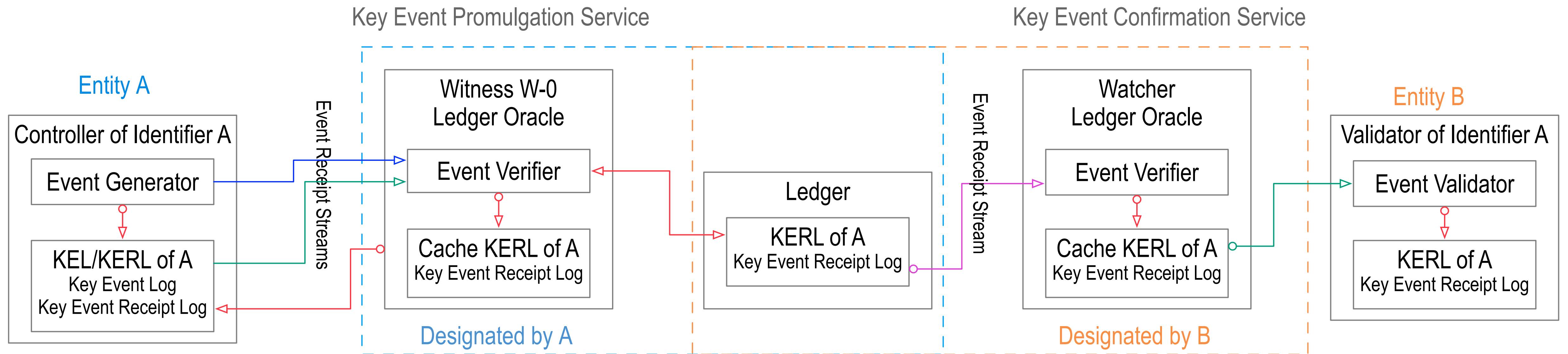
Indirect Mode Full

Indirect Replay Mode



Indirect Mode with Ledger Oracles

Indirect Replay Mode with Ledger Oracle



Separation of Control

Shared (permissioned) ledger = *shared control* over *shared data*.

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

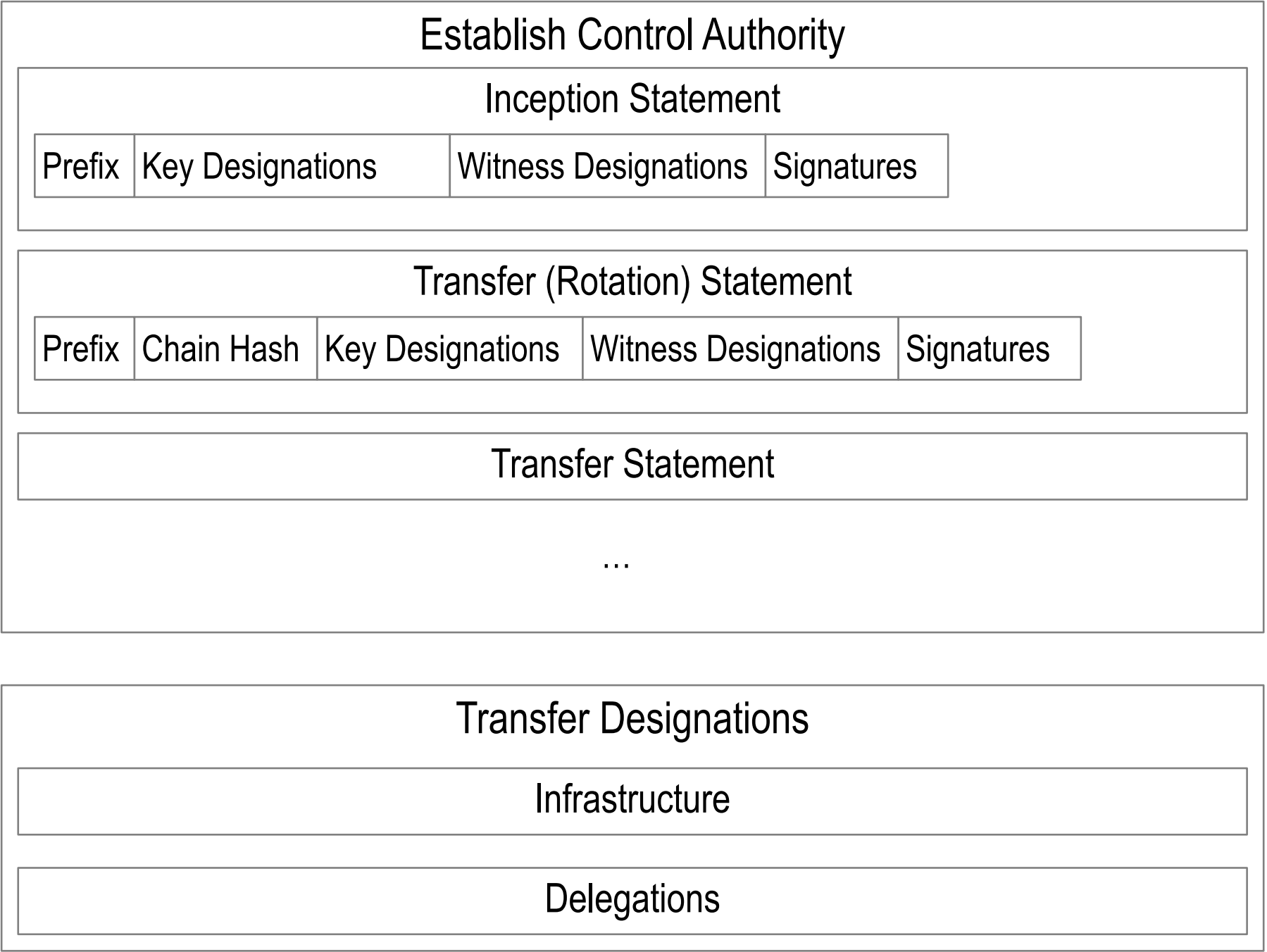
Shared control between controller and validator may be problematic for governance, scalability, and performance.

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

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

Function Stack

KERI



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

On Top of KERI

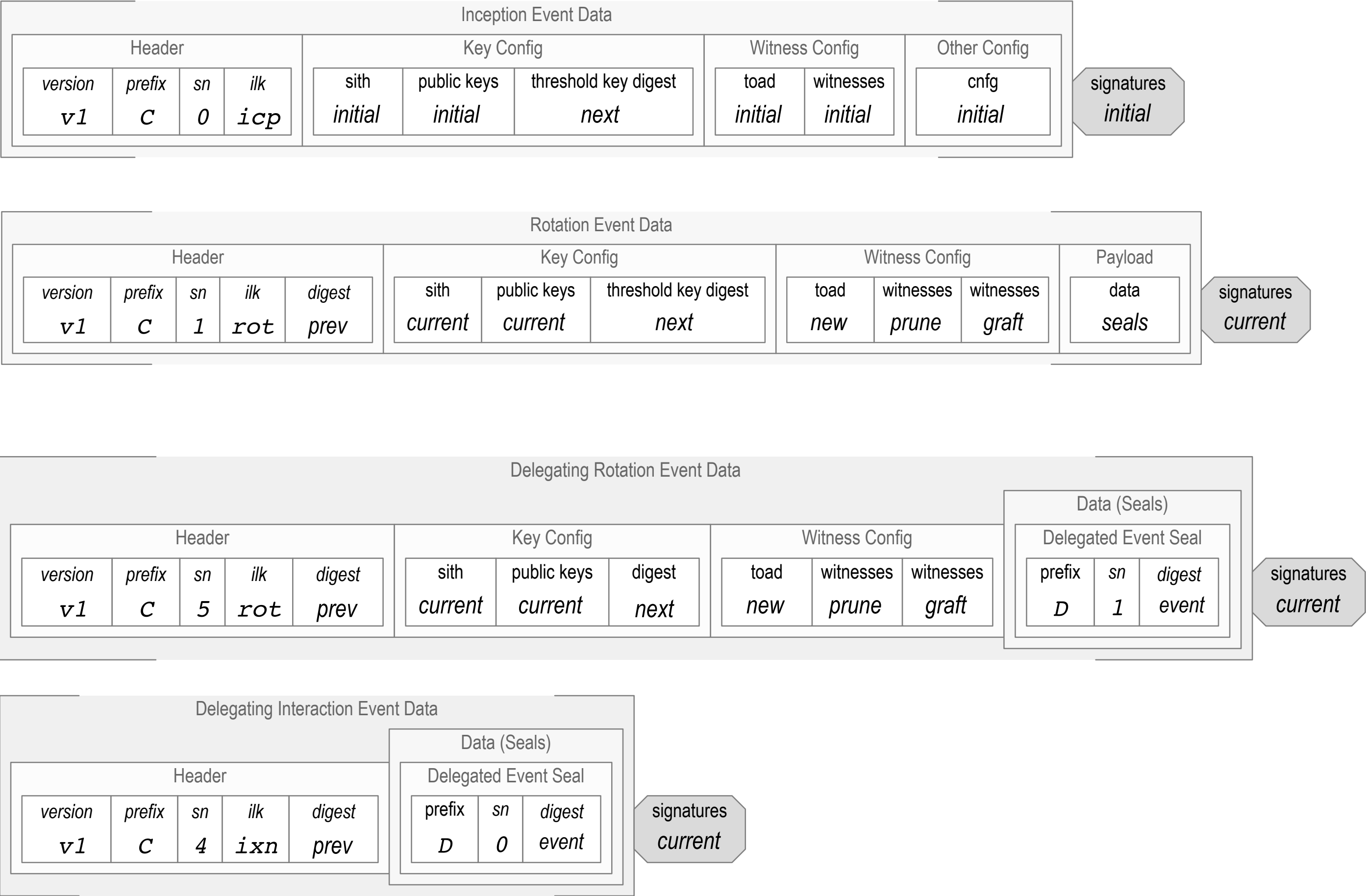
Rotate Prefix vs Rotate Keys

Non-transferable may not rotate keys. May only rotate prefix

Rotate prefix good for bootstrapping. No key event log (KEL) needed.

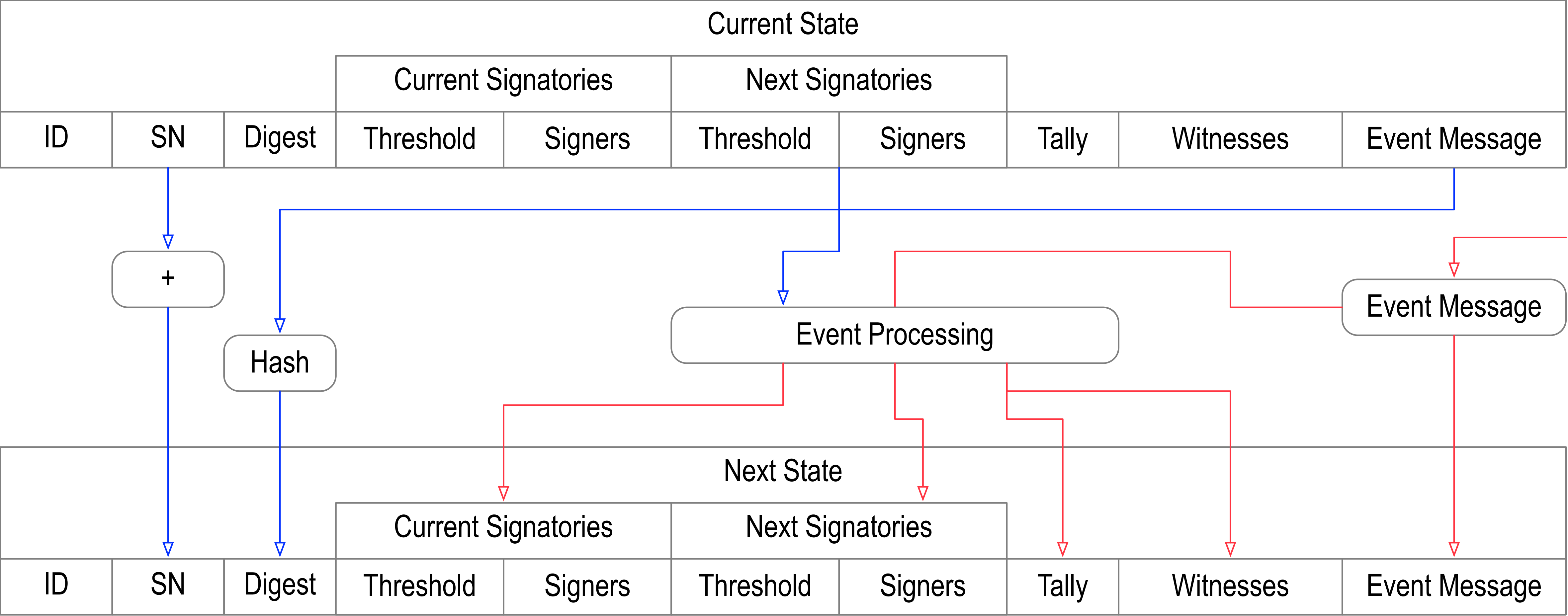
If prefix has no persistent value outside its function and its function may be marshaled by some other prefix controller then rotating prefix may be preferred.

Events



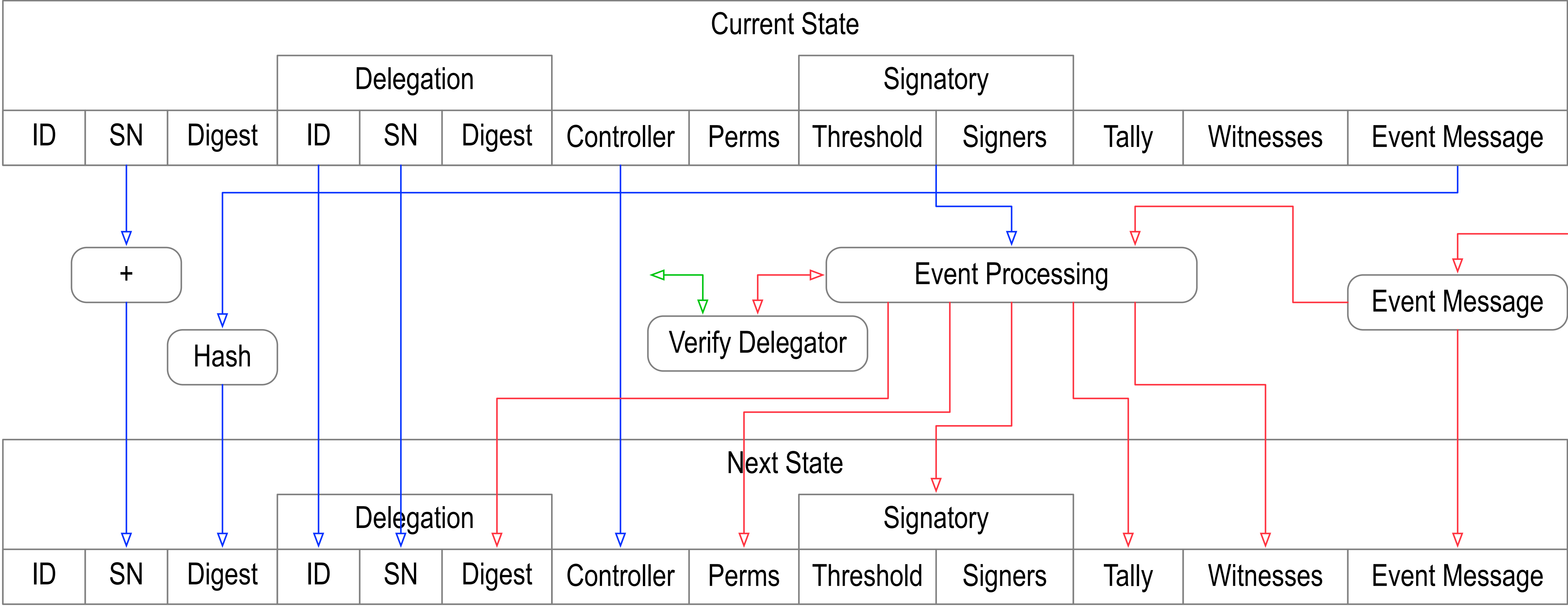
State Verifier Engine

KERI Core — State Verifier Engine

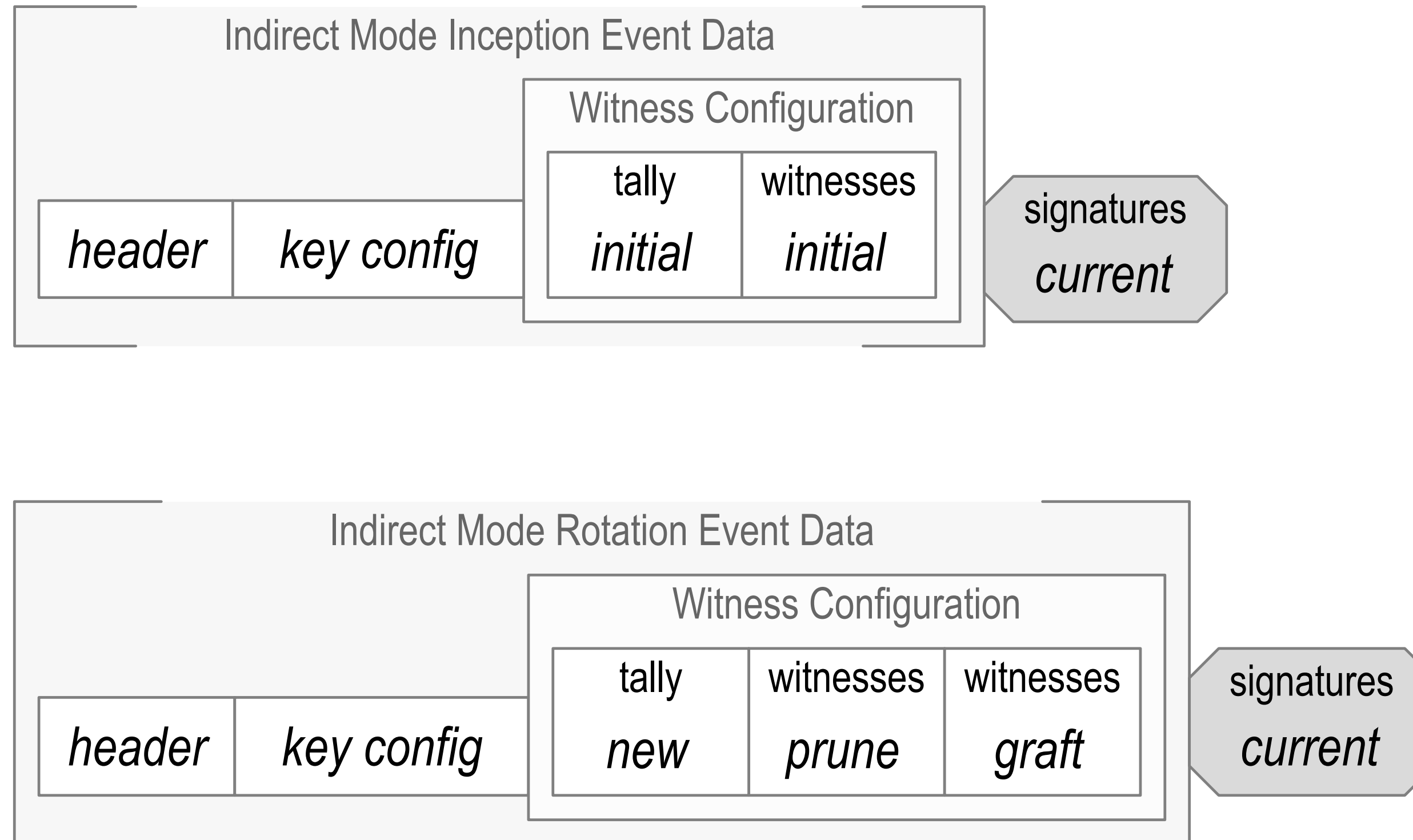


Delegated State Verifier Engine

KERI Delegated Core — State Verifier Engine



Witness Designation



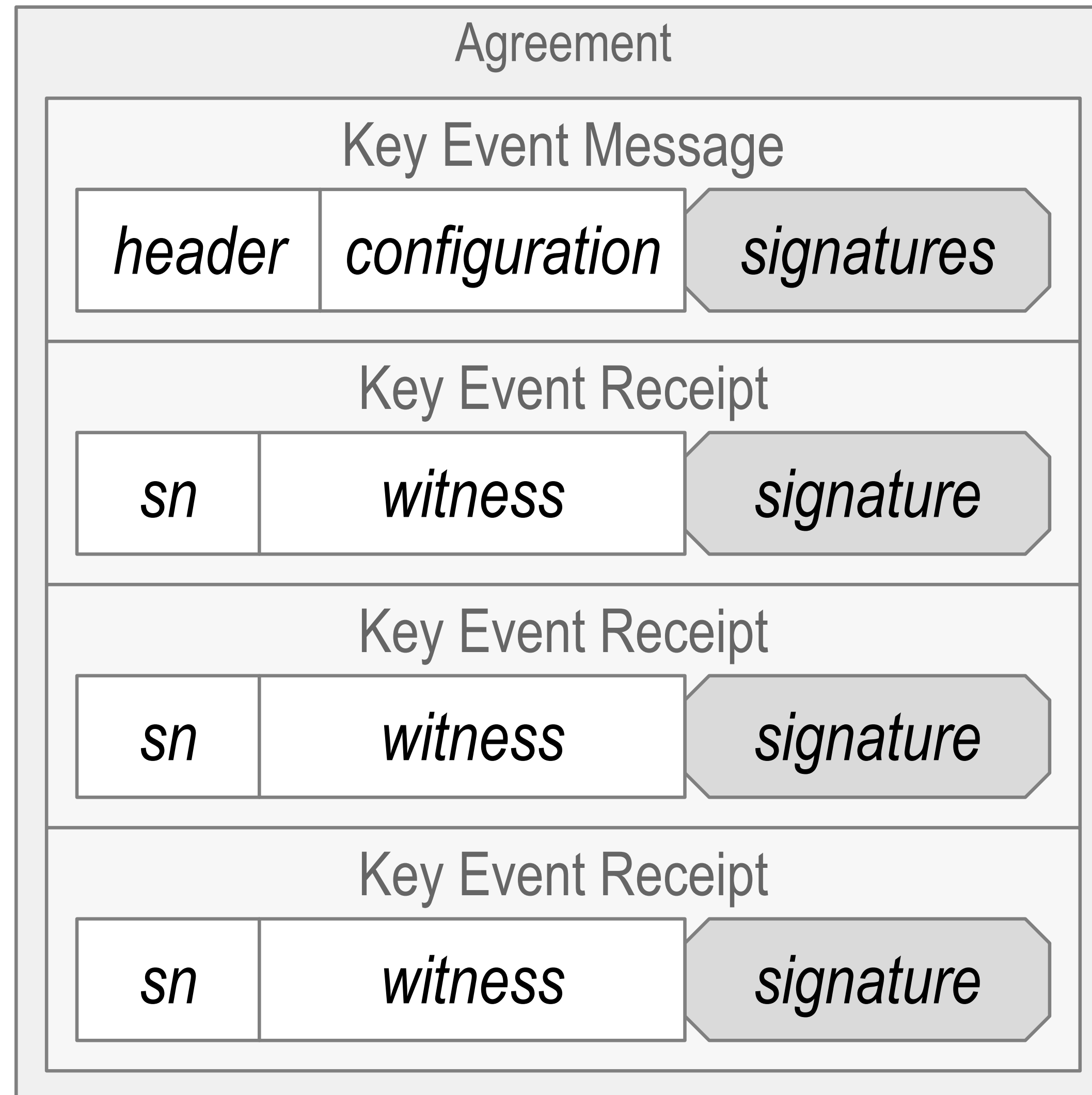
Witnessed Key Event Receipt

Key Event Receipt						
<i>version</i>	<i>prefix</i>	<i>sn</i>	<i>ilk</i>	<i>digest</i>	<i>witness</i>	<i>signature</i>

(KA²CE)

Keri's Agreement Algorithm for Control Establishment

Produce Agreements
with Guarantees



Agreement Constraints

Proper Agreement

$$F + 1$$

Sufficient Agreement

$$M > F$$

$$M \leq N - F$$

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

Intact Agreement

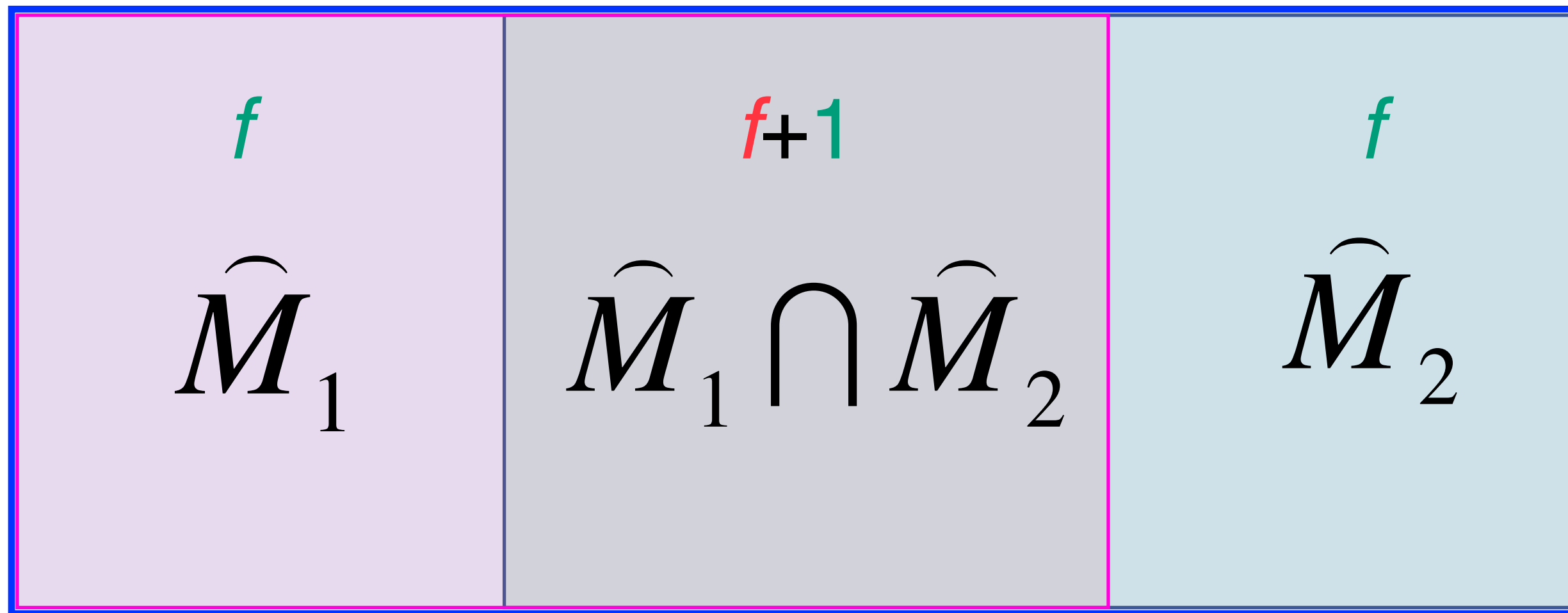
$$N \geq 2F + 1$$

One Agreement or None at All

$$|\hat{N}| = N \quad |\hat{M}_1| = |\hat{M}_2| = M$$

Overlapping Sets

$$\hat{M}_1 \cup \hat{M}_2 = \hat{N}$$



One honest witness if:

$$|\hat{M}_1 \cap \hat{M}_2| \geq F + 1$$

$$|\hat{M}_1 \cup \hat{M}_2| = |\hat{N}| = N$$

$$|\hat{M}_1| + |\hat{M}_2| = |\hat{M}_1 \cup \hat{M}_2| + |\hat{M}_1 \cap \hat{M}_2|$$

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

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

$$M \leq N - F$$

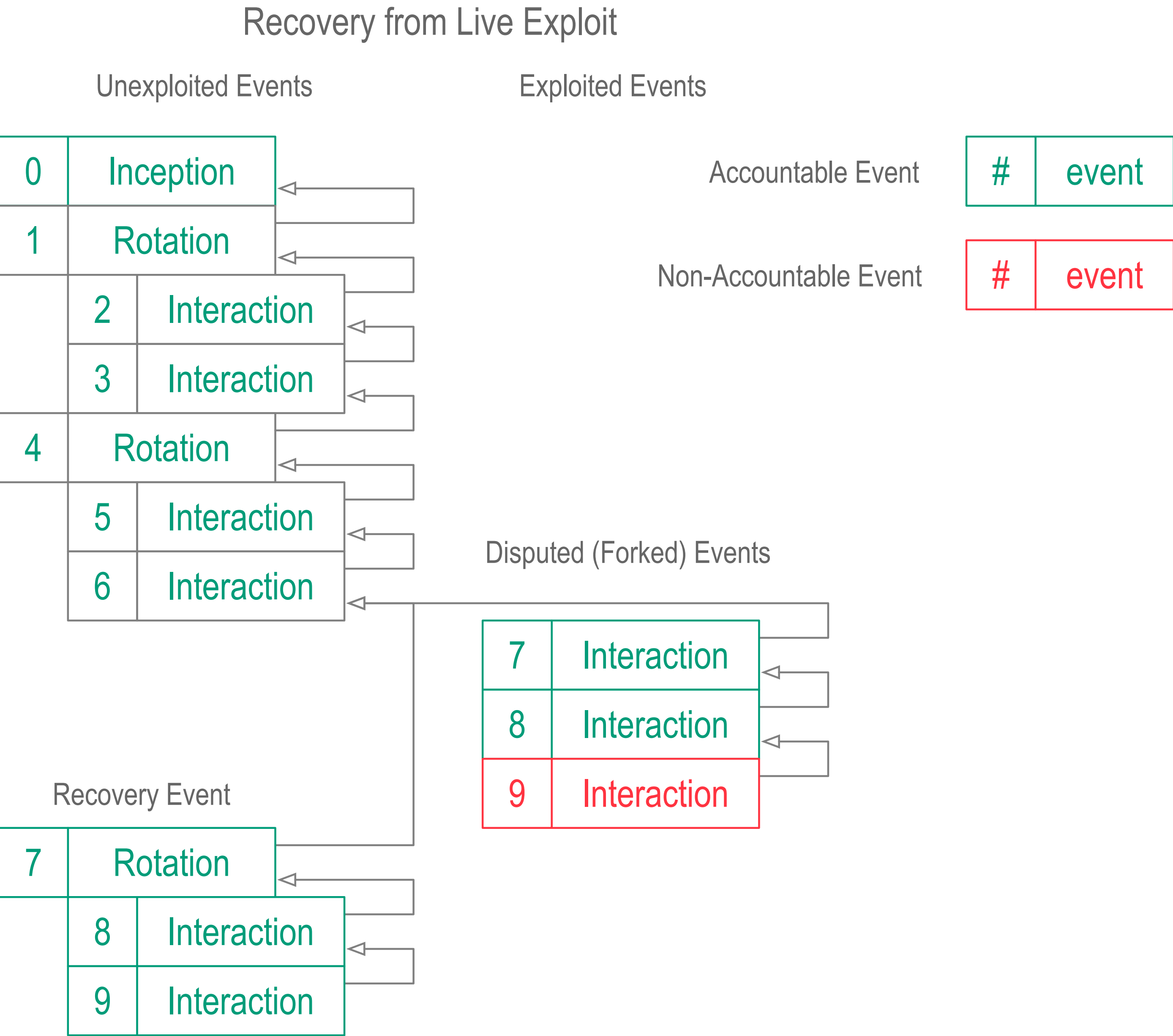
Immune Agreement

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

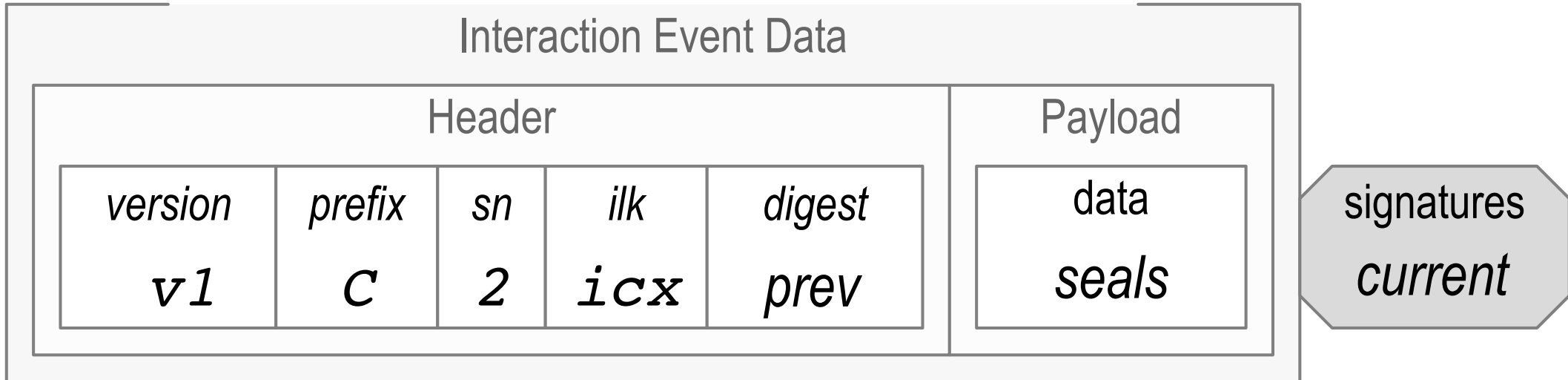
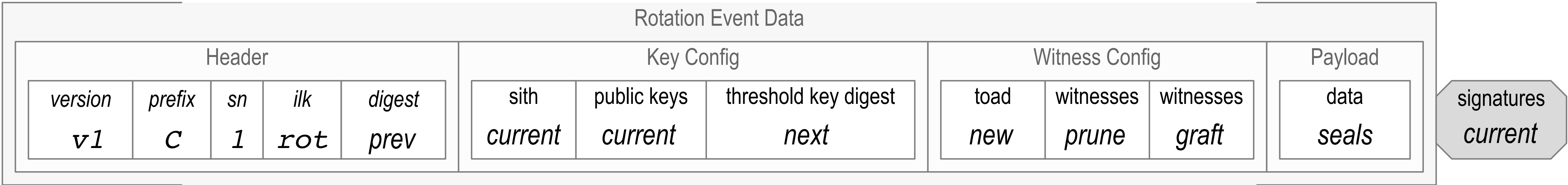
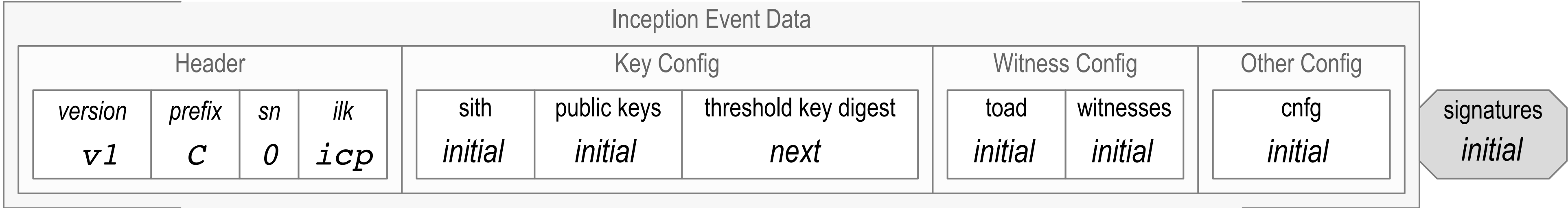
Example Values

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



Generic Event Formats



Generic Inception

$$\boldsymbol{\varepsilon}_0^C = \left\langle \boldsymbol{v}_0^C, \boldsymbol{C}, t_0^C, \mathbf{i}\mathbf{cp}, K_0^C, \widehat{\boldsymbol{C}}_0^C, \eta_0^C \left(\left\langle K_1^C, \widehat{\boldsymbol{C}}_1^C \right\rangle \right), M_0^C, \widehat{W}_0^C, [cnfg] \right\rangle \widehat{\boldsymbol{\sigma}}_0^C$$

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

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

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

$$\widehat{\boldsymbol{\sigma}}_0^C = \boldsymbol{\sigma}_{C^{s_0}} \dots \boldsymbol{\sigma}_{C^{s_{s_0^C-1}}}$$

Generic Rotation

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

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

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

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

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

$$\widehat{\boldsymbol{\sigma}}_{kl}^C = \boldsymbol{\sigma}_{C^{r_l^C + s_0}} \dots \boldsymbol{\sigma}_{C^{r_l^C + s} s_{kl}^{C-1}}$$

Generic Interaction

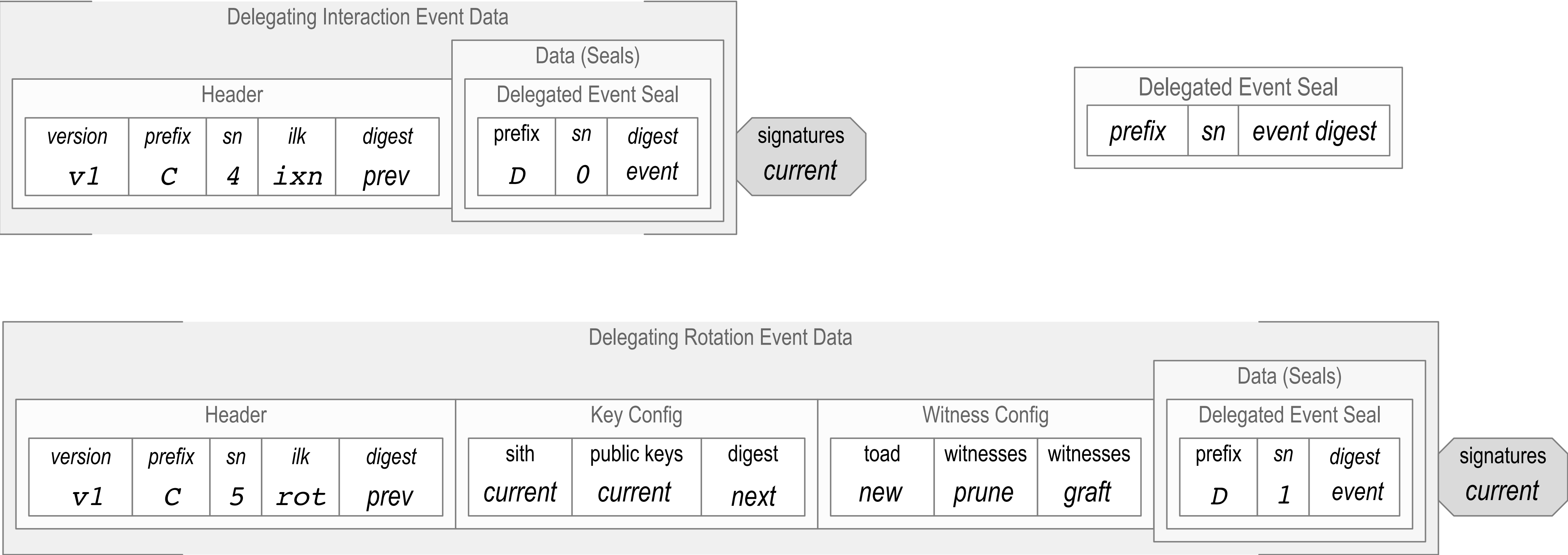
$$\varepsilon_k^C = \left\langle \nu_k^C, C, t_k^C, \eta_k^C \left(\varepsilon_{k-1}^C \right), \texttt{ixn}, [seals] \right\rangle \hat{\sigma}_{kl}^C$$

$$K_l^C$$

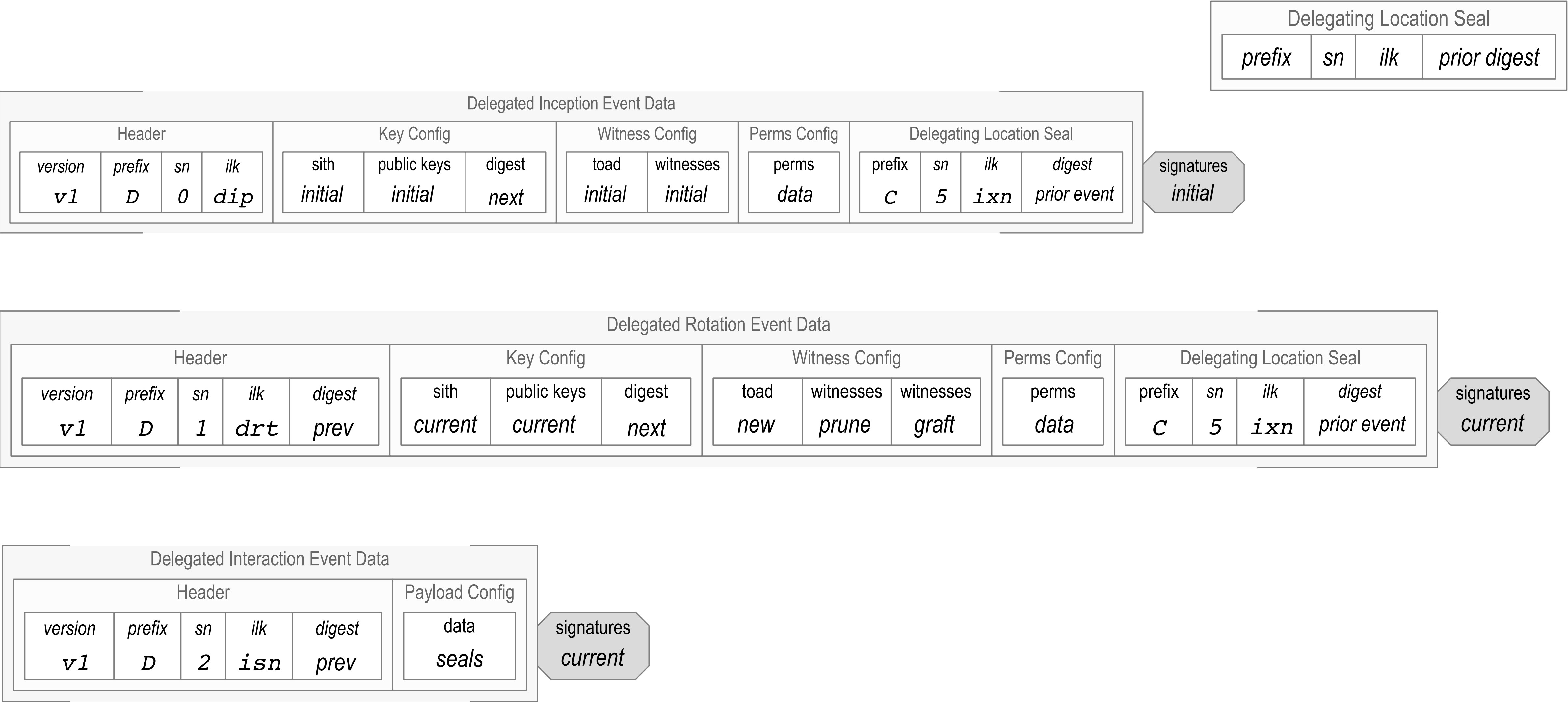
$$\hat{C}_l^C = \left[C^{r_l^C}, \dots, C^{r_l^C + L_l^C - 1} \right]_l^C$$

$$\hat{\sigma}_{kl}^C = \sigma_{C^{r_l^C + s_0}} \dots \sigma_{C^{r_l^C + s_{s_{kl}^C} - 1}}$$

Generic Delegating Event Formats



Generic Delegated Event Formats



Inception Delegation

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

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

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

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

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

$$\widehat{\sigma}_0^D = \sigma_{D^{s_0}} \dots \sigma_{D^{s_{S_0^D-1}}}$$

Rotation Delegation

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

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

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

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

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

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

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

Delegated Interaction

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

Receipt Messages

$$\rho_V^C(\varepsilon_k^C) = \langle \nu_k^C, C, t_k^C, \mathbf{rct}, \eta_k^C(\varepsilon_k^C), V, \sigma_V^C \rangle$$

$$\rho_{W_{li}^C}^C(\varepsilon_k^C) = \langle \nu_k^C, C, t_k^C, \mathbf{rct}, \eta_k^C(\varepsilon_k^C), W_{li}^C, \sigma_{W_{li}^C}^C \rangle$$

$$\rho_{\tilde{W}_{ls}^C}^C(\varepsilon_k^C) = \langle \nu_k^C, C, t_k^C, \mathbf{rct}, \eta_k^C(\varepsilon_k^C), \tilde{W}_{ls}^C, \hat{\sigma}_{\tilde{W}_{ls}^C}^C \rangle$$

$$\tilde{W}_{ls}^C = \left[W_0^C, \dots, W_{N_s^C-1}^C \right]_{ls}^C \quad \hat{\sigma}_{\tilde{W}_{ls}^C}^C = \sigma_{W_{l0}^C}^C, \dots, \sigma_{W_{N_s^C-1}^C}^C$$

$$\rho_{\tilde{W}_{ls}^C}^C(\varepsilon_k^C) = \langle \nu_k^C, C, t_k^C, \mathbf{rct}, \eta_k^C(\varepsilon_k^C) \rangle W_{l0}^C \sigma_{W_{l0}^C}^C, \dots, W_{lN_s^C-1}^C \sigma_{W_{lN_s^C-1}^C}^C$$



Witness Rotations

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

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

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

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

$$M_l \leq N_l$$

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

$$\widehat{U}_{l-1} \subseteq \widehat{W}_{l-1} \quad |\widehat{U}_{l-1}| \geq M_{l-1}$$

$$\widehat{U}_l \subseteq \widehat{W}_l \quad |\widehat{U}_l| \geq M_l$$

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

Complex Weighted Signing Thresholds

$$\widehat{C}_l = [C_l^1, \dots, C_l^{L_l}]_l$$

$$\widehat{K}_l = [U_l^1, \dots, U_l^{L_l}]_l$$

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

$$\widehat{s}_k^l = [s_0, \dots, s_{s_k^l-1}]_k^l$$

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

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

$$U_l^j = 1/K_l$$

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

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

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

BACKGROUND



KERI

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

Derivati on Code	Prefix Description	Data Length Bytes	Pad Length	Count Code Length	Qual Length Base64	Code Length Bytes
-AXX	Count of Attached Qualified Base64 Receipt Couplets	0	0	4	4	3
-BXX	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	Derivat ion 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	Derivat ion 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
AX	Ed25519 signature	64	2	2	88	66
BX	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
0AXX	Ed448 signature	114	0	4	156	117
OBXX						
OCXX						

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
-AXX	Count of Attached Qualified Base64 Signatures	0	0	4	4	3
-BXX	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

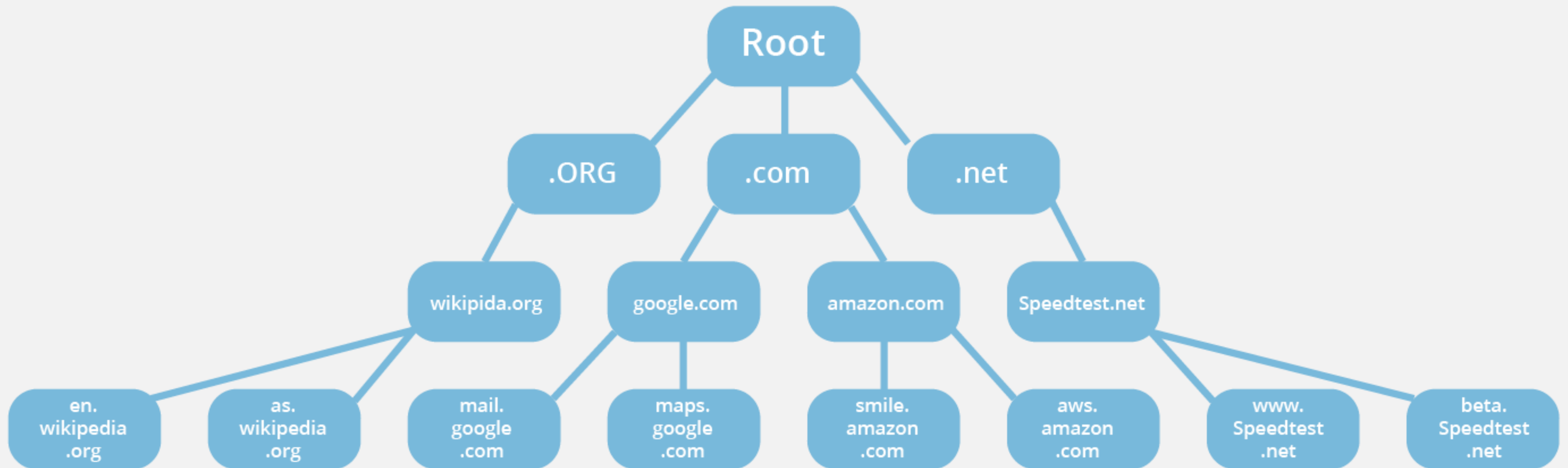
Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary	Base64 Index Decimal	ASCII Char	ASCII Decimal	ASCII Hex	ASCII 8 bit Binary
0	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

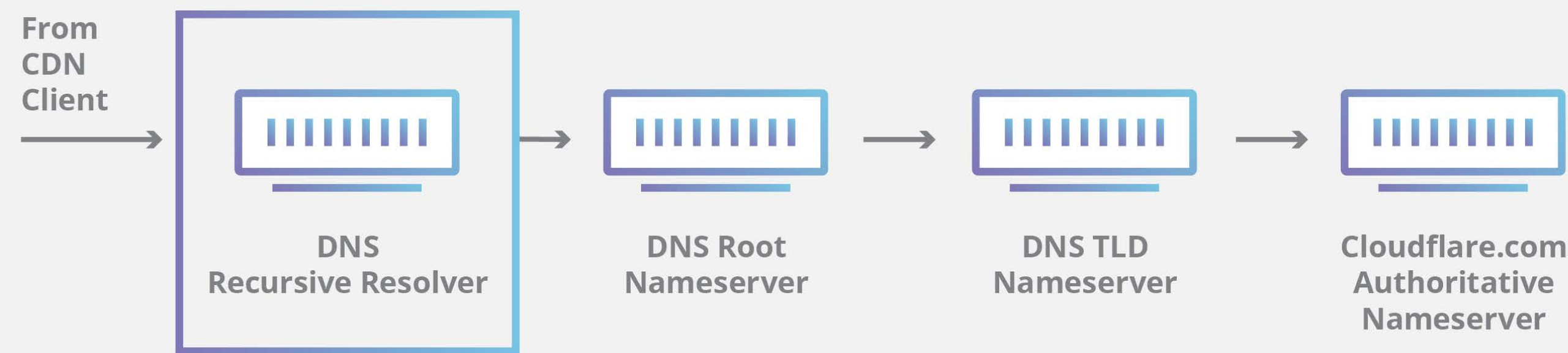
Non-Ledger Based

DNS “Hierarchical” Discovery



DNS “Hierarchical” Discovery

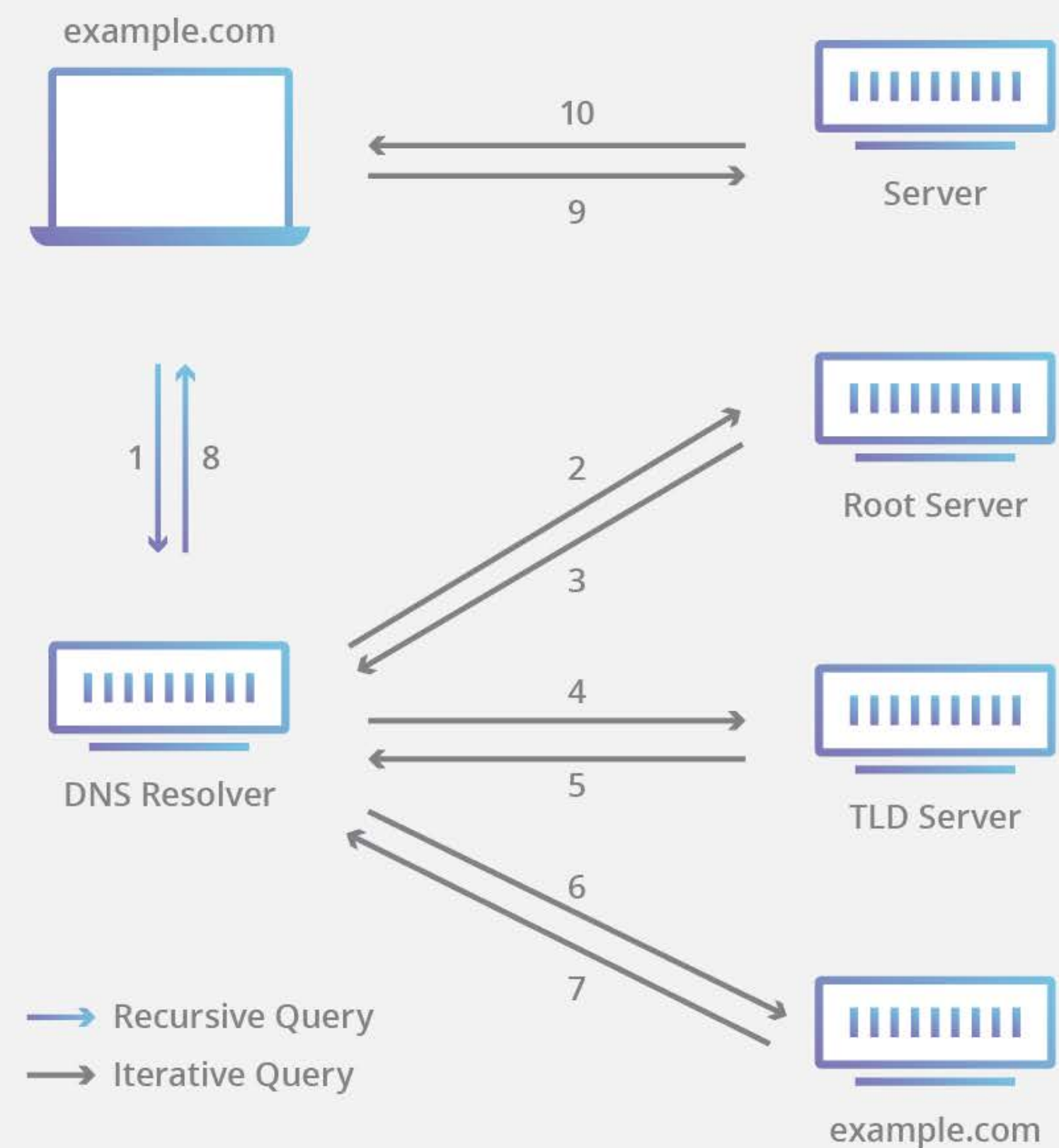
DNS Record Request Sequence



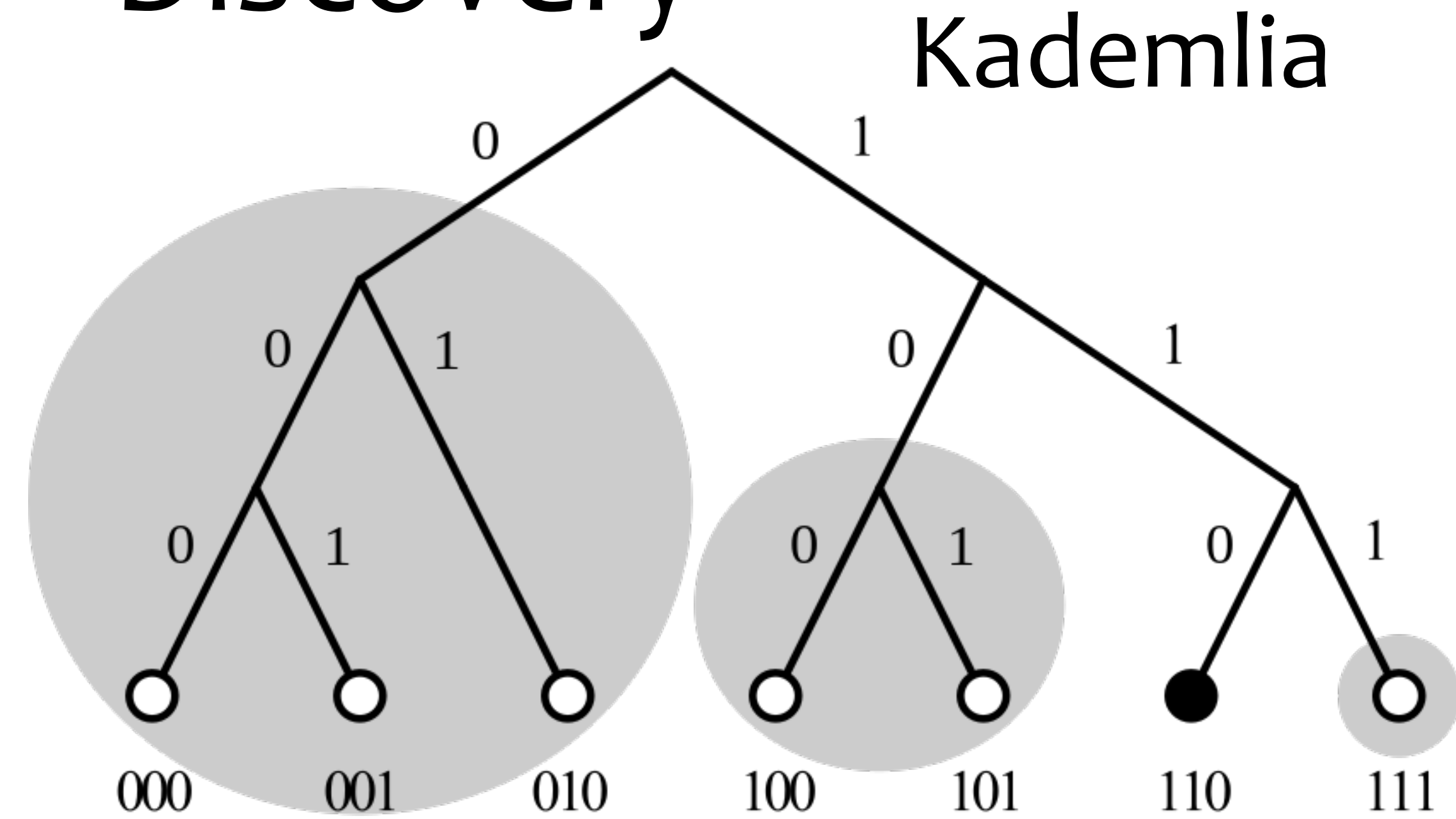
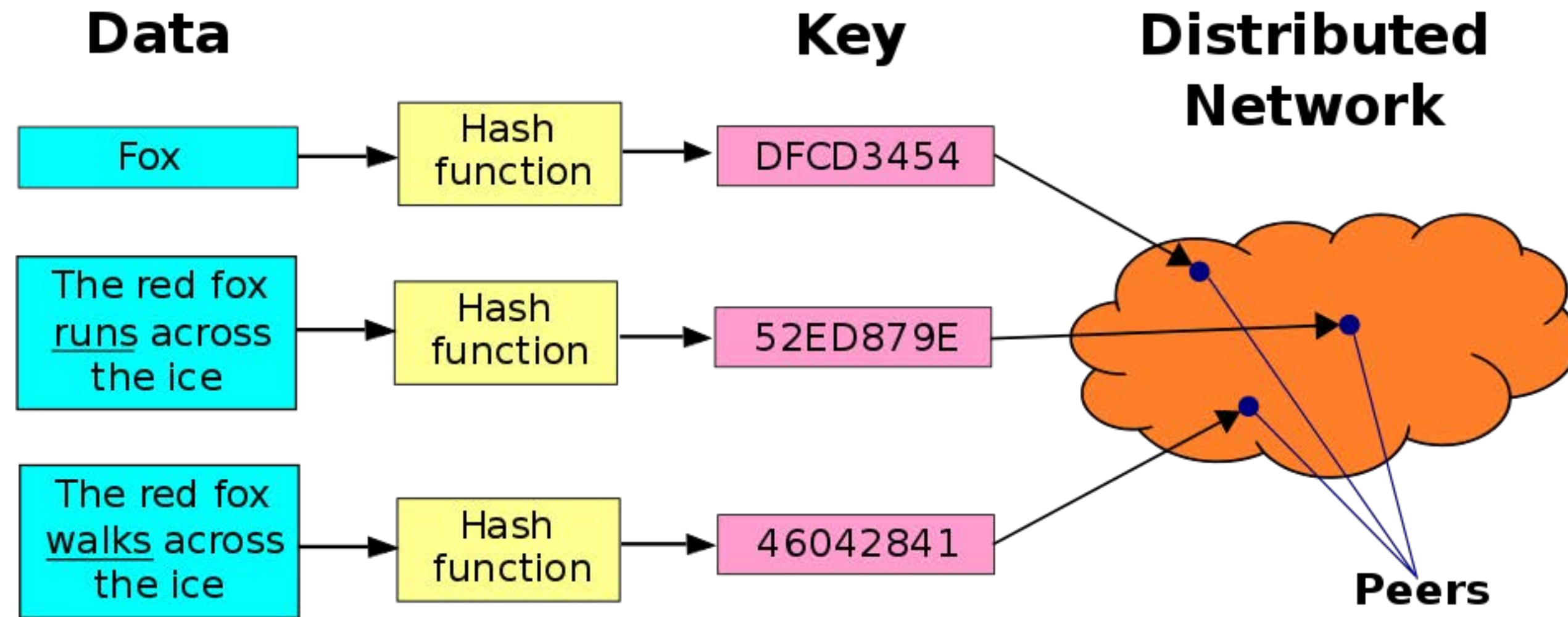
\$ORIGIN example.com.

```
@ 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.
```

Complete DNS Lookup and Webpage Query



DHT “Distributed” Discovery



DHT Discovery for KERI

Resolve Node Prefix to IP Mapping

Prefix to Inception/Latest Rotation Event Caching

-> Extract Witness Prefixes from Event

Witness Prefix to IP Mapping

KERL Query to Witness Node

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)

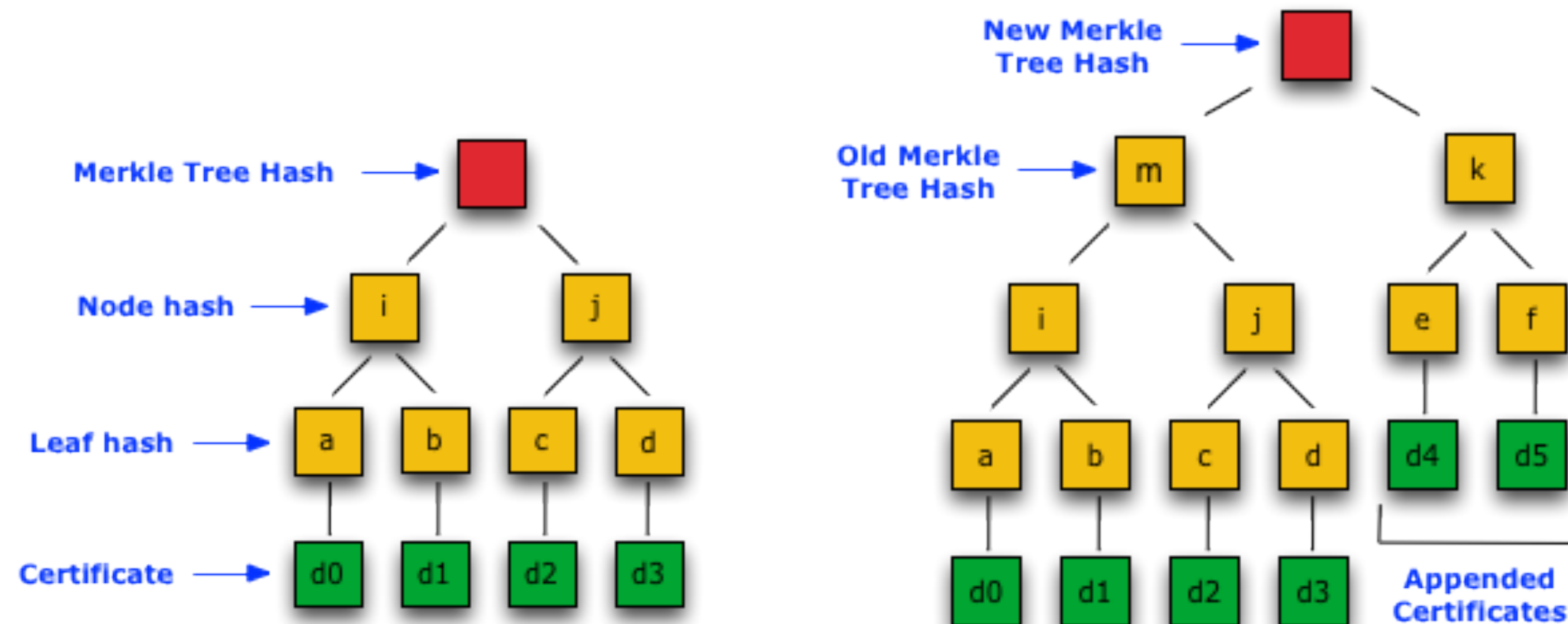


Figure 1

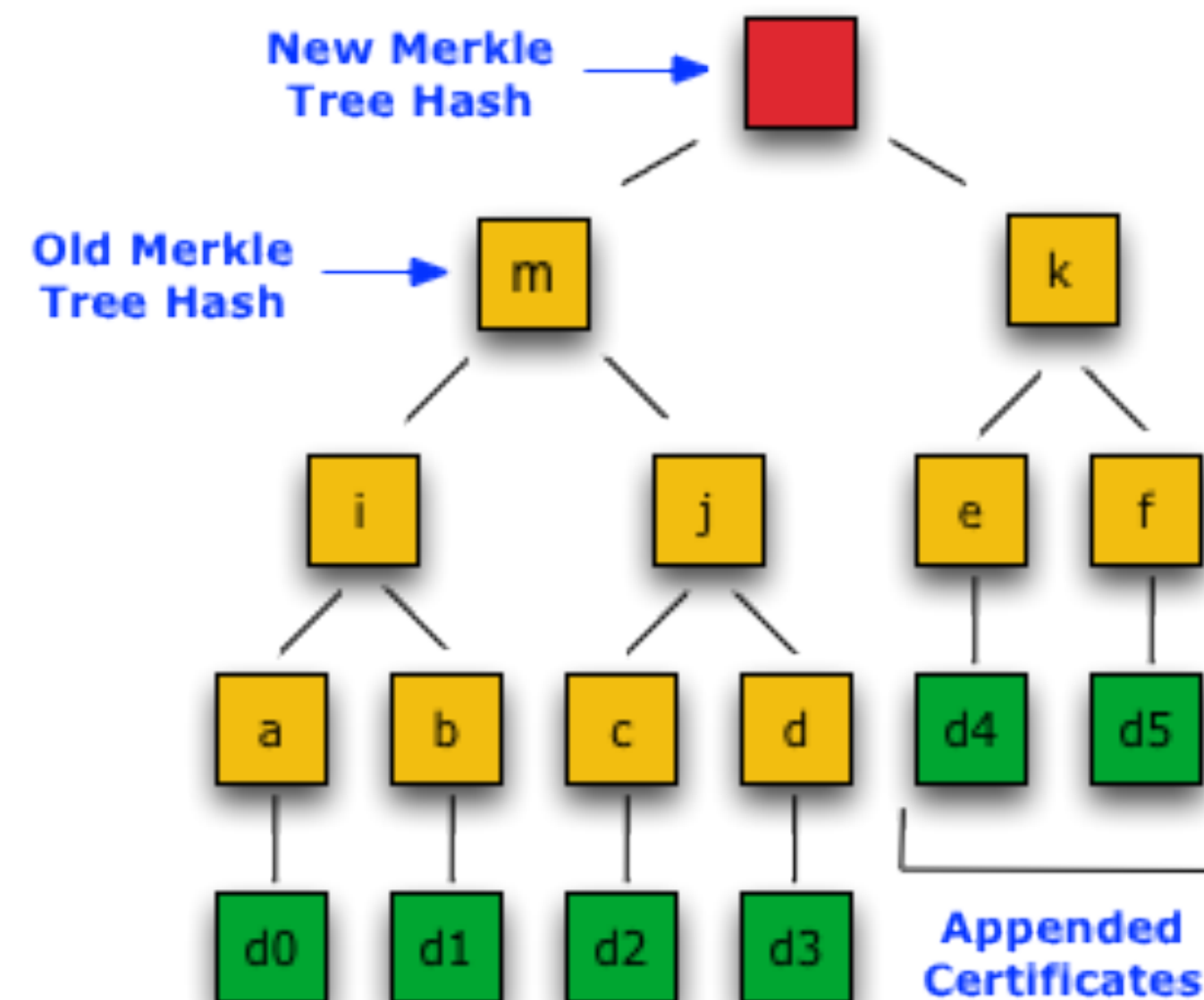


Figure 2

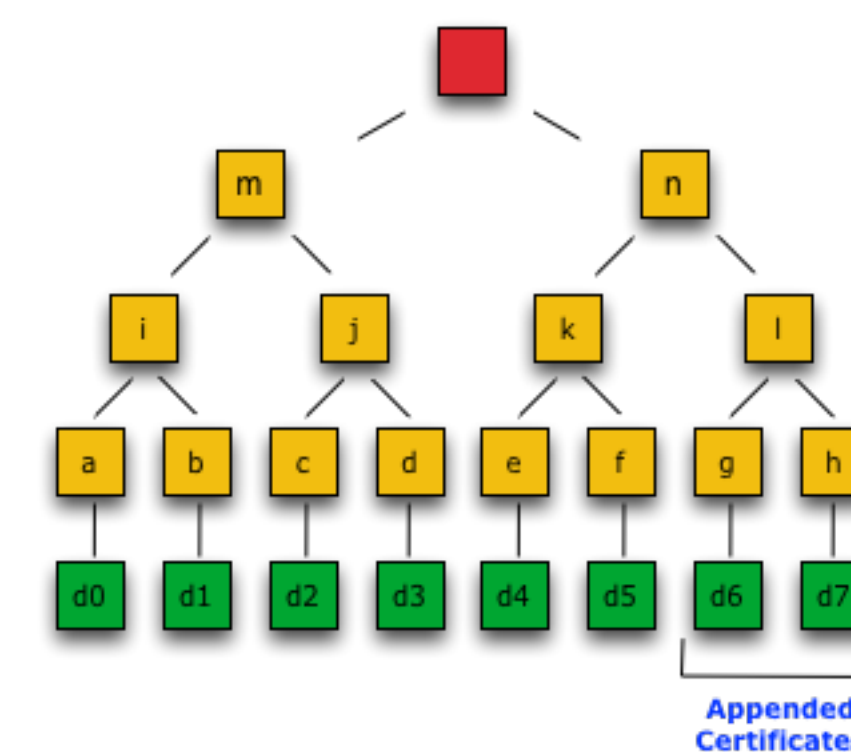


Figure 3

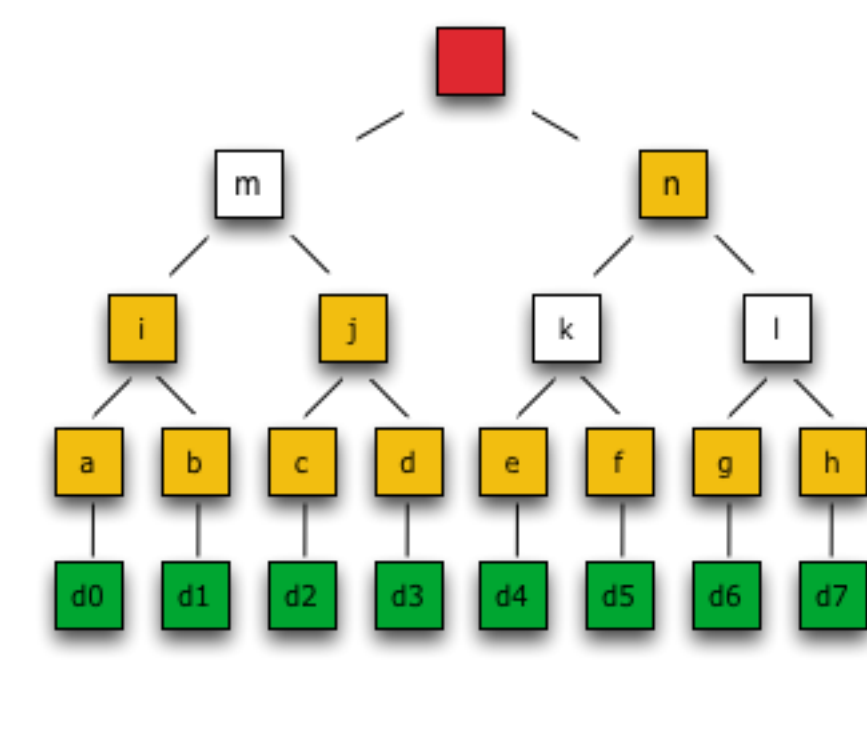


Figure 4