Analysis of did:cel Heartbeat: Relevance to KERI

Daniel Hardman

2025-12-20

Abstract

This paper analyzes the heartbeat mechanism in the did:cel method and its relevance to KERI's approach to retrograde attack defense.

The `did:cel` method specification introduces a "heartbeat" mechanism designed to protect the integrity of the identifier's log [1]. This analysis examines the purpose of the heartbeat feature and evaluates its necessity within the Key Event Receipt Infrastructure (KERI) ecosystem.

The Heartbeat Mechanism in did:cel

In the `did:cel` method, a heartbeat is a periodic, timestamped signature issued by the identifier controller. Its primary function is to mitigate **retrograde attacks** (also known as history rewriting or long-range attacks).

In a retrograde attack, an adversary who compromises a private key used in the past can generate a fraudulent history branching from that point in time [2]. Because the compromised key was valid at the fork point, the new chain appears cryptographically valid.

The heartbeat feature addresses this by requiring regular proof of liveness. It narrows the window for a successful fork to the most recent heartbeat interval. If a verifier encounters two competing histories, the one lacking a continuous stream of valid, recent heartbeats is rejected. This forces the controller to maintain an active presence, verifying that the log is current and reducing the utility of stolen historical keys.

KERI Architecture and Retrograde Defense

KERI employs a different architectural approach to the problem of history rewriting. It relies on **pre-rotation**, **witnesses**, and **duplicity detection** rather than liveness assertions. Consequently, a mandatory heartbeat is redundant within a KERI ecosystem.

Pre-rotation

The primary defense against retrograde attacks in KERI is cryptographic pre-rotation. When a controller establishes a key state (e.g., in an inception or rotation event), they must commit to the *next* rotation key immediately. This is achieved by including a cryptographic digest (hash) of the next key in the current event [3].

The actual public key for the next rotation is not revealed until that rotation occurs. This creates a one-way dependency. Even if an attacker compromises a private key that was active in the past, they cannot use it to fork the history successfully because they cannot satisfy the pre-rotation commitment. The attacker possesses the current signing key but lacks the private key required to generate the public key matching the pre-committed hash. This prevents the seamless extension of a fraudulent chain from a past point of compromise [4].

Duplicity Detection via Witnesses

While `did:cel` uses heartbeats to help a verifier choose the "correct" history among competing forks, KERI treats the existence of any fork as a critical failure of trust. This condition is termed **duplicity** [3].

KERI identifiers replicate their Key Event Logs (KELs) to designated servers called witnesses. Witnesses serve as a check against duplicity; they will not accept a new event if it conflicts with an existing event at the same sequence number.

Verifiers (or their agents, termed watchers) detect duplicity by querying these witnesses. If Witness A reports one version of history and Witness B reports another, duplicity is detected. In KERI, this detection does not trigger a selection process to find the "winner" based on liveness; rather, it provides cryptographic proof that the controller is compromised or dishonest. The identifier is marked as untrustworthy.

Operational Implications: Liveness vs. Security

The `did:cel` heartbeat imposes a liveness requirement: the controller must bring keys online periodically to sign heartbeats. This converts the identity system into a "hot" infrastructure.

KERI separates safety from liveness. It supports "cold" identifiers where keys may remain offline for extended periods. For example, a root authority or a device in long-term storage may not emit events for years. By avoiding a heartbeat requirement, KERI allows the pre-rotated keys to be stored in air-gapped environments or physical safes, reducing the attack surface. Requiring these keys to come online solely to satisfy a heartbeat protocol would undermine this security posture without adding protection against history rewriting, which is already handled by the chain's cryptographic integrity [4].

Conclusion

The `did:cel` heartbeat is a specific mitigation for architectures where past keys can be used to forge alternate histories. KERI neutralizes this threat through pre-rotation, which cryptographically binds the future to the present, and a witness network that detects inconsistencies. Adding a heartbeat mechanism to KERI would introduce unnecessary complexity and operational risk without providing additional security guarantees.

References

[1] Digital Bazaar. 2024. DID CEL Method Specification. Retrieved from https://w3c-ccg.github.io/did-cel-spec/

[2] Hardman, D. 2025. Why Anchored Signatures? Daniel Hardman's Papers. Retrieved from https://dhh1128.github.io/papers/was.html

[3] Smith, S. M. 2024. KERI Specification. Trust Over IP Foundation. Retrieved from https://github.com/trustoverip/kswg-keri-specification

[4] Hardman, D. 2025. A Primer on KERI, ACDCs, and CESR. Daniel Hardman's Papers. Retrieved from https://dhh1128.github.io/papers/keri-primer.html