ANUNA RESEARCH COOPERATIVE · IEEE DAPPS 2026

did:crdt

Coordination-free decentralised identifiers via signed CRDTs

Hugo O'Connor · Claire Barnes

What's a DID?

- A Decentralised Identifier, defined by a W3C spec
- A persistent digital identity controlled by the subject
- Spec defines a flexible interface; many implementations are possible
- Ties in with cryptographic verification for proof of ownership, mutation

```
did:crdt:123 → {
  "id": "did:crdt:123",
  "authentication": [{
    "id": "did:crdt:123#keys-1",
    "type": "JsonWebKey2020",
    "controller": "did:crdt:123",
    "publicKeyJwk": ...
  }]
}
```

What can DID Documents contain?

— Subject identifier (the only mandatory field)

— Controller: DIDs representing any entities that control this DID

— Also known as: other IDs representing the same subject

— Verification methods: associated public keys

— Verification relationships: authentication, assertion, capabilities...

— Services: pointers to other related URIs

What they shouldn't contain: personal information

What can they be used for?

— Persistent IDs for individuals (user accounts, etc.)

— Ephemeral/service-specific IDs for privacy

— Connecting to Verifiable Credentials

— IDs for physical products, components

— IDs for autonomous agents, processes

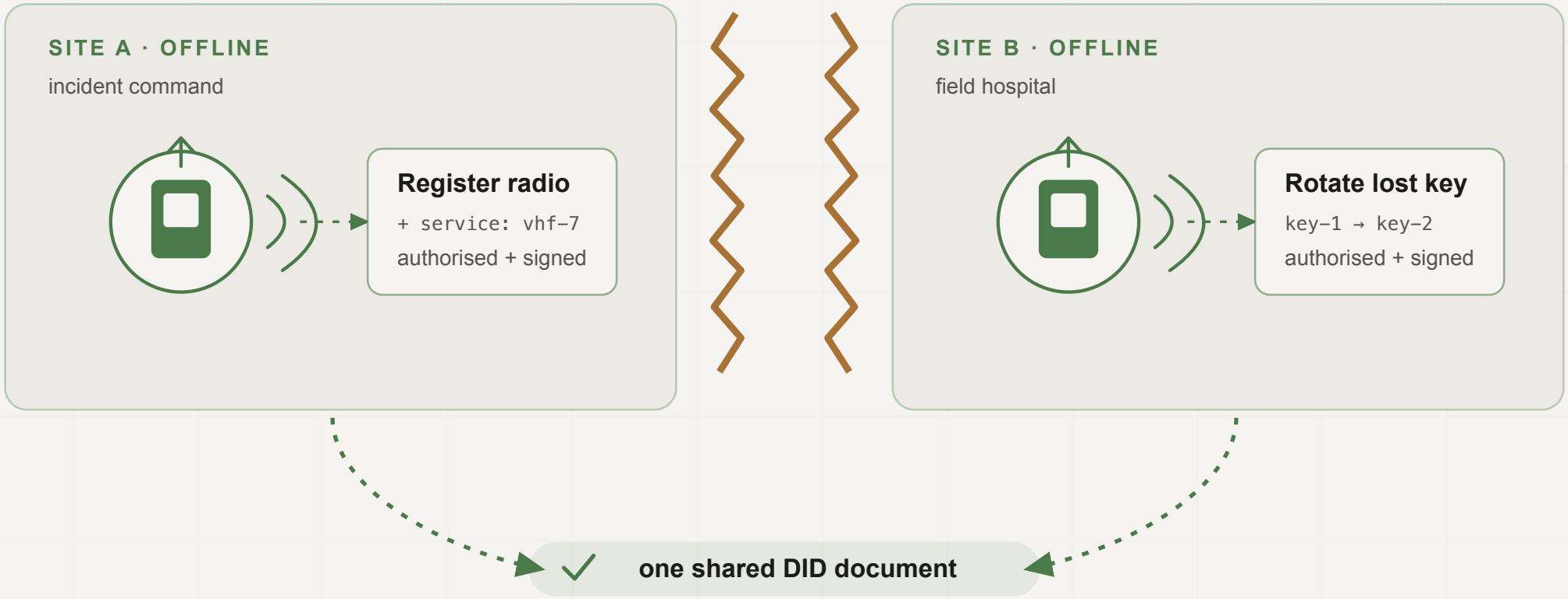
Existing DID types and limitations

APPROACH	HOW UPDATES WORK	LIMITATIONS
`did:key`, `did:peer`	freeze the document	no key rotation, updates
`did:ethr`, `did:sidetree`	ledger / anchored log	needs chain for finality, fees and latency
KERI, `did:webvh`	one totally ordered log	no partitioned updates

None allow updates without coordination

Why do these limitations matter?

MOTIVATING EXAMPLE: NATURAL DISASTER RECOVERY

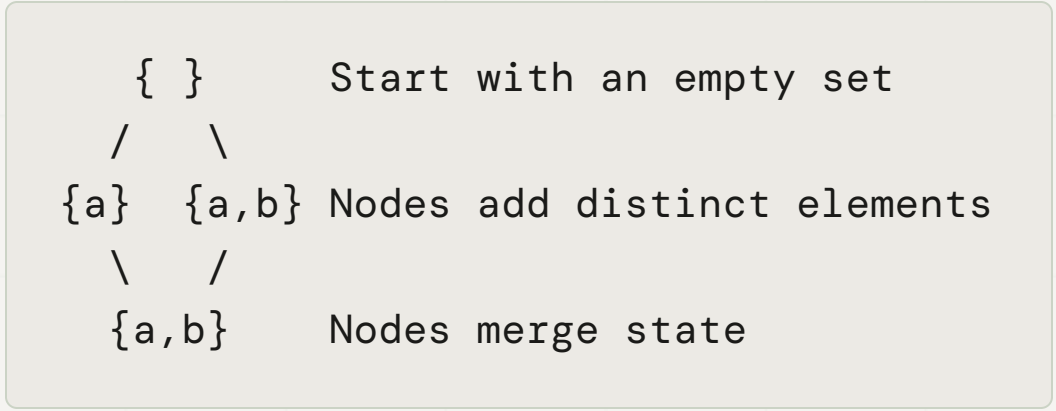


Conflict-free Replicated Data Types

STATE-BASED CRDTS

- Eventual consistency without consensus
- Build our DID document from them and we get convergence for free
- Merge is commutative, associative, idempotent

EXAMPLE: GROW-ONLY SETS



How do we build a DID with CRDTs?

STATE COMPONENT	CRDT	WHY DO WE USE IT?
`verificationMethod`	2P-Set	Add & permanently revoke keys
`service`	ORSWOT	Can re-add removed services
`deactivated`	OR Latch	Deactivation is irrevocable
Other properties	LWW-map	Simple, though not ideal semantics

The product (tuple) of CRDTs is also a CRDT

How do we update over time?

STATE MERGE

— 'Classic' CRDT method

— Convergent merge of values

— Requires shipping full state

— No integrity trail

SIGNED DELTAS

— 'Custom' approach

— Only need to ship deltas

— Signed Merkle DAG, Byzantine-tolerant

— Works well with gossip networks

Kleppmann, Making CRDTs Byzantine Fault Tolerant

The signed delta approach

FIELD	DESCRIPTION
`did`	The DID itself
`τ`	An HLC timestamp
`P`	The hashes of causal parent deltas
`op`	Operation (e.g. `Deactivate`)
`π`	Signature over the rest of the delta

A signed delta is a tuple

`⟨did,τ,P,op,π⟩`

There's a special genesis delta

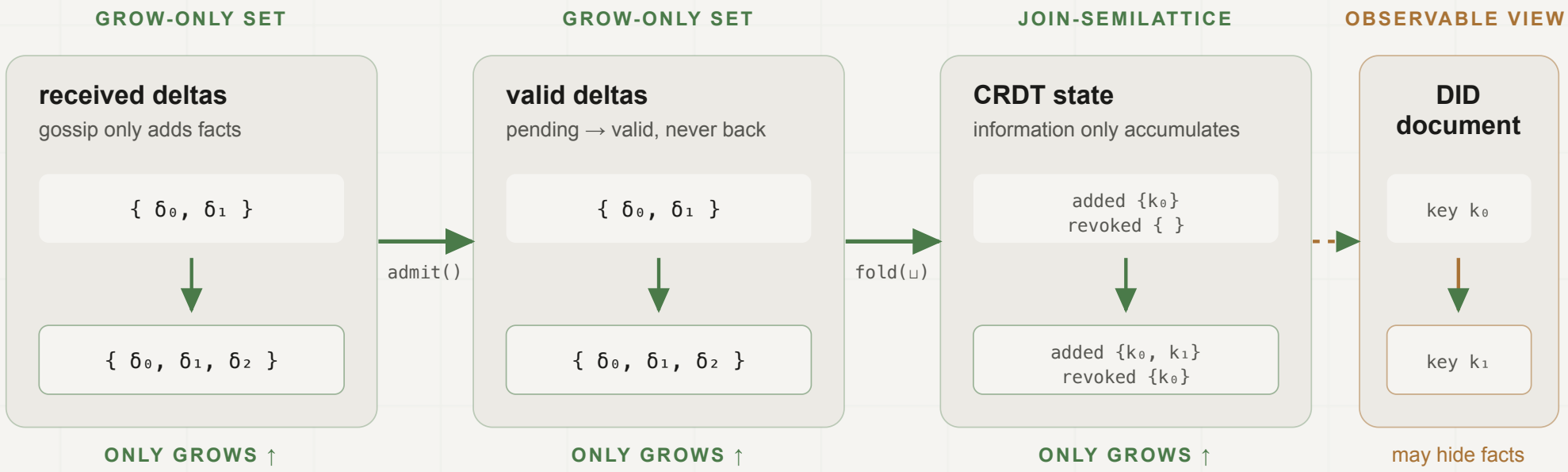
— The DID is derived from a public key

— The first delta adds that key

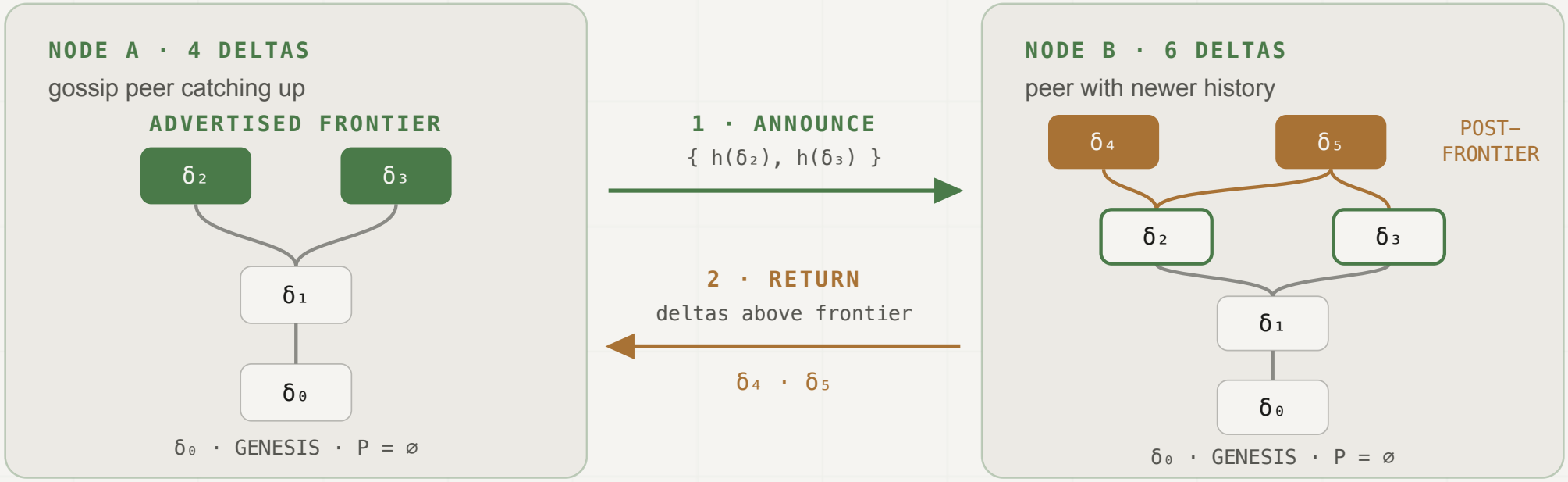
— The HLC is zeroed and P is ∅

CRDTs converge, what about deltas?

We lean on the CALM theorem (Hellerstein & Alvaro)



Anti-entropy over the Merkle DAG



What do we defend against?

With gossip networks as the motivating example, Byzantine fault tolerance is crucial

— Tampering/forgery: all deltas are signed, including genesis

— Replay attacks: deltas are idempotent and commit to their causal history

— Denial of Service: Kleppmann's approach - assume honest nodes can communicate

— Key compromise: containment (revocation), not recovery (future work)

Note that this applies to signed deltas only, state merge is trusted

REFERENCE IMPLEMENTATION

We come with code

https://git.anuna.io/anuna-research/did-crdt

— Rust, WASM-compatible core

— Iroh gossip protocol

— Extensive property-based tests for convergence

WRAPPING UP

Limitations and future work

Key tradeoff: no canonical current value (no total order)

CONCEPTUAL

- Key relationships treated naively
- Sybil resistance
- Checkpoint compaction
- Key compromise recovery
- HLC node ID collisions

PRACTICAL

- Durable persistence
- Discovery (DHT approach, with tradeoffs)
- W3C method specification & registration

QUESTIONS

Thank you

https://git.anuna.io/anuna-research/did-crdt

hugo@anuna.io · claire@anuna.io