Where Trust Bottoms Out: X.509, Certificate Transparency, and KERI's DPKI Architecture

Daniel Hardman

2026-03-05

Abstract

The internet has no native identity layer. To fill that gap, the community built Public Key Infrastructure (PKI) — an administrative model in which Certificate Authorities (CAs) vouch for the binding between cryptographic keys and entities, and Certificate Transparency (CT) logs provide after-the-fact auditability. CT was a genuine improvement in PKI: it made the CA ecosystem more accountable. But it redistributes administrative trust rather than eliminating it, and it runs on cryptographic foundations that are now under an explicit deprecation timeline. The Key Event Receipt Infrastructure (KERI) takes a different approach, shifting the root of trust from administrators to self-certifying identifiers. This paper compares CT with KERI's decentralized witnesses and watchers — their shared commitments, their real differences, and their respective trajectories as the quantum transition moves from planning exercise to engineering deadline.

1. The Identity Problem: Administrative Trust and Its Limits

The internet was built as a network of machines. It addresses endpoints, not the people or organizations that control them. To fill this void, digital architects built Public Key Infrastructure (PKI) — a system that depends on centralized Certificate Authorities (CAs) to vouch for the binding between a cryptographic key and an entity.

The core tension lies in the separation of identifier from key. In the X.509 model, an identifier — a domain name, an email address, an organization name — is a record in a database. The key is a separate artifact. The binding between them is an assertion: a digital certificate signed by a third party. This places the root of trust outside the identity itself. We trust that a key controls an identifier not because of anything inherent in the identifier, but because we trust the administrator who asserted the binding. [1]

Administrators fail. In 2011, the DigiNotar CA was breached by Iranian state actors, who used it to issue fraudulent Google certificates and surveil Iranian dissidents. [2] This was not just a technical incident — it was a demonstration that PKI's administrative trust model carries human costs. The eventual distrust of Symantec's PKI, beginning in 2017, showed the risk isn't only acute compromise; negligent issuance over years can undermine a CA just as thoroughly. [3] TrustCor, removed from major browser trust stores in 2022 following public concerns about its corporate governance and possible intelligence service ties, added another dimension: a CA doesn't need to be hacked to become untrustworthy. [4]

These aren't relics of early infrastructure. In 2024, Google Chrome [5], Mozilla Firefox [6], and

Apple [7] all removed Entrust from their trusted root programs, citing a consistent pattern of compliance failures. [a] Internet measurement data showed Entrust ranked among the top twenty certificate authorities by valid certificate count on the public internet, with hundreds of thousands of active certificates [8], and the browsers' decisions cascaded immediately into emergency migrations across global enterprise infrastructure. The Entrust episode is instructive in two ways. First, CA failures aren't a solved problem — they recur. Second, it makes explicit what has always been true: the ultimate arbiter of CA trustworthiness is a handful of browser vendors. That isn't a flaw so much as a candid description of where administrative trust is actually concentrated.

There's a subtler architectural problem too. X.509 certificates have a mandatory `Validity` field — `NotBefore` and `NotAfter` properties — baked into the standard. [1] Evidence of identity expires by design, not by necessity. The distinction matters. Dynamic privilege tokens — a business license, a domain registration — should expire, because the privilege they confer is contingent. Identity is different. It means *sameness*: the thing that holds constant across contexts. An organization incorporated in 1975 doesn't renew its identity every ninety days. But using a certificate to prove that identity forces exactly that: evidence must turn over, regardless of whether anything has changed.

This evidence churn isn't just a philosophical complaint. It creates ongoing maintenance costs and, as research on revocation behavior has consistently shown, a predictable pressure to defer revocation when a key is compromised. This is because immediate revocation triggers costly emergency reissuance cycles that natural expiration avoids. [9] The community's answer — shorter certificate lifespans — makes things worse. Shrinking to 90-day certificates multiplies the maintenance burden roughly fourfold compared to annual certs, yet still leaves a window of compromised operation on every incident. The tools that manage this churn — certificate lifecycle platforms, revocation services, OCSP responders — represent a recurring operational tax on a design decision baked into the standard forty years ago. And the churn proves less than it appears: certificate renewal demonstrates that a CA is willing to re-assert a binding, not that the underlying key was rotated, that the old key was retired, or that any meaningful change in key management occurred. The holder may reuse the same keypair across CAs, and across contexts — and nothing in the protocol makes this visible.

---

2. Certificate Transparency: Progress and Its Limits

Certificate Transparency was a genuine response to genuine failures. Before CT, a CA could issue a certificate for any domain with no obligation to make that issuance visible to anyone. Domain owners had no reliable way to discover fraudulent certificates issued in their name. CT changed this: CAs must now submit all newly issued certificates to public, append-only, cryptographically verifiable logs. [10] When a log accepts a certificate, it returns a Signed Certificate Timestamp (SCT) — a cryptographic promise to integrate the certificate into a Merkle tree within a specified Maximum Merge Delay (MMD). Modern browsers require multiple SCTs before trusting a certificate. The result is a meaningfully more accountable CA ecosystem: undetected mis-issuance is harder, and the public issuance record has let domain owners and researchers spot anomalies that would once have gone unnoticed.

These are real strengths. CT matters, and this paper would be dishonest to minimize it.

But CT has structural limits worth examining.

Detection, not prevention. CT doesn't stop a fraudulent certificate from being issued and used. It guarantees eventual discoverability. The MMD creates a latency window — typically 24 hours — during which a mis-issued certificate can be actively exploited before it appears in any log. Split-view attacks, where a malicious log operator shows different states to different clients, can extend that window indefinitely. [11] The main proposed defense is a gossip protocol, where clients cross-verify log states with third-party auditors. But clients checking certificate inclusion by querying auditors directly leaks browsing behavior to those auditors, a tension that has resisted easy resolution. [12] This may be why gossip solutions have seen no wide deployments and the 2018 draft RFC about them has been abandoned. CT is a detection system with a latency window and a privacy price tag on its most important mitigation.

Administrative trust redistributed, not eliminated. This is CT's most underappreciated limit. CT doesn't remove administrative trust — it adds layers to it. The original CA trust problem remains. On top of it, CT adds log operators who must be approved and kept trustworthy; their infrastructure must stay available and consistent. The third-party monitors that are supposed to police log behavior vary widely in practice: research has found significant gaps in how actively they inspect logs and how reliably they detect anomalies. [13] On top of that, browser root store programs — run by Google, Mozilla, and Apple — decide which CT logs are trusted. [14] These programs wield immediate, global effect over certificate validity. Above all of this sits the CA/Browser Forum, a private governance body that sets the Baseline Requirements CAs must follow. Its standards have been contested and unevenly enforced, as the Entrust episode itself illustrated. The result is not a decentralized trust hierarchy. It is a more elaborate one, with concentrated administrative authority at several points. [b] Research has documented how CT deployment has produced measurable centralization in the certificate ecosystem, with a small number of operators — predominantly US-based — controlling the infrastructure on which global auditability depends. [15, 16]

Revocation's persistent failure. CT addresses mis-issuance visibility. It doesn't address revocation. When a private key is compromised, the certificate must be revoked. The tools for this — Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP) — are bolt-ons with well-documented problems. CRLs grow as revocations accumulate and scale poorly. [17] OCSP requires querying a central responder for every validation, creating both a privacy leak and a reliability bottleneck. When the responder is offline, browsers typically "fail open" — they accept the unverifiable certificate rather than block traffic. [9] The system is most vulnerable precisely when it is most needed: at the moment of key compromise, its security depends on the availability of a central service that, if absent, defaults to trusting the potentially bad credential.

The quantum problem. CT's logs, SCTs, and inclusion proofs all depend on RSA and elliptic curve cryptography — signature schemes now on an explicit deprecation timeline. In May 2025, Google researchers demonstrated that 2048-bit RSA could theoretically be broken by a quantum computer with one million noisy qubits in a week — a 20-fold reduction in estimated resources compared to their 2019 work. [18] Google followed this in March 2026 with an ambitious 2029 deadline for PQC migration. [19] NIST's transition guidance (an initial public draft pending finalization) calls for deprecating RSA and ECC by 2030 and disallowing them by 2035. [20] Post-quantum signature algorithms produce signatures far larger than classical ones. Integrating them into CT's infrastructure — logs, handshakes, SCTs, inclusion proofs — creates serious performance challenges. The Merkle Tree Certificate proposal (MTC) exists specifically to ad-

dress this, amortizing heavy cryptography across many validations to keep on-the-wire costs manageable. [21] But MTC is an active IETF working group draft, not deployed infrastructure. The CT ecosystem that exists today is not the ecosystem that will need to exist by the mid-2030s.

---

3. Common Ground: What CT and KERI Share

Before the comparison sharpens, it helps to see what these two systems have in common. Their communities sometimes speak as if they're solving entirely different problems. At the level of fundamental mechanism, they are closer than is often acknowledged.

Both CT and KERI rely on **append-only, hash-chained data structures** as their core integrity primitive. CT organizes certificate issuance events into Merkle trees whose roots are periodically published. KERI maintains per-identifier Key Event Logs (KELs) whose events are cryptographically linked by hash to their predecessors. In both systems, tampering with history is detectable because hash chains break.

Both use **cryptographically signed receipts** as the unit of trust propagation. A CT log's SCT is a promise, signed by the log operator, that a certificate has been logged. A KERI witness receipt is a signed acknowledgment that a key event has been received and validated. Neither system reduces security to a single signature from a central authority; both require agreement across multiple independent signers before a record is considered stable.

Both are fundamentally **detection-oriented**. Neither prevents a bad actor. CT doesn't stop a CA from issuing a fraudulent certificate; it makes the fraud visible to monitors. KERI doesn't stop a malicious controller from forking an identity history; it makes the fork detectable by watchers. This is a principled design posture — it reflects the well-established distributed systems insight that guaranteeing prevention requires global consensus at a cost in latency and scalability that both systems deliberately decline to pay. [22] It also has an independent pedigree in systems engineering: Carnegie Mellon's Software Engineering Institute, in foundational work on survivable systems, established that no system can guarantee resistance to all intrusions, and that detection and recovery are therefore first-class design requirements, not fallbacks. [23]

Both face **post-quantum pressure** and both reach for hash-based commitments as long-lived anchors. Hash functions resist quantum attack far better than asymmetric signatures: Grover's algorithm provides only a quadratic speedup against hashes, not the exponential speedup Shor's algorithm provides against RSA and ECC. [24] NIST's post-quantum cryptography evaluation criteria treat a 256-bit hash as providing security comparable to AES-128 key search — approximately 128 bits of post-quantum resistance. [25] CT's Merkle tree structure is built on hashes. KERI's pre-rotation mechanism hides the next controlling key behind a hash. Both systems were building quantum-resistant scaffolding into their designs from the start.

Both also **shift some security burden to an observer layer** that must be actively operated. CT's monitors and auditors must poll logs, compare states, and propagate gossip to close the split-view window. KERI's watchers must poll witnesses, compare log heads, and broadcast evidence of divergence. In both cases, the protocol provides the mechanism for detection; the deployment provides the actual detection.

Where they diverge is not in mechanism but in the *root of trust*. CT's chain of assurance bottoms out in administrators: CA operators, log operators, browser root programs. KERI's bottoms out

in mathematics: the self-certifying identifier derives from the initial public key, and the KEL's validity can be checked by anyone who replays its hash chain, without querying any authority. That difference has architectural consequences that compound over time.

Dimension	PKI / CT	KERI
Root of trust	Administrative (CAs, log operators, browser root programs)	Cryptographic (self-certifying identifier)
Identity continuity across key rotation	Opaque (CA re-asserts continuity via ACME; no cryptographic link between certificates)	Cryptographic (KEL hash chain; identifier persists across rotations)
Revocation	CRL / OCSP (bandwidth-intensive, fail-open risk, privacy leak)	Key state events in KEL (continuously current; no separate revocation layer)
Fraud response	Detection via CT monitors, post-issuance, with MMD latency window	Detection via watchers; duplicity is cryptographically provable
Quantum posture	Migration required across all CAs, logs, certificates, and TLS stacks; ecosystem coordination constrained by lowest-common-denominator devices (firmware, HSMs, embedded systems)	Architectural: pre-rotation hides next key behind a hash; crypto-agility allows algorithm upgrades without changing the identifier
Evidence churn	Mandatory (RFC 5280 Validity fields; evidence expires by design)	Event-driven (evidence is superseded only when facts change)
Governance transparency	Opaque (holder key management unverifiable; single-key control is standard)	Explicit (weighted multisig thresholds published in KEL; all governance decisions are on the record)
Deployment maturity	Globally deployed (billions of certificates); the same scale that defines its strength defines its quantum migration cost	Early standardization and production activity (ISO 17442-3, GSMA Open Verifiable Calling); no migration debt

4. KERI's Architecture: A Different Root of Trust

KERI's foundational move is to make the identifier self-certifying. Rather than leasing an identifier from a registrar and binding it to a key via a CA's assertion, a KERI controller derives their identifier mathematically from their initial public key. The result is an Autonomic Identifier (AID): an identifier whose authenticity can be verified without any external authority. [26] This

does not remove all reliance on administrative trust, but it shifts it to attributes; identity is purely cryptographic. That distinction has huge repercussions for governance, cost, and maintenance and recovery procedures.

The history of an AID is maintained in its Key Event Log (KEL) — an append-only chain of signed events, each linked by hash to its predecessor. The KEL supports three event types: inception (the identifier's creation), rotation (updating controlling keys), and interaction (anchoring data to the log without changing keys). To check the current state of an AID, a verifier doesn't query a directory. They obtain the KEL and replay it. The head of the log *is* the current state, and anyone can verify it offline, at any time. [26]

Pre-rotation is KERI's most important security innovation. The idea is simple. When a controller creates any event, they include the public key for the *current* controlling key — but for the *next* key, they include only a cryptographic hash of its public key. The next key itself stays offline, in cold storage. An attacker who steals the current key can sign events, but cannot rotate to take permanent control: they don't know the next key's value, only its hash, and inverting a modern hash function is not feasible. [26, 24] When the controller needs to rotate, they reveal the next key's public value. Verifiers confirm it matches the committed hash. Trust moves forward one step, and a new pre-rotation commitment is made for the step after that.

Think of it like a sealed succession document held in a vault. The document commits the heir's identity without revealing it. If the reigning key is stolen, the attacker can act in the present, but cannot seize the future — because the succession is sealed, in writing, and they can't forge it.

Pre-rotation has direct quantum implications. A quantum adversary who breaks the current signing key via Shor's algorithm gains the ability to forge interaction events — but not to rotate, because the pre-rotation commitment is protected by a hash which retains its strength against quantum attack. [24, 25] When the legitimate controller detects the compromise through their own watcher, they can execute a recovery rotation using the pre-committed key, issue a new pre-rotation commitment in a post-quantum algorithm, and keep going. The identifier persists. The history persists. The transition is cryptographic, not administrative.

because the pre-rotation commitment is protected by a hash, which retains its strength against quantum attack (§3). [24, 25]

The interaction-event forgery window is real and should not be minimized. But KERI provides mechanisms to shrink it that have no X.509 equivalent. High-stakes operations — credential issuance, for example — are anchored to the KEL, making them visible to witnesses and watchers in near-real-time. Controllers are encouraged to watch their own KELs, and KERI's libraries enforce consistency checks against witness state automatically. An unauthorized anchored event is detectable almost immediately. For operations that don't warrant anchoring, weighted multisig raises the compromise bar further: forging an interaction event for a multisig AID requires compromising multiple keys across multiple custodians. X.509, by contrast, always binds a certificate to a single keypair, and provides no mechanism for the key holder to detect unauthorized use of their own key through protocol machinery. The forgery window in X.509 is not just open — it is invisible to the party most affected.

KERI also brings **governance transparency** through weighted multisig. A PKI certificate is guarded by a single, opaquely managed secret. The key management practices of the certificate holder — the actual locus of risk — are invisible to anyone relying on the certificate. KERI

makes governance explicit: if an identifier is controlled by multiple keys, the threshold policy is published in the KEL as a verifiable commitment. [26] Every signing event, every rotation, every policy change requires publicly verifiable cryptographic proof that the declared threshold was satisfied. This doesn't just improve security for individual identifiers — it makes security postures *comparable* across organizations, which matters for supply chain trust, regulatory compliance, and risk assessment.

Witnesses provide availability and consistency without a blockchain. A witness is a lightweight server the controller designates to store and serve the KEL. When the controller creates an event, they send it to their witnesses; the witnesses validate and return a signed receipt. Once a threshold of receipts is collected, the event is stable. Witnesses make no assertion about identity — they don't claim "this key belongs to this entity." They store and serve events. [26] That is a fundamentally different role than a CA.

To maintain consistency among witnesses, KERI uses KAWA (KERI's Algorithm for Witness Agreement) — a single-phase agreement protocol that provides fault tolerance without the multi-phase commit overhead of classical Byzantine Fault Tolerant algorithms like PBFT. [27, 28, 29] For honest controllers under proper threshold configuration, KAWA provides three properties: *validity* (accepted events were genuinely authorized), *consistency* (no two correct witnesses accept conflicting events at the same sequence number), and *reliability* (the controller can make progress as long as witness faults stay below the threshold). [27]

Watchers are the detection layer. These are entities that check multiple witnesses for the current head of each KEL they monitor. A centralized mindset imagines them as cloud services that poll, and this can be done. However, a more common embodiment is as an in-proc buffer between untrusted external parties and code that consumes key state in a local software stack. If a watcher asks two witnesses for the head of the same identifier and gets different hash values, it has detected *duplicity*: evidence that the controller signed two conflicting events at the same sequence number. The watcher can report both events as cryptographic proof of fraud. That proof doesn't require trusting any authority's interpretation — only the mathematics of the signatures. [26]

The structural distinction between watchers and CT's administrative monitors is worth making explicit. A browser vendor that distrusts a CA is making an enforcement decision — exercising judgment with immediate global consequences, no appeals process, and no external adjudicator. A KERI watcher that detects duplicity is making an observation — surfacing cryptographic evidence that the relying party confirms independently. Watchers cannot unilaterally "distrust" an identifier the way a browser root program can distrust a CA. The worst a dishonest watcher can do is stay silent about duplicity it has seen — a risk addressed by running your own watcher or consulting multiple independent ones.

This is why KERI's security posture is best described as *duplicity-evident* rather than *duplicity-resistant*. It doesn't try to make duplicity impossible — that would require global consensus, with all the costs that entails. It makes duplicity *provable and attributable*, which is sufficient for a system built around reputation, accountability, and recovery rather than prevention. [22]

---

5. Evaluating KERI: Real Tradeoffs, Honestly Stated

A fair analysis has to engage honestly with KERI's challenges. Some tradeoffs are explicit by design; others are genuine tensions the ecosystem hasn't evolved enough to fully resolve.

The scope of formal guarantees. KAWA reflects well-understood principles from the consensus algorithm literature and the CAP theorem. [22, 28, 29] The most rigorous formal analysis specific to KAWA is a 2025 Master's thesis by Del Giudice at ETH Zurich [27] — worth noting because it is not peer-reviewed research, and because KERI's formal literature remains thin. That thinness reflects the protocol's youth, not necessarily structural flaws. Del Giudice's proofs confirm that KAWA's guarantees hold when the controller is honest, witness faults are bounded below the threshold, and communication is reliable. These are the standard conditions under which any threshold-based agreement protocol provides its guarantees, and KERI's specification acknowledges them directly. [26] The distributed systems literature has long established the underlying constraint: you cannot simultaneously guarantee global consistency, full availability, and partition tolerance. [22] KERI explicitly chooses availability and partition tolerance, and handles the consistency gaps through detection.

Malicious controllers. KERI doesn't prevent a malicious controller from trying to fork their identity history. This is explicit in the specification, and it is the right tradeoff given KERI's scale and availability goals. The watcher layer is the designed response. The question isn't "can duplicity occur?" — it can — but "how quickly is it detected, and what happens when it is?" That answer depends on the density and maturity of watchers. It is thus a deployment question, not a protocol question.

Recovery rotation. KERI's designed recovery path is cryptographic, not administrative. At inception, the controller pre-commits the next operational key. If an attacker seizes the current keys, the legitimate controller executes a recovery rotation to pass control to those pre-committed keys. Validators verify the recovery event against the pre-committed hashes, exactly as they verify any rotation. There is no ambiguity about who controls the identifier afterward: the pre-commitment settles it mathematically. [26] The attacker's forked events remain in the record as permanent, cryptographically attributable evidence of mischief — but the fork does not create confusion about control, because only the recovery path matches the commitment made at inception.

KERI shares a second, messier kind of recovery scenario with X509 and all key-based architectures: catastrophic loss, where the controller loses all control of secrets. In KERI, this requires two independent compromises (current keys *and* pre-committed rotation keys). In such a scenario, no cryptographic mechanism can restore control. The identifier must be abandoned, credentials linked to it revoked, and a new identifier established through administrative processes. This is genuine operational pain — but it is also the *only* recovery mode X.509 supports for *any* key compromise, not just catastrophic ones. In PKI, every compromise is resolved administratively: revoke the certificate, petition a CA for a new one, re-establish bindings from scratch. KERI's architectural contribution is that the common case — compromise of operational keys with recovery keys intact — resolves cryptographically, transparently, and without any administrator's involvement.

Witness and watcher discipline. KERI's security depends on verifiers checking witness state before trusting any KEL — comparing log heads across multiple witnesses to detect divergence. Thus, discipline with witnesses and watchers matters.

A common misconception is that this requires a network of large, shared watcher services analogous to CT's monitors. It does not. A watcher can be (and by default usually is) as simple as an in-process component in a verifier's own software stack that queries multiple witnesses directly before reporting any result. In this model, the verifier trusts its own code to compare what witnesses say — not a remote third party. KERI's architects explicitly recommend this local-resolver pattern. A "super watcher" — a shared archival service providing broad coverage in time and space — is another possible deployment pattern. Depending only on superwatchers is undesirable, as it encourages centralization. However, having some widely deployed superwatchers immunizes any verifier who consults one against eclipse and dead attacks, and is thus a desirable security enhancer.

A variant on the watcher concern focuses on witness availability. Not all witnesses need to be up all the time, but enough witnesses need to be available, for any given AID, to satisfy the consensus requirements of watchers that track it. Whether controllers of AIDs are selecting reliable and well distributed witnesses is thus important, and careless choices could produce suboptimal results.

Two things soften this concern. First, CT faces the same structural dependency. CT's defense against split-view attacks also relies on a distributed network of monitors and auditors. That network has received serious investment because CT has been deployed at scale for a decade — but the structural parallel is real. KERI's witness-watcher fabric is newer, not fundamentally different. Second, KERI's early standardization and production activity — ISO 17442-3 for verifiable Legal Entity Identifiers [30], and the GSMA Open Verifiable Calling project, which launched formally in October 2025 with participation from twenty companies across the telecoms ecosystem [31] — demonstrate that the governance and operational frameworks KERI requires are developing in practice, not waiting on theory.

Both CT and KERI ultimately require humans to take security seriously, invest in infrastructure, and respond to detected anomalies. Neither protocol can force that. What both do — each in its own way — is make the evidence of failure cryptographically clear, attributable, and durable. No trust architecture escapes the burdens of monitoring and accountability.

---

6. The Transition Problem: A Comparison of Trajectories

Comparing "deployed CT today" against "deployed KERI today" misses the point. Those aren't the relevant alternatives. PKI and CT are on the eve of a mandatory, global, and expensive architectural transition. KERI, being new, has no such transition to execute. The honest comparison is of *trajectories*: what does it cost to get each system to post-quantum security, and where does that cost fall?

For PKI and CT, the answer is sobering. The quantum threat to signature schemes — the exact mechanisms on which CA roots, certificates, and SCTs depend — is concrete rather than speculative, and its timeline is tightening. Google's 2029 migration deadline [19] is a market reaction, and NIST's call to deprecate RSA and ECC by 2030 and disallow them after 2035 [20] are regulatory deadlines. Yet neither describes when the engineering will actually be done, or when legacy systems will comply.

That's because the migration PKI faces is not a matter of updating a library. Every CA root certificate, every intermediate CA, every issued certificate, every TLS implementation in every

browser, server, IoT device, VPN, firmware stack, and hardware security module will need updating. Signature keys are harder to replace than encryption keys precisely because they are used in many places and tend to be long-lived [18] — and CA root certificates are among the longest-lived signature artifacts in existence, embedded in hardware and operating system trust stores that are updated infrequently or not at all. MTC offers a path to making the CT portion of this migration less painful [21], but MTC itself must be standardized, implemented, and deployed across the same global infrastructure. The long tail of systems running classical PKI will stretch well past any regulatory deadline. Each one is a gap in the dam.

KERI's position is structurally different. Pre-rotation protects the next controlling key behind a hash from the moment of inception. Early adopters are building on post-quantum-capable foundations today. When signing algorithms need upgrading, KERI controllers can do so without changing their identifier, without re-issuing credentials to relying parties, and without coordinating with any CA or root program. The upgrade is a unilateral event in the KEL.

This simple posture is reinforced by the fact that the KERI ecosystem has nothing to unwind. There is no installed base of classically-keyed infrastructure that must migrate before the system can be trusted. No CA hierarchy whose roots need replacement on a global schedule. No legacy behavior to remain compatible with while simultaneously moving to new primitives. By design, early KERI adopters are not accumulating technical debt — they are building on foundations designed for the world that is coming, not the one that existed in 1988.

This is not an argument that KERI's decentralized PKI is a drop-in replacement for classical PKI everywhere, today. It's not that simple. Witness and watcher discipline is still maturing; tooling is young; legal and regulatory recognition of KERI-based credentials is still developing. These are real challenges. But they are constraints of *youth*, not of *architecture*. [d] PKI's constraints, by contrast, are architectural — baked into a standard that mandates certificate expiration, relies on administrators for continuity, and was designed before quantum computing was an engineering concern. Getting PKI to post-quantum security costs: upgrade cost, plus full deployment cost, paid across a global installed base, under time pressure, with a long legacy tail. Getting KERI to equivalent deployment maturity costs: full deployment cost, paid once, on greenfield, without fighting anything that came before.

The question isn't which system is abstractly superior. It is which transition is less costly, less risky, and leaves fewer gaps as the quantum deadline approaches. On current evidence, that question favors KERI — not because KERI is finished, but because it benefits from PKI's lessons learned, and began without the baggage that PKI will struggle to undo.

---

7. Conclusion

CT and KERI aren't best imagined as contemporaneous competitors, so much as responses to different stages of the same problem. CT was a necessary and valuable correction to the visibility failures of the original CA model. KERI addresses the underlying issue CT leaves untouched: administrative trust is still the root of the chain, and administrators — as decades of incidents demonstrate — are a fragile foundation for global identity infrastructure.

CT's strengths — ubiquity, browser integration, legal standing [c], mature tooling — are genuine. They describe the world as it is. However, the same properties that make CT "deployed

and working" today are what make it expensive and slow to upgrade. With a credible 2029 engineering deadline on the horizon, those costs are no longer hypothetical.

KERI's challenges — young deployments, a watcher discipline that's still crystalizing, the operational friction of its recovery model — are also genuine. A KERI deployment without robust watchers and governance is not a secure system. But these are the weaknesses of something still deploying per its design, not something whose design is running out of time.

Both systems make fraud provable rather than impossible. Both require monitoring, accountability, and human judgment to work as intended. Neither offers a free lunch. What they offer are different tools for a world where perfect prevention isn't available: CT provides administrative accountability for a credential ecosystem that is already global; KERI provides cryptographic accountability for an identity ecosystem that is still being built.

The most careful voices studying quantum timelines are shrinking the window for that choice, not expanding it.

Notes

[a] Entrust's compliance failures were not a sudden discovery. Public records from the CA/Browser Forum document a pattern of incidents stretching back years before the 2024 distrust decisions. The timeline raises a structural question: if the system's strength is its ability to detect and remove bad actors, why did detection-to-action take years rather than weeks? The answer is that centralized consensus requires all relevant parties to agree before any acts — and reaching that agreement across browser vendors, affected enterprises, and the CA itself is slow by design. A system that detects misbehavior quickly but responds slowly has a gap that architecture, not just process, should address.

[b] The concentration is worth examining. Browser root store programs — operated by Google, Mozilla, and Apple — function as the apex authority in the CT ecosystem. They decide which CAs and which CT logs are trusted, with immediate global effect and no formal appeals process. No government, no standards body, and no external adjudicator sits above them. The CA/Browser Forum sets baseline requirements, but its enforcement depends on the same browser vendors who participate in drafting those requirements. This is not a conspiracy — it is the predictable consequence of administrative trust: the chain must terminate somewhere, and wherever it terminates, the terminal authority is accountable only to itself.

[c] X.509 certificates currently enjoy legal standing under frameworks like eIDAS and various national digital signature laws that KERI-based credentials do not yet have. This is a real gap, but it is a gap of legal recognition, not of technical capability. KERI's data model is a superset of X.509's: it preserves governance transparency, key history, and trust chain structure that X.509 discards. In principle, X.509-compatible artifacts can be derived from KERI's richer substrate when legal frameworks require them — just as a JPEG can be exported from a RAW photograph. The reverse derivation is not possible. As legal frameworks evolve to accommodate new credential technologies, this asymmetry favors the lossless format as the archival source of truth.

[d] KERI's operational complexity — pre-rotation, witness management, multisig governance, recovery planning — is sometimes cited as a barrier relative to simpler tools like certbot. But the complexity is load-bearing: each feature provides a security guarantee that X.509 cannot offer

and that cannot be retrofitted onto a system not designed for it. An ecosystem must be designed for the strongest guarantees its participants require, with the option to relax them for simpler use cases. X.509 starts from the simple end and cannot scale up to transparent governance, pre-rotation, or multisig. KERI starts from the strong end and permits simpler configurations — a single-key AID with minimal witness infrastructure — when that is all that is needed.

References

[1] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. IETF. https://doi.org/10.17487/RFC5280

[2] Fox-IT. 2012. DigiNotar Certificate Authority Breach "Operation Black Tulip." https://docslib.org/doc/2849857/diginotar-certificate-authority-breach-operation-black-tulip (archived copy)

[3] Sleevi, R. 2017. Distrust of Symantec TLS Certificates. Google Security Blog. https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html

[4] Mozilla. 2022. Concerns about TrustCor Systems. Mozilla Dev Security Policy (discussion thread). https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/oxX69KFvsm4

[5] Google. 2024. Sustaining Digital Certificate Security — Entrust Certificate Distrust. Google Security Blog. https://security.googleblog.com/2024/06/sustaining-digital-certificate-security.html

[6] Wilson, B. 2024. Mozilla's Decision on Entrust's Root CA Certificates Used for TLS. Mozilla Dev Security Policy. https://groups.google.com/a/mozilla.org/g/dev-security-policy/c/jCvkhBjg9Yw

[7] Apple. 2024. Changes to Certification Authorities and Certificates. Apple Support. https://support.apple.com/en-us/121668

[8] Censys. 2024. How the Removal of Entrust from Chrome's Root Store Will Affect the Internet. Censys Blog. https://censys.com/blog/google-entrust-internet/

[9] Liu, Y., Tome, W., Zhang, L., Choffnes, D., Levin, D., Maggs, B., Mislove, A., Schulman, A., and Wilson, C. 2015. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In Proceedings of IMC 2015. https://doi.org/10.1145/2815675.2815685

[10] Laurie, B., Langley, A., and Kasper, E. 2013. Certificate Transparency. RFC 6962. IETF. https://doi.org/10.17487/RFC6962

[11] Chuat, L., Szalachowski, P., Perrig, A., Laurie, B., and Messeri, E. 2015. Efficient Gossip Protocols for Verifying the Consistency of Certificate Logs. In Proceedings of IEEE CNS 2015. https://doi.org/10.1109/CNS.2015.7346817

[12] Meiklejohn, S., Gilad, Y., Goldberg, S., Ioannidis, S., Mislove, A., and Mitzenmacher, M. 2022. SoK: SCT Auditing in Certificate Transparency. Proceedings on Privacy Enhancing Technologies 2022, 4. arXiv:2203.01661. https://arxiv.org/abs/2203.01661

[13] Sun, A., Lin, J., Wang, W., Liu, Z., Li, B., Wen, S., Wang, Q., and Li, F. 2024. Certificate Transparency Revisited: The Public Inspections on Third-party Monitors. In Proceedings of NDSS 2024. https://dx.doi.org/10.14722/ndss.2024.24834

[14] Chrome Team. 2024. Chrome Certificate Transparency Policy. Google. https://googlechrome.github.io/CertificateTransparency/ct_policy.html

[15] Scheitle, Q., Gasser, O., Nolte, T., Amann, J., Brent, L., Carle, G., Holz, R., Schmidt, T. C., and Wählisch, M. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In Proceedings of IMC 2018. https://doi.org/10.1145/3278532.3278562

[16] Azevedo, A. C., Scheid, E. J., Franco, M. F., and Granville, L. Z. 2025. Assessing SSL/TLS Certificate Centralization: Implications for Digital Sovereignty. arXiv:2504.16897. https://arxiv.org/abs/2504.16897

[17] Adams, C. and Lloyd, S. 2003. Understanding PKI: Concepts, Standards, and Deployment Considerations (2nd ed.). Addison-Wesley Professional.

[18] Gidney, C. and Schmieg, S. 2025. Tracking the Cost of Quantum Factoring. Google Security Blog / arXiv:2505.15917. https://security.googleblog.com/2025/05/tracking-cost-of-quantum-factori.html

[19] Adkins, H. and Schmieg, S. 2026. Quantum Frontiers May Be Closer Than They Appear. Google. https://blog.google/innovation-and-ai/technology/safety-security/cryptography-migration-timeline/

[20] Moody, D., Perlner, R., Regenscheid, A., Robinson, A., and Cooper, D. 2024. Transition to Post-Quantum Cryptography Standards. NIST IR 8547 (initial public draft). https://doi.org/10.6028/NIST.IR.8547.ipd

[21] Benjamin, D., O'Brien, D., Westerbaan, B., Valenta, L., and Valsorda, F. 2026. Merkle Tree Certificates. IETF Internet-Draft, work in progress (PLANTS WG). https://datatracker.ietf.org/doc/draft-ietf-plants-merkle-tree-certs/

[22] Brewer, E. A. 2000. Towards Robust Distributed Systems. Keynote, ACM Symposium on Principles of Distributed Computing (PODC). See also: Gilbert, S. and Lynch, N. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News 33, 2, 51–59.

[23] Mead, N., Ellison, R., Linger, R., Longstaff, T., and McHugh, J. 2000. Survivable Network Analysis Method. CMU/SEI-2000-TR-013. Carnegie Mellon University Software Engineering Institute. https://www.sei.cmu.edu/library/survivable-network-analysis-method/

[24] Bernstein, D. J. and Lange, T. 2017. Post-Quantum Cryptography: Dealing with the Fallout of Physics Success. IACR Cryptology ePrint Archive 2017/314. https://eprint.iacr.org/2017/314

[25] National Institute of Standards and Technology. 2016. Post-Quantum Cryptography: Security (Evaluation Criteria). CSRC. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)

[26] Smith, S. M. 2024. Key Event Receipt Infrastructure (KERI) Specification (v2.7.0). Trust Over IP Foundation. https://trustoverip.github.io/kswg-keri-specification/

[27] Del Giudice, D. 2025. A Security Analysis of KERI. Master's Thesis, ETH Zurich. https://doi.org/10.3929/ethz-b-000735690

[28] Castro, M. and Liskov, B. 1999. Practical Byzantine Fault Tolerance. In Proceedings of OSDI 1999. https://pmg.csail.mit.edu/papers/osdi99.pdf

[29] Lamport, L., Shostak, R., and Pease, M. 1982. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4, 3, 382–401. https://doi.org/10.1145/357172.357176

[30] International Organization for Standardization. 2024. ISO 17442-3:2024 Financial Services — Legal Entity Identifier (LEI) — Part 3: Verifiable LEIs (vLEIs). ISO.

[31] GSMA Foundry. 2025. Open Verifiable Calling Is Live. GSMA. https://www.gsma.com/get-involved/gsma-foundry/ovc/open-verifiable-calling-is-live/