Kilimanjaro

TECHNICAL WHITEPAPER

DATA PROTECTION & SOVEREIGNTY

Verifiable Decommissioning

When you delete, the key is destroyed and the data becomes unreadable — in live storage and every backup — and you hold a signed record for the regulator. How Kilimanjaro erases by destroying the key, not scrubbing the rows.

June 2026 · v1.0 · 12 min read

01 “We deleted it” is a claim, not a record
02 The mechanism: destroy the key, not the rows
03 The lifecycle: a signed state machine, not a fire-and-forget delete
04 The receipt: signed, and you can verify it
05 The authorization: no one administrator can erase alone
06 Erasing one person
07 How it composes: custody tiers and derived data

Executive summary

Deleting a row is a claim. Destroying the key leaves a signed record. The hard part of the right to erasure has never been the live record — it is the backups, where scrubbing data is impractical or impossible, and where "we deleted it" quietly becomes unverifiable.

Kilimanjaro erases by cryptographic key destruction — crypto-shred. Data is field-encrypted under a per-project key; once that key is purged, every field encrypted under it, in live storage and in every backup, is permanently undecryptable. No backup is scrubbed; the backup ciphertext was never readable without the key in the first place.

And it is not a fire-and-forget delete. Erasure runs as a signed state machine — immediate functional erasure, a recoverable window, a permanent purge, a signed receipt a regulator can verify against a published key, and, at D+28, a second signed record that the key is gone. This paper walks the mechanism, the lifecycle, the receipt, and — precisely — the boundary of what crypto-shred does and does not cover.

01 “We deleted it” is a claim, not a record

The right to erasure is now table stakes: GDPR Article 17, the CPRA’s right to delete (§1798.105), Australia’s APP 11.2, Saudi PDPL Article 18, Nigeria’s NDPA §37. Each asks for the same thing in its own language — when a data subject’s data must go, it must actually be gone.

Deleting the live row is the easy half. The hard half is everything else: replicas, snapshots, and backups, where deleting individual records is impractical and often impossible. A backup taken yesterday still holds the data you deleted today. "We deleted it" is a claim about the live store; the regulator is asking about the whole footprint, and asking for evidence.

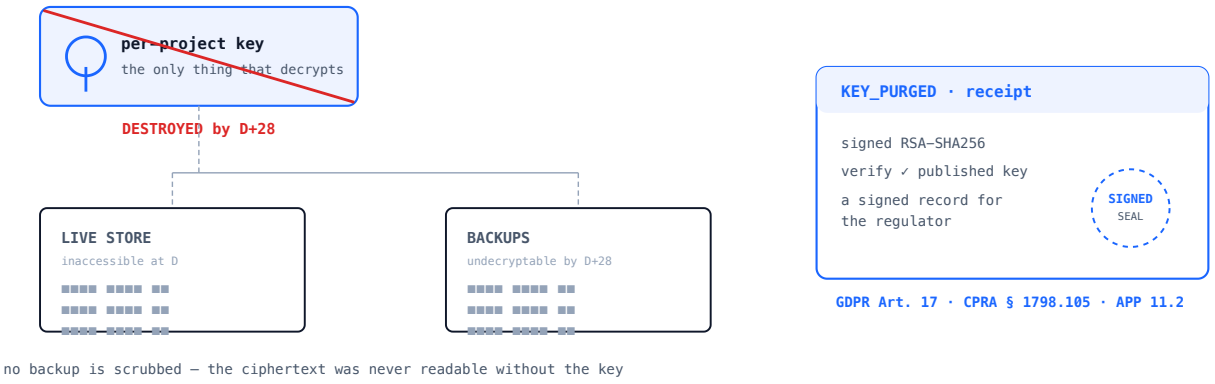
Framework	The erasure obligation
GDPR	Article 17
CPRA	Right to delete, § 1798.105
Australia	APP 11.2
Saudi PDPL	Article 18
Nigeria	NDPA § 37

The right to erasure, phrased in each statute’s own language — all asking for the same thing.

02 The mechanism: destroy the key, not the rows

Kilimanjaro field-encrypts data under a per-project key (the PKEK that wraps each per-field DEK). Erasure is the destruction of that per-project key — the key vault purges the project key and all its versions. Once the key is purged, every field encrypted under it is permanently undecryptable, wherever the ciphertext sits: live tables, replicas, and backups alike. From D the key no longer resolves, and that is reversible until D+28. This is cryptographic erase in the NIST SP 800-88 sense — and it is the only erasure model that reaches backups without scrubbing them.

We scope this precisely, because precision is the whole point. Crypto-shred covers the data encrypted under your per-project key: vault secrets, the fields you designate as encrypted, and tokenized values. Data in non-designated plaintext columns is not key-destroyed — it is erased by deletion, by bounded-retention backups that hold only ciphertext beyond your SLA window, and by tenant isolation. We do not claim crypto-shred erases "all your data"; we claim it for the per-project-key-encrypted surface, and we draw that line in the open.



Crypto-shred: once the per-project key is purged, every field encrypted under it — live and in every backup — is permanently undecryptable, with a signed KEY_PURGED receipt recording the commitment.

03 The lifecycle: a signed state machine, not a fire-and-forget delete

Erasure is not a single irreversible instant — that would make an accidental deletion catastrophic. It is a signed state machine with deliberate recovery windows: a 24-hour window to cancel after approval, then the 28-day window below. At D, the request is soft-deleted and the key stops resolving: that is immediate functional erasure — every key-bearing operation now returns a 410 “project zeroized” response, the clear signal that the guarantee has been delivered. The platform records a signed two-point commitment — the soft-deletion time and the completion deadline at D+28.

The 28-day window does triple duty. It is the immediate functional erasure at D; it is a 28-day, admin-recoverable undo — the safety net for an irreversible action; and it is the permanent purge at D+28, when the key bytes are removed from the key vault and even operator recovery becomes impossible. At D+28 the state machine reaches its terminal state, and a second signed record, KEY_ERASURE_CONFIRMED, records that the key is gone. Functional erasure is immediate; permanence is on a clock you can see.

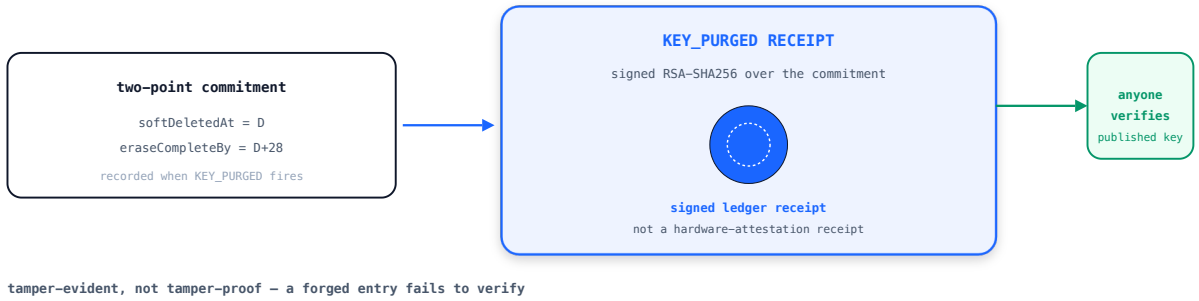


Erasure runs as a signed state machine: functional erasure is immediate at D; a 28-day window provides an admin-recoverable undo; the key bytes are permanently purged at D+28.

04 The receipt: signed, and you can verify it

A regulator does not want assurances; it wants a record it can check. When the key is revoked and scheduled for purge, Kilimanjaro issues a KEY_PURGED receipt — a platform-signed (RSA-SHA256) statement over the two-point commitment, verifiable against a published key. Anyone holding the public key can confirm the platform committed to the erasure, and when.

We are exact about what kind of receipt this is. It is a signed ledger receipt — a signature over the erasure record on the hash-chained audit ledger — not a hardware-attestation receipt. The ledger is tamper-evident by construction, not "tamper-proof," and the receipt proves the platform's signed commitment, not a hardware quote. It shows that the platform revoked the key at D and committed to purge it by D+28; until D+28 an administrator can still abort a key erase. The second record, KEY_ERASURE_CONFIRMED at D+28, is the signed record that the key is gone.

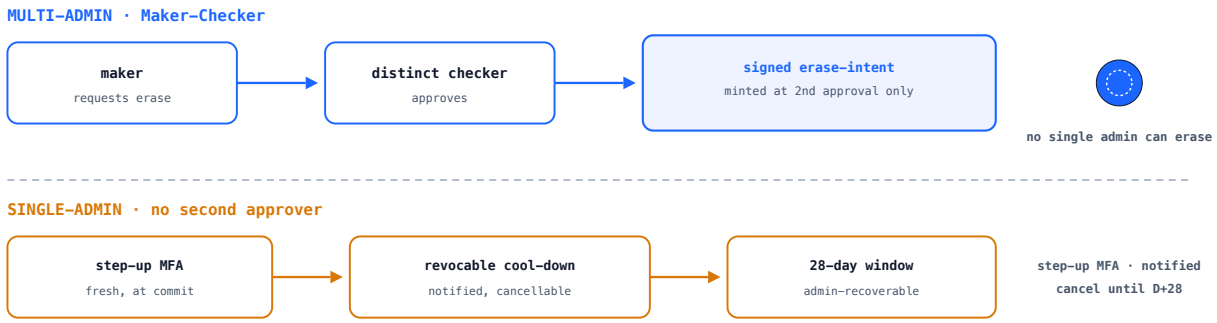


The KEY_PURGED receipt is a platform-signed (RSA-SHA256) statement over the two-point commitment — softDeletedAt and eraseCompleteBy — verifiable by anyone holding the published key. It is a signed ledger receipt, not a hardware-attestation receipt.

05 The authorization: no one administrator can erase alone

Erasure is destructive, so the authority to trigger it is governed. In a multi-admin organization, a destructive erase requires two distinct Org-Admin or Org-Owner approvals — Maker-Checker — and the signed erase-intent is minted at the second approval, attesting that a complete two-party authorization took place. No one administrator can erase the project alone.

We are careful with that claim for smaller teams. In a single-admin organization there is no second approver, so the scoped statement is narrower: a single administrator's erasure needs fresh step-up MFA, is notified, and can be cancelled until D+28. The erase requires fresh step-up MFA at the moment of commit, triggers a revocable, notified cool-down, and still passes through the 28-day recoverable window. We do not claim universal two-person approval where it does not exist; we claim what is true at each org shape.



Multi-admin organizations require two distinct Org-Admin or Org-Owner approvals — Maker-Checker — with the signed erase-intent minted only at the second approval. Single-admin organizations get the narrower, scoped guarantee: fresh step-up MFA at commit, a revocable notified cool-down, and the 28-day recoverable window.

06 Erasing one person

Erasing one person does not destroy the project key. Destroying that person’s field key makes their encrypted data unreadable in live storage at once, and backup copies age out within 28 days. The documentation on cryptographic erasure covers the full lifecycle.

07 How it composes: custody tiers and derived data

Crypto-shred composes with the custody tiers. In the platform-managed tier, the erase intent is signed by a dedicated platform key after you authenticate — it represents your authorization, and it is not customer-key custody. With BYOK or HYOK, the customer's key is detached and retained rather than destroyed by the platform.

The same distinction extends to what Kilimanjaro derives from your data — Context embeddings, for example. They live inside your project's own custody domain and are torn down when the project is. If an embedding is itself field-encrypted under your project key, destroying the key crypto-shreds it too, backups included.

Today, embeddings used for search are typically stored as plaintext rather than encrypted, so they follow the same rule as any other plaintext column: erased by deletion, and gone from backups once those age past your retention window. Either way, nothing derived from your data outlives the project — we do not claim that destroying the key alone erases a plaintext embedding, because it does not.

Case	What happens
Platform-managed erase intent	Signed by a dedicated platform key after you authenticate — your authorization, not customer-key custody
BYOK / HYOK key	Detached from the project and retained, not destroyed by the platform
Field-encrypted embedding	Crypto-shredded when the project key is destroyed, backups included
Plaintext embedding (today's typical case)	Erased by deletion, then gone from backups once they age past your retention window

How the erase intent is authorized, and how derived data (Context embeddings) is actually erased, by custody tier and encryption state.

What we disclose

Scope. Crypto-shred renders permanently undecryptable all data encrypted under your per-project key — vault secrets, the fields you designate as encrypted, and tokenized values — in live storage and in (ciphertext-only) backups. Data in non-designated plaintext columns is erased by deletion, bounded-retention backups, and tenant isolation, not by key destruction. We scope the claim to the per-project-key-encrypted surface, not "all your data."

Authorization and receipt. At the platform-managed tier the erase intent is signed by a dedicated platform key after you authenticate — it represents your authorization; it is not customer-key custody. The KEY_PURGED receipt is a signed ledger receipt (RSA-SHA256, verifiable against a published key), not a hardware-attestation receipt; the ledger is tamper-evident, not tamper-proof.

"No one administrator." That guarantee is precise for multi-admin organizations — two distinct approvers. Single-admin organizations get the narrower, scoped claim: a single administrator's erasure needs fresh step-up MFA, is notified, and can be cancelled until D+28. The 28-day window is a deliberate recoverability net — functional erasure is immediate at D, permanent at D+28. The cryptographic primitives are classical, current-generation (AES-256-GCM-SIV, Ed25519, RSA-2048/SHA-256, HMAC-SHA-256); we make no post-quantum claim.

See the erasure record for yourself

Talk to a Sovereign Engineer about your erasure obligations — and the signed, verifiable receipt Kilimanjaro produces when a key is revoked and scheduled for purge.

Talk to a Sovereign Engineer →

Browse the compliance frameworks