Document 1: Audit-Verifiable Destruction Receipts (EIV Implementation)

Draft — Technical Specification (Hybrid Patent/Spec Format)

---

1. Abstract

This document defines a cryptographic method for proving that an initialization vector or key component has been irreversibly destroyed. The mechanism transforms key expiration from a trust-based operational event into a verifiable cryptographic state transition. It uses Ephemeral Initialization Vectors (EIVs) with bounded lifetimes, forward-secure key evolution, and destruction receipts that are anchored into tamper-evident data structures (e.g., Merkle trees). Auditors can verify destruction without accessing any secret material. This is the first of five documents introducing Cryptographic State Transition Proofs (CSTP).

---

2. Technical Field

This system operates at the intersection of:

- Cryptographic key management
- Compliance and audit systems
- Forward-secure cryptographic state transitions
- Data lifecycle governance (creation, retention, destruction)

---

3. Background and Problem Statement

Modern cryptographic systems rely on encryption keys or initialization vectors (IVs) that are assumed to expire or be destroyed per policy. However:

- There is no cryptographic proof of destruction.
- Keys may persist in RAM, swap, logs, or replicated backups.
- Regulators accept attestations rather than evidence.
- "Delete after 30 days" is not technically enforceable or verifiable.

Existing Merkle-ledger systems prove *existence* of data, not *non-existence*. EIVs invert this: they anchor destruction events instead.

4. Definitions and Symbols

Symbol	Meaning
EIV_i	Ephemeral Initialization Vector at epoch i
T_i	Expiry timestamp or epoch index
H()	Cryptographic hash function
DR_i	Destruction receipt for EIV_i
R_i	Merkle root committing a set of destruction receipts
$path_i$	Merkle branch proving inclusion of DR_i in R_i
sys_meta	System metadata (asset ID, policy ID, node ID, etc.)
$epoch\_root_i$	Root commitment for all receipts in epoch i

5. System Model

5.1 Parties

- **Generator:** System that creates and uses EIVs
- **Destroyer:** Process that evolves EIVs forward and deletes past states
- **Recorder:** Maintains Merkle accumulator of destruction receipts
- **Verifier/Auditor:** Independent process validating receipts

5.2 Assumptions

- Hash function is collision-resistant and one-way
- Prior EIV states are deleted after forward evolution
- Merkle accumulator roots (R_i) are retained and optionally externally anchored
- Auditor does not need access to plaintext or keys

---

6. Core Mechanism

6.1 EIV Generation

$EIV_0 = H(seed \,||\, sys\_meta \,||\, nonce)$
assign expiration T_0

Metadata bound to EIV includes:

- Asset or session identifier
- Creation timestamp
- Allowed persistence duration

Refer to **EIV Diagram 1: Initialization and Metadata Commitment**

6.2 Expiry and Forward Evolution

At time $\geq T_i$, perform:

$EIV_{i+1} = H(EIV_i)$
delete EIV_i

This ensures:

- EIV_i cannot be reconstructed
- EIV_{i+1} is forward-secure
- All encryption reliant on EIV_i becomes unrecoverable

6.3 Destruction Receipt Structure

A destruction receipt DR_i consists of:

Field	Description
$commit_i$	`$H(EIV_i$
epoch	Destruction epoch or timestamp
hash_after	$H(EIV_{i+1})$ proving forward evolution occurred
$path_i$	Merkle path from $commit_i$ to root R_i
root	Merkle root for that epoch
signature (optional)	System or HSM signature

Refer to **EIV Diagram 2: Destruction Receipt Composition**

6.4 Merkle Aggregation and Root Chaining

Receipts from the same epoch are inserted into a Merkle tree:

$R_i = \mathrm{MerkleRoot}(\{commit_1, commit_2, ..., commit_n\})$

Roots may be chained:

$Chain_k = H(R_k \| Chain_{k-1})$

Roots can optionally be:

- Published to transparency logs
- Anchored to blockchain, HSM logs, or time-stamping services
- Submitted to regulatory hash registries

Refer to **EIV Diagram 3: Merkle Accumulator and Root Chaining**

---

7. Verification Process

An auditor verifying destruction of EIV_i:

1. Receives receipt DR_i
2. Computes $commit_i = H(sys\_meta \| T_i \| \ldots)$
3. Verifies Merkle path up to root R_i
4. Checks root against published log or authoritative record
5. Optionally validates signature or external timestamp
6. Confirms non-recoverability of prior state (absence of EIV_i)

No secret material is leaked. No access to destroyed key is required.