Heirloom: A Non-Custodial Protocol for Decentralised Digital Asset Succession

Whitepaper Version 1.0
Status: Mainnet-Ready Protocol Specification
Date: January 2026
Licence: Business Source License (BUSL-1.1)

Abstract

The irreversible nature of blockchain transactions and the absolute sovereignty of private keys have introduced a structural flaw into the digital economy: the absence of a native, trustless mechanism for asset succession. While decentralised finance (DeFi) has succeeded in eliminating intermediaries from payments, lending, and settlement, it has failed to address the inevitability of human mortality and permanent key loss.

This paper introduces **Heirloom**, a fully non-custodial, smart-contract-based protocol designed to enable decentralised inheritance of ERC-20 digital assets without requiring the transfer of custody, disclosure of private keys, or reliance on off-chain legal enforcement. Heirloom operates through immutable allowance-based execution, a unified inactivity deadline ("Proof-of-Life"), and a mathematically rigorous virtual accounting model that preserves proportional entitlements under dynamic balances.

The protocol is deliberately minimalist, permissionless, and governance-free, prioritising correctness, survivability, and legal neutrality over extensibility. This document presents the theoretical foundations, technical architecture, security assumptions, economic design, and legal positioning of Heirloom, with the aim of enabling independent verification, academic scrutiny, and long-term public deployment.

---

1. Introduction: Digital Assets and the Inheritance Vacuum

1.1 The Irreversibility Problem

Public blockchains deliberately eliminate discretionary authority. Transactions are final, keys are absolute, and there exists no native concept of recovery, override, or exception. While this architecture maximises censorship resistance and financial sovereignty, it simultaneously creates a paradox: **assets outlive their owners, but ownership does not survive key loss**.

Unlike traditional financial systems—where inheritance is enforced by courts, registries, and custodians—blockchain systems recognise only cryptographic control. Upon death or permanent loss of access, assets become economically inert, despite remaining cryptographically valid.

Empirical estimates suggest that between 15–25% of major crypto-assets are permanently inaccessible. This represents not merely personal loss, but systemic capital inefficiency.

1.2 Why Traditional Inheritance Models Fail in Web3

Conventional inheritance relies on three pillars:

1. **Identity verification**
2. **Judicial authority**
3. **Custodial enforcement**

All three are antithetical to decentralised systems. Smart contracts cannot verify death, courts cannot compel private keys, and custodians negate self-sovereignty.

Existing Web3 attempts to solve inheritance typically fall into one of the following categories:

- **Key sharing or escrow**, which introduces catastrophic theft risk during the owner's lifetime.
- **Custodial vaults**, which reintroduce counterparty and regulatory risk.
- **Smart-contract wallets**, which require full migration of assets and impose complex operational burdens.

Heirloom rejects all three approaches.

---

2. Design Philosophy and Core Principles

Heirloom is built on five non-negotiable principles:

1. **Non-custodial by construction**
   The protocol must never hold user funds.
2. **Owner sovereignty until inactivity**
   The owner retains full, unrestricted control during life.
3. **Objective execution conditions**
   No oracles, courts, or subjective assessments.
4. **Mathematical determinism**
   Distribution must be provably correct under all balance states.
5. **Protocol immutability**
   No governance, no upgrade keys, no fee manipulation.

These principles intentionally limit feature scope. Heirloom is not a wallet, not an estate planner, and not a DAO. It is a *cryptographic executor of deferred intent*.

---

3. Conceptual Framework: Allowance-Based Succession

3.1 Decoupling Custody from Execution Rights

At the heart of Heirloom lies a fundamental insight: **ERC-20 allowances already encode conditional future execution**.

By granting an allowance to a smart contract, an owner authorises future transfers *without relinquishing custody*. This mechanism is natively supported, widely audited, and universally deployed.

Heirloom leverages this primitive to act as an *executor*, not a vault.

At no point does the protocol:

- Receive tokens
- Lock balances
- Restrict owner spending
- Intercept owner transactions

The allowance merely caps what *may* be transferred **after** predefined conditions are met.

3.2 Why ERC-20 Only

Native Ether does not support allowances. Supporting it would require wrapping, custody, or bespoke wallet logic. Heirloom therefore intentionally limits scope to ERC-20 assets, covering the overwhelming majority of real economic value (stablecoins, governance tokens, RWAs).

This decision prioritises **correctness and security over universality**.

---

4. Proof-of-Life via Inactivity Deadlines

4.1 The Inactivity Assumption

In decentralised systems, death cannot be observed. However, *continued activity can*.

Heirloom therefore defines “death” as **cryptographic inactivity exceeding a user-defined threshold**.

This threshold is encoded as a single timestamp:

ownerExecutionTimestamp

Any interaction with the protocol's resetDeadline function refreshes this timestamp.

4.2 Unified Deadline Model

Unlike designs that maintain per-beneficiary or per-asset timers, Heirloom employs a **single global deadline per owner**.

This has three advantages:

1. **Atomic Proof-of-Life**
   One action secures all legacies simultaneously.
2. **Gas efficiency**
   No fragmented state updates.
3. **Predictability**
   All beneficiaries observe the same execution condition.

The system deliberately treats *key loss* identically to death. This is not a flaw but a design choice: the protocol resolves *access loss*, regardless of cause.

---

5. Virtual Pool Accounting and Mathematical Correctness

5.1 The Dynamic Balance Problem

Because assets remain in the owner's wallet, balances may change over time. A naïve inheritance scheme would fail under partial withdrawals or deposits.

Heirloom solves this using **Virtual Pool Accounting**.

Rather than allocating fixed amounts, the protocol allocates **proportional entitlements** over a moving balance.

5.2 Formal Model

For each token, the protocol tracks:

- Total virtual released amount
- Per-beneficiary claimed amount
- Assigned share (basis points)

Each claim computes entitlement dynamically as a function of:

- Current owner balance
- Total previously released value
- Beneficiary's proportional share

This ensures:

- Order-independent claims
- No over-distribution
- Exact proportionality at all times

The model is resistant to front-running, partial execution, and delayed claims.

---

6. Technical Architecture

6.1 Contract Topology

Heirloom is implemented as a single immutable contract acting as:

- Registry
- Executor
- Accounting engine

No proxy pattern is used. No upgrade path exists.

6.2 Data Structures

Legacies are stored in tightly packed structs indexed by owner address.

Auxiliary indexed arrays enable frontend discovery without sacrificing execution efficiency.

Pagination is mandatory for all enumerable views to avoid gas exhaustion.

6.3 Token Transfer Safety

All token interactions use SafeERC20.

Fee-on-transfer and deflationary tokens are handled by measuring actual received balances, ensuring accounting correctness even under non-standard token mechanics.

---

7. Security Model

7.1 Threat Model

Heirloom assumes:

- Adversarial beneficiaries

- Adversarial tokens
- Absent or hostile frontend
- Permanent protocol immutability

It does **not** assume:

- Honest behaviour
- Legal enforcement
- Centralised maintenance

7.2 Defensive Measures

- Reentrancy protection
- Strict CEI pattern
- Explicit invariants on share allocation
- Hard-coded fee constants
- No owner or admin privileges

Once deployed, the protocol is operationally sovereign.

---

8. Economic Design and Fee Philosophy

8.1 Fixed-Fee Model

All protocol fees are immutable constants.

This eliminates:

- Governance risk
- Fee capture attacks
- Rent-seeking incentives

Fees serve three purposes only:

1. Spam prevention
2. Network sustainability
3. Economic seriousness signalling

8.2 Absence of a Token

Heirloom intentionally has no native token.

There is no governance, no staking, no inflation, and no financial abstraction layer beyond execution fees.

This is a protocol, not a financial instrument.

---

9. Legal Positioning

Heirloom is software.

It:

- Does not custody assets
- Does not intermediate transactions
- Does not determine beneficiaries
- Does not verify identity or death

Execution is entirely user-defined and cryptographically enforced.

As such, the protocol operates as **neutral technical infrastructure**, analogous to a compiler or encryption library.

---

10. Conclusion

Heirloom addresses one of the last unsolved problems in decentralised finance: **what happens when the key holder is gone**.

By refusing custody, governance, and subjective judgment, it achieves something rare in Web3: a system that remains correct even when everyone disappears—including its creators.

It does not promise comfort.
It promises finality, correctness, and permanence.

Legal Disclaimer

Heirloom is provided “as is”, without warranty of any kind. The protocol does not constitute legal, financial, or estate-planning advice. Users are solely responsible for compliance with applicable laws and for ensuring that their use of the protocol aligns with their personal, legal, and jurisdictional circumstances.

The developers, contributors, and publishers of this document disclaim any liability arising from the use, misuse, or inability to use the protocol.

Appendix A: Claiming Assets Without a Frontend (Etherscan Procedure)

This appendix ensures protocol survivability even if all user interfaces cease to exist.

Step 1: Locate the Contract

Navigate to the verified Heirloom contract on Etherscan.

Step 2: Connect Wallet

Use “Write Contract” → “Connect to Web3”.

Step 3: Verify Eligibility

Call the read function to confirm:

- Deadline has expired
- You are listed as a beneficiary

Step 4: Execute Claim

Call claimLegacies with:

- Owner address
- Token address
- Your beneficiary index

Step 5: Confirm Transfer

Etherscan will display the transaction and transferred amount.

No frontend, operator, or third party is required.
