TVS CodeStamp
Threat Model & Trust Design – Technical Whitepaper (v1.0)

Author: TimeVaultSecure Research Team
Date: April 2025

1. Introduction

TVS CodeStamp is a decentralized, cryptographic timestamping mechanism designed for proof-of-existence, authenticity, and time anchoring of digital data — without requiring third-party authorities or blockchain infrastructure. This document describes the trust assumptions, security goals, threat surfaces, and design principles behind TVS CodeStamp.

2. Functional Architecture

Each file uploaded to TimeVaultSecure receives a 64-character hexadecimal identifier:

tvs_codestamp = HMAC(daily_secret, file_hash + timestamp)

Where:
- file_hash = SHA-256 hash of file contents
- timestamp = UNIX time in seconds
- daily_secret = HMAC(base_secret, YYYY-MM-DD)
- base_secret = private server-side 256-bit key (never exposed)

The result is stored in a downloadable PDF and may be verified offline at any time.

3. Security Assumptions & Guarantees

Feature	Guarantee
Code uniqueness	Every second & file_hash yields unique code
Backdating resistance	Impossible without access to base_secret
Offline verifiability	Any party can recompute from file_hash + timestamp + algorithm
Independence	Verification does not require TVS servers or user login
Tamper-proofing	HMAC-SHA256 ensures non-reversibility and key-dependent entropy
Optional anchoring	Code + hash can be submitted to OpenTimestamps

4. Threat Scenarios

Scenario 1: TSA Equivalence / Notary Challenge
Threat: TSA claims are considered more "legally binding"
Mitigation: TVS CodeStamp may be optionally notarized or blockchain-anchored. Full audit trail + open algorithm provides legal traceability.

Scenario 2: Secret Key Leakage
Threat: Compromise of base_secret
Impact: Backdating becomes theoretically possible.
Mitigation: Daily key rotation, key encryption, vault/HSM storage. Public fingerprints of daily secret available (optional).

Scenario 3: User submits fake timestamp
Threat: Tampering with timestamp in PDF
Impact: Code verification fails
Mitigation: Timestamp not accepted unless HMAC check passes

5. Cryptographic Design

- HMAC with SHA-256 (or optionally SHA3 / BLAKE3)
- Deterministic: same input always yields same code
- Resistant to replay: depends on precise second
- Sensitive to byte-level change in file
- One-way: HMAC cannot be reversed to reveal timestamp or file_hash

6. Trust Model

Component	Trust Assumption
base_secret	Kept securely server-side
daily_secret	Derived & deleted after use (optional)
user input	Hash must match real file hash
client system	Can verify independently

7. Transparency & Auditing

- Public algorithm
- Optional open-source CLI verifier
- Optionally published test vectors
- Blockchain anchoring planned (OpenTimestamps)
- No usage logs, no tracking, no server dependence

8. Use Cases

- Proof of authorship / first publication
- Time-bound access control (capsules)
- Secure digital legacy packaging
- Timestamping of contracts, ideas, media files
- Offline legal proof without blockchain costs

9. Future Directions

- Add QR with direct verification link
- Implement post-quantum version (SHA3, BLAKE3-HMAC)
- Integrate with TVS Capsule unlock rules
- Timestamp smart contracts & interactions

10. Summary

TVS CodeStamp brings cryptographic certainty to time.
It removes the need for centralized timestamp providers while remaining verifiable, secure, lightweight and user-controlled.

By combining strong cryptography, open access, and offline independence — TVS sets a new standard in privacy-first timestamping for the post-authority internet.