Ɉ Montana v4.3

Ɉ Montana: Time-Proven Human Temporal Currency

Version 4.3 Alejandro Montana alejandromontana@tutamail.com **December 2025**

Abstract

A peer-to-peer quantum-resistant electronic cash system without reliance on financial institutions. Existing cryptocurrency solutions—Proof of Work and Proof of Stake—scale influence through purchasable resources, concentrating power in capital owners.

Ɉ Montana ($MONT) builds consensus on Proof of Time. Influence accumulates through time presence, not resource expenditure. The network timestamps blocks through sequential computation that cannot be parallelized or accelerated.

Core innovations: - **Proof of Time** — VDF-based consensus where time cannot be bought or accelerated - **11 Pantheon Gods** — Modular architecture with clear separation of concerns - **12 Apostles** — Trust network with collective accountability - **Hal Humanity System** — Sybil resistance through graduated trust and time-locked proofs - **Bitcoin Anchoring** — TIME dimension tied to 210,000 BTC blocks per epoch - **Post-Quantum Cryptography** — SPHINCS+, SHA3-256, SHAKE256 VDF

Time cannot be bought, manufactured, or transferred—only spent. Humanity cannot be faked across Bitcoin halvings—only proven.

Table of Contents

1. Introduction

The cypherpunk movement envisioned cryptographic systems that shift power from institutions to individuals. Bitcoin delivered a monetary system without central authority. But Bitcoin's consensus mechanism contains a flaw that becomes more apparent with time: influence scales with capital.

Proof of Work requires specialized hardware. A participant with capital purchases ASICs and controls hashrate proportional to investment. Proof of Stake makes this explicit—stake coins, receive influence. Both systems work. Both systems concentrate power.

True decentralization requires a resource that cannot be accumulated, purchased, or transferred.

Time is that resource.

A node operating through a full Bitcoin halving cycle (210,000 blocks, ~4 years) accumulates the same influence whether owned by a billionaire or a student. This time is measured in Bitcoin blocks, resets at each halving, and is irreversible. It cannot be bought on an exchange. It cannot be rented from a cloud provider. It can only be spent—by existing.

1.1 The Quantum Threat

Current cryptographic systems face an existential threat: quantum computers. Shor's algorithm breaks ECDSA, RSA, and X25519. Conservative estimates place cryptographically-relevant quantum computers 10-15 years away.

Ɉ Montana implements quantum-resistant cryptography to ensure long-term security.

1.2 The Humanity Problem

TIME proves existence, not uniqueness. An attacker can create 100 keypairs, wait 4 years, and control a coordinated network.

The Hal Humanity System solves this: proving humanity, not just cryptographic identity.

Named after Hal Finney (1956-2014), who received the first Bitcoin transaction and understood Sybil resistance before anyone else.

2. The Plutocracy Problem

All existing consensus mechanisms suffer from the same fundamental weakness: resource dependence creates plutocratic capture.

In Proof of Work, hash rate is purchasable. In Proof of Stake, the problem is structural. Delegated systems (DPoS) merely add intermediaries.

The solution is to build consensus on resources that cannot be unequally distributed.

- **Time** passes for everyone at the same rate. This is physics.
- **Humanity** cannot be multiplied. One person = one human.

---

3. Verifiable Delay Functions

A Verifiable Delay Function (VDF) requires sequential operations to compute, but whose output can be efficiently verified.

3.1 SHAKE256 VDF Construction

```
Parameters:
  H = SHAKE256 (extendable-output function)
  T = iteration count (1,000,000 iterations = 10 minutes)

Compute(x, T):
  state$_0$ = x
  for i = 1 to T:
    state$_i$ = H(state$_{i-1}$)
  return state⍰

Verify(x, y, proof, T):
  return STARK_verify(x, y, proof, T)
```

Quantum Resistance: SHAKE256 security relies only on hash function properties.

3.2 VDF Synchronization

Each VDF proof depends on the previous block hash:

```
VDF_input = SHA3_256(prev_block_hash || height)
VDF_output = Compute(VDF_input, T)
```

Pre-computation is impossible because `prev_block_hash` is unknown until the previous block is finalized.

3.3 Seven Temporal Levels

```
┌─────────────────────────────────────────────────────────────┐
│  ADAM: GOD OF TIME - 7 TEMPORAL LEVELS                      │
├─────────────────────────────────────────────────────────────┤
│  Level 0: PROOF_OF_HISTORY     6 seconds    SHA3 chain      │
│  Level 1: VDF_CHECKPOINT       10 minutes   VDF proof       │
│  Level 2: VRF_LEADER           1 hour       Leader lottery  │
│  Level 3: DAG_FINALITY         6 hours      PHANTOM order   │
│  Level 4: BITCOIN_ANCHOR       24 hours     BTC hash        │
│  Level 5: EPOCH_BOUNDARY       ~4 years     BTC halving     │
│  Level 6: GENESIS_ROOT         Forever      Network birth   │
└─────────────────────────────────────────────────────────────┘
```

---

4. Network Architecture

4.1 DAG-PHANTOM Ordering

Montana uses DAG-based consensus with PHANTOM ordering for horizontal scaling.

```
┌─────────────────────────────────────────────────────────────┐
│                    DAG BLOCK STRUCTURE                      │
├─────────────────────────────────────────────────────────────┤
│  parent_hashes: List[bytes]  # Multiple parents (up to 10)  │
│  vdf_weight: int             # Cumulative VDF computation   │
│  blue_score: int             # PHANTOM blue set score       │
│  finality_score: float       # 0.0 → 0.99 (irreversible)    │
└─────────────────────────────────────────────────────────────┘
```

Fork Resolution Algorithm: 1. **Blue blocks count** — PHANTOM algorithm determines honest vs attacker blocks 2. **Cumulative VDF weight** — Honest work proves time investment 3. **Lexicographic tip hash** — Deterministic tie-breaker

4.2 Finality State Machine

```
PENDING (0) → TENTATIVE (3+ confirmations)
            → CONFIRMED (6+ confirmations)
            → FINALIZED (score ≥ 0.95)
            → IRREVERSIBLE (score ≥ 0.99)
```

4.3 Block Production

Any node with weight ≥10% of network can produce blocks. Selection uses ECVRF for randomness.

```python
def is_eligible_producer(node, epoch_seed):
    vrf_output = ECVRF_prove(node.private_key, epoch_seed)
    threshold = node.weight * MAX_VRF_VALUE
    return int.from_bytes(vrf_output[:8], 'big') < threshold
```

---

5. The Five Fingers of Adonis

Reputation system using five-dimensional assessment.

5.1 THUMB: TIME (50%)

The dominant factor. Saturates at 210,000 Bitcoin blocks (~4 years). Resets at each halving.

```
def compute_time_score(btc_height):
    blocks_since_halving = btc_height % 210_000
    return min(blocks_since_halving / 210_000, 1.0)
```

Why reset at halving? - Prevents permanent entrenchment of early nodes - Creates natural "election cycles" for network trust - Ties reputation directly to unfakeable Bitcoin time - All nodes restart from 0 — proving continued operation

5.2 INDEX: INTEGRITY (20%)

No protocol violations. Decreases with misbehavior.

```
def compute_integrity_score(node):
    violations = count_violations(node)
    return max(0, 1.0 - violations * 0.1)
```

5.3 MIDDLE: STORAGE (15%)

Percentage of chain history stored.

```
def compute_storage_score(node):
    return node.stored_blocks / total_blocks
```

5.4 RING: EPOCHS (10%)

Bitcoin halvings survived. Unfakeable through time manipulation.

```
def compute_epochs_score(first_btc_height, current_btc_height):
    epochs_survived = (current_btc_height // 210_000) -
        (first_btc_height // 210_000)
    return min(epochs_survived / 4.0, 1.0)  # Saturates at 4
        halvings (16 years)
```

5.5 PINKY: HANDSHAKE (5%)

Mutual trust bonds via the 12 Apostles system.

```
def compute_handshake_score(node):
    weighted_handshakes = sum(
        compute_handshake_value(node.number, partner.number)
        for partner in node.apostles
    )
    return min(weighted_handshakes / 12.0, 1.0)
```

5.6 Total Reputation Score

```
def compute_reputation(node):
    return (
        0.50 * compute_time_score(node) +
        0.20 * compute_integrity_score(node) +
        0.15 * compute_storage_score(node) +
        0.10 * compute_epochs_score(node) +
        0.05 * compute_handshake_score(node)
    )
```

---

6. The Twelve Apostles

Each node chooses exactly 12 trust partners.

6.1 Design Philosophy

```
Trust Manifesto:
Before forming a handshake, ask yourself:

  Do I know this person?
  Not an avatar — a human.

  Do I trust them with my time?
  Willing to lose if they fail?

  Do I wish them longevity?
  Want them here for years?

  If any answer is NO — do not shake.
```

6.2 Why Twelve?

- **Dunbar's inner circle:** Humans maintain ~12-15 close relationships
- **Manageable responsibility:** You can truly know 12 people
- **Game-theoretic limit:** Prevents trust dilution

6.3 Seniority Bonus

Older nodes vouching for newer nodes carries more weight:

```
def compute_handshake_value(my_number, partner_number):
    if partner_number < my_number:
        # Older partner vouching for me
        return 1.0 + log10(my_number / partner_number)
    return 1.0

# Node #1000 shakes #50: value = 1 + log10(1000/50) = 2.30
# Node #1000 shakes #999: value = 1 + log10(1000/999) = 1.00
```

6.4 Collective Slashing

Attack the network, lose your friends:

```
ATTACKER_QUARANTINE_BLOCKS = 180_000  # ~3 years

# Penalties:
# - Attacker: TIME=0, INTEGRITY=0, quarantine
# - Vouchers: -25% integrity (those who vouched for attacker)
# - Associates: -10% integrity (those vouched by attacker)
```

All handshakes dissolved. Trust network damaged. This isn't punishment—it's accountability.

---

7. The Hal Humanity System

Proof of Human, not just Proof of Time.

7.1 The Sybil Problem

TIME proves existence, not uniqueness.

Attack Scenario:

```
Year 0: Attacker creates 12 keypairs
Year 4: All 12 survive halving, EPOCHS=0.25
        Form mutual handshakes (no humanity check)
Result: One person controls full 12-Apostle network
```

7.2 Graduated Trust Model

```
┌─────────────────────────────────────────────────────────────┐
│  TIER 3: TIME-LOCKED (Ultimate - 12 Apostles max)           │
│  ═══════════════════════════════════════════════            │
│                                                             │
│  • Survived 1+ Bitcoin halvings with valid commitment       │
│  • UNFAKEABLE - requires actual time passage                │
│  • Weight: 1.0                                              │
├─────────────────────────────────────────────────────────────┤
│  TIER 2: SOCIAL (Bridge - 6 Apostles max)                   │
│  ═══════════════════════════════════════════                │
│  • Built social graph through handshakes                  │
│  • Sybil cost: real human connections over time           │
│  • Weight: 0.6                                            │
├─────────────────────────────────────────────────────────────┤
│  TIER 1: HARDWARE (Bootstrap — 3 Apostles max)            │
│  ═══════════════════════════════════════════              │
│  • TPM/Secure Enclave/FIDO2 attestation                   │
│  • Sybil cost: physical device ($50-500)                  │
│  • Weight: 0.3                                            │
└─────────────────────────────────────────────────────────────┘
```

7.3 Time-Locked Identity Proofs

Core mechanism anchored to Bitcoin halvings:

```python
# COMMITMENT PHASE (at epoch N)
def create_identity_commitment(pubkey, secret, btc_block_hash):
    commitment_hash = SHA3_256(pubkey + secret + btc_block_hash)
    return IdentityCommitment(
        pubkey=pubkey,
        commitment=commitment_hash,
        epoch=current_epoch,
        btc_height=epoch * 210_000
    )

# PROOF PHASE (at epoch N+1, after halving)
def create_time_locked_proof(commitment, secret,
          current_btc_hash):
    proof = STARK_prove(
        public_inputs=[commitment.commitment, current_btc_hash],
        private_inputs=[secret, commitment.btc_block_hash],
        statement="commitment_valid_and_epoch_passed"
    )
    return TimeLockProof(commitment=commitment, stark_proof=proof)
```

7.4 Sybil Economics

Tier	Sybil Cost per Identity
HARDWARE	$50-500 (physical device)
SOCIAL	Months/years (real connections)
TIME-LOCKED	4+ years (Bitcoin halving)

At Tier 3: 100 fake identities = 400 years of waiting.

7.5 Hardware Attestation (Tier 1)

Three attestation methods:

TPM 2.0:

```
# Quote mechanism proves:
# 1. Device has genuine TPM
# 2. PCR values match expected state
# 3. Montana pubkey is bound via TPM key
```

Apple Secure Enclave:

```
# DeviceCheck/App Attest API proves:
# 1. Device has genuine Secure Enclave
# 2. Attestation key is hardware-bound
# 3. Montana pubkey is bound via enclave key
```

FIDO2/WebAuthn:

```
# FIDO2 authenticator proves:
# 1. Device has genuine FIDO2 authenticator
# 2. User verified via biometric/PIN
# 3. Montana pubkey is bound via credential
```

7.6 Social Graph Verification (Tier 2)

```
class SocialVerifier:
    def verify_social_proof(self, proof):
        # Check minimum connections
        if proof.connection_count < 3:
            return False, "Need 3+ connections"

        # Check clustering coefficient
        cc =
        self.graph.compute_clustering_coefficient(proof.pubkey)
        if cc > 0.8:  # Too clique-like
            return False, "Clustering too high"

        # Check temporal correlation
        tc = self.graph.get_temporal_correlation(proof.pubkey)
        if tc > 0.5:  # Batch created
            return False, "Temporal correlation too high"

        return True, "Social proof valid"
```

7.7 Handshake Integration

Humanity tier limits Apostle count:

```
def can_form_handshake(my_profile, target):
    max_allowed = my_profile.max_apostles  # 3, 6, or 12
    if len(my_apostles) >= max_allowed:
        return False, f"Tier limited to {max_allowed}"

    if target.humanity_score < 0.3:
        return False, "Target humanity score too low"
    return True, "Requirements met"
```

---

8. Anti-Cluster Protection

Defense against coordinated attacks.

8.1 Behavioral Correlation Detection

```
class ClusterDetector:
    def compute_correlation(self, node_a, node_b):
        # Timing correlation (50% weight)
        timing_corr = count_simultaneous() / total_actions

        # Action distribution (30% weight)
        dist_corr = cosine_similarity(actions_a, actions_b)

        # Block height patterns (20% weight)
        height_corr = jaccard_similarity(heights_a, heights_b)

        return 0.5*timing + 0.3*dist + 0.2*height
```

8.2 Global Byzantine Tracker

```
class GlobalByzantineTracker:
    def compute_fingerprint(self, node):
        return {
            'join_time': node.join_timestamp,
            'rep_growth_rate': compute_growth_rate(node),
            'timing_entropy': compute_timing_entropy(node),
            'dimension_balance': compute_balance(node)
        }

    def detect_clusters(self, nodes):
        fingerprints = [self.compute_fingerprint(n) for n in
        nodes]
        clusters = hierarchical_clustering(fingerprints,
        threshold=0.8)
        return clusters
```

8.3 Global Cluster Cap

No cluster can exceed 33% of network influence.

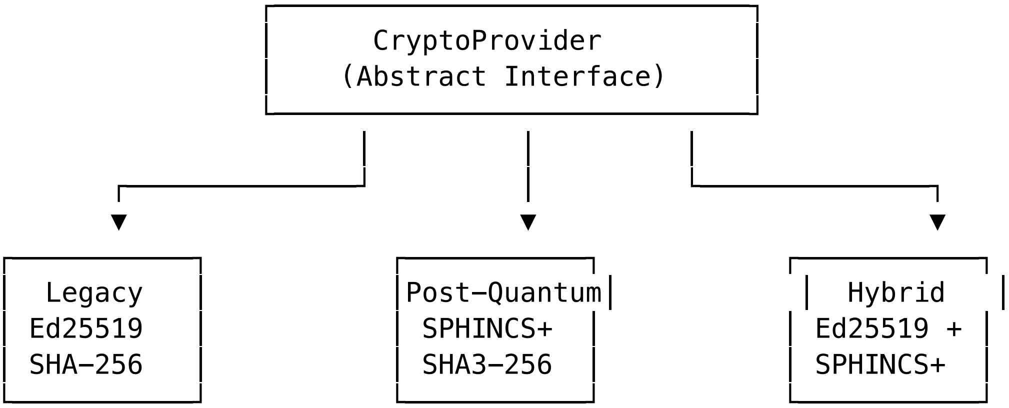
```
MAX_CLUSTER_INFLUENCE = 0.33

def apply_cluster_cap(probabilities, clusters):
    for cluster in clusters:
        cluster_share = sum(prob[n] for n in cluster.members)
        if cluster_share > MAX_CLUSTER_INFLUENCE:
            reduction = MAX_CLUSTER_INFLUENCE / cluster_share
            for node in cluster.members:
                probabilities[node] *= reduction
    return probabilities
```

9. Post-Quantum Cryptography

Complete quantum-resistant cryptographic stack following NIST standards.

9.1 Crypto-Agility Architecture



9.2 Algorithm Selection

Function	Algorithm	Standard	Security
Signatures	SPHINCS+-SHAKE-128f	NIST FIPS 205	128-bit PQ
Hashing	SHA3-256	NIST FIPS 202	128-bit PQ
VDF	SHAKE256 + STARK	—	128-bit PQ
Key Exchange	ML-KEM-768	NIST FIPS 203	128-bit PQ

9.3 Signature Implementation

```
from pantheon.prometheus import pq_crypto

# Generate keypair
public_key, private_key = pq_crypto.sphincs_keygen()

# Sign message
signature = pq_crypto.sphincs_sign(message, private_key)

# Verify signature
valid = pq_crypto.sphincs_verify(message, signature, public_key)
```

10. Attack Resistance

10.1 Attack Vector Matrix

Attack	Difficulty	Mitigation
Flash Takeover	IMPOSSIBLE	210,000 BTC blocks (~4 years) saturation
Slow Takeover	VERY HARD	Behavioral correlation + 33% cluster cap
Sybil via Keypairs	VERY HARD	Hal Humanity System (N × 4 years)
Fake Apostle Network	HARD	Humanity tier limits
Hardware Spoofing	HARD	Multiple attestation sources
Quantum Attack	IMPOSSIBLE	SPHINCS+, SHA3, SHAKE256
Eclipse Attack	BLOCKED	Minimum 8 outbound connections
VPN Spoofing	BLOCKED	ASN + rDNS detection

10.2 51% Attack Requirements

To control majority weighted influence:

Requirement	Threshold
TIME	210,000 BTC blocks (~4 years, resets at halving)
EPOCHS	4+ years (survive 1+ halving)
Humanity	Tier 3 proof (4+ years per identity)
Apostles	12 (requires Tier 3)

10.3 Sybil Attack Cost

Fake Identities	Cost at Tier 3
1	4 years
10	40 years
100	400 years
1000	4000 years

11. Network Protocol

11.1 Message Types

```
Type 0-15:   Standard (VERSION, BLOCK, TX, PING, PONG)
Type 100-102: Noise Protocol handshake
Type 200:    HUMANITY_PROOF
Type 201:    APOSTLE_HANDSHAKE
Type 202:    SLASHING_EVIDENCE
```

11.2 Noise Protocol Encryption

All peer connections use Noise Protocol Framework, XX pattern:

```
from noiseprotocol import NoiseConnection

def establish_connection(peer_pubkey, my_keypair):
    conn =
        NoiseConnection.from_name(b'Noise_XX_25519_ChaChaPoly_SHA256')
    conn.set_keypair_from_private_bytes(Keypair.STATIC,
        my_keypair)
    conn.start_handshake()

    # Three-way handshake
    message_1 = conn.write_message()  # → peer
    conn.read_message(response_1)      # ← peer
    message_2 = conn.write_message()  # → peer

    return conn  # Encrypted channel established
```

11.3 Bitcoin Oracle

Real-time BTC block verification via multiple APIs:

```
BTC_APIS = [
    "https://blockstream.info/api",
    "https://mempool.space/api",
    "https://blockchain.info",
]

def get_btc_block_hash(height):
    results = []
    for api in BTC_APIS:
        try:
            hash = fetch_block_hash(api, height)
            results.append(hash)
        except:
            continue

    # Require 2/3 consensus
    if len(results) >= 2:
        return most_common(results)
    return None
```

11.4 Network Security

Feature	Implementation
Static IP	Only static IPs allowed (no dynamic residential)
VPN Blocking	ASN-based detection of VPN/Tor/Proxy
Eclipse Defense	MIN_OUTBOUND_CONNECTIONS = 8
Rate Limiting	Per-IP and per-subnet throttling
Sybil Protection	Node registration after block validation only

---

12. Privacy

Tiered privacy model.

12.1 Privacy Tiers

Tier	Hidden	Fee Multiplier	Status
T0	Nothing	1×	Production
T1	Receiver	2×	Production
T2	+ Amount	5×	Planned
T3	+ Sender	10×	Planned

12.2 Stealth Addresses (T1)

Each transaction generates a unique one-time address:

```
def generate_stealth_address(recipient_pubkey):
    # Generate ephemeral keypair
    r = secrets.token_bytes(32)
    R = scalar_mult(r, G)

    # Compute shared secret
    shared = SHA3_256(scalar_mult(r, recipient_pubkey))

    # One-time address
    P = recipient_pubkey + scalar_mult(shared, G)

    return P, R  # P = stealth address, R = ephemeral pubkey
```

12.3 Ring Signatures (LSAG)

Linkable Spontaneous Anonymous Group signatures:

```python
def lsag_sign(message, private_key, ring_pubkeys):
    n = len(ring_pubkeys)
    key_image = compute_key_image(private_key)

    # Generate ring signature
    c = [0] * n
    s = [0] * n

    # ... ring signature construction

    return RingSignature(c=c[0], s=s, key_image=key_image)
```

---

13. Emission Schedule

13.1 Supply Parameters

```
Name: Ɉ Montana
Symbol: Ɉ
Ticker: $MONT
Base unit: 1 Ɉ = 1 second
Total supply: 1,260,000,000 Ɉ (21 million minutes)
Halving interval: 210,000 blocks (~4 years)
Full emission: ~132 years
```

13.2 Block Rewards

Era	Block Reward	Cumulative Supply
1	50 minutes	630,000,000 Ɉ
2	25 minutes	945,000,000 Ɉ
3	12.5 minutes	1,102,500,000 Ɉ
4	6.25 minutes	1,181,250,000 Ɉ
…	…	→ 1,260,000,000 Ɉ

13.3 Temporal Compression

Reward ratio converges from 5:1 to 1:1. Inflation asymptotically approaches zero.

```
I(t) = reward(t) / supply(t)
lim(t→∞) I(t) = 0
```

Nash’s Ideal Money realized.

---

14. Implementation

14.1 Repository Structure

```
montana/
├── pantheon/                           # 11 GODS
│   ├── adam/                           # God of Time
│   │   └── adam.py                     # VDF, PoH, 7 temporal levels
│   ├── paul/                           # Network
│   │   ├── network.py                  # P2P, Noise Protocol
│   │   ├── bootstrap.py                # Node discovery
│   │   └── rheuma.py                   # Anti-DDoS
│   ├── hades/                          # Storage
│   │   ├── database.py                 # SQLite backend
│   │   ├── dag.py                      # DAG structure
│   │   └── dag_storage.py              # DAG persistence
│   ├── athena/                         # Consensus
│   │   ├── consensus.py                # DAG ordering, finality
│   │   └── engine.py                   # Unified engine
│   ├── prometheus/                     # Cryptography
│   │   └── pq_crypto.py                # VDF, VRF, SPHINCS+
│   ├── plutus/                         # Wallet
│   │   └── wallet.py                   # Argon2id, AES-256-GCM
│   ├── nyx/                            # Privacy
│   │   ├── privacy.py                  # LSAG, Stealth, Pedersen
│   │   └── tiered_privacy.py           # Transaction builder
│   ├── themis/                         # Validation
│   │   └── structures.py               # Block, Transaction
│   ├── iris/                           # RPC Server
│   │   └── rpc.py                      # JSON-RPC 2.0
│   ├── apostles/                       # 12 Apostles Trust
│   │   └── trust.py                    # Handshakes, seniority
│   └── hal/                            # Humanity
│       ├── reputation.py               # Five Fingers
│       ├── behavioral.py               # Sybil detection
│       ├── slashing.py                 # Penalty manager
│       ├── hardware.py                 # TPM/Enclave/FIDO2
│       ├── social.py                   # Social graph
│       └── timelock.py                 # Time-locked proofs
├── tests/
│   ├── test_integration.py             # 48 integration tests
│   ├── test_dag.py                     # 48 DAG tests
│   ├── test_fuzz.py                    # 27 fuzz tests
│   └── test_security_proofs.py         # Security proofs
└── Montana_v4.3.md                     # This document
```

14.2 Module Summary (11 Gods)

Module	Name	Responsibility
ADAM	God of Time	7 temporal levels, Bitcoin anchor, VDF
PAUL	Network	P2P, Noise Protocol, bootstrap

Module	Name	Responsibility
HADES	Storage	SQLite, DAG persistence
ATHENA	Consensus	DAG ordering, finality
PROMETHEUS	Crypto	VDF, VRF, SPHINCS+, Ed25519
PLUTUS	Wallet	Keys, transactions, encryption
NYX	Privacy	T0/T1, LSAG, stealth addresses
THEMIS	Validation	Block/transaction validation
IRIS	RPC	JSON-RPC 2.0 server
APOSTLES	Trust	12 Apostles, seniority bonus
HAL	Humanity	Reputation, Sybil detection, slashing

14.3 Running a Node

```
pip install pynacl gmpy2 noiseprotocol
python node.py --run
```

14.4 RPC Interface

```
# Get node status
curl -X POST http://localhost:8332 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"getinfo","params":[],"id":1}'

# Get balance
curl -X POST http://localhost:8332 \
  -d '{"jsonrpc":"2.0","method":"getbalance","params":[],"id":1}'

# Send transaction
curl -X POST http://localhost:8332 \
  -d '{"jsonrpc":"2.0","method":"sendtoaddress","params":
        ["<address>",100],"id":1}'
```

---

15. Conclusion

15.1 Security Guarantees

1. **No instant takeover:** TIME resets at each halving — no permanent advantage
2. **Cluster cap:** No coordinated group exceeds 33% influence
3. **Quantum resistance:** Signatures and VDF secure against quantum computers
4. **Sybil resistance:** N fake identities = N × 4 years
5. **Time-locked identity:** Bitcoin halving anchors cannot be faked
6. **Collective accountability:** 12 Apostles + slashing creates real consequences
7. **Bitcoin-anchored time:** 210,000 blocks to saturate, then reset
8. **Clean architecture:** 11 production-ready modules

15.2 Final Statement

Ɉ Montana removes capital as the basis of influence. The system uses: - **Time** — cannot be purchased, accelerated, or concentrated - **Humanity** — cannot be multiplied across Bitcoin halvings

With quantum-resistant cryptography and the Hal Humanity System, these guarantees extend indefinitely into the future.

"Running bitcoin" — *Hal Finney, January 2009*

"Time is priceless. Humanity is sacred. Now both have cryptographic proof."

Ɉ

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] D. Boneh et al., "Verifiable Delay Functions," CRYPTO 2018.

[3] NIST FIPS 202, 203, 205 — Post-Quantum Standards, 2024.

[4] H. Finney, "RPOW - Reusable Proofs of Work," 2004.

[5] R. Dunbar, "How Many Friends Does One Person Need?" 2010.

[6] Y. Sompolinsky, A. Zohar, "PHANTOM: A Scalable BlockDAG Protocol," 2018.

Appendix A: Constants Reference

```
# ============================================================
# REPUTATION WEIGHTS (Five Fingers)
# ============================================================
WEIGHT_TIME = 0.50       # THUMB
WEIGHT_INTEGRITY = 0.20  # INDEX
WEIGHT_STORAGE = 0.15    # MIDDLE
WEIGHT_EPOCHS = 0.10     # RING
WEIGHT_HANDSHAKE = 0.05  # PINKY

# ============================================================
# EPOCHS
# ============================================================
HALVING_INTERVAL = 210_000  # Bitcoin blocks per epoch
MAX_EPOCHS_FOR_SATURATION = 4  # 16 years

# ============================================================
# 12 APOSTLES
# ============================================================
MAX_APOSTLES = 12
MIN_INTEGRITY_FOR_HANDSHAKE = 0.50
HANDSHAKE_COOLDOWN = 86400  # 24 hours

# ============================================================
# HAL HUMANITY SYSTEM
# ============================================================
MAX_APOSTLES_HARDWARE = 3      # Tier 1
MAX_APOSTLES_SOCIAL = 6        # Tier 2
MAX_APOSTLES_TIMELOCKED = 12   # Tier 3

HUMANITY_WEIGHT_HARDWARE = 0.3
HUMANITY_WEIGHT_SOCIAL = 0.6
HUMANITY_WEIGHT_TIMELOCKED = 1.0

HANDSHAKE_MIN_HUMANITY = 0.3

# ============================================================
# SLASHING
# ============================================================
ATTACKER_QUARANTINE_BLOCKS = 180_000  # ~3 years
VOUCHER_INTEGRITY_PENALTY = 0.25      # -25%
ASSOCIATE_INTEGRITY_PENALTY = 0.10    # -10%

# ============================================================
# ANTI-CLUSTER
# ============================================================
MAX_CORRELATION_THRESHOLD = 0.7
MAX_CLUSTER_INFLUENCE = 0.33
MIN_NETWORK_ENTROPY = 0.5

# ============================================================
# DAG CONSENSUS
# ============================================================
MAX_PARENTS = 10
MIN_WEIGHT_THRESHOLD = 0.10  # 10% of network weight to produce blocks
FINALITY_THRESHOLD_TENTATIVE = 3
FINALITY_THRESHOLD_CONFIRMED = 6
FINALITY_SCORE_FINALIZED = 0.95
FINALITY_SCORE_IRREVERSIBLE = 0.99

# ============================================================
# VDF PARAMETERS
# ============================================================
VDF_ITERATIONS = 1_000_000  # ~10 minutes on reference hardware
VDF_CHECKPOINT_INTERVAL = 10_000
VDF_FALLBACK_TO_VRF = True

# ============================================================
# NETWORK
# ============================================================
MIN_OUTBOUND_CONNECTIONS = 8
MAX_INBOUND_CONNECTIONS = 125
DEFAULT_P2P_PORT = 9333
DEFAULT_RPC_PORT = 8332
```

Ɉ Montana Technical Specification v4.3 December 2025