Hearts Currency Technical Specification

Tool Type: Technical Documentation

Primary Users: Developers, system architects, financial technologists

When to Use: When implementing Hearts infrastructure or integrating with existing systems

Estimated Usage Time: Reference material for implementation (varies by application)

Overview

The Hearts Currency Technical Specification provides detailed technical documentation for the Hearts currency system. This comprehensive specification covers the cryptographic foundations, protocol design, data structures, security considerations, and interoperability standards required to implement Hearts and connect it with other systems. The document follows industry-standard technical formatting and includes implementation examples in multiple programming languages.

Key Components

1. Core Protocol Design

Fundamental technical architecture:

- **Consensus Mechanism**: Proof of Care validation with multi-signature verification
- **Block Structure**: Data format and field specifications
- **Transaction Types**: Complete transaction taxonomy with parameters
- **Tokenomics**: Supply management and inflation controls
- **Governance Functions**: On-chain governance implementation

Code Example:

```
// Hearts transaction structure
const heartsTransaction = {
  transactionId: "0x1a2b3c...", // Unique identifier
  timestamp: 1621785600,        // Unix timestamp
  sender: "0xAbCd...",          // Sender wallet address
  recipient: "0x1234...",       // Recipient wallet address
  amount: {
    hearts: 5,                  // Integer Hearts units
    leaves: 25                  // Integer Leaves units (1/100 Heart)
  },
  contributionRef: "0xEf5d...", // Reference to validated care contribution
  validatorSigs: [              // Required validator signatures
    { validator: "0x5678...", signature: "0x9abc..." },
    { validator: "0xdef0...", signature: "0x1234..." }
  ],
  metadata: {                   // Optional contextual information
    category: "CARE_DIRECT",    // Contribution category
    community: "0x2b3c...",     // Community identifier
    notes: "Elder support - transportation assistance"
  }
};
```

2. Cryptographic Implementation

Security and cryptographic specifications:

- **Key Management**: Standards for key generation and storage
- **Signature Scheme**: Elliptic Curve Digital Signature Algorithm (ECDSA) with secp256k1
- **Hash Functions**: SHA-256 for general hashing, Keccak-256 for address derivation
- **Zero-Knowledge Proofs**: zk-SNARK implementation for privacy-preserving validation
- **Quantum Resistance**: Transition roadmap to lattice-based cryptography

Code Example:

```
// Validator signature verification function
function verifyValidatorSignature(transaction, validatorIndex) {
  const validator = transaction.validatorSigs[validatorIndex];
  const message = hashTransaction(transaction);

  // Using elliptic curve signature verification
  return ecVerify(
    message,
    validator.signature,
    validator.validator
  );
}

// Future quantum-resistant implementation (planned for 2028)
function quantumResistantVerify(transaction, validatorIndex) {
  const validator = transaction.validatorSigs[validatorIndex];
  const message = hashTransaction(transaction);

  // Using NIST-approved lattice-based signature verification
  return latticeVerify(
    message,
    validator.signature,
    validator.validator
  );
}
```