# Bitcoin Lending Protocol

## Product Requirements Document

### Phase 2: Bitcoin-Native Custody with Threshold Signatures

**Version:** 1.2
**Date:** January 2026
**Status:** Draft for Review
**Author:** Jamie (Project Lead)
**Builds On:** Phase 1 (Stacks/sBTC Implementation)

---

## Table of Contents

---

## 1. Executive Summary

Phase 2 removes the dependency on sBTC by implementing **Bitcoin-native custody** using threshold signature schemes. This advancement enables the protocol to hold actual Bitcoin directly, eliminating trust in the sBTC peg mechanism while maintaining the oracle-free competitive bidding auctions proven in Phase 1.

### Key Innovation

**Threshold signature custody** where Bitcoin collateral is held in a multi-signature address controlled by a decentralized network of validators. No single entity can access funds, and the protocol operates trustlessly on native Bitcoin without wrapped tokens.

**Phase 2 Objectives**

| Objective | Description |
|---|---|
| **Eliminate sBTC Dependency** | Direct Bitcoin custody removes trust in sBTC peg |
| **Decentralized Validators** | Network of independent validators secure collateral |
| **Threshold Signatures** | M-of-N multisig using Schnorr/FROST or MuSig2 |
| **Maintain Stacks Benefits** | Smart contract logic remains on Stacks for low fees |
| **Backwards Compatible** | Phase 1 loans continue operating; gradual migration |
| **Enhanced Security** | No custodian, no peg risk, pure Bitcoin security |

**Phase 2 Targets**

| Metric | Target |
|---|---|
| **Launch Timeline** | 9 months (Months 8-17 after Phase 1) |
| **Validator Network** | 15+ independent validators at launch |
| **Threshold Scheme** | 10-of-15 multisig (67% threshold) |
| **Migration Volume** | 50%+ of Phase 1 loans migrate to native BTC |
| **New Loan Volume** | $5M+ in native BTC loans by Month 13 |
| **Validator Uptime** | 99.5%+ availability |
| **Total Budget** | $463,000 |

**Strategic Positioning**

Phase 2 positions the protocol as the **first truly trustless Bitcoin lending platform** with: - No wrapped tokens (eliminates WBTC/sBTC trust assumptions) - No centralized custody (eliminates Celsius/BlockFi risk) - No oracles (maintains Phase 1's competitive bidding innovation) - Pure Bitcoin security (leverages Bitcoin's $2T security budget)

This makes the protocol **the gold standard** for Bitcoin-backed lending, setting up Phase 3's multi-chain liquidity integration.

---

## 2. Problem Statement

### 2.1 Limitations of Phase 1 (sBTC Dependency)

While Phase 1 successfully demonstrates oracle-free lending with competitive bidding auctions, it relies on **sBTC** as a Bitcoin representation on Stacks. This introduces several trust assumptions:

### Trust in sBTC Peg Mechanism

- **Current sBTC Design**: Decentralized signers maintain 1:1 peg
- **Trust Assumption**: Signers must be honest and available
- **Risk**: If signers collude or fail, peg could break
- **Scale Limit**: sBTC capacity limited by signer network

### Wrapped Token Risks (Historical Context)

- **WBTC**: Centralized custody by BitGo (single point of failure)
- **renBTC**: Project shut down, holders scrambled to redeem
- **hBTC**: Low liquidity, trust in centralized bridge
- **General Issue**: All wrapped BTC requires trusting custodians

### Market Perception

- Users prefer **actual Bitcoin** over representations
- "Not your keys, not your coins" applies to wrapped tokens
- Institutional users require custody transparency
- Maximum TVL limited by comfort with sBTC peg

## 2.2 The Native Bitcoin Challenge

Holding actual Bitcoin in a trustless protocol is technically challenging:

**Bitcoin's Limited Scripting**: - No Turing-complete smart contracts - Limited programmability compared to Ethereum/Stacks - Multisig requires on-chain transaction for setup - Complex business logic not feasible on Bitcoin L1

**The Dilemma**: - Smart contracts on Stacks (flexible, low-cost, programmable) - Bitcoin custody on Bitcoin (trustless, secure, native) - Need: **Bridge these two worlds without centralization**

## 2.3 Existing Solutions Are Insufficient

**BitVM**: - Theoretical framework for Bitcoin computation - Not production-ready (highly experimental) - Requires optimistic verification (complex) - No mature implementation available

**Federated Sidechains** (Liquid, RSK): - Require trusting federation of signers - Not meaningfully more decentralized than sBTC - Add another layer of abstraction

**Lightning Network**: - Designed for payments, not lending collateral - Requires channels and liquidity management - Not suitable for long-term collateral locks

**What We Need**:  Direct Bitcoin custody (no wrapped tokens)
 Decentralized validator network (no federation trust)

Threshold signatures (no single point of failure)
Integration with Stacks (leverage smart contract logic)
Production-ready today (not experimental)

---

## 3. Solution Overview

### 3.1 Architecture: Hybrid Bitcoin + Stacks

Phase 2 uses a **hybrid architecture** that leverages the strengths of both chains:

**Bitcoin Layer** (Custody): - Holds actual BTC collateral in threshold multisig addresses - Provides ultimate security through Bitcoin's consensus - Uses Taproot/Schnorr for efficient multisig - No trust assumptions beyond Bitcoin's security model

**Stacks Layer** (Logic): - Maintains smart contract logic for auctions, loans, NFTs - Coordinates validator actions through consensus - Provides user interface and state management - Keeps transaction costs low (~$1-5 vs Bitcoin's $10-50)

**Validator Network** (Bridge): - Decentralized network of 15+ independent operators - Run both Bitcoin and Stacks nodes - Hold threshold signature key shares - Sign Bitcoin transactions based on Stacks contract state

### 3.2 Threshold Signature Scheme

**Technology**: FROST (Flexible Round-Optimized Schnorr Threshold) or MuSig2

**Configuration**: 10-of-15 multisig - 15 total validators - 10 signatures required (67% threshold) - 5 validators can be offline without stopping operations - 6+ colluding validators needed to steal (highly unlikely with good incentives)

**Key Generation** (Distributed):

```
No single party ever has full private key!

Each validator i generates:
- Secret share: s_i (kept private, never shared)
- Public key share: P_i = s_i * G

Combined public key:
- P = P_1 + P_2 + ... + P_15

Bitcoin Address:
- Taproot address derived from P
- Looks like any normal Bitcoin address
- No on-chain indication of multisig (privacy!)
```

**Signature Generation** (Threshold):

```
To spend Bitcoin:
1. Stacks contract reaches consensus on action
2. 10+ validators see the consensus
3. Each validator creates partial signature
4. Coordinator aggregates 10+ partial signatures
5. Combined signature is valid for public key P
6. Broadcast to Bitcoin network
```

### 3.3 Loan Lifecycle with Native Bitcoin

**1. Borrower Creates Loan Request**:

```
User → Stacks Contract
- Specify: loan amount, max repayment, duration
- Contract generates unique Bitcoin address (threshold multisig)
- Borrower sends BTC to this address
- Bitcoin tx confirmed → Stacks contract activated
```

**2. Auction & Bidding** (Unchanged from Phase 1):

```
- Competitive bidding auction on Stacks
- Lenders place bids (total repayment amounts)
- Lowest bid wins when auction ends
- All logic on Stacks (cheap, fast)
```

**3. Loan Finalization**:

```
Stacks Contract → Validators
- Contract finalizes auction with winning bid
- Winning lender provides Stablecoin on Stacks
- Contract signals validators to release BTC
- 10+ validators sign Bitcoin transaction
- BTC transfers to borrower
- Lender receives NFT position
```

**4. Repayment**:

```
Borrower → Stacks Contract
- Borrower repays stablecoin on Stacks (fixed amount from auction)
- Contract signals validators to release collateral
- 10+ validators sign Bitcoin transaction
- BTC returns to borrower
- Both NFTs burned
```

**5. Default Scenario**:

```
If borrower doesn't repay by maturity:
- Stacks contract enters "defaulted" state
- Lender can claim collateral
- Contract signals validators
- 10+ validators sign Bitcoin transaction
- BTC transfers to lender
```

### 3.4 Validator Network Design

**Validator Roles  Responsibilities**: - Monitor both Bitcoin and Stacks blockchains - Maintain synchronized state view - Participate in threshold signing when required - Monitor other validators for liveness - Submit signed transactions to Bitcoin network

**Requirements**: - Run full Bitcoin node (verify all Bitcoin state) - Run full Stacks node (verify all Stacks state) - Maintain >99% uptime - Secure key management (HSM recommended) - Network connectivity (low latency) - Stake/bond collateral (economic security)

**Incentives**: - Earn fees: 0.1% of loan volume processed - Distributed proportionally to all 15 validators - Slashing for downtime or misbehavior - Reputation system for reliable validators

**Validator Selection & Rotation  Phase 2 Launch** (Months 7-13): - **Permissioned set**: 15 curated validators - Selection criteria: reputation, uptime history, geographic diversity - Includes: protocol team (3), community members (5), partners (7) - Fixed set for initial 6 months (stability focus)

**Future Phase** (Post-Phase 2): - **Permissionless entry**: Anyone can apply to be validator - Stake requirement: e.g., $50,000-100,000 in STX or BTC - Rotation mechanism: validators can be voted in/out - Governance: STX holders or future DAO

**Geographic Distribution**  Target distribution for censorship resistance: - North America: 5 validators (33%) - Europe: 5 validators (33%) - Asia: 3 validators (20%) - Other: 2 validators (14%)

No more than 3 validators in same jurisdiction.

### 3.5 Security Properties

**Trustlessness**:  No single validator can steal funds (need 10/15)
No custodian (pure threshold cryptography)
No federation (open validator set long-term)
No wrapped token (actual Bitcoin)

**Liveness**:  5 validators can be offline without issue
System continues with 10+ online validators
Graceful degradation (not binary failure)
Automatic failover if validators drop

**Censorship Resistance**: Geographic distribution prevents single-jurisdiction attack
Multiple validator entities (individuals, companies, DAOs)
6+ colluding validators needed to censor (highly unlikely)

**Attack Scenarios**:

| Attack | Difficulty | Mitigation |
|---|---|---|
| Steal funds | Need to compromise 10/15 validators | High threshold, diversity, monitoring |
| Censor transaction | Need to control 6/15 validators | Geographic distribution, incentives |
| Network partition | Need to split validators | Multiple network paths, redundancy |
| Key extraction | Need to compromise validator HSMs | Hardware security, secure boot |
| Social engineering | Need to trick 10/15 operators | Strict procedures, multi-approval |

### 3.6 Benefits vs Phase 1

| Aspect | Phase 1 (sBTC) | Phase 2 (Native BTC) |
|---|---|---|
| **Collateral** | sBTC (wrapped) | Bitcoin (native) |
| **Trust Assumptions** | sBTC signers | Validator threshold |
| **Peg Risk** | Yes (sBTC peg) | No (actual BTC) |
| **Bitcoin Security** | Indirect | Direct |
| **Market Perception** | "Wrapped token" | "Real Bitcoin" |
| **TVL Ceiling** | Limited by sBTC supply | Limited by validator capacity |
| **User Confidence** | Medium-High | Very High |
| **Custody Transparency** | sBTC multisig | Observable threshold |

**Why This Matters**: - Institutional users prefer actual Bitcoin custody - Eliminates entire category of risk (peg breaks) - No dependency on sBTC project success - Scales independently - More credible for large loans ($500K+)

---

## 4. User Personas

### 4.1 Primary Persona: Large-Scale Bitcoin Miners (Borrowers)

Phase 2 targets **larger mining operations** that Phase 1's sBTC might not accommodate:

#### Demographics

- **Operation size**: 100-5,000 ASICs (medium to large miners)
- **Geography**: USA (30%), Canada (20%), Kazakhstan (15%), Russia (10%), Norway (10%), Other (15%)
- **Technical level**: Very high - infrastructure operators
- **Bitcoin philosophy**: Bitcoin maximalists, only use BTC
- **Monthly revenue**: $100,000 - $5,000,000+

#### Needs and Goals

- **Primary**: Large working capital loans ($100K-$2M)
- **Secondary**: Trustless custody (no wrapped tokens acceptable)
- **Tertiary**: Institutional-grade security for large amounts
- **Financial**: Competitive rates for large loans (5-8% APR)

#### Pain Points with Phase 1

- sBTC supply might be insufficient for large loans
- Uncomfortable with wrapped token for large amounts
- Prefer direct Bitcoin custody
- Institutional policies may prohibit wrapped assets
- Higher perceived risk with sBTC peg for multi-million dollar amounts

#### User Story

> *"I run a 1,000 ASIC operation in Texas. My monthly electricity bill is $300,000. When BTC dropped from $100K to $85K, I needed $1M to cover operations without selling my mined Bitcoin.*
>
> *Phase 1 with sBTC was interesting, but for $1M+ I need actual Bitcoin custody. My CFO won't approve wrapped tokens for that size. We need to see our BTC locked in an on-chain multisig address.*
>
> *With Phase 2's threshold custody, I can borrow $1M against 15 BTC collateral. I can verify the multisig address on Bitcoin. I can see the validator signatures. That's institutional-grade security.*
>
> *I'm willing to pay 6-7% APR if it means trustless custody of actual Bitcoin."*

**— Marcus, 42, Mining Operation Director, Houston TX**

**Success Metrics for This Persona**

- 20+ large miners using Phase 2 by Month 13
- Average loan size: \$250,000 - \$1,000,000
- Total volume: \$5M+ from large miners
- 80%+ of these miners prefer Phase 2 over Phase 1
- Repeat borrowing rate: >60%

---

### 4.2 Primary Persona: Institutional Lenders (Liquidity Providers)

**Demographics**

- **Profile**: Family offices, crypto hedge funds, high net worth individuals
- **Experience level**: Advanced - understand Bitcoin/DeFi deeply
- **Capital**: \$500,000 to \$50M+
- **Investment thesis**: Conservative crypto yield with Bitcoin collateral
- **Risk tolerance**: Low to medium - want best security

**Needs and Goals**

- **Primary**: Institutional-grade custody transparency
- **Secondary**: Large position sizes (\$100K-\$1M per loan)
- **Tertiary**: Verifiable Bitcoin custody (not wrapped tokens)
- **Trust**: Proof that collateral is in decentralized threshold custody

**Pain Points with Phase 1**

- sBTC limits confidence for large positions
- Wrapped tokens not acceptable to compliance teams
- Peg risk unacceptable for multi-million portfolios
- Want to see actual Bitcoin multisig on-chain

**User Story**

> *"I manage a \$20M family office crypto portfolio. We're interested in Bitcoin-collateralized lending at 7-9% APR - much better than tradfi bonds.*
>
> *But we won't touch wrapped Bitcoin. We've seen WBTC centralization, we've seen renBTC shut down. For us, it's actual Bitcoin or nothing.*
>
> *Phase 2's threshold custody is exactly what we need. We can verify the Bitcoin multisig address. We can see the validator signatures on every transaction. We can audit the whole process.*
>
> *We're prepared to deploy \$2-5M into Phase 2 if security is rock-solid."*

— **Elizabeth, 55, Family Office Manager, Singapore**

**Success Metrics for This Persona**

- 10+ institutional lenders in Phase 2
- Average position size: $200,000+
- Total institutional capital: $3M+
- 90%+ cite "native Bitcoin custody" as key reason
- Zero security incidents

---

### 4.3 Secondary Persona: Bitcoin Maximalists (Both Roles)

**Demographics**

- **Profile**: Hardcore Bitcoin believers
- **Philosophy**: Bitcoin only, no altcoins, no wrapped tokens
- **Technical level**: Very high - run own nodes
- **Values**: Decentralization, trustlessness, censorship-resistance

**Needs and Goals**

- **Primary**: Pure Bitcoin solution (no compromises)
- **Secondary**: Verifiable decentralization (no trust required)
- **Tertiary**: Support Bitcoin-native infrastructure

**Why Phase 2 Appeals to Them**

- Actual Bitcoin (not wrapped)
- Threshold custody (no custodian)
- Decentralized validators (no single point of control)
- Verifiable on Bitcoin blockchain
- Aligns with Bitcoin ethos

**User Story**

> *"I don't touch wrapped Bitcoin. I don't touch altcoins. Bitcoin is the only real money.*
>
> *Phase 1 with sBTC? That's an altcoin token. Not interested.*
>
> *Phase 2 with threshold signatures holding actual Bitcoin? Now that's interesting. That's how Bitcoin lending should work. Pure Bitcoin security."*

— **Max, 38, Bitcoin Developer, Remote**

---

## 5. Feature Requirements

### 5.1 Threshold Custody Features (Must Have - P0)

**FR1.1: Distributed Key Generation (DKG)** **Description**: Generate threshold multisig keys without any single party knowing the full private key.

**Functional Requirements**: - 15 validators participate in DKG ceremony - Each validator generates secret share (never leaves their system) - Combined public key computed from all shares - No validator ever has full private key - Verifiable Secret Sharing (VSS) ensures correctness - Public key shares published on-chain for transparency

**DKG Process**:

```
Round 1: Commitment Phase
- Each validator commits to their secret polynomial
- Commitments published and verified

Round 2: Share Distribution
- Each validator sends encrypted shares to others
- Shares transmitted over secure channels

Round 3: Verification
- Each validator verifies received shares
- Complaints raised if shares invalid

Round 4: Finalization
- If no complaints: DKG succeeds
- Combined public key computed
- Key shares ready for threshold signing
```

**Acceptance Criteria**: - [ ] DKG ceremony completes with 15 validators - [ ] Combined public key is Taproot-compatible - [ ] Each validator can prove their key share is valid - [ ] Full private key is never reconstructed - [ ] Process completes in <10 minutes - [ ] Ceremony can be audited on-chain

**Security Properties**: - No single validator learns full key - No subset <10 validators can reconstruct key - Malicious validators detected during verification - Byzantine fault tolerant (up to 5 malicious validators)

---

**FR1.2: Threshold Signature Generation** **Description**: Generate valid Bitcoin signatures using threshold cryptography (10-of-15).

**Functional Requirements**: - Any 10+ validators can create valid signature - Partial signatures combined into final signature - Final signature indistinguishable from single-key signature (Taproot privacy) - No interactive rounds needed (non-interactive threshold signing) - Signature valid for Bitcoin consensus rules

**Signing Process**:

```
Step 1: Request Initiated
- Stacks contract determines action needed (e.g., release collateral)
- Contract publishes signing request with:
  - Transaction details (outputs, amounts, fees)
  - Verification proof (e.g., loan was repaid)
  - Nonce for this signing session

Step 2: Validators Verify
- Each validator independently verifies:
  - Stacks contract state is correct
  - Action is authorized per contract rules
  - Bitcoin transaction matches contract intent
  - No double-signing or replay

Step 3: Partial Signatures
- Each validator that agrees creates partial signature
- Partial signature based on their key share
- Partial signatures published to coordinator

Step 4: Aggregation
- Once 10+ partial signatures collected
- Coordinator aggregates into final signature
- Final signature is standard Schnorr signature

Step 5: Broadcast
- Bitcoin transaction with final signature broadcast
- Confirms in Bitcoin block
- Stacks contract updated with confirmation
```

**Acceptance Criteria**: - [ ] 10 validator signatures sufficient to sign - [ ] Signing completes in <5 minutes typically - [ ] Failed signing doesn't lock system (retry possible) - [ ] Signature passes Bitcoin consensus validation - [ ] Process works even with 5 validators offline - [ ] All partial signatures logged for audit

**Edge Cases**: - Exactly 10 validators online → Should still work - 11+ validators but disagree on action → Majority rules - Validators sign conflicting transactions → First-seen-valid wins - Network partition splits validators → Larger partition continues

---

**FR1.3: Bitcoin Transaction Construction** **Description**: Build valid Bitcoin transactions that validators will sign.

**Functional Requirements**: - Parse Stacks contract state to determine Bitcoin action - Construct appropriate Bitcoin transaction: - Collateral lock (P2TR output to threshold address) - Collateral release (spend from threshold address to borrower) - Collateral claim (spend from threshold address to lender on default) - Calculate appropriate fees (dynamic fee estimation) - Handle UTXO management (coin selection) - Support RBF (Replace-By-Fee) for stuck transactions

**Transaction Types**:

**Type 1: Collateral Lock** (Borrower → Threshold Address)

```
Inputs:
  - Borrower's UTXOs (sufficient BTC + fees)

Outputs:
  - Threshold address: [collateral_amount]
  - Change address: [remaining if any]

Note: Borrower signs this themselves (not threshold)
```

**Type 2: Collateral Release** (Threshold Address → Borrower)

```
Inputs:
  - Threshold address UTXO: [collateral_amount]

Outputs:
  - Borrower address: [collateral_amount - fee]

Signature: Threshold signature (10-of-15 validators)
Trigger: Stacks contract confirms repayment received
```

**Type 3: Collateral Claim** (Threshold Address → Lender)

```
Inputs:
  - Threshold address UTXO: [collateral_amount]

Outputs:
  - Lender address: [collateral_amount - fee]

Signature: Threshold signature (10-of-15 validators)
Trigger: Stacks contract confirms default (maturity passed, no repayment)
```

**Fee Management**: - Query mempool for current fee rates - Target 1-2 block confirmation (medium priority) - Allow manual fee bumping if transaction stuck - Reserve small amount of BTC for fee buffer

**Acceptance Criteria**: - [ ] Transactions are valid per Bitcoin consensus - [ ] Fees are competitive (not overpaying) - [ ] All outputs are correct addresses and amounts - [ ] Change handling works correctly - [ ] RBF enabled for all threshold-signed transactions - [ ] Transaction monitoring until confirmation

---

**FR1.4: Validator Monitoring & Coordination** **Description**: Monitor validator health and coordinate threshold signing sessions.

**Functional Requirements**: - Real-time monitoring of all 15 validators: - Online/offline status - Last heartbeat timestamp - Bitcoin node sync status - Stacks node sync status - Partial signature latency - Coordinator selects best validators for signing - Automatic failover if validators drop during signing - Alert system for validator issues - Dashboard showing validator network health

**Monitoring Metrics**:

| Metric | Green | Yellow | Red |
|---|---|---|---|
| Uptime | >99% | 95-99% | <95% |
| Response Time | <30s | 30-60s | >60s |
| Bitcoin Sync | <1 block behind | 1-5 blocks | >5 blocks |
| Stacks Sync | <1 block behind | 1-5 blocks | >5 blocks |
| Signing Success | >99% | 95-99% | <95% |

**Coordinator Role**: - Aggregates partial signatures - Requests retries if some validators timeout - Publishes final signature to Bitcoin network - Updates Stacks contract with Bitcoin tx confirmation - Not trusted (anyone can verify signatures are correct)

**Acceptance Criteria**: - [ ] All validators monitored in real-time - [ ] Dashboard shows current network health - [ ] Alerts fire for validator issues (email, Discord) - [ ] Coordinator successfully aggregates signatures - [ ] System continues with 10+ validators online - [ ] Graceful degradation if validators drop

---

### 5.2 Smart Contract Integration (Must Have - P0)

**FR2.1: Bitcoin Event Monitoring** **Description**: Stacks contracts must monitor Bitcoin blockchain for collateral deposits and confirmations.

**Functional Requirements**: - Stacks contract listens for Bitcoin transactions - Detects deposits to threshold addresses - Requires 3+ Bitcoin confirmations before activating loan - Updates contract state when Bitcoin events occur - Handles Bitcoin reorgs gracefully

**Bitcoin → Stacks Flow**:

```
1. Borrower sends BTC to threshold address
2. Bitcoin transaction confirms (1 block)
3. Validators observe transaction
4. Validators submit proof to Stacks contract
5. After 3 confirmations, contract activates loan
6. Auction begins on Stacks
```

**Proof Structure**:

```
Bitcoin Transaction Proof {
  txid: Bitcoin transaction ID
  block_height: Block containing transaction
  merkle_proof: Proof transaction is in block
  outputs: List of outputs with amounts
  confirmations: Number of confirmations
}
```

**Acceptance Criteria**: - [ ] Contract correctly parses Bitcoin transaction data - [ ] 3+ confirmations required before activation - [ ] Reorgs handled (contract waits for finality) - [ ] Invalid proofs rejected - [ ] Multiple validators can submit same proof (deduplicated)

---

**FR2.2: Validator Action Requests** **Description**: Stacks contract publishes requests for validators to sign Bitcoin transactions.

**Functional Requirements**: - Contract emits "action required" events: - `release-collateral` (on successful repayment) - `claim-collateral` (on default) - `refund-collateral` (if auction fails) - Events include all necessary transaction details - Validators automatically respond to valid requests - Multiple validators verify before signing - Prevents unauthorized Bitcoin spends

**Action Request Format**:

```
{
  action: "release-collateral" | "claim-collateral" | "refund-collateral",
  loan-id: uint,
  bitcoin-txid: (buff 32),          ;; Collateral UTXO
  recipient: (string-ascii 62),      ;; Bitcoin address
  amount: uint,                      ;; Satoshis
  fee: uint,                         ;; Satoshis for miner fee
  nonce: uint                        ;; Unique per request
}
```

**Validator Verification Checklist**:

```
Before signing, each validator checks:
 Stacks contract state matches request
 Action is authorized per contract rules
 Recipient address matches loan participant
 Amount matches contract records
 No duplicate/replay request (nonce check)
 Bitcoin UTXO actually exists and is spendable
```

**Acceptance Criteria**: - [ ] Events correctly formatted and complete - [ ] Validators receive events in real-time (<10s) - [ ] All validators independently verify before signing - [ ] Malicious requests rejected by honest validators - [ ] Partial signature aggregation works correctly - [ ] Bitcoin transaction broadcasts after 10+ signatures

---

**FR2.3: Cross-Chain State Synchronization** **Description**: Keep Stacks and Bitcoin states synchronized despite different block times.

**Functional Requirements**: - Bitcoin block time: ~10 minutes (variable) - Stacks block time: ~10 minutes (anchored to Bitcoin) - Handle timing differences gracefully - Maintain consistency even during reorgs - Provide UI with accurate status across both chains

**State Machine**:

```
Loan Status Flow (spans both chains):

1. "Created" (Stacks) → Waiting for BTC deposit
2. "Collateral-Pending" (Bitcoin tx seen, <3 confirmations)
3. "Auction" (Bitcoin confirmed, auction active on Stacks)
4. "Active" (Auction finalized on Stacks)
5a. "Repay-Pending" (Repayment on Stacks, BTC release initiated)
5b. "Released" (BTC release confirmed on Bitcoin)
6. "Completed" (Both chains finalized)

OR

5a. "Default" (Maturity passed on Stacks)
5b. "Claim-Pending" (BTC claim initiated)
5c. "Claimed" (BTC claimed confirmed on Bitcoin)
6. "Defaulted" (Both chains finalized)
```

**Acceptance Criteria**: - [ ] State transitions happen correctly across chains - [ ] No state desync even during reorgs - [ ] UI reflects current state accurately - [ ] Timing edge cases handled (e.g., Stacks faster than Bitcoin) - [ ] Recovery possible if synchronization fails

---

### 5.3 User Interface Updates (Must Have - P0)

**FR3.1: Native Bitcoin Deposit Flow** **Description**: Users must be able to lock actual Bitcoin as collateral.

**User Flow**:

```
1. User creates loan request on Stacks
```

```
2. UI generates unique threshold multisig Bitcoin address
3. UI displays address as:
   - Text (copyable)
   - QR code (scannable)
   - Verification info (multisig details)
4. User sends BTC from their wallet
5. UI monitors Bitcoin mempool for transaction
6. Shows "Pending confirmation" (0-3 blocks)
7. After 3 confirmations: "Collateral locked, auction starting"
```

**Address Display**:

```
Send Bitcoin Collateral


Amount: 1.5 BTC

[QR CODE]

bc1p8xj2f7a...k3m9tn2 [Copy]

   Do NOT send from exchange!
    You must control the address

   This is a 10-of-15 multisig address
    View validators →

Status: Waiting for your deposit...
[View on Bitcoin Explorer]
```

**Acceptance Criteria**: - [ ] Address displayed clearly (text + QR) - [ ] Address is valid Bitcoin address (bech32) - [ ] Clear instructions to avoid exchange deposits - [ ] Real-time monitoring of Bitcoin mempool - [ ] Progress indicator (0/3, 1/3, 2/3, 3/3 confirmations) - [ ] Link to Bitcoin explorer for transparency

---

**FR3.2: Validator Transparency Dashboard** **Description**: Users must be able to verify the decentralization and security of the validator network.

**Dashboard Sections**:

1. **Validator List**:

```
Active Validators (15 total)
```

```
 ID         Location   Uptime     Last      Status
                                  Seen

 Val-01     USA        99.8%      5s ago     Up
 Val-02     Germany    99.9%      3s ago     Up
 Val-03     Japan      99.7%      8s ago     Up
 Val-04     Canada     99.6%      15s ago    Up
 Val-05     UK         97.2%      2m ago     Slow
 ...        ...        ...        ...       ...
```

**2. Geographic Distribution Map**: - World map showing validator locations - Color-coded by status (green/yellow/red) - Hover for details (name, uptime, last activity)

**3. Network Health**:

```
Overall Status:  Healthy

Active Validators: 14/15 (93%)
Signing Threshold: 10 required
Current Capacity: 14 available (140% of minimum)
Average Response Time: 12 seconds
Last Signature: 3 minutes ago (Loan #157)
```

**4. Recent Signatures**:

```
  Recent Threshold Signatures

 Loan    Action     Sigs      Bitcoin Tx

 #157    Release    12/15     3a8f2c...
 #155    Release    11/15     7b3d91...
 #154    Claim      13/15     2f8a34...
 #152    Release    10/15     8c2f74...
```

**Acceptance Criteria**: - [ ] All 15 validators listed with current status - [ ] Geographic map shows distribution - [ ] Real-time updates (<30 second refresh) - [ ] Historical uptime data available - [ ] Links to validator profiles (if public) - [ ] Clear explanation of threshold security model

---

**FR3.3: Bitcoin Transaction Monitoring** **Description**: Users must be able to track Bitcoin transactions related to their loans.

**Loan Detail View**:

```
  Loan #42 - Bitcoin Transactions


  Collateral Deposit
  1.5 BTC → bc1p8xj2f7a...k3m9tn2
  Txid: 3a8f2c91...
  Confirmations: 87
  [View on Bitcoin Explorer]



  Collateral Release (pending)
  1.5 BTC → bc1qxy8w2n3...r5t7m9
  Status: Waiting for 10 validator signatures
  Current: 8/10 signatures received
  ETA: ~5 minutes
```

**Acceptance Criteria**: - [ ] All Bitcoin transactions displayed clearly - [ ] Confirmation count shown and updated - [ ] Links to Bitcoin explorer for verification - [ ] Threshold signature progress visible - [ ] Clear status indicators (pending, confirmed, failed)

---

### 5.4 Migration Features (Should Have - P1)

**FR4.1: Phase 1 → Phase 2 Loan Migration** **Description**: Allow existing Phase 1 (sBTC) loans to migrate to Phase 2 (native BTC).

**Functional Requirements**: - User can opt-in to migrate active loan - Process: 1. Repay Phase 1 loan (get sBTC back) 2. Convert sBTC → BTC (via sBTC redemption) 3. Deposit BTC to Phase 2 threshold address 4. Create equivalent Phase 2 loan - Preserve favorable terms from Phase 1 if possible - Migration window: Months 7-13 (during Phase 2 rollout)

**Migration UI**:

```
Your Phase 1 Loan: #123
Collateral: 1.5 sBTC
Status: Active


  Migrate to Phase 2?

  Benefits:
     Native Bitcoin custody
     Eliminated sBTC peg risk
     Institutional-grade security

   Process:
   1. Repay current loan (automatic)
   2. Convert sBTC to BTC
   3. Deposit to Phase 2

   [Migrate Now] [Learn More]
```

**Acceptance Criteria**: - [ ] Migration process is seamless (1-click if possible) - [ ] Clear explanation of benefits - [ ] No loss of value during migration - [ ] Migration tracking in UI - [ ] Support and documentation available

---

## 6. Architecture Requirements

### 6.1 Validator Node Requirements

**Hardware**: - CPU: 8+ cores (Intel/AMD x86_64) - RAM: 32GB minimum (64GB recommended) - Storage: 2TB NVMe SSD (Bitcoin full node ~600GB, Stacks ~100GB, growth room) - Network: 100 Mbps+ bandwidth, <50ms latency to other validators - Uptime: >99.5% target

**Software**: - Bitcoin Core v25.0+ (full node, txindex enabled) - Stacks Node v2.5+ (full node) - Validator software (custom, written in Rust) - HSM support (optional but recommended): YubiHSM 2, Ledger, etc.

**Security**: - Key material stored in HSM or secure enclave - Firewall: Only p2p ports open (not RPC) - Regular security updates - Intrusion detection system (IDS) - Encrypted communications between validators

**Monitoring**: - Prometheus metrics exported - Grafana dashboards for node health - Alert manager for issues - Logging aggregation (e.g., ELK stack)

---

### 6.2 Threshold Cryptography Stack

**Library**: Secp256k1-zkp (Schnorr threshold)

**Alternative**: FROST Implementation (production-ready)

**Key Features Needed**: - Schnorr signature support (Bitcoin Taproot) - Threshold signature generation (M-of-N) - Distributed key generation (DKG) - Verifiable secret sharing (VSS) - Non-interactive signing (no interactive rounds)

**Integration Points**: - Rust validator software calls library - Bitcoin transaction signing pipeline - Key share management (encrypted storage) - Partial signature aggregation

---

### 6.3 Network Communication

**Validator-to-Validator**: - Protocol: libp2p (peer-to-peer networking) - Transport: TCP with TLS encryption - Gossip protocol for partial signatures - DHT for peer discovery - NAT traversal support

**Validator-to-Coordinator**: - REST API for submitting partial signatures - WebSocket for real-time updates - Authentication via validator public keys - Rate limiting per validator

**Coordinator-to-Bitcoin**: - Bitcoin Core RPC API - Transaction broadcasting via multiple nodes - Mempool monitoring - Fee estimation queries

---

### 6.4 Disaster Recovery

**Validator Failure Scenarios**:

**Scenario 1: Single Validator Down** - Impact: None (need 10/15, have 14 remaining) - Response: Monitor, contact operator, wait for recovery - Threshold: Alert if down >24 hours

**Scenario 2: 5 Validators Down** - Impact: Minimal (still have 10/15) - Response: Urgent investigation, activate backup validators - Threshold: P1 alert, all hands on deck

**Scenario 3: 6+ Validators Down** - Impact: CRITICAL (below 10/15 threshold) - Response: Emergency protocol: - Pause new loans immediately - Prioritize bringing validators back online - Activate backup validators - Consider emergency validator recruitment - Allow existing loans to complete without new signatures - Threshold: P0 alert, 24/7 response

**Key Recovery**: - Each validator backs up encrypted key share - Recovery requires validator identity proof - No single backup location (distributed) - Regular backup testing (quarterly)

**Network Split**: - Monitor for network partitions - Larger partition (8+ validators) continues - Smaller partition waits for reconnection - No conflicting signatures across partitions

---

## 7. Non-Functional Requirements

### 7.1 Security (Critical - P0)

**NFR1.1: Threshold Security** **Requirements**: - 10-of-15 threshold enforced cryptographically - No single validator can steal funds - No subset <10 validators can reconstruct key - Key generation ceremony auditable - All signatures verified before Bitcoin broadcast

**Acceptance Criteria**: - [ ] Cryptographic proof that threshold holds - [ ] Independent security audit of threshold implementation - [ ] Penetration testing of validator network - [ ] Bug bounty program ($100K+ rewards) - [ ] No critical vulnerabilities in audit

---

**NFR1.2: Validator Security** **Requirements**: - Validator keys stored in HSMs (hardware security modules) - No remote access to validator key material - Multi-factor authentication for validator access - Regular security updates within 48 hours - Intrusion detection on all validator nodes

**Acceptance Criteria**: - [ ] 100% of validators use HSMs or secure enclaves - [ ] Security audits pass for all validator setups - [ ] No successful attacks on validator keys - [ ] Incident response plan tested quarterly

---

### 7.2 Reliability (High Priority - P0)

**NFR2.1: Validator Uptime** **Requirements**: - Individual validator uptime: >99.5% - Network capacity: Always >10 validators online - Maximum signing latency: <5 minutes (typical: <2 minutes) - Zero downtime for user-facing operations

**Monitoring**: - Real-time validator health checks - Alert if <12 validators online - Automated failover to backup validators - Geographic redundancy

**Acceptance Criteria**: - [ ] Network uptime: 99.9%+ over 6 months - [ ] No periods with <10 validators - [ ] Average signing time <3 minutes - [ ] Zero failed signatures due to validator unavailability

---

**NFR2.2: Bitcoin Transaction Reliability** **Requirements**: - 100% of legitimate transactions must eventually confirm - Failed transactions must be retryable - RBF (Replace-By-Fee) for stuck transactions - Fee estimation accuracy: ±20% of optimal

**Acceptance Criteria**: - [ ] <0.1% transaction failure rate - [ ] All failures are recoverable - [ ] Median confirmation time: <30 minutes - [ ] Maximum confirmation time: <6 hours (even with low fees)

---

### 7.3 Performance (Medium Priority - P1)

**NFR3.1: Signing Performance Targets**: - Threshold signature generation: <2 minutes (typical) - Maximum signing latency: <5 minutes - Support 10+ concurrent signing sessions - No performance degradation with load

**Acceptance Criteria**: - [ ] 95th percentile signing time <3 minutes - [ ] 99th percentile signing time <5 minutes - [ ] Can handle 50 loans per day - [ ] No bottlenecks under load testing

---

### 7.4 Decentralization (High Priority - P0)

**NFR4.1: Geographic Distribution Requirements**: - No more than 3 validators in same country - No more than 5 validators in same region - Minimum 3 continents represented - No cloud provider hosts >40% of validators

**Initial Distribution**: - North America: 5 validators - Europe: 5 validators - Asia: 3 validators - Other: 2 validators

**Acceptance Criteria**: - [ ] Geographic diversity meets targets - [ ] No single jurisdiction can censor - [ ] Cloud provider diversity maintained - [ ] Validators in politically diverse regions

---

**NFR4.2: Validator Independence Requirements**: - No single entity controls >2 validators (13%) - Validators have different operators/owners - No conflicts of interest (validators can't be borrowers/lenders) - Public disclosure of validator identities (optional anonymity)

**Acceptance Criteria**: - [ ] All validators are independent entities - [ ] No undisclosed common ownership - [ ] Conflicts of interest documented and mitigated - [ ] Validator registry publicly available

---

## 8. Success Metrics

### 8.1 Launch Success Metrics (Months 14-15)

**Critical Launch KPIs** (must achieve all):

| Metric | Target | Measurement | Critical? |
|---|---|---|---|
| **Validator Network** | 15/15 validators online | Real-time monitoring | CRITICAL |
| **Network Uptime** | 99.9%+ from day 1 | Uptime monitoring | CRITICAL |
| **DKG Success** | 1 successful ceremony | Audit logs | CRITICAL |
| **First Native BTC Loan** | Within 48 hours | Blockchain data | CRITICAL |
| **Zero Critical Bugs** | 0 in first 2 weeks | Bug tracker | CRITICAL |
| **Threshold Signatures** | 100% success rate | Transaction logs | CRITICAL |

**Soft Launch Metrics** (Month 15): - 5+ native BTC loans created - 3+ loans successfully repaid - 10+ lenders participating - $500K+ native BTC loan volume - Average loan size: $50K+ - No validator downtime >30 minutes - Zero signature failures

---

### 8.2 Growth Metrics (Months 15-16)

**Volume Growth**:

| Milestone | Target | Timeline |
|---|---|---|
| **First $100K** | Week 1 of soft launch | Month 15, Week 1 |
| **First $500K** | Week 4 of soft launch | Month 15, Week 4 |
| **First $1M** | End of soft launch | Month 15 end |
| **$2M Milestone** | Mid full launch | Month 16, Week 2 |
| **$5M Milestone** | End of Phase 2 | Month 16 end |

**User Growth**: - Native BTC borrowers: 15+ by Month 16 - Native BTC lenders: 30+ by Month 16 - Large loans (>$100K): 5+ by Month 16 - Repeat borrowers: 30%+ by Month 16 - Active loans: 20+ by Month 16

**Market Share** (vs Phase 1 sBTC): - Month 15: 20% of new loans use native BTC - Month 16: 50%+ of new loans use native BTC - Migration: 30%+ of Phase 1 loans migrate to Phase 2

---

### 8.3 Technical Performance Metrics

**Validator Network Health**:

| Metric | Target | Measurement |
|---|---|---|
| **Validator Uptime** | 99.9% per validator | Prometheus metrics |
| **Network Uptime** | 99.9% overall | Consensus tracking |
| **Signature Success Rate** | 100% | Transaction logs |
| **Signature Time** | <10 seconds avg | Performance logs |
| **DKG Ceremony Time** | <30 minutes | Ceremony logs |
| **Partial Signature Time** | <5 seconds | Individual validator logs |
| **Network Latency** | <2 seconds p95 | P2P metrics |
| **Bitcoin Confirmations** | Detected within 2 blocks | Blockchain monitoring |

**Cross-Chain Performance**: - State sync lag: <1 block between Bitcoin and Stacks - Reorg handling: 100% successful up to 6 blocks - Deposit detection: Within 2 Bitcoin blocks - Withdrawal execution: Within 6 Bitcoin blocks - State inconsistencies: 0

**Operational Performance**: - Alert response time: <5 minutes (P1), <30 minutes (P2) - Incident resolution: <2 hours (P1), <24 hours (P2) - Disaster recovery: <4 hours to full operation - Key recovery: <2 hours per validator - Monitoring coverage: 100% of critical systems

---

### 8.4 Security Metrics

**Zero Tolerance Metrics** (any failure = immediate halt): - Unauthorized transactions: 0 - Key compromises: 0 - Fund losses: $0 - Successful attacks: 0 - Critical vulnerabilities: 0 (post-audit)

**Audit Compliance**: - All critical findings resolved: 100% - All high findings resolved: 100% - Medium findings: Resolved or documented - Re-audit passed: Yes

**Bug Bounty Results**: - Critical vulnerabilities reported: 0 expected - Valid submissions: <5 low/medium - Average response time: <48 hours - Average fix time: <7 days (critical), <30 days (high)

---

### 8.5 Migration Success Metrics

**Phase 1 → Phase 2 Migration**:

| Metric | Target | Timeline |
|---|---|---|
| **Migration Opt-in Rate** | 30%+ | Month 14-16 |
| **Successful Migrations** | 100% of attempts | Month 14-16 |
| **Migration Time** | <24 hours per loan | Month 14-16 |
| **Value Preserved** | 100% (zero loss) | Month 14-16 |
| **User Satisfaction** | 4.5+/5 rating | Post-migration survey |

**Migration Volume**: - $500K+ migrated from sBTC to native BTC - 10+ loans successfully migrated - Zero migration failures or stuck loans

---

### 8.6 Validator Performance Metrics

**Individual Validator KPIs**:

| Metric | Target | Consequence if Failed |
|---|---|---|
| **Uptime** | 99.5% per validator | Warning after 3 violations |
| **Signature Participation** | 95%+ of signing rounds | Investigation after <90% |
| **Response Time** | <10 seconds | Warning after consistent delays |
| **Geographic Distribution** | 3+ continents | Minimum diversity requirement |
| **Hardware Compliance** | Meets minimum specs | Cannot join network |

**Network Diversity**: - North America: 4-6 validators - Europe: 4-6 validators - Asia: 2-4 validators - Other regions: 1-3 validators - No single entity: >3 validators (20%)

**Validator Economic Viability**: - Protocol fees collected: $5,000+ by Month 16 - Per-validator earnings: $333+/month by Month 16 - Validator ROI: Break-even by Month 21 (5 months post-launch)

---

### 8.7 User Experience Metrics

**Native BTC Loan Creation Flow**: - Completion rate: >80% - Average time: <10 minutes - Drop-off rate: <20% - Error rate: <2%

**Validator Transparency**: - Users can view validator status: 100% - Users understand threshold model: 70%+ (survey) - Trust in decentralization: 4+/5 rating

**Support & Documentation**: - User questions answered: <2 hours - Documentation completeness: 95%+ - Tutorial completion rate: 60%+

---

### 8.8 Failure Criteria (Red Flags)

**Critical Failures** (immediate intervention required):

1. **Validator Network Failure**
   - <10 validators online for >1 hour
   - Signature success rate <95%
   - DKG ceremony failure (cannot retry)
2. **Security Incident**
   - Any unauthorized transaction
   - Any key compromise
   - Critical vulnerability discovered post-launch
3. **Technical Failure**
   - State inconsistency between Bitcoin and Stacks
   - Reorg not handled correctly (funds lost/stuck)
   - Network partition lasting >6 hours
4. **Low Adoption**
   - <$500K volume by Month 16
   - <10 native BTC loans by Month 16
   - <20% new loans use native BTC by Month 16
5. **Migration Failure**
   - Migration success rate <90%
   - Value lost during migration
   - User complaints >10% of migrations

**Warning Indicators** (require action but not halt): - Validator uptime 95-99% (target: 99.9%) - Signature time >20 seconds (target: <10s) - User growth <50% of target - Migration rate <20% (target: 30%)

---

### 8.9 Success Milestones (Month-by-Month)

**Month 14: Launch Preparation** - All 15 validators deployed - DKG ceremony completed - Security audit passed - Operational tooling deployed - Disaster recovery tested - Migration tools ready

**Month 15: Soft Launch** - First native BTC loan funded - 5+ loans created - $500K volume - Zero critical bugs - 99.9% uptime maintained - Positive user feedback

**Month 16: Full Launch** - $5M volume achieved - 15+ borrowers - 30+ lenders - 50%+ market share (new loans) - 30%+ migration rate - Validator network profitable

**Months 17-18: Buffer Period (Stabilization)** - 2 months continuous operation - No critical incidents - Validator uptime maintained 99.9% - User satisfaction high (4.5+/5) - Ready for Phase 3

---

## 9. Phase 2 Deliverables

### 9.1 Deliverable Overview

**Total Deliverables**: 15
**Total Budget**: $463,000
**Timeline**: 9 months (Months 8-16)
**New Deliverables**: 6 (D2.1a, D2.3a, D2.4a, D2.4b, D2.10, D2.11)
**Enhanced Deliverables**: 1 (D2.6 - Security Audit)

---

### 9.2 Deliverable Summary Table

| Code | Deliverable | Timeline | Budget | Type | Dependencies |
|---|---|---|---|---|---|
| **D2.1** | Threshold Crypto Library | Month 8 | $45,000 | Core | None |
| **D2.1a** | Validator Recruitment Program | Months 8-10 | $25,000 | **NEW** | None |
| **D2.2** | Validator Node Software (Rust) | Months 8-10 | $62,000 | Core | D2.1 |
| **D2.3** | DKG Ceremony Implementation | Months 9-10 | $28,000 | Core | D2.1, D2.2 |
| **D2.3a** | DKG Ceremony Coordination | Months 10-12 | $20,000 | **NEW** | D2.2, D2.3 |
| **D2.4** | Stacks Bitcoin Bridge Logic | Months 9-11 | $45,000 | Core | D2.2 |
| **D2.4a** | Enhanced State Management | Months 10-11 | $28,000 | **NEW** | D2.4 |

| Code | Deliverable | Timeline | Budget | Type | Dependencies |
|---|---|---|---|---|---|
| **D2.4b** | Network Protocol (libp2p) | Months 10-11 | $21,000 | **NEW** | D2.2 |
| **D2.5** | Validator Network Launch | Months 11-12 | $38,000 | Core | D2.1a, D2.2, D2.3 |
| **D2.6** | Security Audit (Enhanced) | Months 12-13 | $82,000 | **EXPANDED** | All above |
| **D2.7** | Frontend Updates (Native BTC) | Months 11-13 | $25,000 | Core | D2.4 |
| **D2.8** | Migration Tools | Months 13-14 | $12,000 | Core | D2.7 |
| **D2.10** | Operational Tooling | Month 13 | $15,000 | **NEW** | D2.5 |
| **D2.11** | Disaster Recovery Testing | Month 13 | $12,000 | **NEW** | D2.5, D2.10 |
| **D2.9** | Launch & $5M Volume | Months 14-16 | $5,000 | Core | All above |

**TOTAL**: $463,000 over 9 months

---

### 9.3 Detailed Deliverable Specifications

**D2.1: Threshold Crypto Library Integration ($45,000)** **Timeline**: Month 8 (1 month)
**Budget**: $45,000
**Team**: Cryptography specialist + Rust developer

**Scope**: - Research and select optimal threshold signature scheme - Options: FROST (Flexible Round-Optimized Schnorr Threshold) - secp256k1-zkp with MuSig2 - Custom implementation evaluation - Integrate selected library with Rust validator software - Implement Bitcoin Taproot signature generation - Comprehensive test suite for cryptographic correctness - Performance benchmarking on standard hardware

**Deliverables**: - [ ] Library selected and justified (technical report) - [ ] Integration complete and compiling - [ ] 100% test coverage of cryptographic functions - [ ] Benchmark results (<5 seconds signing time) - [ ] Documentation on cryptographic approach - [ ] Test vectors for Bitcoin signature verification - [ ] Independent cryptographer review completed

**Success Criteria**: - Signatures valid per Bitcoin consensus rules - Signing time <5 seconds on standard hardware (4-core CPU) - 100% test pass rate across all test vectors - Independent cryptographer sign-off - No known vulnerabilities in selected library - Compatible with Bitcoin Taproot

**Acceptance Testing**:

```
#[test]
fn test_threshold_signature_bitcoin_valid() {
    let keypair = generate_threshold_keypair(10, 15);
    let message = bitcoin_tx_hash();
    let partial_sigs = sign_partial(keypair, message, 10); // 10-of-15
    let combined = aggregate_signatures(partial_sigs);

    assert!(verify_bitcoin_signature(combined, message));
}
```

**Dependencies**: None (start immediately Month 8)

---

**D2.1a: Validator Recruitment Program ($25,000) NEW** **Timeline**: Months 8-10 (3 months)
**Budget**: $25,000
**Team**: Project lead + community manager

**Scope**: - Define validator qualification criteria - Technical: Rust experience, sysadmin skills, 24/7 availability - Financial: $800 hardware budget, $170/month operating costs - Geographic: Target 3+ continents - Source and outreach to potential validators - Bitcoin developer communities - Stacks ecosystem participants - Professional node operators - Conduct technical interviews (1 hour each, 20+ candidates) - Hardware and infrastructure verification - Legal agreement preparation and signing - Onboarding with training materials and docs - Establish communication channels (Discord, Telegram, email)

**Deliverables**: - [ ] 15 committed validators signed on - [ ] Validator qualification criteria document - [ ] Geographic distribution achieved (3+ continents) - [ ] Validator agreement contracts (all signed) - [ ] Onboarding materials (setup guides, video tutorials) - [ ] Communication infrastructure (Discord server, mailing list) - [ ] Public validator registry with contact information - [ ] Hardware verification reports (all 15 validators)

**Success Criteria**: - 15 validators recruited and committed by Month 10 - Geographic distribution: - North America: 5 validators - Europe: 5 validators - Asia: 3 validators - Other: 2 validators - All validators pass technical assessment - All validators sign legal agreement - All validators complete onboarding training - All validators have hardware ready by Month 11 - 90%+ participation in communication channels

**Recruitment Funnel**:

```
Target: 15 validators
Outreach: 50+ candidates
Interviews: 25 candidates
Offers: 18 candidates
Accepted: 15 validators
```

**Geographic Distribution Example**: - USA (2), Canada (2), Mexico (1) = 5 North America - UK (2), Germany (1), France (1), Netherlands (1) = 5 Europe - Japan (1), Singapore (1), South Korea (1) = 3 Asia - Australia (1), Brazil (1) = 2 Other

**Dependencies**: None (can start Month 8)

---

**D2.2: Validator Node Software ($62,000)** **Timeline**: Months 8-10 (3 months)
**Budget**: $62,000
**Team**: 2 Rust developers

**Scope**: - Build Rust application for validator nodes - Core features: - Monitor Bitcoin blockchain (via Bitcoin Core RPC) - Monitor Stacks blockchain (via Stacks node RPC) - Participate in threshold signature ceremonies - P2P communication with other validators (libp2p - see D2.4b) - Manage encrypted key shares (HSM or secure file storage) - Export Prometheus metrics for monitoring - Logging with structured output (JSON) - Configuration management (TOML files) - CLI for validator operations - Security hardening - Resource optimization (<4GB RAM, <10% CPU idle) - Documentation and deployment guides

**Deliverables**: - [ ] Rust binary (Linux x86_64) - [ ] Docker image for containerized deployment - [ ] Configuration files and examples - [ ] Comprehensive setup documentation - [ ] API documentation (internal) - [ ] Monitoring dashboards (Grafana - see D2.10) - [ ] Unit tests (>90% coverage) - [ ] Integration tests with Bitcoin/Stacks testnets

**Success Criteria**: - Binary runs stably for 7+ days without restart on testnet - Resource usage acceptable: - RAM: <4GB under load - CPU: <10% idle, <50% under signing - Disk: <100GB for blockchain data - Network: <1Mbps bandwidth - All functionality tested (unit + integration) - Documentation clear enough for operators to self-deploy - Zero memory leaks in 7-day run - Graceful handling of network interruptions

**Key Modules**:

```
// Validator software architecture
mod bitcoin_monitor;   // Watch Bitcoin blockchain
mod stacks_monitor;    // Watch Stacks blockchain
mod threshold_signer;  // Participate in signing (uses D2.1)
mod p2p_network;       // Communicate with peers (D2.4b)
mod key_manager;       // Manage encrypted key shares
mod config;            // Configuration management
mod metrics;           // Prometheus metrics export
mod cli;               // Command-line interface
```

**Dependencies**: D2.1 (threshold crypto library)

---

**D2.3: DKG Ceremony Implementation ($28,000)** **Timeline**: Months 9-10 (2 months)
**Budget**: $28,000
**Team**: Cryptography specialist + Rust developer

**Scope**: - Implement Distributed Key Generation (DKG) protocol - Support 15-validator ceremony (10-of-15 threshold) - Byzantine fault tolerance (detect and exclude malicious validators) - Verification and commitment rounds - Export combined public key (Taproot-compatible) - Ceremony auditability (all messages logged) - Recovery from failed ceremonies - Timeouts and retry logic

**Deliverables**: - [ ] DKG ceremony module in validator software - [ ] Ceremony coordinator scripts - [ ] Test ceremonies on testnet (3+ successful runs) - [ ] Audit logs from test ceremonies - [ ] Documentation on ceremony process - [ ] Troubleshooting guide - [ ] Ceremony verification tools

**Success Criteria**: - DKG completes successfully with 15 honest validators on testnet - DKG detects and excludes malicious participants (tested with 3 bad actors) - Combined public key is Taproot-compatible - Ceremony takes <10 minutes for 15 validators - 100% of key shares verified correct - Audit trail complete and verifiable - Can recover from failed ceremony attempts

**DKG Process**:

```
Phase 1: Commitment (each validator commits to random value)
Phase 2: Share Distribution (validators exchange shares)
Phase 3: Verification (validators verify shares received)
Phase 4: Key Aggregation (combine shares → public key)
Phase 5: Confirmation (all validators confirm success)

Total time: <10 minutes for 15 validators
```

**Dependencies**: D2.1 (crypto library), D2.2 (validator software)

---

**D2.3a: DKG Ceremony Coordination ($20,000) NEW** **Timeline**: Months 10-12 (3 months)
**Budget**: $20,000
**Team**: Project lead + coordinator

**Scope**: - Plan and schedule testnet DKG rehearsals - Coordinate across time zones (3+ continents) - Send calendar invites and reminders - Prepare step-by-step ceremony guide - Conduct 2+ testnet DKG ceremonies - Provide real-time support during ceremony - Troubleshoot issues as they arise - Document all issues and resolutions - Schedule and facilitate mainnet DKG ceremony - Coordinate 15 validators in real-time - Monitor ceremony progress - Verify successful completion - Post-ceremony verification - Test combined public key - Verify key shares distributed correctly - Test threshold signatures - Create ceremony audit trail - Log all messages and actions - Document timeline and participants - Publish ceremony report

**Deliverables**: - [ ] 2+ successful testnet DKG rehearsals - [ ] Detailed ceremony procedures and runbooks - [ ] Coordinator playbook (step-by-step) - [ ] Mainnet DKG ceremony successfully completed - [ ] Ceremony audit logs (public) - [ ] Post-ceremony verification report - [ ] Lessons learned document - [ ] Emergency rollback procedures (if needed)

**Success Criteria**: - Testnet DKG ceremonies: 2+ successful completions - All 15 validators participate in testnet rehearsals - Mainnet DKG completes in <30 minutes - All validators successfully generate key shares - Combined public key verified on Bitcoin testnet - Test transactions signed successfully post-DKG - Zero key shares compromised or lost - Complete audit trail available publicly

**Coordination Timeline**:

```
Month 10: Testnet Rehearsal 1
Month 11: Testnet Rehearsal 2
Month 12: Mainnet Ceremony (Week 1)
Month 12: Post-ceremony Verification (Week 2)
```

**Dependencies**: D2.2 (validator software), D2.3 (DKG implementation)

---

**D2.4: Stacks  Bitcoin Bridge Logic ($45,000)** **Timeline**: Months 9-11 (3 months)
**Budget**: $45,000
**Team**: Smart contract developer + Bitcoin developer

**Scope**: - Update Stacks smart contracts (Clarity): - Monitor Bitcoin deposits via STX transaction proofs - Emit validator action requests (withdrawals) - Track Bitcoin transaction confirmations (6+ blocks) - Handle cross-chain state synchronization - Bitcoin transaction builder: - Construct withdrawal transactions - UTXO selection and management - Fee estimation (dynamic based on mempool) - RBF (Replace-By-Fee) support for stuck transactions - Integration with validator network: - Action request queue - Signature aggregation coordination - Transaction broadcast

**Deliverables**: - [ ] Updated Stacks contracts (Clarity) - [ ] Bitcoin transaction construction library (Rust) - [ ] UTXO management module - [ ] Fee estimation module (connects to mempool API) - [ ] Integration tests (cross-chain scenarios) - [ ] API for frontends to query bridge status - [ ] Documentation on bridge architecture - [ ] Testnet deployment and testing

**Success Criteria**: - Deposits detected within 2 Bitcoin blocks - 100% of valid action requests result in Bitcoin transactions - No funds ever stuck due to bridge logic - State sync works even during Bitcoin reorgs (tested up to 6 blocks) - Fee estimation accurate (within 20% of actual) - RBF works for stuck transactions - Zero fund losses in testnet testing

**Bridge Flow**:

```
Deposit Flow:
User → Bitcoin TX → Bitcoin Network → Stacks Proof → Smart Contract → Credit User

Withdrawal Flow:
Smart Contract → Action Request → Validators → Threshold Sign → Bitcoin TX → Bitcoin Networ
```

**Dependencies**: D2.2 (validator software)

---

**D2.4a: Enhanced State Management ($28,000) NEW Timeline**: Months 10-11 (2 months)
**Budget**: $28,000
**Team**: Distributed systems specialist + Rust developer

**Scope**: - Handle Bitcoin blockchain reorganizations (reorgs) - Detect reorgs up to 6 blocks deep - Roll back affected state on Stacks - Retry affected transactions - Notify users of reorg events - Stacks blockchain state synchronization - Handle Stacks microblocks - Synchronize with Bitcoin state - Resolve state conflicts - Cross-chain state consistency verification - Periodic state checksums - Automated consistency checks - Alert on inconsistencies - Atomic state updates - Ensure Bitcoin and Stacks state updated together - Transaction-like semantics for state changes - Conflict resolution mechanisms - Handle concurrent state updates - Prioritize Bitcoin state as source of truth - State rollback and recovery procedures - Recover from crashed state - Rebuild state from Bitcoin blockchain - Comprehensive state logging and debugging

**Deliverables**: - [ ] Reorg handling logic (up to 6 blocks) - [ ] State synchronization protocol - [ ] State verification tests (100+ scenarios) - [ ] Atomic update implementation - [ ] Conflict resolution algorithms - [ ] State recovery procedures (documented and tested) - [ ] Monitoring dashboards for state health - [ ] Alert rules for state inconsistencies

**Success Criteria**: - Handle Bitcoin reorgs up to 6 blocks deep (tested 10+ times) - Stacks state syncs within 1 block of Bitcoin state - Zero state inconsistencies in 30-day testnet run - Atomic updates succeed 100% of time - Recovery from crashed state in <5 minutes - State verification tests pass 100% - Reorg handling time <30 seconds

**State Management Architecture**:

```
struct StateManager {
    bitcoin_state: BitcoinState,      // Bitcoin blockchain state
    stacks_state: StacksState,        // Stacks blockchain state
    consistency_checker: Checker,     // Verify cross-chain consistency
    reorg_handler: ReorgHandler,      // Handle Bitcoin reorgs
    logger: StateLogger,              // Comprehensive logging
}
```

**Dependencies**: D2.4 (bridge logic)

---

**D2.4b: Network Protocol Implementation ($21,000) NEW Timeline**: Months 10-11 (2 months)
**Budget**: $21,000
**Team**: Network engineer + Rust developer

**Scope**: - Implement libp2p for validator-to-validator communication - Peer discovery and connection management - NAT traversal for validators behind firewalls - TLS encryption for all messages - Gossip protocol for signature requests - Efficient message propagation - Deduplicate messages - Prioritize urgent requests - Partial signature aggregation protocol - Collect M-of-N partial signatures - Verify each partial signature - Aggregate into final signature - Network topology optimization - Minimize latency between validators - Redundant connections for reliability - Message authentication and encryption - Each message signed by sender - Prevent spoofing and replay attacks - Network resilience testing - Handle validator disconnections - Recover from network partitions - Test with 5/15 validators offline

**Deliverables**: - [ ] libp2p integration in validator software - [ ] Gossip protocol implementation - [ ] Signature aggregation module - [ ] Peer discovery mechanism - [ ] Network monitoring tools - [ ] Performance benchmarks (latency, throughput) - [ ] Network partition recovery tests - [ ] Documentation on network protocol

**Success Criteria**: - All 15 validators connected via libp2p - Gossip reaches all peers in <5 seconds - Signature aggregation completes in <10 seconds - Network handles 10+ signing requests per minute - Resilient to 5/15 validator failures - Network recovers from partitions in <2 minutes - Zero message loss in normal conditions - Latency p95: <2 seconds

**Network Architecture**:

```
Gossip Network (libp2p):

  Val 1       Val 2       Val 3
```

```
                  Val 15


Signature Request → Gossip → All Validators → Partial Sigs → Aggregator
```

**Dependencies**: D2.2 (validator software)

---

**D2.5: Validator Network Launch ($38,000)** **Timeline**: Months 11-12 (2 months)
**Budget**: $38,000
**Team**: DevOps engineer + project lead

**Scope**: - Deploy all 15 validator nodes to production hardware - Coordinate with each validator operator - Verify hardware meets specifications - Install and configure validator software - Establish network communication - Configure libp2p connections - Test peer connectivity - Verify gossip propagation - Initial testing on Bitcoin and Stacks testnets - Test threshold signatures - Test DKG ceremony on testnet - Simulate signing scenarios - Mainnet launch preparation - Final hardware checks - Security hardening - Backup procedures - Mainnet DKG ceremony (see D2.3a) - Post-launch monitoring and support

**Deliverables**: - [ ] 15 active validators on mainnet - [ ] Public validator registry - [ ] Network health dashboard (real-time) - [ ] Validator operating procedures (documented) - [ ] Emergency contact list (24/7 coverage) - [ ] Launch checklist (completed) - [ ] Post-launch report (first week)

**Success Criteria**: - All 15 validators online and signing - Network uptime >99% from day 1 - Geographic distribution achieved: - North America: 5 validators - Europe: 5 validators - Asia: 3 validators - Other: 2 validators - No validator downtime >1 hour in first week - All validators responding to signature requests - Network latency acceptable (<2s p95)

**Launch Checklist**:

```
Pre-Launch (Month 11):
[ ] Hardware verified for all 15 validators
[ ] Software deployed to all validators
[ ] Network connectivity tested
[ ] Testnet DKG ceremony successful (D2.3a)
[ ] Emergency procedures documented

Launch Day (Month 12, Week 1):
```

```
[ ] Mainnet DKG ceremony (D2.3a)
[ ] All validators online
[ ] Combined public key verified
[ ] Test signature successful
[ ] Monitoring active

Post-Launch (Month 12, Week 2-4):
[ ] 24/7 monitoring
[ ] Daily health checks
[ ] Performance optimization
[ ] Issue resolution
```

**Dependencies**: D2.1a (recruitment), D2.2 (software), D2.3 (DKG implementation)

---

**D2.6: Security Audit (Enhanced Scope) ($82,000) EXPANDED**
**Timeline**: Months 12-13 (2 months)
**Budget**: $82,000 (was $52,000, +$30,000)
**Team**: External audit firm(s)

**Scope** (Enhanced): - **Threshold Cryptography Audit** ($30,000 - specialist required) - DKG implementation security - Partial signature security - Key share storage and encryption - Threshold parameter validation (10-of-15) - Cryptographic primitives correctness - **Validator Network Audit** ($22,000) - P2P network security (libp2p) - Gossip protocol vulnerabilities - Sybil attack resistance - Network partition handling - Validator authentication - **Bridge Logic Audit** ($15,000) - Cross-chain state management - Bitcoin transaction construction - UTXO management security - Reorg handling security - Withdrawal authorization - **Economic Attack Vectors** ($10,000) - Game theory analysis - Validator collusion scenarios - 51% attack scenarios - Griefing attacks - Incentive compatibility - **Follow-up Re-audit** ($5,000) - Verify all critical findings fixed - Re-test high severity issues - Final security sign-off

**Audit Firms** (Preferred): 1. **Trail of Bits** (threshold crypto specialists) 2. **Least Authority** (distributed systems experts) 3. **CoinFabrik** (Clarity/Stacks experience) 4. **NCC Group** (general blockchain security)

**Deliverables**: - [ ] Comprehensive audit report (PDF) - [ ] Findings list with severity ratings: - Critical: Immediate fund risk - High: Significant security issue - Medium: Edge case vulnerabilities - Low: Code quality improvements - [ ] Remediation recommendations for each finding - [ ] Follow-up review after fixes implemented - [ ] Public publication of final audit report - [ ] Security best practices document

**Success Criteria**: - Zero critical vulnerabilities remaining - All high-severity issues resolved - All medium issues: fixed or documented with accepted risk -

All low issues: fixed or backlogged - Audit firm provides final sign-off - Report published publicly on protocol website

**Audit Timeline**:

```
Month 12, Week 1-2: Audit kick-off, code review
Month 12, Week 3-4: Threshold crypto deep dive
Month 13, Week 1: Findings report delivered
Month 13, Week 2-3: Development team addresses findings
Month 13, Week 4: Follow-up re-audit
Month 13 end: Final report published
```

**Why $82K? (was $52K)**: - Threshold cryptography requires specialized auditors (+$15K) - 15-validator distributed system more complex than anticipated (+$8K) - Economic game theory analysis essential (+$5K) - Extended audit time for thoroughness (+$2K)

**Dependencies**: All prior deliverables (D2.1-D2.5, D2.4a-D2.4b, D2.10-D2.11)

---

**D2.7: Frontend Updates ($25,000)** **Timeline**: Months 11-13 (3 months)
**Budget**: $25,000
**Team**: Frontend developer + UX designer

**Scope**: - Native Bitcoin deposit UI - Display threshold multisig address (P2TR Taproot) - QR code for mobile deposit - Deposit tracking and confirmation status - Threshold multisig address generation - Derive address from combined public key - Display address in multiple formats (Bech32m) - Bitcoin transaction monitoring - Real-time deposit detection - Confirmation count display - Withdrawal status tracking - Validator transparency dashboard - Show all 15 validators and their status - Uptime and performance metrics - Geographic distribution visualization - Migration from Phase 1 interface - Migrate sBTC loan to native BTC - Explain benefits of migration - Step-by-step migration wizard

**Deliverables**: - [ ] Updated React frontend (Next.js) - [ ] Native BTC loan creation flow - [ ] Validator dashboard page - [ ] Migration wizard UI - [ ] User documentation (written guides) - [ ] Video tutorials (5+ videos) - [ ] Mobile responsive design - [ ] Cross-browser testing (Chrome, Firefox, Safari)

**Success Criteria**: - Users can create native BTC loans in <10 minutes - Bitcoin addresses displayed clearly (QR + text) - Validator status visible and understandable - Migration wizard completion rate >80% - Mobile responsive (tested on iOS + Android) - Error handling graceful (no crashes) - User testing with 5+ users: 4.5+/5 rating

**UI Flows**:

```
Native BTC Loan Creation:
1. Connect Wallet → 2. Choose Native BTC → 3. Enter Loan Parameters
```

```
→ 4. Display Deposit Address (QR) → 5. Wait for Confirmations
→ 6. Create Auction → 7. Monitor Bids

Validator Dashboard:
1. Navigate to "Validators" → 2. See 15 Validator Cards
→ 3. View Status (Online/Offline) → 4. View Geographic Map
→ 5. View Performance Metrics

Migration Wizard:
1. Select sBTC Loan → 2. Review Benefits → 3. Initiate Migration
→ 4. Wait for sBTC Unlock → 5. Deposit Native BTC
→ 6. Migration Complete
```

**Dependencies**: D2.4 (bridge logic)

---

**D2.8: Migration Tools ($12,000)** **Timeline**: Months 13-14 (2 months)
**Budget**: $12,000
**Team**: Smart contract developer + frontend developer

**Scope**: - Smart contracts for Phase 1 → Phase 2 migration - Unlock sBTC collateral from Phase 1 loan - Create equivalent native BTC loan in Phase 2 - Preserve loan terms (amount, maturity, lender) - Transfer NFT positions (borrower/lender) - Migration UI workflow - Select loans to migrate - Review migration details - One-click migration execution - Track migration progress - User communication and education - Email notifications about migration - FAQ on migration process - Benefits of native BTC vs sBTC - Migration incentives (if any) - Analytics on migration rates - Track migration adoption - Identify barriers to migration - Report on migration success

**Deliverables**: - [ ] Migration smart contracts (Clarity) - [ ] Migration UI workflow (React components) - [ ] User guides and FAQs (written + video) - [ ] Email notification system - [ ] Migration tracking dashboard (for team) - [ ] Migration analytics reports

**Success Criteria**: - 30%+ of Phase 1 users migrate to Phase 2 - Migration completes in <24 hours per loan - Zero value lost during migration - No failed migrations (or 100% recoverable) - High user satisfaction: 4.5+/5 rating (post-migration survey) - Clear communication: 80%+ users understand process

**Migration Flow**:

```
User Experience:
1. User sees "Migrate to Native BTC" banner on dashboard
2. User clicks "Learn More" → Sees benefits (eliminate peg risk, etc.)
3. User selects loans to migrate
4. User reviews migration details (preview)
5. User clicks "Migrate Now"
```

```
6. Smart contract unlocks sBTC, returns to user
7. User deposits native BTC to new address
8. New native BTC loan created with same terms
9. NFTs transferred to new loan
10. Migration complete (<24 hours)
```

**Dependencies**: D2.7 (frontend updates)

---

**D2.10: Operational Tooling ($15,000) NEW** **Timeline**: Month 13 (1 month)
**Budget**: $15,000
**Team**: DevOps engineer

**Scope**: - Grafana dashboards for comprehensive monitoring - Validator health dashboard (15 validators) - Network performance dashboard - Signature request dashboard - Cross-chain state dashboard - Bitcoin/Stacks blockchain monitoring - Alert history dashboard - PagerDuty integration for 24/7 alerting - Critical alerts (P1): Validator down, signature failure - High alerts (P2): Slow performance, network issues - Medium alerts (P3): Warnings, informational - Automated health checks - Validator availability checks (every 1 minute) - Signature success rate checks - Network latency checks - Bitcoin/Stacks node health checks - Performance metrics collection (Prometheus) - Export metrics from all 15 validators - Aggregate network-wide metrics - Historical metrics storage (30 days) - Log aggregation and analysis (ELK stack) - Centralized logging from all validators - Full-text search across logs - Log retention policy (90 days) - Incident response runbooks - 10+ common scenarios documented - Step-by-step resolution procedures - Escalation paths - Backup and recovery automation - Automated key share backups (encrypted) - Database backups (validator state) - Disaster recovery procedures (documented and tested - see D2.11)

**Deliverables**: - [ ] Grafana dashboard suite (10+ dashboards) - [ ] PagerDuty alert rules (20+ alert types) - [ ] Health check scripts (automated, scheduled) - [ ] Prometheus metric exporters (deployed to all validators) - [ ] ELK stack deployed (centralized logging) - [ ] Incident response runbooks (10+ scenarios) - [ ] Backup automation scripts (tested) - [ ] Operations manual (comprehensive documentation)

**Success Criteria**: - All 15 validators reporting metrics to Grafana - Dashboards show real-time status (<10s delay) - Alerts fire within 1 minute of issue detection - 100% of critical incidents detected by monitoring - Logs searchable and easily analyzable - Runbooks tested with 3+ real scenarios - Backups automated and verified (restore tested) - Operations team trained on all tools

**Dashboard Examples**:

```
Validator Health Dashboard:
- 15 validator status cards (green/yellow/red)
- Uptime percentage per validator
- Last signature time per validator
- Resource usage (CPU, RAM, disk)

Network Performance Dashboard:
- Signature request queue length
- Average signature time (p50, p95, p99)
- Network latency between validators
- Gossip propagation time

Alert Examples:
P1: Validator {name} offline for >5 minutes
P1: Signature failure rate >5%
P2: Signature time >20 seconds (p95)
P2: Network latency >5 seconds (p95)
P3: Validator resource usage high (>80%)
```

**Dependencies**: D2.5 (validator network launched)

---

**D2.11: Disaster Recovery Testing ($12,000) NEW** **Timeline**: Month 13 (1 month)
**Budget**: $12,000
**Team**: DevOps engineer + project lead

**Scope**: - Test key recovery procedures - Simulate 3 validator key losses - Recover keys from encrypted backups - Verify recovered validators can sign - Test network recovery (validator failures) - Simulate 5 validators going offline - Verify 10-of-15 threshold still works - Test network recovery when validators return - Test coordinator failure and replacement - Simulate coordinator node failure - Failover to backup coordinator - Verify signing continues without interruption - Test Bitcoin node failure scenarios - Simulate Bitcoin node crash - Verify validator detects issue - Test automatic reconnection - Test Stacks node failure scenarios - Simulate Stacks node crash - Verify state recovery - Test bridge logic failover - Test HSM failure and recovery - Simulate HSM malfunction - Test key extraction from backup - Restore key to replacement HSM - Document all disaster recovery procedures - Step-by-step recovery guides - Emergency contact procedures - Escalation protocols

**Deliverables**: - [ ] Key recovery procedure (tested 3+ times successfully) - [ ] Network partition recovery (tested 5+ scenarios) - [ ] Coordinator failover procedure (tested 2+ times) - [ ] Node failure recovery runbooks (Bitcoin + Stacks) - [ ] HSM failure recovery guide (tested 1+ time) - [ ] Disaster recovery test report (comprehensive) - [ ] Updated incident response procedures - [ ] Disaster recovery drill schedule (quarterly)

**Success Criteria**: - Key recovery tested successfully 3+ times (all 3 succeeded)

- Network recovers from 5/15 validators down (tested 5+ times) - Coordinator replaced without downtime (tested 2+ times) - Bitcoin node failures handled gracefully (tested 3+ times) - Stacks node failures handled gracefully (tested 3+ times) - HSM failures recoverable in <2 hours (tested 1+ time) - All procedures documented clearly - All procedures verified by 2+ team members

**Test Scenarios**:

```
Scenario 1: Key Loss Recovery
1. Simulate 3 validators lose key shares
2. Retrieve encrypted backups from secure storage
3. Decrypt backups using master key
4. Restore key shares to validators
5. Verify validators can participate in signing
6. Time to recovery: <2 hours

Scenario 2: Network Partition
1. Simulate 5 validators disconnected (network partition)
2. Verify 10 remaining validators can still sign
3. Simulate reconnection of 5 validators
4. Verify network resumes normal operation
5. Time to recovery: <5 minutes

Scenario 3: Coordinator Failure
1. Simulate coordinator node crash
2. Verify backup coordinator takes over
3. Verify signing requests continue
4. Verify no signatures lost
5. Time to failover: <30 seconds
```

**Dependencies**: D2.5 (validator network), D2.10 (operational tooling)

---

**D2.9: Launch & First $5M Native BTC Volume ($5,000)** **Timeline**: Months 14-16 (3 months)
**Budget**: $5,000
**Team**: Marketing + community manager

**Scope**: - Marketing for Phase 2 launch - Press release announcing native BTC custody - Blog posts on threshold signatures - Twitter/X campaign - Target institutional users - Direct outreach to mining operations - Pitch to DeFi protocols - Pitch to Bitcoin treasury companies - Content creation - Case studies (use cases for native BTC loans) - Security documentation (audits, architecture) - Comparison: native BTC vs sBTC vs WBTC - Conference presentations - BTC Prague (if timing aligns) - Bitcoin Conference - Stacks ecosystem events - Community engagement - Discord/Telegram growth - AMA sessions with validators - Educational webinars

**Deliverables**: - [ ] $5M+ in native BTC loan volume by Month 16 - [ ] 15+ large loans (>$100K each) - [ ] 5+ institutional users (mining operations, treasuries) - [ ] 10+ press mentions or articles - [ ] 3+ conference presentations - [ ] Community growth: 2000+ Discord members

**Success Criteria**: - $5M volume milestone achieved by Month 16 end - Average loan size >$150K (institutional focus) - 50%+ of new loans use native BTC (vs Phase 1 sBTC) - 3+ major mining operations using protocol - Positive press coverage (no FUD) - Community engaged and active

**Marketing Timeline**:

```
Month 14: Pre-launch teasers, documentation
Month 15: Soft launch announcement, case studies
Month 16: Full launch PR, institutional outreach
```

**Dependencies**: All prior deliverables (network operational)

---

## 10. Technical Constraints

### 10.1 Bitcoin Constraints

**Bitcoin Script Limitations**: - No smart contracts (simple scripting only) - Multisig requires on-chain setup (expensive) - Taproot helps but still limited - Must use threshold signatures (off-chain aggregation)

**Block Time Variability**: - Bitcoin blocks: 10 min average (can vary 1-60 min) - Confirmation delays impact user experience - Must design UX around variable timing - Can't guarantee exact loan activation time

**Transaction Costs**: - Bitcoin fees spike during high activity - Median fee: $5-20 per transaction - Spikes: $50-100+ per transaction - Must reserve fee budget for each loan - RBF allows fee bumping if stuck

**Mitigation**: - Use SegWit and Taproot (lower fees) - Batch transactions where possible - Dynamic fee estimation - Allow manual fee bumping - Buffer for fee reserves

---

### 10.2 Stacks Constraints

**Block Time Coupling**: - Stacks blocks anchored to Bitcoin - Stacks block Bitcoin block 10 minutes - Cross-chain coordination slower than single-chain - Must design for ~10 min latencies

**Contract Upgradability**: - Phase 2 contracts should be immutable - If bugs found, must deploy new version - Migration process needed for users - No emergency pause (by design)

---

### 10.3 Threshold Cryptography Constraints

**Signing Complexity**: - Need 10/15 validators to sign - Requires network coordination - Latency: 1-5 minutes typical - If validators slow/down, delays possible

**Key Management**: - Each validator must securely store key share - Lost key shares = reduced threshold (bad) - Backup and recovery critical - No key rotation (would require new DKG)

**Threshold Fixed**: - 10-of-15 set at launch - Changing threshold requires new DKG - Can't easily add/remove validators - Future: need validator rotation mechanism

---

## 11. Migration from Phase 1

### 11.1 Backwards Compatibility

**Phase 1 Continues Operating**: - sBTC loans remain fully functional - No forced migration - Users choose when to migrate - Phase 1 and Phase 2 coexist indefinitely

**Why Keep Phase 1**: - Smaller loans (<$50K) may prefer Phase 1 (simplicity) - Users comfortable with sBTC can continue - No disruption to existing users - Gradual transition reduces risk

---

### 11.2 Migration Incentives

**For Borrowers**: - Better security (native BTC, no peg risk) - Institutional-grade custody - Larger loan amounts supported - Increased confidence for big positions

**For Lenders**: - Native Bitcoin custody (not wrapped) - Transparency (see multisig on Bitcoin) - Larger positions possible - Better for institutional compliance

**For Protocol**: - Demonstrates trustless custody capability - Positions for Phase 3 (cross-chain) - Attracts larger users - Removes sBTC dependency risk

---

### 11.3 Migration Support

**User Education**: - Blog posts explaining Phase 2 benefits - Video tutorials on migration process - FAQs addressing concerns - Case studies from early adopters

**Technical Support**: - Discord support channel - Migration assistance from team - Troubleshooting guides - White-glove service for large users

**Financial Incentives** (Optional): - Gas rebates for migrations - Lower fees for early Phase 2 users - Bonus for large volume migrations

---

## 12. Risk Assessment

### 12.1 Technical Risks

**Risk 1: Threshold Cryptography Bugs** **Description**: Critical bugs in threshold signature implementation could lead to inability to sign or, worse, key compromise.

**Likelihood**: Low (with professional audit)
**Impact**: Critical (complete system failure)
**Risk Score**: HIGH

**Mitigation Strategies**: - Use battle-tested threshold crypto library (FROST or MuSig2) - Independent cryptographer review (included in D2.1) - Comprehensive test suite (100% coverage) - Enhanced security audit with threshold specialists ($30K, D2.6) - Testnet deployment for months before mainnet - Formal verification of critical functions (if possible) - Bug bounty program post-launch ($50K+ rewards)

**Residual Risk**: Low after mitigations

---

**Risk 2: Validator Recruitment Failure NEW** **Description**: Unable to recruit 15 qualified, reliable validators by Month 10.

**Likelihood**: Medium (validator recruitment is challenging)
**Impact**: High (delays launch or reduces security)
**Risk Score**: HIGH

**Mitigation Strategies**: - Start recruitment early (Month 8, D2.1a) - Dedicated recruiter and $25K budget - Target 25+ candidates for 15 slots (funnel) - Geographic diversity incentives - Clear validator compensation model ($333+/month at $5M volume) - Backup plan: Launch with 12 validators (8-of-12 threshold) if needed - Partnership with existing node operators (Stacks, Bitcoin)

**Contingency**: - If <15 validators by Month 10: Extend timeline by 1 month - If <12 validators: Cannot launch safely, abort Phase 2

**Residual Risk**: Medium (recruitment is hard to predict)

---

**Risk 3: DKG Ceremony Failure NEW** **Description**: DKG ceremony fails, and validators cannot generate threshold keys.

**Likelihood**: Medium (15-party coordination complex)
**Impact**: High (delays launch by weeks)
**Risk Score**: HIGH

**Mitigation Strategies**: - 2+ testnet DKG rehearsals (D2.3a, $20K budget) - Detailed ceremony procedures and runbooks - Real-time coordinator support during ceremony - Byzantine fault tolerance (handle malicious validators) - Ceremony timeouts and retry logic - Backup ceremony date scheduled (if first attempt fails) - Post-ceremony verification and testing

**Contingency**: - If first ceremony fails: Analyze issues, schedule retry within 1 week - If testnet rehearsals fail: Do not proceed to mainnet until resolved

**Residual Risk**: Low (with 2+ testnet rehearsals)

---

**Risk 4: Cross-Chain State Inconsistency NEW** **Description**: Bitcoin and Stacks blockchain state becomes inconsistent, causing fund locks or losses.

**Likelihood**: Medium (distributed systems are complex)
**Impact**: Critical (user fund losses)
**Risk Score**: HIGH

**Mitigation Strategies**: - Enhanced state management module (D2.4a, $28K) - Handle Bitcoin reorgs up to 6 blocks - Atomic state updates (Bitcoin + Stacks together) - Comprehensive state verification tests (100+ scenarios) - Automated consistency checks (every block) - Alert on any inconsistencies (PagerDuty P1) - State recovery procedures (tested in D2.11)

**Contingency**: - If inconsistency detected: Emergency pause, investigate, recover from Bitcoin chain - If state unrecoverable: Manual intervention by validators

**Residual Risk**: Medium (complex distributed systems always have edge cases)

---

**Risk 5: Network Protocol Performance Issues NEW** **Description**: libp2p network protocol too slow, signatures take >30 seconds.

**Likelihood**: Low (libp2p is proven)
**Impact**: Medium (poor UX, but functional)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Network protocol implementation (D2.4b, $21K) - Performance benchmarks (signature time <10s target) - Network optimization (topology, redundant connections) - Extensive testing with 15 validators on testnet - Monitor network latency in production (Grafana) - Graceful degradation (if slow, still works)

**Contingency**: - If too slow: Optimize network topology, upgrade validator hardware - Target: <10s average, <20s p95

**Residual Risk**: Low (can optimize post-launch if needed)

---

**Risk 6: sBTC Peg Failure** **Description**: sBTC loses its peg to Bitcoin during Phase 2 development or transition.

**Likelihood**: Low (but not zero)
**Impact**: High (Phase 1 users affected, complicates migration)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Phase 2 eliminates reliance on sBTC (entire point of Phase 2) - Migration tools ready (D2.8) to move users to native BTC - Clear communication to users about sBTC risks - Phase 1 continues operating (users can repay loans) - Cannot prevent sBTC issues (systemic risk)

**Contingency**: - If sBTC depegs during Phase 2: Accelerate Phase 2 launch, prioritize migration - Communicate with Phase 1 users about migration incentives

**Residual Risk**: Medium (systemic sBTC risk until migration complete)

---

**Risk 7: Security Audit Delays** **Description**: Security audit takes longer than expected, delaying launch.

**Likelihood**: Medium (audits often delayed)
**Impact**: Medium (launch delay of 2-4 weeks)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Enhanced audit budget ($82K) includes contingency - Book audit firm early (Month 10) - Provide audit firm with complete code by Month 12 - Respond quickly to audit findings (dedicated dev time) - Plan for 2-month audit duration (accounts for delays)

**Contingency**: - If audit delayed: Accept delay, do NOT launch without audit - Security > speed

**Residual Risk**: Medium (audits are hard to predict)

---

### 12.2 Operational Risks

**Risk 8: Validator Downtime NEW** **Description**: Multiple validators go offline, reducing signing capacity.

**Likelihood**: Medium (operational issues happen)
**Impact**: Medium (slower signatures, but functional)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - 10-of-15 threshold (can tolerate 5 validators down) - Geographic distribution (no single region failure) - Operational tooling (D2.10) for proactive monitoring - 24/7 alerting (PagerDuty) for validator issues - Disaster recovery procedures (D2.11) for quick recovery - Validator SLAs (99.5% uptime required)

**Contingency**: - If 6+ validators down: Network still functional (10-of-15) - If 10+ validators down: Emergency procedures, contact all validators - If <10 validators: Signing stops, emergency intervention

**Residual Risk**: Low (10-of-15 threshold provides resilience)

---

**Risk 9: Operational Tooling Not Ready NEW** **Description**: Monitoring and operational tools not ready by launch, causing "blind" operations.

**Likelihood**: Low (dedicated deliverable D2.10)
**Impact**: High (poor incident response)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Dedicated operational tooling deliverable (D2.10, $15K, Month 13) - Deploy tooling before launch (Month 13) - Test all tools with validators in Month 13 - Train team on operational procedures - Block launch until tooling verified

**Contingency**: - If tooling not ready by Month 13 end: Delay launch by 2 weeks - Do NOT launch without monitoring

**Residual Risk**: Low (tooling is mandatory pre-launch)

---

**Risk 10: Disaster Recovery Procedures Untested NEW** **Description**: Disaster happens, and team doesn't know how to recover (key loss, validator failures).

**Likelihood**: Medium (disasters happen)
**Impact**: High (extended outage, loss of user confidence)
**Risk Score**: HIGH

**Mitigation Strategies**: - Dedicated disaster recovery testing (D2.11, $12K, Month 13) - Test key recovery procedures (3+ tests) - Test network partition recovery (5+ tests) - Test coordinator failover (2+ tests) - Document all procedures (step-by-step) - Block launch until DR tested

**Contingency**: - If disaster occurs: Follow documented procedures (D2.11) - If procedures fail: Emergency escalation, manual intervention

**Residual Risk**: Low (comprehensive testing reduces risk)

---

### 12.3 Business Risks

**Risk 11: Low Adoption (Users Prefer sBTC)** **Description**: Users don't see value in native BTC, continue using Phase 1 sBTC.

**Likelihood**: Low (native BTC is superior)
**Impact**: High (Phase 2 underutilized)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Clear communication of benefits (no peg risk, pure Bitcoin) - Migration tools make it easy (D2.8) - Institutional marketing (native BTC appeals to institutions) - Phase 1 continues operating (not forced migration) - Success metric: 30%+ migration rate (realistic target)

**Contingency**: - If adoption low (<20% by Month 16): Analyze barriers, address UX issues - Consider migration incentives (lower fees for native BTC loans)

**Residual Risk**: Low (native BTC is objectively better)

---

**Risk 12: Competitive Pressure** **Description**: Competitor launches similar native Bitcoin lending before we do.

**Likelihood**: Low (we're pioneering this space)
**Impact**: Medium (market share dilution)
**Risk Score**: LOW

**Mitigation Strategies**: - First-mover advantage (no known competitors doing threshold sigs) - Open-source builds trust and community - Strong Phase 1 foundation (user base and traction) - Validator network decentralization (moat) - Security audit and transparency (competitive advantage)

**Residual Risk**: Low

---

### 12.4 Regulatory & Legal Risks

**Risk 13: Regulatory Uncertainty** **Description**: Unclear regulatory status of decentralized Bitcoin custody.

**Likelihood**: Medium (DeFi regulation evolving)
**Impact**: High (could force shutdown or compliance)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Truly decentralized (15 independent validators) - No central entity controls funds (threshold signatures) - Open-source and transparent - No KYC/AML (permissionless protocol) - Legal consultation before launch - Monitor regulatory developments - Non-profit foundation structure (planned)

**Contingency**: - If regulation requires compliance: Explore options (validator licensing, etc.) - Geographic diversity helps (not all jurisdictions)

**Residual Risk**: Medium (regulatory landscape uncertain)

---

### 12.5 Financial Risks

**Risk 14: Budget Overruns** **Description**: Development costs exceed $463K budget.

**Likelihood**: Medium (complex project)
**Impact**: Medium (delays or scope cuts)
**Risk Score**: MEDIUM

**Mitigation Strategies**: - Detailed budget breakdown ($463K across 15 deliverables) - $17K contingency (4% of budget) - Prioritized deliverables (can cut D2.10, D2.11 if desperate) - Milestone-based funding (tranches reduce risk) - Regular budget reviews (monthly)

**Contingency**: - If budget overruns: Cut scope (reduce to 10 validators, skip DR testing) - Minimum viable Phase 2: ~$428K (saves $35K)

**Residual Risk**: Medium (budgets always have uncertainty)

---

**Risk 15: Funding Not Secured** **Description**: Unable to secure $463K funding from grants.

**Likelihood**: Medium (grant funding competitive)
**Impact**: High (cannot execute Phase 2)
**Risk Score**: HIGH

**Mitigation Strategies**: - Diversified funding strategy (HRF, Spiral, Stacks, OpenSats) - Strong Phase 1 track record (de-risks Phase 2) - Apply to multiple funders simultaneously - Clear value proposition (first native BTC lending) - Backup plan: Reduced scope if partial funding

**Contingency**: - If <$463K secured: Launch with 10 validators (8-of-10), skip DR testing - Minimum funding needed: ~$400K

**Residual Risk**: Medium (funding always uncertain)

---

### 12.6 Risk Matrix Summary

| Risk | Likelihood | Impact | Score | Mitigation |
|---|---|---|---|---|
| Threshold Crypto Bugs | Low | Critical | HIGH | Strong |
| **Validator Recruitment** | **Medium** | **High** | **HIGH** | Good |
| **DKG Ceremony Failure** | **Medium** | **High** | **HIGH** | Good |
| **State Inconsistency** | **Medium** | **Critical** | **HIGH** | Good |
| **Network Performance** | **Low** | **Medium** | **MEDIUM** | Good |
| sBTC Peg Failure | Low | High | MEDIUM | Accepted |
| Security Audit Delays | Medium | Medium | MEDIUM | Good |
| **Validator Downtime** | **Medium** | **Medium** | **MEDIUM** | Strong |
| **Tooling Not Ready** | **Low** | **High** | **MEDIUM** | Strong |
| **DR Procedures Untested** | **Medium** | **High** | **HIGH** | Strong |
| Low Adoption | Low | High | MEDIUM | Good |
| Competitive Pressure | Low | Medium | LOW | Good |

| Risk | Likelihood | Impact | Score | Mitigation |
|---|---|---|---|---|
| Regulatory Uncertainty | Medium | High | MEDIUM | Monitor |
| Budget Overruns | Medium | Medium | MEDIUM | Good |
| Funding Not Secured | Medium | High | HIGH | Good |

**Overall Risk Assessment**: MEDIUM-HIGH
**Risk Tolerance**: Acceptable for Phase 2 with strong mitigation strategies
**New Risks Added**: 6 risks related to enhanced scope (validator recruitment, DKG, state management, network, tooling, DR)

---

## 13. Timeline and Budget

### 13.1 Phase 2 Overview

**Timeline**: 9 months (Months 8-16)
**Budget**: $463,000
**Start**: Month 8 (after Phase 1 completes Month 7)
**Launch**: Month 14 (soft), Month 16 (full)
**Buffer**: Months 17-18 (2-month stabilization)
**Phase 3 Start**: Month 18

**Why 9 Months?** (was 7 months) - +2 months for validator recruitment and onboarding - +1 month for operational tooling and disaster recovery - Dedicated soft launch month before full launch - More realistic timeline for complex distributed system

**Why $463K?** (was $312K) - +$91K development (6 new deliverables) - +$30K enhanced security audit - +$33K operations (tooling + DR testing) - Total: +$151K for production-ready system

---

### 13.2 Phase 2 Gantt Chart (9 Months)

```
MONTH 8: Foundation & Recruitment (Start after Phase 1 complete)

Week 1-4:   D2.1  Threshold Crypto Library
Week 1-12:  D2.1a Validator Recruitment BEGIN
Week 2-12:  D2.2  Validator Software Start
```

```
Deliverables Due: D2.1 Complete
Budget Spent: $45K (10% of total)


MONTH 9: Development & Network Building

Week 1-8:   D2.2  Validator Software Continue
Week 1-8:   D2.3  DKG Implementation
Week 1-12:  D2.4  Bridge Logic Start
Week 1-8:   D2.1a Validator Recruitment Continue

Deliverables Due: D2.3 Complete
Budget Spent: $115K cumulative (25%)


MONTH 10: Integration & Testing

Week 1-4:   D2.2  Validator Software Complete
Week 1-4:   D2.1a Validator Recruitment Complete
Week 1-12:  D2.3a DKG Coordination BEGIN
Week 1-8:   D2.4  Bridge Logic Continue
Week 1-8:   D2.4a State Management BEGIN
Week 1-8:   D2.4b Network Protocol BEGIN

Deliverables Due: D2.2, D2.1a Complete
Budget Spent: $202K cumulative (44%)


MONTH 11: State Management & Network Protocols

Week 1-4:   D2.4  Bridge Logic Complete
Week 1-4:   D2.4a State Management Complete
Week 1-4:   D2.4b Network Protocol Complete
Week 1-8:   D2.5  Validator Network Launch BEGIN
Week 1-12:  D2.7  Frontend Updates BEGIN
Week 1-8:   D2.3a DKG Coordination Continue

Deliverables Due: D2.4, D2.4a, D2.4b Complete
Budget Spent: $297K cumulative (64%)


MONTH 12: Security & Coordination

Week 1-4:   D2.5  Validator Network Deploy
Week 1:     D2.3a Mainnet DKG Ceremony
Week 2-4:   D2.3a Post-DKG Verification
Week 1-8:   D2.6  Security Audit BEGIN
Week 1-12:  D2.7  Frontend Updates Continue

Deliverables Due: D2.5, D2.3a Complete
Budget Spent: $393K cumulative (85%)


MONTH 13: Audit, Tooling & Final Prep

Week 1-2:   D2.6  Security Audit Complete
Week 2-3:   D2.6  Address Audit Findings
Week 4:     D2.6  Follow-up Re-audit
Week 1-4:   D2.7  Frontend Updates Complete
Week 1-8:   D2.8  Migration Tools BEGIN
Week 1-4:   D2.10 Operational Tooling
Week 1-4:   D2.11 Disaster Recovery Testing

Deliverables Due: D2.6, D2.7, D2.10, D2.11 Complete
Budget Spent: $451K cumulative (97%)


MONTH 14: Launch Preparation & Migration

Week 1-2:   Pre-launch Validation
Week 2-3:   Mainnet Deployment
Week 3-4:   D2.8  Migration Tools Complete
Week 1-4:   D2.9  Marketing & Launch
Week 4:       SOFT LAUNCH

Deliverables Due: D2.8 Complete, Soft Launch
Budget Spent: $458K cumulative (99%)


MONTH 15: Soft Launch & Early Adoption

Week 1-4:   D2.9  First Native BTC Loans
Week 1-4:   Close Monitoring (24/7)
Week 1-4:   User Onboarding
Week 1-4:   Migration Support

Target: $500K-$1M volume, 5+ loans, zero critical bugs
Budget Spent: $461K cumulative (99.5%)


MONTH 16: Full Launch & Growth
```

```
Week 1:       FULL PUBLIC LAUNCH
Week 1-4:   D2.9  Volume Growth
Week 1-4:   Institutional Outreach
Week 1-4:   Migration Push
Week 4:       $5M VOLUME MILESTONE

Deliverables Due: D2.9 Complete ($5M volume achieved)
Budget Spent: $463K (100%)


MONTHS 17-18: Buffer Period (Stabilization)

Month 17-18: Continuous Operation
             Monitor Validator Performance
             Optimize Based on Usage
             Prepare for Phase 3

Target: 99.9% uptime, no critical incidents, Phase 3 planning


MONTH 18: Phase 3 Begins

Phase 3 Multi-Chain Expansion Starts
```

---

### 13.3 Detailed Budget Breakdown ($463,000)

**Category 1: Development ($283,000 - 61%)**

| Item | Cost | Duration | Description |
|---|---|---|---|
| **Threshold Crypto (D2.1)** | $45,000 | 1 month | FROST/MuSig2 integration |
| **Validator Software (D2.2)** | $62,000 | 3 months | Rust application, monitoring |
| **DKG Implementation (D2.3)** | $28,000 | 2 months | Key generation ceremony |
| **Bridge Logic (D2.4)** | $45,000 | 3 months | Stacks Bitcoin integration |

| Item | Cost | Duration | Description |
|---|---|---|---|
| **Frontend Updates (D2.7)** | $25,000 | 3 months | Native BTC UI, validator dashboard |
| **Migration Tools (D2.8)** | $12,000 | 2 months | Phase 1 → Phase 2 migration |
| **Validator Recruitment (D2.1a)** | $25,000 | 3 months | **NEW** - Recruit 15 validators |
| **State Management (D2.4a)** | $28,000 | 2 months | **NEW** - Reorg handling, state sync |
| **Network Protocol (D2.4b)** | $21,000 | 2 months | **NEW** - libp2p, gossip protocol |
| **SUBTOTAL** | **$283,000** | | |

**Changes from Original**: +$91K for 3 new deliverables

---

**Category 2: Security & Audit ($82,000 - 18%)**

| Item | Cost | Duration | Description |
|---|---|---|---|
| **Threshold Crypto Audit** | $30,000 | 2 weeks | Specialist audit (ENHANCED) |
| **Validator Network Audit** | $22,000 | 2 weeks | Distributed systems security |
| **Bridge Logic Audit** | $15,000 | 1 week | Cross-chain security |
| **Economic Attack Analysis** | $10,000 | 1 week | Game theory, incentives |
| **Follow-up Re-audit** | $5,000 | 1 week | Fix verification |
| **SUBTOTAL** | **$82,000** | | |

**Changes from Original**: +$30K for enhanced scope (was $52K)

---

**Category 3: Operations ($76,000 - 16%)**

| Item | Cost | Duration | Description |
|---|---|---|---|
| **DKG Coordination (D2.3a)** | $20,000 | 3 months | **NEW** - Ceremony coordination |
| **Validator Network Launch (D2.5)** | $38,000 | 2 months | Deploy 15 validators, mainnet |
| **Operational Tooling (D2.10)** | $15,000 | 1 month | **NEW** - Grafana, PagerDuty |
| **Disaster Recovery (D2.11)** | $12,000 | 1 month | **NEW** - Testing procedures |
| **SUBTOTAL** | **$76,000** | | |

**Changes from Original**: +$33K for 3 new deliverables (was $43K)

---

**Category 4: Marketing ($5,000 - 1%)**

| Item | Cost | Duration | Description |
|---|---|---|---|
| **Launch Marketing (D2.9)** | $5,000 | 3 months | PR, content, outreach |
| **SUBTOTAL** | **$5,000** | | |

**No Change from Original**

---

**Category 5: Contingency ($17,000 - 4%)**

| Item | Cost | Description |
|---|---|---|
| **Contingency Reserve** | $17,000 | 4% buffer for overruns |
| **SUBTOTAL** | **$17,000** | |

**Changes from Original**: Reduced from $20K (3% reduction)

**Total Phase 2 Budget: $463,000**

| Category | Amount | % of Total | Change |
|---|---|---|---|
| Development | $283,000 | 61% | +$91K |
| Security & Audit | $82,000 | 18% | +$30K |
| Operations | $76,000 | 16% | +$33K |
| Marketing | $5,000 | 1% | – |
| Contingency | $17,000 | 4% | -$3K |
| **TOTAL** | **$463,000** | **100%** | **+$151K** |

### 13.4 Funding Strategy (Revised)

**Target Funding**: $463,000

**Recommended Funding Sources**

| Funder | Target Ask | Likelihood | Timeline | Focus |
|---|---|---|---|---|
| **HRF** | $200,000 | High (70%) | 3-4 months | Bitcoin freedom tech, decentralization |
| **Spiral (Block)** | $200,000 | Medium (60%) | 3-4 months | Bitcoin infrastructure, threshold sigs |
| **Stacks Foundation** | $50,000 | Medium (50%) | 2-3 months | Stacks ecosystem, bridge logic |
| **OpenSats** | $13,000 | Low (40%) | 3-4 months | Open-source Bitcoin projects |

**Total Potential**: $463,000

**Application Strategy**: - **Month 7**: Submit HRF and Spiral applications simultaneously - **Month 8**: Submit Stacks Foundation application - **Month 8**: Submit OpenSats application - **Month 9-10**: Follow up with all funders - **Month 11**: Expect funding decisions

**Why Split Applications?** - Smaller asks = higher success probability - Different pitches to different funders: - **HRF**: Freedom tech, decentralization, censorship resistance - **Spiral**: Bitcoin infrastructure, threshold signatures innovation - **Stacks**: Bridge logic, Stacks ecosystem integration - **OpenSats**: Open-source contribution to Bitcoin - Diversifies funding risk (not dependent on single funder)

---

### 13.5 Milestone-Based Fund Release (Recommended)

**Tranche Structure** (for funders):

**Tranche 1 (40% = $185,200)**: Upon grant approval - Deliverables: D2.1, D2.1a, D2.2, D2.3 (foundation work) - Timeline: Months 8-10 - Justification: Upfront investment in crypto library, software, recruitment

**Tranche 2 (40% = $185,200)**: Mid-phase checkpoint - Deliverables: D2.4, D2.4a, D2.4b, D2.5 (network operational) - Milestone: Testnet DKG successful, all 15 validators online - Timeline: Months 11-12 - Justification: Major technical milestones achieved

**Tranche 3 (20% = $92,600)**: Post-launch success - Deliverables: D2.6, D2.7, D2.8, D2.9, D2.10, D2.11 (launch + volume) - Milestone: Mainnet launched, $5M volume achieved - Timeline: Months 13-16 - Justification: Success demonstrated

**Benefits**: - De-risks funding for grant providers - Ensures accountability and progress - Aligns incentives (team funded as deliverables complete)

---

### 13.6 Contingency Planning (If Funding Falls Short)

**Scenario: Only $428K secured** (vs $463K target)

**Cost-Saving Measures** (save $35K): 1. Reduce validators to 12 (8-of-12 threshold) - saves $15K (D2.1a, D2.5) 2. Skip disaster recovery testing (D2.11) - saves $12K 3. Reduce operational tooling scope (D2.10) - saves $8K

**Impact**: - 12 validators still secure (8-of-12 = 67% threshold) - Disaster recovery procedures documented but not tested - Basic monitoring only (no advanced tooling)

**Minimum Viable Phase 2**: $428K

---

**Scenario: Only $400K secured** (vs $463K target)

**Additional Cuts** (save $63K total): - Above measures ($35K) - Reduce DKG coordination (D2.3a) - saves $10K (self-coordinate) - Reduce state management (D2.4a) - saves $10K (basic reorg handling) - Skip migration tools (D2.8) - saves $12K (manual migration)

**Impact**: - Launch possible but with reduced features - Higher operational risk - Manual processes instead of automation

**Absolute Minimum**: $400K (anything less, delay Phase 2)

---

### 13.7 Phase 2 Success Criteria (Ready for Phase 3)

**Technical Success**: - [ ] All 15 validators operational (or 12 minimum) - [ ] 99.9% network uptime for 2 months (Months 17-18) - [ ] Zero critical bugs or security incidents - [ ] Threshold signatures 100% success rate - [ ] Cross-chain state consistency maintained

**Business Success**: - [ ] $5M+ native BTC loan volume achieved - [ ] 15+ borrowers, 30+ lenders - [ ] 50%+ of new loans use native BTC (vs Phase 1) - [ ] 30%+ of Phase 1 loans migrated - [ ] Validator network profitable ($333+/month per validator)

**Operational Success**: - [ ] All operational tooling deployed and functional - [ ] Disaster recovery procedures tested and documented - [ ] 24/7 monitoring and alerting operational - [ ] Incident response procedures proven effective

**Phase 3 Ready if**: - All above criteria met - 2 months continuous operation (buffer period) - Team ready for multi-chain expansion - Funding secured for Phase 3 ($500K)

---

### 13.8 Phase 2 → Phase 3 Transition

**Buffer Period**: Months 17-18 (2 months)

**Purpose**: - Stabilize Phase 2 operations - Monitor validator performance - Optimize based on usage patterns - Prepare for Phase 3 (multi-chain expansion)

**Activities During Buffer**: - Continuous monitoring of 15 validators - Performance tuning and optimization - Community feedback collection - Phase 3 planning and design - Phase 3 grant applications

**Phase 3 Start**: Month 18 (not Month 16)

**Why Buffer?** - Phase 3 is even more complex (multi-chain) - Need proven Phase 2 stability before expanding - Gives time to secure Phase 3 funding - Reduces risk of cascading failures

## Appendix A: Threshold Signature Primer

### What is Threshold Cryptography?

**Traditional Multisig** (Bitcoin):

```
3-of-5 Multisig:
- Generate 5 separate private keys
- Derive 5 public keys
- Create multisig address from the 5 public keys
- To spend: Need 3 of the 5 private keys to sign

Problem: All 5 keys and signatures visible on-chain
Size: Large (multiple signatures = more bytes = higher fees)
```

**Threshold Signatures** (Schnorr/FROST):

```
3-of-5 Threshold:
- Generate 1 shared private key (distributed among 5 parties)
- Each party holds a "key share" (none have full key)
- Derive 1 public key
- Create single-sig address from public key (looks normal!)
- To spend: 3 parties create "partial signatures"
- Combine partial signatures into 1 final signature

Benefit: Only 1 signature on-chain (privacy + efficiency)
Size: Same as normal single-sig (lower fees)
Privacy: No one knows it's multisig!
```

### Why Use Threshold Signatures?

1. **Privacy**: On-chain looks like normal Bitcoin address (Taproot)
2. **Efficiency**: Single signature = lower fees
3. **Security**: No single party has full key
4. **Flexibility**: Threshold can be any M-of-N
5. **Decentralization**: Distributed trust across validators

### How It Works (Simplified)

**Key Generation**:

```
1. Each validator generates a random secret (s_i)
2. Each validator computes their public key share (P_i = s_i * G)
3. All public key shares are combined: P = P_1 + P_2 + ... + P_N
4. P is the combined public key (Bitcoin address derived from this)
5. Nobody ever has the combined private key!
```

**Signing**:

```
1. Validators agree on message to sign (Bitcoin transaction)
2. Each validator creates a partial signature using their secret share
3. Coordinator collects M partial signatures (e.g., 10 of 15)
4. Coordinator aggregates partial signatures into final signature
5. Final signature is valid for public key P
6. Broadcast Bitcoin transaction with signature
```

### Security Properties

**Threshold Property**: - Need M parties to sign (e.g., 10 of 15) - Any M parties can create valid signature - Fewer than M parties cannot sign

**No Single Point of Failure**: - No single party has full key - Must compromise M parties to steal funds - Much harder than compromising 1 custodian

**Verifiable**: - Anyone can verify signatures are correct - Partial signatures can be checked before aggregation - Faulty/malicious validators detected

---

## Appendix B: Validator Operations Manual

### Validator Setup

**Prerequisites**: - Dedicated server (32GB RAM, 8 CPU, 2TB SSD) - Bitcoin Core installed and synced - Stacks Node installed and synced - HSM for key storage (recommended)

**Installation**:

```
# Download validator binary
wget https://github.com/btc-lending/validator/releases/v2.0.0/validator

# Create config file
cat > config.toml <<EOF
[bitcoin]
rpc_url = "http://localhost:8332"
rpc_user = "user"
rpc_pass = "pass"

[stacks]
rpc_url = "http://localhost:20443"

[validator]
key_storage = "hsm"  # or "file"
hsm_slot = 0
p2p_port = 9001
```

```
metrics_port = 9090
EOF

# Run validator
./validator --config config.toml
```

**DKG Ceremony Participation**:

```
# When DKG announced, join ceremony
./validator join-dkg --ceremony-id 12345

# Follow prompts, verify other participants
# Ceremony completes automatically
# Key share encrypted and stored securely
```

### Daily Operations

**Monitoring**:

```
# Check validator status
./validator status

# View recent signatures
./validator signatures --count 10

# Check connectivity to other validators
./validator peers
```

**Logs**:

```
# View logs
tail -f /var/log/validator/validator.log

# Look for errors or warnings
grep ERROR /var/log/validator/validator.log
```

**Metrics**: - Prometheus metrics at http://localhost:9090/metrics - Grafana dashboards for visualization - Alert manager for issues

### Incident Response

**Validator Down**: 1. Check if Bitcoin/Stacks nodes synced 2. Restart validator software 3. Verify connectivity to peers 4. Monitor signing participation 5. Contact coordinator if issues persist

**Key Recovery**: 1. Retrieve encrypted key backup 2. Verify identity with coordinator 3. Restore key share to HSM 4. Resume operations 5. Verify signing works correctly

**Emergency Shutdown**:

```
# Graceful shutdown
./validator shutdown --graceful

# Emergency stop (only if needed)
kill -9 $(pgrep validator)
```

---

## Appendix C: Economic Model

### Validator Incentives

**Revenue**: - Protocol fee: 0.1% of loan volume - Example: $5M volume in Month 13 = $5,000 total fees - Split: $5,000 / 15 validators = $333 per validator per month

**Costs**: - Hardware: $800 one-time - Hosting: $100/month - Electricity: $20/month - Maintenance: $50/month (time) - Total monthly: ~$170

**Net Profit**: $333 - $170 = **$163/month per validator**

**Break-Even Analysis**: - Hardware payback: $800 / $163 = ~5 months - After 5 months: $163/month pure profit - Annual profit (Year 1): $163 × 7 months = $1,141 - Annual profit (Year 2+): $163 × 12 = $1,956

**Scaling**: - $10M volume → $665/month → $495/month profit - $50M volume → $3,333/month → $3,163/month profit - $100M volume → $6,666/month → $6,496/month profit

**Conclusion**: Validators become profitable at ~$5M+ volume, highly profitable at $50M+.

---

## Appendix D: Glossary

**DKG (Distributed Key Generation)**: Protocol to generate threshold keys without any single party knowing the full key

**FROST**: Flexible Round-Optimized Schnorr Threshold signatures (threshold signature scheme)

**HSM (Hardware Security Module)**: Physical device for storing cryptographic keys securely

**Partial Signature**: Signature fragment created by one validator using their key share

**Taproot**: Bitcoin upgrade enabling efficient threshold signatures (Schnorr-based)

**Threshold Signature**: Single signature created by M-of-N parties without reconstructing private key

**Validator**: Node operator who holds a key share and participates in threshold signing

**VSS (Verifiable Secret Sharing)**: Cryptographic technique ensuring key shares are correctly distributed

---

**End of Product Requirements Document - Phase 2**

**Document Version**: 1.0
**Last Updated**: January 12, 2026

---

*This PRD defines Phase 2 of the Bitcoin Lending Protocol, implementing Bitcoin-native custody with threshold signatures to eliminate wrapped token dependencies while maintaining the oracle-free competitive bidding innovation from Phase 1.*