M BridgeConnect

MBridge AI Technical Whitepaper
Version 1.3

Executive Summary

MBridge AI introduces a novel tokenized system that maintains stability through BTC reserves while offering controlled borrowing capabilities. The system employs sophisticated caps and limits to ensure long-term sustainability.

Core Technical Architecture

1. Reserve Management System

BTC Reserve Structure

- 70% of all token sales converted to BTC reserves
- Real-time reserve tracking and verification
- Multi-signature custody system for reserve management

Reserve Ratios

- Minimum Reserve Ratio: 60% of total token market cap
- Maximum Borrowable Reserve: 50% of total BTC reserves
- Emergency Reserve: 20% of total reserves (never used for lending or buybacks)

2. Borrowing Mechanism

System-Wide Limits

- Maximum System Borrow Cap: 30% of available BTC reserves
- Minimum Reserve Buffer: 40% of total reserves must remain unborrowed
- Dynamic adjustment based on market conditions

Individual Borrowing Limits

- Maximum per wallet: 2% of total available borrowing capacity
- Minimum borrow amount: 0.1 BTC
- Maximum borrow amount: 5 BTC
- Collateral ratio: 30% (Example: $10,000 in tokens can borrow $3,000 in BTC)

Borrowing Parameters

- Interest rate: 5% APR, paid in MBridge tokens
- Maximum loan duration: 90 days
- Early repayment allowed with no penalty
- Automatic liquidation at 85% collateral ratio

3. Buyback Mechanism

Trigger Conditions

- Initiated when token price drops 30% below 7-day moving average
- Maximum daily buyback: 1% of total market cap
- Minimum price impact requirements

Execution Parameters

- Buyback increment: 0.1% of daily volume per transaction
- Minimum time between buybacks: 1 hour
- Maximum slippage: 1%
- Smart contract-controlled execution

4. Token Distribution

Initial Sale Structure Phase 1:

- Price: $1 per token
- Allocation: 10 million tokens
- Minimum purchase: $3,000 HKD
- Maximum purchase: $100,000 HKD

Phase 2:

- Price: $1.20 per token
- Public sale through DEX
- No individual limits

5. Interest Distribution

- 90% of interest used for token burns and buybacks
  - 45% direct token burns
  - 45% market buybacks
- 10% to operational treasury

6. Safety Features

Circuit Breakers

- Borrowing suspended if reserves drop below 40%
- Automatic buyback suspension if daily limit reached
- Emergency shutdown capability with multi-sig

Risk Management

- Real-time monitoring of reserve ratios
- Automatic liquidation engine
- Oracle price feed redundancy

Smart Contract Architecture

```solidity
// Core contracts structure
contract MBridgeToken {
    uint256 public constant TOTAL_SUPPLY = 1_000_000_000 * 10**18;
    uint256 public constant MAX_SYSTEM_BORROW = 100 * 10**18; // 100 BTC
    uint256 public constant MAX_USER_BORROW = 5 * 10**18;    // 5 BTC
    uint256 public constant MIN_BORROW = 0.1 * 10**18;      // 0.1 BTC

    // Borrowing state
    mapping(address => uint256) public userBorrows;
    uint256 public totalBorrows;

    // Reserve tracking
    uint256 public btcReserves;
    uint256 public minimumReserves;

    // Buyback parameters
    uint256 public constant DAILY_BUYBACK_LIMIT = 1_000_000 * 10**18;
    uint256 public constant BUYBACK_INTERVAL = 1 hours;

    // System state
    bool public borrowingPaused;
    bool public buybackPaused;
}
```

Implementation Timeline

Phase 1: Q1 2025

- Smart contract development and auditing
- Reserve management system implementation
- Initial security features

Phase 2: Q2 2025

- Borrowing mechanism deployment
- Initial token sale
- Basic buyback system

Phase 3: Q3 2025

- Advanced buyback mechanisms
- Enhanced monitoring systems
- Public sale launch

Phase 4: Q4 2025

- Full system optimization
- Advanced risk management features
- Complete decentralization of key functions

Risk Mitigation Strategies

1. **Reserve Protection**
- Multi-signature requirements for reserve management
- Time-locked transactions for large reserve movements
- Regular third-party audits
2. **Market Protection**
- Gradual buyback system
- Price impact limitations
- Circuit breakers
3. **User Protection**
- Clear liquidation parameters
- Transparent reserve reporting
- Emergency withdrawal mechanisms

Conclusion

MBridge AI's technical architecture prioritizes security, stability, and sustainable growth through carefully designed parameters and risk management systems. The combination of controlled borrowing, systematic buybacks, and reserve management creates a robust ecosystem for long-term value preservation.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";

contract MBridgeToken is ERC20, ReentrancyGuard, Pausable, AccessControl {
    bytes32 public constant MANAGER_ROLE = keccak256("MANAGER_ROLE");

    // System Constants
    uint256 public constant TOTAL_SUPPLY = 1_000_000_000 * 10**18; // 1 billion tokens
    uint256 public constant MAX_SYSTEM_BORROW = 100 * 10**18;      // 100 BTC
    uint256 public constant MAX_USER_BORROW = 5 * 10**18;         // 5 BTC
    uint256 public constant MIN_BORROW = 0.1 * 10**18;            // 0.1 BTC
    uint256 public constant COLLATERAL_RATIO = 30;                // 30%
    uint256 public constant LIQUIDATION_THRESHOLD = 85;           // 85%

    // Borrowing state
    mapping(address => uint256) public userBorrows;
    uint256 public totalBorrows;

    // Reserve tracking
```

```
uint256 public btcReserves;
uint256 public minimumReserves;

// Buyback parameters
uint256 public constant DAILY_BUYBACK_LIMIT = 1_000_000 * 10**18;
uint256 public lastBuybackTime;
uint256 public constant BUYBACK_INTERVAL = 1 hours;

// Price tracking
uint256 public btcPrice;
uint256 public tokenPrice;

// Events
event Borrowed(address indexed user, uint256 amount);
event Repaid(address indexed user, uint256 amount);
event ReservesUpdated(uint256 newReserves);
event BuybackExecuted(uint256 amount, uint256 price);

constructor() ERC20("MBridge AI Token", "MBR") {
    _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
    _setupRole(MANAGER_ROLE, msg.sender);
    _mint(msg.sender, TOTAL_SUPPLY);
}

// Borrowing functions
function borrow(uint256 amount) external nonReentrant whenNotPaused {
    require(amount >= MIN_BORROW, "Below minimum borrow amount");
    require(amount <= MAX_USER_BORROW, "Exceeds maximum borrow amount");
    require(totalBorrows + amount <= MAX_SYSTEM_BORROW, "System borrow limit reached");
    require(userBorrows[msg.sender] + amount <= MAX_USER_BORROW, "User borrow limit reached");

    uint256 requiredCollateral = calculateRequiredCollateral(amount);
    require(balanceOf(msg.sender) >= requiredCollateral, "Insufficient collateral");

    userBorrows[msg.sender] += amount;
    totalBorrows += amount;

    emit Borrowed(msg.sender, amount);
}

function repay(uint256 amount) external nonReentrant {
    require(userBorrows[msg.sender] >= amount, "Repay amount exceeds borrow");

    userBorrows[msg.sender] -= amount;
    totalBorrows -= amount;

    // Handle interest payment and burning
    uint256 interest = calculateInterest(amount);
    _burn(msg.sender, interest);

    emit Repaid(msg.sender, amount);
}

// Reserve management
function updateReserves(uint256 newReserves) external onlyRole(MANAGER_ROLE) {
    btcReserves = newReserves;
    minimumReserves = (newReserves * 40) / 100; // 40% minimum reserve

    emit ReservesUpdated(newReserves);
}

// Buyback functions
function executeBuyback(uint256 amount) external onlyRole(MANAGER_ROLE) {
    require(block.timestamp >= lastBuybackTime + BUYBACK_INTERVAL, "Buyback too soon");
```

```
        require(amount <= DAILY_BUYBACK_LIMIT, "Exceeds daily buyback limit");

        lastBuybackTime = block.timestamp;

        emit BuybackExecuted(amount, tokenPrice);
    }

    // Internal functions
    function calculateRequiredCollateral(uint256 borrowAmount) internal view returns (uint256) {
        return (borrowAmount * 100) / COLLATERAL_RATIO;
    }

    function calculateInterest(uint256 amount) internal pure returns (uint256) {
        return (amount * 5) / 100; // 5% interest
    }

    // Admin functions
    function pause() external onlyRole(DEFAULT_ADMIN_ROLE) {
        _pause();
    }

    function unpause() external onlyRole(DEFAULT_ADMIN_ROLE) {
        _unpause();
    }

    // Price oracle functions
    function updatePrices(uint256 _btcPrice, uint256 _tokenPrice) external onlyRole(MANAGER_ROLE) {
        btcPrice = _btcPrice;
        tokenPrice = _tokenPrice;
    }
}
```