Fondo Común: A Decentralized Micro-Lending Protocol on Massa Network

Cosmas Ken

July 2025

Abstract

Fondo Común is a decentralized finance (DeFi) protocol on the Massa Network, implementing a trustless micro-lending system via a shared liquidity pool. Utilizing Massa's Autonomous Smart Contracts (ASC) for deterministic, autonomous repayment scheduling and DeWeb for an on-chain React-based frontend, the protocol ensures fault-tolerant, intermediary-free operation. This whitepaper provides a rigorous technical specification, including smart contract logic, frontend architecture, and cryptographic guarantees, targeting the $200B microfinance market's accessibility challenges.

1 Introduction

Microfinance serves over 200 million users globally, yet DeFi lending protocols (e.g., Aave, Compound) enforce over-collateralization (typically 150-200%), excluding low-income users. Centralized microfinance platforms introduce single points of failure and intermediary risks. Fondo Común leverages Massa Network's layer-1 capabilities to deliver a trustless micro-lending pool, supporting loans of 0.01-0.1 ETH equivalent with autonomous repayment enforcement and a fully on-chain, React-based frontend via DeWeb.

2 System Architecture

Fondo Común operates as a single smart contract with an on-chain React frontend, integrated with Massa's ASC and DeWeb. The system supports:

- **Liquidity Pool**: Aggregates user deposits for lending.
- **Micro-Loans**: Loans with fixed 7-day terms, enforced by ASC.
- **Frontend**: React SPA on DeWeb, interfacing with the contract via Massa SDK.

2.1 Threat Model

The protocol assumes:

- Honest-but-curious users who follow the protocol but may attempt to exploit state inconsistencies.
- Potential denial-of-service (DoS) attacks on contract calls, mitigated by Massa's high-throughput consensus.
- No trusted third parties; all execution is on-chain.

3 Smart Contract Specification

The FondoComun contract, written in a Solidity-like language for Massa, manages deposits, loans, and repayments. Key state variables and functions are defined below.

3.1 State Variables

- `mapping(address => uint256) balances`: Tracks user deposits.
- `mapping(address => uint256) loans`: Tracks active loans per user.
- `uint256 poolBalance`: Total available liquidity.
- `uint256 constant LOAN_DURATION = 7 days`: Fixed loan term.
- `uint256 constant MAX_LOAN_AMOUNT = 0.1 ether`: Maximum loan size.
- `uint256 constant MIN_LOAN_AMOUNT = 0.01 ether`: Minimum loan size.

3.2 Core Functions

Algorithm 1 Fondo Común Smart Contract Logic

```
function DEPOSIT
Require: msg.value $> 0$
    balances[msg.sender] += msg.value
    poolBalance += msg.value
    emit Deposited(msg.sender, msg.value)
end function
function BORROW(uint256 amount)
Require: amount $\geq$ MIN_LOAN_AMOUNT
Require: amount $\leq$ MAX_LOAN_AMOUNT
Require: amount $\leq$ poolBalance
Require: loans[msg.sender] $= 0$
    loans[msg.sender] = amount
    poolBalance -= amount
    payable(msg.sender).transfer(amount)
    emit Borrowed(msg.sender, amount)
    scheduleRepayment(msg.sender, amount, block.timestamp + LOAN_DURATION)
end function
function REPAY(address borrower)
Require: loans[borrower] $> 0$
Require: balances[borrower] $\geq$ loans[borrower]
    amount = loans[borrower]
    balances[borrower] -= amount
    poolBalance += amount
    loans[borrower] = 0
    emit Repaid(borrower, amount)
end function
function GETSTATUS(address user)
    return (poolBalance, balances[user], loans[user])
end function
```

3.3 Autonomous Smart Contracts (ASC)

Massa's ASC enables the `scheduleRepayment` function to execute `repay` deterministically after `LOAN_DURATION`. This leverages Massa's parallel execution model, ensuring low-latency scheduling without external oracles. The repayment process is gas-optimized, with state updates batched to minimize blockchain overhead.

Listing 1: Fondo Común Smart Contract

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

contract FondoComun {
    mapping(address => uint256) public balances;
    mapping(address => uint256) public loans;
    uint256 public poolBalance;
    uint256 public constant LOAN_DURATION = 7 days;
    uint256 public constant MAX_LOAN_AMOUNT = 0.1 ether;
    uint256 public constant MIN_LOAN_AMOUNT = 0.01 ether;

    event Deposited(address indexed user, uint256 amount);
    event Borrowed(address indexed borrower, uint256 amount);
    event Repaid(address indexed borrower, uint256 amount);

    function deposit() external payable {
        require(msg.value > 0, "Deposit must be greater than 0");
        balances[msg.sender] += msg.value;
        poolBalance += msg.value;
        emit Deposited(msg.sender, msg.value);
    }

    function borrow(uint256 amount) external {
        require(amount >= MIN_LOAN_AMOUNT, "Amount below minimum");
        require(amount <= MAX_LOAN_AMOUNT, "Amount exceeds maximum");
        require(amount <= poolBalance, "Insufficient pool funds");
        require(loans[msg.sender] == 0, "Existing loan must be repaid");

        loans[msg.sender] = amount;
        poolBalance -= amount;
        payable(msg.sender).transfer(amount);
        emit Borrowed(msg.sender, amount);
        // Massa ASC schedules repayment
        scheduleRepayment(msg.sender, amount, block.timestamp + LOAN_DURATION);
    }

    function repay(address borrower) external {
        uint256 amount = loans[borrower];
        require(amount > 0, "No active loan");
        require(balances[borrower] >= amount, "Insufficient funds");

        balances[borrower] -= amount;
        poolBalance += amount;
        loans[borrower] = 0;
        emit Repaid(borrower, amount);
    }

    function getStatus(address user) external view returns (uint256, uint256,
        uint256) {
        return (poolBalance, balances[user], loans[user]);
    }
}
```

3.4 Security Considerations

- **Reentrancy**: Mitigated by updating state (`poolBalance`, `loans`) before external calls (`transfer`).
- **Overflow/Underflow**: Solidity 0.8.0 ensures arithmetic safety.
- **Front-Running**: ASC's deterministic scheduling prevents transaction ordering attacks.
- **DoS**: Massa's high-throughput consensus minimizes gas-based attacks.

4 Frontend Architecture

The DeWeb frontend is a React single-page application (SPA) hosted on Massa's on-chain storage, leveraging modern JavaScript (ES6+) and Tailwind CSS for styling. It interacts with the smart contract via Massa's SDK, ensuring trustless data retrieval and transaction submission.

4.1 Component Structure

- **PoolDashboard**: Displays `poolBalance`, `balances[user]`, and `loans[user]`.
- **LendButton**: Triggers `deposit()` with wallet integration.
- **BorrowButton**: Calls `borrow(amount)` with input validation.

4.2 React Implementation

Listing 2: React Frontend (DeWeb)

```
import React, { useState, useEffect } from 'https://cdn.jsdelivr.net/npm/react@18.2.0/+esm';
import { MassaSDK } from 'https://cdn.jsdelivr.net/npm/massa-sdk@latest/+esm';
import 'https://cdn.tailwindcss.com';

const FondoComunApp = () => {
    const [poolBalance, setPoolBalance] = useState(0);
    const [userBalance, setUserBalance] = useState(0);
    const [userLoan, setUserLoan] = useState(0);
    const [amount, setAmount] = useState(0.01);

    useEffect(() => {
        const fetchStatus = async () => {
            const status = await MassaSDK.contractCall('FondoComun', 'getStatus', [window.massaWallet.address]);
            setPoolBalance(status[0] / 1e18);
            setUserBalance(status[1] / 1e18);
            setUserLoan(status[2] / 1e18);
        };
        fetchStatus();
    }, []);

    const deposit = async () => {
        await MassaSDK.contractCall('FondoComun', 'deposit', [], { value: amount * 1e18 });
    };

    const borrow = async () => {
        await MassaSDK.contractCall('FondoComun', 'borrow', [amount * 1e18]);
    };

    return (
        <div className="min-h-screen bg-gray-100 text-blue-900 flex justify-center items-center">
            <div className="p-6 border border-blue-900 rounded-lg">
                <h2 className="text-2xl font-bold">Fondo Común</h2>
                <p>Pool Balance: {poolBalance} ETH</p>
                <p>Your Balance: {userBalance} ETH</p>
                <p>Your Loan: {userLoan} ETH</p>
                <input
                    type="number"
                    value={amount}
                    onChange={(e) => setAmount(e.target.value)}
                    className="border p-2 m-2"
                    placeholder="Amount (ETH)"
                />
                <button onClick={deposit} className="bg-teal-500 text-white p-2 m-2 rounded hover:brightness-110">
                    Lend
                </button>
                <button onClick={borrow} className="bg-teal-500 text-white p-2 m-2 rounded hover:brightness-110">
                    Borrow
                </button>
            </div>
        </div>
    );
};

export default FondoComunApp;
```

4.3 DeWeb Integration

The React SPA is compiled to static HTML/CSS/JS and stored on Massa's DeWeb, leveraging its on-chain hosting for resilience. The frontend uses the Massa SDK to sign transactions and query contract state, ensuring gas-efficient interactions.

5 User Flows

1. **Deposit**:
    - Input: User submits `msg.value` via `deposit()`.
    - Output: Updates `balances[msg.sender]` and `poolBalance`.
2. **Borrow**:
    - Input: User submits `amount` (0.01-0.1 ETH).
    - Output: Transfers `amount`, updates `loans[msg.sender]`, schedules repayment via ASC.
3. **Status Check**:
    - Input: User queries `getStatus(address)`.
    - Output: Returns `(poolBalance, balances[user], loans[user])`.

4. **Repayment**:
    - Input: ASC triggers `repay(borrower)`.
    - Output: Deducts `loans[borrower]` from `balances[borrower]`, updates `poolBalance`.

6 Performance Analysis

- **Gas Complexity**:
    - `deposit()`: $O(1)$ for state updates.
    - `borrow()`: $O(1)$ for state updates and transfer.
    - `repay()`: $O(1)$ for state updates.
- **Latency**: Massa's parallel execution ensures sub-second transaction confirmation.
- **Scalability**: Supports thousands of concurrent users due to Massa's sharded consensus.

7 Product-Market Fit

Fondo Común targets 1B+ gig economy workers and unbanked individuals, addressing:

- **Accessibility**: No collateral requirements, unlike Aave's 150%+ ratios.
- **Trustlessness**: Eliminates intermediaries, unlike Kiva's centralized model.
- **UX**: React-based UI with one-click actions, targeting non-crypto users.

Market size: $200B microfinance, $80B DeFi TVL (2025 estimates).

8 Massa Network Synergy

- **ASC**: Deterministic repayment scheduling, gas-efficient via Massa's execution model.
- **DeWeb**: On-chain React SPA, eliminating IPFS or centralized hosting vulnerabilities.
- **Throughput**: Massa's 10,000+ TPS supports high-frequency micro-loans.

9 Implementation and Deployment

The MVP, deployed on Massa testnet, includes:

- **Contract**: Single `FondoComun` contract.
- **Frontend**: React SPA on DeWeb, integrated with Massa SDK.
- **Submission**: GitHub repo (`https://github.com/cosmasken/Fondo-Comun`), 2-minute demo video, README with deployment instructions.

10 Future Enhancements

- **Dynamic Interest**: Implement yield curves based on `poolBalance` utilization.
- **Reputation System**: On-chain credit scores using transaction history.
- **Multi-Pool**: Segmented pools by risk or loan size.
- **UI Enhancements**: Real-time loan visualizations, user profiles.

11 Conclusion

Fondo Común delivers a scalable, trustless micro-lending protocol, leveraging Massa's ASC and DeWeb for unparalleled autonomy and resilience. By addressing microfinance accessibility, it positions itself as a pioneering DeFi primitive.