RheoFi

The Isolated-Pool Lending Protocol

v1.0

2025

Ancilar

Abstract: RheoFi Protocol (“RheoFi”) is an algorithmic money-market system designed to bring a complete decentralized lending and credit platform to the XRPL EVM Sidechain. RheoFi enables users to supply cryptocurrencies as collateral, earn a variable interest rate paid per block, and borrow over-collateralized assets at rates determined by a utilization-driven yield curve. Unlike single-pool money markets, RheoFi segregates assets into Isolated Liquidity Pools, each with independent risk parameters, oracle configuration, and reward schedules, so that risk in one pool cannot propagate to another. Prices are validated by a three-tier Resilient Oracle, and residual bad debt is absorbed by a dedicated Risk Fund through permissionless on-chain auctions.

RheoFi leverages the XRPL EVM Sidechain for fast, low-cost transactions and direct access to the XRPL asset ecosystem.

Table of Contents

Introduction

RheoFi Protocol is designed to provide a complete algorithmic money market on the XRPL EVM Sidechain. The protocol draws on the design lineage of Compound [1] and MakerDAO [2], and on the isolated-pool architecture introduced in Venus v4 [3], combining these patterns into a single auditable codebase adapted for the XRPL ecosystem.

Problems

Decentralized money markets now hold tens of billions of dollars in collateral, but large shared-liquidity designs have repeatedly exhibited systemic risk: a single compromised oracle, a poorly parameterized collateral asset, or an illiquid long-tail token can threaten the solvency of the entire protocol. Accumulated bad debt in one market has historically spilled over to unrelated markets sharing the same reserves.

Second, most existing DeFi lending stacks are deployed on chains where transaction fees and confirmation latency make small-ticket operations and rapid liquidations costly. The XRPL ecosystem, which holds one of the largest non-EVM asset bases in crypto, has until recently lacked a native lending venue capable of using those assets as productive collateral.

Solution

RheoFi unifies money-market lending and isolated-pool risk segregation into a single contract suite on the XRPL EVM Sidechain. Users can supply assets, earn interest, and borrow against their collateral in one venue. Each supported market is housed in an Isolated Pool with its own comptroller, risk parameters, oracle, and caps, so losses in any single pool cannot reach users of other pools.

Use Cases

Alice holds XRP and wishes to access USD-denominated liquidity without selling her position and realizing capital gains. She bridges XRP onto the XRPL EVM Sidechain, supplies it to a RheoFi pool, enables it as collateral, and borrows USDC directly from the same pool. Her XRP continues to earn supply interest while she retains upside exposure. When she is ready, she repays her debt with accrued interest and redeems her full collateral.

RheoFi

The isolated-pool lending platform on the XRPL EVM Sidechain.

Key features:

- Borrow cryptocurrencies and stablecoins with no credit check, settled directly on the XRPL EVM Sidechain.
- Supply cryptocurrencies and stablecoins to earn a variable APY backed by over-collateralized loans.

- Risk-isolated pools: every market lives inside a pool with its own comptroller, oracle, and caps.
- Three-tier Resilient Oracle and Risk-Fund-backed shortfall auctions.

Supplying Assets

RheoFi users may supply supported cryptocurrencies or digital assets into any pool that lists them. Supplied assets are pooled in smart contracts and can be used as collateral or simply held for yield. Suppliers receive an **rToken** (for example, *rUSDC* or *rWETH*) — an ERC-20 receipt token whose exchange rate against the underlying asset appreciates as interest accrues. Supply is redeemable at any time subject to pool liquidity.

When a user supplies underlying assets, the rToken market **mints** new rTokens to the user at the current exchange rate.

Figure 1. Depositing collateral: underlying in, rToken minted.

Withdrawal is the mirror operation: the user returns their rTokens to the market, which **burns** them and releases the underlying asset at the current exchange rate.

Figure 2. Withdrawing collateral: rToken burned, underlying out.

Holding an rToken does not automatically expose the holder to liquidation risk. Collateralization is an explicit opt-in via the pool's Comptroller; users that hold rTokens purely for yield remain outside the liquidation surface.

Borrowing Assets

Users who wish to borrow must first supply collateral and opt it in as collateral within the target pool. Each collateral asset has a governance-configured *collateral factor* (typically between 40% and 90% depending on risk), which determines the share of its value that may be borrowed against. Borrowers pay interest per slot at a rate set by the pool's interest-rate model; there are no fixed repayment schedules, and debt can be repaid at any time.

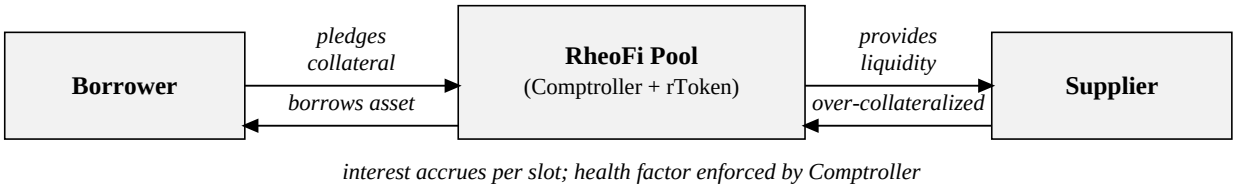

Figure 3. Borrowing against supplied collateral, backed by pool suppliers.

If the value of a borrower's collateral falls — or the value of their debt rises — such that the position's *health factor* drops below 1, the position becomes liquidatable. Anyone may then call the liquidation path to repay part of the debt in exchange for the borrower's collateral at a discount.

Protocol Architecture

The protocol is architected as an evolution of the Compound and MakerDAO designs, extended with the isolated-pool, resilient-oracle, and shortfall-auction patterns introduced in Venus v4 [3] and adapted for the XRPL EVM Sidechain.

Isolated Pools and the Comptroller

Every market in RheoFi is contained inside an **Isolated Pool** anchored by a **Comptroller**. The Comptroller enforces market listing, account-level liquidity checks, collateral-factor and liquidation-threshold parameters, supply and borrow caps, and the close factor and liquidation incentive. The **PoolRegistry** records every deployed pool along with its metadata.

Comptroller bounds, enforced on-chain (Comptroller.sol):

```
MIN_CLOSE_FACTOR_MANTISSA       = 0.05e18  (5%)
MAX_CLOSE_FACTOR_MANTISSA       = 0.90e18  (90%)
MAX_COLLATERAL_FACTOR_MANTISSA  = 0.95e18  (95%)
liquidationThreshold            >= collateralFactor
```

An account's liquidity is computed in the Comptroller by summing, over every market the account is active in, the weighted collateral value and the total borrow value. Because this iteration is restricted to the Comptroller's own market list, no action in any other pool can alter a user's solvency state in the current pool.

rTokens

An rToken is an interest-bearing ERC-20 representing a pro-rata claim on the underlying asset supplied to a market. Interest accrues via an ever-increasing exchange rate rather than by rebasing. The stored exchange rate is:

```
if totalSupply == 0:
    exchangeRate = initialExchangeRateMantissa
else:
    exchangeRate = (totalCash + totalBorrows + badDebt - totalReserves)
                   * 1e18 / totalSupply
```

Interest is accrued lazily at the start of every state-changing call via accrueInterest(). Letting slotDelta be the number of slots since the last accrual and r the borrow rate per slot:

```
simpleInterestFactor = r * slotDelta
interestAccumulated  = simpleInterestFactor * totalBorrows
totalBorrowsNew      = totalBorrows + interestAccumulated
totalReservesNew     = totalReserves + reserveFactor * interestAccumulated
borrowIndexNew       = borrowIndex * (1 + simpleInterestFactor)
```

A borrower's outstanding balance is always recomputed as principal × borrowIndex / interestIndexAtBorrow, so interest is applied in O(1) per user without any loops over accounts.

Interest Rate Model

Borrow rates are driven by market utilization. Let U be the utilization rate, b the base rate per slot, m the slope multiplier, j the jump multiplier, and k the kink utilization.

```
U = (borrows + badDebt) / (cash + borrows + badDebt - reserves)  (cap 1e18)
```

```
if U <= k:
    borrowRate = b + (U * m) / 1e18
else:
    normalRate = b + (k * m) / 1e18
    borrowRate = normalRate + ((U - k) * j) / 1e18
```

The supply rate is derived from the borrow rate net of the reserve share:

```
oneMinusRF = 1e18 - reserveFactor
rateToPool = (borrowRate * oneMinusRF) / 1e18
supplyRate = (U * rateToPool) / 1e18
```

Below the kink, rates rise gradually to keep borrowing attractive while utilization is productive. Above the kink, the jump multiplier sharply penalizes further borrowing, repelling marginal demand and incentivizing new supply — a self-correcting mechanism that preserves redeemability for existing suppliers.

RheoFi also ships a two-kinks variant (TwoKinksInterestRateModel) with breakpoints KINK_1 < KINK_2, three slope multipliers, and two base rates, used on markets that benefit from a finer slope structure. Deployed contracts follow a transparent naming scheme: JumpRateModelV2_base{X}bps_slope{Y}bps_jump{Z}bps_kink{K}bps_bpy{N}. A representative XRPL-testnet configuration uses base = 0%, slope = 10%/yr, jump = 250%/yr, kink = 80%, and blocksPerYear = 63,072,000 (≈ 2.08-second slot time).

Resilient Price Oracle

All valuations pass through the **ResilientOracle**, which routes each price query through up to three independent feeds identified by role:

```
enum OracleRole { MAIN, PIVOT, FALLBACK }
  MAIN     = primary feed   (e.g. Chainlink)
  PIVOT    = cross-check    (configurable)
  FALLBACK = backup feed    (configurable)
```

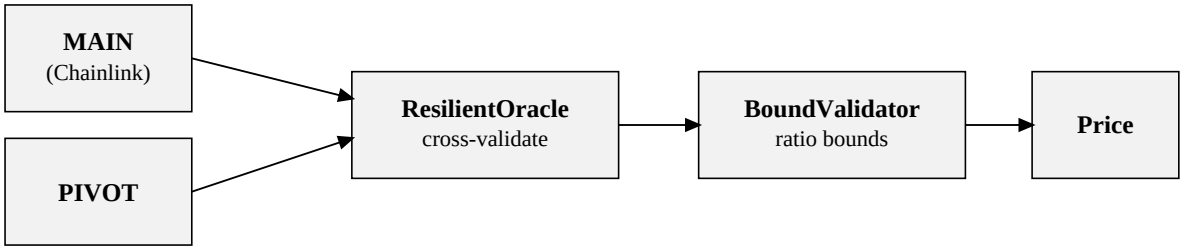

Figure 4. Oracle resolution: feeds are cross-validated by the BoundValidator before a price is returned.

For each asset, the oracle resolves a price as follows:

- If the oracle is paused globally, revert.
- Fetch the pivot price if the pivot is enabled.
- Try the main price validated against the pivot; if valid, return it.
- Otherwise try the fallback price validated against the pivot; if valid, return it.
- Otherwise validate the main price against the fallback price; if valid, return the main price.
- Otherwise revert with invalid resilient oracle price.

Validation is handled by BoundValidator, which enforces per-asset upper and lower ratio bounds:

```
anchorRatio = (anchorPrice * 1e18) / reportedPrice
require(lowerBoundRatio <= anchorRatio <= upperBoundRatio)
```

Each feed configuration also carries a maxStalePeriod: a feed whose last update is older than this threshold is treated as invalid, and the oracle falls through to the next tier. Supported adapters include Chainlink and additional adapters configurable per asset; all configuration changes are gated by the Access Control Manager.

Liquidations

A position becomes liquidatable when the weighted collateral value falls below the total borrow value — equivalently, when the health factor drops below 1. For each market the account is in:

```
weightedCollateral += liquidationThreshold * exchangeRate * oraclePrice * rT
okenBalance
totalBorrow        += oraclePrice * borrowBalance
```

```
HealthFactor       = weightedCollateral / totalBorrow
```

A separate computation using collateralFactor in place of liquidationThreshold gates new borrows and withdrawals. Because liquidationThreshold ≥ collateralFactor by construction, there is always a safety margin between the point at which a user can no longer increase their risk and the point at which the position can be liquidated.

On a liquidation call:

```
maxClose    = closeFactorMantissa * borrowBalance                    (partia
l)
seizeAmount = repayAmount * liquidationIncentive * pBorrow / pCollateral
seizeTokens = seizeAmount / exchangeRate
```

```
protocolShare = DEFAULT_PROTOCOL_SEIZE_SHARE_MANTISSA = 5e16 (5%)
```

The close factor caps the share of a single position that may be liquidated in one call, damping MEV-driven over-liquidation. 5% of every seized collateral balance is routed to the Protocol Share Reserve and flows downstream to the Risk Fund.

Risk Fund and Shortfall Auctions

If a position becomes insolvent faster than liquidators can clear it, residual unbacked debt is written off into the rToken's badDebt accumulator via healAccount(). Residual bad debt is then resolved by the **Shortfall** contract through a competitive on-chain auction funded by the **Risk Fund**.

```
enum AuctionType   { LARGE_POOL_DEBT, LARGE_RISK_FUND }
enum AuctionStatus { NOT_STARTED, STARTED, ENDED }
```

```
MAX_BPS              = 10000
DEFAULT_INCENTIVE_BPS = 1000   (10%)
```

```
badDebtPlusIncentive = poolBadDebt * (1 + incentiveBps / 10000)
if riskFund >= badDebtPlusIncentive:
    type = LARGE_RISK_FUND
else:
    type = LARGE_POOL_DEBT
```

In a *LARGE_POOL_DEBT* auction, bidders compete on an increasing percentage of pool debt they will cover in exchange for the Risk Fund balance. In a *LARGE_RISK_FUND* auction, bidders compete on a decreasing share of the Risk Fund they will take in exchange for covering the fixed debt. Any account may trigger an auction once the pool's total bad debt exceeds the configured minimum threshold; any account may bid; an auction settles once no new bid is placed within nextBidderBlockLimit of the last bid.

The Risk Fund (RiskFundV2) tracks reserves on a per-pool basis via poolAssetsFunds[comptroller][asset], so an auction for one pool never consumes reserves accumulated by another. The fund is replenished by the 5% protocol seize share, by a governance-configured share of accrued market reserves, and by revenue streams from across the protocol.

Rewards Distributor

Each pool may have one or more **RewardsDistributor** contracts attached to its Comptroller. Every distributor is bound to a single reward token and maintains per-market supply and borrow indices; emission speeds are configured independently for supply and borrow sides of every market. Multiple distributors may run concurrently on the same pool, supporting overlapping incentive programs without any code changes. Indices start at INITIAL_INDEX = 1e36 and are updated lazily:

```
delta = currentSlot - supplyState.slot
if delta > 0 and supplySpeed > 0 and totalSupply > 0:
    accrued      = delta * supplySpeed
    ratio        = accrued * 1e36 / totalSupply
    supplyIndex += ratio
```

Security

The RheoFi codebase inherits the audit lineage of Venus v4 and adds XRPL-specific reviews. Fifteen prior engagements are archived under audits/ and cover the isolated-pool core, the rewards distributor, the risk fund and shortfall auction, the comptroller, forced liquidations, the time-based accrual architecture, and the native-token gateway. Auditors include PeckShield, Hacken, Certik, Quantstamp, FairyProof, and Pessimistic. A RheoFi-specific mainnet audit is planned prior to mainnet deployment.

Known risk classes and their mitigations:

- **Oracle risk.** Mitigated by the three-tier Resilient Oracle, per-asset deviation bounds, and staleness thresholds.
- **Smart-contract risk.** Mitigated by extensive auditing, upgradeable proxies gated by Timelock + ACM, and pause switches.
- **Liquidation latency.** Mitigated by partial liquidation, the close factor, and permissionless liquidator access.

Conclusion

RheoFi gives the XRPL EVM Sidechain a complete, auditable lending stack, synthesizing the strongest ideas from a decade of on-chain money-market research: Compound-style rTokens, Venus-style isolated pools, resilient multi-tier oracles, and permissionless bad-debt auctions.

By defaulting to risk isolation, oracle redundancy, transparent on-chain parameters, and permissionless liquidation and auction flows, RheoFi is designed to scale safely from day one while inviting builders, liquidity providers, and risk contributors to let their assets flow.

References

1. Robert Leshner and Geoffrey Hayes. *Compound: The Money Market Protocol.* February 2019. https://compound.finance/documents/Compound.Whitepaper.pdf

2. Maker Foundation Team. *The Maker Protocol White Paper (DAI).* December 2017. https://makerdao.com/whitepaper/

3. Venus Protocol. *Venus v4: Isolated Pools, Resilient Oracle, and Shortfall.* 2023. https://docs.venus.io

4. XRPL Labs. *XRPL EVM Sidechain Documentation.* https://xrpl.org/xrpl-evm-sidechain.html

5. RheoFi Protocol. *Smart-contract source repository.* https://github.com/ANCILAR/rheofi-contracts

Disclaimer

This document is for general information purposes only. It does not constitute investment, financial, legal, or tax advice, nor a recommendation or solicitation to buy or sell any token, security, or other instrument. RheoFi is experimental software; the protocol is currently deployed on the XRPL EVM Sidechain testnet, and mainnet deployment is subject to further audits and governance approvals.

Interacting with RheoFi involves substantial risk, including but not limited to smart-contract vulnerabilities, oracle manipulation, liquidity and liquidation risk, regulatory changes, and the potential total loss of capital. Forward-looking statements and parameter ranges reflect the state of the codebase at the time of writing and may change through governance; deployed values should always be verified on-chain before use. The RheoFi contributors and any associated entities disclaim all liability for any loss or damage arising, directly or indirectly, from reliance on this document or from interaction with the protocol.