Lemond Protocol

Introduction

Lemond is scheduled to initially run on OKExChain, supporting BTC, ETH, OKB, OKT, USDK, USDC, DAI, USDT firstly and will have a follow-up deployment to Ethereum and other important public chains.

Lemond is a decentralized, open-source, autonomous non-custodial liquidity market protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an overcollateralized (perpetually) or undercollateralized (one-block liquidity) fashion.

Each asset supported by the Lemond Protocol is integrated through a lToken contract, which is an EIP-20 compliant representation of balances supplied to the protocol. By minting lTokens, users (1) earn interest through the lToken's exchange rate, which increases in value relative to the underlying asset, and (2) gain the ability to use lTokens as collateral.

lTokens are the primary means of interacting with the Lemond Protocol; when a user mints, redeems, borrows, repays a borrow, liquidates a borrow, or transfers lTokens, she will do so using the lToken contract.

Each market has its own lToken (lETH, lOKT…), which you'll receive when you supply that asset to the protocol; lTokens represent your balance in Lemond. View them in your wallet, program them, or send them to cold storage — it's all up to you!

Simply by holding lTokens, you'll earn interest, which accumulates through the lToken's exchange rate — over time, each lToken becomes convertible into an increasing amount of its corresponding token.

Allocation

Token: LEMD
Total Supply: 1,000,000,000
60% Loan Mining + Liquidity Mining
15% DAO Governance
5% Seed Round Investment
10% Strategic Investment
3% Team Incentive
2% Advisor
1% Public Circulation on Exchange
4% Eco-incentive + Initial Liquidity

52,000,000 LEMD: 20% of LEMD to be distributed upon the exchange's public issuance; 20% every month after the public issuance with an exception of no LEMD being distributed 1 month after the public issuance.

100,000,000 LEMD: 30% of LEMD to be distributed upon the exchange's public issuance ; 30% of LEMD to be distributed upon the second month after the public issuance. 40% of LEMD to be distributed upon the third month after the public issuance.

Protocol Math

The Lemond protocol contracts use a system of exponential math in order to represent fractional quantities with sufficient precision.

Most numbers are represented as a mantissa, an unsigned integer scaled by 1 * 10 ^ 18, in order to perform basic math at a high level of precision.

lToken and Underlying Decimals

Prices and exchange rates are scaled by the decimals unique to each asset; lTokens are with 8 decimals, while their underlying tokens vary, and have a public member named decimals.

lToken	lToken Decimals	Underlying	Underlying Decimals
lETH	8	ETH	18
lOKB	8	OKB	18
lDAI	8	DAI	18
lOKT	8	OKT	18
lUSDC	8	USDC	6
lUSDK	8	USDK	6
lWBTC	8	WBTC	8
lUSDT	8	USDT	18

Interpreting Exchange Rates

The lToken Exchange Rate is scaled by the difference in decimals between the ltoken and the underlying asset.

```
onelTokenInUnderlying = exchangeRateCurrent / (1 * 10 ^ (18 + underlyingDecimals - lTokenDecimals))
```

Here is an example of finding the value of 1 lOKT in ETH with Web3.js JavaScript.

```
const lTokenDecimals = 8; // all lTokens have 8 decimal places
const underlying = new web3.eth.Contract(erc20Abi, otkAddress);
const lToken = new web3.eth.Contract(lTokenAbi, lOTKAddress);
const underlyingDecimals = await underlying.methods.decimals().call();
const exchangeRateCurrent = await lToken.methods.exchangeRateCurrent().call();
const mantissa = 18 + parseInt(underlyingDecimals) - lTokenDecimals;
const oneLTokenInUnderlying = exchangeRateCurrent / Math.pow(10, mantissa);
console.log('1 lOTK can be redeemed for', oneLTokenInUnderlying, 'OTK');
```

There is no underlying contract for ETH, so to do this with lETH, set underlyingDecimals to 18.

To find the number of underlying tokens that can be redeemed for lTokens, multiply the number of lTokens by the above value oneLTokenInUnderlying.

```
underlyingTokens = lTokenAmount * oneLTokenInUnderlying
```

Calculating Accrued Interest

Interest rates for each market update on any block in which the ratio of borrowed assets to supplied assets in the market has changed. The amount interest rates are changed depends on the interest rate model smart contract implemented for the market, and the amount of change in the ratio of borrowed assets to supplied assets in the market.

Interest accrues to all suppliers and borrowers in a market when any address interacts with the market's ltoken contract, calling one of these functions: mint, redeem, borrow, or repay. Successful execution of one of these functions triggers the accrueInterest method, which causes interest to be added to the underlying balance of every supplier and borrower in the market. Interest accrues for the current block, as well as each prior block in which the accrueInterest method was not triggered (no user interacted with the ltoken contract). Interest compounds only during blocks in which the ltoken contract has one of the aforementioned methods invoked.

Calculating the APY Using Rate Per Block

The Annual Percentage Yield (APY) for supplying or borrowing in each market can be calculated using the value of supplyRatePerBlock (for supply APY) or borrowRatePerBlock (for borrow APY) in this formula:

```
Rate = ltoken.supplyRatePerBlock(); // Integer
Rate = 37893566
ETH Mantissa = 1 * 10 ^ 18 (ETH has 18 decimal places)
Blocks Per Day = 4 * 60 * 24 (based on 4 blocks occurring every minute)
Days Per Year = 365

APY = ((((Rate / ETH Mantissa * Blocks Per Day + 1) ^ Days Per Year)) - 1) * 100
```

Here is an example of calculating the supply and borrow APY with Web3.js JavaScript:

```
const ethMantissa = 1e18;
const blocksPerDay = 4 * 60 * 24;
const daysPerYear = 365;

const ltoken = new web3.eth.Contract(cEthAbi, cEthAddress);
const supplyRatePerBlock = await ltoken.methods.supplyRatePerBlock().call();
const borrowRatePerBlock = await ltoken.methods.borrowRatePerBlock().call();
const supplyApy = (((Math.pow((supplyRatePerBlock / ethMantissa * blocksPerDay) + 1,
daysPerYear))) - 1) * 100;
const borrowApy = (((Math.pow((borrowRatePerBlock / ethMantissa * blocksPerDay) + 1,
daysPerYear))) - 1) * 100;
console.log(`Supply APY for ETH ${supplyApy} %`);
console.log(`Borrow APY for ETH ${borrowApy} %`);
```

Rather than individual suppliers or borrowers having to negotiate over terms and rates, the Lemond protocol utilizes an interest rate model that achieves an interest rate equilibrium, in each money market, based on supply and demand. Following economic theory, interest rates (the “price” of money) should increase as a function of demand; when demand is low, interest rates should be low, and vise versa when demand is high. The utilization ratio U for each market a unifies supply and demand into a single variable:

$$Ua = Borrowsa /(Casha + Borrowsa)$$

The demand curve is codified through governance and is expressed as a function of utilization. As an example, borrowing interest rates may resemble the following:

$$Borrowing\ Interest\ Rate_a = 2.5\% + U_a \times 20\%$$

The interest rate earned by suppliers is implicit, and is equal to the borrowing interest rate, multiplied by the utilization rate.

Liquidity Incentive Structure

The protocol does not guarantee liquidity; instead, it relies on the interest rate model to incentivize it. In periods of extreme demand for an asset, the liquidity of the protocol (the tokens available to withdraw or borrow) will decline; when this occur, interest rates rise, incentivizing supply, and disincentivizing borrowing.

ltoken Contracts

Each money market is structured as a smart contract that implements the specification. User's balances are represented as ltoken balances; users can mint (uintamountUnderlying) ltokens by supplying assets to the market, or redeem(uint amount) ltokens for the underlying asset. The price (exchange rate) between ltokens and the underlying asset increases over time, as interest is accrued by borrowers of the asset, and is equal to:

$$exchangeRate = \frac{underlyingBalance + totalBorrowBalance_a - reserves_a}{lTokenSupply_a}$$

As the market's total borrowing balance increases (as a function of borrower interest accruing), the exchange rate between ltokens and the underlying asset increases.

Risk & Liquidation

If the value of an account's borrowing outstanding exceeds their borrowing capacity, a portion of the outstanding borrowing may be repaid in exchange for the user's ltoken collateral, at the current market price minus a liquidation discount; this incentives an ecosystem of arbitrageurs to quickly step in to reduce the borrower's exposure, and eliminate the protocol's risk.

The proportion eligible to be closed, a close factor, is the portion of the borrowed asset that can be repaid, and ranges from 0 to 1, such as 25%. The liquidation process may continue to be called until the user's borrowing is less than their borrowing capacity.

Any address that possesses the borrowed asset may invoke the liquidation function, exchanging their asset for the borrower's ltoken collateral. As both users, both assets, and prices are all contained within the Lemond protocol, liquidation is frictionless and does not rely on any outside systems or order-books.

Interest Rate Mechanics

Lemond money markets are defined by an interest rate, applied to all borrowers uniformly, which adjust over time as the relationship between supply and demand changes.

The history of each interest rate, for each money market, is captured by an Interest Rate Index, which is calculated each time an interest rate changes, resulting from a user minting, redeeming, borrowing, repaying or liquidating the asset.

Market Dynamics

Each time a transaction occurs, the Interest Rate Index for the asset is updated to Lemond the interest since the prior index, using the interest for the period, denominated by r * t, calculated using a per-block interest rate:

$$Index_{a,n} = Index_{a,tn} — 1t \times (1 + r \times t)$$

The market's total borrowing outstanding is updated to include interest accrued since the last index:

$$totalBorrowBalance_{a,n} = totalBorrowBalance_{a,(n-1)} * (1 + r * t)$$

And a portion of the accrued interest is retained (set aside) as reserves, determined by a reserveFactor, ranging from 0 to 1:

$$reserves_a = reserves_{a,(n-1)} + totalBorrowBalance_{a,(n-1)} * (r * t * reserveFactor)$$

Borrower Dynamics

A borrower's balance, including accrued interest, is simply the ratio of the current index divided by the index when the user's balance was last checkpointed.

The balance for each borrower address in the ltoken is stored as an account checkpoint. An account checkpoint is a Solidity tuple <uint256 balance, uint256 interestIndex>. This tuple describes the balance at the time interest was last applied to that account.

Borrowing

A user who wishes to borrow and who has sufficient balances stored in Lemond may call *borrow(uint amount)*on the relevant ltoken contract. This function call checks the user's account value, and given sufficient collateral, will update the user's borrow balance, transfer the tokens to the user's address, and update the money market's floating interest rate.

Borrows accrue interest in the exact same fashion as balance interest was calculated in section 3.2; a borrower has the right to repay an outstanding loan at any time, by calling *repayBorrow(uint amount)* which repays the outstanding balance.

Liquidation

If a user's borrowing balance exceeds their total collateral value (borrowing capacity) due to the value of collateral falling, or borrowed assets increasing in value, the public function liquidate(address target, address collateralAsset, address borrowAsset, uintcloseAmount) can be called, which exchanges the invoking user's asset for the borrower's collateral, at a slightly better than market price.

Price Feeds

A Price Oracle maintains the current exchange rate of each supported asset; the Lemond protocol delegates the ability to set the value of assets to a committee which pools prices from the top 10 exchanges. These exchange rates are used to determine borrowing capacity and collateral requirements, and for all functions which require calculating the value equivalent of an account.

Governance

Lemond will begin with centralized control of the protocol (such as choosing the interest rate model per asset), and over time, will transition to complete community and stakeholder control.

The following rights in the protocol are controlled by the admin:
The ability to list a new ltoken market
The ability to update the interest rate model per market
The ability to update the oracle address
The ability to withdraw the reserve of a ltoken
The ability to choose a new admin, such as a DAO controlled by the community; because this DAO can itself choose a new admin, the administration has the ability to evolve over time, based on the decisions of the stakeholders.

Summary

Lemond is a Protocol initialized on OKExChain with lower gas cost and higher efficiency. Besides the basic loan function, Lemond would bring more in the following steps, all is aiming for a Juicy DeFi environment.