RockX ETH2.0 Liquid Staking Explained

RockX Team

August 3, 2022

Abstract

Staking, a cryptoeconomic primitive that allows participants to earn yield in exchange for locking tokens, has taken center stage over the past two years. Under Proof of Stake consensus mechanism, instead of using computational power, validators lock ("stake") a certain amount of the network's native cryptoasset as collateral to create new blocks. In return, they earn inflationary rewards and transaction fees.

1 Introduction

1.1 What is Proof of Stake?

Proof of Stake is the consensus protocol utilised in Ethereum 2.0 (ETH2). The consensus protocol helps everyone to know what transactions have been processed and in what order, which is known as validation.

ETH2 currently has a Proof of Stake (PoS) chain called the Beacon Chain, which is faster, more energy efficient and more decentralised than the current consensus protocol Ethereum is utilising (Proof of Work). Users deposit ETH and provide an Ethereum node to perform the required validation. As a reward for providing the node, the Beacon Chain gives node operators additional ETH on top of their deposits. These rewards are minted in return for helping secure the network.

1.2 What is liquid staking?

It is clear that PoS chains will be an integral part of the future of crypto, and become the foundation layer of which DeFi and metaverses will be built on.

However, PoS comes with some drawbacks for those wanting to participate directly as validators:

It requires technical know-how to set up and operate 32ETH is a significant barrier for regular token holders to participate in POS validation Staked tokens are locked up and become illiquid assets In order to solve these issues, liquid staking protocols were born. Liquid staking abstracts the depositing of tokens from running a validator node.

In exchange for their tokens, depositors receive a representative (uniETH) token from the protocol which is a claim on the tokens they have staked.

1.3 What is uniETH?

uniETH represents the staked ETH plus all future staking rewards. uniETH does not grow in quantity over time but instead, grows in value, i.e. 1 uniETH becomes worth increasingly more than 1 ETH.

1.4 What should I stake with RockX instead of staking directly to ETH2?

RockX removes several drawbacks that exist with Proof of Stake on ETH2.

The Beacon Chain requires a minimum deposit of at least 32 ETH. RockX will allow anyone to earn the reward on any amount of ETH deposited with us.

When depositing ETH on the Beacon Chain, users are required to have technical knowledge of interacting with smart contracts. RockX handles all interaction with the Beacon Chain in our users' place. The Beacon Chain will also require users who make deposits to be technically proficient at running Ethereum nodes 24/7, at the same time keeping that node online and secure. RockX provides this service in our users' place.

As ETH2 is being rolled out in several phases. We are currently in phase 1, the merge, and depositing now means your deposit is locked until phase 2 arrives, which could very well still be a long time away. With RockX you instantly get uniETH when depositing and do not need to be locked with us. It can be traded, sold or held at any time, which provides our users with liquidity on their staked ETH.

1.5 What is the staking period for uniETH?

The staking period to redeem the underlying ETH on uniETH will only be confirmed till ETH2 goes into phase 2. However, you will receive uniETH when you deposit and it will still gain staking rewards over time in terms of the token value. uniETH can also be sold and traded on various DEXs and CEXs if there is liquidity available for the trade.

1.6 What is the minimum deposit?

RockX gives everyone the opportunity to earn rewards on any amount of ETH, as we do not have a minimum. We do recommend a deposit of at least 0.01ETH to make your transaction worthwhile. When you stake ETH, you will receive uniETH, which gains rewards over time based on the performance of our nodes on the Beacon Chain.

1.7 What is the maximum deposit?

There is no limit on the amount of ETH you can stake with RockX on ETH2. The more ETH, the more rewards you will be receiving.

2 RockX uniETH staking algorithm

2.1 Terminology

ETH $1 \cdot ETH \equiv 10^{18}$

TotalSupply current total supply of uniETH.

TotalStaked total ethers staked to validators.

TotalDebts total unpaid debts, generated from **redeemFromValidators()**, awaiting to be paid by turning off validators and initiates debt clearance procedure.

TotalPending pending ethers to be staked.

RewardDebts the remaining ethers from debt clearance procedure.

AccountedUserRevenue overall net revenue which belongs to all uniETH holders.

ReportedValidators latest reported active validators.

ReportedValidatorBalance latest reported overall balance of active validator.

RecentEthersMoved the amount this contract receives recently from validators.

CurrentReserve overall assets under management, given as:

$$CurrentReserve = TotalPending + TotalStaked + AccountedUserRevenue - TotalDebts - RewardDebts$$

Exchange Ratio Defined as symbol ρ of uniETH to ETH, given as:

$$\rho = \frac{TotalSupply}{CurrentReserve}, \{normally : \rho \leq 1.0\}$$

managerFeeShare share of the manager fee, represented as 1 in 1000, managerFeeShare $\in [0, 1000]$

2.2 The liquid staking algorithm explained

In this section, we'll explain the details of liquid staking below from a user's perspective.

2.3 Stake ethers to mint uniETH

A user calls function **mint()** with a specific amount of **ethersToStake** ethers to mint **uniETH**, in each function call, we have:

Theorem 2.1

$$minted_{uniETH} := \rho \cdot ethersToStake$$

$$TotalSupply = TotalSupply' + minted_{uniETH}$$

$$TotalPending = TotalPending' + ethersToStake$$

Users receives $minted_{uniETH}$ token of uniETH if **mint()** succeeds. **uniETH** is a standard **ERC-20** compliant contract issued by RockX staking contract only, can it can be exchanged at will, uniETH can also be traded on **DEX**s for liquidation other than redeeming from official contract as long as there are sufficient liquidity on DEXs.

2.4 Initiating delegation into ETH2 offical contract

At any time **TotalPending** has more than $32 ethers$, the contract manager can call **stake()** function to stake the ethers into Ethereum 2.0 staking contract, then **TotalPending** moves to **TotalStaked** and keeps **TotalPending** less than $32 ethers$, we calculate the changes as followings:

Lemma 2.2

$$ethersToStake := \lfloor \frac{TotalPending'}{32ETH} \rfloor \cdot 32ETH$$

The ethers to delegate to ETH2 official contract is rounded to $N \cdot 32ETH$ as above, then:

Theorem 2.3

$$TotalPending = TotalPending' - ethersToStake$$

$$TotalStaked = TotalStaked' + ethersToStake$$

The timing for calling stake() function mainly considers to maximize capital efficiency, if we stake too soon, the cost for running a single validator is way too expensive, on the contrary, if we stake too late, the staked ethers does not generate rewards during the period before ethers staked to official ETH2 staking contract, so we have built an off-chain program to provide a seamless and comprehensive solution for staking, allowing maximum capital efficiency on assets while earning cryptonative yields.

2.5 Redeeming staked ethers from official RockX smart contract

Users call function **redeemFromValidators()** with a specific amount of **ethersToRedeem** ethers expected to redeem, the amount uniETH to be burnt is exactly to $N \cdot 32ETH$ worth of uniETH, then we have:

Theorem 2.4

$$burned_{uniETH} := \rho \cdot ethersToRedeem$$

$$TotalSupply = TotalSupply' - burned_{uniETH}$$

$$TotalDebts = TotalDebts' + ethersToRedeem$$

Redeeming(or Unstaking) works as by turning off validators, to wait until the validators to be offline and return the staked ethers to the contract. So it's a time-consuming asynchronous process. The benefit from redeeming directly from this contract is that, there's no slippage as in CEX or DEX, but you have to be patient. Once the ethers returned, you'll be notified to claim the ethers.

Note: redeeming function will only be available after ETH2.0 merged.

2.6 How ethers returns to this contract from validators?

The way the withdrawals implementation is currently, withdrawals are not like normal transactions, and are instead system level transactions that update an account's balance without a transaction in or out. In our contract implementation, we use **accountedBalance** to track the in and out of the ethers, and compare **accountedBalance** with **this.balance**, the difference is the ethers reward returned from validators. We utilize variable **RecentEthersMoved** to track this difference:

$$RecentEthersMoved = accountedBalance - this.balance$$

Ethers moves from contract to validators, or vice versa, we can assume: **RecentEthersMoved** is the ethers moved from $validators \implies contract$.

2.7 Stopping validators for debt clearance

Whenever a validator stopped by a node operator, the ethers are supposed to return to this contract by the setting of WITHDRAWAL_CREDENTIALS, once the ethers returned, the oracle calls **validatorStopped()** function, along with the following parameters:

valueStopped The ethers sent-back from validators to the liquid staking contract.

validatorStopped The count of stopped validators.

Suppose:

$$valueStopped \geq amountUnstaked$$

and meanwhile, **RecentEthersMoved** will be updated as explained in section:2.6.

$$RecentEthersMoved = RecentEthersMoved' + valueStopped$$

If we do not have **slashing**, then we have:

Lemma 2.5

$$amountUnstaked := 32ETH \cdot validatorStopped$$

$$incrRewardDebt := valueStopped - amountUnstaked$$

then, we use these 2 varaibles above to update the following variables:

Theorem 2.6

$$RewardDebts = RewardDebt' + incrRewardDebt$$

$$TotalPending = TotalPending' + Max(0, amountUnstaked - TotalDebts) + incrRewardDebt$$

$$TotalStaked = TotalStaked' - amountUnstaked$$

2.8 Calculating rewards

An oracle service will push overall alive validators balance periodically, there're 2 things **pushBeacon()** do to calculate the reward:

2.8.1 Adjusting reward base

Firstly, pushBeacon will check if there is new validator alive, the reward base will be aligned to the new staked value plus previously reported validators balance, the **reward base** is the defined as the base-point to check for balance increasement, given:

aliveValidator The count of validators alive

we have:

Theorem 2.7

$$RewardBase = ReportedValidatorBalance + Max(0, aliveValidator - ReportedValidators) \cdot 32ETH$$

2.8.2 Reward distribution

Secondly, revenue will be accumulated as followings if there's any. Normally ethers will either stays in contract or validators, the overall assets under management is:

$$TVL := ethersInContract + ethersInValidators$$

During calculation of rewards changes, we only consider the balance change in alive validators, we can assume that, if:

$$aliveBalance + RecentEthersMoved \geq RewardBase$$

then rewards has generated, we formalize the formula as below, given:

aliveBalance The balance of current alive validators

we have:

Theorem 2.8

$$r := Max(0, aliveBalance + RecentEthersMoved - RewardBase)$$

$$AccountedUserRevenue = AccountedUserRevenue' + r \cdot \frac{(1000 - managerFeeShare)}{1000}$$

$$RecentEthersMoved = 0$$

$$ReportedValidators = aliveValidator$$

$$ReportedValidatorBalance = aliveBalance$$

As **RecentEthersMoved** will only be counted during reward distribution, the variable will be reset to 0 at the end of each **pushBeacon()** call. The exchange ratio will go down by the change of **AccountedUserRevenue**.

2.9 Slashing

Nobody expects slashing, but what if it happens, under such rare case, users will face a sudden change of **exchange ratio**, actually we made a tiny modification to handle this situation in **pushBeacon()** and **validatorSlashedStop()**

2.9.1 Slashing notification in validatorSlashedStop()

Given:

remainingEthers The ethers left after slashing

slashedAmount The ethers slashed

slashedValidators The count of slashed validators.

we have:

Theorem 2.9

$$TotalPending = TotalPending' + remainingEthers$$

$$TotalStaked = TotalStaked' - 32ETH \cdot slashedValidators$$

$$RecentSlashed = RecentSlashed' + slashedAmount$$

;

2.9.2 Modification of rewards calculation in pushBeacon()

Theorem 2.10

$$r := Max(0, aliveBalance + RecentEthersMoved + RecentSlashed - RewardBase)$$

$$AccountedUserRevenue = AccountedUserRevenue' + r \cdot \frac{(1000 - managerFeeShare)}{1000}$$

$$RecentEthersMoved = 0$$

$$RecentSlashed = 0$$

$$ReportedValidators = aliveValidator$$

$$ReportedValidatorBalance = aliveBalance$$

3 DAO Governance

(TBD)

4 Conclusion

(TBD)

A Appendix

Initial Stage:				
User A Stakes 32 ETH	**ASSETS**	ETH	**LIABILITY**	uniETH
	User A Deposit	32	User A uniETH	32
	Total Assets	32	Total Liability	32
	SwapRatio			1

Stage 1:				
Got 0.32 ETH Rewards	**ASSETS**	ETH	**LIABILITY**	uniETH
	User A Deposit	32	User A uniETH	32
	Mining Rewards	0.32		
	Total Assets	32.32	Total Liability	32
	SwapRatio			1.01

Stage 2:				
User B Stakes 64 ETH	**ASSETS**	ETH	**LIABILITY**	uniETH
	User A Deposit	32	User A uniETH	32
	User B Deposit	64	User B uniETH	63.36633663
	Mining Rewards	0.32		
	Total Assets	96.32	Total Liability	95.36633663
	SwapRatio			1.01

Stage 3:				
User B transfer 32 uniETH to User C	**ASSETS**	ETH	**LIABILITY**	uniETH
			User A uniETH	32
			User B uniETH	31.36633663
			User C uniETH	32
	Total Assets	96.32	Total Liability	95.36633663
	SwapRatio			1.01

Stage 4:				
User A unstakes equivalent 32 ETH value of uniETH	**ASSETS**	ETH	**LIABILITY**	uniETH
	Before Redeeming	96.32	User A uniETH	32
			User B uniETH	31.36633663
			User C uniETH	32
	User A Redeems	-32	User A Burned	-31.68316832
	Total Assets	64.32	Total Liability	63.68316831
	SwapRatio			1.01

Table 1: RockX ETH2 Liquid Staking Balance Sheet