RockX ETH2.0 Liquid Staking Explained

RockX Team

August 15, 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 a Proof of Stake consensus mechanism, instead of using computational power, validators lock ("stake") a certain amount of the network's native cryptoasset as collateral, thus becoming eligible 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 a node and the required $32 \cdot ETH$ deposit are significant barriers 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. $1 \cdot uniETH$ becomes worth increasingly more than $1 \cdot ETH$. If you deposit $1 \cdot ETH$ initially, then your $1 \cdot uniETH = 1 \cdot ETH$, however after receiving some rewards and the total claim you have is $1 \cdot ETH + 0.2 \cdot ETH$ rewards, then your $1 \cdot uniETH = 1.2 \cdot ETH$.

1.4 How are staking rewards distributed?

As all the ETH deposited through this protocol is pooled together to provide the $32 \cdot ETH$ for each node, the node will receive the rewards and automatically distribute it across all staking participants based on how much ETH they staked of the 32 ETH total. I.e. if you staked 3.2 ETH you will receive 10% of each reward. Each time rewards are distributed (every block), they will be added to the initial stake amount, eventually compounding future earnings as more nodes are added, while immediately increasing the value of the uniETH tokens representing each stake.

1.5 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 $32 \cdot ETH$. RockX will allow anyone to earn rewards 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 for our users. The Beacon Chain will also require users who make deposits to be technically proficient at running Ethereum nodes, keeping their own node online and secure 24/7. RockX provides this service for our users.

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. uniETH can be held, traded, or sold at any time, providing our users with instant liquidity on their staked ETH.

1.6 What is the staking period for uniETH?

The staking period to redeem the underlying ETH with uniETH will only be confirmed once ETH2 goes into phase 2. However, you will instantly receive uniETH when you deposit ETH and you will still gain staking rewards over time as your uniETH will increase in value as your rewards are added to your initial stake. uniETH can also be sold and traded on various DEXs and CEXs if there is liquidity available for the trade.

1.7 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.

1.8 What is the maximum deposit?

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

2 RockX uniETH staking algorithm

2.1 Terminology

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

TotalSupply current total supply of uniETH, the total supply of uniETH is proportional to total ETH staked.

TotalStaked total staked to validators.

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

TotalPending pending ETH to be staked.

RewardDebts the remaining ETH from debt clearance procedure.

UserRevenue overall net revenue which belongs to all uniETH holders.

ReportedValidators latest reported active validators.

ReportedValidatorBalance latest reported overall balance of active validators.

RecentEthersMoved the amount this contract receives recently from validators.

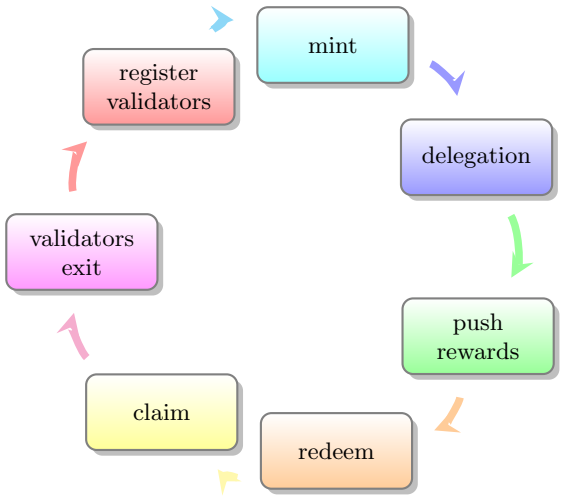


Figure 1: Lifecycle of RockX Staking Contract.

CurrentReserve overall assets under management, given as:

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

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

$$\rho = \begin{cases} \frac{TotalSupply}{CurrentReserve} & CurrentReserve \in (0, +\infty) \\ 1 & CurrentReserve = 0 \end{cases} \tag{1}$$

normally: $\rho \leq 1.0$

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

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

2.2 Stake ETH to mint uniETH

A user calls function **mint()** with a specific amount of **ethersToStake** ETH 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 only by RockX staking contract. Besides redeeming for the staked ETH through our smart contract, uniETH can also be traded on **DEX**s as long as there is sufficient liquidity on DEXs.

The general rule for **minting** and **redeeming** is: keep the exchange ratio - ρ invariant, if ρ changes during **minting** and **redeeming** process, users can arbitrage.

Proof. To prove ρ invariant and irrelevant of ETH to stake, for $CurrentReserve \in (0, +\infty)$:

$$\rho = \frac{TotalSupply}{CurrentReserve} = \frac{TotalSupply' + \rho' \cdot ethersToStake}{CurrentReserve' + ethersToStake}$$

as by definition:

$$\rho' = \frac{TotalSupply'}{CurrentReserve'}$$

we have:

$$\rho = \frac{CurrentReserve' \cdot \rho' + \rho' \cdot ethersToStake}{CurrentReserve' + ethersToStake} = \frac{\rho' \cdot (CurrentReserve' + ethersToStake)}{CurrentReserve' + ethersToStake}$$

finally:

$$\rho = \rho'$$

□

2.3 Initiating depositing into ETH2 offical contract

At any time **TotalPending** has more than $32 \cdot ETH$, the contract manager can call **stake()** function to stake the ETH into Ethereum 2.0 staking contract. As a result, ETH in **TotalPending** moves to **TotalStaked** and keeps **TotalPending** less than $32 \cdot ETH$, we will calculate the changes as follows:

Lemma 2.2.

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

The ETH to deposit to ETH2 official contract is bounded to $N \cdot 32ETH$ as above, we define the depositing process as:

Theorem 2.3.

$$TotalPending = TotalPending' - ethersToDeposit$$

$$TotalStaked = TotalStaked' + ethersToDeposit$$

In depositing process 2.3, ρ is kept invariant as **CurrentReserve** does not change, as:

$$\begin{aligned} CurrentReserve &:= \cdots + (TotalPending' - ethersToDeposit) + (TotalStaked' + ethersToDeposit) \\ &= \cdots + TotalPending' + TotalStaked' \end{aligned} \quad (2)$$

2.3.1 Timing

The timing for calling stake() contract function mainly considers maximising capital efficiency, if we stake prematurely, the cost for running a single validator is way too expensive. On the flipside, if we are slow to stake, the deposited ETH does not generate rewards during the period prior to it begin staked in the official ETH2 staking contract. Hence, there is a need to build an off-chain program to provide a seamless and comprehensive solution for staking, allowing maximum capital efficiency on assets while earning cryptonative yields.

2.4 Redeeming staked ETH from official RockX smart contract

Users call contract function **redeemFromValidators()** with a specific amount of **ethersToRedeem** ETH 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, waiting for the validators to be offline and returning the staked ETH to the contract. It's a time-consuming asynchronous process. The benefit from redeeming directly from this contract is that there's no slippage which you may face in a CEX or DEX. However as this process takes time, you have to be patient. Once the ETH is returned, you'll be notified[1] to claim it.

ρ is also invariant and irrelevant of ETH to unstake.

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

2.5 How ETH return to this contract from validators?

At current, Ethereum 2.0 withdrawals implementation are not like normal transactions. Instead, they are system level transactions that update an account's balance without a transaction in or out. In our contract implementation, we use **accountedBalance** to track explicit ETH in and out from contract functions, and compare **accountedBalance** with **this.balance**, the difference is the ETH rewards[2] returned from validators. We utilise variable **RecentEthersMoved** to track this difference:

$$RecentEthersMoved = accountedBalance - this.balance$$

Ethers moves from contract to validators, or vice versa, here we define: **RecentEthersMoved** is the ETH moved from $validators \Longrightarrow contract$. The balancing syncing is done in **syncBalance()**

2.6 Stopping validators for debt clearance

Whenever a validator is stopped by a node operator, the ETH are supposed to return to this contract due to the setting of WITHDRAWAL_CREDENTIALS in depositing [3], once the ETH returned, the oracle calls **validatorStopped()** function, along with the following parameters:

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

validatorStopped The count of stopped validators.

Suppose:

$$valueStopped \geq amountUnstaked$$

and meanwhile, ETH in validators has been transferred to contract and **RecentEthersMoved** has been updated as explained in section:2.5:

$$RecentEthersMoved = RecentEthersMoved' + valueStopped$$

If we do not have **slashing**, then we can deduce from the parameters that:

Lemma 2.5.

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

$$incrRewardDebt := valueStopped - amountUnstaked$$

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

Theorem 2.6.

$$RewardDebts = RewardDebt' + incrRewardDebt$$

$$\begin{aligned} TotalPending = TotalPending' + incrRewardDebt \\ + Max\{0, amountUnstaked - TotalDebts\} \end{aligned} \quad (3)$$

$$TotalStaked = TotalStaked' - amountUnstaked$$

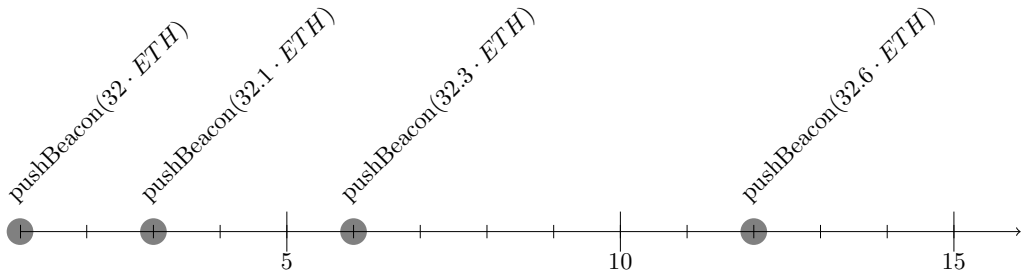
[1] available on web staking portal, https://unieth.rockx.com/

[2] Here we suppose nobody transfers ETH to this contract intentional or unintentional, if this happens, the contract treat the ETH as rewards, shared by all **uniETH** holders

[3] Initiating depositing into ETH2 offical contract

2.7 Calculating rewards

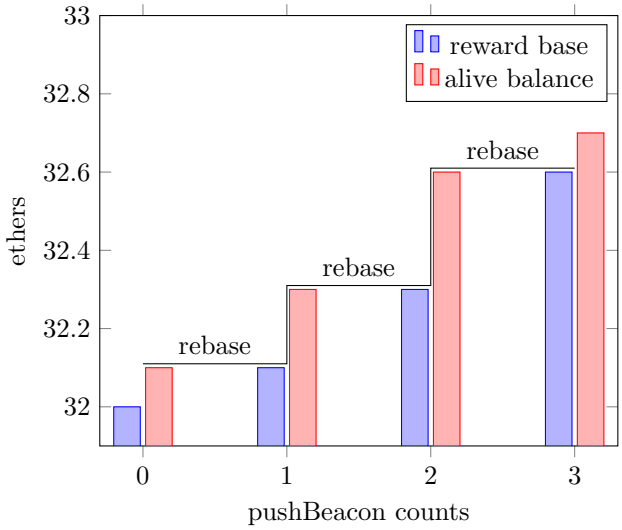
Rewards are aggregated on a daily basis, and will be reflected from the decrease of **exchange ratio**-ρ, meaning you receive more ETH back.



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

2.7.1 Adjusting reward base

Firstly, pushBeacon will check if any new validators has gone alive, the reward base will be adjusted to align to the newly staked ETH plus previously reported validators balance ETH, the **reward base** is defined as the reference point to compare to current balance, and it'll be updated in each consecutive pushBeacon() call with current **aliveBalance**.



Given:

aliveValidator The count of validators alive

we have:

Theorem 2.7.

$$RewardBase = ReportedValidatorBalance + Max\{0, aliveValidator - ReportedValidators\} \cdot 32 \cdot ETH$$

2.7.2 Reward distribution

Normally ETH will either stay in contract or validators, hence the overall assets under management or TVL is as follows:

$$TVL := ethersInContract + ethersInValidators$$

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

$$aliveBalance + RecentEthersMoved \geq RewardBase$$

then positive rewards have generated, we formalise the formula as below, given:

aliveBalance The balance of current alive validators

we have:

Theorem 2.8.

$$r := Max\{0, aliveBalance + RecentEthersMoved - RewardBase\}$$

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

$$RecentEthersMoved = 0$$

$$ReportedValidators = aliveValidator$$

$$ReportedValidatorBalance = aliveBalance$$

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 - ρ will go down by the change of **UserRevenue**, which allows the **uniETH** holder to get back more ETH when redeems.

2.7.3 Manager's Fee Withdrawal

Withdrawal before ETH2.0 merge is still not implemented yet, but here is the general process:

As WITHDRAWAL_CREDENTIALS has set to the contract address, to withdraw the manager's fee from validators, the manager MUST submit a request of withdrawal from the contract. Once the ETH arrive, the manager can claim them.

The key issue here is, does manager's withdrawal affects the calculation of rewards, as **aliveBalance** has reduced? The simple answer is - NO. The ETH moves from validators to the contract is accounted in **RecentEthersMoved**. It won't be an issue as long as RecentEthersMoved is accounted for, to make sure this happens, we'll call **syncBlanace()** to update RecentEthersMoved each time manager withdraws.

2.8 Slashing

Slashing is when a large portion of a validator's stake is removed from the network. This usually happens when a validator breaks the rules that are designed to prevent attacks on the network. Being slashed means that the validator will be forced to exit the beacon chain at some point in the future, receiving penalties until it does so.

Nobody expects slashing, but when it happens, the contract MUST handle it. Basically, users will face a sudden change of **exchange ratio -** ρ, in our implementation we made a tiny modification to handle this situation in **pushBeacon()** and **validatorSlashedStop()**

2.8.1 Handling slashing in validatorSlashedStop()

Given:

remainingEthers The ETH left after slashing

slashedAmount The ETH slashed

slashedValidators The count of slashed validators.

we have:

Theorem 2.9.

$$TotalPending = TotalPending' + remainingEthers$$

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

$$RecentSlashed = RecentSlashed' + slashedAmount$$

ρ will have a sudden changed due to the change of **TotalPending** and **TotalStaked** as:

$$\rho = \frac{TotalSupply}{CurrentReserve - 32 \cdot ETH \cdot slashedValidators + remainingEthers}$$

2.8.2 Modification of rewards calculation in pushBeacon()

Theorem 2.10.

$$r := Max\{0, aliveBalance + RecentEthersMoved + RecentSlashed - RewardBase\}$$

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

$$RecentEthersMoved = 0$$

$$RecentSlashed = 0$$

$$ReportedValidators = aliveValidator$$

$$ReportedValidatorBalance = aliveBalance$$

You may think it's a bit strange that we include **RecentSlashed** to rewards accumulating, just remember, ρ has already been changed, and **aliveBalance** has been decreased by **slashedAmount**, at the moment of slashing, we can say:

$$\begin{aligned} r := Max\{0, (aliveBalance' - slashedAmount) \\ + RecentEthersMoved + slashedAmount - RewardBase\} \end{aligned} \quad (4)$$

3 DAO Governance

The DAO token currently not completed yet, the supposed scenario is to vote on the key parameters of staking contract, such as:

Manager's Fee Setting to vote to change manager's fee percentage

Qualified Validators Registration to vote on validator keys proposed by other institutions, or to remove a validator.

Granting Roles to vote to grant specific roles to account addresses, like: *ORACLE_ROLE, REGISTRY_ROLE, MANAGER_ROLE, PAUSER_ROLE*

Upgrading Contract to vote to upgrade staking contract implementation, in case of critical issues or external

ETH2 Phase Change to vote to switch to another phase of ETH2.0, which will enable redeeming such as.

The current staking contract has been implemented to be upgradable, using transparent proxy.

4 Conclusion

RockX ETH2.0 Liquid Staking gives everyone the opportunity to earn rewards on any amount of ETH, and gains rewards over time based on the performance of our nodes on the Beacon Chain. Allowing retail users to participate in ETH2.0 network maintainence works like Inclusive Financial system in real world. Besides stakers has the ability to hedge their uniETH tokens to avoid losing money.

The overall design of this liquid staking protocol treats funds security as a core goal when utilising the fund to earn rewards. The source code and architecture has already been open-sourced to the public.

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: Understanding RockX ETH2 Liquid Staking from Balance Sheet