The Everywhere LST: Subsquid

Pierre (Dorian) Spiegel
Thunderhead
pierre@thunderhead.xyz

Addison Spiegel
Thunderhead & MIT
addison@thunderhead.xyz

Abstract

Subsquid is a distributed query engine and data lake able to serve advanced queries at a cheaper cost than traditional RPCs. It is built on Arbitrum and uses a typical DePIN architecture, making it a perfect network to implement the "everywhere lst" thesis. Staked SQD (stSQD) is advantageous to traditional staking because of its optimized APR, hassle-free infrastructure management, and accessible DeFi ecosystem. The Staked SQD protocol is a set of secure, trust-minimized contracts governed by Staked SQD holders that onboards operators, rebalances for maximal rewards, and more.

1 Introduction

The Subsquid network consists of a network of hundreds of workers (currently 750). These workers are non-trivial to run, require significant amounts of infrastructure and earn rewards based on worker performance (e.g uptime). Additionally, there is a bond to run a worker (100k SQD, roughly $8k USD). If a staker does not meet this threshold, they must delegate and earn inferior rewards; worker self-stake earn 20% APR while delegators earn 10%. StakedSQD is essentially a pool - thus stakers can earn worker APR with any stake-size. The protocol automatically rebalances between delegation and self-stake to optimize the network's reward formula [2]. From a UX perspective, users simply deposit SQD for stSQD and start earning rewards instantly. The contracts automatically optimally apportion the SQD across a diverse and sophisticated operator set. Users also do not need to run any infra, rather they watch their balance increment every block. Lastly, the entire protocol is non-custodial because stSQD holders have veto rights prior to any governance action.

2 Architecture

2.1 Shares and Rebasing

Lido's rebasing stETH relies on extrinsic transactions for discrete daily reward distributions with an accounting system of underlying shares and an increasing yield factor ($balance = shares * yieldFactor$) to modify user balances. Lido rebases once a day which increases all user balances in one chunk. However, Staked SQD offers near-instant unstakes so a Lido-esque reward scheme would allow malicious actors to mint tokens, wait for the rebase, and then instantly unstake, diluting other users' deposits. To remedy, we update the LST with the rewards earned for the past epoch, and rather than distributing all rewards to users instantly, they linearly earn rewards over the span of the next epoch. Thus ensuring that all users earning rewards are actually contributing to the protocol. Our share calculation is done by

$$yieldFactor = e_{supply} + \frac{rewards * (t - e_{timestamp})}{epochLength} \text{if } e_{timestamp} < t < e_{timestamp} + epochLength$$

where e is the state at the end of the last epoch and t is the timestamp. The new share price is calculated solely based on onchain data from Subsquid contracts.

2.2 Operators & WorkerManagers

There are two types of yield on Subsquid: worker yield and delegator yield. Rewards are paid out to both parties every epoch, with workers earning 20-30% and delegators earning 10%. In order to be a worker, one must bond 100,000 $SQD and run complicated worker infrastructure. Rewards are paid out to workers based on the utilization rate of the network and the worker's liveness/tenure. Delegators can bond any amount of tokens to a worker and they earn half the rewards of the worker proportional to their stake.

2.2.1 Onboarding Operators

One of the most important parts of Staked SQD is the operator set. It is imperative that operators are of the highest quality and reliability as possible. Operators are selected in a two step process. Thunderhead and the community identify and due-diligence operators, indexing on technical prowess, reliability, contributions and other factors. Operators then sign an SLA and a proposal is created to onboard them to the protocol. The proposal contains the operator manager address and operator fee. Operators are compensated as either a percentage of rewards or a fiat equivalent amount of SQD. stSQD holders will vote on this proposal; if a certain percentage of holders reject the proposal, the operator will not be onboarded (see governance section for more detail). Once an operator is added the Staked Squid protocol will deploy a new WorkerManager.

2.3 WorkerManager

A caveat of Subsquid is that the pending reward of workers is an internal variable, thus it is not possible to perform a ready only query of an operators pending rewards. We instead must claim the rewards and observe the resultant balance change. This means that in order to measure the reward output of different operators, the protocol must have an address solely responsible for the workers of that operator. WorkerManagers are contracts deployed by Staked SQD when governance adds a new operator. Each WorkerManager is the staker and manager of all workers that an operator runs. Upon every rebase the WorkerManager claims the rewards of all workers and reports that back to the main contract.

2.4 Registering Workers

In order to maximise the percentage of SQD staked we have a semi-automatic staking system. StakeStack is a double ended queue (allows for removal and addition at the front or back) that contains an operator and allowance of number of workers they can stake. At any time an operator can register workers with metadata and peerIds of their infrastructure. The number of workers they can register at any given time is determined by the amount of free SQD tokens and whether there is enough SQD to fill their slot (and previous slots) in the StakeStack.
In addition, if there is a situation where there are insufficient workers in the StakeStack to maximise working capital, there is functionality to delegate tokens to any worker of choice.

2.5 Deregistering Workers

Unstaking workers is done in two transactions: deregistering and withdrawing. After one deregisters a worker, they have to wait one epoch (which is 100 blocks) before withdrawing the bond. Surplus SQD from incoming stakes will typically fulfill user's burns. However, if it is a larger burn and there is no available SQD to fill it, Staked SQD will automatically deregister enough workers pulled from UnstakeStack (a double ended queue containing the peerId of a worker), to fulfill the burn. Once the epoch ends and the bond is able to be withdrawn, a user can call *redeem* which will withdraw the bond from the deregistered workers and transfer SQD to the user.

2.6 Rebasing & Rewards

Rebasing is done by calling the rebase function on the contract with no parameters. Rebasing claims all delegation rewards and then iterates through all operators and claims rewards from their respective WorkerManager. Each operator has their own fee, which is a percentage of their rewards or some fiat amount of SQD. Operators can claim their respective fees. There is also a service fee, which is a share taken from the entirety of rewards as a management

fee for the protocol. The yield factor calculation is updated with these rewards to increase user balances over the next 24 hour period.

3 Vesting Contracts

A supermajority of SQD supply is vested[1]. However, institutions, early investors, and employees are still able to stake these tokens to support the collaterization of the network. Staked SQD is integrated into the SQD protocol vesting contracts to enable locked holders to stake their tokens. Investors with non-trivial token balances find it arduous to find worker operators. If they opt for delegation, they will need to continually rebalance their delegations to optimize their APR. Staked SQD solves this problem by automatically rebalancing between self-stake and delegation to maximize rewards. Investors are often busy people - Staked SQD offers a one-click solution to earn premier rewards and simultaneously support the network.

3.1 Implementation

The VestingManager is a contract that allows Subsquid vesting contracts to use Staked SQD. The VestingManager allows vesting contracts to mint and burn stSqd and claim rewards to an external address while ensuring it cannot transfer principal out. When a vesting contract calls *mint*, the Manager increases the balance of their address in a a ledger called *staked* and transfers the corresponding amount of stSqd to their vesting contract address. When a vesting contract wants to claim rewards they call *claim* on the VestingManager, which transfers $balance + unstaked - staked$ to a beneficiary address of the vesting contract's choice.

4 Governance

It is important that Staked SQD is non-custodial, trust-minimized, and user controlled. Maintaining a separation of powers within governance is a key part of this. The protocol has two-pronged governance structure. A multisig with Thunderhead team members and associates may create proposals for the protocol. StakedSQD holders may veto proposals within the voting window. This ensures that holders approve of all operators onboarded, changes to the protocol, and more.

4.1 Usage

Governance will primariliy onboard and manage operators. Contract upgrades and other parameter changes will likely be unnecessary. A single proposal is created for each operator onboarding or other parameter chagne.

5 Conclusion

Users and institutions may interact with the protocol at stakedsqd.fi and institutional.stakedsqd.fi. Users can mint stSQD in less than a minute and start earning hyper-optimized rewards immediately. Users can retrieve their SQD in 30m. The protocol is non-custodial and trust-minimized.

References

[1] Subsquid Labs. *Subsquid Network Tokenomics*. URL: https://docs.subsquid.io/subsquid-network/tokenomics/.

[2] Subsquid Labs. *Subsquid Network Whitepaper*. URL: https://docs.subsquid.io/subsquid-network/whitepaper/#reward-rate.