GYSR Token and Core

Alex Koren
alex@gysr.io

Devin Conley
devin@gysr.io

October 9, 2020

Abstract

This technical whitepaper introduces and describes GYSR Token and Core. It covers contract mechanics, creation, funds flows, and security. The whitepaper describes economics, theory, and design decisions for each component of the system.

1 Motivation

We built GYSR to make yield farming and token distribution easy and fair. We want to align creators and investors so that projects have the greatest chance of succeeding. We want GYSR to incentivize users to invest in the long term value of those projects by rewarding financial commitments and participation. GYSR is built to be an easily configurable yield farming platform that connects project owners and token creators with investors who believe in the work and want to be a part of its growth.

2 Introduction

GYSR is a configurable on-chain system of smart contracts on the Ethereum blockchain. It implements a generalized mechanic for token distribution and incentive programs. This mechanic lets creators promote a particular useful behavior and incentivizes investors to take long term holding strategies and participate in the project ecosystem. Further, the GYSR system is designed as a platform for decentralized finance projects, both representing a global index for investors, and also providing continuous funding to those projects.

The system revolves around three core contracts [1]:

- The Geyser Factory contract ("Factory")
- The Geyser contract ("Geyser")
- The $GYSR ERC20 [2] contract ("$GYSR")

The first of the three contracts, the Factory, allows any Ethereum user to generate and deploy their own Geyser. They are able to select a contract address defining the token they'd like to distribute ("reward token") as well as a contract address defining the token that will need to be staked in order to earn that distribution ("staking token"). There is further configuration available, described in Section 4.1, Geyser Deployment.

The Geyser contract implements the core funding, staking, and reward mechanisms. Each time funding is added, an unlock schedule is defined for those reward tokens. The Geyser allows the deposit (staking) of staking tokens by any user in order to earn a portion of the reward tokens that are unlocked. This reward is distributed when the user withdraws (unstakes) their staking token. The Geyser contract is further described in Section 4, Geyser Mechanics.

The platform value of GYSR is based upon the $GYSR ERC20 token. $GYSR can be spent, at the time of unstaking from a Geyser, to further multiply accrued share of reward tokens. This new component was introduced to further align incentives, act as a diversified investment asset, and provide a source of continuous funding for projects. The theory and economics is explained in detail in Section 5, $GYSR Theory.

3 Background

3.1 Token distribution

Historically, token distribution has been done via 3 methods:

- Initial Coin Offering
- Airdrop
- Initial Exchange Offering

These methods have been performed with varying degrees of success. Initial coin offerings have been mired in regulatory confusion, often resulting in tokens categorized as securities [3]. Airdrops are paradoxical; token creators are devaluing their new asset by presenting it as cheap to mint and distribute. Finally, initial exchange offerings centralize ownership and value-setting of the token, antithetical to a decentralized medium of trust and exchange.

3.2 Ampleforth geyser

On June 23rd 2020, Ampleforth launched a new innovative distribution strategy using a geyser[4] which distributes tokens to AMPL/ETH market makers. This system is completely on-chain, and incentivizes investors to hold long term and provide market liquidity. Ampleforth's geyser ensured that enough AMPL was available for new actors to invest while disincentivizing large shorts.

This token distribution strategy is also known as Continuous Vesting Token Distribution (CVTD). Further explanation of the core of a CVTD's mechanics are out of scope for this paper and are well explained in Ampleforth's documentation [5]. If unfamiliar with geyser distribution, we recommend reading Ampleforth's CVTD RFC before continuing.

The Ampleforth geyser has been highly effective and is the primary inspiration for the work described in this whitepaper.

4 Geyser Mechanics

The Geyser contract implements the standard IStaking interface as defined in EIP-900 [6]. The contract is adapted from and extends the core of Ampleforth's geyser. In general, the Geyser creator has a supply of some ERC20 token, the reward token, that they'd like to distribute and/or use as incentive. In Ampleforth's case, the coin was AMPL. In addition, the Geyser creator will define an ERC20 token, the staking token, that will be deposited into the Geyser by users. For Ampleforth's geyser, this was the Uniswap LP token representing AMPL / ETH market liquidity. This incentivized AMPL owners to create a more liquid market. Users will earn reward tokens based on how much staking token they've deposited (staked) and how long they keep it staked before withdrawing (unstaking).

To create a Geyser, the creator first executes a transaction with the Factory to construct a new Geyser with the desired configuration. The creator can then fund the Geyser by locking up a certain amount of their reward token to be distributed over time. Once the Geyser has been funded, any user can stake some amount of the staking token, and will immediately start to accrue rewards. Finally, when the user unstakes their tokens (optionally applying a $GYSR bonus, defined in 4.4, Unstaking), their earned portion of the reward tokens will be distributed to them. The available reward tokens are defined by the funding schedules the creator configured when funding the Geyser.

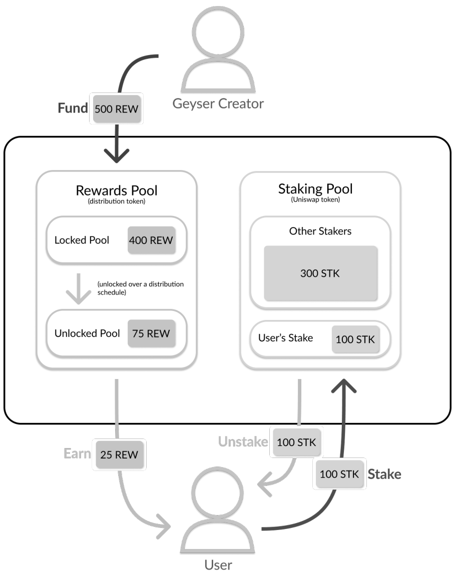


Figure 1: Path of reward token from the creator, held in the Geyser, and distributed during unstake

4.1 Deployment

The Geyser creator is able to define the reward token, staking token, and configure the CVTD time-based multiplier during construction. The time-based multiplier is a function of a minimum bonus (must be greater than or equal to zero), the maximum bonus (must be greater than or equal to the minimum), and the period over which the bonus is linearly earned on a given stake.

4.2 Funding

Only the creator is able to fund the Geyser reward pool. When a funding operation is executed, the owner will specify the amount of reward token to deposit, the period over which that reward will be unlocked, and optionally a time offset to begin the unlocking period.

A Geyser may be funded multiple times, but there is a hard limit on the number of active funding schedules. If the limit is reached, the owner must wait until an older schedule expires before funding the Geyser again.

4.3 Staking

Any user may stake their staking tokens with the Geyser. This operation does not differ from the original Ampleforth implementation.

4.4 Unstaking

Similarly to the CVTD, the user may unstake their staking tokens at any time, receiving their portion of reward tokens based on accrued staking share seconds and the earned time-based multiplier.

However, during unstaking, the user is also able to optionally apply $GYSR for an additional multiplier towards their reward. The following is used to calculate the earned rewards:

S_{user} : user share-seconds

S_{total} : total share-seconds

B_{time} : time bonus (defined in the CVTD)

B_{GYSR} : GYSR bonus (defined in section 7, $GYSR Multiplier)

U : total unlocked rewards defined by funding schedule

$$\text{reward} = U \cdot \left(\frac{B_{\text{time}} B_{\text{GYSR}} S_{\text{user}}}{S_{\text{total}} - S_{\text{user}} + B_{\text{time}} B_{\text{GYSR}} S_{\text{user}}} \right) \tag{1}$$

Example

Bob creates a Geyser and funds it with 10,000 of his ERC20 token, $REW, set to distribute it over 10 days. These $REW will be rewarded for anyone who stakes the staking token, $STK, in the Geyser. In addition, Bob configures the Geyser to provide an additional 3x time-based multiplier earned over a 5-day staking period. Alice stakes 100 $STK on day 0. On day 6, Jim stakes 1000 $STK. Finally, on day 10, Alice decides to unstake her 100 $STK. Alice has contributed 100 $STK x 10 days of staking. Since Jim has contributed 1000 $STK x 4 days of staking, the total pool amount is 5000 share-days. However, Alice will earn the full 3x multiplier because she stayed in the Geyser for more than the 5 days. In addition, Alice uses 1 $GYSR during her unstake, and the current multiplier for 1 $GYSR is 2x. Therefore Alice is entitled to:

$$\text{reward} = 10,000 \cdot \left(\frac{3 \cdot 2 \cdot 1000}{5000 - 1000 + 3 \cdot 2 \cdot 1000} \right) = 6,000 \ \$REW$$

4.5 $GYSR Withdrawal

All of the applied $GYSR is sent to the contract address. This balance can be withdrawn by the Geyser creator at any time. This acts as a continuous funding mechanism for the creator's continued development and support of their project.

4.6 Preview

The user may execute a read-only preview operation which calculates the expected reward distribution based on the amount unstaked and number of $GYSR applied. This function also returns estimated overall multiplier, raw share seconds that would be burned, and total unlocked rewards. With this added capability, users are more informed and can act with discretion when investing and unstaking.

5 Theory

The mechanics of GYSR have been designed with various key goals in mind. This section will discuss the thinking and theory behind those goals.

5.1 Independent utility

The Geyser contract will maintain all core usability and functionality entirely independent of the value and even existence of $GYSR. In no way is $GYSR a necessity for the Geyser to operate. The core Geyser contract can safely be used on its own in order to promote long-term holding, market liquidity, and targeted distribution. This is intentional such that the success and growth of $GYSR is more natural and only aligned with its usefulness as an investment accelerant.

5.2 $GYSR as a universal diversified asset

The $GYSR multiplier mechanic also offers users an opportunity to "hedge their bets" when investing into early stage projects. $GYSR token can be spent universally to capitalize on increased returns for any Geyser-distributed token and can therefore be considered a diversified asset. $GYSR is effectively an index fund across all Geyser-distributed tokens.

5.3 $GYSR as a source of continuous funding

Many early-stage projects struggle with funding. While token sales and traditional investors can provide significant capital, that money comes in sporadic chunks and can often contradict the core principles of decentralized finance. Additionally, the project's token is usually tied to future value and is not liquid.

As a naturally diversified store of value, the in-flow of $GSYR can provide another option. By incentivizing the end user to spend $GYSR during unstaking, we provide a continuous and stable source of funding to the Geyser creator.

5.4 Responsive value

To increase the likelihood of, but not the reliance on, $GYSR being used, a scaling mechanic was implemented. This mechanic increases the multiplier earned when $GYSR usage is low and decreases when $GYSR usage is high. This usage is Geyser-specific so that one Geyser's market and utilization will not affect another's. This scaling mechanic also lets us ignore the disparities in market cap, individual value, and liquidity, across different tokens.

6 Design decisions

6.1 Introduction of $GYSR

Certainly the most notable design decision is the introduction of $GYSR token. This was added as an additional component for GYSR as a way to further align incentives and sustainably promote the ecosystem of a new project. In addition to existing benefits, $GYSR acts as both an index of diversified early-stage investment to the user, and a source of continuous funding to the creator.

6.2 Shares instead of tokens

In the Ampleforth CVTD, all staking calculations use shares instead of raw token count. This is done to handle the dynamic nature of the AMPL token balances due the rebasing mechanic. While the large majority of ERC20 tokens do not require this, we decided to maintain the functionality for completeness.

6.3 Bonus considered in total share seconds

In the Ampleforth CVTD, the bonus share count is not considered in "total share seconds" when computing the proportion of the unlocked reward pool owed to the user when unstaking. This means that in certain edge case, the user could theoretically be owed a reward greater than the unlocked pool. This transaction would obviously fail.

We decided to consider the "multiplied user share seconds" as part of the total in order to bound that proportion to $(0, 1]$. This is especially important given the added $GYSR multiplier.

6.4 Limited number of active funding schedules

In the Ampleforth CVTD, there is a hard limit on the number of funding schedules. This is to ensure that the accounting calculations, which grow linearly with the number of funding schedules, do not cause the transaction to exceed the gas limit.

We keep this restriction and set the hard limit to 16 concurrent active funding schedules. This number was selected as a balance between providing flexibility and limiting complexity. We make the assumption that 16 active funding schedules should be more than enough for any use-case or incentive program. Our tests also show that this will keep transaction costs 20x under the current gas limit (12.5M) even in the worst case.

Additionally, we have implemented "expiration" functionality for old funding schedules. This fully deletes and removes the stale schedule in order to free up space for further funding.

6.5 Configurable time bonus

The time bonus parameters are configurable at the time of construction. This is exposed in order to be flexible to the goals and incentive model desired by the Geyser creator. There is no hard limit on the max time bonus, and it can also be removed entirely by setting both the min and max values to 0. There is a tradeoff to not implementing a maximum time bonus. If the value is set too high, there will be a race to unstake first after accruing a high time bonus because it can disproportionately result in larger rewards for the earlier unstake and take away from the expected rewards of those that unstake afterwards. It is left up to the Geyser creator to be responsible for the potential misalignment of unstake timing and it's possible harm to fair distribution.

6.6 Unstaking order is first-in, last-out

There is a question about which stakes should be unstaked first. Since a user can stake tokens in separate transactions, some stakes will have accrued larger multipliers than others. We have decided to continue with Ampleforth's model of "first-in, last-out" when unstaking. It provides a simple rule set for users to follow, but means that users will not have the flexibility to take out their stakes that may have reached their maximum time-based multiplier without unstaking any newer stakes first.

6.7 Minimum $GYSR applied is 1.0

In order to keep the mathematics manageable, the minimum amount of $GYSR than can be applied is 1.0. Users can choose to apply 0 $GYSR and not earn any additional multipliers, but cannot apply between 0 and 1 exclusively. This is due to the nature of logarithms, which are relied upon for multiplier calculations, resulting in negative numbers for values between 0 and 1. We simply implement an exception case for 0 $GYSR being applied.

6.8 GYSR bonus is applied to the entire unstake

When a user spends $GYSR, that bonus is applied to the entire amount being unstaked when computing the earned reward. This is done for the sake of simplicity and for ease of use.

This does mean that there is some variability in the value of spending $GYSR based on the amount being unstaked. However, this already true due to other GYSR dynamics such as time bonus, responsive multiplier function, and order of different users unstaking.

6.9 Time delayed funding ("Boiler")

We introduce the concept of a "Boiler", which is a Geyser that has been funded for unlocking over a future time period. This allows a project creator to lock up future rewards and build momentum for an upcoming launch and also lets early investors get a head start on earning rewards. To configure a Geyser as a "Boiler", the owner will simply specify a future start time for the unlocking period in the fund operation. This also generally allows for more control and flexibility when defining an overall funding schedule.

6.10 Total $GYSR supply is 10M

We have settled on 10M for the total supply of $GYSR token. This decision is based on the results of simulation for $GYSR token economics, where we optimized for price stability and broad utility. The theory, method, and results of these experiments will be described more thoroughly in a supplementary paper.

7 $GYSR Multiplier

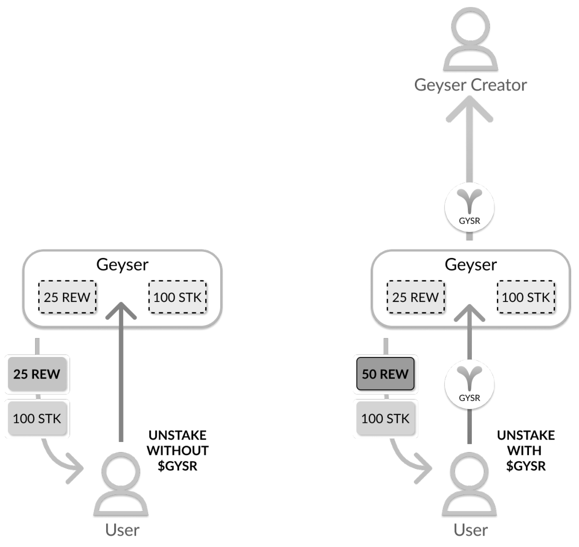


Figure 2: The application of $GYSR during unstake will multiply the proportion of reward tokens earned, defined by the formulas in 7.3.

7.1 Challenges

$GYSR is a universal multiplier of shares representing other assets. These reward assets have highly variable value, total supply, and unlocking schedules. Further, these numerous token markets are completely independent and unaware of each other. That said, $GYSR must still converge to some common value.

In order to promote healthy financial ecosystems for the reward token, applying more $GYSR on a single unstake operation should compound the multiplier. The more $GYSR a staker spends, the higher the multiplier on their share of distributed rewards. The function should allow this value to scale up, but still maintain stability.

Any responsive pricing system like this is going to become a target for exploitation. A user with large amounts of $GYSR could potentially abuse this scaling mechanic and receive massive rewards in return. Since rewards are received from a shared pool, their gain would reduce the portion of rewards earned by other users. There must be some diminishing return on $GYSR spent. Similarly, a single wealthy user could artificially distribute their activity across multiple accounts or multiple transactions, with the goal of manipulating the system. This is known as a Sybil attack [7]. We must be resilient and agnostic to this type of behavior.

7.2 Criteria

Given this complexity, there are a few critical criteria that must be met:

- The amount of $GYSR required for the multiplier must be unrelated to both the total supply of the Reward Token and the Staking Token
- There must be some natural limitation on the multiplier to reduce exploitation
- The multiplier must be responsive to meet Geyser-specific usage
- The multiplier must be resilient to manipulation by bad actors

7.3 Multiplier function

With the above challenges and criteria in mind, the following function was designed.

Let

D_{total} : total reward tokens distributed

D_{GYSR} : total reward tokens distributed that were boosted by \$GYSR

X : number of \$GYSR applied to an unstake operation

Let R represent the proportion of total distributed reward tokens which have had \$GYSR applied. Note that in this calculation, we do not consider the quantity of \$GYSR applied to each operation as that value is already implicitly included when computing the reward amount.

$$R = \frac{\mathrm{D}_{\text{GYSR}}}{\mathrm{D}_{\text{total}}} \tag{2}$$

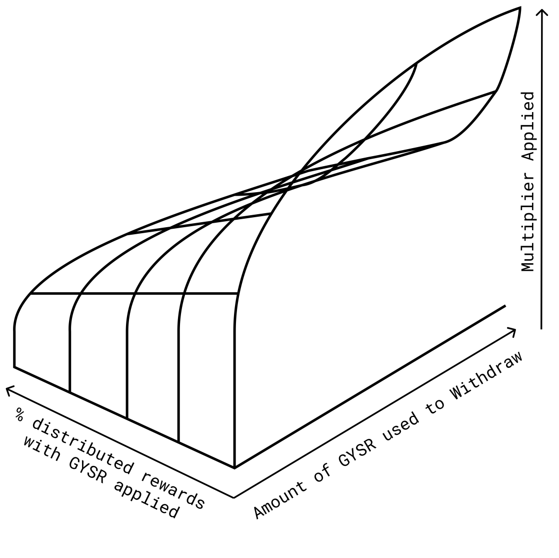
Finally define the multiplier B_{GYSR}

$$B_{\text{GYSR}}(X) = 1 + \log_{10}\left(\frac{0.01 + X}{0.01 + R}\right) \tag{3}$$

This value, B_{GYSR}, is used in the unstaking process, defined in Section 3.4.

This function achieves a few notable objectives and logical key values:

- Stable but responsive
- Agnostic to unique accounts and number of unstake operations
- 1 \$GYSR -> 3x at 0.0 usage
- 1 \$GYSR -> 2x at 0.1 usage
- 10 \$GYSR -> 2x at 1.0 usage
- Absolute limit of 10x given 10M total \$GYSR supply and 0.0 usage



Multiplier Applied
Amount of GYSR used to Withdraw
% distributed rewards with GYSR applied

Figure 3: The curve of \$GYSR multiplier based on usage and quantity applied

8 Web Application

In order to make the Factory, Geysers, and \$GYSR easy to access and use, a web application was developed in tandem with the smart contracts. The web application surfaces all of the core functionality of the smart contracts, including the ability to deploy new Geysers through the Factory, stake and unstake to and from a Geyser, and manage any Geysers a user has created.

8.1 Geyser Creation

The Factory can be accessed by any user and used to generate a new Geyser contract. The following values are required:

- Reward token address
- Staking token address
- Minimum time multiplier
- Maximum time multiplier
- Time multiplier duration

With this configuration, any user who stakes the staking token will earn the reward token based on their quantity staked, the time they've remained staked, and the multipliers applied to that time. The time multipliers configurable within the web application are within zero and five, inclusive. This is to provide suggested values, but the contract does not specifically set an upper limit for the time multiplier. In addition, the duration of the period to earn the time multiplier is defined in days on the web application. These days are converted into seconds before deploying the contract and thus can be given as fractional values.

There is a known risk in the contract that allows a creator to include an extremely high time bonus multiplier. This creates a race for stakers to unstake and earn the large multiplier first, removing a disproportionate amount of the rewards. The Geyser Factory and Geyser contracts do not explicitly limit the time bonus multiplier value because we want to give Geyser creators full flexibility, however, the web application limits the multiplier via the interface to a normal use maximum: 5x. Geyser creators can override this by interacting directly with the contract.

8.2 Geyser Management

The web application offers an interface to creators to manage Geysers they've created. The first operation available on this page is to fund the Geyser. The creator can select the number of tokens to send to the distribution pool where they begin in the locked state. The creator also defines a distribution period over which the locked tokens will become unlocked and available as withdrawable rewards. The distribution period is defined in days, but can be provided as a fractional value as it will be converted to seconds before being sent to the smart contract.

The second operation available is to withdraw $GYSR that has been spent on the Geyser. This methods takes an argument for the withdrawal amount and can only be called by the owner. The amount of $GYSR specified will be sent to the owner's address.

8.3 Geyser

For a given Geyser, the web application will provide an interface for interacting with the contract. Users can:

- Stake tokens
- Unstake tokens and optionally apply $GYSR
- See personal and global stats for the Geyser

When staking or unstaking, the web application will provide estimates for the expected rewards. Due to the nature of the contract, all estimates given are subject to change on any action taken. This can involve additional tokens being staked, other users unstaking, $GYSR applications, or additional funding added to the Geyser.

In addition, when loading a Geyser via the web application, the Geyser is accessed by address. Since the web application cannot differentiate between the address of a GYSR provisioned Geyser and a contract with an identical ABI, the web application must be responsible for checking the legitimacy of the contract. When loading a contract, the web application will query the Geyser Factory's list of created Geysers to see whether or not the contract was created by the Geyser Factory or was manually deployed (and potentially tampered with). The web application will then surface warnings to users attempting to access Geysers that were not deployed through the Factory.

References

[1] GYSR.io. *GYSR core*. URL: https://github.com/gysr-io/core.

[2] *ERC-20 Token Standard*. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[3] SEC. *SEC Spotlight on Initial Coin Offerings*. URL: https://www.sec.gov/ICO.

[4] Ampleforth. *About the Geyser*. URL: https://www.ampltalk.org/app/forum/ampl-geyser-19/topic/about-the-geyser-21/.

[5] Ampleforth. *Ampleforth CVTD*. URL: https://github.com/ampleforth/RFCs/blob/master/RFCs/rfc-1.md.

[6] *EIP 900 - Staking*. URL: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-900.md.

[7] John (JD) Douceur. "The Sybil Attack". In: *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*. Jan. 2002. URL: https://www.microsoft.com/en-us/research/publication/the-sybil-attack/.