Valkyrie Swap

Decentralize Financial System

V2.4

Valkyrie Group

Julius Petersen&Mandalorian

2020.11

Table of Contents

What is Vswap

Vswap is an automatic token exchange protocol based on Ethereum. It is designed for ease of use, high gas utilization, censorship resistance, and zero draw.

Ease - of - use:

Exchange of Token A for Token B can be completed in Vswap as long as one transaction is issued. In other exchanges, two transactions may be issued: the first transaction is converted into Token A for some kind of media currency, such as ETH, DAI, etc., and then the second transaction is converted into Token B.

Gas efficiency:

The amount of gas consumed on Vswap is the lowest of several major de-centralization exchanges on Ethereum, which means that miners have to pay the fewest fees in Vswap.

Censorship resistance:

The anti-censorship is reflected in the fact that there is no threshold for the new Token to be put on Vswap, and anyone can put any Token on Vswap.

Zero rent extraction :

In the Vswap protocol design, the development team does not extract fees from the transaction, and all fees from the transaction are returned to the liquidity provider.

An automated market maker (AMM) traditionally trades in an Order Book, which records the direction of buying and selling, quantities and bids. The exchange matches buyers and sellers, and when the highest price in the order book falls below or equals the lowest price, a deal is made and a new transaction price is created. The traditional transaction has the following characteristics: the market must have the user to carry on the order, must have a certain amount of order (market flow). Orders must overlap to close, that is, the purchase price is higher than or equal to the selling price. Assets need to be stored on an exchange.

In the order-book model market, buyers expect to buy the underlying asset at the lowest price, while sellers expect to sell the same asset at the highest price. If a transaction is to take place, buyers and sellers must agree on a price: either the buyer raises his offer or the seller lowers it. If neither side changes its offer, then it is up to market makers to get involved. In short, a market maker is an entity that promotes trading. It can place orders in both buying and selling directions. It allows participants who want to trade to complete the transaction by matching the order with the market maker's, instead of waiting for the counterparty to appear, which greatly improves the liquidity of the market.

Vswap is a system deployed on Ethereum, which can perform about 15 transactions per second. This is not feasible for order book exchanges, mainly because the order book model relies on one or more external market makers to make a continuous market on an asset, while Ethereum's TPS is low and the blockchain network does not support high frequency trading by market makers. Without market makers, the exchange's liquidity would immediately be reduced, which would be a poor experience for users.

Vswap adopts the liquidity pool plus constant multiplication formula, which is an automatic market maker (AMM) mode, to realize the exchange of assets. The automatic market maker mode does not require the buyers and sellers to issue orders, nor does it overlap the orders of the buyers and sellers, so it can carry out free trading.

Liquidity pool: Use liquidity pool to provide transactions between buyers and sellers. Market makers simply put Tokens into the liquidity pool.

Constant multiplication formula: The system automatically calculates the buying and selling price according to the amount of Tokens in the liquidity pool.

Liquidity Pool

A liquidity pool is a general term for all tokens and funds locked in smart contracts. Liquidity is the conversion of funds into tokens, or the conversion of tokens into funds.

A complete flow pool is divided into two parts, each representing a different currency, and becomes a trading pair. In Vswap V1, there is ETH and ERC20 Token, and in Vswap V2, direct exchange of different ERC20 tokens is supported. So the flow pool in Vswap V2 can allow different ERC20 tokens on both sides, in which the ETH will convert into ETH Token automatically. For simplicity, take the ETH-ERC20 trading pair as an example.

Vswap aggregates all the market-makers' ETH to the left of the liquidity pool and all the ERC20 to the right of the liquidity pool. If a user buys an ERC20 token, the ERC20 token is transferred to the user from the right side of the pool, and the ETH paid by the user is added to the left side of the pool. The system then recalculates the price in the pool and waits for the next transaction.

Vswap needs to rely on external funds to provide liquidity for smart contracts, and the users who provide liquidity to Vswap's liquidity pool are called liquidity providers. When the Liquidity provider adds Liquidity to Vswap, Vswap will cast a Liquidity Pool Token(LP). The amount of LP Token cast is related to the proportion of the capital added by the user in the Liquidity Pool. The Liquidity provider can choose to destroy the Liquidity Token held by the provider at any time. In order to encourage more liquidity to be provided to Vswap's liquidity pool, Vswap takes 0.3% of the total value of each smart contract executed as a transaction fee and rewards the fee to liquidity providers and Token holders who add funds to Vswap's liquidity pool in a variety of ways.

Constant Multiplication Formula

Assuming that there is a flow pool of ETH-DAI trade pairs in Vswap, the user needs a method to determine the transaction price when using DAI to exchange with the ETH.

Vswap pricing model is very concise, its core idea is a constant product formula

$$x * y = k$$

Where x and y respectively represent the number of two kinds of assets in the liquidity pool, and K is the product of the number of two kinds of assets.

Assuming that the product K is a fixed constant, when the user uses X asset to exchange Y asset from the liquidity pool, the number of X assets in the liquidity pool will increase and the number of Y assets will decrease.

Since k is constant, as x increases by

ΔX,

you have to reduce y by

ΔY

to keep the equation constant

$$(x + \Delta X) \times (y - \Delta Y) = k$$

The fee is not taken into account here. If the system needs to calculate the fee, the formula is as follows

$$(x + \Delta X_{\gamma}) \times (y - \Delta Y) = k'$$

$$k' > k$$

Indexes including

ρ=0.3%
γ=1-ρ

ΔX_γ

represents assets added to the pool after fees are deducted. The calculated k' will be greater than k due to the added funds in the liquidity pool, which will be derived later.

There will be some derivation of mathematical formulas. For the convenience of understanding, we will first derive the situation without transaction fee, and the derivation process including transaction fee will be put in the following text.

Transaction Fee not Included

Transaction price calculation

The calculation of transaction price is divided into two types:

InputPrice:

How many ΔY can be exchanged by putting ΔX Tokens into the Liquidity pool.

OutputPrice: After taking out ΔY Tokens from the flow pool, how many ΔX Tokens need to be put into the flow pool.

Among them:

$$\Delta x = \frac{\beta}{1-\beta} x$$

$$\Delta y = \frac{\alpha}{1+\alpha} y$$

The more Δx is added to the pool, the more $\Delta \mathrm{Y}$ can be taken out of the pool, but $\Delta \mathrm{Y}$ will only constantly approach the number of tokens existing in the pool, and will not exceed it, which means that the user can never buy out all tokens in the pool at one time. It can be seen that the constant multiplication formula can provide unlimited liquidity for the market, as long as the user wants to trade will be successful.

Trading Slippage

Transaction slippage point refers to: "When buying and selling tokens, the difference between the actual paid price and the expected transaction price is called transaction slippage point".

The purchase of $\Delta \mathrm{Y}$ Token from the flow pool requires $\Delta \mathrm{X}$, so the purchase price this time is:

$$P = \frac{\Delta \mathrm{X}}{\Delta \mathrm{Y}}$$

$$P = \frac{\Delta X}{\Delta \mathrm{Y}} = \frac{\alpha x}{\frac{\alpha}{1+\alpha} y} = (1+\alpha)\frac{x}{y}$$

$$P^{'} = \frac{x}{y}$$

$$\mathrm{SlippagePoint} = P - P^{'} = (1+\alpha)P^{'} - P^{'} = \alpha P^{'}$$

$$and$$

$$P^{'} = \frac{x}{y}$$

Represents the price in the current liquidity pool

It can be seen from the formula that the slippage point of the transaction is linear with

$$\alpha = \frac{\Delta \mathrm{X}}{x}$$

Slippage (deviation) amplitude depends on the

$$\frac{\Delta X}{x},$$

Bigger the pool is, the smaller slippage point will be. When the capital pool is fixed, the smaller online trading volume within a block time will lead the smaller slippage point.

Online trading volume within a block time is related to two factors:
the speed of block confirmation and the market price volatility.

Therefore, Vswap's trade slippage point of constant multiplication market making is highly correlated with the following three factors:

The size of the pool
Block confirmation speed
Market volatility

Current Liquidity Pool Price

The purchase of ΔY tokens from the current pool will definitely cause price fluctuations in the liquidity pool:

$$P = \frac{x'}{y'} = \frac{(1+\alpha)x}{\frac{1}{1+\alpha}y} = (1+\alpha)(1+\alpha)\frac{x}{y}$$

Among them: P is the current price in the liquidity pool after the purchase of token.

It can be seen from the formula that the coin price in the liquidity pool is a quadratic function of the buying quantity.

Arbitrageurs occur when the price in the pool is out of line with the external market price. Arbitrageurs monitor prices on exchanges around the world. Once arbitrage opportunities exist between the two exchanges, they can buy low and sell high on both sides to earn the middle spread.

Because of the existence of arbitrageurs, the price of Vswap will not be out of line with the global market. When the price in Vswap is higher than the external market, the arbitrageur will buy from the external market at a low price and then sell to Vswap at a high price. When the price of Vswap is lower than that of the external market, the arbitrageur will perform the reverse operation.

Transaction Fee Included

The following calculation reformulates the above formula in the case of transaction fees.
Calculate the transaction price

$$x'_{\rho} = x + \Delta x = (1+\alpha)x = \frac{1+\beta\left(\frac{1}{\gamma}-1\right)}{1-\beta}x$$

$$y'_{\rho} = y - \Delta y = \frac{1}{1+\alpha\gamma}y = (1-\beta)y$$

$k' > x'_{\rho}y'_{\rho} = (1+\beta(\frac{1}{\gamma}-1))xy > k$ *k '> k calculated after adding transaction fee*

In addition to the fact that K will increase in the case of transaction fees, there is another situation that will change the value of K. K will increase when liquidity is added to the pool, and k will decrease when liquidity is retrieved from the pool.

Among them:

$\alpha = \frac{\Delta x}{x}$

$\beta = \frac{\Delta y}{y}$

$0 \le \rho < 1$

$\gamma = 1 - \rho$

The current transaction fee is:

$$\rho = 0.3\%$$

$$\Delta x = \frac{\beta}{1-\beta} \cdot \frac{1}{\gamma} \cdot x$$

$$\Delta y = \frac{\alpha\gamma}{1+\alpha\gamma} \cdot y$$

Trading slippage

$$P = \frac{\Delta x}{\Delta y} = \frac{\alpha x}{\frac{\alpha\gamma}{1+\alpha\gamma} \cdot y} = \frac{1+\alpha\gamma}{\gamma} \cdot \frac{x}{y}$$

$$\text{SlippagePoint} = P - P' = \frac{1+\alpha\gamma}{\gamma} \cdot \frac{x}{y} - \frac{x}{y} = \frac{1+(\alpha-1)\gamma}{\gamma} \cdot \frac{x}{y}$$

The Token value in the liquidity pool

$$\frac{x'_\rho}{y'_\rho} = \frac{(1+\alpha)}{\frac{1}{1+\alpha\gamma} y} = (1+\alpha)(1+\alpha\gamma)\frac{x}{y}$$

Impermanent Loss Design

Participation swap for liquidity mining will lead a normal phenomenon: assets deposited in the current pool revealed less value than the previous value, this kind of situation is called "impermanent loss".Impermanent loss is defined as: the difference between Liquidity Minng profit and encryption currency stay in the wallet , formula is:

$$\mathrm{L} = profit_{lp} - profit_{hodl}$$

Impermanent loss is usually caused by price fluctuations in the liquidity pool. Here is an example

$$e \times t = k$$

$$p = t/e$$

$$e = \sqrt{\frac{k}{p}}$$

$$t = \sqrt{k \times p}$$

Where: e represents the number of ETH in the liquidity pool, t represents the number of DAI in the liquidity pool, p represents the price in the liquidity pool.

At the beginning the user adds to the liquidity pool

$e = 100$

$t = 10{,}000$

Current market price:

$$p = \frac{t}{e} = 100$$

$$k = et = 100 * 10{,}000 = 1{,}000{,}000$$

So let's say that now the external market price has changed, the price is going to be

$$p^{'} = 120$$

As a result of price differentials, arbitrageurs begin to work by trading to keep the prices in the pool in line with the external market. At this point, the latest amount in the liquidity pool is

$$e^{'} = \sqrt{\frac{k}{p}} = \sqrt{\frac{1{,}000{,}000}{120}} = 91.28709291752769$$

$$t^{'} = \sqrt{k \times p} = \sqrt{1{,}000{,}000 \times 120} = 10954.45115010332$$

Calculate the assets put into the pool:

$$\begin{aligned} \upsilon_1 &= e^{'} \times p^{'} + t^{'} \\ &= 91.2870929175\ 2769 * 120 + 10954.4511501033\ 23 \\ &= 21908.9023002066\ 5 \end{aligned}$$

Calculate the impermanent loss:

$$L = \frac{\upsilon_1}{\upsilon_0} = \frac{21908.9023002066\ 46}{22{,}000} = 0.9958595463\ 9384 \approx 99.59\%$$

After the price went from 100 to 120, assets added to the liquidity pool were only 99.59 per cent of what they had been before, losing about 4 per cent. As long as the participant does not withdraw the asset from the liquidity pool at this time, once the asset price in the liquidity pool returns to 100, the participant will have no loss, which is why it is called impermanent loss.

Impact of Impermanent Loss on liquidity provider: It can provide liquidity for the liquidity pool with small fluctuation between two currency prices.

Mean-reverting pairs: For example, the exchange of stable coins has the smallest fluctuation of the two pairs, which can reduce impermanent loss to the greatest extent.

Correlated pairs: there is positive correlation to trading, ETH/ZRX, for example, the two direction of currency fluctuations are basically consistent, with the rise and fall, the relative volatility between the two.

Uncorrelated pairs: the correlation of trading, such as the ETH/DAI, for this trade to provide liquidity, making the transaction fees may cover impermanent loss.

Direction of Inverse correlated pairs: Inverse correlated pair price fluctuations on the contrary, the relative range between the two is the largest.

According to the conclusion given by the formula, Vswap will subsidize and reward impermanent loss of some specific currency pairs.

Here is the formula for calculating impermanent Loss

$$impermanentLoss = \frac{2 \times \sqrt{priceRatio}}{1 + priceRatio} - 1$$

Where: $priceRatio$ is the ratio of price changes

The following is the derivation without fees

$$et = e^{'}t^{'} = k$$

$$\frac{e}{t} = P$$

$$\frac{e^{'}}{t'} = P^{'}$$

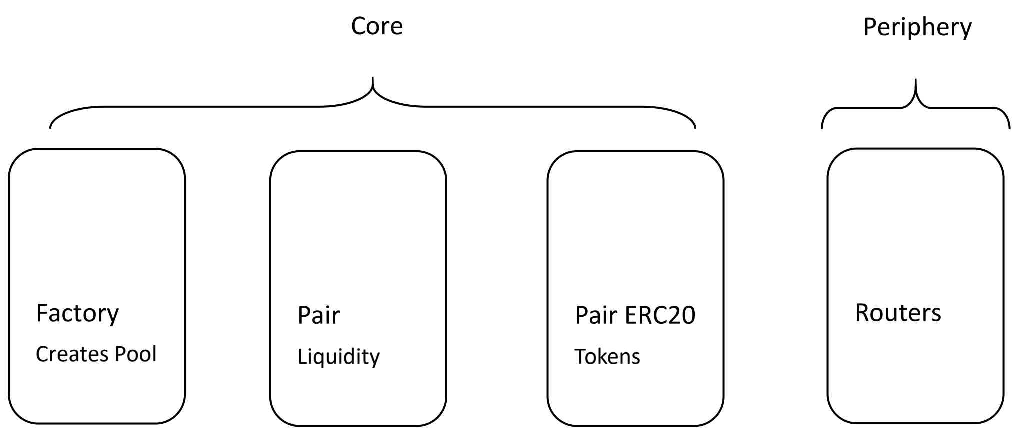
$$\frac{e'}{e} = \sqrt{\frac{P^{'}}{P}}$$

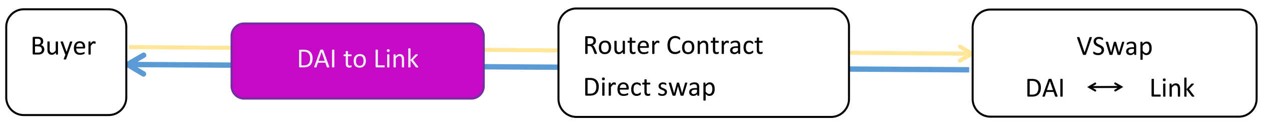
$$\frac{t^{'}}{t} = \sqrt{\frac{P}{P^{'}}}$$

$$priceRatio = \frac{P^{'}}{P}$$

$$\begin{aligned}
V_0 &= eP^{'} + t \\
&= e\frac{t^{'}}{e^{'}} + t \\
&= \frac{et^{'} + e^{'}t}{e^{'}} \\
L &= \frac{V_1 - V_0}{V_0} \\
&= \frac{V_1}{V_0} - 1 \\
&= \frac{2t^{'}}{\frac{et^{'} + e^{'}t}{e^{'}}} - 1 \\
&= \frac{2e^{'}t^{'}}{et^{'} + e^{'}t} - 1 \\
&= \frac{\frac{2e^{'}t^{'}}{et^{'}}}{\frac{et^{'}}{et^{'}} + \frac{e^{'}t}{et^{'}}} - 1 \\
&= \frac{\frac{2e^{'}}{e}}{1 + \frac{e't}{et^{'}}} - 1 \\
&= \frac{2\sqrt{\frac{P^{'}}{P}}}{1 + \frac{P^{'}}{P}} - 1 \\
&= \frac{2\sqrt{priceRatio}}{1 + priceRatio} - 1
\end{aligned}$$

Contract Structure and Transaction Flow





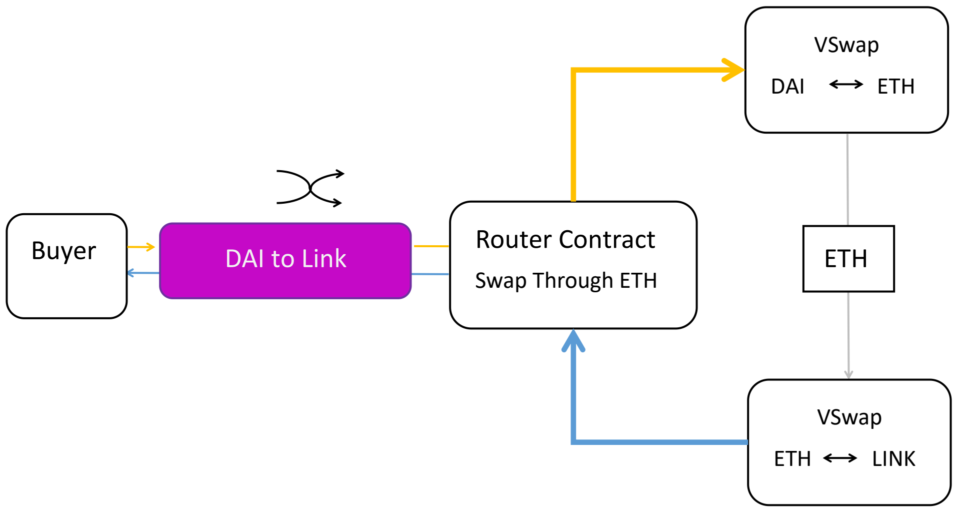
The swapExactTokensForTokens and swapTokensForExactTokens method on the Router contract can be invoked to make such a transaction.

The Exact term used in the names of these methods stands for the tokens you want to trade. In transactions from DAI to ETH, if you need a certain number of ETH in return, you use swapTokensForExactTokens.

On the other hand, if you want to trade an exact DAI amount for a corresponding ETH value, you will use swapExactTokensForTokens. This agreement runs through the smart contract.

In addition to direct exchange, users can still choose to exchange between two tokens, using the ETH as an intermediate token. This becomes useful when there is no pool of input and output tokens, but there is a pool between the ETH and two tokens.

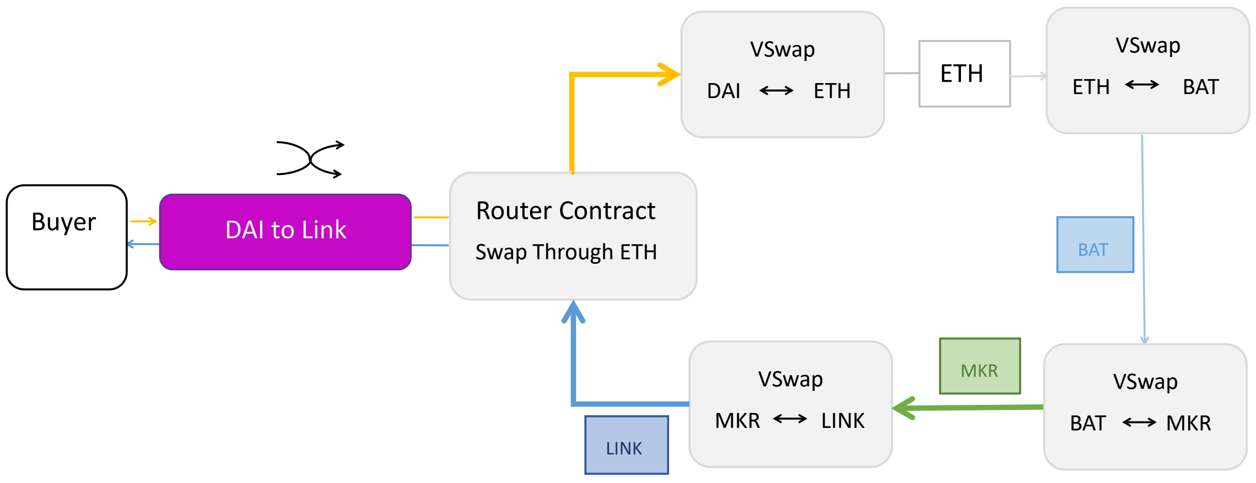
In the case of DAI and LINK exchanged via the ETH, the resulting exchange flow is as follows.



The corresponding soldesert solution to allow this transaction is swapExactETHForTokens and swapETHForExactTokens, and swapTokensForExactETH and swapExactTokensForETH.

The final way to exchange tokens is to route the exchange to multiple ERC20 tokens, or "any pair of ERC20 tokens," and then to the output token you want. Of course, native ETH tokens can also exist in any pair.

Consider the following illustration to move the DAI value through a series of tokens before reaching the required LINK output token.



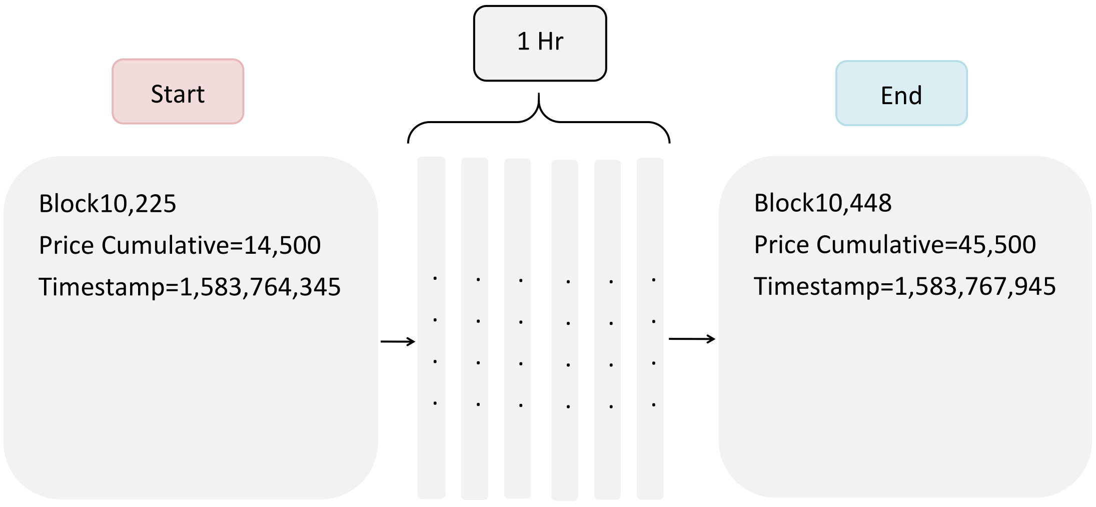
The price predictor mechanism allows developers to calculate the average token price based on the change in the price of the token over a number of blocks, which also represent a period of time through their timestamps.

The accumulation of this time period can be the last hour, 24 hours or more.

Traditional DEFI Swap provides token prices, but it does not store any historical values on the chain. Instead, Dapp developers are responsible for accumulating prices over a period of time to calculate the average price over that period.

These prices are known as "time-weighted average prices," or TWAPS.

The idea is to calculate the average price of a block by dividing the cumulative price (the token price of each block during the duration) by the timestamp duration (the timestamp at the end of the duration minus the timestamp at the beginning of the duration).The following diagram summarizes this calculation.



$$TWAP = \frac{priceCumulative^{End} \cdot priceCumulative^{Start}}{Timestamp^{End} \cdot Timestamp^{Start}}$$

$$= \frac{45500 - 14500}{1{,}583{,}767{,}945 - 1{,}583{,}764{,}345} = \frac{31000}{3600} = 8.61$$

Method to realize ZKRollup

The essence of ZK Rollup is to compress the user state on the chain and store it in a Merkle tree, and transfer the change of the user state to the chain, and at the same time guarantee it through the proof of ZK-SNARK (Zero-Knowledge Concise Non-interactive Knowledge Argumentation) The correctness of the user status change process under the chain. The cost of directly processing user status changes on the chain is relatively high, but only using the smart contract on the chain to verify the correctness of a zero-knowledge proof is relatively low. In addition, necessary transfer information will also be submitted to the contract along with the certificate, which is convenient for users to check accounts.

There are two types of roles in the ZK Rollup system: transactor and relayer,

Transactor, that is, ordinary user, corresponds to an external account on Ethereum. The user constructs the transfer transaction and signs it with the private key, and then sends the transaction to the relayer.
The relayer is responsible for collecting and verifying the user's transactions, then packaging the transactions in batches, and generating the ZK-SNARK Proof, and finally submits the core data in the user transaction, the ZK-SNARK Proof and the Merkle root of the new global user state to the chain Smart contract.

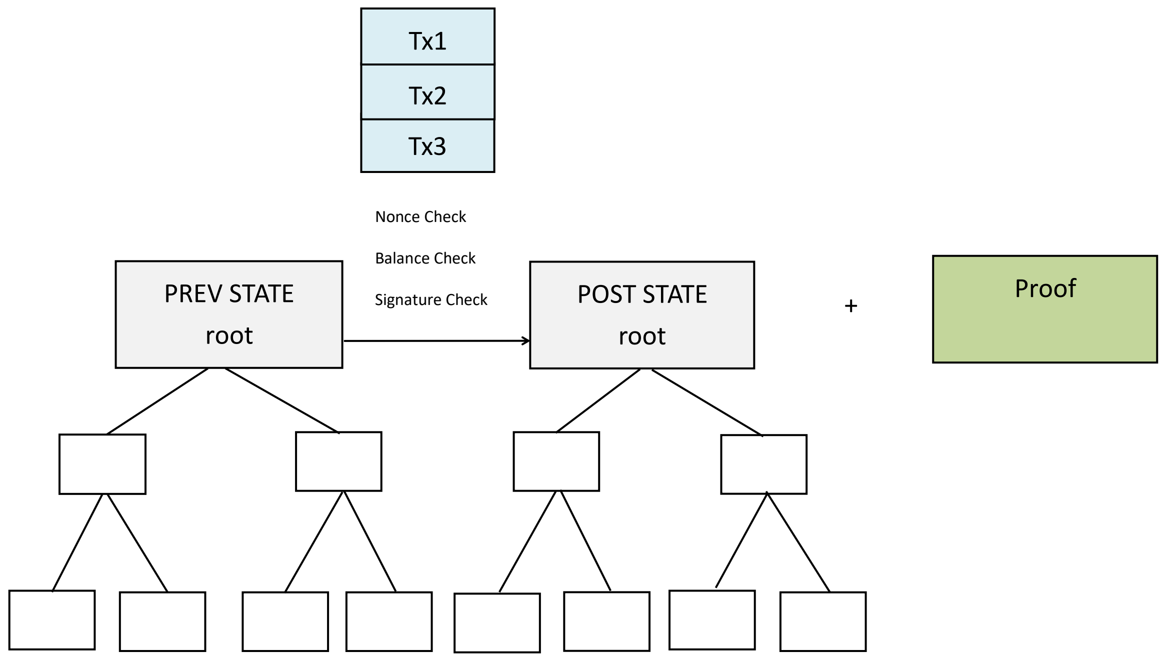
When the relyer receives the transaction, it must "execute" it. The execution of transaction is to change the state of the relevant account, and STF is a function to change the state of the account. STF is an abbreviation for state transition function.

State is for state machines, each state machine has a state at a certain moment. Assuming that the current state of the state machine is PREV_STATE, then there are n Actions T[1], T[2], ..., T[n] that act on the state machine in turn, and then the state of the state machine is POST_STATE, which can be expressed as:

$$\text{POST\_STATE} = \text{STF}\left(\text{PREV\_STATE}, T[1], T[2], T[3], \ldots, T[n],\right)$$

If the above Action is replaced by a transfer transaction, and the account collection in the system is regarded as a state machine, then the entire process is also the process of on-chain transaction execution. The execution of the transaction changes the global state of the entire chain. The global state on the chain is the state collection of each account. The states of all accounts form a Merkle tree. The leaf nodes of the tree are the account states, and the root of the tree can be directly used to represent the state collection. Therefore, the aforementioned PREV_STATE and POST_STATE are also the roots of the global account state tree.

The following figure shows this working process. The blue one represents the transaction sent by the transactor, and the grey one represents the merkle tree maintained by the relay. After the relay executes the transaction, the local merkle tree root will be converted from the prev state root to the post state root, and the green one in the figure represents the zero-knowledge proof generated by the relayer to prove that the account state transfer is effective.



Relayer submits the PREV STATE root, POST STATE root, transaction data and proof to the smart contract. After the contract verification proof is passed, the new state will be written to the chain. The contract does not need to separately verify the legitimacy of each transaction , It only needs to verify whether the proof is valid, which reduces the gas consumption on the chain, and the transaction data is stored in the cheaper location in CALLDATA. Every state transition outside the chain needs to provide a zero-knowledge proof, which is verified by the contract on the main chain, and the state can only be changed after the verification is passed. That is, every state transition strictly relies on cryptographic proof.

Of course, the relayer will not provide services to the transactor for free. After all, the relayer needs to consume gas to submit proof and data to the chain. Therefore, the transaction sent by the transactor to the relayer must also include the corresponding handling fee.

The size of the proof generated by ZK Rollup (small) and the verification time (basically constant soon) will not increase with the growth of the number of transactions, so ZK Rollup can greatly improve TPS. The only thing that affects the performance of the ZK Rollup chain is the cost of storing data on the chain of CALLDATA. With the upgrade of Ethereum, the cost of CALLDATA usage is reduced to 1/4 of the original, and the performance of ZK Rollup has been increased by 4 times, and the TPS can reach nearly 2000.

In the data on the chain, the PREV STATE root, POST STATE root and proof will basically not change with the growth of the transaction, only the transaction data will become larger with the growth of the transaction.In order to be able to accommodate more data in a blockchain, the transactions on the chain need to be compressed.

The simplest account status can include: public key, nonce and balance of the account. The leaf node has a unique position in the Merkle tree, so the index information of the position can indirectly refer to this account information. ZK Rollup uses the merkle tree to record the address, so that the account address can be expressed as the index value of the merkle tree, and the size of the address data is reduced from the original 20 bytes to 3 bytes.

In addition to compressing the account address, we can also compress the transfer amount data. For example, the amount of money on Ethereum is represented by a 256-bit large integer, but in actual use, very few large amounts and small amounts are rarely used. Therefore, if we assume that the smallest unit of transfer in the system is 0.001 ETH, and 4 bytes are used to represent the transfer amount, we can support transfers of 0.001 ~ 4,294,967.295 ETH, which is enough for a general system. If it is not enough, you can appropriately add more bytes to represent the amount, or introduce floating-point number representation.

The transaction fee is similar to the transfer amount. The actual transaction fee will fluctuate within a certain range, so we can also set a minimum unit for fee, for example: 0.001 ETH. Then use 1 byte to represent the fee of 0.001 ~ 0.255 ETH. The fee here is the transaction fee paid by the transactor to the relayer.

Similarly, we assume that the number of transfers of an account in a regular environment will not reach tens of thousands, so it is almost enough to use 2 bytes to represent the nonce of the account, because the range that 2 bytes can represent is 0~65535 .

Finally, the signature field cannot be compressed. Taking Ethereum as an example, the signature (r, s, v) requires a total of 65 bytes.

Therefore, the format of a transaction is as follows:

from	to	value	fee	nonce	signature
3B	3B	4B	1B	2B	65B

When implementing a specific system, we need to adjust the field length according to the actual situation to prevent field overflow, but in principle, it can be saved. Because the less transaction data, the more transactions can be accommodated on the premise of the same storage capacity. In fact, the transaction data submitted on the chain is the simplified information of the transaction, and does not include the nonce and signature parts. In this case, it is 11 bytes, and the data on the chain will become smaller.

In the ZK Rollup system, all user account information is managed by a Merkle tree. The root of the Merkle tree is recorded in the smart contract on the chain, and the value of this root also represents the current state of all accounts in the entire system. When a user initiates a transfer transaction, this state will change, but the change must be made in accordance with the rules. The work done by the relayer in this process includes:

1. First of all, must ensure the legitimacy of the transaction
2. Whether the transfer account has enough token to pay the transfer amount and fee
3. Is the nonce transferred out of the account correct?
4. Is the signature of the transfer transaction correct?
5. Next, the relayer executes the transfer transaction, modifies the leaf nodes of the transfer-out account and transfer-in account in the Merkle tree, and then recalculates the

root of the new Merkle tree.
6. Repeat the second step, the relayer will process multiple transactions at once in the sequence, and then submit the root of the finally calculated Merkle tree as the new state to the contract on the chain.

However, there is a problem with the above process: if only the root of the state tree is submitted to the contract, how can the user trust that the new state root is truthfully calculated based on the above logic. What if the relayer does evil and deliberately increases the transaction fee?
One way to solve this problem is to require the relay to submit the root of the state tree to the contract, and at the same time submit all transactions to the contract, so that anyone can verify whether the relay is cheating when calculating the new state tree based on these transactions. . But this is tantamount to moving all the data off the chain back to the chain, failing to achieve the purpose of layer 2 expansion.

Using zero-knowledge proofs can solve this problem well. ZK in ZK Rollup is also the abbreviation of Zero-Knowledge. After the relayer collects a series of transactions, it needs to use pre-defined ZK circuits to generate a Proof:

Make sure that the nonce, value, fee, etc. in each transaction T[1], T[2], …, T[n] are all correct and the signature is correct.Make sure that there is no problem in the state transition process, that is, STF(PREV_STATE, T[1], T[2], …, T[n]) = POST_STATE.Then submit this Proof together with POST_STATE, t[1], t[2], …, t[n] to the chain contract. Among them, t[1], t[2], …, t[n] are simplified information of transaction, excluding nonce and signature. So t[i] is smaller than T[i].

Then the smart contract only needs to verify whether the Proof is correct. If the Proof is correct and the state saved in the contract is equal to PREV_STATE, then the new state POST_STATE will be recorded in the contract, replacing the original state.

Since the relayer must generate a ZK-SNARK Proof before submitting it to the contract, if the relayer modifies the user's transaction maliciously, the Proof will not be verified.

In addition, since the transactions t[1], t[2], …, t[n] submitted to the chain do not contain nonce and signature, the data on the chain will become smaller (in the above example, each Transaction will only have 11 bytes on the chain).

At this time, the relayer can no longer modify the user's transaction due to the limitation of the proof. However, a malicious relayer can still refuse to serve a transactor and not collect the transactor's tranaction. In order to prevent this behavior, the contract must support on-chain withdrawal, that is, any transactor can withdraw its token from the chain.

The zero-knowledge proof used in this process is ZK-SNARK. In order to facilitate the understanding of its working process, the following first introduces the relevant knowledge of ZK SNARK through an example

Suppose that A needs to prove to B that he knows c1, c2, and c3, so that

$(c1 \cdot c2) \cdot (c1 + c3) = 7$

According to convention, c1, c2, and c3 need to be kept secret from B.

The first step is to "balance" the calculation, and draw the original calculation into such a "calculation gate circuit" through basic operators.

$$\mathrm{S1} = \mathrm{C1} \cdot \mathrm{C2}$$
$$\mathrm{S2} = \mathrm{C1} + \mathrm{C3}$$
$$\mathrm{S3} = \mathrm{S1} \cdot \mathrm{S2}$$

By adding intermediate variables, complex calculations are evened out, and the simplest gate circuit expression is used. The new gate circuit is equivalent to the original calculation.

The second step is to express each gate circuit as an equivalent vector dot product form, and this process becomes R1CS.

For each gate, we define a set of vectors (a, b, c) such that $s.a * s.b - \mathrm{s.c} = 0$. Where s represents all input vectors, which is $[C1, C2, C3, S1, S2, S3]$. In order to make the addition gate can be expressed in the same way, we add a dummy variable to become one, and the s vector becomes $[one, C1, C2, C3, S1, S2, S3]$.

Corresponding to the first gate, calculate $a = [0,1,0,0,0,0,0]b = [0,0,1,0,0,0,0]c = [0,0,0,0,1,0,0]$

Substitute s, a, b and c into

$s.a * s.\mathrm{b} - \mathrm{s.c} = 0$, get $\mathrm{C1} * \mathrm{C2} - \mathrm{S1} = 0$,

that is, this vector expression is the same as the first gate Are completely equivalent. In the same way we can calculate

The second gate $a = [1,0,0,0,0,0,0]b = [0,1,0,1,0,0,0]c = [0,0,0,0,0,1,0]$

The third gate $a = [0,0,0,0,1,0,0]b = [0,0,0,0,0,1,0]c = [0,0,0,0,0,0,1]$

The next step is the most important step. The vector expression is expressed as a polynomial, so that the verification of the vector is transformed into the verification of the polynomial. This process is called QAP (Quadratic Arithmetic Programs).

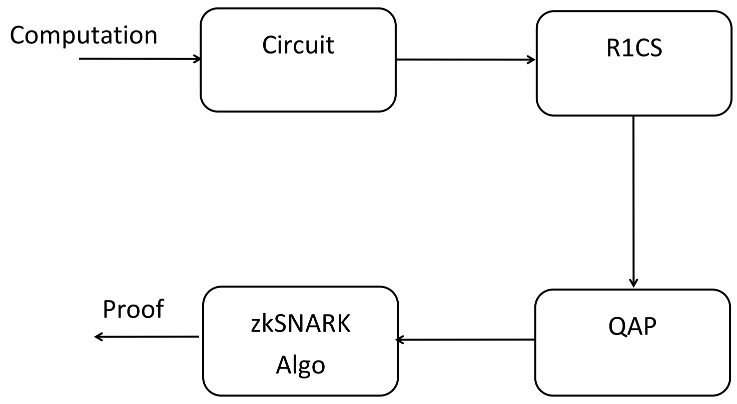
We choose 1, 2, 3 and find a set of polynomials

$$\begin{aligned}
a &= [P_{a1}(x), P_{a2}(x), P_{a3}(x), P_{a4}(x), P_{a5}(x), P_{a6}(x), P_{a7}(x)] \\
b &= [P_{b1}(x), P_{b2}(x), P_{b3}(x), P_{b4}(x), P_{b5}(x), P_{b6}(x), P_{b7}(x)] \\
c &= [P_{c1}(x), P_{c2}(x), P_{c3}(x), P_{c4}(x), P_{c5}(x), P_{c6}(x), P_{c7}(x)]
\end{aligned}$$

So that the values of the a, b, and c arrays correspond to the vectors of the aforementioned three gate circuits when the polynomial takes values x =1, 2, and 3.

Take the polynomial $P(x) = s.a(x) * s.b(x) - s.c(x)$, according to the original definition, when x takes the value 1, 2 or 3, P(x) =0. According to polynomial characteristics, P(a)=0 is equivalent to P can be divisible by (x-a), P(x) must be divisible by (x-1)(x-2)(x-3), which means that there is H (x), let $P(x) = T(x) * H(x)$, where $T(x) = (x-1)(x-2)(x-3)$. Note that the process of QAP transforms the values of the original three points into a polynomial, which is equivalent to inserting a lot of meaningless values in between. The values of these values are irrelevant to the original formula. That is to say, the verification of the polynomial is not equivalent to the verification of the original calculation, but the verification of the polynomial also verifies the original calculation. Finally, we transform the proof of the original formula into a polynomial proof. As long as we prove that $P(x) = T(x) * H(x)$, the original formula can be verified.

For the ZK Rollup system, the entire certification and verification process is



1. Each specific transaction problem can be abstracted into a mathematical expression. ZK Rollup packs multiple transactions to generate a mathematical expression, and then expresses it evenly into an arithmetic circuit, that is, a generated circuit expression.
2. After having this circuit expression, construct a constraint system, the most commonly used is R1CS (rank-1 constraint system, first-order constraint system)
3. After having this set of constraint systems, this problem is transformed into a problem that can be satisfied by polynomials, called QAP problem.
4. Then after the QAP problem is obtained, the ZK-SNARK algorithm system is used to give the final proof of generation.
verification:
1. The smart contract verifies the correctness of Proof according to the constructed ZK-SNARK algorithm.
2. If the Proof is correct and the state saved in the contract is equal to PREV_STATE, then the new state POST_STATE will be recorded in the contract, replacing the original state.

NFT Community Structure

Vswap is building sub-communities based on V-wallet for NFT auction system.

Digital artists join in the Vswap sub-community and they are the creators of NFT.

The sub-community is the sponsor merchant and guarantor of NFT assets in the Vswap community.

The wallet and the smart contract on the chain will form a NFT auction platform. On this platform, NFT assets will be settled through on-chain transactions, and the corresponding digital artwork information will be presented on the Vswap encrypted wallet page.

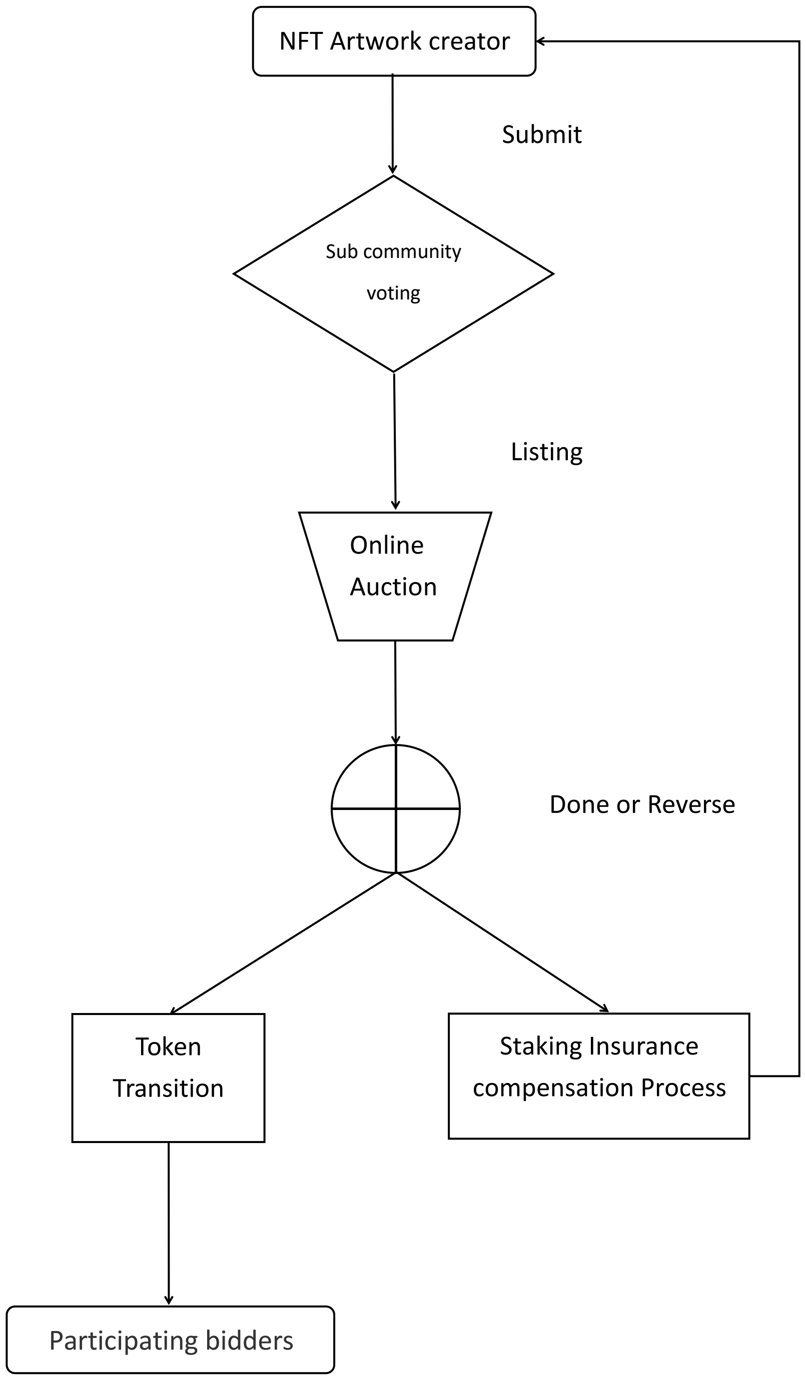
Crypto artists join the sub-community to create NFT assets on behalf of the sub-community. NFT assets will first be voted and sorted within the sub-community. The top-ranked works will be submitted to the auction market by the sub-community. Each auction submission quota of the sub-community will be allocated according to the amount of deposit staked by the sub-community for the NFT Auction deposit on the chain.

The purpose of the deposit staking:

1. The volume of deposit is linked to the number of NFT works submitted for each auction of the sub-community.
2. In order to prevent fraud of NFT works, the sub-community needs to use warranty pledge to provide insurance, to build up auction reserve price repurchase valuation guarantee for the artworks to be on listing.
3. In the transaction process, use the warranty to make priority settlement and payment for creators.
4. After the transaction process fails, the auction tokens already paid by the bidders will be paid back by the deposit first.

In order to expand the development of crypto NFT artwork market, the Vswap system provides a reward mechanism for deposit mining. The deposit pledger can stake VAL to the smart contract of the sub-community. The staking behavior provides an insurance mechanism for NFT auctions. The funds need to bear the compensation for losses in the NFT trading process, but they will also be rewarded by the system because of the interest of staking.

The NFT transaction process is as follows


NFT Artwork creator
Submit
Sub community voting
Listing
Online Auction
Done or Reverse
Token Transition
Staking Insurance compensation Process
Participating bidders

Development Plan and Roadmap

Jan. 2021 V-wallet online,open the registration

Feb. 2021 V-wallet dock in ETH,TRX

March 2021 Vswap ETH DEX online

April 2021 Vswap BSC DEX online

May 2021 Vswap ETH Layer 1 aggregator beta online

May~June 2021 Vswap VAL AMM LPs start mining

June 2021 Yellow Paper of Valkyrie Swap release regarding further plan and roadmap

June 2021 Vswap ETH Layer 2 structure Beta release

June 2021 Vswap NFT market online

July 2021 NFT Auction start

Before October 2021 Vswap Ethereum L2 ZK Rollup solution release

2021 Middle term research and design tasks:

Defi Loan

Futures Function

VSwap IDO Platform

Decentralize Wallet private Key management

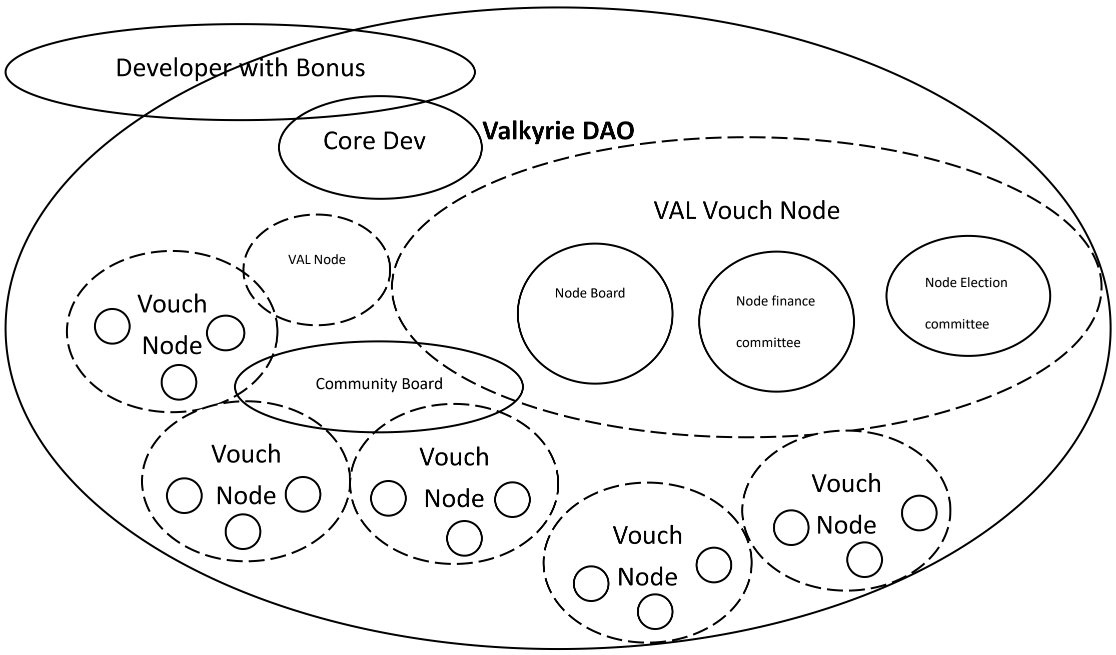
V-Dapp Store

Community NFT Auction Platform

Mandalorian Plan

The Valkyrie Swap project has attracted developers from all over the world. These Defi technology enthusiasts will develop and promote smart contracts within the VSwap ecosystem. Valkyrie Swap provides a rating system and a reward system for the developer community, ensuring that every line of selected code is rewarded with tokens. All reward tokens will be distributed globally through the developer community and website. The Mandalorian project is open source, and every developer can contribute from around the world and participate in evaluation tasks. This new form of collaboration will provide the developer community with the greatest freedom and the greatest reward.

The Governance Structure



Core Developer:Developers who write the smart contracts on the blockchain and the Valkyrie Swap architecture. They have the right to propose.

Dev Community and Bounty Devs : Bounty project developers under the Community Board program

Token Holder: Any holder of a VAL token

VAL Node: Any token holder can initiate a node of his own as the founder

VAL Vouch Node: Vouchers with governance power, 51 nodes of the whole network selected by voting every year.

Node Board: Governance implementer within the node community

Node Finance Committee: The allocation committee of node revenue, node development, node function and other interests.

Node Election Committee: A committee that carries out credible elections at nodes and receives free aid funds from donors to serve the node in obtaining the election.

Community Board is a voting committee composed of 51 trusted nodes, which votes on the proposals, governance, awards, etc. The voting weight of 51 trusted nodes in the Community Board is the number of votes received/the total number of votes cast in that year's election. The principle of Community Board proposal adoption is to obtain the support of 26 or more trusted nodes with more than 50% votes.

Node is a community organization voluntarily established by all members of V community. Each address holding VAL token can declare the establishment of autonomous node. Every year’s Up Helly Aa Holiday, the last Tuesday of January is the online voting day. At 23:59 on Monday, the subsystem takes a snapshot of the VAL block chain address. The snapshot evidence obtained determines the number of valid votes cast for each address in that year's node election. The next day community members will elect 51 global nodes from the list of valid node candidates, which will exercise governance, proposal, and distribution rights on behalf of all VAL community members in the New Year. The voting weight of these 51 nodes in the Community Board will be determined according to the distribution of votes in this election.

Each node community can publish an annual plan, vision, benefit distribution advocacy and other announcements that are appropriate for the development and governance of the community so that the community can know and recommend better implementation programs and people.

Rights and Governance of Token

1. The right to propose and vote

Any address on the blockchain holding a VAL token has the community voting right.

2. Node application

Each Token holding address can initiate an application for establishing a node through the application page. After the successful application of a node, the node applicant and all Token holding addresses in the node can obtain the node NFT eco benefit and other node rights.

3. Transaction fee rights

Swap Protocol will automatically initiate a VAL repurchase plan based on the surplus transaction fee based on the Community Board proposal.

4. Liquidity mining interests

Holding VAL Token will be able to participate in the mining of VSwap Defi liquidity pool and thus obtain profits.

5. Developer community rights

The developer community will need to cooperate with sub-community who stakes VAL for deposit to get permission to list the dapp and start own business.

6. Trusted node mortgage interest

Vouch Node needs to mortgage a certain amount of VAL to prove the credibility of its Node proposal and the validity of the election, thus obtaining the actual benefit of the mortgage.

7. The deposit for the on-line reward project

When the third party dapp listing the Valkyrie wallet, the development team will need to stake VAL.

8.Listing project IDO deposit

VAL is the margin pledge voucher for the crowdfunding project in Vswap IDO channel.

9. Functional deployment rights

Node communities need to pledge or pay VAL to grant the right to use public products in the community channel. The concept of public products includes but is not limited to: third-party applications, auxiliary functions, payment tools, statistical tools, transaction tools, etc., to meet the conditions set by the reward project.

10. NFT souvenirs

All NFT souvenirs and artworks from The V community can be exchanged with VAL. The auction of NFT artworks require VAL deposit from initiator.

Token Allocation Plan

Early User Awards 1,200,000

Global Community Activity awards 1,500,000

Special reward for the first 300,000 users 4,200,000

Angel Round Offer 610,000

IDO round 3,268,000

Community NFT market Insurance Pool mining 4,678,000

Swap AMM LP and L2 Mining Pool,Staking Pool 10,180,000

Mandalorian Developer Funding 2,700,000

Core Developers 3,542,000

Consultant 322,000

Total: 32,200,000 VAL

Formula of special reward for the first 300,000 users

$$P_{Initial} = P_{firstweek}\left(1 + \sum_{1}^{week} 0.95^{week}\right)$$

Customer bonus Max is up to 300,000 ETH addresses,FCFS basis.

The token shares of Core Developers will be locked on public address for global supervision. After 24 months of the launch of Vswap, the locked shares will be released on a monthly linear basis.