Decentralized Portfolio Derivatives Exchange V1 Based on Ethereum layer2 Arbitrum rollup

OPSWAP.IO

Opswap is a decentralized portfolio derivatives trading platform developed based on Ethereum layer2. It is the first to use Ethereum layer2 expansion technology Arbitrum Rollup to develop the spot digital currency trading based on AMM automatic market making technology, and the first original perpetual contract (no gambling) based on spot margin trading. It can support up to 10 times leverage.

1. Development Background

Since June 2020, DeFi on Ethereum has been hot. The old loan agreement represented by Compound has launched its own tokens in the form of liquidity mining on Uniswap, completely triggering liquidity mining in the DeFi field, resulting in abnormal congestion on the network on Ethereum, with dozens of dollars of GAS fees completely shutting out small amount players. Each transaction and each operation need to wait for confirmation of at least one block, and the real-time performance is poor. It has exposed that the expansion problem of Ethereum is imminent, and it is also a bottleneck in the development of Ethereum at this stage.

Vitalik, founder of Ethereum, has proposed that the whole Ethereum ecosystem needs to focus its power on the side chain of rollup to cope with the expansion demand in the medium and even long term before the launch of 2.0. At present, the layer2 technology, which consists of ZKrollup (zero knowledge proof) in the payment direction and optimisticrollup (fraud proof) in the complex intelligent contract direction, has become the mainstream. It can be predicted that 2021 will be a year when a large number of rollup projects will break out.

For Rollup, it improves the on-chain efficiency by combining on-chain and off-chain, putting the simple merkle tree on the chain, and putting the complex operation process off the chain. Because the current practice shows that Layer2 will be more than 100 times faster in TPS than Layerl. Opswap build DEX based on Arbitrumrollup (also called complex OptimisticRollup) technology, which makes Opswap have obvious advantages, with almost GAS fee free, instant confirmation of transaction and 0 slippage in leverage contract transaction. It has comprehensively surpassed the centralized exchange in terms of performance and security.

2. Differences between Arbitrumrollup technology on and other Rollup solutions which Opswap is based

ArbitrumRollup is an optimistic rollup expansion technology based on Ethereum developed by the offchainlabs team, and there is no authoritative classification for the current block chain technology, so we can deem that Rollup is essentially a Layer2. On the whole, Rollup is characterized by the calculation off the chain +the data on the chain and verification in the form of Fraud proof, that is, a small number of data roots are put on the chain, and the process of calculating the data is conducted off the chain. Many calculations are very complicated, which will reduce the efficiency of the block chain system.

At the same time, Rollup doesn't have all the data on the chain. Its on-chain data is limited to the input for each transaction, but does not include its final state. It's like we're going from one place to another, and the data on the chain shows only the route, not including what we did at both places. For Rollup, fraud proof will e performed every time the system completes a transaction and other actions.

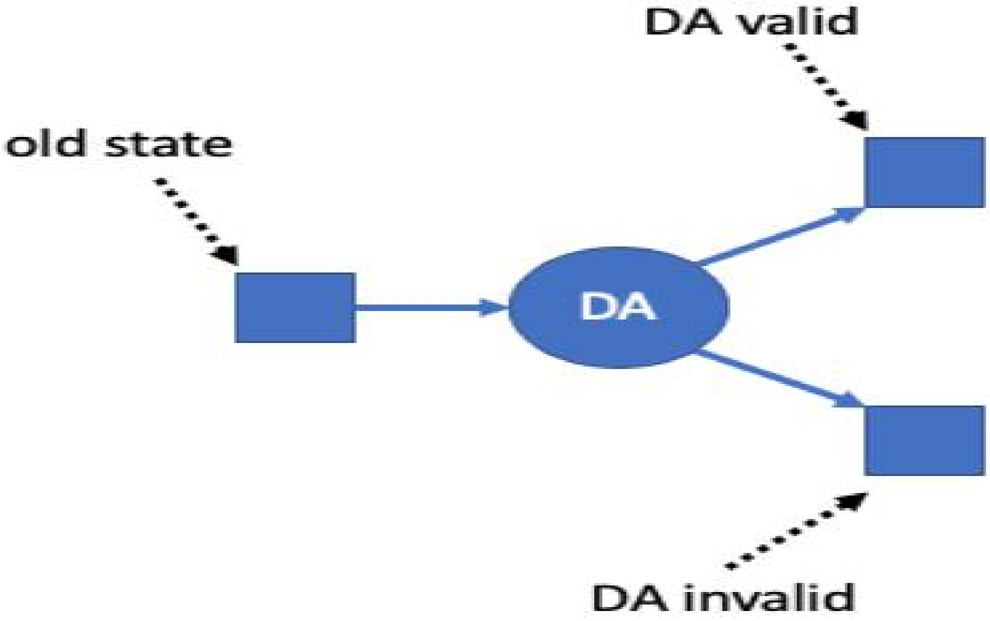
The current mainstream optimistic rollup is called one-round interactive Rollup, while Arbitrum rollup is called multi-round interactive optimistic rollup. Arbitrum rollup performs more round of fraud proof than other types of optimistic rollup. One-round optimistic rollup needs to simulate a complete call on the chain, i.e. frequent call contract, when resolving fraud disputes, which is costly. If an attacker attacks a transaction regardless of cost, the attacker will eventually fail, but it will affect the normal operation of other contracts on the project.

Arbitrum rollp is not limited to this, it will narrow the scope of the dispute to a single, without affecting the execution of other normal contracts, and can resolve the dispute on the chain at a lower cost. At the same time, Arbitrum rollp's multi-round fraud proof also makes the protocol deployed on it more secure.

3.Working Principle of Arbitrum rollup Technology

Let’s start with the basics. The state of your VM is organized as a Merkle Tree, so a cryptographic hash of the VM’s state can be computed. At any point in the protocol, there is some state of the VM that is fully confirmed and final. Its hash is stored on-chain.

A participant in the protocol can make a Disputable Assertion (DA) which claims that starting in some state-hash, and with some technical preconditions, the VM can execute a specified number of steps of computation, resulting in a specified new state-hash, and with the VM making specified payments and emitting specified log events during that computation. The DA might be valid (i.e., truthful) or invalid. The party making the DA will be required to stake a deposit on the validity of the DA. (More on stakes and how they work below.)



A Disputable Assertion creates a decision point for the protocol

As depicted on the left, a Disputable Assertion creates a logical decision point that the protocol will eventually have to resolve. If the DA is valid, the system will enter a new state on the upper right, with a new state hash, and with the side-effects (payments and logs) specified in the DA. Or on the other branch the DA is invalid; it is rejected and the state is unchanged.

3.1 The old Arbitrum protocol

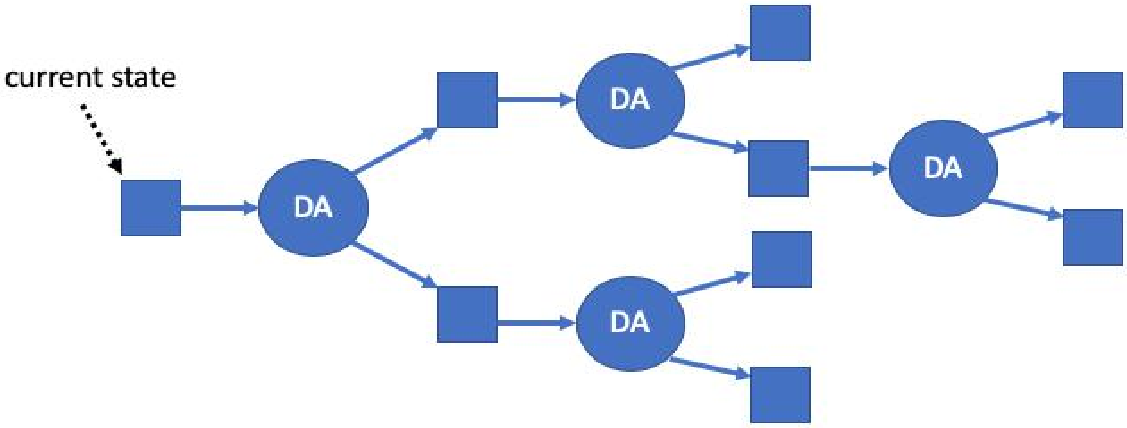
The original Arbitrum protocol dealt with Disputable Assertions one at a time. A DA would be asserted by some party, then a challenge period would elapse, during which anybody could challenge the DA. If there was no challenge, the DA would be confirmed; otherwise the dispute protocol would be run and the DA would be canceled (to be safe in case the asserter and challenger were colluding to “cook” the dispute result).

This was simple but had two drawbacks. First, because only one DA could be active at a time, the rate of progress of a VM would be limited. Essentially, progress had to stop during each challenge period. Second, a malicious party could freeze a VM by deliberately challenging all DAs made for that VM. This would cost the attacker a series of stakes, but if they were willing to pay this cost they could hold up progress for a long time, in at least some scenarios.

3.2 New and improved

The new Arbitrum Rollup protocol, which I'm describing in this post, addresses both of those drawbacks. Multiple DAs can be "pipelined" so that a VM can progress just as fast as validating nodes can emulate the VM's computations. Second, as we'll see below, a bad actor can't slow up progress, they can only temporarily delay on-chain recognition of outcomes that are already "trustlessly final" for honest parties,How does that work? Let's dig into the new protocol….

Each state can have at most one DA following from it. If a DA has no following state, then anybody can create a DA that follows it, creating a new branch point. The result will be a tree of possible futures.



A tree of possible futures

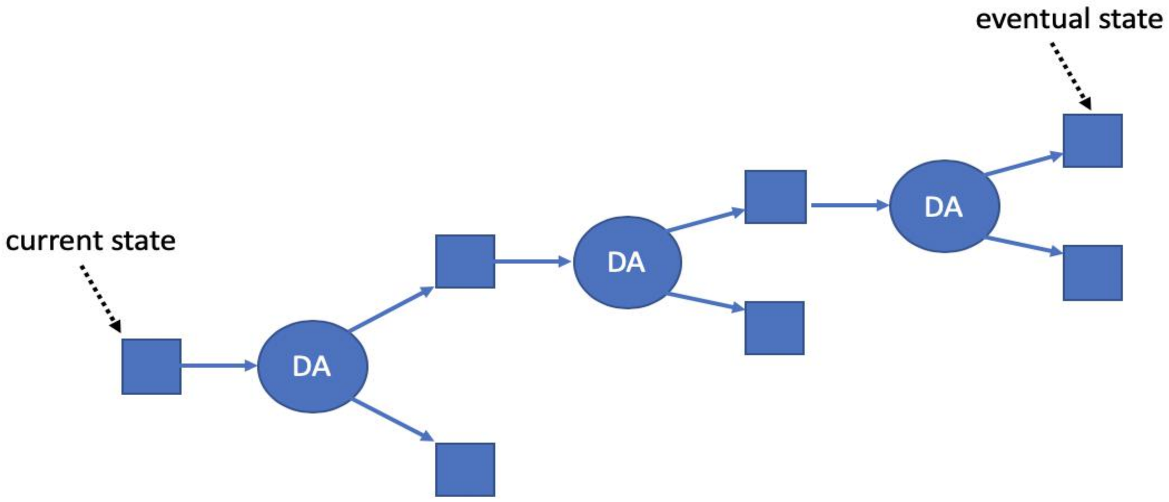
4.Staking

Another important part of the protocol is staking. Anybody can put a stake on one of the square boxes in the tree. By staking on a square you are asserting that that square will eventually be confirmed by the protocol. In other words, you're asserting that you have taken the correct branch at each DA along the path from the current state to the square you're staked on. If you are wrong, you can expect to lose your stake.

Staking actions cannot be undone. You can move your stake to the right — choosing to go up or down at each branch point — but you can't move to the left because that would amount to undoing a staking commitment that you made earlier.

The party who makes a Disputable Assertion is required to stake on the "DA valid" successor of that DA. Normally they'll be able to satisfy this requirement by moving an existing stake to the right to put it onto the required successor square. (In the rare case where they can't do that, they can put down an extra stake on the required square. But note that they would then be staked on two inconsistent paths, so that they would eventually have to lose at least one of the two stakes — it's not a smart move to contradict yourself.)

One more detail about stakes: if the square you are staked on is confirmed and becomes the accepted history, you have the option of recovering your stake. That means that if you are correct, you can keep your stake where it is and wait for the system to "catch up" with you, and then you'll be able to recover your stake.



A more typical state tree — a sequence of truthful assertions

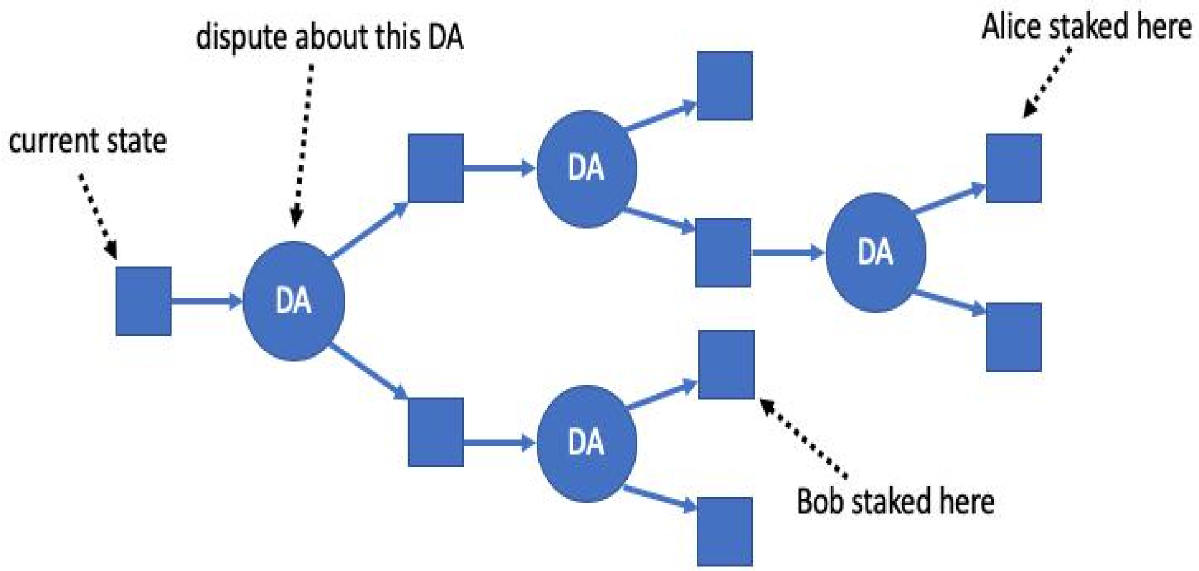
At this point you might be worried that the tree of possibilities can get very large and branch-y. That's not likely to happen in practice, because it would require multiple parties to stake on outcomes that are mutually inconsistent. Only one of them can be correct, and everybody else will lose their stake. More likely is that the "tree" is really a chain of valid DAs, one after another, and all of the stakes are on the same outcomes.

4.1 Staking Deadlines

We need the system to make a decision about each Disputable Assertion before too much time passes. So when a DA is added to the chain, creating a branch point, a deadline is associated with that DA. The deadline is far enough in the future that everyone will have time to check whether the DA is valid and get a transaction on-chain to stake on an outcome of the DA, if they choose to do so. If anyone wants to commit themselves to a stake for or against the validity of that DA, they must do so before the deadline. (Stakes can still be introduced after the deadline, but they do not participate in deciding for or against that DA.) Once the deadline has been reached, all of the stakes relevant to deciding that DA will be known.

4.2 Disputes

If Alice and Bob are staked on different squares, one of two things will be true. Either there will be a rightward-moving path from one of them to the other — meaning their claims are consistent with each other — or there will not be such a path. If there is not a rightward-moving path connecting Alice and Bob's squares, then they must disagree about something. There will always be a unique point of dispute between them — a unique DA for which one of them is staked on that DA being valid and the other is staked on it being invalid.



Alice and Bob are set up for a dispute

Whenever there is a dispute between two parties, the system can start an interactive dispute resolution protocol between them. I don't have space to describe the dispute resolution protocol here — suffice it to say that it is a bisection-type interactive protocol similar to what we have described in other Arbitrum documents.

The result of the dispute resolution protocol is that one party will be found to be incorrect. That party will forfeit their stake. The stake will be erased from the square it is on. Part of it will be given to the other party in the dispute, and the rest will be burned.

Multiple disputes can be going on at the same time, but each staker can be in at most one dispute at a time.

Because losers' stakes will be erased, each dispute will reduce the amount of disagreement in the system. Parties who lose their stakes can re-stake if they want, but the new stakes won't be able to affect DAs whose staking deadlines have already passed. The effect of this is that after a DA's staking deadline has passed, disputes will progressively eliminate any disagreement about how to treat that DA.

4.3 Confirming Results

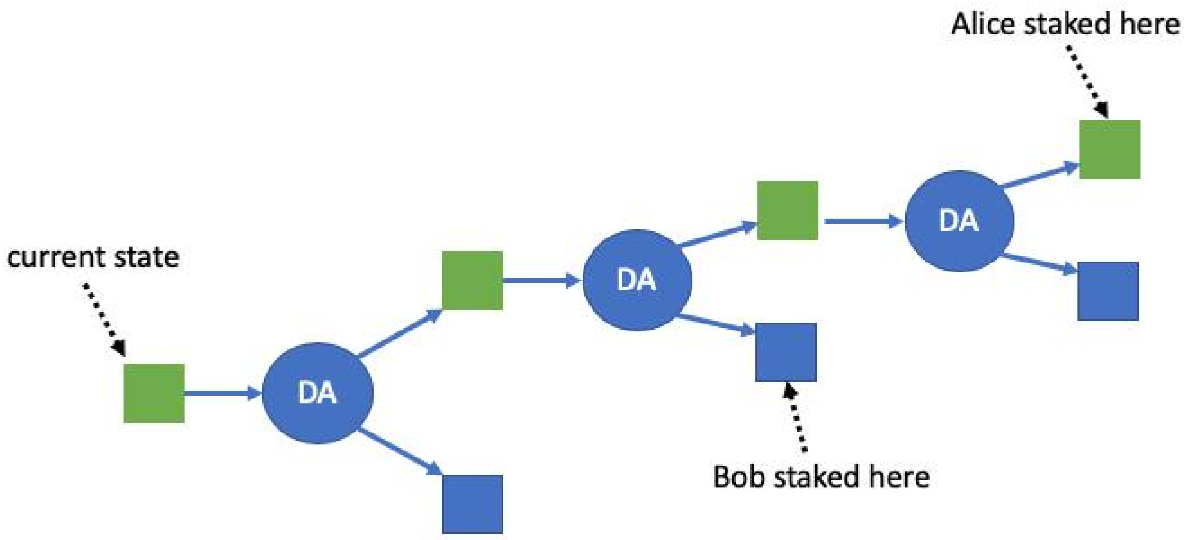
Once a DA's staking deadline has passed, and all of the timely (placed before the staking deadline) stakes that remain are on the same branch from that DA, the system can confirm the result of that DA. The DA is either accepted or rejected, and the current state moves to the appropriate square to the right of the DA. If the DA is confirmed as valid, its side-effects, such as payments, are effectuated on-chain. This is how the state of the VM moves forward.

In the common case, parties will behave honestly, because they won't want to lose their stakes by staking on false claims. Only valid DAs will be asserted, in a single chain, and nobody will stake on the invalid branch of any DA. In this case, every DA can be confirmed immediately when its staking deadline expires.

5.Why It's Trustless

An important property of Arbitrum Rollup is that it's trustless — a single honest party can force the VM to behave correctly and make progress. To see why, imagine that Alice always stakes on the truthful branch at each DA, and she asserts DAs if the tree ever gets empty.

Because Alice is staked on true branches, she will win every dispute she gets into. If anybody else disagrees with Alice, they will either (a) lose their stake in an unrelated dispute with a third party, or (b) eventually get into a dispute with Alice and lose their stake to her. Either way, everyone who disagrees with Alice will eventually lose their stake. Only stakes that agree with Alice will survive, so Alice's path through the tree will eventually be the only one that has timely stakes on it — and Alice's path will be confirmed.



If Alice is honest, the green squares will eventually be confirmed, no matter what anyone else does

Because the system is trustless in this way, if Alice is staked on a square and she knows the path to that square is truthful, Alice can be certain that the square she is on will eventually be confirmed. As far as Alice is concerned, that path is as good as final.

Even if you're not staked on a path, if you see that several people are staked on it, and you trust that at least one of those people is honest, then you can be sure that that path will be confirmed eventually — as far as you are concerned, that path is as good as final.

5.1Benefits of trustless finality

Why is it valuable to have trustless finality for outcomes? The classic example comes from previous discussions of other rollup protocols. Suppose a VM is going to make a payment to Alice. The payment event is on the honest path, but it's going to be a while before the square where the payment happens will be confirmed on-chain.

Trustless finality gives Alice a way to get her money right away. If Bob has unencumbered money, he can give it to Alice immediately, in exchange for Alice assigning the future not-yet-confirmed payment to Bob (plus paying Bob a minimal fee). Bob will only want to do that if he can be sure that the payment will actually happen. Bob can make sure of that by staking on the honest outcome — then he will have trustless confidence that the payment will eventually happen. It's not only Bob who can do that. Anybody who has funds can lend them to Alice and others like her, in the same way. Those people can compete with each other by offering lower fees, driving down Alice's cost to get her funds right away.

The key point is that the viability of this kind of market mechanism depends on trustless finality. The delay in on-chain confirmation of a thing is less of an inconvenience if "everybody" already knows that that thing will eventually be confirmed.

This is true not only for payments but for other things a VM does. If the VM is going to emit a log item announcing that something has happened, trustless finality means that anyone can act with certainty that the log item will be recognized on-chain.

5.2Delay attacks

Because the system is trustless, bad actors can't force an incorrect outcome. What they can try to do is slow down progress. Doing this requires them to sacrifice stakes, which will be costly if the stake amount is significant.

Let's imagine that somebody is motivated to launch delay attacks, and they're willing to sacrifice stakes. What is the worst damage they can do?

The first thing to note is that bad actors can't prevent honest parties from continuing to build out the honest branch of the tree. And they can't prevent honest parties from gaining trustless confidence in the eventual confirmation of the honest branch.

All that an attacker can do is to stake on false branches to delay on-chain confirmation of the honest path. Each stake they place will create one more dispute against an honest party, in which the honest party takes a big chunk of the attacker's stake. Once all of the attacker's stakes have been taken, on-chain progress will continue.

What if the attacker places multiple stakes on false outcomes? Then those stakes will have to be taken one by one in disputes. If there are multiple people staked on the honest outcome, those people can all enter disputes against the attackers, working in parallel to take the attacker stakes. And notice that it will be obvious to everyone what is happening, and lots of people will want to get in on the action, staking on the true outcome so they can join the feeding frenzy of people using disputes to grab attacker stakes. If there are K people staking on the honest side, it will cost the attacker K stakes to buy one dispute period of delay. If the attacker puts down even more stakes, that will likely attract even more honest stakers. That’s a bad dynamic for the attacker.

6. Difference between DEX and CEX

6.1 In centralized exchange, the exchange controls your assets

From the moment you transfer your asset to a centralized exchange, the actual control right of the asset has been changed. All data on your account is just the data presented by the trading platform. In fact, your personal digital assets are deposited in the cold and hot wallet of the trading platform.

Since the address private key assigned by the platform also belongs to you, the currency in your platform account is always yours, unless you give the private key to someone else. Obviously, the asset security levels involved in the two approaches are very different.

In the centralized trading platform, all trading behaviors depend on the platform as the central organization. You have to apply for the platform approval to withdraw your currency, and you have to complete the transaction through the platform. In a word, the centralized exchange has absolute discourse and control rights.

6.2 At DEX, smart contract solve trust issues

DEX's transactions are guaranteed by smart contract, which is a program that can not be tampered with and can not be stopped, and will be triggered automatically as long as objective conditions are met. "code is law", a dogma that programmers believe in, is also the key to blockchain's technical solution to trust issues.

7. Benefits and Trading Logic of Opswap

Opswap thoroughly solves the disadvantages of centralized exchange for a long time: malicious acts such as platform escape, cash withdrawal restriction, data control, artificial intervention in futures trading, disconnection when price fluctuates violently, stuck and the like, and ensures the security of users' assets to the greatest extent. Opswap provides users with better privacy, greater transparency in asset operations, and the auditability of regulatory guarantees.

Different from centralized exchange, Opswap does not control users' assets. On the contrast, assets is often deposited in a decentralized manner by digital wallets and smart contract. Thus, no entity acts as the owner of all cryptocurrency, and then the risk of loss is much lower.

7.1 Opswap eliminates intermediary costs and takes you into the era of cryptography economy

Since asymmetric cryptography is conducted by "private key for signature, public key for verification", it breaks the dependence of symmetric cryptography on cryptography verification institutions (such as banks, websites, etc.), and it is also the core of the decentralized advantage of cryptographic consensus.

Cryptographic consensus can eliminate the third-party trust intermediary, which is the direct reason that why Satoshi Nakamoto created Bitcoin and the logical starting point of the Bitcoin White Paper.

DEX eliminates the cost of running centralized exchanges. Of course, there are more operating costs of the centralized economy that can be eliminated, and the key to opening the cryptography economy is controlled by us.

7.2 Opswap spot trading logic

Opswap takes AMM automatic market maker technology as the basis, and the important core of Opswap adopts the classic constant product (xy=k) pricing formula, which has been fully verified in Uniswap and other projects.

Swap mode:

Without counterparties, and the order will immediately execute the transaction directly with the market price by setting a appropriate slippage.

In essence, it is equivalent to combining the trading advantages of CEX with that of DEX, and the resulting ignition point will be a landmark milestone in the field of digital currency.

7.3 Trading logic of Opswap perpetual contract

The first spot perpetual contract based on AMM market-making algorithm supports up to 10 times leverage. A margin exchange function has been added to the perpetual contract. For example, user A deposits 1 ETH margin in the perpetual contract and selects more than 10 times, then the contract will use liquidity to exchange 100 ETH spots for it in advance and deposit them in the user's margin. When ETH rises, user A's profit will be liquidated and 100 ETH will be sold at the current price. The price difference will be the profit obtained by user A. When ETH falls, user A sells ETH, then the corresponding margin will be lost. All transactions are based on spot exchange, fundamentally solving the problem of data controlling in centralized exchange.

8. Security of Opswap

Security has always been the top priority for Opswap. Due to the relatively complex business of decentralized contracts, the security challenges are relatively higher. The key security enhancements to Opswap spot and contract in code security, margin mechanism, risk control and other aspects have been made.

As a DeFi product based on smart contract, the security of smart contract code is the key to the security of the whole product. When developing smart contract, security has always been an important part of the products, giving priority to mature and robust technologies to develop our products. The code for the Opswap contract has been fully tested and a 100% test coverage for all possible branches of code that could be run has been achieved.

On the other hand, Opswap works with the industry's best security teams to commission them to conduct comprehensive security audits of Opswap's smart contract, and future upgrades and new products of Opswap will continue to commission the industry's best security teams to conduct code audit.

Email：contact@opswap.io

Discord：https://discord.com/invite/rTWTZrRJKW

Twitter：https://twitter.com/opswapio

Telegram：https://t.me/opsglobalOfficial

Gitbook：https://app.gitbook.com/@opswap123/s/opswap/