ZKSWap V2 Whitepaper

1. Introduction

Decentralized finance, also known as DeFi, has been growing rapidly, with blockchain developers creating a range of applications based on Ethereum network and major public chains, including exchanges, collateral lending, stablecoins, insurance, oracles, games, etc forming an increasingly holistic decentralized financial ecosystem.

On-chain assets have been growing rapidly and kept setting new records since 2019, with the size of assets locked in various DeFi protocols surpassing $100 billion by May 2021. As more blockchain applications are developed and implemented, new users continue to become familiar and engaged, it is bound to further drive the development of DeFi and the continued activity and growth of on-chain assets.

Public-chains like Ethereum are currently experiencing congestion, creating sharp increases in transaction fees during user and transaction surge. For example, Ethereum based trading exchange Uniswap has frequently charged over a hundred dollars transaction fee for a single transaction at peak times, which can be too much for an average user.

Congestion and surge in transaction fees due to low TPS and high users volume are the sore spots where ZKSwap V1 came into play. Launched in February 2021, the platform gives the option for users to transfer all ERC20 tokens to Layer2 through Zk-Rollups technology, which guarantees the consistency of Layer1 and Layer2 states based on zero-knowledge proofs. This allows all exchanges to take place on Layer2, with zero Gas costs for real-time exchanges (no more waiting for a block confirmation time) and unlimited scalability, free from Ethernet TPS and block confirmation times, giving DEX a seamless CEX like (Centralised Exchange) experience while keeping asset security in real time.

Building on ZKSwap V1, the ZKSwap V2 has the following new features:

1) "Unlimited Token Listing – users can add any Token by themselves and create trading pairs with a set fee;

2) Optimized implementation of circuit branching to improve circuit efficiency – supports one account and two balance modifications;

3) Optimize the withdrawal process – In V1, the withdrawal operation is bundled with the block verification operation. The number of withdrawals in the block is limited due to the per gas fee limit.

The ZKSwap V2 can reduce the withdrawal time by increasing the circuit efficiency and the team will promote the "Layer2 for All" multi-chain ecosystem strategy by deploying the protocol on BSC, HECO and OKEx Chain.

2. Architecture Optimisation

2.1. Token Management

ZKSwap V2 will support three types of Token: Fee Token, User Token and LP Token, supporting 2^16 Token are supported in total.

Type	Number	Comments
Fee Token	0-31	0 – ETH (32 in total)
User Token	31-16383	16352 tokens
LP Token	16384 – 65535	49152 tokens

2.1.1 Fee Token

Fee Token can only be added by the Governor, mainly for ETH, ZKS, USDT, WBTC.

2.1.2 User Token

User Token can be added by users with a flat fee.

2.1.3 LP Token

LP Token can be added automatically when a user creates a pair. The governor can set a limit to the number of pairs that can be created.

2.1.4 Governor Config

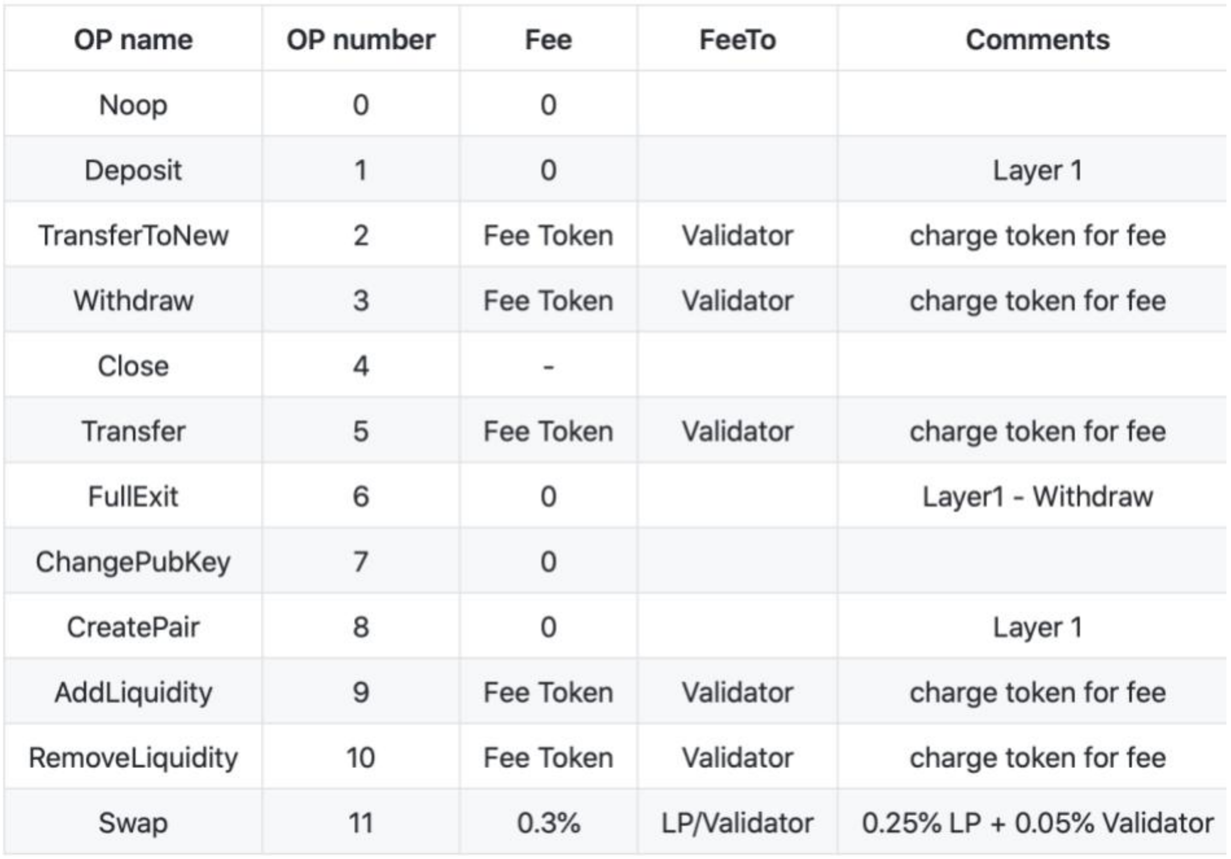
The Governor can set the supporting fees for adding User Tokens and modifying the creation of pairs.

2.2. Account Management

The ZKSwap V2 version will support 2^28 accounts. Account 0 is the Validator account.

2.3. Fee Model

For Layer2 transactions on ZKSwap V2, users can designate one of the Fee Tokens as the payment fee type. The user-designated Fee Token can be used to offset the fee after the free limit for transactions, liquidity adding and removal is exceeded.



OP name	OP number	Fee	FeeTo	Comments
Noop	0	0		
Deposit	1	0		Layer 1
TransferToNew	2	Fee Token	Validator	charge token for fee
Withdraw	3	Fee Token	Validator	charge token for fee
Close	4	-		
Transfer	5	Fee Token	Validator	charge token for fee
FullExit	6	0		Layer1 - Withdraw
ChangePubKey	7	0		
CreatePair	8	0		Layer 1
AddLiquidity	9	Fee Token	Validator	charge token for fee
RemoveLiquidity	10	Fee Token	Validator	charge token for fee
Swap	11	0.3%	LP/Validator	0.25% LP + 0.05% Validator

Based on ZKSwap V1, the calculation model for Swap transactions has been optimized.

Once the pool is created and liquidity is injected, users holding the corresponding tokens can swap in the pool.

Here we take the example of exchanging A for B. Assume there are x_i A and y_i B tokens in the pool before the swap, and the user wants to transfer m A to the pool. Then after the swap, there would be x_{i+1} A and y_{i+1} B tokens in the pool.

If A is a fee token, the system will first deduct 0.0005mA as the agreed transaction fee from the pool – where we can find the number of A written in the following format: $x_{i+1} = x_i + 0.9995m$. And the user can receive $y_i - y_{i+1}$ B tokens.

According to Uniswap's AMM algorithm, after deducting the liquidity fee of $(x_{i+1} - 0.0025m) * y_{i+1} = x_i * y_i$. Thus, the user will receive$y_i - y_{i+1} = y_i - \frac{x_i * y_i}{x_i + 0.997m}$ B tokens.

If A is not a fee token, the number of A token in the pool can be represented as$x_{i+1} = x_i + m$ and the user can receive $y_i - y_{i+1}$ B tokens correspondingly, keeping $(x_{i+1} - 0.0025m) * y_{i+1} = x_i * y_i$ after deducting 0.0025mA as the liquidity fee according to the AMM algorithm. Output $y_i - y_{i+1} = y_i - \frac{x_i * y_i}{x_i + 0.9975m}$ B tokens, on which the number of tokens the user will receive is $0.9995 * (y_i - y_{i+1})$ after a 0.05% fee is deducted by the protocol.

The liquidity fee is automatically added to the reserve of the pool after the transaction, so the reserve of the entire pool after the transaction becomes $x_{i+1} * y_{i+1} = c_{i+1} > c_i$. The total number of LP Tokens remains the same as no new LP tokens are generated or burnt, i.e. $n_{i+1} = n_i$. This means that the shares of all LPs remain the same, but the total amount of pool reserve per unit share has increased.

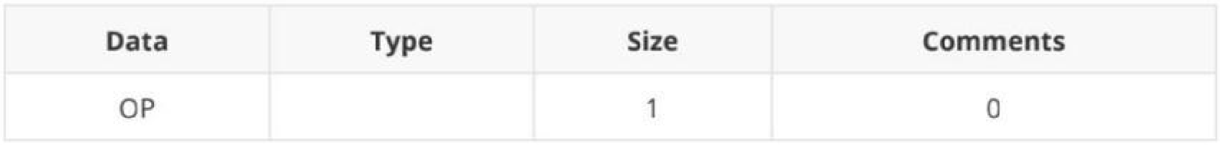
2.4. Pub Data of Transactions

All transactions on ZKSwap V2 (including Layer1 / Layer2 transactions) need to be packed and submitted as Pub Data to Layer1, with a chunk size of 11 bytes to maintain optimal circuit performance. Swaps and transactions only need 2 chunks.

2.4.1 Noop

The ZKSwap V2 allows empty transactions to populate the Layer2 blocks.

a. pub data



Data	Type	Size	Comments
OP		1	0

The total length of pub data = 1 byte.

2.4.2 Deposit

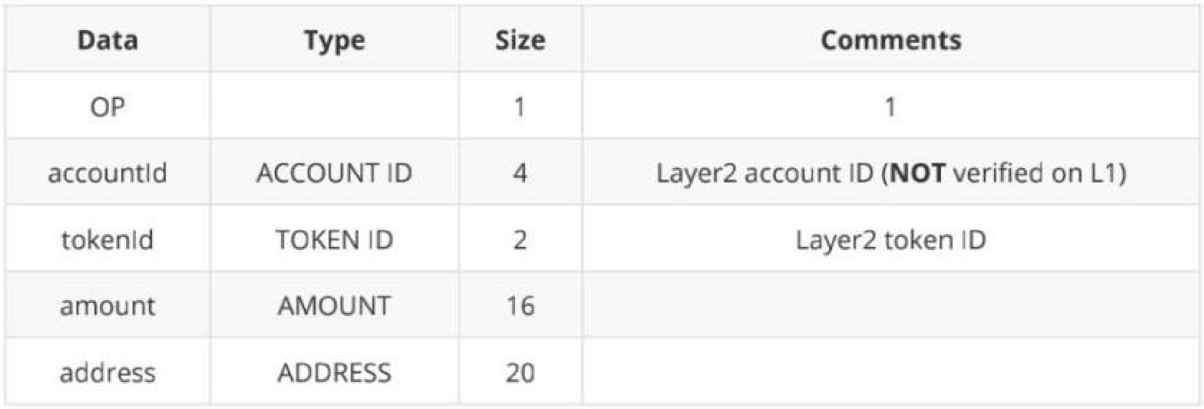
When a user makes a deposit from Layer1 to the ZKSwap platform, the system maps the user's Layer1 assets to Layer2.

a. Access

Access deposit ERC20 (IERC20_token, uint104_amount, address_franklinAddr)

function deposit ETH (address_franklinAddr)

b. pub data



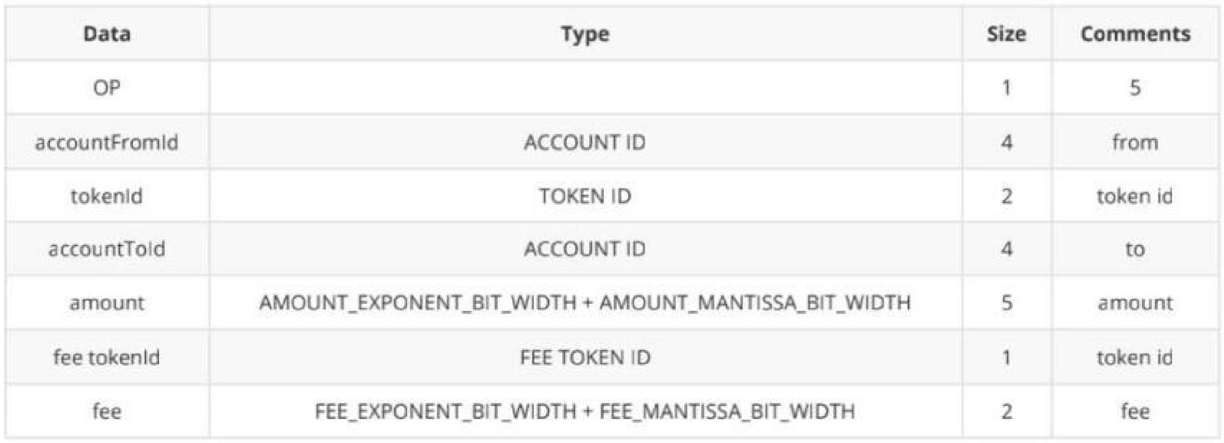
Data	Type	Size	Comments
OP		1	1
accountId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
tokenId	TOKEN ID	2	Layer2 token ID
amount	AMOUNT	16	
address	ADDRESS	20	

Total length of pub data = 1+4+2+16+20 = 43 bytes generating 4 chunks in total.

2.4.3 Transfer

User can make transactions with any Token via ZKSwap Layer2

a. pub data

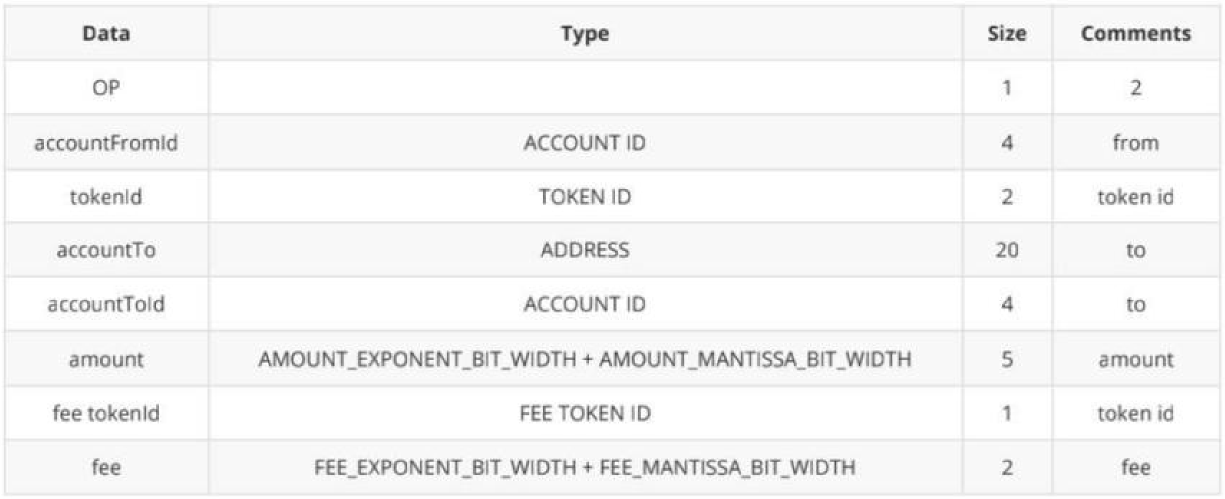


Data	Type	Size	Comments
OP		1	5
accountFromId	ACCOUNT ID	4	from
tokenId	TOKEN ID	2	token id
accountToId	ACCOUNT ID	4	to
amount	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	amount
fee tokenId	FEE TOKEN ID	1	token id
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	fee

Total length of pub data = 1+4+4+5+1+12 = 19 bytes, generating 2 chunks in total.

2.4.3 Transfer To New

Users can initiate any transaction at ZKSwap Layer2 without needing the recipient to create an account in advance.



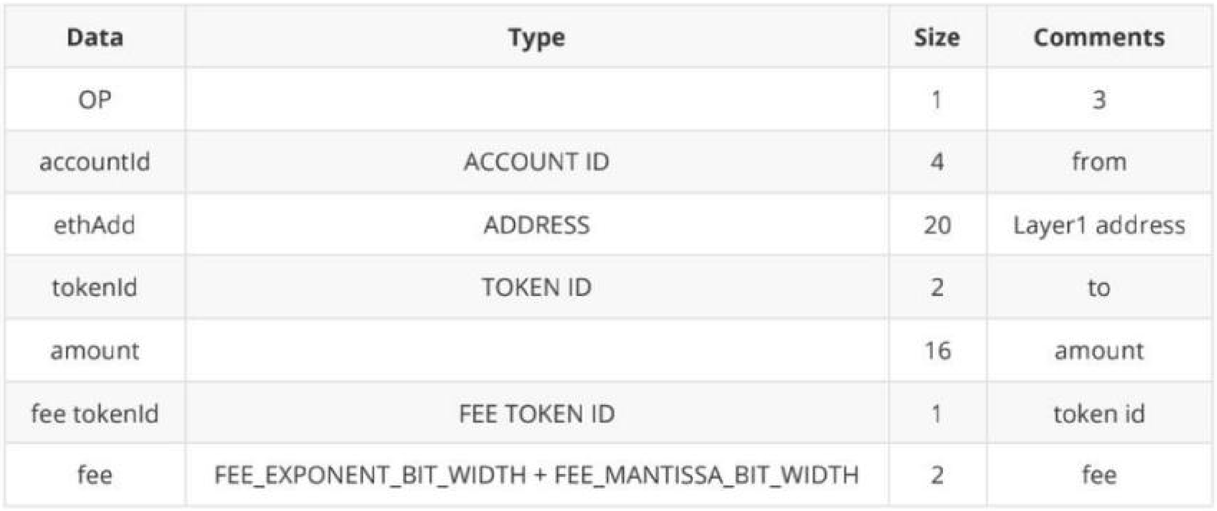
Data	Type	Size	Comments
OP		1	2
accountFromId	ACCOUNT ID	4	from
tokenId	TOKEN ID	2	token id
accountTo	ADDRESS	20	to
accountToId	ACCOUNT ID	4	to
amount	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	amount
fee tokenId	FEE TOKEN ID	1	token id
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	fee

Total length of pub data = 1+4+2+20+4+5+1+2 = 39 bytes, generating 4 chunks.

2.4.5 Withdraw

The user can initiate any token withdrawals at ZKSwap Layer2 to Layer1.

a. pub data



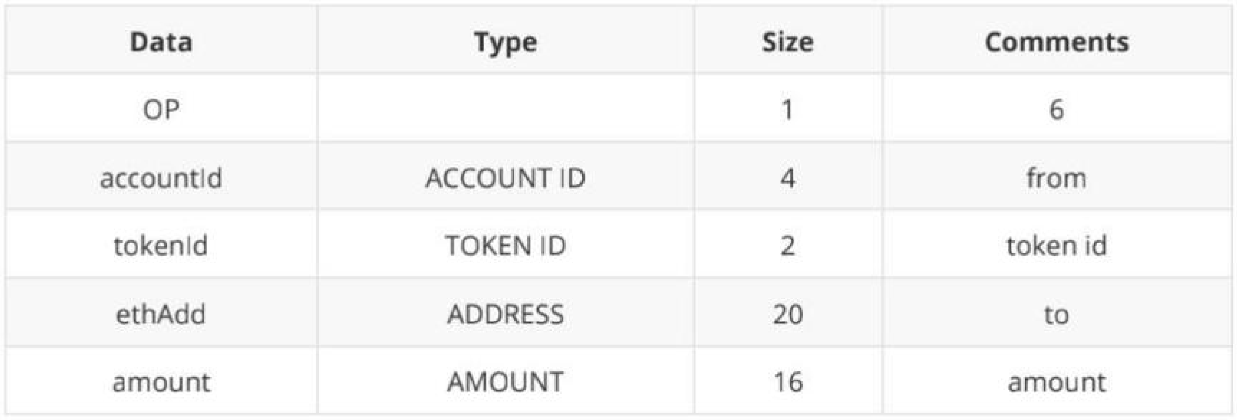
Data	Type	Size	Comments
OP		1	3
accountId	ACCOUNT ID	4	from
ethAdd	ADDRESS	20	Layer1 address
tokenId	TOKEN ID	2	to
amount		16	amount
fee tokenId	FEE TOKEN ID	1	token id
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	fee

Total length of pub data = 1+4+20+2+16+1+2 = 46 bytes, generating 5 Chunks.

2.4.6 FullExit

The user can initiate a FullExit request to withdraw assets directly from ZKSwap Layer1, which requires proof from Layer2.

a. pub data



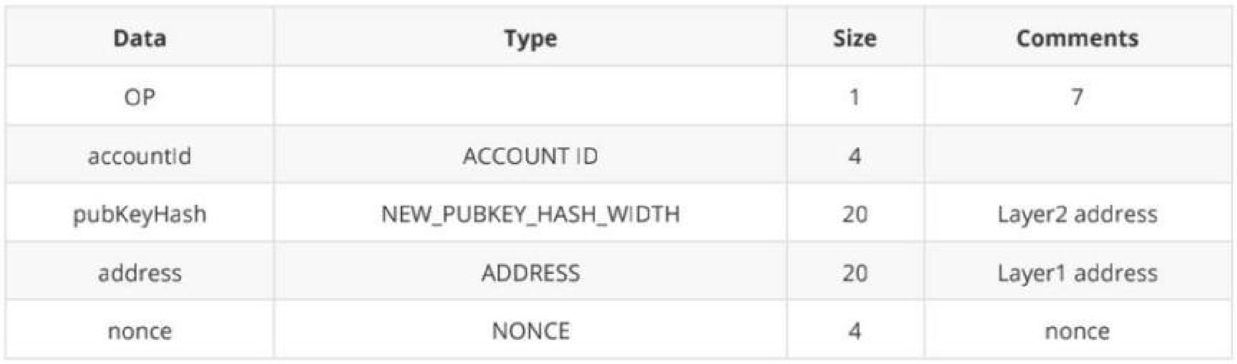
Data	Type	Size	Comments
OP		1	6
accountId	ACCOUNT ID	4	from
tokenId	TOKEN ID	2	token id
ethAdd	ADDRESS	20	to
amount	AMOUNT	16	amount

Total length of pub data = 1+4+2+20+16 = 43 bytes, generating 4 Chunks.

2.4.7 ChangePubKey

The user can withdraw any Token from Layer1 by initiating a withdrawal transaction to ZKSwap Layer2.

a. pub data



Data	Type	Size	Comments
OP		1	7
accountId	ACCOUNT ID	4	
pubKeyHash	NEW_PUBKEY_HASH_WIDTH	20	Layer2 address
address	ADDRESS	20	Layer1 address
nonce	NONCE	4	nonce

Total length of pub data = 1+4+20+20+4 = 49 bytes, generating 5 Chunks.

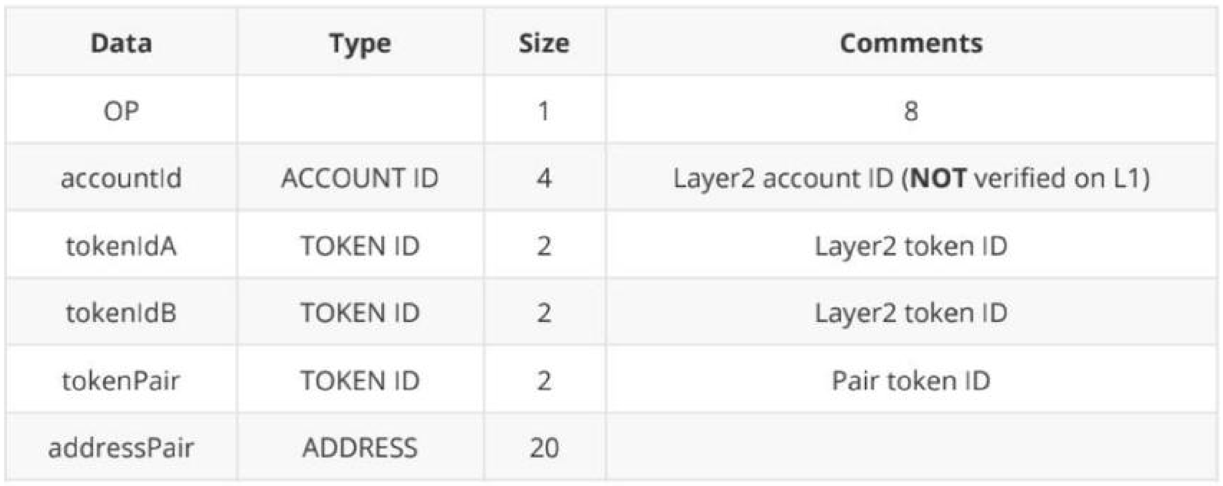
2.4.8 Create Pair

The user initiates the creation of a transaction pool via ZKSwap Layer1. The creation of a pool requires the creation of a corresponding smart contract (LP token) in Layer1.

a. Access function

function createPair (address_tokenA, address_tokenB)

b. pub data



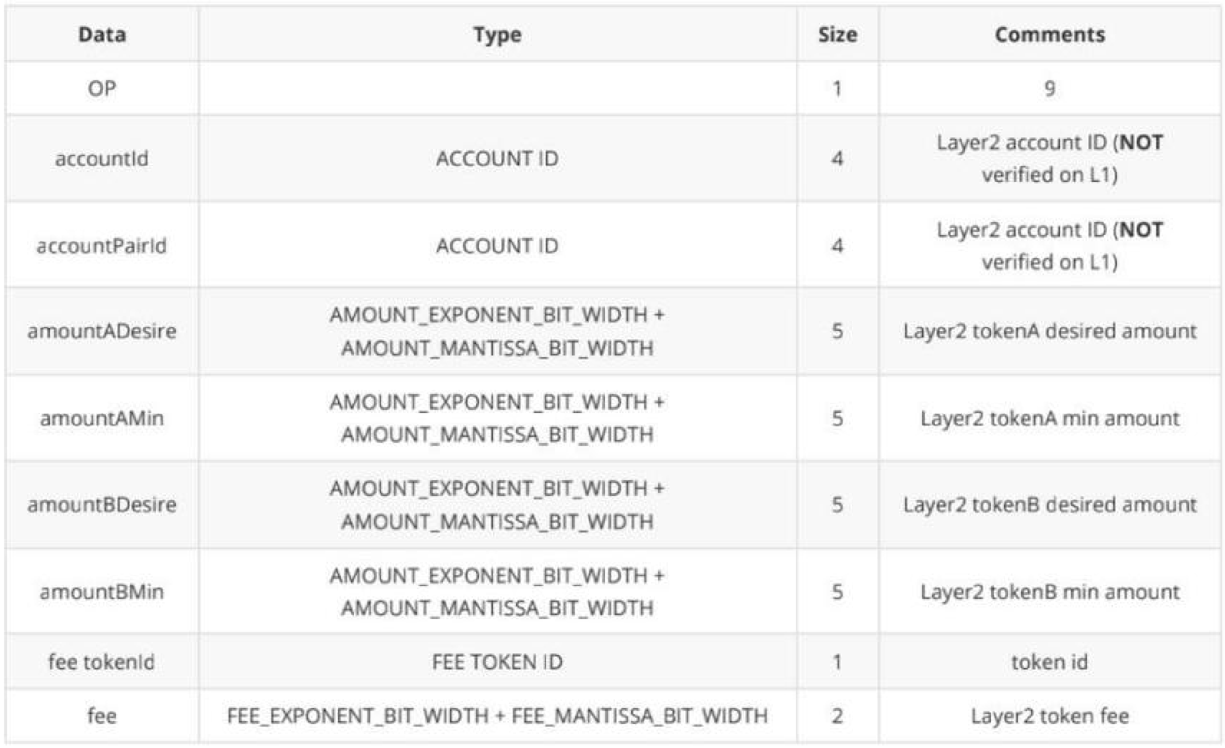
Data	Type	Size	Comments
OP		1	8
accountId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
tokenIdA	TOKEN ID	2	Layer2 token ID
tokenIdB	TOKEN ID	2	Layer2 token ID
tokenPair	TOKEN ID	2	Pair token ID
addressPair	ADDRESS	20	

Total length of pub data = 1+ 4+2+2+2+2+20 = 31 bytes, generating 3 chunks.

2.4.9 AddLiquidity

Users can add liquidity at ZKSwap Layer2.

a. pub data



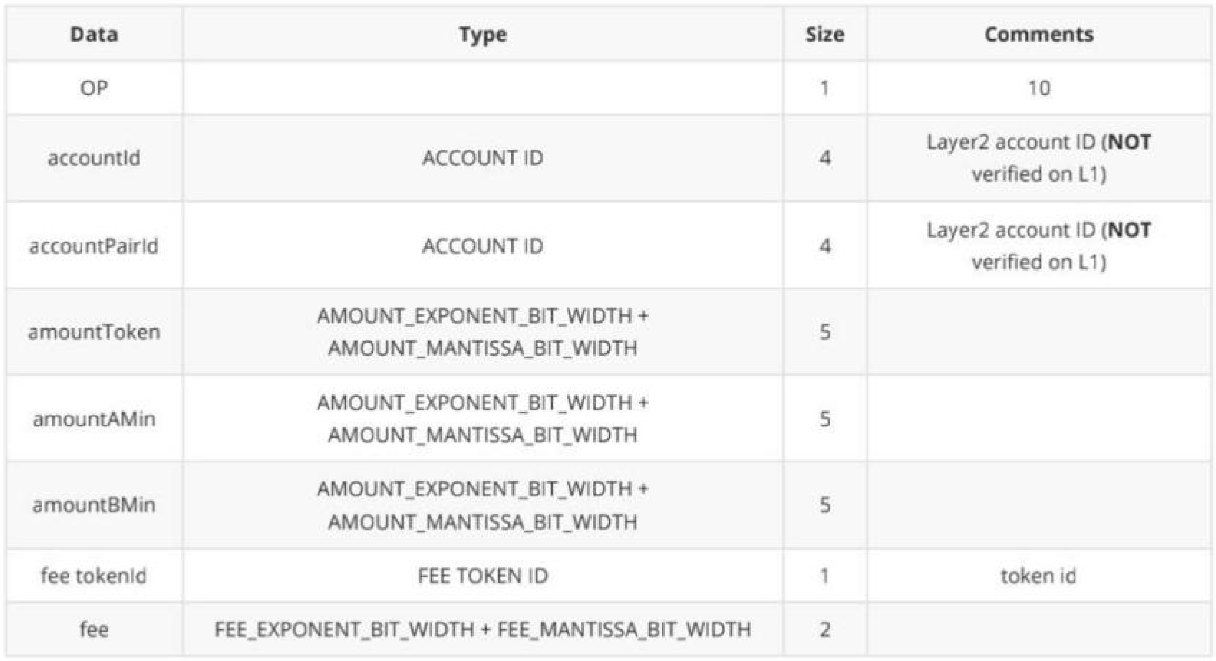
Data	Type	Size	Comments
OP		1	9
accountId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
accountPairId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
amountADesire	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	Layer2 tokenA desired amount
amountAMin	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	Layer2 tokenA min amount
amountBDesire	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	Layer2 tokenB desired amount
amountBMin	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	Layer2 tokenB min amount
fee tokenId	FEE TOKEN ID	1	token id
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	Layer2 token fee

Total length of pub data = 1+4+4+5+5+5+5+5+1+2 = 32 bytes, generating 3 chunks.

2.4.10 RemoveLiquidity

Users can remove liquidity at ZKSwap Layer2.

a. pub data



Data	Type	Size	Comments
OP		1	10
accountId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
accountPairId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
amountToken	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	
amountAMin	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	
amountBMin	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	
fee tokenId	FEE TOKEN ID	1	token id
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	

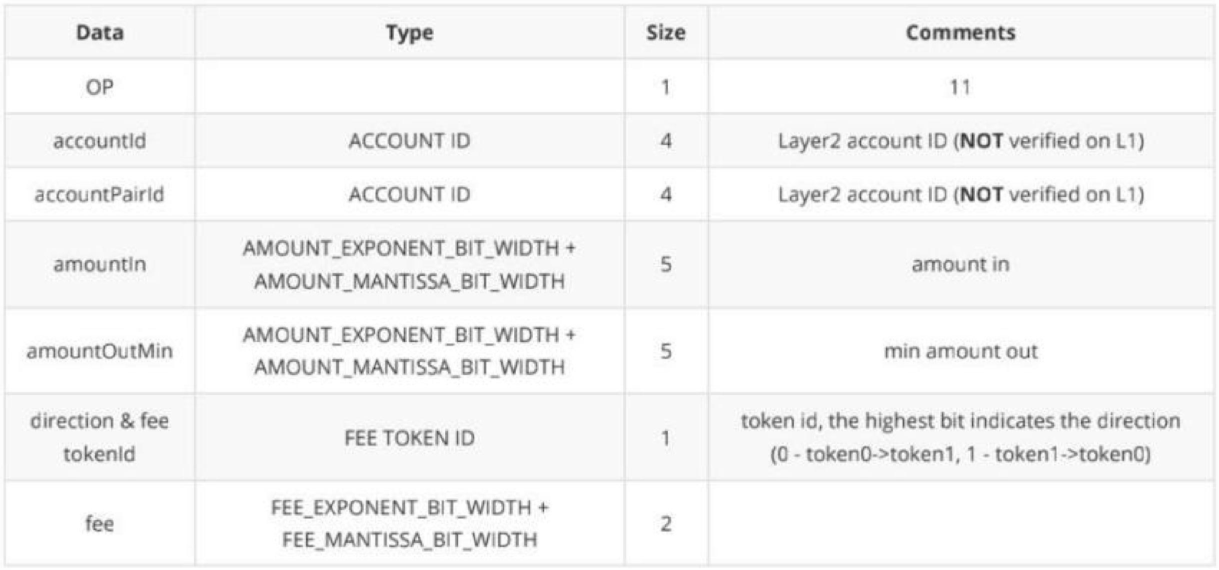
Total length of pub data = 1+4+4+5+5+5+5+1+2 = 27 bytes, generating 3 chunks.

2.4.11 Swap

Users can swap two Tokens at ZKSwap Layer2.

a. pub data

The total length of pub data = 1+4+4+5+5+1+2 = 22 bytes, generating 2 chunks.



Data	Type	Size	Comments
OP		1	11
accountId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
accountPairId	ACCOUNT ID	4	Layer2 account ID (**NOT** verified on L1)
amountIn	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	amount in
amountOutMin	AMOUNT_EXPONENT_BIT_WIDTH + AMOUNT_MANTISSA_BIT_WIDTH	5	min amount out
direction & fee tokenId	FEE TOKEN ID	1	token id, the highest bit indicates the direction (0 - token0->token1, 1 - token1->token0)
fee	FEE_EXPONENT_BIT_WIDTH + FEE_MANTISSA_BIT_WIDTH	2	

2.5. Circuit Optimization

To reduce the number of chunks for AMM related operations, the number of "branches" needs to be reduced in ZKSwap V2 accordingly. The original V1 design had an Account and a Balance of Tokens in the state tree forming a "Branch". In AMM-related operations, the original design is less efficient if the Fee Token approach is applied, and requires a "Branch" that supports one Account and two Token balances.

1) Account Audit Path (aap)

2) 2 Balances (balance0 / balance1) under the Account in the premodified Audit Pat (bap0/bap1)
3) 2 Balances (balance0'/balance1') under the Account in the premodified Audit Pat (bap0'/bap1')

Before modification, the circuit needs to prove :

1. balance0 + bap0 ==> b_root0

1. balance1 + bap1 ==> b_root0

1. account (b_root0) + aap ==> root

After the balance modification, the circuit needs to prove :

1. balance0' + bap0 ==> b_root0'

1. balance1 + bap1' ==> b_root0'

1. balance1' + bap1' ==> b_root0''

1. account(b_root0'') + aap ==> root'

3. Cash withdrawal optimisation

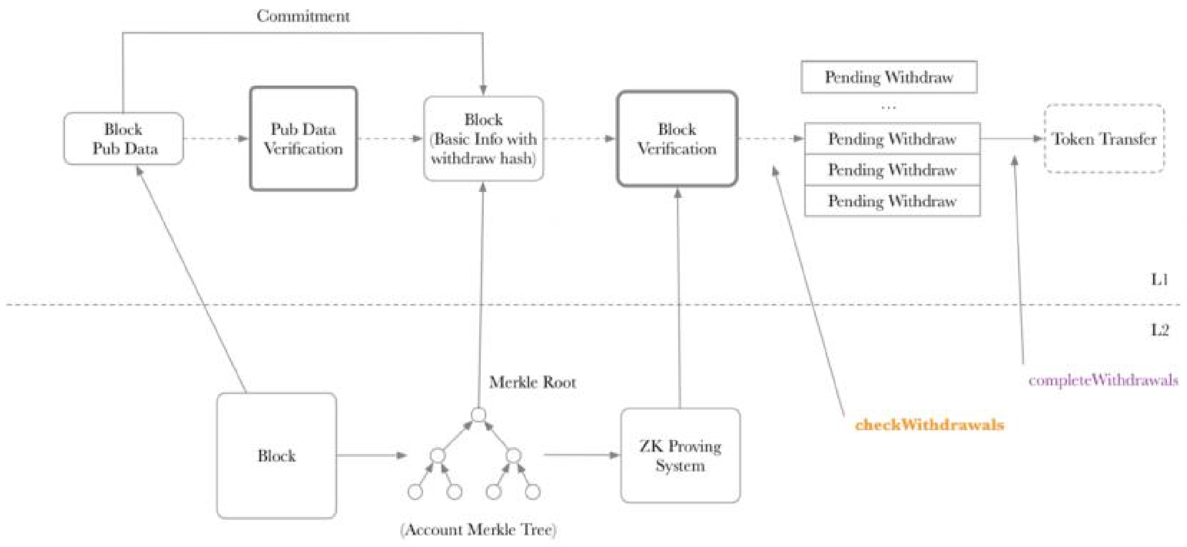
3.1 ZKSwap V1 version

Withdraw is the process by which the user withdraws the Token from Layer-2, unlocks it from the ZKSwap contract and sends it to the corresponding Layer-1 account.

In the ZKSwap V1 version, the Withdraw operation is initiated by the user from Layer-2. Upon receiving a withdrawal request from the user for a Token, the ZKSwap server updates the status of the corresponding Token under the corresponding account and sends the updated status tree root hash and the Withdraw operation applied In ZKSwap V1, the withdrawal operation is bundled with the block verification operation. The number of withdrawals in a block is limited due to the per-gas cost limit. In V1, the number of withdrawals per block was limited to 4.

3.2 ZKSwap V2

In ZKSwap V2, we have separated the “Withdraw” operation from the block “Verify” . The number of Withdrawals in a block would no longer be limited, which significantly increases the efficiency of withdrawals.



After a block has been verified, pending withdrawals are created and completed via functions like <checkWithdrawals> <completeWithdrawls>. completeWithdrawls need to be enhanced to prioritise a pending withdrawal. Block verification is implemented for multi-block aggregation. checkWithdrawals is processed for each verified block. The gas cost to create a PendingWithdraw is around 7w. Based on a maximum gas cost of 1250w for a transaction, the maximum number of withdrawals a block can support is 178.

4. Summary and Outlook

ZKSwap is a ZK-Rollups based technology, a decentralized Layer-2 token AMM protocol. It allows users to transfer, trade, add liquidity and remove liquidity in real time without having to pay for gas,and can significantly reduce the barriers and costs of use.

ZKSwap V2 version has been optimized for Token management, account management, Fee model, withdrawal process, etc. The number of Tokens supported by the platform has been greatly expanded and users can upload unlimited tokens (all ERC20 compliant tokens) on their own. In addition to eliminating the gas fee, the platform also innovatively supports user-selected transaction fee Tokens, further enhancing the user experience.

ZKSwap V2 is still being developed with the support of L2 Labs, a team that will promote the "Layer2 for all" multi-chain ecosystem strategy, which aims to achieve a trusted public chain scaling scheme through cryptography and algorithms, and reduce the cost for users and developing teams.

At the same time, ZKSwap will also launch the Ethereum Layer 2 NFT protocol, enabling users to create and distribute NFTs on ZKSwap Layer 2, and enabling Layer1 and Layer 2 interoperability. NFT on the ZKSwap platform will co-exist with AMM functionality and support other Ethereum-based NFT platforms to access this Layer 2 protocol. In the future, L2 Labs will advance Layer 2 expansion in more fields to become more secure, universal and open as a Layer 2 financial infrastructure.