Whitepaper

Hinkal Protocol

Preface

Mass adoption of Web3 is only possible with compliant privacy.

- Institutions can only join DeFi if they are confident that no illicit counterparty can transact with them.
- Enterprises will only use wallets for salaries if they feel safe about the privacy of these transactions.
- And B2B payments in stablecoins can only happen if this can be made privately.

Hinkal Protocol is the standard of privacy, making mass adoption happen. Hinkal Protocol implements an Ethereum zero-knowledge privacy solution: a smart contract that accepts deposits in ETH/ERC-20 tokens so that a deposited amount can be later withdrawn, transferred, or swapped/staked with no reference to the original transaction. Furthermore, it enhances security for users with a privacy-preserving KYC layer - minimizing the risks of being banned/sanctioned by government entities without revealing users' Personal Identifiable Information (PII).

1. Introduction

The transparency of transactions on a blockchain network presents a significant challenge for the handling of confidential information. As all transactions are publicly visible on the blockchain, anyone with knowledge of a specific wallet address can easily analyze on-chain activity, determine asset holdings, view payments, and trace the source of funds. This presents a significant concern for institutional investors and enterprises utilizing blockchain technology for trading and payments.

While centralized exchanges provide privacy as a default for institutions, decentralized exchanges do not. This lack of privacy leaves institutions vulnerable to financial loss as

other market participants may be able to front-run them during token and coin purchases.

Currently, in the decentralized finance (DeFi) landscape, there are several notable developments. Firstly, following the collapse of FTX, institutions are shifting towards non-custodial solutions as they do not trust intermediaries, resulting in increased adoption of DeFi platforms such as decentralized exchanges (DEXs) and lending protocols. Secondly, the DeFi industry in the United States is preparing for tighter regulation from the Office of Foreign Assets Control (OFAC), with a focus on becoming permissioned and better suited for institutional use through implementation of know-your-customer (KYC) protocols. Finally, DEXs are actively working to serve institutions by implementing KYC and clean liquidity pools, as this represents a large and crucial market for their growth as the retail user base matures.

Hinkal Protocol addresses the need for privacy in the distribution channels of these institutions and enterprises: DEXs, lending protocols, and wallets through the development of a KYC-powered privacy protocol. This will enable individual, institutional and enterprise users to feel secure in their transactions.

2. How it works

Hinkal Protocol accepts ETH/ERC-20 tokens as deposits, which can be later privately transferred, swapped/staked, or withdrawn with no reference to the original address. Each user holds a shielded address where tokens are stored after depositing. Hinkal Protocol uses zkSNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) to send private transactions. A user can generate a zkSNARK of the transaction without revealing key transaction parameters such as the origin/destination addresses or the amount of the transaction.

When a user decides to deposit funds in Hinkal Protocol - he is re-directed to the KYC aggregator aiPrise. aiPrise gives an identifier to the wallet and sends a request to the KYC provider (IDology, Veriff, ShuftiPro). The KYC provider collects PII, verifies the user, and issues proof that the user has passed KYC. After aiPrise sends the verification result to Hinkal Protocol, it is recorded as a non-transferrable access token in a smart contract (in case of successfully passing KYC). Only after obtaining an access token, the user can deposit funds.

Every time a user performs a transaction using Hinkal Protocol, the browser wallet computes a Zero Knowledge Proof (ZKP) and creates (or nullifies) a commitment. A

commitment is a cryptographic primitive that allows a user to commit to a chosen value while keeping it hidden to others, with the ability to reveal the committed value later. Commitments are created and spent after deposits, transfers, and swaps. A Nullifier is the result of a combination of a commitment and the nullifier key. Once the nullifier is broadcasted on-chain, the commitment is considered spent. Commitments are stored in a Merkle Tree structure. The root of this Merkle Tree is stored on-chain.

Withdrawal of funds from the smart contract can be made in two different ways:

a) The withdrawal transaction is generated by the recipient himself. The recipient should have enough funds to pay a gas fee for the transaction.

b) Coins/tokens are withdrawn through a Relayer with the percentage of the deposited amount sent as a fee to the Relayer’s address. The value is deducted from the total amount deposited. In this case, the withdrawal transaction is initiated by the Relayer, who pays a gas fee for the transaction. Relayers never have custody of the funds and instead, simply relay the transaction message.

2.1. Setup

To make the Hinkal Protocol work, the following contact setup is required:

a) Hinkal Pool

b) MerkleTree

c) AccessToken

d) RelayStore

Hinkal Pool smart contract contains crucial functions of the protocol - deposit, transfer, withdrawal, and swap (including staking). It is the backbone contract for the protocol, and Hinkal front-end directly interacts with it. In addition, Relayers are communicating with Hinkal Pool to withdraw funds on behalf of a target wallet address.

MerkleTree smart contract stores commitments when users deposit, transfer and withdraw funds. Merkle Tree structure is optimal for checking whether a user holds funds in the protocol. Specifically, verifying whether a leaf is a part of a Merkle Tree is done in a linear time.

AccessToken. Before executing a deposit or withdrawal operation, Hinkal Pool checks if the user passed the verification by aiPrise. It sends requests to AccessToken Smart Contracts requiring information on the KYC/AML checks.

RelayStore smart contract stores credentials and ratings of the Relayers. Ratings are used to determine the chance a given relayer will be chosen for a withdrawal operation. As of now, only Hinkal Protocol is able to add more relayers into RelayStore, but eventually, the process of adding new Relayers will be governed by Hinkal DAO. The list of all Relayers can be extracted from RelayStore through getRelayList function call.

2.2. Access Token

The user must pass a KYC verification before depositing. After the user clicks the deposit button, he is prompted to walk through KYC verification flow by aiPrise. At this point, neither Hinkal Protocol nor aiPrise has access to inputted information - it is securely sent to the KYC provider. Once the verification of PII is done, the user is led to an access token minting step, where he can mint a non-transferable token proving that he passed the KYC checks. Once he successfully minted an access token, he is eligible to deposit funds into the smart contract.

No single party has a link between PII and wallet information: Hinkal Protocol has wallet address information - but no PII, while the KYC provider has PII but no wallet address. This preserves the necessary level of user anonymity. The only way the information can be revealed - if a court issues a subpoena to check certain transactions - in this case Hinkal Protocol should cooperate with aiPrise and the KYC Provider to link wallet and PII information**.** This mechanism ensures that users can transact securely.

Important: We suggest waiting some period after receiving an access token and before the user deposits funds to Hinkal Protocol. This way, one can ensure no other party can access the user's PII even if the KYC provider's database gets hacked. It is achieved by disentangling the time when the user received an access token and of depositing funds.

2.3. Deposit

For depositing, the user can choose any amount of ETH or ERC-20 token. Deposits convert publicly visible ERC tokens into a token commitment that holds the same value as that of the original token, and the Hinkal public key of the intended commitment

owner. The commitment is stored in MerkleTree smart contract as a leaf if the amount is successfully deposited.

2.4. Transfer

Transfers enable the transmission of up two commitments of the same asset between two parties by nullifying the previous commitments and creating up to two new commitments. To ensure a recipient receives the secret information required to spend their commitments, the sender encrypts the secrets (*salt*, *value*, address) of the commitment sent to the recipient and proves using ZKP that they encrypted this correctly with the recipient's public key. A Transfer ZK Proof proves that the prover has nullified up to two old commitments which existed in the Merkle Tree, created one new ones, and encrypted its information for the recipient. In either case, the information leaked will be that an Ethereum address has nullified commitments amongst the commitment pool owned by the transmitter, and that new commitments have been created. Information on the new owner, which commitments were spent or the amount transferred remains private.

2.5. Withdrawal

Withdraw is the operation of nullifying an existing Hinkal commitment and converting it into publicly visible ERC tokens with the same value as the burnt commitment. Withdraw is the opposite operation to Deposit. Similarly to transfers, withdrawals accept as input up to two commitments.

A Withdraw ZK Proof proves that the prover has nullified up to two old commitments which existed in the MerkleTree.

Information leaked during a withdrawal includes the address of the address that withdrew the commitment and the value address of the token withdraw.

The user can manually choose over 2 types of withdrawal:

1. To withdraw on his own, meaning that the connected withdrawing address has to pay a gas fee.
2. Withdrawing using a Relayer. Suppose a wallet address is newly created and does not possess the required amount of funds to cover a gas fee associated with a withdrawal transaction. As in the first option, the user has to provide the recipient address, but no prior Ether balance is needed in that address, since the Relayer is

incurring an associated gas fee. For the withdrawal service, the Relayer takes a fixed fee (0.3%) from the withdrawal amount. The total withdrawal amount will be determined according to the formula:

$$Amount_{withdrawal} = Amount_{deposited} - Fee_{relayer}$$

During the withdrawal, the user generates a zk-SNARK, which is sent to Hinkal Pool smart contract, that verifies the validity of the proof and checks Nullifier Hash. The Nullifier Hash produced during the verification allows Hinkal Pool smart contract to control that the withdrawal is made only once.

2.6. Swaps and DeFi Operation

Hinkal Swaps are allowing users to swap ERC-20 tokens privately. To achieve privacy, Hinkal uses relayers and Uniswap API. When a user wants to swap assets, he generates zk-proof that he is eligible to swap a given amount of the first asset in return for the incoming amount of the second asset.

From the viewpoint of outsiders, the swap transaction will be submitted from the relayer's public Ethereum address to Hinkal Pool smart contract. In the transaction, Hinkal Pool will call Uniswap Smart Contract in order to swap the assets. If for some reason, the swap will fail, then the whole transaction will be reverted, so no additional commitments and nullifiers will be created. Important to mention, the relayer does not have custody of the swapped assets.
Similarly to swaps, Hinkal implements private lending with AAVE and private staking with Lido protocol. Conceptually, lending and staking are similar to swaps, since these transactions include the exchange of one token for another type of token. For AAVE, the user exchanges tokens to A-tokens and for Lido, ether is exchanged for wstEth.

Since the protocol implements UTXO based model of commitments and nullifiers, the swapped amount should be known at the time of zk-proof generation. At the time of the swap transaction submission, it is unknown how many other swap transactions will be submitted in the current block. This factor generates unpredictable slippage from the viewpoint of the user and relayer. In order to mitigate the uncertainty factor, the slippage

risk is shifted to the relayer. To accommodate the additional risk, the relayer charges 10 Beeps more than in the case of a withdrawal transaction.

3. Competition

Hinkal Protocol offers a competitive advantage over other privacy-focused protocols through its emphasis on safety, private swaps, and integration with decentralized exchanges (DEXs), lending protocols, and wallets.

Firstly, the protocol's safety measures ensure that bad actors are unable to access the platform, mitigating government-related risks and ensuring that only customers who prioritize the privacy of their transactions are served.

Secondly, Hinkal Protocol allows for compliant private swaps, currently within the Polygon ecosystem, with plans to expand to cover most tokens within the Ethereum Virtual Machine (EVM) ecosystem.

Lastly, the protocol's ability to integrate with leading DeFi protocols such as DEXs, lending protocols, and wallets allows for a larger distribution channel to be captured, rather than building a distribution path from scratch. Unlike other privacy protocols, Hinkal Protocol has the necessary structure that enables DEXs to safely integrate privacy without worrying about additional risks, such as potential sanctions from the Office of Foreign Assets Control (OFAC) for non-compliance.

4. Decentralization plan

The plan to decentralize Hinkal Protocol consists of several steps:

1. Different KYC providers will have access to Hinkal Protocol, so users can choose a provider to pass verification. KYC providers will compete with each other to offer a quality product at a competitive price. The competition will eventually drive KYC prices down and create more rigorous identity checking.
2. Introduction of incentives to decentralize Relayer network. The Relayer network will be extended to multiple players who will compete to get higher user traction. It is important to have an external Relayer network to decentralize the protocol, since it will eliminate a centralized player which runs Relayers.
3. Hinkal Protocol will allow users to possess fragmented identity. The user will be able to pass a KYC check using one account and prove the fact of passing KYC with

other accounts. The user will have a secret key or mnemonic, and he will use it to generate a zk-proof containing information that he holds credentials. This step will eliminate the need to pass KYC separately for a deposit and withdrawal transaction.

5. Mission

Hinkal Protocol builds the future of financial privacy for the next billion of Web3 users, including institutions and enterprises. We believe that compliant privacy is the biggest trigger for the broader adoption of blockchain as the financial rails.