Chainflip Protocol Whitepaper

Simon Harman

24th June, 2022

Fourth Revision
First Version Published May, 2020

Developed and written with the help of Dr. Maxim Shishmarev, Thomas Nash, Daniel Lennon, Shaun Janse van Vuuren, and the rest of the Chainflip Labs team.

Chainflip Labs GmbH
Berlin, Germany
https://chainflip.io
team@chainflip.io

Contents

Introduction & Hypothesis **3**
About this Paper 4

The Chainflip Core Protocol **5**
Conceptually Separating Settlement and Accounting Layers 5
Defining the Validators 5
The Ground Floor: Constructing Vaults 6

Ingress: How Chainflip Gets Assets **7**
Witnessing 7
Case 1: Staking to the State Chain 8
Case 2: Adding Collateral to a Liquidity Position 8
Case 3: Conducting a Swap 9

Egress: Sending Assets out of Chainflip **9**
Threshold Signature Scheme (TSS) Signing Ceremonies 9
The Advantages of Using FROST 9
The FROST Signing Scheme in Chainflip 10
Transaction Signing 10
Consistent Broadcast 11
Case 1: Withdrawing Staked FLIP 11
Case 2: Withdrawing Collateral from a Liquidity Position 12
Case 3: Paying out users for Swaps 12

The Chainflip JIT AMM - A Novel Protocol for Capital Efficient Cross-Chain Swaps **13**
AMM Development Background 13
Just In Time AMM Core Features 14
How JIT Works 14
An Example of JIT Swaps 14
Some Caveats to the Example 15
Other Benefits & Tradeoffs of the JIT AMM 16
The Extremes of Capital Efficiency 16
Constraining Liquidity Pools For Efficiency 17
The Network Fee 18

Ingress Addressing and Relayers **19**
Ingress Addressing 19
Relayers 20
Relayer Fees 20
Direct Vault Trades 21
Swap Intent & Vault Expiry: Handling Vault Rotations 21

Chainflip Validators **22**
The Authority Set 22
Authority Set Auctions 23
Backup Validators 24
Reputation & Slashing 25
Heartbeats & Positive Reputation 25
Suspension Conditions 25
Reputation Rollover 26
Backup Validator Reputation 26

Governance Processes **27**
The Governance Keys 27
Voting in New Keys 28
Runtime Upgrades & Hard Forks 28

Security Considerations & Features **29**
Potential Protocol Risks . . . 29
Protocol Level Security Features . . . 29
State Chain Safe Mode . . . 29
StakeManager Protection . . . 30
Contract Vault Protection . . . 31
Security Ratio . . . 31
Supermajority Attacks . . . 32

Future Work: "Hub and Spoke" Cross-chain Liquidity Network of Networks **33**

Conclusion **33**

Introduction & Hypothesis

For over a decade, the primary method of transferring value between blockchains has been the centralised exchange. This reality has limited composability between ecosystems and has made it difficult for specialist or alternative blockchains to be readily adopted by increasingly large Web3 userbases. Despite this, the vast majority of Web3 users rely on non-custodial wallets to perform even the simplest functions with DeFi, NFTs, DAOs, and a myriad of other applications.

Engineering a generalised solution to this problem has been under discussion as far back as 2012[1]. A decade later, and centralised exchanges still have a firm grip on the market share, but on-chain solutions have been making significant progress.

Uniswap, a competing implementation of the core Automated Market Maker (AMM) idea first created by Bancor in 2016, revolutionised crypto-markets forever by popularising liquidity pools and on-chain trading in mid-2020. After fighting off several competing protocols, Uniswap v3 has cemented the Uniswap product as the dominant solution[2] to swaps on Ethereum mainnet since its release in mid-2021.

Uniswap achieved this by enabling on-chain trading not just for end users, but also by allowing other products and contracts to programmatically swap tokens behind the scenes, something which had not been previously possible. This allowed for the creation of a variety of user experiences and DeFi products which have generated significant fee revenue on Ethereum since 2020. Efficient token swaps powered by Uniswap have enabled the DeFi ecosystem as a whole to rapidly evolve, and created an ecosystem of trading and finance run entirely on-chain.

Author's assertion: *"With no Uniswap, there would probably be no Aave, 1inch, Curve, Yearn, or even OpenSea. On-chain trading is an immense force-multiplier and is perhaps one of the most important primitives in all of crypto development."*

Due to the protocol's success, several Uniswap-based protocols have appeared on alternative smart-contract blockchains. This has allowed for relatively easy on-chain trading in a single-chain smart contract execution environment, with much of the same positive impacts Uniswap had on Ethereum within each individual blockchain ecosystem.

However, Web3 extends far beyond a single environment. As scalability problems continue to impact the user experience of popular chains, either through high transaction fees, unreliable network performance, or additional friction in the user experience (such as that with sharding and L2s), it is clear that multiple, mutually isolated execution environments will coexist long into the future. It is also clear that application specific blockchains and whole ecosystems are still cut off from on-chain markets, and that there is no good method of programmatically trading between any of them.

Since the original Chainflip Whitepaper was published in May 2020, dozens of "cross-chain" solutions have emerged on the scene. Whilst it is clear that there is huge market demand within the sector, none have achieved widespread adoption to the extent of Uniswap, in spite of the fact that the addressable market seems much bigger than ERC-20 tokens alone. Reasons for that can be summarised by the following:

- The proposed solutions are often overcomplicated, under-scrutinised, and/or highly centralised[3], exposing LPs and users to ongoing risks.
- Software vulnerabilities[4] have been so frequent[5] and expensive[6] to the point that some users have come to trust centralised exchanges *more* than trustless solutions.
- Many products have been built that rely on creating or utilising synthetic assets which violate the sovereignty of execution environments[7], fragment liquidity, and ultimately degrade the user experience for everyone.
- The typical cross-chain user experience is almost universally poor and badly explained.
- Products that do offer native swapping often do so at rates which can be described as vaguely similar to the global index price at best, and often with unacceptably high slippage. Accurate and fair swap pricing should not be an afterthought.
- So called "generic cross-chain messaging" solutions are extremely limited in what they can practically enable, making the addressable market for these products small relative to that of spot markets in general.

Rather, the method for transferring of coins between chains would be better if:

- It is wallet and chain agnostic. That is, it supports any generic wallet that can send ordinary transactions on any type of blockchain.
- It does not require native chains to support a specific execution protocol or make changes to its underlying consensus rules or infrastructure, meaning it is generalised and not limited to the Ethereum Virtual Machine (EVM) or smart contract enabled chains.
- It executes as much computation off-chain as possible, meaning that gas usage on expensive chains is minimised and the execution environment can be customised to better suit the use case.
- It does not involve any 'wrapped' or synthetic assets. That is, there was simply one generic transaction submitted to conduct the swap, and users are not exposed to any risk after the swap is complete.
- Developers could easily leverage the technology to improve their own products through simple Remote Procedure Calls (RPC) calls, meaning no special wallets, Software Development kits (SDKs), or other complex frameworks would be needed for the protocol to be leveraged by application developers.

Programmatic swapping is what Uniswap enables on Ethereum. Chainflip's ultimate goal is to enable programmatic swapping for all major blockchains, unlocking new possibilities for product developers, and offering users an easy to use, reliable, secure, and permissionless method of avoiding custodial exchanges altogether.

Our thesis is that if this goal can be achieved, then other types of interoperability solutions will not be desirable for most users, and the Web3 space can finally begin to rely on on-chain markets as a universal mainstay of the industry.

About this Paper

This is the fourth version of this whitepaper. The first was written under the name "Blockswap" and was published in May of 2020. Original contributors to the Blockswap paper include Kee Jefferys, Jason Rhinelander, Dr Maxim Shishmarev, and Simon Harman. The original Chainflip paper can still be read online, but this updated version is intended to reflect the actual design of Chainflip in practice, with a multitude of changes to the protocol having been made through the process of 18 months of implementation.

This paper is not intended to be a canonical description of all components and their functions. Our documentation is kept up to date most frequently and in more detail at `https://docs.chainflip.io`. The goal of this paper is to offer a medium level overview of the system, how it is intended to fit together and why certain design choices have been made in relation to the thesis. To do this, the components of the Chainflip design must be laid out, but as a distributed and continuous system, there can be no natural logical place to begin, as every major competent is co-dependant on others. Instead, let's begin with some helpful abstractions to conceptualise the system.

The Chainflip Core Protocol

Sometimes called a "Cross-Chain Liquidity Network," the core of the Chainflip idea is to use MPC (Multi-Party-Computation), and in particular TSS (Threshold Signature Schemes) to create aggregate keys held by a permissionless network of 150 **Validators**. These Validators control simple smart contracts/wallets called **Vaults**, on multiple blockchains simultaneously, giving rise to a fully decentralised **Settlement Layer**. This is paired with an **Accounting Layer**, which uses the Substrate based **State Chain** to track balances, process events, and execute instructions.

Conceptually Separating Settlement and Accounting Layers

Although actual implementation itself does not function with a perfect separation of the Accounting and Settlement Layers, breaking down the system into two layers is a helpful way to conceptualise its structure.

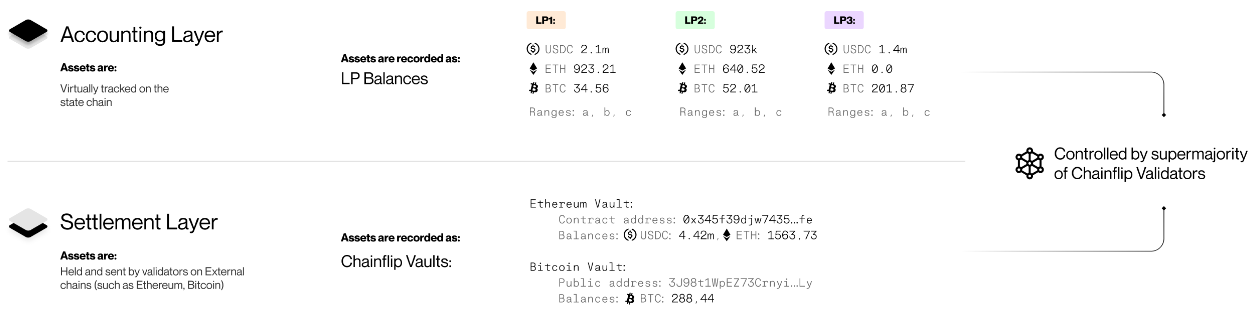


Fig 1: The Settlement and Accounting Layers of Chainflip

Although the assets are held in native wallets on chains such as Ethereum, Bitcoin, and so on, all of the trading and account keeping occurs on the State Chain. This means that swap execution and AMM design can be heavily customised for the specific use-case of cross-chain value transfer.

In practice, the Accounting Layer exists on the **State Chain**, which is an "intermediate" blockchain running the Aura Proof of Stake consensus system as provided by Substrate, a blockchain framework developed by Parity and the Web3 Foundation. Adding additional events and extrinsics to the base functionality of Substrate enables the accounting functionality of Chainflip.

At a high-level, the **Chainflip Just In Time (JIT) AMM** is roughly based on the Uniswap v3 AMM design. However, unlike Uniswap v3, the JIT AMM is not represented as a series of smart contracts on a single blockchain environment. Rather, the JIT AMM is **virtual**. This means that funds are not held directly in on-chain pools, but rather the AMM is part of the Accounting Layer. The Accounting Layer tracks incoming funds into the vaults using information fed from the Settlement Layer, and determines what should be sent from them. This works in a similar fashion to how centralised cryptocurrency exchanges store all of the deposits in a unified wallet system, with the balances of individual users tracked in a connected but logically separated accounting and trading system.

The use of a separate Accounting Layer dramatically simplifies the work needed to support individual chains, as rather than needing to write swapping logic in a range of smart contract and scripting languages on external blockchains, the swapping logic is entirely contained within the Chainflip State Chain environment, meaning all of the abstractions needed to support a given chain can be managed entirely within the Settlement Layer, and are much simpler in comparison.
The two systems, although logically separated, are controlled by the same set of Validators on the Chainflip network.

Defining the Validators

A Validator, in the context of Chainflip, is a server operating a Chainflip State Chain node, as well as additional 'sidecar' functionality through the **Chainflip Engine (CFE)**, an off-chain processing module. This paper has a full section on the Validator functions, requirements, and rules, but for the purposes of introduction, here is a high level description:

Up to 150 Validators can participate in the protocol's **Authority Set** based on their stake and jointly operate both layers. All Validator slots are entirely permissionless, where any Validator operator with sufficient FLIP to outbid others can become a member of the Authority Set, by winning slots during regular **auctions**. Each individual Validator has its own set of private-keys which it uses to participate in the consensus of the State Chain, as well as to generate secrets for the purposes of MPC. Validators in the Authority Set perform the following functions:

- The Accounting Layer (State Chain):
  - Maintain and update the State Chain through Aura proof of stake consensus.
  - Come to consensus on incoming deposits and register them on the chain (Witnessing).
  - Come to consensus on proposed transactions to sign/broadcast.
- The Settlement Layer (Chainflip Engine):
  - Participate in TSS Keygen ceremonies to create Vaults.
  - Participate in TSS Signing ceremonies to move funds for the protocol.
  - Monitoring and Parsing incoming transactions to vaults on external chains to be Witnessed.

With this two-layer abstraction laid out, let's use it to understand more components and how they come together to solve the cross-chain swapping problem.

The Ground Floor: Constructing Vaults

Fundamental to any native asset swap flow is for there to be a pool of liquid assets somewhere. With no source of on-chain liquidity, there would be nothing to swap. At the ground floor of any credible native cross-chain swap protocol, a general solution to flexible liquidity storage must be engineered. In this section, we'll pull apart what a **Vault** is and where it fits into the wider protocol.

"A fundamental limitation of other interoperability protocol designs is that they lack the ability to secure native funds on host blockchains. So called 'messaging' or 'interoperability' protocols offer insufficient functionality to carry out large scale cross-chain trading. These protocols only enable 1 to 1 swaps, which limit their utility to users. This type of protocol design cannot fulfil the functionality ultimately necessary to render a centralised exchange offering obsolete, and in turn might explain why they have achieved limited adoption."

A **vault** is a method of storing funds on a given blockchain that is controlled by the Chainflip protocol. This could come in a few forms, with different blockchains offering different features and limitations, but broadly there are two categories of vault which are used to store funds to be used in the AMM:

1. **Vault Contract:** A smart contract (EVM or non-EVM) which holds a pool of assets on a given chain. The vault smart contract can send funds with the approval of an aggregate key held in the KeyManager contract. The aggregate key is controlled by the Authority Set in a 100 of 150 threshold signature scheme (FROST) handled off-chain. When the Authority Set is changing, the old Authority Set approves a transaction which changes the aggregate key from the outgoing Authority Set to the new aggregate key, controlled by the new Authority Set in the KeyManager contract.
2. **Native Wallet Vaults:** A native wallet or set of wallets controlled by the Authority Set in a 100 of 150 threshold signature scheme and is handled off chain. As with the vault contract above, the native wallet vault still relies on the FROST threshold signature scheme to function. The details of the native wallets vary from blockchain to blockchain, but rely on native multisig schemes offered as features on each blockchain. Funds are held in each Native Wallet Vault until a new Authority Set is determined, after which a **Vault Rotation** occurs. The new Authority Set will create its own Native Wallet Vault, and the outgoing Authority Set will sweep funds from its own Wallet Vault into the new one over the expiry period. This means that there will be multiple 'active' vaults during a vault rotation, until all outgoing vaults have expired. See *Swap Intent & Vault Expiry: Handling Vault Rotations* for more information.

Although not used for storing AMM funds, there is also a third protocol controlled vault-like contract that fills a special function called the:

3. **Stake Manager Contract:** The Stake Manager Contract is a smart contract (initially just on the Ethereum network) which holds FLIP funds as collateral for accounts on the State-Chain. Like the Vault Contract & Native Wallet Vaults, signature verification is delegated to the KeyManager contract, which verifies an aggregate signature produced by the current Authority Set. Unlike the Vault Contract, users directly interact with this contract, but can only claim FLIP from this contract with a valid signature from the aggregate key - or in other words, with the off-chain approval of a transaction by the Authority Set.

In all of these cases, a new Authority Set must perform all of the necessary key-generation ceremonies for all deployed vaults before authority over the State Chain consensus, vault contract, and stake manager contract are handed over, and before any funds are swept from old Native Wallet Vaults into the new.

No matter the underlying implementation of each vault, the end result is that there is an account on each supported blockchain under the control of the Chainflip protocol. This forms a functional, decentralised, and generalised Settlement Layer on which liquidity pools and staking functionality can be virtually constructed using assets held in these vaults.

Ingress: How Chainflip Gets Assets

Ingress is the generic process that enables funds to be sent to the Chainflip Settlement Layer, but is processed slightly differently depending on the intention of the user. The three cases that would induce the ingress process are:

1. A user wants to stake FLIP to the State Chain (for Validator auction bidding, relayer operation, or any other reason);
2. A Liquidity provider wants to **add liquidity** to their position, or;
3. A user wants to **swap** using the Chainflip protocol.

Witnessing

The main commonality between the three cases is **Witnessing**. All of the above processes rely on the ability for the Validator network to come to consensus about what deposits have actually been made to the vaults. Conceptually, we can say that the Witnessing process is functionally equivalent to a Chainflip-specific Oracle service.

The Statechain can tell Validators what to look at on external chains. Certain contracts on Ethereum, particular wallet addresses on Bitcoin, and a myriad of other addresses and contracts on other chains can all be added to a kind of "watchlist" for the Validators to keep an eye on.

As an essential function of the Settlement Layer, Validators are required to maintain connections to light clients, full nodes, or other kinds of services which allow the Validator to query all Chainflip supported blockchains. The way this is configured is up to each individual operator, but by losing connections to the external blockchains, the Validator will be sent offline until it has resolved the problem.

Once a related transaction to an address or contract on the "watchlist" occurs on the external chain, the Validators must wait until a pre-specified confirmation window has passed. The exact length of the confirmation window (measured in blocks) is assessed on a case-by-case basis for different supported blockchains.

Author's assertion: *"No decentralised interoperability protocol can escape the reality of blockchain finality and is uniquely vulnerable to rollbacks between independent chains, thus resulting in an unavoidable delay before secure witnessing and processing can occur."*

However, as soon as a block is detected on the supported blockchain which causes a deposit to be considered ‘final’ according to the Chainflip protocol’s definition, every Validator in the Authority Set submits a ‘witness’ extrinsic to the State Chain. When over $\frac{2}{3}+1$ of the Authority Set (100 of 150 in the standard case) have done so, this finalises the witness event. From then on, the State Chain registers the deposit as ‘real,’ and the transaction details such as the TXID, asset, amount, and any relevant metadata are recorded on the State Chain.

If a Validator submitted a witness extrinsic which did not align with what other Validators have seen on the supported blockchain, the witness extrinsic would never reach sufficient consensus on the State Chain. This ‘false’ witness could then be used as proof to punish the Validator that submitted it. Similarly, nodes which frequently fail to witness events that the other $\frac{2}{3}+1$ of Validators catch within a given time frame could also be grounds to programmatically punish a Validator through Slashing.

Witnessing is effectively how assets are passed through the Settlement Layer into the Accounting Layer, and is at the heart of all ingress processes. Let’s look at each one of the three cases that invoke ingress:

Case 1: Staking to the State Chain

FLIP is an ERC20 token. This means that in order to acquire a balance on the Chainflip State Chain, users have to lock collateral directly in the Stake Manager smart contract.

1. The staker submits a transaction to the Ethereum chain that calls a function in the Stake Manager contract that passes in their ChainflipAccountID (a hex representation of a substrate SS58 encoded public key) and the amount of FLIP tokens being staked.
2. The Authority set listens to the Ethereum blockchain for transactions related to the StakeManager contract. Once the staking transaction has been confirmed on the Ethereum network, the Authority Set commences the **Witnessing** process.
3. As the transaction becomes witnessed, the deposit is considered “settled” - and so the account specified in the contract call becomes funded on the State Chain (or conceptually, the “virtual Accounting Layer”) with that amount of FLIP which can then be used to bid in auctions for Authority set slots as a Validator, or simply to pay for LP or Relay transactions on the State Chain.

Case 2: Adding Collateral to a Liquidity Position

Liquidity Providers use a similar process as described above, to add collateral to fund their liquidity positions in the Chainflip AMM. To pass in funds, Liquidity providers need to have an on-chain account for the Chainflip State Chain and refund addresses ready for each asset.

1. The liquidity provider creates a State Chain account by staking a nominal amount of FLIP using the Staking process in Case 1. This is used to pay for State Chain fees.
2. The LP then submits an extrinsic to the State Chain requesting a deposit address for each asset it wishes to deposit to use as collateral for their LP position. For example, a BTC-USDC pool would require the LP to submit two extrinsics requesting ingress addresses for BTC and USDC.
3. Once confirmed on the State Chain, the Liquidity provider locally derives deterministically generated ingress addresses for both BTC and USDC. This can be done easily by taking the primary multisig key generated by the current Authority Set keys and the confirmed index reserved using the request-ingress extrinsic. In the case of Bitcoin, the result is a Hierarchical Deterministic (HD) wallet address, and in the case of Ethereum for USDC, is an address derived using the CREATE2 function, both of which are totally unique to that liquidity provider, but linked to each of the asset’s vaults in the Chainflip Settlement Layer. See Ingress Addressing & Relayers for more information.
4. The Liquidity provider then sends funds using any wallet to each derived address. The transactions are in turn witnessed by the Validators, resulting in the credit of balances for each asset on the State Chain.
5. Finally, the Liquidity provider can submit another extrinsic to utilise the balance of funds available in their State Chain accounts to open a liquidity position, which will then include their funds in the virtual BTC-USDC liquidity pool in the ranges specified by the LP. Ranges can be easily updated by submitting follow up extrinsics to the State Chain.

Case 3: Conducting a Swap

Finally, and perhaps most importantly, a third type of ingress process is used by users to send funds to be swapped in the virtual Chainflip JIT AMM, although in this case, the requirement to have a State Chain account is circumvented.

1. A user requests a swap using a **Relayer**. A relayer can be run by anyone and is used to bypass the need for on-chain accounts. All the user needs to provide is the asset they have, the asset they want, and a **Payout Address** where they wish their funds to be sent to after their funds have been swapped. Relayers are covered in more detail in the Appendix. The Relayer submits this information to the State Chain as a **Swap Intent** extrinsic.

2. Once the swap intent is included in a State Chain block, the user can once again locally derive a **Swap Address** unique to their swap intent.

3. If they send funds of the correct asset to that address, the Authority Set will witness the deposit and automatically begin the process of swapping the asset through the event driven Chainflip JIT AMM.

It is important to note that the unique Swap Address derived by the user has a limited time to live. Because a Vault Rotation could occur at any time, it is essential that every time a new Swap is made, users generate a new Swap Intent to ensure that the unique deposit address generated is derived from the latest Vaults. *See Ingress Addressing and Relayers for more information.*

Egress: Sending Assets out of Chainflip

Getting funds into the protocol is one thing, but sending funds from it is the more complex part. Chainflip also has a generic processes for sending or issuing funds out of the protocol: Egress. These cases are:

1. Sending FLIP funds from the StakeManager contract when a Validator or other State Chain account holder wishes to **withdraw FLIP**;

2. Sending assets to a liquidity provider that is **withdrawing their collateral**, or;

3. Sending the output of a swap to a user's **Payout Address**.

As with ingress, the process for egress differs slightly from case to case, but shares a common process: **TSS (Threshold Signature Scheme) Signing**.

Threshold Signature Scheme (TSS) Signing Ceremonies

When an event on the State Chain occurs that signals to the Authority Set that a signature is needed, the Validators begin a ceremony. Wherever possible, outputs will be batched together to save on gas fees and to minimise the number of signing ceremonies required to send users their required funds.

TSS signing in Chainflip is conducted using the FROST multisig scheme, which relies on Schnorr signatures for fast and scalable multi-party-computation, allowing for a large set of signers. FROST allows Chainflip to secure all Vaults using the same Authority Set of 150 Validators, offering substantial benefits in terms of economic security and a simpler Vault management logic compared to other Cross-Chain Liquidity Networks.

The Advantages of Using FROST

In the original Chainflip Whitepaper, we described both the use of Schnorr based Threshold Signing Schemes[8] and GG20[9], the latter being what is currently deployed in THORChain. GG20, although able to be applied to almost any blockchain, constrains Validator set sizes due to scalability, making vault management systems more complex in order to maintain decentralisation. This has been seen in practice within the THORChain protocol design, where until very recently, the system forced individual Validators to hold user funds on their own hot-keys. This was done to avoid the computation and time cost of a GG20 signing round with the 36 other participants in a given key.

Instead, Chainflip employs the Flexible Round Optimized Schnorr Threshold (FROST) signing scheme[10], which to our knowledge, makes the Chainflip protocol the first FROST protocol user of this scheme within the wider blockchain ecosystem.

This scheme relies on EdDSA/Schnorr signatures to function, and thus is not supported by all blockchains. However, the few remaining chains which lack support for these signature types are largely unpopular Bitcoin forks that have not integrated the Taproot changes.

The advantages of using the FROST Scheme above GG20 become apparent when observing signing times in both benchmark testing and in production environments. FROST allows Chainflip to scale to a 100 of 150 system for all supported chains with signing times anticipated to be below 3 seconds in production based on current observations. This means Chainflip Validators can run on less expensive hardware while providing better security and simpler architecture overall than would otherwise be the case with GG20, making room for more supported chains.

Some chains also have relatively scalable multisignature systems built into their chain natively, which if leveraged correctly can increase the speed and efficiency of vault management operations further still. Chainflip may leverage these systems to integrate some chains.

Many more chains don't necessarily support EdDSA signature types natively, but do also have Turing-complete smart contract capabilities, which means that EdDSA aggregate key verification can be programmed into a contract directly. This applies to all EVM compatible chains and many more besides, making FROST widely applicable.

The FROST Signing Scheme in Chainflip

At vault creation, N selected parties perform a ceremony to create a single aggregate key for the vault or wallet. The protocol starts by having each party i generate a local key pair (x_i, Y_i), the public component of which (Y_i) is broadcast to all other parties, while the secret component (x_i) is split into N shares, with each party j confidentially receiving exactly one share ss_{ij} according to the Verifiable Shamir Secret Sharing scheme.

The latter provides two important properties:

1. Each party j can verify that each share ss_{ij} it received from party i is valid without knowing x_i;
2. Any combination of t shares ss_{ij} can be used to "reconstruct" the initial secret x_i. (The shares can also be used to perform some distributed computations on x_i, such as Schnorr signing, without actually reconstructing it.)

In case party j receives an invalid share from party i, it can challenge i, forcing it to broadcast the share, thus revealing it to other parties. Doing so ensures that either j receives a valid share in the end, or the ceremony is aborted and i gets reported by the majority. Note that only a small number of secret shares can be revealed this way, which does not compromise the security of the protocol.

After all secret shares have been distributed correctly, the resulting aggregate public key can be computed by any party as $\sum_{i \in 1..N} Y_i$. The aggregate private key, however, only exists implicitly as a combination of secret shares distributed between parties, and it is never explicitly computed. As an additional security measure, we check that the key shares have been correctly distributed by having all nodes perform a dummy signing ceremony with the new key before any funds can be sent to it.

Transaction Signing

The Validators that successfully generated an aggregate key with the above protocol can collaborate in a ceremony to sign a transaction, spending funds associated with that key on the target blockchain.

Recall that given a key pair (x, Y), a Schnorr signature over message m is constructed as follows. First, a secret unique nonce k is chosen with a cryptographically secure PRNG. Then commitment R is derived as $R = g^k$, which is hashed together with m to produce challenge $e = Hash(R||m||Y)$ (). The resulting signature is the pair $(e,\ s)$, where $s = k - xe$.

To sign a transaction with an aggregate key, a subset of t parties are selected to collaborate in generating the signature. In our distributed setting, parties generate secret nonce k in such a way that the commitment R is known to all parties, while k only exists implicitly as a combination of each party's local secret k_i and is never explicitly constructed.

Each party then computes challenge e and uses its secret key shares to generate a local signature s_i, which can be verified against the party's public key by others. Note that the property of Schnorr signatures allows combining all s_i into an aggregate signature s, which is valid with respect to the aggregate key, and a transaction signed this way can be submitted to the blockchain by any party.

Consistent Broadcast

The above protocols require all broadcasts to be consistent, that is, the messages that a given party sends to its peers during a given ceremony stage must all be the same. While this could be achieved by requiring parties to upload messages to some centralised location, this would clearly be sub-optimal for an otherwise decentralised system.

Instead, we achieve this by extending every regular broadcast stage with an extra round of communication in which the parties reveal all the messages they received in the stage prior. The idea is to only consider a message to be consistently broadcast if the majority of nodes received the same message.

If for any broadcasting party the peers can't come to consensus on the content of the message, the party is considered to have performed an inconsistent broadcast and gets reported, and the ceremony can be restarted without the dishonest party.

Note that for both protocols we manage to uphold an important invariant: the protocol is either successful, or we can detect and eliminate the parties responsible for failure, ensuring that we can always make progress.

This scheme forms the basis for all Egress Transactions from the Chainflip protocol. Coming back to the three generic processes for Egress, we have:

Case 1: Withdrawing Staked FLIP

When a Validator wishes to unstake or extract some of their rewards, or other users with a State Chain account wish to take out the FLIP locked on the State Chain to use on the Ethereum network, the user needs approval from the Authority Set.

1. The user submits an extrinsic to the State Chain requesting a withdrawal, which contains the balance of tokens they wish to withdraw and the Ethereum address they wish to be paid out to.
2. Upon the inclusion of the request in the next block, the Authority Set checks that the request is valid, and if so proceeds to commence a Signing Ceremony. The result of the Signing Ceremony is a **Withdrawal Certificate**, which is a valid signature from the aggregate key posted to the State Chain. The signed data includes the balance that can be withdrawn and the payout address.
3. The user takes their withdrawal certificate and passes it into a function in the StakeManager contract on Ethereum. This function will check the validity of the certificate by verifying the signature and certificate expiry, and if successful will start a 2 day timer (explained below) that the user must wait for until they can finally claim their FLIP.
4. After the delay window has passed, the user can call a final claim function on the StakeManager contract. The contract will send FLIP tokens to the account holder. If this claim function is not called after a defined period of time, the withdrawal certificate is no longer valid and will not be accepted by the smart contract. Instead, the user will have to repeat this process.
5. After the FLIP has been withdrawn from the contract, the Authority Set witnesses the egress transaction and records it as settled in the Accounting Layer.

The delay mechanism is installed to catch scenarios where falsely produced withdrawal certificates (such as in the unlikely event of a super majority attack or software exploit) can not be used to claim arbitrary amounts of FLIP. Additional governance mechanisms exist to prevent and mitigate the severity of attacks and exploits, *covered in more detail later in this paper.*

Case 2: Withdrawing Collateral from a Liquidity Position

Not only is the ability to withdraw liquidity essential, but it is expected to be fairly frequent in the JIT AMM. Liquidity Providers will often have to rebalance their asset float across the markets which they operate on to maintain a balanced and effective market making strategy.

1. The user submits an extrinsic to the State Chain which closes or reduces their liquidity position, leaving their collateral in a “free” state where it can be withdrawn.

2. The user submits a second extrinsic to the State Chain which signals the network to send the specified amount of assets to their refund address, which would have been specified when the account was created. Refund addresses can also be updated without creating a new account. This instruction becomes a **Pending Egress Transaction**, and is passed to the Settlement Layer to be settled.

3. The Authority Set commences a Signing Ceremony to create a valid transaction to send funds from the relevant vault to the user, deducting the gas free required from the balance owed to the user.

4. The signed transaction is broadcast to the relevant external chain by a nominated Validator, and the user receives their funds less the gas fee.

5. The Authority Set witnesses the egress transaction and records the transaction as settled in the Accounting Layer.

Case 3: Paying out users for Swaps

Finally, paying out users in the asset of their choice completes the functionality needed for the Chainflip protocol and represents the most common expected egress transaction.

1. Once a swap has been automatically executed on the State Chain, the assets to be paid out to the user are listed as a **Pending Egress Transaction**.

2. Using the Payout address specified in the User’s Swap Intent, and bundling the Pending Egress Transaction with others into a single batch transaction, the Authority Set conducts a Signing Ceremony to produce a valid transaction for that chain.

3. The signed transaction is broadcast to the relevant external chain by a nominated Validator, and the user receives their funds less their share of the gas fee.

4. The Authority Set witnesses the egress transaction and records the transaction as settled in the Accounting Layer.

These three cases capture all current forms of Egress transactions, though this could be extended in the future to facilitate more generalised cross-chain functionality or more complex interactions with other on-chain markets. Focusing on the specific use case of native cross-chain swaps, let’s next examine the AMM designed for the Chainflip protocol.

The Chainflip JIT AMM - A Novel Protocol for Capital Efficient Cross-Chain Swaps

With the primitives for depositing and withdrawing capital as well as sending and receiving arbitrary assets defined, we can now discuss the Chainflip JIT AMM at a higher level of abstraction.

The Chainflip JIT AMM exists as an event driven process that occurs on the Chainflip State Chain. This differentiates it from typical AMMs which rely on agent initiated smart contract interactions, and is instead automatically and deterministically executed by the Authority Set. This opens up the AMM design to a wide variety of new functionality enabled by a purpose built execution environment.

Chainflip's AMM design differs substantially from industry standards due to limitations introduced by the nature of cross-chain transfers. The Chainflip AMM protocol has several distinguishing features which radically alter the optimal liquidity provider strategy and offer significant capital efficiency and pricing accuracy advantages for users. We call this style of AMM a "JIT (Just In Time Liquidity) AMM."

There are few examples of similar trading products in use today, with Gnosis' CowSwap[11] being the only noteworthy example. However, the JIT AMM shares little in common with the "Coincidence of Wants" protocol designed by Gnosis, which does its best to avoid the use of Market Makers. On the contrary, the JIT AMM makes it far more compelling for Market Makers to participate, greatly increasing the likelihood of deeper and more reliable liquidity for users.

At a high level, the JIT AMM relies on Market Makers frontrunning each other to offer users the best possible price on their swaps. This is achieved by enabling rapid range order updates and by batching swaps together. It relies on a software driven trading strategy and is not expected to be favourable to passive liquidity providers, but should yield significant benefits for end users in terms of swap pricing and overall fees.

Rather than expose users to the risk of frontrunning and other MEV opportunities seen in typical single chain AMM environments, the JIT AMM attempts to flip frontrunning on its head, making it very easy for Market Makers to frontrun each other instead. By offering a fixed reward in the form of a liquidity fee, a known delta naturally incentivises Market Makers to offer the best price they can on any given batch of swaps, ensuring users receive close to global index prices in all but the most extreme scenarios.

The goal is to maximise capital efficiency and compensate for the downsides of cross-chain trading by offering a feature-set capable of higher efficiency swapping compared to sequentially executed AMMs like Uniswap and THORChain.

AMM Development Background

Uniswap V3 Introduced the concept of range orders into the AMM world, which brought about a number of improvements to the capital efficiency and user experience of many of Uniswap's most popular pools. Chainflip, as a cross-chain product, does not share the same execution environment assumptions as typical AMMs, and therefore additional features are required to prevent front-running which negatively impacts users. Unlike in a single chain environment, Chainflip must address the following issues:

- The protocol must wait several blocks to confirm external chain deposits, due to the risk of chain reorganisations external to Chainflip. As a decentralised and programmatic system, there is no way to manually reorder transactions if this occurs, and so several block confirmations are needed to operate safely. Given that there is a non-trivial delay to confirm incoming swap deposits, and they are on public chains, all market participants will know the swaps that will be executed before they occur. This leaves the protocol vulnerable to various forms of front-running.
- Furthermore, the confirmation times also cause pricing and arbitrage to be significantly delayed, resulting in prolonged price differences compared to market index prices. This can be observed on other AMM-based cross chain DEXes that have not mitigated this problem, where prices consistently deviate from the market rate by as much as 10%.

Finally, new proposed features of the AMM would be largely untested in the wild, as existing AMM designs have not been built with custom execution environments in mind and as such there are few examples of protocols with similar properties to derive learnings from.

There is also a unique opportunity to explore the possibilities of AMMs running on a custom execution environment. As Chainflip runs in its own independent execution environment, much of the swap execution can be automated and executed by the validator network without complex user interactions. This should be leveraged to the maximum extent, and new features should be designed to capitalise on this for the benefit of users.

Just In Time AMM Core Features

The JIT AMM offers several new features on top of the existing Uniswap v3 AMM design. These features mitigate the unique problems with cross chain swaps, and leverage the unique advantages afforded by a custom execution environment, with the three core features that drive the JIT AMM being:

- Faster-than-swap Range Order Updates - Range Orders can be updated before known swaps are executed, meaning Market Makers can actively respond to incoming trade flow.
- Swap Batching - Swaps are grouped together and executed periodically, cancelling out a lot of slippage, and rendering trade frontrunning unprofitable.
- Single-sided Range Limit Orders - Limit orders that can be placed by Liquidity Providers over any price range, but can not be executed against by other LPs.

These features are implemented directly in the Chainflip State Chain and accompanying software that Chainflip Validators run.

How JIT Works

The core concept that drives the JIT AMM design is to flip frontrunning on its head. Instead of users being front run by MEV seeking bots, the protocol naturally incentivises liquidity providers to front run each other to the benefit of the user. Several months after first publishing our protocol design of a JIT AMM, MEV seekers spotted examples of this in action on Uniswap v3. For trades of a large enough size, there are opportunities to front-run other LPs even on the expensive and generalised Ethereum blockchain, proving the viability of this strategy in the wild.

The fixed liquidity fee in each pool (between 5 and 30 bps in most pools) acts as a built-in delta for Market Makers to attempt to capture. The way Market Makers capture that delta is to move their liquidity position right up to the market price. The market price is dictated by the availability of instantaneous liquidity on separate primary and secondary markets (including centralised exchanges, derivatives markets, and any other liquidity sources, such as OTC desks) just before a swap is executed. By opening hedging positions or taking existing liquidity on other markets, Market Makers can capture the liquidity fee entirely if they take on more price risk than all other liquidity providers. Once the swap is executed, the Market Makers can update their range order again to rebalance their position ready for the next swap batch.

As long as a few Market Makers compete with each other for the delta, this protocol design should ensure that users performing swaps are always getting market pricing or better than market pricing at the time of swap execution, with reliable fees and minimal slippage, and that capital efficiency in this protocol should far exceed all other existing AMM designs depending on the level of competition between Market Makers.

An Example of JIT Swaps

Let's trace the path of a typical swap to examine how this works in practice. For this example, our hypothetical user will swap USDC (ERC20) for DOT (Native), and Market Makers A, B, and C will compete to win the liquidity fee from the trade. There is a 25bps liquidity fee on this pool.

- The user (A DOT buyer) generates a quote which creates a unique deposit address associated with their destination address and swap intentions for DOT. The user initiates the swap by depositing 10,000 USDC to their swap address.
- The Ethereum blockchain includes the USDC deposit in the next block. The Market Makers spot that the deposit has occurred and track other upcoming USDC deposits. There are also DOT sellers making deposits on the Polkadot chain. Although it will take several Ethereum and Polkadot blocks for the transactions to be recognized on the Chainflip State Chain, the Market Makers can watch both the Polkadot and Ethereum chains to calculate the overall direction of the trades in the next Chainflip batch ahead of execution.

- Chainflip requires 6 Ethereum blocks to consider a deposit transaction 'final.' It also processes swaps in 6 Ethereum block batches (or about 18 Polkadot blocks), so there is a minimum of a 6 block delay between swap deposits being made on the Ethereum blockchain and the swap being executed on Chainflip. This means Market Makers have about 90 seconds where they know exactly what the swaps will be, can source liquidity or calculate risk on other markets, and can adjust their range orders via a state chain transaction.
- While this is happening, the user is waiting between 6 and 12 Ethereum blocks for the batch to be executed. In this case, there are 10 trades totaling 280,000 worth of volume in the batch, however only 90,000 worth of that volume is buying DOT. Therefore, for everyone's trades in the batch to go through, overall the batch needs to sell about 190,000 worth of DOT. To capture up to 25bps of fees on the total volume of 280,000 (about $700), the Market Makers must now offer the best possible price on 190,000 worth of DOT within the 6 block delay window. This is where the name "Just In Time" comes into the picture, as they need to get liquidity sources and push forward some buy liquidity "Just In Time" to win the fees.
- By having additional capital float on other exchanges like FTX, Binance, and so on, Market Makers can use risk model calculations to determine their best possible price for the trade. Using this calculation, they update their single-sided range orders and move their USDC in the USDC-DOT pool right up to that price. In this case, Market Maker A moves their USDC range order to have 190,000 of USDC liquidity concentrated at 23.45 per DOT, whereas Market Maker B moves their 190,000 of USDC with a concentration at 23.47 per DOT.
- The Chainflip network executes the batch. Even though the overall direction is selling, even those users who are buying do not suffer unnecessary slippage as the Market Makers have sourced liquidity externally in order for the pool price to naturally gravitate close to the market index price. Market Maker B captures 100% of the liquidity fee, assuming they concentrated ahead of all other Market Makers. The Market Makers that didn't take the trade make no fees, but also experience zero impermanent loss (as their balances remain unchanged).
- Market Maker B, knowing that they have won the fee and how much DOT they just bought, goes ahead and sells 190,000 of DOT at 23.47 or higher on other markets. If managed correctly, the Market Maker just pulled in $700 or more of profit on one batch, and did so by taking on a mere 6 seconds of price risk. These opportunities occur every time a batch is executed, which occur as often as every couple of minutes for each pool.
- The Chainflip Validator network now sends the funds to the users, who receive funds based on a swap price that closely tracks the global market price. Our DOT buyer receives their funds directly to their native DOT wallet.
- Market Maker B might now make some withdrawals or deposits to their LP position to rebalance their portfolio to prepare to capture future batch opportunities, while Market Makers A and C are ready to take opportunities in batches in the immediate future. The next one is just 6 Ethereum blocks away.

This swap flow relies on Market Makers executing behaviour which is nearly identical to that of typical software driven OTC desks, but in an open and competitive environment. This strategy can easily be integrated into typical market making strategies as a method of generating trade flow without needing to build or expand a customer base.

Some Caveats to the Example

In reality, it would likely be rare that a single liquidity provider takes 100% of the liquidity fee, but rather one or two Market Makers taking the large majority of a given trade with inconsequential amounts filled by an assortment of other liquidity providers just due to the nature of AMM curves. In any case, the Market Makers will always know what trades they (and everyone else) just executed and will follow the same steps and principles. Furthermore, in the wider Chainflip protocol design, it is intended to have dozens of these pools operating simultaneously, all moving at slightly different speeds on the basis of the block and confirmation times of each chain. A BTC to ETH swap for example would not be facilitated in a single swap. Instead, a user's funds would automatically be routed through two pools, BTC-USDC and then USDC-ETH, which would involve two swap batches that both follow the same rules as described above.

Due to this arrangement, Market Makers must not only track future deposits into the pool in question, but also the expected path of deposits into other pools that will ultimately be routed into the pool in question in order to accurately predict batches. Routing everything through USDC does mean that users will be exposed to USD for a short window of time while they wait for their swaps to be routed, which may not be desirable in some circumstances, but is assessed to be a worthwhile tradeoff in order to minimise liquidity fragmentation, which would arguably be worse for users under normal market conditions. If particular routes within the protocol become popular enough, direct asset-to-asset pools (such as ETH-BTC) can be added through protocol upgrades to allow users to avoid this price exposure.

Lastly, with all market making strategies, there are degrees of complexity. Advanced risk and prediction modelling would certainly give competing Market Makers an edge over one another. At a basic level, a JIT bot could be written that would simply wait until the future batch is completely known, make an instantaneous assessment of liquidity on other order books, update the single-sided range order price, and then wait for the batch to confirm before market buying or selling on other order books. This way, the bot would never make trades on other books unless it knew for a fact that it had taken some flow on the JIT AMM, and wouldn't rely on any advanced modelling to function at its core. It would however be quite easy to outcompete this simple JIT bot with a more holistic market making strategy. It can therefore be expected that as the volume increases, natural competitive dynamics will drive the performance of Market Makers to constantly improve, leading to even better pricing for users.

Other Benefits & Tradeoffs of the JIT AMM

Using this swap flow has some other non-negligible benefits for users. Firstly, by grouping trades into blocks or even batches of blocks, frontrunning traders becomes nonsensical, as all traders in the batch get the same price. Furthermore, for a good majority of trades in normal market conditions, this liquidity strategy should totally neutralise effective slippage for the bulk of users.

There are however some tradeoffs which we must accept to achieve this. Namely, this protocol can not give users certainty about the ultimate outcome of their swap ahead of time. Until all trades are in, the final state of the batch can not be guaranteed, and even then until the range order updates are in, a final slippage/pricing calculation can not be made. This is exacerbated in multi-swap trade routes.

However, with the development of risk models and prediction models displayed on trade estimation tools on user front ends, it will be relatively straightforward to inform the users about the likely outcome of their trades given the intended size, current market state, and historical data.

On top of this, if a pool ever becomes very imbalanced because of large trades in one direction in a short window of time, users who are relying on the JIT model for accurate pricing might end up suffering. This is because all of the Market Makers would have been cleaned out on one side, and large amounts of passive liquidity are not generally expected to be prevalent on JIT AMMs to mitigate this problem. Market Makers also have to wait longer to rebalance a portfolio than normal AMMs, as there is an additional lag to confirm deposits and process withdrawals than other on-chain AMMs. That being said, rebalancing wouldn't take any longer than on a typical centralised exchange.

This could be mitigated again by displaying a warning to users when generating quotes if the front-end detects a current imbalance or significant deviation in the relevant pools from index prices. Better yet, automated systems can be implemented which delay the execution of batches until the system has been rebalanced.

The Extremes of Capital Efficiency

The TVL of the pools in a JIT AMM can not be compared to a typical liquidity pool, as the TVL of a pool actually represents as much as half of the maximum practical trade size - a kind of capital efficiency that pushes at the extremes of what is possible. If all LPs are executing an active strategy, a pool with popular assets that has a TVL of 10m could in theory tolerate a trade batch of up to around 3.5m - 4m in one direction with sub 2% price impact in most cases (depending on the trade pair and global liquidity state across all markets). This kind of capital efficiency can not be replicated effectively on any normal AMM. This price impact estimation is based on observing typical instantaneous liquidity on major pairs using tools such as cryptosor.info, but further reductions in price impact are possible.

However, it is not possible to have greater than 100% capital efficiency. Large deposits that exceed or heavily exhaust the available liquidity in the pool change the underlying game theory, and incentivise the Market Makers to collude rather than compete. If Market Makers know that they can't fill the order, and also know that the other Market Makers can't fill the order, Market Makers stand to gain much more if they all shift their range orders away from the market price to effectively buy the incoming deposit at a fraction of the global market price.

This isn't too different to what happens when liquidity pools on other AMMs accept huge deposits with non linear slippage, where exceeding the effective liquidity of the pool progressively degrades the effective price. However, because the Market Makers on the JIT AMM have time to respond to incoming deposits, there is greater room for exploitation. It only makes sense for the Market Makers to play this new game if liquidity on one side will definitely be exhausted in a given trade, but the consequences for the trader in this case are worse than if they had used a typical AMM.

For those large trade batches with significant direction changes, there is also a form of arbitrage that could mitigate the likelihood of pool exhaustion occurring. If a large deposit is detected, arbitrageurs could make counter deposits within the same block to be exposed to the price impact expected from the overall batch. This would effectively allow the arbitrageur to buy cheap or sell at a higher than index price, but the opportunities would only exist when large imbalanced batches are expected. Arbitrageurs following this strategy would help to reduce the impact of large trades on other traders in the batch, further improve capital efficiency, and would mean that the Market Makers would be able to handle even larger flows without exhausting liquidity.

Another potential feature that would help avoid exhausting liquidity pools and incentivising predatory Market Maker behaviour is to break up extremely large deposits automatically. By splitting large deposits into several of the upcoming batches, arbitrageurs and Market Makers would have more time to handle the trade. This is similar to how most OTC desks handle large orders on the backend in any case. The depositor should theoretically get similar prices as if they had used an OTC desk if this deposit splitting feature is implemented effectively. At this stage, we don't plan to implement this feature early on, but if proven to be important in the wild, a protocol upgrade can occur.

Constraining Liquidity Pools For Efficiency

The JIT AMM has a fixed number of pools. New pools can only be added by agreement of the Validators. As a standard practice, every new supported blockchain will have a liquidity pool on Chainflip which pairs its main native asset (ETH on Ethereum, SOL on Solana, etc.) with USD. In the case of our ecosystem, this will be collateralised in USDC issued on Ethereum to begin with. This design is intended to limit liquidity fragmentation and improve capital efficiency. Based on data from both centralised exchanges and leading DEXes, we have assessed that most Web3 users have stopped trading between token to token (such as ETH>BTC) and instead trade directly in and out of USD (ETH>USD, USD>BTC) . Not only that, but often the most efficient route between tokens on Uniwap v3 regularly routes swaps through USD regardless of the assets involved.

We assess this phenomenon to be as a result of a behaviour change in crypto market participants in recent years, where trading is mainly denominated in USD, as opposed to the BTC and ETH base pairs observed prior to 2018. Further to this, USD trading has always been the preferred base asset for market-neutral active Market Makers. Software driven trading strategies allow firms to offer traders very low slippage by holding positions across multiple markets simultaneously, made easier and cheaper through USD denominated pairs. Leveraging this fact for the benefit of the user, Chainflip only offers a USD pairing to all supported assets by default.

If particular routes prove to be very popular, greater efficiency can be achieved by selective fragmentation. For example, if lots of traders are conducting a ETH - USD - BTC route, it may make sense to also make a ETH/BTC pool so that the intermediary USD swap can be skipped, although we expect it will be difficult to make a case for this in the real world given the impact of fragmented liquidity on the ultimate swapping price.

The Network Fee

Although we don't intend to cover the Chainflip token economics extensively in the whitepaper, it is worth mentioning that it is essential that the FLIP token has a source of value capture. Without it, there is no long term guarantee that the collateral staked into Validators will be enough to maintain enough economic security to make the protocol viable.

To capture value for holders of the FLIP token, a 0.10% fee is collected from each swap in USD. This is done automatically at the first time a trade route goes through USD. For example, a USD to BTC trade would deduct it before the first swap is executed, whereas an ETH to USD to SOL trade would have the network fee deducted between the two swaps.

The USD is then periodically used to purchase FLIP automatically from the native FLIP/USD pool. This purchased FLIP is then destroyed, offsetting the emission of new FLIP that is used to pay rewards to Validators.

You can read more about the token economics and expected performance of this value capture system in the Chainflip Emissions & Incentives documentation page[12].

JIT AMM Development

The JIT AMM is under active development. We are constantly researching and monitoring the specific parameters and rules of the protocol to ensure that an optimal balance of pricing, liquidity, and efficiency is achieved, even as market dynamics impact the game theoretical implications of given strategies employed on the AMM. For the most up to date designs and specifications of the JIT AMM, refer to `https://docs.chainflip.io`

Ingress Addressing and Relayers

In this paper, we briefly covered how unique deposit addresses can be reserved for depositors of various kinds to the protocol. In this component analysis, we'll explore at a lower level how Ingress Addressing works and discuss Relayers in more detail.

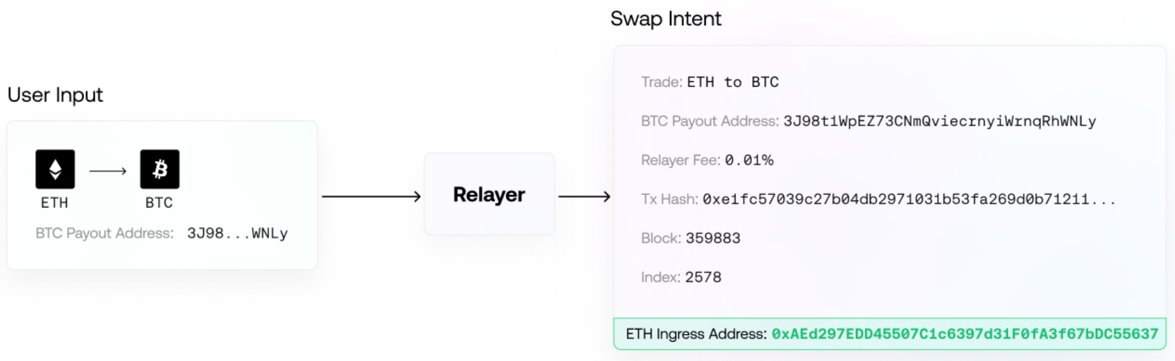
Ingress Addressing

Almost all contemporary blockchains have some sort of multi-sig compatible addressing system where anyone can derive a very large number valid wallet addresses that are controlled by a single public key. Given N number of addresses can be derived as f(pubkey, index), Ingress Addressing isn't really about the addresses at all, but rather reserving an index for the user.

We tie the index (and address derived from it) directly with a Swap Intent. A Swap Intent is essentially a request, not only to register the Payout Address and other trade details, but to reserve an index from which a unique Ingress Address can be derived. Similarly, when Liquidity Providers wish to add funds to their position, they also request an index on the State Chain to lock in their unique Deposit Address.

The method of deriving a unique deposit address based on an index varies based on the blockchain that the Swap Deposit is intended to be made on. Ethereum and its derivative chains, for example, rely on the CREATE2 function, Bitcoin uses its own Hierarchical Deterministic (HD) wallet system, and for Polkadot, another Substrate specific system. Each new blockchain type added will require developing abstractions to handle these varying multisignature systems.

This is really no different to the way that centralised exchanges handle user deposits, deriving unique deposit addresses and then sweeping the funds from those public addresses into the main wallet as required.



BTC Payout Address: 3J98...WNLy

Trade: ETH to BTC

BTC Payout Address: 3J98t1WpEZ73CNmQviecrnyiWrnqRhWNLy

Relayer Fee: 0.01%

Tx Hash: 0xe1fc57039c27b04db2971031b53fa269d0b71211...

Block: 359883

Index: 2578

ETH Ingress Address: 0xAEd297EDD45507C1c6397d31F0fA3f67bDC55637

The process for reserving and receiving an Ingress Address through a Swap Intent follows this process:

1. The user, through a Relayer (discussed below), has an extrinsic submitted to the State Chain which includes all provided Swap Intent data.
2. The Relayer returns the transaction hash to the user to verify the state of the Swap Intent
3. The user can verify the inclusion of the Swap Intent in a block through a secondary chain data source if needed
4. The user can also locally derive the Ingress Address based on the index/nonce that was included in the successfully submitted Swap Intent, and when ready, makes a deposit to that address to initiate the swap.

It should be noted that designers of front-end interfaces for the Chainflip protocol should be careful to offer users access to the transaction hash of Swap Intent transactions, and should avoid deriving the addresses server-side. By doing it this way, the integrity of the interface and the Relayer can be verified by the user through any secondary source of chain data, such as third party block explorers. This is an essential design standard, necessary to uphold the credibility of the protocol, and to promote trustless interactions in general.

Relayers

Any state chain node with a balance can become a Relayer. Their role is to construct and submit Swap Intent extrinsics to the blockchain for themselves, but mostly on behalf of end users. This role is essential in enabling the Chainflip protocol's default user experience: without an access point to submit a transaction to the state chain, users would have to be forced to include complex transaction metadata in their swap deposits to vaults. This would increase gas fees for users and would also necessitate the use of specialised wallets and a myriad of SDK integrations to overcome this issue.

A Relayer should offer an endpoint that takes an input of swap intent information, and returns the extrinsic hash once the Relayer has submitted it to the state chain. From there, the Swap Intent can be verified through any secondary source before a deposit to the resulting Swap Address is made.

Although the role of a Relayer is relatively straightforward, one of the major challenges with running a public endpoint to the State Chain is the threat of blockchain spam. To overcome this for the sake of network integrity, Relayers pay a small state chain transaction fee in FLIP every time they submit a Swap Intent extrinsic to the state chain. These FLIP transaction fees are burned.

This places the burden of spam prevention on the Relayer. If their endpoint is abused, they end up paying for the associated transaction fees. As a result, Relayers will need to design systems which protect against spam attacks. Private Relayers sidestep this issue, but public ones will need to ensure their web interface or API is appropriately protected, and that requests are appropriately filtered.

It is important that Relayer operators are provided with the tools to reduce the risk of spam attacks, and to mitigate the costs should they occur. It is assessed that the kinds of spam attacks possible through Relayers and the strategies to mitigate them are not unique to Chainflip, but exist across all public endpoints, websites, and internet services of all kinds. Thus, the mitigation techniques will broadly reflect the best practices in use across the internet today. What differentiates Chainflip Relayers is that there is a more significant direct financial cost associated with the requests made by potential spammers, but the tradeoff to achieve the desired user experience should be worth it.

Relayer Fees

This design then raises the question: why would anyone run a Relayer if they have to pay for third parties to use it, especially when those third parties may be malicious?

The answer lies in Relayer Fees. Relayer operators can choose to charge a fee for the use of their endpoint, and can be set at any value from 1 basis point to 1000 basis points. As a part of the Swap Intent extrinsic, Relayers include an otherwise empty relayer fee value which will instruct the network to charge an additional fee (similar to the Network Fee) which is deducted from any completed swap that they relayed the Swap Intent for. This fee is taken in USDC immediately after the Network Fee has been deducted.

This way, anyone wishing to integrate Chainflip into their wallet, web interface, or other Web3 product can benefit from getting their users trading on the protocol. The better the balance Relayer operators can strike between attracting users and managing Swap Intent transaction fees, the more profits they can expect to make.

Relayer fees are expected to be an important driver of protocol growth, filling the role of an equivalent affiliate scheme common in many centralised exchanges, and even some decentralised ones.

It also removes some of the incentives to fragment the cross chain swapping app by offering an easy way to collect fees based on the same protocol in the backend, but with different trading interfaces, target users, and product experiences.

With that said, many Relayers will probably not take a fee, particularly private Relayers set up by professional traders to rapidly interact with the network for software trading strategies.

Direct Vault Trades

Ingress Addressing can be bypassed through direct vault trades. This process requires the user to send funds directly to the Vault's primary wallet or contract address, but must include all of the required information to derive a Swap Intent in the transaction metadata in the external blockchain ingress transaction. Upon witnessing this direct transfer, Validators will parse the metadata and immediately include the trade in the Swap Queue.

It is expected that this direct addressing method will be used by professional traders to speed up their swap executions, bypassing Relayers and Swap Intent transactions in general. It could also be used by application developers who need to be able to send non-interactive transactions to vaults. By not requiring a call-and-response from the State Chain, and instead only needing to know the current address of the contract or vault, this enables some use cases that would otherwise not be possible.

With that said, we do not anticipate this method being popular amongst typical users, mainly due to the fact that it increases gas costs as a result of the extra metadata that has to be included on the external chain, and would require many different specific wallet integrations to be convenient.

Swap Intent & Vault Expiry: Handling Vault Rotations

As raised in other sections, one of the more complex aspects of processing Swaps is handling cases of Vault Rotation. Vault Contracts will be easier to manage since the contract address itself shouldn't change, but Native Wallet Vaults create a brand new wallet every time even one member of the Authority Set changes.

To illustrate the problem, consider a user that, through a Relayer, successfully submits a Swap Intent Extrinsic at 13:00 local time, attempting to sell DOT tokens. Shortly afterwards, at 13:14 local time, the latest auction is completed. By 13:27, all of the necessary key generation ceremonies and tests have been completed, and the Vaults have been rotated to the new Authority Set. By the time the user has located their wallet and initiated a transfer, it's 13:35 local time.

The user has just sent funds to a Vault controlled by an old Authority Set. If no process to handle this scenario is built, the user simply loses their deposit forever.

To overcome this problem, the Chainflip protocol forces members of an **Outgoing Authority Set** to remain online and maintain all Native Wallet Vaults for 2 weeks after a Vault Rotation process has been started. This way, if any deposits are made in this window of time, they can be witnessed and swept into the new Native Wallet Vault.

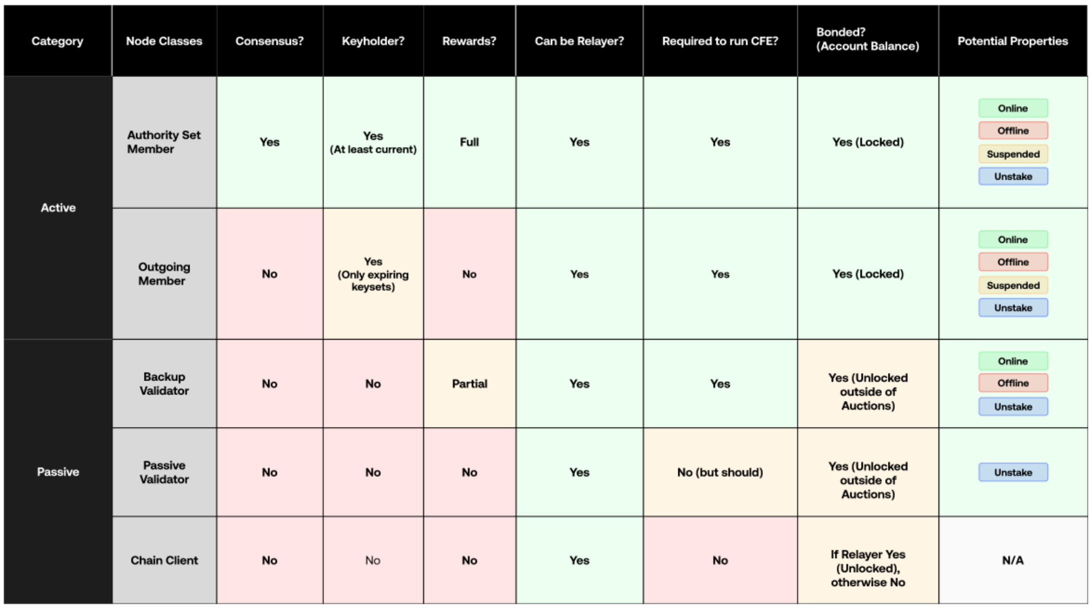
During this window of time, we say that the Vault is **Expiring**, and once that window of time has passed and all remaining funds in the Vault at that time are swept into the latest Vault, it has then **Expired**.

It should be noted that in most cases, the Authority Set will likely contain largely the same members in each successive epoch with only minor membership changes each auction. Therefore, most members of the Outgoing Authority Set will also be members of the current Authority Set, and won't be impacted by the added responsibility of maintaining an old Vault.

For those Validators which are Unstaking or have been outbid in the latest auction, they remain as an Active Validator not in the Authority Set, and are instead called an **Outgoing Member**. While the Vaults which they control have not yet expired, they remain bonded and liable for slashing, and must witness ingress transactions related to the expiring Vaults, and sign transactions to sweep funds out of them.

Chainflip Validators

Validators are Chainflip State Chain blockchain nodes. Anyone can run the Validator software and move their Validator through different states based on their actions in auctions.



Category	Node Classes	Consensus?	Keyholder?	Rewards?	Can be Relayer?	Required to run CFE?	Bonded? (Account Balance)	Potential Properties
Active	Authority Set Member	Yes	Yes (At least current)	Full	Yes	Yes	Yes (Locked)	Online, Offline, Suspended, Unstake
	Outgoing Member	No	Yes (Only expiring keysets)	No	Yes	Yes	Yes (Locked)	Online, Offline, Suspended, Unstake
Passive	Backup Validator	No	No	Partial	Yes	Yes	Yes (Unlocked outside of Auctions)	Online, Offline, Unstake
	Passive Validator	No	No	No	Yes	No (but should)	Yes (Unlocked outside of Auctions)	Unstake
	Chain Client	No	No	No	Yes	No	If Relayer Yes (Unlocked), otherwise No	N/A

The Authority Set

The Authority Set forms the basis of most of the functionality in the Chainflip protocol. The Authority Set, with some exceptions, is largely responsible for not only block production and consensus, but also witnessing, and operating a share of the current Vaults.

The size of the Validator Authority Set is limited to 150 due to the scalability limitations of communication and transaction signing protocols used within the Threshold Signature Schemes. Even as real world performance improves with optimisation, the additional performance capacity can be used to support more chains and more frequent signing ceremonies rather than more Validators.

To become active in the Authority Set, an operator bids by staking FLIP. While a minimum stake is required for Validators, the actual amount of FLIP the operator must stake to be selected is determined by the bidding process, where the top 150 bidding nodes with the greatest bids will form the next Validator Authority Set. Validators that are a part of the existing Authority Set automatically include their unclaimed rewards to top up bids for the next auction.

This market dynamic has several important properties. Validators that have been penalised for excessive downtime will have less FLIP staked compared to other active Validators, making them more likely to be outbid by new operators and must either increase their bid to maintain their position or face being replaced. This dynamic also allows the staking requirement to scale with protocol growth to better address collateralisation.

A new Authority Set will be selected under two possible circumstances:

1. The percentage of Validators in the current Authority Set that have gone offline exceeds a safety threshold, triggering an Emergency Rotation; or,
2. The lifetime of the current Authority Set runs out. By default, this will be set to 7 days, but could be changed through governance processes in the future.

In either case, the next Authority Set will be selected based on the highest bids that have been registered on the Chainflip network. A testing round is conducted to ensure Validators that have been selected are online at the time of selection, and continues until the superset of nodes only contains nodes that have successfully participated in this selection test.

Authority Set Auctions

There's a range of ways that protocols allocate positions of relative authority in consensus networks. DPoS[13], PoA[14], Eth2.0 style PoS[15], and simpler fixed staking requirements[16] are just a handful of examples. With the exception of the few collusion-prone blockchains that have a tiny Validator set, most permissionless networks don't have a cap on the number of Validators allowed on the network. Chainflip does, which places some constraints on the allocation system. The goals of the allocation system are as follows:

- Maximise the collateral locked in Validators.
- Encourage a relatively even distribution of collateral across the Validator network.
- Minimise active involvement in the auction process for adequately collateralised nodes (reducing manual intervention required to run a node).
- Encourage a relatively stable Authority Set without excluding new entrants
- Minimise gas costs associated with participating in the process.

To achieve this set of goals, we designed a minimally-interactive system to allow market dynamics to handle much of the process of allocating slots. It's been partly inspired by the winners of the 2020 Nobel Prize in Economics, Robert Wilson and Paul Milgrom[17], who through their studies in auction theory designed a system for equitably selling off a set of partially-fungible slots in a finite set called SMR (Simultaneous Multi-Round) Auctions. The most prominent example of this was when the FCC sold off broadcasting frequencies in the US to great effect[18] using this system.

Chainflip's auction process shares some concepts with SMR auctions but does have notable differences. Perhaps a better name for this style of auction could be a **Simultaneous Single-Round Open Dutch (SSOD) Auction**, as while participants are only participating in a single continuous round, they are incentivised to openly place the maximum bid they can as there is no downside risk to paying too much. Here are the rules for the auction process:

1. The auction starts halfway between the start and the expected end of the Epoch.
2. Any amount that is staked at the start of the auction will be automatically counted as a bid and remains locked for the duration of the auction (this includes existing Authorities and any rewards they earned before the auction started).
3. New bids that are placed either top up existing bids or enter a new candidate into the auction. Additional bids can be placed at any time during the auction, however once placed, they remain locked until the end of the auction.
4. The auction resolves at the end of the epoch, as follows:
   (a) The bounds for the number of bidders included in the next authority set are determined according to the previous set size and the maximum of 150 bidders. Where the previous set size is smaller than 150, the growth of the set size is limited to an additional 50
   (b) The token bond amount is defined as the minimum bid of all winning bids (Minimum Active Bid). The bond amount is the amount of tokens that are locked for each Authority Member and cannot be withdrawn for the duration of an Epoch.
5. The new proposed authority set is then required to cooperate to generate new aggregate threshold keys. If, during this key generation process, there are any nodes that fail to participate, these nodes are added to an exclusion list, and we begin including Backup Validators to try and fill the remaining slots. They will have less than the minimum bond, but will still be included in the new Authority Set. Only a maximum of the top one third of Backup Validators from the auction resolution will be included to replace offline bidders that would have otherwise become Authority members.
6. After the key generation ceremony is complete, the auction is deemed successful, and all failed bidders may withdraw their stakes. All Authorities may also withdraw any amount of tokens in excess of the now-locked bond from their stake.

This system is used because it avoids expensive on-chain last minute bidding wars with all participants trying to stake the minimum possible whilst winning slots. In this SSOD Auction system, each Validator should just bid the absolute maximum they can because at the end of the auction, they always have the option of withdrawing whatever they didn't need. It's only the bottom set of Validators that may need to quickly top up their bids before the end of auctions in order to protect their existing slots. It should also lead to a relatively even spread of collateral across all Authority Set Validators.

Chainflip Validator Auctions are covered in more detail in the Chainflip Validator Documentation page[19].

Backup Validators

One of the downsides with this auction design is that it leaves would-be Validators with insufficient capital empty handed after each auction cycle. This is bad, as these prospective Validators are earning no rewards and thus have no incentive to maintain the Validator node they have spent time and resources setting up. The same goes for Validators which have been outbid.

This means they'll be less likely to stick around and bid again in the next auction, as they still have to pay for and maintain their infrastructure in the meantime.

Further to this, because an Emergency Rotation is always possible, the design should make sure that there are always some extra nodes on standby ready to fill slots which are freed up by an Emergency Rotation scenario, or by otherwise successful bidders being offline during the Key Generation step in the process.

To resolve this, we introduce a passive class of Validator called 'Backup Validators.' By defining a set of nodes which are not included in the Authority Set, but are given a small reward for being around and staying alive, we ensure that getting outbid, being offline, or failing to bid enough to join the set could still be a profitable exercise for these operators. It also ensures that there's always a set of online nodes monitoring the state chain and ready to participate.

Instead of paying an equal reward to the Backup Validators, a fixed reward is distributed proportionally to Backup Validators based on their stake size. This is because we want to incentivise these Backup Validators to have the highest amount they can in case of an Emergency Rotation, in which the highest bidding backup nodes would be included first, and also to incentivise the nodes to hold onto their stakes and await the next auction.

For Backup Validators, we also allow bids to be placed outside of the normal auction cycle, and immediately reflect increased bids for Backup Validators in the rewards they are paid. This provides a direct and immediate incentive to stake as much as possible as soon as possible, both increasing total bidding and increasing the likelihood that these more Active and collateralised Backup Validators will be included in the next set.

The rules for Backup Validators are as follows:

- A fixed reward of FLIP (much less than the Authority Set reward) is allocated to the Backup Validators for a given Epoch.
- There is a limit on the number of Backup Validator slots - one third of the current Authority Set size. Any bidding node outside this limit at the end of the auction is treated as a Passive Validator.
- So long as Backup Validator remains alive and staked, rewards will be paid to it based on their stake, proportional to their share of staked FLIP in the total number of FLIP staked in Backup Validators.
- Backup Validators will never earn more than Authority Members.
- A Backup Validator will not earn rewards if it is offline. Backup Validators can come back online at any time and resume earning rewards.
- Backup Validators can unstake themselves at any time outside of the regular auction window just like the Authority Set. However, once the next auction begins, they must wait until the end of the auction to unstake, as their stake (including unclaimed rewards) will be automatically treated as a bid.

Reputation & Slashing

Validators are generally expected to be honest actors, but penalties must exist to ensure the reliable performance of Validator duties and for countering subversive behaviour. Validators hold shares in keys which can not be used to move protocol funds unless at least 100 of the 150 maximum Validators are online and can sign transactions. The protocol therefore needs to make sure sufficient penalties exist to incentivise consistent, secure, and high-performance Validators.

That being said, whilst it's necessary to discourage Validators from going offline at any point, there are legitimate reasons to do so which we should respect. Updating Validator software is one such legitimate reason, and penalising Validators for minor periods of being offline could also lead to negative results for the network.

Taking this all into account, Chainflip operates a unified slashing and reputation system that rolls all penalties into a time-based system with both **positive and negative reputation** scores, designed to balance the above considerations.

Validators start with their reputation at 0. Until they join the Authority Set, their reputation will stay at 0. Validators can be in one of three onchain states which affect their reputation score:

1. **Online** - The Validator is submitting **heartbeats** (covered below) and is not suspended - the Validator will earn rewards and will not be slashed while in this state regardless of their reputation score, and will gain reputation over time.
2. **Offline** - The Validator has failed to submit a heartbeat, which is due to some or all of the Validator's critical functions failing. Reputation will decrease over time while in the offline state.
3. **Suspended** - The Validator has been given a time-penalty for failing to complete a process in time or for doing something suspicious or incorrectly. The Validator will lose as much reputation as if they had been offline for the duration of the penalty.

If a Validator has a negative reputation and is offline or suspended, they will be **periodically slashed** until they achieve the online state again. The more negative the reputation, the greater the rate of slashing.

Heartbeats & Positive Reputation

Reputation will be accrued by a Validator as long as they are considered "online" on the State Chain. It makes sense to cap the amount of uptime credit available to any single Validator in order to prevent some strange scenario where a long-running Validator can be offline for many days at a time without suffering punishment. Chainflip has an initial cap of 48 hours on uptime credit, which should be more than enough time for any honest Validator to debug issues with their setup. This cap may be reduced in the future.

Being online is the only way to accrue reputation. If a Validator has a negative balance but is back online, they will start being slashed as soon as they are offline or suspended again. This is a strong incentive to fix issues and be reliable long before a reputation balance goes negative.

The way Chainflip ascertains whether a Validator is online or not is through **Heartbeat Extrinsics**. At regular intervals, Validators must submit a special transaction to the State Chain, simply proving that their hardware is connected and active, though it doesn't prove their overall performance.

If a Validator doesn't submit a valid heartbeat before the interval deadline occurs on-chain, they are automatically moved into the Offline state and will start losing reputation. The Validator will be moved back into the Online state at the next interval deadline that they submit a valid heartbeat for.

Suspension Conditions

For now, Validators can be suspended for two reasons, both relating to the Egress process:

1. **Failing to Participate in a Signing Ceremony:** During a signing ceremony, it is possible that a Validator fails to sign or communicate messages before the ceremony times out. This has the impact of requiring the entire signing ceremony to be conducted again, costing users valuable time. After a failed ceremony, the online Validators vote through consensus on Validators which misbehaved or failed in the signing ceremony, knocking them into the suspended state. The ceremony is then attempted again with a new set.

2. **Failing to Broadcast:** Account-based chains require our Validators to have funded accounts in order to broadcast transactions. The process of generating an output for these chains also includes a process by which the broadcaster is selected. The signed transaction to be broadcasted should be saved on the State Chain. Failure by a Validator to submit this signed transaction indicates a failure to broadcast the transaction to the external chain. This will require a new proposer to be selected by the State Chain, incurring a slight delay. The initially-proposed broadcast Validator will then be suspended.

As Chainflip is developed, more advanced suspension conditions can be introduced to better incentivise Validators to be performant. An example would be, "consistent failure to witness", where a Validator does not submit a witness extrinsic for an otherwise confirmed transaction after a timeout threshold. This is considered future work, as it would be trivial to dodge penalties by simply copying witness transactions without implementing a commit-reveal scheme. We don't consider this to be immediately necessary, but as the number of required chain clients grows, Validators may be more inclined to try and avoid maintaining so many connections, thus possibly necessitating implementing the scheme to ensure secure witnessing.

In the suspended state:

- The Validator will not be selected as a signer or a broadcaster,
- The Validator is liable to be slashed, if this state is coupled with a negative reputation value,
- The Validator is still expected to participate in witnessing and submitting heartbeats.

Validators, once the required time has passed, can have their suspended state lifted by submitting a valid heartbeat. If this does not happen by the end of the heartbeat interval, the Validator will be downgraded to offline, meaning they'll have to wait for the next heartbeat interval before they get the chance to return to an online state.

Tuning the time-penalties and exact numbers in regards to how reputation affects slashing is important. It is impossible to know exactly why a Validator triggered a suspension condition. In otherwise performant Validators, a situation could arise where they may have just gone offline before they were called upon, and therefore a suspension occured. In this situation however, the Validator should not experience any major difference in reputation. Similarly, Validators which just submit heartbeats in an attempt to evade higher server costs will get repeatedly suspended and then slashed anyway. Finding the exact parameters to strike a balance between the different cases is an ongoing process.

Reputation Rollover

Reputation which has been accrued by a Validator should roll over into the next epoch if they make it to the next Authority Set. This allows them to drop out of the Authority Set and then re-enter in a future epoch with the same amount of reputation. It should be noted that the reputation is not taken into account during the Validator rotation process. Instead, any Validators that are offline (as defined by the State Chain), at the time of an auction being completed, will not be considered for selection in the new Authority Set.

Backup Validator Reputation

The reputation and slashing system only applies to active Validators - that is, Authority Set members and outgoing members. Backup Validators don't have a reputation and instead are simply not rewarded when offline, and can never be slashed. Their online state is tracked using heartbeat submissions.

Governance Processes

The Chainflip protocol will not be set in stone. A governance process must exist in order to upgrade the network to support new blockchains, new pools, update fees, and adjust rewards as needed. Further, governance processes can be implemented which could significantly reduce the potential impact of unexpected events such as supermajority attacks, ransom attacks, and software exploits.

However, in order to preserve the decentralised nature of the protocol, the following principles must be observed:

1. Governance processes in a decentralised protocol should not be able to be unilaterally executed by any one party.
2. An honest Validator network should be able to expel any nominated governance body.
3. Most importantly, effective control over user funds should never be given to any single party. Effective control means the ability to unilaterally control the flow of funds, or to permanently block access to funds.

In this section, we present a range of governance features of the Chainflip protocol designed to upgrade and protect it.

The Governance Keys

The Governance Keys are arbitrary keysets that are nominated implicitly by the Validator network. The keys themselves can be any multisignature scheme that is compatible with the governance pallet that Chainflip Validators run. What this means is that the Governance Keys could be managed by any Threshold Signature Scheme, standard n of m scheme, or even a single key (though this would likely never be voted for). Who is behind said keys is a social matter - DAOs, councils, companies or even single individuals could be elected as a governance keyholder with enough backing, though we will refer to them generally as “Councils.” A supermajority of Validators can at any time propose to elect a new Council.

There are two Governance Keys in the Chainflip protocol, each held by different Councils and used for different purposes.

1. **The Governance Council:** This key holds many responsibilities, and is intended to be held by the primary developers of the protocol, or a body that represents them. Outside of managing the upgrades process, the Governance Council can activate essential governance processes which aim to protect the protocol from attacks, and software exploits. However, these security features of the Chainflip protocol often require an additional signature from the Community Council to enable the full governance capabilities built into the protocol.
2. **The Community Council:** This key acts as a check-and-balance for the Governance Council. Intended to be held by a body of community members separate to the primary developers of the protocol, this key has no special abilities by itself, but it is required to allow the Governance Council to use the more powerful security and governance features of the protocol. If used correctly, this key prevents the Governance Council holders from being a threat to the protocol, or exerting unilateral and effective control over the protocol through high-stakes governance actions.

The reason the role of check-and-balance isn’t simply left to the Validators by default is because there may be scenarios where they aren’t the best arbiters of community interests. This is especially true in the unlikely cases where some or all Validator keys have been compromised by hostile attackers. Further to this, Validators make up the heart of the protocol, but may not share the same views as users, liquidity providers, and developers that are also essential to any Web3 protocol’s long term success. By forcing Validators to nominate a distinct and smaller governing body, the network can enact governance votes faster and with fewer conflicts of interest.

The keys are automatically deployed to Smart Contract Vaults and the StakeManager contract by the Validator network, though a cooldown period is enforced to mitigate against attacks involving software exploits.

Voting in New Keys

Once the protocol is up and running, the Governance Council and the Community Council can be rotated only through a token-weighted vote on the state chain. Anyone with a token balance on the State chain can vote in proposals, even those currently staked into Validators. The minimum threshold for a successful proposal is a $\frac{2}{3} + 1$ super majority.

Anyone can create a new governance process to vote on the replacement of either one of the keys on the State Chain at any time. Voters must opt in to any key rotation proposal in order for a vote to be counted. The proposals last for seven days, and the token balances are only counted at the end of the proposal timeframe. Anyone that is actively unstaking or undergoing claims at the end of the vote will not have their tokens counted towards the result.

In order to mitigate against a scenario where a hostile supermajority has overridden this process in an attempt to use the security features against the protocol and its users, a cooldown period is required. By enforcing State Chain rules which prevent the rotation of both governance keys for a fourteen day period, a hostile supermajority will have to wait a week before the new key is installed and usable.

Liquidity providers should therefore receive as much as two weeks warning that the protocol is being taken over by a new party that may or may not be connected to the existing users and developers of the protocol. From there, they can decide if they wish to keep their assets in the system, or withdraw before any actions can be conducted by the new keyholders. Similarly, validators may wish to unstake during this cooldown window if they decide they do not support the holders of the new governance keys.

Thus, any party can become a Governance keyholder, but it requires not only support from a supermajority of Validators and other token holders, but also backing from the liquidity providers in place.

Runtime Upgrades & Hard Forks

The Runtime Upgrade process, which is enabled by the Substrate blockchain framework, changes the default nature of protocol upgrades from the traditional forking system, one which relies on manual binary upgrades, into one whereby the Governance Council has the ability to upload changes to the blockchain code directly to each individual node, and at a specified blockheight, have the responding nodes switch over to the new protocol rules seamlessly. This is a great feature to enable fast and lower-friction development, but doesn't work perfectly in Chainflip.

Due to the Chainflip Engine, a sidecar binary that runs outside of the State Chain runtime, Chainflip Validators won't always be able to be seamlessly upgraded in this way. If any changes in a runtime upgrade also require a change in the Chainflip Engine, Validators that have not updated this secondary binary may be suspended by the other Validators when they can't meet the newly required functionality specified in the runtime upgrade.

Traditional forks are also still possible using the Grandpa finality module that ships with Substrate, although we suspect that the governance processes designed for the protocol will be a more appropriate method for both upgrades and for making major project direction changes amongst competing groups of participants by voting on new governance keys.

Security Considerations & Features

Decentralisation presents a challenging array of security considerations that centralised exchanges simply don't have to deal with. Previous attempts at secure cross-chain systems have proven to be challenging in the wider ecosystem, with losses from security incidents in the cross-chain sector easily reaching into the hundreds of millions of dollars. We have thought long and hard about Chainflip's protocol security, and take it very seriously. Here, we'll explore what some of the security properties of the Chainflip protocol are, the range of security risks we have identified, the assumptions that we've made, and what the features designed to mitigate those risks are.

Potential Protocol Risks

No security profile can be built without a firm understanding of the risks which the protocol is exposed to. The surface area for faults and exploits with bad outcomes for protocol participants is not infinite, but it is larger than most blockchain projects. To begin, let's enumerate where things could go wrong:

1. Liquidity could be lost or stolen through the AMM/State Chain logic - LPs and swappers could experience potentially serious or even complete losses if the AMM logic or other related accounting systems within the blockchain fail or are exploited. By undermining the intended logic of the Accounting Layer, Validators could distribute funds incorrectly, leading to incorrect payouts. Attackers, if such exploits are discovered, could use them to drain funds into their own wallets. This type of security issue has happened in THORChain.
2. Liquidity could be lost or stolen through the vault management system - LPs and swappers who have deposited funds and are waiting to be paid out could experience potentially serious or even complete losses if the Chainflip engine is compromised or exploited in this way. Close attention needs to be paid to TSS ceremonies, broadcasting logic, and key rotation processes in the vault management system.
3. Validator private keys could be compromised, seriously undermining the security of the system as a whole if enough keys are compromised. Superminority and supermajority attacks could occur if this happens at a large enough scale. Individual Validators could also have all of their FLIP stolen if their Validator key is compromised.
4. Validator private keys could be lost, which if it occurs could result in the liquidity held by Validators being unable to be moved, potentially forever if enough keys can not be recovered.
5. The FLIP token market could collapse if the code which handles emission, rewards, and claims is compromised or exploited.
6. External chains could be attacked or experience major reorgs which would likely break the Chainflip Accounting Layer if incoming transactions that have been witnessed are later rolled back on the external chain

Protocol Level Security Features

Many of the above risk areas can largely be addressed by enforcing good engineering processes, a comprehensive integration and unit testing suite, comprehensive internal and external audits, and offering significant bounties to penetration testers. However, any complex software system that assumes that everything will work as intended, even if all of the correct precautions have been taken, has failed to develop a comprehensive security plan.

Redundancy is a key security principle and especially when applied in complex distributed systems with many moving parts like the Chainflip protocol. Our unique security features are designed to offer redundancy during many potential security incidents.

State Chain Safe Mode

As all of the above risks will impact the State Chain, and in fact most of the risks could arise from State Chain code, a system to pause all chain activity is essential in the case of a security incident.

The Safe Mode is a chainstate where Validators temporarily agree to halt most core functions of the protocol. Blocks will continue to be produced, so that key governance extrinsics and events can be processed, but other functionality is heavily reduced.

In Safe Mode:

- Witness extrinsics from the block at which the Safe Mode was activated are not accepted. Any ingress events that are not finalised will have to wait until after Safe Mode is deactivated to be rescanned and processed.
- Auctions, and vault rotations are no longer automatically triggered.
- No Egress transactions are processed (No swap payouts, no LP withdrawals, and no new staking claims will be processed).
- Supply sync transactions are suspended.

Validators can choose to opt-in or opt-out of Safe Mode delegation. This is done during the registration process and can be changed by submitting an extrinsic to the chain at any time. Validators which opt-in to Safe Mode delegation defer their actions to that of the Governance Council, meaning if the Governance Council submits a "safe mode request" governance extrinsic to the chain, those validators automatically count as having voted to enter Safe Mode.

A threshold of only 50% of the authority set needs to vote in favour of Safe Mode to activate it. Turning it on gives Validators, Developers, and the Community time to analyse security incidents in a reduced risk state. During Safe Mode, runtime upgrades and other fixes can be deployed to the network, allowing for a measured recovery process.

Once the incident is resolved, the network can exit Safe Mode by a supermajority vote on a governance extrinsic that can be submitted by anyone. Once again, opted-in Validators delegate their vote to the governance key.

StakeManager Protection

The StakeManager is an Ethereum smart contract that allows holders of FLIP, an ERC20 token, to put up collateral on the Chainflip State Chain. This collateral is credited on the State Chain as an account balance which can be used to run Validators, Relayers, or perform active liquidity management activities. The StakeManager contract also processes claims to send unlocked ERC20 tokens back to account holders after receiving a valid withdrawal certificate (a TSS 101 of 150 signature from the current Authority Set). The StakeManager can call the FLIP Token contract to mint more tokens if required as well.

There are quite a few different ways this could be exploited. Namely, risks 1, 2, 3 and 5 are all relevant and could all cause losses if the StakeManager contract is involved. To provide a layer of protection against risks 1, 2, 3, and 5, the StakeManager comes with a feature which adds a 2 day delay before a user can actually claim tokens (executing a claim) from the contract after submitting a withdrawal certificate (registering a claim) to the smart contract. The delay itself changes little, but in combination with its second feature, it can play a major role in providing redundancy against attacks and losses related to bridging FLIP.

The second feature allows the Governance key to suspend the StakeManager from being able to execute claims, or register new ones. If an issue where fraudulent or incorrect registered claims are detected within the delay window, negative impacts from FLIP bridging related incidents can be negated.

That being said, freezing claims indefinitely is not a solution. Any registered false claims would be able to be executed when the StakeManager is unfrozen again if claims did not expire after some period of time. As the StakeManager enforces an expiry time, which is set at the time of registering the claim, claim executions can not be processed by the StakeManager contract after 3 times the length of the claim delay. In other words, claims expire after 144 hours.

This means that in a security incident where the StakeManager is frozen, a graceful recovery is possible if underlying issues on the state chain can be resolved and the StakeManager is unfrozen after all registered claims have expired, which should be within 144 hours.

During this time, the state chain and vaults could continue operating, unless they too have been affected by the incident, in which case Safe Mode is likely to also be active.

NOTE: The current iteration of the StakeManager contract features a governance withdrawal function in tandem with the community key. It is unclear if this is necessary or desirable going forward, and its removal is slated for discussion.

Contract Vault Protection

Risks 1, 2, 3, 4, and 6 apply to the Vaults, and the losses that could be experienced due to these risks would impact liquidity providers, who could lose all of their funds, as well as swappers who are waiting for a payout.

Smart Contract Vaults also come with similar capabilities as the StakeManager, however unlike the StakeManager, there is no delay on payouts from the vaults like there is on claims in the StakeManager. This is necessary to ensure fast swapping times for users, but it does make the protections weaker than in the StakeManager contract.

All smart contract based vaults can be frozen by the Governance Council. Upon witnessing this function being called, the Validators will no longer be able to process egress transactions from that Contract Vault. It is assumed that Safe Mode may already be active at this time in most scenarios, so this could already be the case, but does add an additional layer of protection for the large pools of liquid funds held in these vaults.

While frozen, the Community Council plays a very significant role. If it is assessed that the contract or the network could be irreversibly compromised, or that the aggregate key has been lost or corrupted for good, a final option exists to ensure that funds remaining in the Contract Vaults can be recovered and returned to users and liquidity providers.

Within the Contract Vaults, the Community Council can call a function which authorises the Governance Council to withdraw all funds in the vault. This check and balance ensures that the Governance key never has unilateral control of the funds and must first both freeze the vault contract and then achieve approval from the community key before this extreme recovery method can be used.

This protection feature is considered very important considering that it is expected that over 80% of all assets held on the AMM are likely to be held in Contract vaults, as opposed to Native Wallet vaults.

Security Ratio

Although assessed to be incredibly unlikely, even without this security feature, to prevent financially motivated attackers from benefiting from a collusion attempt the Security Ratio is enforced by the validator network. As the chainstate is updated with ingress, egress, and trade information, it is possible to deterministically calculate the value of assets controlled by the validator network as well as the value of FLIP staked in validators.

Because only $\frac{2}{3} + 1$ of the validators are required to conduct a supermajority attack, the total value stored by the validators should not exceed the locked collateral of $\frac{2}{3} + 1$ of the validator network. This is the security ratio, where:

$$\text{Security Ratio} = (\text{Liquidity TVL} : \text{Collateral}) \times (2/3 + 1)$$

For a variety of reasons, it is assessed that exceeding this ratio presents no immediate or significant danger, unless the ratio climbs well beyond 1:1. For example, a ratio of 2:1 would imply that the value of the stored TVL would yield a theoretical maximum attack profit of 100% if a perfect supermajority attack was conducted. However, it is not clear that a one off 100% yield would be enough to justify executing a supermajority attack, given the opportunity cost of future validator returns, collateral appreciation, the risk of failure, complete loss, and criminal prosecution. The practical risk should be continuously assessed and the security ratio reviewed if it becomes a limiting factor in scaling the protocol.

In any case, the Security Ratio of 1:1 will be enforced by the validator network to prevent buildups of collateral not being fully utilised.

When the security ratio approaches or exceeds 1:1, the Validators will be required to automatically purge liquidity held by liquidity providers. Given that refund addresses have already been provided, it is possible to initiate an automatic egress return transaction for any liquidity provider at any time.

When the conditions for purging have been met, Validators will deterministically select liquidity used the least over the last 30 days. Any liquidity not yet deployed in pools would be the first to be purged. Secondly, liquidity held in unconcentrated ranges would be purged next, and so on, until the Security Ratio is reduced to below 1:1.

Deposits to liquidity positions will also be blocked when the ratio reaches 1:1. To prevent disruption to active liquidity strategies in the JIT AMM, only the least active liquidity providers and new liquidity providers will be blocked from making deposits until the ratio drops below 1:1 again. If the ratio exceeds 1.3:1, then all deposits will be blocked.

Given it is possible that large swaps by users could push the ratio above any of these thresholds temporarily, the security ratio is calculated using rolling averages at regular time intervals rather than instantaneous calculations. This also helps reduce the frequency of on-chain calculations required.

Supermajority Attacks

A hypothetical existential threat exists in the Chainflip protocol: the Supermajority Attack. Chainflip requires a signature from 101 of the 150 Validators to produce valid transactions to move funds on external chains. If a malicious supermajority were to exist, they could easily steal the funds in all vaults. Likewise, a supermajority would be required to alter State Chain history, confirm false witness transactions, or other potential actions which could result in a loss of protocol funds.

The Chainflip protocol guarantees that as long as an honest **superminority** (a blocking vote of 50 nodes) exists in the Authority Set, and the protocol has not malfunctioned, all witnessing, State Chain execution, and egress transactions on other chains will be executed deterministically, with no scope for falsification.

If a **supermajority** of the Authority Set colludes or has their keys compromised, then it should be assumed that all funds could be stolen, and complete disruption, takeover or failure of the network is possible. Chainflip provides no security guarantees in a supermajority attack situation, and in fact nor does any other credibly decentralised blockchain network of any kind. Correctly designed economic security systems are intended to prevent supermajority attacks, but the possibility of one can never be totally ruled out, and so is treated as an accepted risk, though we strongly believe it will never occur.

The risk of collusion is mitigated mostly through the enforcement of the Security Ratio, though without guaranteeing complete protection. It will always be possible in decentralised networks that if enough irrational actors exist, collusion can occur. However, we assume that there will always be a rational superminority in the Authority Set. Extensive reasoning on this subject is possible, though we don't think it's worth discussing here.

Another vector for supermajority attack is a wide scale validator key compromise event. Using modern server architecture however, it shouldn't be possible to compromise over 100 individual servers. Validator operators are also incentivised to secure their keys, as exposing them could result in all of their own staked FLIP being stolen. However, maintaining secure Validator nodes is essential, and software exploits (not just in Chainflip software, but also the operating system and other packages running on the machine) which allow for remote code execution or root access could potentially expose the keys of multiple Validators if they share the same exploitable architecture, which could potentially lead to a supermajority attack.

That being said, aside from setting sensible defaults for the software that is shipped and the support that is given, Validator server security isn't really protocol design, and therefore must be considered another accepted risk. If a common exploit that could expose private keys existed across 100 servers running Chainflip Validators, such as a zero-day exploit in a common linux distribution, it's likely that the exploit could be used to target much bigger and easier targets, and would likely result in mass security breaches across dozens of high-value software protocols globally. Aside from encouraging Validator operators to keep up to date with security patches, there's not a lot Chainflip developers can do about that, and so it's an accepted risk that at least a superminority of Validators will remain both honest and secure.

Future Work: “Hub and Spoke” Cross-chain Liquidity Network of Networks

This paper has covered the core protocol in detail, but it has also alluded to the idea that there are limits on the number of assets that can be supported natively by one cross-chain liquidity network, such as the main Chainflip network. There are three limiting factors that constrain the maximum number of blockchains that can be supported:

1. Complexity - As more blockchains are added, Validators need to add more light client connections to remain in sync and witness events on each of those blockchains. Maintaining dozens of client connections to maintain their Online status is expected to become increasingly challenging, particularly as smaller and less secure chains are added to the network. There are also more transaction types and more advanced logic required to handle these new chains, increasing complexity, which inevitably increases the security risk surface area.
2. Economic Security - Another limiting factor is the collateral put up by validators to secure the liquidity in the system. If the security ratio is reached, no further liquidity can be stored in this current design. We have devised ways to handle this and safely extend the liquidity capacity of the network, but those solutions add considerable complexity to the vault management system. As more chains are added, more liquidity is required to support each of the new chains, using up the limited amount of collateral that can be in the system at any given time.
3. Signing Scheme Scalability - As more chains are added, the frequency of expected signing ceremonies required to be performed by the Validators increases, thus also increasing the hardware and bandwidth requirements of Validator operators. This is not expected to be a problem in and of itself, but a system where a Chainflip Validator needs to be run on multiple machines instead of a single server instance would not be desirable.

To overcome these challenges, one potential solution is to create ‘subnetworks’ - secondary instances of the Chainflip protocol that are collateralised separately to the main network, support a different range of assets, but still share a common USD pairing. This way, the USD can be routed between the main Chainflip network and the subnetworks as a means of completing swaps between assets supported by the different networks, in the same way that swaps are all done through USD on the main network anyway.

There are two possible ways of achieving this:

1. Where the main and subnetworks treat each other as separate and use the native swapping functions built into the protocol to achieve the swaps. This would keep the networks totally separate in the case of a security incident, but it would mean that swaps would be more expensive to execute if the route crosses networks, and would require an on-chain USD transfer from network to network for each swap between the networks.
2. Where the main and subnetworks have a shared security model which allows for the virtual routing of USD between them, avoiding the need for on-chain transfers every time swaps occur, but also introducing more security risks by exposing both networks to each other’s assets.

It is not yet known what a sustainable number of assets on the main network is. Time will tell, but it will likely become self-evident to the protocol developers and community when a subnetwork is required. In any case, this future work presents the opportunity to extend the functionality of the Chainflip product to support as many blockchains and assets as the centralised exchanges are capable of.

Conclusion

Chainflip offers a protocol for native cross-chain swaps that brings the average user experience of decentralised on-chain trading much closer to that of centralised exchange offerings, while maximising the benefits of decentralisation for the benefit of the user.

Given its generalised nature, the protocol can be extended to support almost any blockchain network, and offer users, protocols, and product access to markets that would otherwise only be reachable through centralised offerings.

It offers extreme capital efficiency possibilities for Market Makers, greatly improving the average user experience, whilst also being less reliant on large amounts of liquidity in order to function.

We hope that these qualities will drive forward the Chainflip protocol as the best chance of replacing centralised exchanges as the primary method of spot trading within the Web3 industry.

References

[1] Sergio Demian Lerner (2012) P2PTradeX: P2P Trading between cryptocurrencies. Available at: https://bitcointalk.org/index.php?topic=91843.0.

[2] @hagaetc (2022) DEX metrics. Available at: https://dune.com/hagaetc/dex-metrics.

[3] @Trayvox (2022) Layer Zero Thread. Available at: https://twitter.com/trayvox/status/1508734174705987586

[4] Prabhjote Gill (2022) One of the largest cryptocurrency swapping platforms just lost 1.3 million as users failed to update approvals. Available at: https://www.businessinsider.in/investment/news/one-of-the-largest-cryptocurrency-swapping-platforms-just-lost-1-3-million-as-users-failed-to-update-approvals/articleshow/88992186.cms

[5] Rekt.news (2022) Wormhole REKT. Available at: https://rekt.news/wormhole-rekt/

[6] Rekt.news (2021) Polynetwork REKT. Available at: https://rekt.news/polynetwork-rekt/

[7] vitalik Buterin (2022) We are the EF's Research Team. Available at: https://old.reddit.com/r/ethereum/comments/rwojtk/ama-we-are-the-efs-research-team-pt-7-07-january/hrngyk8/

[8] Stinson, D.R. and Strobl, R., 2001, July. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Australasian Conference on Information Security and Privacy. Springer, Berlin, Heidelberg.

[9] Rosario Gennaro, R. and Goldfeder, S., 2021. One Round Threshold ECDSA with Identifiable Abort

[10] Komlo, C. and Goldberg, I., 2020, October. FROST: flexible round-optimized Schnorr threshold signatures. In International Conference on Selected Areas in Cryptography (pp. 34-65). Springer, Cham.

[11] CoW Protocol Overview - CoW Protocol. (2022). Available at: https://docs.cow.fi/.

[12] Emission & Incentive Design - Chainflip Docs. (2022). Available at: https://docs.chainflip.io/concepts/components/emission-and-incentive-design.

[13] Delegated Proof-of-Stake Consensus (DPoS) - BitcoinWiki. (2022). Available at: https://en.bitcoinwiki.org/wiki/DPoS.

[14] Proof of authority - Wikipedia. 2019. Available at: https://en.wikipedia.org/wiki/Proof-of-authority.

[15] Proof-of-stake (PoS) — ethereum.org. 2022. Available at: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[16] Oxen Service Nodes - Oxen Docs. [no date]. Available at: https://docs.oxen.io/about-the-oxen-blockchain/oxen-service-nodes.

[17] Paul R. Milgrom and Robert B. Wilson - The Prize in Economic Sciences 2020 . 2020. Available at: https://www.nobelprize.org/prizes/economic-sciences/2020/press-release/

[18] Auction Designs. 2017. Available at: https://www.fcc.gov/auctions/auction-designs.

[19] Validators & Auctions (SSOD) - Chainflip Docs. (2022). Available at: https://docs.chainflip.io/concepts/components/validators-and-auctions-ssod.