CHAINFLIP

Whitepaper

https://chainflip.io - 2nd of September, 2020 - Version 1.1 (last updated 9th November, 2020)

Abstract

Existing decentralised token swapping solutions suffer from a range of issues that severely limit usability, privacy, and practicality. Chainflip is a protocol for automated cross-chain token swaps that resolves these issues. The protocol described in this whitepaper will allow users to automatically swap tokens without relying on or using centralised service providers, wrapped tokens, or specialised software. Fees to compensate both network and liquidity providers are included in each swap, removing the need to obtain native tokens to pay gas fees. The operations of the system are primarily executed by a network of staked vault nodes, which jointly manage and secure both the volume and diversity of liquidity required to facilitate token swaps. The network of vault nodes acts as the network's decentralised authority, and achieves consensus over the state of the swaps, liquidity, and balances of the network using parameters outlined by a permissionless distributed database.

-

1. Introduction

Roadblocks to the adoption of decentralised trading solutions have been both technical and psychological in nature. Decentralised order book based exchanges such as Etherdelta, 0x and IDEX have largely failed to attract large user numbers or liquidity due to the complexity associated with managing an orderbook in a decentralised manner[1]. Additionally, they often require 'gas' or native tokens to interact with the DEX (regardless of what is being traded), which greatly hinders the user experience of these applications.

Most successful decentralised exchanges have relied on liquidity pools instead. In these systems, liquidity providers contribute an equal amount of liquidity to both sides of a liquidity pool, and a smart contract that determines the price between the two assets and presents the price-as-a-ratio. Price imbalances are corrected by arbitrageurs, who profit from buying or selling assets at values which are divergent from external markets.

Uniswap's ability to facilitate quick and convenient transfers between Ethereum tokens has demonstrated the value of liquidity pools[2]. However, with hundreds of mainstream blockchains, and thousands of tokens being used daily, it is clear the Uniswap concept must be extended beyond the Ethereum ecosystem. A more generalised method to transfer value between blockchains is needed. As a basic example, Bitcoin is recognised for its relative stability and use as a store of value, whereas Ethereum is generally utilised for programmatic interaction with smart contracts and the creation of tokens. Allowing users to quickly and trustlessly swap between currencies on different chains would represent increased flexibility and freedom in the way capital is allocated across different blockchains.

However, the liquidity pool swapping concept has not yet been widely explored in a cross-chain context. Most of the recent work in this area has focused on wrapping non-Ethereum assets into synthetic tokens (also known as 'wrapped' tokens), allowing those tokens to be traded on Ethereum[3] [4]. This is not ideal, as these tokens must be 'unwrapped' before they regain the properties of their native chain.

A system where tokens can be natively traded across blockchains without needing to obtain synthetic assets or specific tokens to pay 'gas' fees would significantly improve the situation. This is what Chainflip achieves.

Chainflip accomplishes this by establishing a network of bonded nodes which can collectively view, send, and receive transactions from multiple blockchains in parallel. Using these new nodes, transactions from any chain can be formed into liquidity pools. This allows any swap to occur between two pools, for example, BTC can be swapped with ETH through a single transaction which executes two trades: BTC to DAI, and then DAI to ETH. At no stage does the swapper need to have custody over any DAI, nor do they require native Chainflip tokens (FLIP), as the network fees paid in FLIP are deducted automatically from the swap and routed through the liquidity pools, where it is ultimately burned.

---

[1] "DEX Tracker - Decentralized Exchanges Trading Volume - DeFi." https://defiprime.com/dex-volume
[2] "Whitepaper - Uniswap." https://uniswap.org/whitepaper.pdf
[3] "Press Release - WBTC." 31 Jan. 2019, https://www.wbtc.network/assets/WBTC_Press_Release_Jan_2019.pdf
[4] "Republic Protocol Whitepaper - GitHub Pages." https://republicprotocol.github.io/whitepaper/republic-whitepaper.pdf

1.1 Decentralised Bridging

Allowing cross-chain interaction has been an ongoing issue for researchers and crypto enthusiasts; various options have been explored in pursuit of this goal. For several years, atomic swaps were heralded as the ultimate solution for trustlessly swapping assets cross-chain.

Unfortunately, atomic swaps require the use of Hashed TimeLock Contracts (HTLC) and specialised wallets, which few blockchains support. Rather, the method for transferring of coins between chains would be better if:

1. It was wallet agnostic. That is, it supported any generic wallet that can send ordinary transactions on a given blockchain;
2. It did not require the native chain to support exotic protocols or make changes to its underlying consensus rules or infrastructure, and;
3. It did not involve any 'wrapped' or synthetic assets. That is, there was simply one generic transaction submitted to conduct the swap.

Chainflip achieves this by using a system of staked nodes which generate and operate multi signature vaults. These 'vault nodes' are tasked with collectively managing liquidity pools and regulating each other's behavior.

2. Basic Structure

2.1 Overview

Key to any trustless swapping tool is a method of trustlessly securing funds which pass through it. Uniswap, Curve, and other existing liquidity pool platforms rely on the security of Ethereum smart contracts to allow users to trustlessly send funds in and out of these platforms. Chainflip, being cross-chain, cannot rely on the security of a single smart contract to produce the desired outcome.

Instead, Chainflip relies on a system of *vaults*, which trustlessly secure funds of users of the platform. One vault is established for each supported blockchain and is operated by *vault nodes*, a special type of server that stakes into the network to earn rewards.

Vault nodes and their vaults give Chainflip the ability to store funds in a secure and trustless manner, but unlike smart contract code, does not give a definitive ruleset for how funds should be processed once in the vaults. To accomplish this the Chainflip design includes a *state chain*. The state chain is operated by vault nodes, allowing them to come to consensus on when transactions should be created and to whom they should be sent.

By applying the rules of the state chain and the trustless nature of the vaults, users can use Chainflip to trustlessly swap assets across chains in a way that meets the three primary objectives of Chainflip.

2.2 Components

2.2.1 Vaults

Vaults are jointly managed cryptocurrency wallets controlled by staked nodes called *vault nodes*. To create these vaults, vault nodes participate in a setup process where new nodes are deterministically chosen to serve in the next active vault. These nodes jointly construct a threshold signature wallet from which transactions can only be sent if a given threshold of vault nodes sign a transaction. The schemes used to generate the vaults do not require a trusted dealer or reveal keys when signing transactions.

2.2.2 Vault Nodes

There are many types of bonded or staked nodes in the cryptocurrency space, but in other systems they are often not individually sufficiently staked for them to safely form the quorums used to secure vaults (see 5.1). Instead we need a new tier of bonded nodes, called *vault nodes*, which perform an extended set of operations from a typical blockchain node. These nodes require larger stakes, earn rewards from the block reward, and maintain the state chain for Chainflip and the requisite daemons for supported coins.

2.2.3 State Chain

The *state chain* is a side chain which acts as Chainflip's coordination mechanism. It contains all of the data pertaining to vault contents, as well as a ruleset for how to deal with transactions once they enter a vault. It is through the state chain that vault nodes come to consensus on when and where to send a transaction.

2.2.4 Quoters

Quoters are the interface between the user and the state chain. A quoter's main function is to insert *quotes* into the state chain on behalf of a user. Quotes contain swap details such as receiving and destination addresses, and optional additional details such as slippage limits, return addresses, and timeout rules. Quotes are also used to provide liquidity to and withdraw liquidity from liquidity pools. Quotes are the mechanism by which all users and liquidity providers interact with the system.

2.2.5 Liquidity Pools

Liquidity pools are an abstraction that consists of a reserved portion of two vaults. For example, a BTC/DAI liquidity pool would have a reserved portion of the Bitcoin and DAI vaults. Each blockchain only requires one vault, but each vault may be split among multiple liquidity pools. Liquidity providers add liquidity to these pools in order to earn fees when people trade across the pool.

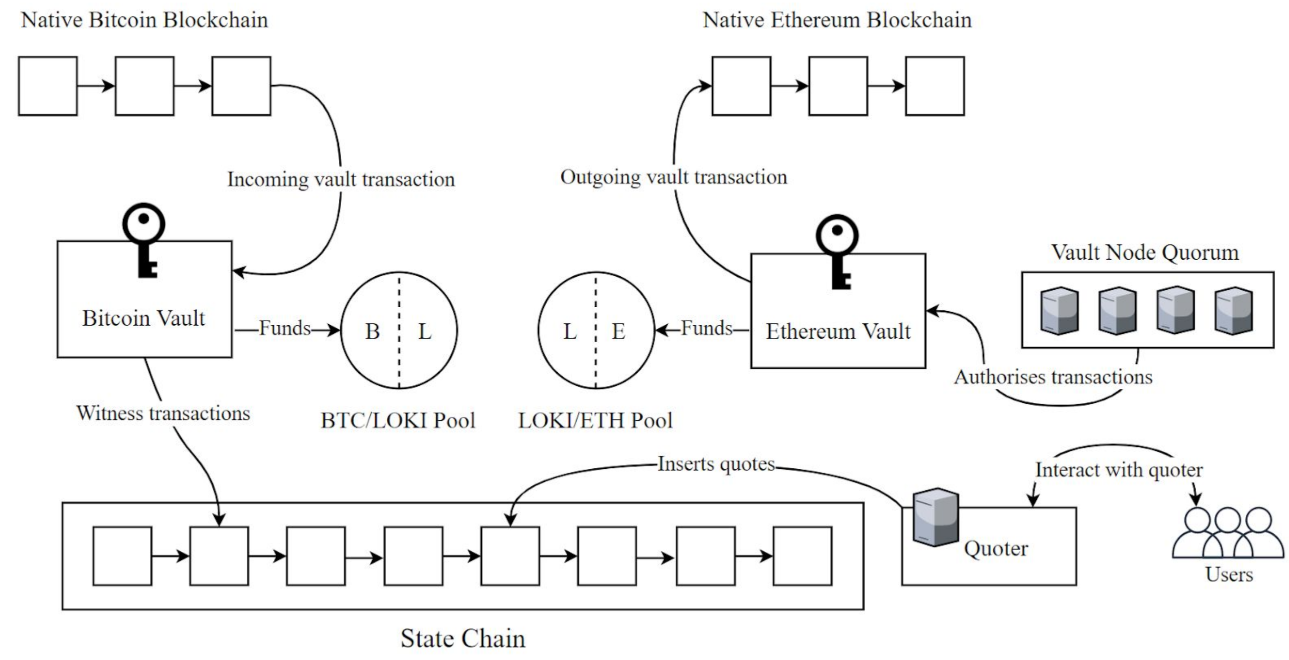


Figure 1: Users interact with quoters to generate receiving addresses for vaults, quoters insert this information to the state machine. Users can swap funds by sending native chain transactions to vaults, which are then swapped using the attached liquidity pools. Liquidity providers fund these liquidity pools and earn fees when users swap cryptocurrency. Withdrawals are authorised by groups of vault nodes.

3. Vault Design

3.1 Vault construction

The general approach for constructing vaults is to use shared multisignature keys which require a two-third majority of a vault's nodes in order to submit a transaction. Rather than rely on a one-size-fits-all approach, the vault management process must be optimised for each chain to provide the maximum security and efficiency for Chainflip. Most chains fall into one of three main categories, which will cover the vast majority of popular blockchains and crypto tokens:

1. Blockchains which natively use EdDSA for transaction signing;
2. Blockchains with a smart contract system that supports verification of EdDSA signatures; or
3. Blockchains which support neither a smart contract system nor EdDSA transaction signatures.

3.1.1 Native Ed25519

For coins which use EdDSA natively, we use threshold signature aggregation as described in *Stinson and Strobl (2001)*[5]. This algorithm works via secret sharing during generation and signing, which allows any *t* of *N* signers to aggregate individual signatures and produce native, verifiable EdDSA signatures without any participant possessing more than a single share of the secret used to sign a transaction. Although this approach is theoretically applicable over any Schnorr signature, our primary focus is on specifically Ed25519 as this is by far the most common signature type currently in use among cryptocurrencies.

[5] "Provably Secure Distributed Schnorr Signatures" http://cacr.uwaterloo.ca/techreports/2001/corr2001-13.ps

A native Ed25519 coin vault initially constructs a shared *t*-of-*N* signature through a vault generation procedure (as described in Stinson and Strobl). All *N* participants must distribute information about their share of the secret to all other nodes, and all *N* participants use their collection of secret shares to calculate the group public key. The algorithm allows identification of any nodes that attempt to cheat during generation, so key generation failure can be blamed on specific participants.

Transaction signing uses a similar secret sharing approach, but only requires *t* nodes to generate a combined, valid Ed25519 signature. A vault node leader will be randomly selected to create a transaction and initiate communication with other vault nodes. Each of the vault nodes in the process contributes an individual, verifiable signature for the transaction that they share within the signers subgroup. As in generation, this procedure permits detection of cheaters. Once *t* valid individual signatures are collected, they are aggregated into a single Ed25519 signature which is verifiable on the native chain using the group public key, and the transaction can be submitted to the blockchain.

Some major blockchains which this scheme applies to include Loki, Monero, Polkadot, Ripple, and Stellar.

3.1.2 Ed25519 Support in Smart Contracts

Although some coins (such as Ethereum) do not use native Ed25519 key generation and signing for transactions themselves, they do support verification of Ed25519 signatures within smart contracts[6]. This is highly beneficial as it allows for the creation of vaults as smart contracts, while also allowing much faster Ed25519 threshold signature calculations compared to relying on ECDSA threshold signing calculations on each vault transaction.

Updating smart contract vaults takes two steps. First, a *t* of *N* Ed25519 signature is constructed by *N* vault nodes as is done for native Ed25519 chains. The old vault updates the smart contract holding the vault funds to include the new Ed25519 public key of the vault node group. The smart contract will be deployed once per chain in a bootstrap process, and will have additional functions, such as accepting deposits and effecting withdrawals when given a valid Ed25519 signature over the withdrawal details for the stored public key, as well as a function to update the Ed25519 public key for vault rotations.

When vault nodes want to send a transaction from the vault, they construct a message specifying recipient and amount details plus a unique nonce (to prevent replay attacks). This message is then shared between vault nodes, as described in the native Ed25519 signature, until *t* valid signatures have been generated to produce a valid group signature. At this point, any node can submit the outgoing transaction to the contract using the valid group signature.

Smart contracts also allow vaults to have additional rules around the control of funds which allows Chainflip to prevent dishonest vault node minorities (see 6.2.4)

Some major blockchains that can support Ed25519 verification within smart contracts include Ethereum, EOS, and Tron.

---

[6] "Ethereum/EIPs - GitHub." 27 Jul. 2017, https://github.com/ethereum/EIPs/blob/master/EIPS/eip-665.md

3.1.3 EdDSA Alternative

For coins that lack both Ed25519 signatures or smart contracts which can verify Ed25519 signatures, we fall back to threshold ECDSA signatures as described in *Gennaro, Goldfeder (GG20)*[7]. This scheme supports *t*-of-*N* multi signatures for ECDSA type signatures with an identifiable abort if a dishonest signer is detected. However, it does so considerably less efficiently than our Ed25519 Threshold signature scheme, requiring both more network communication rounds and considerably slower computations.

Performance is noticeably lower for threshold ECDSA signatures, especially as the size of the signing quorum increases. Because of the computational cost of these multi-signature transactions, we cannot feasibly sign transactions on the fly while also employing larger vaults;
instead we work around this performance impact by periodically batch processing transactions by creating multi-output outgoing vault transactions every few minutes. Although this slows down the withdrawal of coins on the native chain using GG20, we anticipate the average delay will not be particularly noticeable when considering average block times required for mining and confirmations of a transaction.[8]

Although Schnorr signatures have long been discussed for integration into Bitcoin[9], they are not currently available. Chainflip will have to use the ECDSA GG20 scheme for Bitcoin and its major forks, including Litecoin and Zcash.

3.1.4 Benchmarking Threshold Signatures

To compare the two signature schemes, we conducted several signing benchmarks of different potential quorum and threshold sizes for Ed25519 and ECDSA threshold signatures. In order to model the latency required, we assume 200ms latency for each communication round in the algorithms:[10] This adds 400ms and 600ms (2 and 3 rounds) to Ed25519 generation and signing, respectively; and 800ms and 1400ms (4 and 7 rounds) to ECDSA generation and signing. Benchmark signing was conducted using a single core of a modern desktop system.[11] In actual deployment, vault node computational resources and latency between nodes will vary considerably and so these graphs should be viewed as a rough approximation of a real world scenario.[12]

---

[7] "One Round Threshold ECDSA with Identifiable Abort." https://eprint.iacr.org/2020/540.pdf
[8] For example, if a BTC transaction batch ran every 2 minutes, it would add an average of 1 minute to the average 30 minutes needed to mine and confirm a transaction.
[9] "bitcoin/bips: Bitcoin Improvement Proposals BIP-0340 - GitHub." https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki
[10] It is *single-trip* latency that is relevant here, and so this 200ms figure corresponds to 400ms of the more commonly referenced round-trip latency (i.e. ping times) between nodes. However, it also corresponds to the *worst* latency between pairs of vault nodes as nodes need responses from all other nodes before proceeding with the next round.
[11] Ryzen 3900x system.
[12] The depicted curves were estimated using the relationship $gen\_time = \beta N^2 t$ and $sign\_time = \gamma t^3$ for each equation, which yielded nearly perfect fits ($R^2 > 0.9999$ for the Ed25519 estimates and $R^2 > 0.97$ for the GG20 estimates). The prediction curves are then divided by N and t, respectively, to reflect that the computational work is perfectly distributed across the N and t participating vaults. Estimates were $\beta = .0001424$ and $\gamma = .0001423$ for Ed25519; and $\beta = 0.0003136$, $\gamma = 0.0006491$ for GG20.

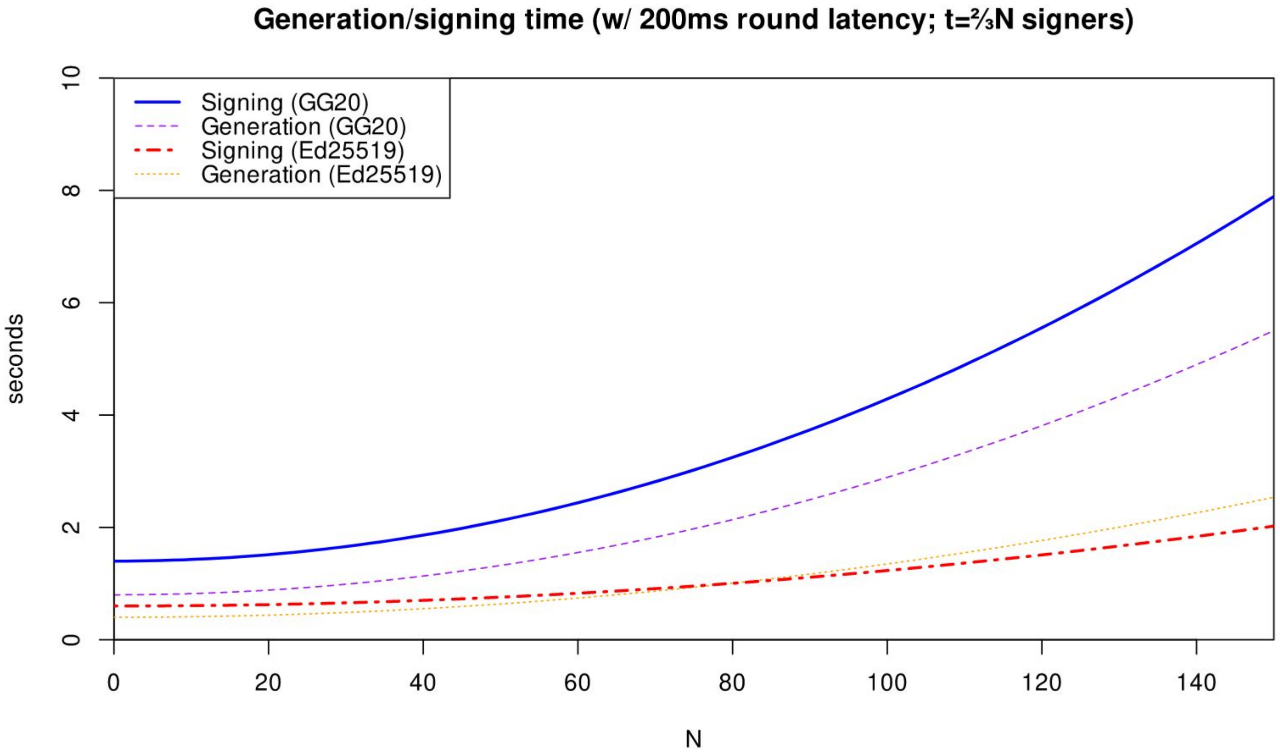


Figure 2: Network generation and signing times for Ed25519 and GG20 Threshold signatures with different numbers of signatories

As can be seen, Ed25519 signing scales considerably better with vault size, allowing for larger vaults and faster signing. Thus, where possible, vault schemes which use Ed25519 threshold signatures, either natively or within smart contracts, are the preferable choice.

3.2 Daemons

To be able to detect and validate incoming transactions, vault nodes need to run the daemons of all supported blockchains. When supported by the native chain, vault nodes only need to run light nodes, such as simplified payment verification (SPV) nodes[13]. This way vault nodes do not necessarily need to operate as full nodes for every supported chain, reducing the hardware requirements for vault nodes.

3.3 Vault Node Selection & Bidding

While there are many vaults run by different groups of vault nodes, there is also the *superset* of active vault nodes. From this superset nodes are selected to participate in individual vaults and have write access to the state chain. The superset of nodes is determined by a process called *vault node selection.*

To become active in the vault node superset an operator must bid for a position in the next vault node selection. While a minimum stake is required for vault nodes, the actual amount of FLIP the operator must stake to be selected is determined by the bidding process, where the N nodes with the greatest bids will form the next vault superset. Nodes that are active in a current selection can automatically use their

[13] "Bitcoin: A Peer-to-Peer Electronic Cash System - Bitcoin.org." https://bitcoin.org/bitcoin.pdf

unpaid rewards as bids for the next selection round. By forcing node operators to compete for selection in a limited number of possible node slots, a market dynamic is introduced for staking requirements and collateralisation.

This market dynamic has several important properties. Nodes that have been penalised for excessive downtime will have less FLIP staked compared to other active nodes, which means they are more likely to be outbid by new operators and must either increase their bid to maintain their position or face being removed in the next selection. This dynamic also allows the staking requirement to scale with platform growth to better address collateralisation.

The superset of possible nodes is limited in number in order to better scale communication and transaction signing protocols. Active vault nodes that have been selected must participate, at a minimum, in state chain notarisation, even if they are not ultimately included in any vaults.

The vault node selection process is triggered under two possible circumstances:

1. The percentage of vault nodes in the current superset that have gone offline exceeds a safety threshold; or,
2. The lifetime of the superset exceeds the 28 day limit.

In either case, the next superset will be selected deterministically based on the current stakes and bids that have been registered on the Chainflip network. A testing round is conducted to ensure nodes 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.

3.4 Vault Rotation

Each vault is constructed from a *subset* of the overall vault node superset. The process that replaces an existing subset with a new subset is called *vault rotation.*

3.4.1 Rotation Frequency

Vault rotation is triggered under two possible circumstances:

1. The percentage of nodes that have gone offline in a current vault has exceeded a safe threshold; or,
2. The vault superset is being rotated.

The frequency with which vaults rotate has an effect on the likelihood that an attacker with a large percentage of the vault node network could gain controlling majority of a vault. By reducing the vault rotation period, the chance an attacker will gain control of any single vault is reduced over time (see 6.2.2).

On the other hand, node operators are imperfect, so it is expected that some nodes will have long periods of unexpected downtime, despite penalties. Excessively long lockup periods are also unattractive to potential vault node operators, negatively impacting collateralisation and therefore the security of the platform.

The limit on the lifetime of a vault is the same as a superset lifetime with an added 48 hour overlap to allow for handling delayed incoming transactions. With vault rotation occurring after a new superset has been selected, there is a period of time in which both incoming and outgoing vault nodes from the superset must remain online before any stakes are unlocked. Practically speaking, this means that the minimum staking period for a vault node is 30 days.

From a new superset determined during vault selection, the second process of vault rotation begins with randomly assigning members of the superset into one or more subsets, each of which are used in different vaults. Once these subsets have been selected, new vaults must be created or updated before any assets are transferred to the new subsets.

3.4.2 Creating new vaults

The state chain is used to conduct a vault creation ceremony between members of a new subset. Depending on the design of the particular vault to which they have been assigned, this process can involve signing ceremonies, updating smart contracts, and multiple synchronous communication rounds. Once each vault is successfully created, and proof of successful tests have been submitted to the state chain, the new vault is ready to receive assets from the outgoing vault. If any vaults are created incorrectly, the nodes that cause the failures are removed from the rotation and replaced by other members randomly selected from the superset. Once all new vaults are live with no failures, the final step of vault rotation occurs.

3.4.3 Transitioning to new vaults

Once all the new vaults are registered in the state chain, all quoters will need to point towards the new vault, resulting in a situation where the liquidity of a single liquidity pool could be split between as many as four different vaults at once. Any new swaps or stakes are sent to the new vaults to avoid users sending funds to old vaults. The state chain contains rules to determine from which vault the funds from swaps and pool transactions should be sent from during this transition period.

Once the transition period has elapsed, all quotes referencing old vaults will expire. The old vault signs one final transaction, transferring the remaining contents of the old vault into the new vault. Once this is witnessed, the old vault is now considered decommissioned, and all of its previously held assets and balances have been transferred to the new vault.

Once a vault is decommissioned, it is assumed the required majority of signers will move on and anything left inside the old vault will generally be unspendable unless members of an old vault agree to process a late transaction. Once a vault is decommissioned, the stakes of the nodes from the vault not included in the new superset are unlocked.

4. The State Chain

The state chain acts as the coordinator between all vault nodes, allowing all nodes to come to consensus about the current state of the liquidity pools, swaps, and vault balances. Functionally, the state chain is a permissioned consensus-based distributed database that is kept in sync with other vault nodes, allowing for a shared state to be maintained.

Two parties are permissioned to alter the state chain: vault nodes and quoters. Vault nodes can submit two types of transactions — witness transactions and pool balance transfers — to the state chain, which serve to update the state of balances and swaps inside liquidity pools. Quoters can create and submit swap quotes, liquidity provision quotes, and liquidity withdrawal quotes.

4.1 State chain transaction types

4.1.1 Witness Transactions

The vault nodes submit 'witness' transactions whenever they see and receive enough confirmations on incoming transactions to their vault. This witness transaction is then signed by other vault nodes who witness the same transaction on the native chain, and once it gains enough signatures, it is considered valid and included in the state chain. Because of the submitted quote, if the incoming transaction has an identifier which matches the quote, the vault nodes must now process that swap using the incoming transaction.

Outgoing transactions work in a similar way. Once a swap or withdrawal has been processed, vault nodes will witness the outgoing transaction from the vault on the native chain and submit that change of state to the state chain.

4.1.2 Pool Balance Transfers

Each vault has a balance, but that balance can be distributed across multiple liquidity pools. In cases where multiple pools rely on the same vault, a supermajority of vault nodes can transfer balances between the pools on the state chain without creating any outgoing transactions on the native blockchain for that vault. This will be the case for many swaps, where a swapper can route their incoming BTC through the BTC-DAI and DAI-ETH pools, generating an outgoing ETH transaction and causing a pool balance transfer in the DAI vault, without ever holding DAI themselves: instead the state chain records a change in the DAI holdings of the two liquidity pools without sending a second transaction on the Ethereum blockchain.

4.1.3 Quotes

A quote is a user-defined rule for the state chain, and is produced at the request of the user when they want to provide liquidity, withdraw liquidity, or conduct a swap. The quote will contain all of the user-defined parameters associated with that quote type, and gives the vault node network all of the information it requires to process these actions once the correct conditions are met in the state chain. Quotes are generated and inserted into the state chain by quoters.

Swaps

To complete a swap, vault nodes need to be able to differentiate incoming transactions. When providing a quote, the quoter will need to generate a unique chain-specific identifier for the deposit, usually accomplished by generating a new address to be used when sending funds to the underlying vault. Quotes can also include rules such as slippage-limits, return addresses, and timeout rules. Once the quote is in the state chain, users can verify it and its contents publicly before sending funds to be swapped.

Liquidity Provision

Liquidity provision works in a similar way to regular swaps. Liquidity providers are required to interact with a quoter to fund liquidity pools.

Because of the cross-chain nature of Chainflip, and because liquidity providers often do not have a perfectly balanced portfolio of assets to add into liquidity pools, Chainflip supports asymmetric liquidity provision. This means anyone with any ratio of assets between two sides of a liquidity pool — including just one of the assets — can easily provide liquidity for that pool without having to manually rebalance their portfolio: instead, Chainflip automates the rebalance by performing an implicit asset swap of the provided liquidity within the liquidity pool itself.

Before adding liquidity to a pool, a liquidity provider must supply a quoter with the parameters for their liquidity, including their return address(es), a withdrawal code, and a quote expiry time. With this information, the quoter can generate address(es) for the provider to send their liquidity to, and add the quote to the state chain.

Once the user sends funds to the address(es) specified in the liquidity provision quote — or once the quote expires — the vault nodes execute the provision quote. Deposited amounts that precisely match the existing liquidity pool ratio are credited directly, while any remaining balance is handled as if the provisioner had first performed a swap to the required ratio. This liquidity is then added to the pool for use by swappers.

Liquidity Withdrawal

In order to retrieve liquidity being used in a pool, providers can generate a withdrawal quote through quoters. By specifying return addresses and initially generating a withdrawal keypair providers have multiple means of authentication which can be used to trigger a withdrawal.

The provider can sign using either of their return addresses or their withdrawal code. This signature is injected into the state chain by a quoter in the form of a withdrawal quote, which can also include additional parameters, like adding missing return addresses or forcing an asymmetric withdrawal.

Once the withdrawal quote is in the state chain and is processed by the vault nodes, the vaults will send the funds that belong to the provider directly to the return addresses. In the case of an asymmetric withdrawal where only one return address has been specified in the withdrawal quote, one side of the withdrawal will be swapped into the other asset before all of the liquidity is sent as a single outgoing transaction by the vault.

5. Attack Prevention & Incentives

In analysing the design of Chainflip, it is important first to understand the types of attacks to which the Chainflip system could be vulnerable. There are three main types of actors considered in designing the security for the system:

1. Financially motivated attackers, who will attempt to steal cryptocurrency or ransom other user’s coins for a profit;

2. Non-financially motivated attackers, who act not for their own financial gain, but to shut down the service and/or cause the assets in the liquidity pools to be frozen or destroyed; and lastly,
3. Financially motivated honest actors, who use the system as intended in order to maximise financial benefit.

For all of these examples, consider that each vault requires a 2/3rd supermajority of signers (t of N) in order to generate a valid transaction.

Financially Motivated Attackers

Vault nodes have shared custody over liquidity provider's funds. This is an attractive target for financially motivated attackers who would want to steal these funds. In order to take these funds directly, the attacker would need to control at least 2/3rds of vault nodes to sign a transaction. The attacker can gain control by owning *t* nodes themselves, or convince other node operators to collude in order to sign a malicious transaction. If the attacker achieves this, they effectively seize direct control of the vault's funds.

There's also a more subtle attack: if the attacker controls a superminority (>1/3rd) they can block valid transactions from occurring. The attacker can simply stop signing transactions in the vault and attempt to ransom the vault's contents from the liquidity providers. This way, the attacker can extract value from the users without ever having to acquire the full 2/3rds of vault nodes required to pull off a complete theft of the pool.

Non-financially Motivated Attackers

Further major attacks are possible if the attacker is not financially motivated. Conducting a denial of service attack by destroying a superminority of vault node keys would be effective in preventing the use of the system. Such an attack would quickly destroy trust in the Chainflip network as an effective tool for cross-chain swaps.

Financially Motivated Honest Actors

We assume that most actors are motivated by financial profit, and so do not consider the goodwill of participants as a given. Without aligned profit incentives, we would observe lower participation and a greater chance of attackers corrupting the system, which is why we also consider incentivising good behavior for the purposes of security.

5.1 Vault Collateralisation & Incentives

One of the simplest ways to protect assets held in vaults is to force vault node operators to stake FLIP in order to join the network, using their capital as collateral, with a block reward being provided by the network as an incentive. It is the yield from the block reward that will attract collateral to the vault nodes in the first place, allowing us to use that collateral to provide Chainflip with security.

The incentive for vault nodes comes at a network cost in the form of emission. Given this emission is created programmatically and not in accordance with how much liquidity is actually in the system, we must define the amount of rewards given to vault nodes. Recent analysis of staking on the Loki network has shown the velocity of money has dramatically reduced with the introduction of staking, and it seems clear that this effect will be observed in other Proof-of-stake or similar collateral based security systems, including Chainflip. This means that FLIP created and rewarded to vault nodes should only slightly impact the purchasing power of FLIP, although emission does have a long term effect on the overall value of the token. To counteract this emission, FLIP is used and burned in each swap, and if burned in sufficiently large numbers, will be able to offset the newly created tokens awarded to the vault node operators.

Furthermore, as each swap will inadvertently involve purchasing FLIP through the system, a base level of direct demand pressure for the token will exist for as long as the system is used, and is directly proportional to the usage of the system. This creates a dynamic where every owner of the FLIP token has a direct incentive to improve, develop, and encourage the usage of the system.

Vault nodes must be collateralised sufficiently to prevent ransom attacks and outward theft by a supermajority. Intuitively, one might expect that to adequately protect $1m of liquidity, vault nodes would need significantly more than $1m of collateral to protect that liquidity from being stolen by a malicious attacker with enough vault nodes to form a supermajority. But in fact, using additional security countermeasures, collateralisation can be equal to or less than the total liquidity in the vaults.[14]

5.2 Vault Attack Countermeasures

5.2.1 Slashing

Chainflip contains a *slashing* mechanism that allows the vault node network to destroy the stake of nodes that are found to have acted against the consensus of the state chain. Given this mechanism, many of the financially motivated attacks become both unprofitable and prohibitively expensive if the value of the stake in the vault nodes exceeds or matches that of the possible windfall from a given attack.

For instance, if there is $1m worth of BTC in the Bitcoin vault, but the value of the FLIP collateral staked to run a majority of vault nodes for Bitcoin matches or exceeds $1m, then the attack is not profitable as long as the attacking vault nodes' stakes can be effectively slashed.

Theft can be detected by observers of the state chain, as all outgoing transactions from a vault require an input to be valid. Only when a valid incoming swap on another chain with a matching quote comes in, or a valid liquidity withdrawal request is published, will observers consider an outgoing transaction from the vault to be valid. Upon the detection of a violation, a simple majority of the superset of vault nodes can submit a slashing transaction to the Chainflip network to destroy the stakes of the offending vault nodes. Although a dishonest majority of the superset could arbitrarily slash the stakes of honest nodes, with no evidence of a theft on the state chain, this kind of attack would be obvious and action could be taken through off-chain governance, restoring the victim's stakes and destroying those of the attacker's.

5.2.2 Vault Randomisation

Some of the vaults in Chainflip can be controlled by the entire superset of vault nodes. For instance, the Ethereum vault can easily support 150 vault nodes in its subset (see 3.1.4). GG20-based vaults such as Bitcoin do not scale as well. For vaults with scaling limitations or reduced expected liquidity, it may make

[14] It is worth pointing out that the practicalities of acquiring enough FLIP to outbid enough vault nodes control even 40% of the network is considered to be extremely difficult. With limited liquidity and sufficient distribution of supply to rational economic actors, any attempts to conduct a hostile takeover of the vault node network would take months or years and cost an extraordinary sum of input capital. Moreover, a sustained takeover attempt would send the cost of FLIP higher, which in turn increases collateralisation costs. It is for this reason the Chainflip team considers it acceptable to have the network be under-collateralised to a degree.

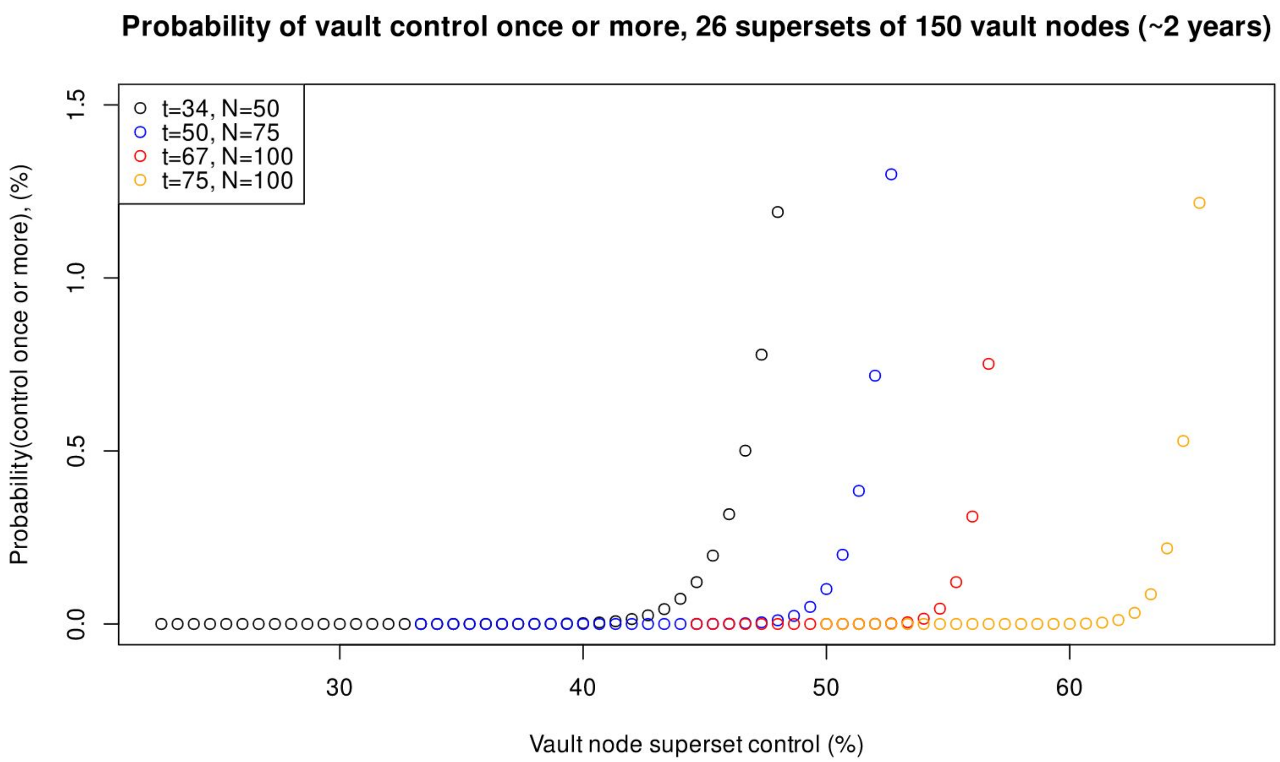
sense to construct vaults from smaller subsets of the vault node superset. Ultimately the vault size required for each supported chain will depend on both the performance implications and security required by that chain.

For vaults which do not employ the full superset of vault nodes, *vault randomisation* is used to limit the potential windfall of any one attack. It involves deterministically selecting the members of a given vault in such a way that the system cannot be influenced to drive a given actor's nodes into a specific vault or to predict vault composition in advance. This way, even if the attacker controls enough nodes to be *able* to form a supermajority in a smaller vault, the random nature of selection means it is still extremely unlikely to occur in practice.

To illustrate the combined effectiveness of vault randomisation and slashing, consider a scenario where there are 150 active vault nodes (the superset) with an aggregate of $7m worth of FLIP staked into vault nodes, all of which are participating in vaults, and 3 vaults with the following liquidity balances and vault sizes:

DAI: $4m (150 nodes)
DOT: $3m (150 nodes)
BTC: $1m (75 nodes)

In this scenario, the attacker must control *at least* 50 nodes — ⅔ majority of 75 nodes — to have any chance of a successful attack against BTC. However, the chance that all the attacker's 50 nodes will be randomly selected to become a part of the same vault is practically zero; in practice an attacker would need more nodes than this minimum.[15] Because Chainflip vault rotations only occur every 28 days, attackers with large but non-majority control will still have to maintain their position over an impractically long time to have any reasonable chance of success from repeated attempts to form a vault majority.



[15] More precisely: 2.61262×10^{-21}; this is a hypergeometric distribution.

Figure 5: Vault compromise over a two year period is shown for different numbers of controlled nodes and different signing thresholds.

Even with half of the vault node superset controlled (75 vault nodes), attackers still have a chance of 0.100882% to successfully form a supermajority in this scenario over 2 years. This represents an enormous opportunity cost, but moreover even if the attacker could steal the $1m worth of BTC, they would be detected and lose their FLIP collateral. In this example their stake would be worth at least 2.3 times as much as the BTC. The attack is simply not profitable.

The Sybil resistant nature of other systems such as the Loki network's existing staking system demonstrates that this kind of vault node staking design will adequately defend the network from people attempting to acquire a majority stake in the vault node network, and thus attacks on vaults.

5.2.3 Vault Collateralisation

By randomising vaults and slashing bad actors, Chainflip can rely on vault nodes even when they are not substantially overcollateralised. There is no hard limit to what would be considered an 'acceptable' collateralisation — the greater the collateralisation, the less likely an attack of this type is possible. But given the extreme costs of owning large percentages of the circulating supply, the low iteration speed on vault rotations, the randomisation of vaults into randomly selected groups of nodes, and the slashing of collateral upon a theft, an attack of this kind would not be viable. The total value stored in vaults could exceed the collateralisation to a moderate extent without greatly impacting the security of the system, but it is acceptable for Chainflip vault nodes to be collateralised for as much liquidity Chainflip holds.

A hard limit of collateralisation will be enforced, where liquidity providers will be prevented from adding liquidity when the total value of the liquidity pools reaches a multiple of the value of the collateral staked in vault nodes, preventing liquidity providers from causing the system to be substantially under-collateralised. The rest will be left to market dynamics — if liquidity providers are uncomfortable with the level of collateralisation available, they can choose to withdraw their liquidity.

5.2.4 Vault Timeouts

In the vault scheme, another serious attack under consideration is a ransom or burning attack. This kind of attack can be conducted by anyone controlling a superminority of nodes. In a t-of-N threshold signing scheme, the value required to form a superminority is $N-t+1$. For example, where there are 99 vault nodes (N) and 66 are required to sign (t), 34 vault nodes can form a superminority. This gives the attacker a blocking vote on all transactions in the vault. If they are financially motivated, they could prevent all outgoing transactions from the vault and demand payment from the liquidity providers or other parties to get them to 'unfreeze' the vault. If they were simply a malicious attacker, they could erase the private keys of their voting block so no transaction could ever be signed from the vault again.

In normal vault operation, outgoing transactions should be happening very frequently. A scenario in which no transaction leaves the vault for several days would never occur in a normal period of operation. For smart contract based blockchains, we can add a function in the vault smart contract which allows a community-defined emergency backup address to withdraw all funds if there is no activity in the vault after a set period of time, making it possible for the community to pre-approve the recovery from a timeout situation. By adding the timeout function, any ransom attacks or general breakdown or failure of the vault as a whole can be recovered manually. This effectively renders these kinds of attacks ineffective, albeit disruptive.

For vaults, protocol level changes can be made to provide similar protections to smart-contract based vaults. By enforcing the exposure of view keys and key images for vault nodes, this will ensure that funds in vaults that have no activity after a specified timeout period are permanently frozen, re-minted, and given back to liquidity providers.

5.2.5 Penalty System

The requirements of running a vault node are likely to be significantly higher than that of a typical Service Node. In order to ensure vault nodes are maintaining good uptime, signing transactions in a timely manner, and correctly processing swaps without compromising the security of funds, we need to implement a system of penalising nodes that are under-performing or have gone offline.

Enforcement will be achieved through a credit system. Vault nodes will earn credits when they are in the first group of nodes to sign witness transactions or outgoing transactions from a vault. Credits may also be deducted from vault nodes when they exhibit poor behavior, like refusing to sign valid transactions, going offline, or being consistently slow to sign transactions.

The credit system works as a cumulative scoring system with all nodes starting at 0 credits. When nodes unstake, a negative credit score will lead to a portion of their stake being slashed. Nodes that have a positive score have more leeway during short periods of poor behavior, like slower than normal signing, or temporary outages. This system is designed to encourage node operators whose score has gone into the negative territory to stay online, boosting their credits into positive territory to avoid funds being slashed, and to reward well performing nodes with more breathing room in case of unexpected issues.

5.3 Front Running Attack Countermeasures

Since all transactions must be sent using public native blockchains of the supported cryptocurrencies, there is a risk of front running attacks.

For example, if Alice wants to buy 1 ETH with Bitcoin, then Bob may monitor the Bitcoin blockchain and the Chainflip state chain waiting for a Bitcoin transaction destined for the vault address. When he sees this transaction, Bob can submit his own transaction with a higher BTC transaction fee than Alice, which is likely to confirm and execute on the Chainflip state chain faster than Alice's transaction. By doing this, Bob has effectively pushed the price of ETH up, which means when Alice's trade executes, she gets a worse rate. Bob can now sell his ETH back into the liquidity pool getting an increased price due to Alice moving the market behind him. This attack already occurs on existing platforms and means users often get worse rates than quoted[16].

5.3.1 Slippage Limits

There are some well established solutions to these problems. The most obvious is to allow the user when setting up their quote to specify a maximum allowable amount of slippage. The quoter would insert this limit into the state chain with the quote, and once the incoming transaction is received, the nodes would not execute the trade if the slippage exceeds the limit specified in the quote, instead returning the

[16] "Warning Bots Front Running Uniswap Contracts - Reddit."
https://www.reddit.com/r/UniSwap/comments/b4jkly/warning_bots_front_running_uniswap_contracts/

incoming assets to a specified return address. This does not completely negate front running, however it limits the degree and capacity of front running attacks.

5.3.1 Transaction Ordering

Ordering algorithms which cannot be influenced by the front runner can also limit front running attacks. If we assume the swapper pays a high enough fee to ensure their transactions make it into the first block after broadcast, then the frontrunner can only race to be included in that block. The vault nodes will see both transactions in the same block and instead of assigning the order of execution based on the order of the block, they can assign order based on the time a quote was activated in the state chain.

6. Future work

6.1 Liquidity Pool Fees & Other Balancing Problems

One of the key discussions in the AMM sector is the fee structure for liquidity provision. Much of the conversation has surrounded the appropriateness of the UniSwap fee model, and its impact on liquidity providers and their impermentant loss.

Uniswap offers swappers a price based on the size of their trade (a depth-based calculation called "price impact") and charges a flat 0.3% fee on top[17]. This has the effect of making it prohibitively expensive to consume large percentages of the liquidity of one side of the pool during a swap. This is a necessary design element of liquidity pools — without an exponential relationship between price and liquidity, pools could be drained on one side at no real cost to the trader.

Much of the contention around the fee model is related to impermanent loss, with some pundits arguing liquidity providers should be prioritised above all others using liquidity pools[18]. The assertion that liquidity providers are the most important class of user is intuitive — without liquidity, no one gets to process swaps, and the more liquidity there is, the cheaper it is to execute trades. This is the express purpose of the Continuous Liquidity Provision (CLP) fee model[19].

However, increasing the 'price impact' relationship (making bigger trades more expensive), as seen in CLP, has an impact on the profitability of arbitrage traders. This in turn has an effect on the efficiency of the market and the ability of the liquidity pool to match the conditions present on other markets. Increasing fees also discourages regular users. While liquidity providers may be more insulated against losses, ultimately it is their reliance on swapping that makes liquidity provision potentially profitable in the first place.

All of this is to say that there is no right answer to the problem of setting the correct fee structure. Not only are the impacts of different fee models incredibly difficult to measure, but different markets have different requirements, with markets such as DAI/USDT having different properties from that of ETH/LINK for instance, which may make a different fee model more suitable. Part of the future work of Chainflip is to

[17] "Fees - Uniswap." https://uniswap.org/docs/v2/advanced-topics/fees/
[18] "Revisiting Fees and Impermanent Loss | THORChain - Medium." https://medium.com/thorchain/revisiting-fees-and-impermanent-loss-4fbf9ee35fd5
[19] "Continuous Liquidity Pools - THORChain." https://docs.thorchain.org/how-it-works/continuous-liquidity-pools.

critically analyse fee models and apply them as appropriate to the various liquidity pools created in Chainflip.

7. Conclusion

This paper has proposed a protocol which enables decentralised cross-chain asset swaps. The design of Chainflip's vaults and the underlying instruments which govern them enables cross-chain swaps to be completed without the use of additional software, collateral, or synthetic assets.

Collateral based security systems such as Proof of Stake have been used to produce many new ways of utilising distributed blockchain node networks. Chainflip is a further application of this concept, wherein staked nodes are used to securely construct and manage vaults. These vaults are drawn upon to form token liquidity pools which pair supported tokens and allow users to perform swaps. The state chain allows vault nodes to come to consensus on the state of the Chainflip system. Users interact with the state chain through quoters, which lodge swap requests on behalf of users.

The Chainflip protocol will enhance the usability and universality of all supported tokens, and help remediate splintered cryptocurrency markets. Additionally, the proliferation of decentralised finance products aims to reify the original thesis of cryptocurrency and blockchain technology: providing decentralised, secure, and versatile asset exchange solutions for the twenty-first century and beyond.