ZetaChain: A Blockchain with Native Cross-chain Smart Contracts

2023-02-22, v0.4.0

ZetaChain
team@zetachain.com

Read this whitepaper in español, simplified chinese, traditional chinese, japanese, hindi, korean, vietnamese, deutsch, português, français, russian, ukranian, persian, türkçe, indonesian, and filipino (non-English translations are created by the community, not ZetaChain).

Abstract

This white paper proposes ZetaChain, a blockchain with generic omnichain smart contract support that connects both smart contract blockchain such as Ethereum, Ethereum L2 rollups, Solana, Terra, and Algorand, and even non smart contract blockchains such as Bitcoin and Dogecoin. ZetaChain consists of a Proof-of-Stake blockchain and observers and signers for external blockchains. The observers scan external chains for relevant events, transactions, and states at a point in time, and reach consensus on observation on ZetaChains blockchain. The signers collectively possess a single Threshold Signature Scheme (TSS) key that is able to send authenticated messages to external chains and hold assets like normal accounts/addresses on external chains. Smart contracts on ZetaChain support arbitrary logic that executes conditionally on external chain events, and can directly update external chain states via its TSS signed transactions. ZetaChain thereby enables omnichain dApps that interact with different blockchains natively and directly without wrapping or bridging any assets.

1. Introduction

It's hard to imagine a single blockchain would suffice for all our society's use cases. A multi-chain future seems inevitable. However, a multi-chain future without interoperability between the blockchains could be paralleled to the Internet before TCP/IP. Today's blockchains are too fragmented and are by-nature not interoperable, hindering mass adoption of the technologies. For example, a decentralized application (dApp) must be married to a specific blockchain. If a user onboards into the crypto ecosystem through a given dApp, this fragmentation makes immense barriers for the user to fluidly adopt or try a dApp on another chain. To address the issues of interoperability, there have been a few proposals and projects that specifically emphasize the ability to inter-operate. However, the majority of interoperability systems only apply to specific blockchains, standardize their protocols within their own systems requiring other blockchains to adopt or through complicated, restricted, and/or less secure bridges to join (see Figure 1). In this whitepaper we propose a novel, public L1 blockchain that actively and agnostically connects blockchains and facilitates interoperability. Furthermore, we propose a generic smart contract on blockchain that can hold and manipulate assets on external blockchains directly, thereby enabling generic smart contracts that can custody assets on external chains. This opens the door to boundless cross-chain dApps.

Blockchains are naturally closed systems. The goal of this whitepaper is to design and specify a practical system that is generic in its inter-blockchain capability, without forcing existing blockchains to adopt new standards or a new blockchain that every assets needs to move to, and do so in a decentralized, byzantine fault tolerant way. In other words, we aim to create a public blockchain that supports real cross blockchain transactions, message delivery, and general cross-chain smart contracts. According to our extensive survey, to satisfy this goal, the best pragmatic approach is the decentralized notary scheme on top of an incentivized Proof-of-Stake replicated state machine (aka blockchain) which we call ZetaChain.

ZetaChain is first of all a public blockchain with Proof-of-Stake validators. It's trusted that a super majority (>66% nodes) of the validator nodes are honest and act according to protocol, and collectively serve as notaries. In addition to being a blockchain, interoperability requires observing other blockchains. Thus each ZetaChain validator node is attached with an observer that scans other blockchains for relevant events (event log, transaction, or state at a certain time). The observers report the relevant events to ZetaChain and reach consensus. ZetaChain uses custom logic to update its state in response to the reported events. On the other hand, in order to change state on other blockchains, each validator is also attached with a signer holding a key share. Collectively all the validators hold a single public/private key pair which can initiate transactions on other blockchains to change state directly. The signature scheme can be some kind of threshold signature scheme such as GG18/GG20 ECDSA/EdDSA, or BLS threshold/aggregate signatures, depending on the cryptography on different chains and their smart contract capability/cost. The presence of a single public key and address in the ZetaChain system allows ZetaChain to custody assets on external blockchains which might not have adequate smart contract capability such as Bitcoin. Such ability allows powerful cross-chain (or omnichain) dApps to be built on top of native ZetaChain cross-chain smart contracts. This capability looks much like on Ethereum where a smart contract can be trusted to manage assets according to predetermined rules, except on ZetaChain, a smart contract can leverage and manage assets on any connected blockchain.

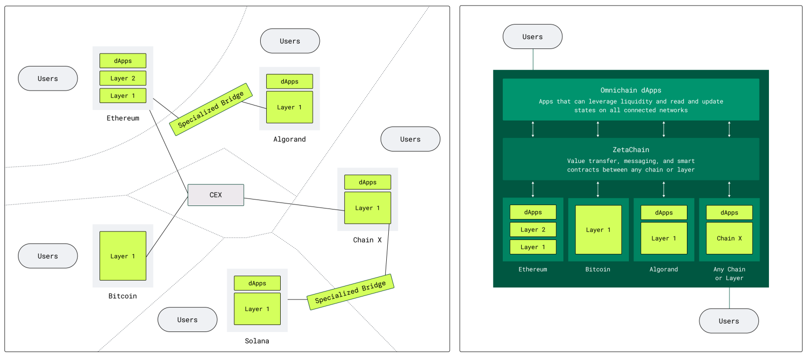

Figure 1. Before and After ZetaChain. Sub figure (a) on the left: Current ecosystem. Users and developers are siloed into respective chains, and current cross-chain solutions are disparate, resulting in major, growing fragmentation. Sub figure (b) on the right: Ecosystem with ZetaChain. Users, developers, and apps can operate across chains in a seamless manner. New paradigm of Omnichain dApps enabled.

In summary, ZetaChain is designed to be a decentralized cross-blockchain smart contract platform. The vision of ZetaChain is to be a public computer on all important blockchains, on top of which cross-blockchain decentralized applications can be easily built as public, trustless, and persistent smart contracts.

2. Background: Evolution of Blockchains

2.1. Bitcoin: the original decentralized cryptocurrency

Blockchain, pioneered by Bitcoin, is a decentralized and permissionless public ledger built on cryptography. The core mechanism is byzantine fault tolerant distributed consensus, which Bitcoin solved by a combination of techniques from cryptography, economic incentives, and computer science. Key innovations in Bitcoin include the use of elliptic curve digital signatures algorithm (ECDSA) for self-custody of funds, and the use of Proof-of-Work to reach distributed consensus (ordering of the ever growing log of transactions) and maintain resistance against sybil attacks. Bitcoin also introduces the first major application of blockchain technology—a p2p cryptocurrency. The Bitcoin network has been highly successful, even though it has not fulfilled its promise of being electronic cash. Rather, Bitcoin has become the most secure, decentralized, and stable store of value due to its technical simplicity and robustness, high degree of decentralization and low barrier of participation, and predictable and conservative monetary policy.

The Bitcoin network consists of nodes connected by a p2p network. Participants include users and miners. The Bitcoin network collectively maintains a growing ledger that is a sequence of user transactions. A user transaction is a signed message that spends a certain amount of coins controlled by the user. The Bitcoin network does not explicitly maintain the balance state of each account; the only state of the network is the set of current UTXOs—unspent transaction outputs. A users balance of BTC is the sum of all UTXOs that can be spent by the user. A user transaction includes one or more UTXOs as inputs, and creates one or more UTXOs as outputs, thus changing the state (UTXO set). Bitcoin supports a limited form of scripting: a transaction can send coins to a script, and whoever can satisfy the script (i. e. supply data to make it evaluate to 1) may spend the coins. The scripting language is deliberately simple and Turing-incomplete — namely without branch and looping structures — but supports quite a few simple but fundamentally useful applications such as multi-sig, atomic swaps, etc.

2.2. Ethereum: the programmable blockchain with smart contracts

While Bitcoin is conceptually a simple ledger (ordered sequence of transactions) with basic scripting features that has served as the canonical example of a blockchain, it is not the limit of what a blockchain can do. For example, due to the limited scope of the verification function of the Bitcoin protocol, its not possible to issue new coins on the Bitcoin network. The Bitcoin network is not programmable in the sense that an arbitrary state transition function can be implemented. The only state transition function that Bitcoin supports is the hard-coded UTXO set change. In summary, no applications other than BTC currency can piggy-back on the Bitcoin network, inheriting its consensus, decentralization, and security. To extend the scope of blockchain to support Turing-complete programmability, Ethereum was born. Ethereum borrows the Proof-of-Work from Bitcoin for its consensus, and has made several important innovations that make it a public programmable blockchain. First, Ethereum defines a virtual machine (EVM) that provides a Turing-complete sandbox environment to specify arbitrary state transition functions (smart contracts). Second, Ethereum moves away from the UTXO model in Bitcoin to an account-based system where account stores state. There are two kinds of accounts: External Owned Accounts (EOA) which are controlled by a private key, and smart contract accounts which work autonomously according to their own logic. The availability of smart contracts on Ethereum makes it one of the most widely used dApp blockchains with thousands of applications deployed, such as financial derivatives, exchanges, NFTs, gambling, and games. Smart contracts on Ethereum are like objects in an object-oriented programming language where state can be stored and functions can be called to change its state. Users can interact with smart contracts by sending messages to it, and smart contracts can also send messages to other smart contracts (invoking) to change their state. The smart contracts can enable very complex applications, and can enable some very powerful operations such as flash-loans or flash-swaps that have no analogy in non-blockchain applications. This is made possible by the powerful atomicity of transaction that invokes smart contract functions: it either completes or completely reverts. Over the years more and more blockchains such as Polkadot, Solana, Avalanche, and Cosmos have arisen and support nearly Turing-complete smart contracts.

2.3. Emergence and challenges of multi-chain

While some people tend to favor one chain to rule them all, the reality is that blockchain technology and markets are evolving at astonishing pace and it is becoming more and more apparent that the future of the ecosystem will be comprised of multiple blockchains serving their own purpose with their own tradeoffs in terms of security, decentralization, scalability, speed, cost, compliance, and so on. In this multi-chain future, a key limitation is that blockchains are designed to be a closed system. Transactions that happen on a blockchain can only rely on the state of their respective blockchain, and can only modify the state of their respective blockchain. External information cannot be reliably brought to the blockchain without a trusted third party (oracle). Transactions that involve multiple blockchains must go through a trusted party, such as a centralized exchange. As a result, there is currently no decentralized, permissionless, and public service that facilitates generic atomic transactions (not only atomic swapping, but also arbitrary logic) that involves multiple blockchains.

Popular cross-chain or cross-blockchain strategies include side-chains, relays, notary schemes, hash time-lock contracts, and blockchains of blockchains.

- First, side-chains/relays are popular solutions to implement bridges that primarily enable portable assets. In these, some assets have a home ledger that is authoritative on its ownership, but through bridges one can move the asset to other blockchains while being confident that the asset is able to move back to the home blockchain. Relay is one of the direct mechanisms to facilitate interoperability, where instead of relying on a trusted intermediary to provide information on chain A to chain B, chain B implements a thin client of chain A using smart contracts, and is able verify whether a particular event, transaction, or state at certain point in time has occurred on chain A. This is often called trustless because there is no additional trust assumption beyond trusting the two involved chains. In other words, no trust is required on the validity of the delivery mechanism of messages from chain A to chain B, other than that the message is delivered and delivered in time. Examples of relays include the BTCRelay on Ethereum (a SPV client of Bitcoin) and the Rainbow bridge of Ethereum on the NEAR blockchain. Relays are also popular mechanisms for side-chains.

- Second, notary schemes are mechanisms where a trusted entity (or a set of) is tasked with notarizing claims such as event X has happened on blockchain A. The most obvious notary schemes are centralized exchanges, which are trusted

Strategy	Use scenarios	Trust Assumption
Relay/Side-chain	Portable Assets	Trustless
Notary Scheme	Arbitrary	Trustful
HTLC	Atomic Swaps	Trustless
BoB	New blockchains	Trustless/Trustful

Table 1. Comparison of existing cross-chain strategies

entities to facilitate cross-blockchain exchanges of coins. Notary schemes do not have to be centralized; for example the Interledger project, in its atomic mode can be categorized as a decentralized, byzantine fault tolerant notary scheme to facilitate cross-ledger transfers. Note that notary schemes are the most flexible in terms of interoperability use cases, because they are able to act with arbitrary logic in response to events on discrete blockchains. Another notable decentralized notary scheme is THORChain which implements a DEX for native coins across several different chains, using a set of incentivized validators as notaries.

- Third, hash time-lock contracts (HTLC) are constructs of smart contracts that can facilitate atomic swaps across blockchain chains trustlessly without additional trust beyond the participating two blockchains. The key words are atomic and trustless. Atomic means that the transactions (involving two parties) are either complete or revert (as if nothing has happened). Trustless means no third-party needs to be trusted for the atomic swap. It works roughly by two parties interactively deploying and interacting with smart contracts on both sides. The core idea is with a hash of secret that is conceived by party A and used by both parties, and party A is forced to reveal the secret when claiming party B's coin, which can then be used by party B to claim party A's coin. Examples of HTLC include XClaim BTC/Ethereum or BTC/Polkadot bridge, and the Lightning Network on Bitcoin.

- Fourth, blockchains of blockchains (BoB) are frameworks that provide data, network, consensus, incentive, and contract layers for constructing application-specific blockchains that inter-operate between each other. Note that BoB does not solve current interoperability problems directly. Rather it enables the creation of new inter-operable blockchains. To connect to legacy chains, some sort of bridge or other mechanism shown above must be employed. Important examples of BoB are Polkadot and Cosmos, built on Substrate and Tendermint as consensus engines, and XCMP and IBC as cross-chain communication protocols.

Each of these broad strategies has its strengths and weaknesses in technical complexity, trust assumptions, level of interoperability, and use cases. Our discussion here is brief and incomplete, but still we can very roughly categorize the characteristics of these strategies; see Table 1 for a comparison of these strategies.

3. Interoperability Related Work

In this section, we pick some of the recent and most relevant projects, ideas, and trends to provide context for this paper and ZetaChain. For more academic cross-blockchain research please refer to a comprehensive survey [1].

3.1. Cross-chain Communication

A basic building block of any cross-blockchain interoperability is the ability to communicate and prove to chain B that a certain transaction happened on chain A.

BTCRelay [4], Rainbow Bridge [5]: Consider the task of building a one-way bridge on Ethereum from Bitcoin. When a user on Bitcoin sends 1 BTC to a given custody address, one wrapped BTC is issued on Ethereum. To do this in a trustless way, a smart contract on Ethereum can verify the transaction on Bitcoin, and issue a corresponding wrapped BTC coin on Ethereum. BTCRelay is such an example. For an Ethereum smart contract to verify the transaction on Bitcoin, someone (off-chain service) can submit the transaction, together with the transaction Merkle proof. The Ethereum smart contract verifies the proof based on the chain of block headers stored in the smart contract. This smart contract is essentially a light client of Bitcoin. Even though the strength of the proof is a bit lower than a full node (would be vulnerable to certain 51% attacks), this kind of bridge is strong and trustless, albeit rather expensive to operate because the chain of the block headers have to be constantly updated in the smart contract. The Rainbow Bridge is also a good example of a trustless bridge, between Ethereum and NEAR.

Wormhole [19]: Wormhole is also a cross-chain message delivery service, but it's not trustless. Rather, it depends on a set of validator nodes to attest the validity of the message delivered. Consider the same task of building a one-way bridge on Ethereum from Solana. When a user sends 1 SOL to a certain custody address, one wrapped SOL is issued on Ethereum. The Ethereum smart contract does not verify the transaction on Solana in order to issue the wrapped coin; it trusts that the super majority of the set of Wormhole validators are honest and correct. The security of Wormhole relies on the super-majority of the validators being honest. It appears that Wormhole relies on reputations of validators to build trust.

LayerZero [13]: LayerZero is a communication layer for facilitating cross-chain message delivery. It's essentially a weaker form of Relay (see introduction about Relay). The idea is to enable Chain B to verify that a given transaction or event has happened on Chain A. If Chain B supports general smart contracts, a light client of chain A can be implemented in a smart contract so as to verify information about Chain A in a trustless manner. However, even a light client can be expensive to run in a smart contract, both in terms of computation and storage; for example the BTCRelay on Ethereum appears to be discontinued. LayerZero reduces these costs with an ultra-light client on smart contract which does not report and store the whole chain of block headers (or a significant part of it). Rather, LayerZero relies on trusting a block header without a chain of block headers that can trace back to some known trusted block. The key assumption of the security of LayerZero is that the two parties—Relayer, who provides proof of transaction, and Oracle, who provides the block header—are non-colluding. In our terminology and categorization, LayerZero is not trustless due to the trust needed for the independence of the two parties. We use a stricter definition of trustless as where the validity (not necessarily liveness, censorship resistance) of messages does not depend on trust in anything other than the two participating blockchains. If the relayer and oracle collude, they can defraud LayerZero by making up an invalid block header (costs about 2 Ether to compute PoW nonce which is the coinbase reward of each block), and make chain B believe that a non-existent transaction has happened on chain A. LayerZero essentially outsources their security to third-party relayer and oracle.

IBC [10]: Inter-Blockchain Communication (IBC) protocol is a TCP/IP-like protocol for communication between sovereign blockchains. IBC is an end-to-end, connection oriented, stateful protocol between blockchains. Practically, IBC usually requires fast finality chains such as Tendermint, and the blockchain must support IBC protocol such as Cosmos SDK-built chains. For the blockchains that support IBC, they can establish connections, and through these connections one blockchain can verify proofs against the consensus states of another blockchain. Each blockchain that supports IBC must run a light client that is capable of verifying proofs on the other blockchain in order for them to be connected. The IBC module must also handle production of proofs, and a separate process (relayer) must relay the packet and proof to the counterparty chain. Among the blockchains that support IBC, very strong interoperability can be established, such as coin transfer, atomic swaps, cross-chain decentralized exchanges, and even cross-chain smart contracts. The major drawback of IBC is that it requires adoptionwhich is a lot to ask of other blockchains, and also might not be possible for legacy blockchains.

3.2. Cross-blockchain Asset Transfer

Hop [9]: Hop is a protocol to send coins across rollups and their underlying L1 in a trustless manner. Rollups are by default siloed systems and the asset transfer between rollups and L1 can be slow and expensive. For example, optimistic rollups usually take a week to exit into L1; on the other hand, zk-rollups can instantly validate exit but it involves high computation which is expensive on L1. Hop solves the problem of moving coins across rollups by creating bridges and bridge coins, and uses AMM markets to exchange coins rather than sending coins directly. Specifically, Hop creates bridge coins for each rollup, and the bridge coins can be moved around in batches so as to decrease the cost. The bridge coin acts as an intermediary asset in transferring a coin on rollup A to rollup B. Hop uses the existing rollup bridges to do cross-rollup transactions so it does not need a separate off-chain service.

Connext [3]: Connext is a trust-minimized solution for cross-chain asset swaps. The idea is somewhat like generalized atomic-swaps, using Hash Time Locked Contracts (HTLC) to ensure transaction atomicity. It uses a network of off-chain routers to create a market and AMM style pricing mechanism. The safety of user funds do not depend on third-parties, only the liveness of the system does. Compared to Hop, Connext uses off-chain services and therefore can connect beyond rollups on a single L1; compared to externally verified solutions, Connext is application specific and not general purpose. For example, it cannot be adapted to send arbitrary messages or cross chain contract calls.

Multichain [17]: Multichain (previously Anyswap) is a cross-chain bridge and cross-chain router network. The network consists of smart contracts on connected chains, and the Fusion network. The key technology is distributed TSS key among MPC nodes, and DCRM (Distributed Control Rights Management) [6]. The TSS key scheme used in Multichain is GG20 [8], the same as THORChain and ZetaChain. The DCRM Multi-party Interoperability and Custody is a decentralized custodian technique. DCRM consists of two important functions: Lock-In, and Lock-Out. In Lock-in, a user locks an external asset (such as Bitcoin), and the system generates a wrapped version of Bitcoin owned by a newly generated distributed TSS private key. The system generates an address on Bitcoin, and the user transfers Bitcoin to that address. When the transfer completes, the Fusion node picks up the confirmation on the Bitcoin network, and issues a wrapped version of BTC to the user on the Fusion network. The Lock-In is thus complete. The Lock-Out process is similar but in reverse. It appears that Multichain is a bridge that locks coins on connected chains and wraps them on the Fusion blockchain. Multichain can therefore be considered as a centralized bridge.

THORChain [15], Sifchain [18], Chainflip [16]: THORChain (along with similarly built competitors like Sifchain and Chainflip) is a decentralized liquidity network that facilitates AMM style native L1 coins on different blockchains, including Bitcoin, Litecoin, Bitcoin Cash, Ethereum. Notably, THORChain is not, strictly speaking, a bridge, as it does not lock & wrap coins and transact on wrapped coins. Rather, THORChain is an application-specific blockchain that maintains the pool, logic, and management of vaults on different chains for swapping. THORChain distributes the signing key using the GG20 TSS scheme and has its own implementation based on Binance's TSS library. ZetaChain is in-part inspired by the design of THORChain, and can be thought of as a simpler and more generalized platform which enables not only swapping, but a generic smart contract platform that allows arbitrary cross-chain applications to be built easily. For example, developers can implement similar functionality to THORChain as a smart contract on ZetaChain.

Synapse [14]: According to public information, Synapse is supposed to be an externally verified validator set based system for cross-chain swaps. It issues AMM smart contracts on external chains, and some composite stablecoin as an intermediary asset to cross-chain. To move the intermediary stablecoins across chains it appears to use a burn and mint strategy. Detailed public information about their validator mechanism is not available at the time of writing this paper.

3.3. Cross-blockchain Smart Contract

Quant Network [20]: Functionality-wise, the Quant network and its Overledger [20] is the closest to ZetaChain. The Quant network is a centralized service that provides a standardized web-service-based access to the connected public or private blockchains, or regional legacy database ledgers. It supports general programmability triggered by events on those blockchains (transaction to/from a given address, smart contract interaction, events, state changes, etc.), via popular languages and frameworks such as Javascript, Java, Python, etc. ZetaChain aims to achieve similar general programmability, but with an incentivized public blockchain, with far reduced trust assumptions, more transparency, complete verifiability and auditability.

ICP/Chain-Key [2]: The Internet Computer Protocol (ICP) has proposals to enable interoperability to the Bitcoin network via its Chain-Key technology, which is similar to the distributed threshold signature scheme. With Chain Key, ICP in principle can custody funds on the Bitcoin network. It's unclear how ICP observes the Bitcoin network, and how their smart contract platform interacts with external blockchains.

HyperService [11]: HyperService proposes a cross-chain smart contract platform that is chain agnostic. It consists of two components: a high level language HSL to describe a cross-chain dApp, and an execution layer that ensures financially atomic transactions.

3.4. Blockchain of Blockchains (BoB)

The most prominent BoBs are Cosmos and Polkadot. BoBs are usually frameworks that aim at tight inter-operable application-specific blockchains. Polkadot, for example, provides a relay chain which handles all consensus, and Parachains which can be different blockchains with different state-transition functions. The Parachains are tightly integrated and can inter-operate seamlessly via the relay-chain.

The Cosmos ecosystem, on the other hand, does not share consensus, so the interoperability between Cosmos chains is less tight. Every Cosmos chain is sovereign with their own choice of consensus (typically Tendermint-based fast finality). The Cosmos ecosystem relies on the IBC protocol (see section 3.1), and special blockchains called Hubs to facilitate cross-chain asset transfers, and even cross-chain smart contracts.

To enjoy interoperability in Cosmos or Polkadot, the blockchains typically need to be built on some common ground. Legacy blockchains, or new blockchains with their own consensus, cannot be part of BoBs.

4. ZetaChain Blockchain Architecture

4.1. Overview

At a high level, ZetaChain is a Proof of Stake (PoS) blockchain built on the Cosmos SDK and Tendermint PBFT consensus engine. As a result, ZetaChain enjoys fast block time (~5s) and instant finality (no block confirmation needed, no re-organization allowed). The Tendermint PBFT consensus engine has been demonstrated to scale to ~300 nodes in production, and with future upgrades with BLS threshold signatures the number can potentially increase to 1000+. The throughput of transactions of the Tendermint consensus engine that ZetaChain uses can reach 4000+ transactions per second (TPS) under ideal network conditions [10]. Note that the cross-chain TPS cannot reach nearly as high because cross-chain transactions latency/throughput may be limited by external chain latency/throughput, TSS key-sign throughput, and various other factors such as external node RPC speed, etc.

The ZetaChain architecture consists of a distributed network of nodes, often referred to as validators. Validators act as decentralized observers that can reach consensus on relevant external state and events, and can also update external chain state via distributed key signing. ZetaChain accomplishes these functions in a decentralized (without a single point of failure, trustless, permissionless), transparent, and efficient way. Contained within each validator is the ZetaCore and ZetaClient. ZetaCore is responsible for producing the blockchain and maintaining the replicated state machine. ZetaClient is responsible for observing events on external chains and signing outbound transactions. ZetaCore and ZetaClient are bundled together and run by node operators. Anyone can become a node operator to participate in validation provided that enough bonds are staked. See Figure 2 for a high level illustration.

Validators: ZetaChain uses the Tendermint consensus protocol which is a partially synchronous Byzantine Fault Tolerant (BFT) consensus algorithm. Each validator node can vote on block proposals with voting power proportional to the staking coins (ZETA) bonded. Each validator is identified by its consensus public key. Validators need to be online all the time, ready to participate in the constantly growing block production. In exchange for their service, validators will receive block rewards, and potentially other rewards such as gas fees or processing fees, proportional to their bonded staking coins.

Observers: Another set of important participants of ZetaChain consensus are the observers who reach consensus on external chain events and states. The observers watch externally connected chains for certain relevant transactions/events/states at particular addresses via their full nodes of external chains. The observers can be further divided into two roles: sequencer and verifier. The sequencer discovers relevant external transactions/events/states and reports to verifiers; the verifiers verify and vote on ZetaChain to reach consensus. The system requires at least one sequencer and multiple verifiers. The sequencer does not need to be trusted, but at least one honest sequencer is needed for liveness.

Signers: The ZetaChain collectively holds standard ECDSA/EdDSA keys for authenticated interaction with external chains. The keys are distributed among multiple signers in such a way that only a super majority of them can sign on behalf of the ZetaChain. Its important to ensure that at no time is any single entity or small fraction of nodes able to sign messages on behalf of ZetaChain on external chains. The ZetaChain system uses bonded stakes and positive/negative incentives to ensure economic safety.

In practice, all above roles (except sequencer) are collocated in the same computer node, sharing software and credentials such as validator keys and bonded stakes and the associated rewards/slashing. ZetaChain is planned to transition from Proof-of-Authority at first to a fully delegated Proof-of-Stake (DPoS) model over time, and gradually delegate the governance of the blockchain to ZETA coin holders via on-chain voting.

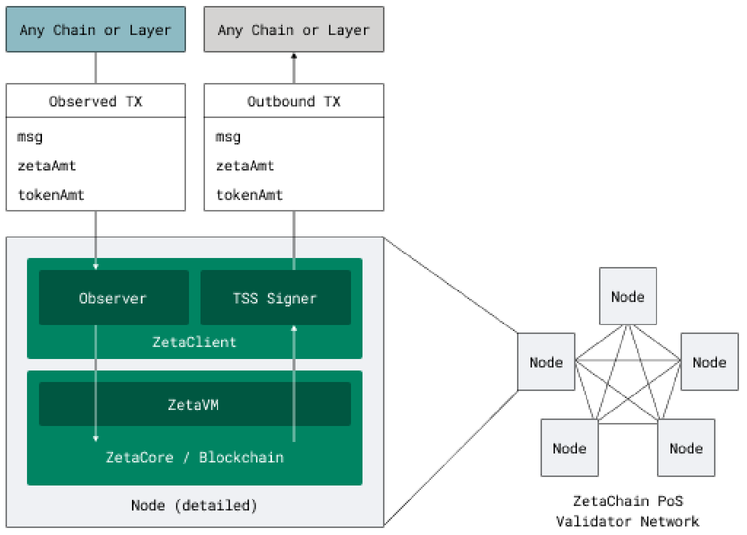

Figure 2. ZetaChain High Level Architecture.

4.2. Observers

Observers are tasked with monitoring external chains for relevant transactions. Observers are continually scanning for external chain events responsible for both burning and minting the native coin (ZETA), messages & smart contract calls, as well as other events that dApps register on ZetaChain. Each observer independently observes using its own full node of external chains, and all the observations must reach consensus on the ZetaChain before being considered finalized. Once events are finalized, it automatically triggers an execution of ZetaChain logic, which can be defined as a custom Cosmos SDK module, or ZetaChain native smart contract.

There are two modes of observation: Active and Passive mode. Active observation constantly scans the external blockchains for relevant transactions/events/states. Passive mode relies on a sequencer (or a small set thereof) to scan and report transactions/events, together with Merkle proof. The observers verify the proof and reach consensus on the verification on-chain. The active mode has the advantage of being always live and censorship-resistant due to decentralization, but the cost of each node is high because it needs full nodes (of external chains) for the scanning. Passive mode is much less costly, as verification can be done with a light client. Only one or a few sequencers need access to a full node, which is much cheaper and makes scaling to multiple external chains and more validator nodes much easier. The disadvantage of passive mode is that the liveness of external chain inbound observation is dependent on the sequencer, and also subject to censorship by the sequencer. This is the same situation as the optimistic rollup where the liveness of the rollup is dependent on a sequencer. To mitigate this, everyone is able to be a sequencer if they so choose, and a sequencer can be incentivized by the creation of a competitive market. In particular, dApps have a vested interest in running a sequencer. Another advantage of running passive observation mode with a sequencer is that the dApps are in control of the observation ordering. For efficiency reasons, the active mode does not enforce observation ordering. But if the observation ordering is important to a dApp, it can opt to run its own sequencer in synchronous observation mode (i. e. wait for each observation to be finalized by ZetaChain before moving on to the next).

4.3. Multi-party Threshold Signature Scheme

ZetaChain needs to hold an account on external chains in order to custody funds on that chain (manage a pool, vault, etc.), and to perform privileged actions (burn, mint, move funds out of the vault, etc.). This is required for general-purpose cross-chain smart contracts, as a core feature of smart contracts is to manage assets autonomously. On Ethereum for example, a smart contract has an address and can hold any asset like an External Owned Address (EOA, normal user account). This ability enables many powerful applications such as AMM pools, lending/borrowing pools, etc., where users pool their assets and let smart contracts manage them according to a smart contracts predetermined rules. In order to hold an account, ZetaChain needs to have a private key. To avoid a single point of failure (single location of the private key, single dealer in generating the key), ZetaChain needs a distributed threshold signature scheme.

This is also needed to support non-smart-contract chains such as Bitcoin, Dogecoin, or smart-contract platforms that are expensive to verify multi-sig. To avoid any single point of failure, ZetaChain uses state-of-the-art multi-party threshold signature

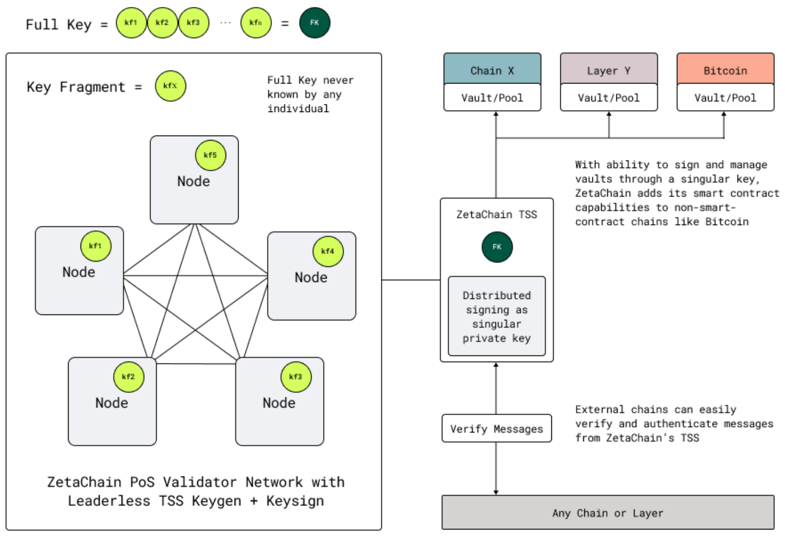
Full Key = kf1 kf2 kf3 ··· kfn = FK
Key Fragment = kfx
Full Key never known by any individual
Node
kf1
kf2
kf3
kf4
kf5
ZetaChain PoS Validator Network with Leaderless TSS Keygen + Keysign
Chain X
Layer Y
Bitcoin
Vault/Pool
With ability to sign and manage vaults through a singular key, ZetaChain adds its smart contract capabilities to non-smart-contract chains like Bitcoin
ZetaChain TSS
FK
Distributed signing as singular private key
Verify Messages
External chains can easily verify and authenticate messages from ZetaChain's TSS
Any Chain or Layer

Figure 3. Leaderless TSS Keygen and Keysign Overview

scheme (TSS) [7, 8] based on implementations from THORChain TSS [15] and Binance tss-lib [12]. To the outside world, the ZetaChain validators collectively possess a single ECDSA/EdDSA private key, public key, and address, and the signature signed by ZetaChain can be verified efficiently and natively by standard ECDSA/EdDSA verification procedure by the connected blockchains. Internally, the private key is generated without a dealer, and the private key is distributed in all the validators. At no time is a single entity or a minority of validators able to piece together the private key and sign messages on behalf of the whole network. The key generation and signing procedures are done by Multi-Party Computation (MPC) which reveal no secret of any participating node. Because ZetaChain can hold a TSS key and address, ZetaChain can support smart contracts that can manage native vaults/pools on connected chains including Bitcoin. This effectively adds smart contract capabilities to the Bitcoin network, and potentially other non-smart contract blockchains. The TSS employed by ZetaChain gives the performance and convenience of hot wallet with cold wallet level security. See Figure 3 for an illustration.

To sign in a decentralized manner, ZetaChain employs a multi-party t, n-threshold ECDSA scheme based on [7]. This leaderless Threshold Signature Scheme (TSS) performs key generation and signing in a distributed fashion. That is, no single validator or outside actor has access to the *complete* private key at any point in time, and no private information is leaked in key generation or signing. For efficiency, ZetaChain employs batched signing and parallel signing to improve signers throughput.

4.4. Cross-Chain Smart Contract and Zeta Virtual Machine

The ZetaChain hosts an Ethereum Virtual Machine (EVM) compatible execution layer called zEVM. Aside from supporting all features of EVM and normal interactions with EVM (contract creation, contract interaction, composition of contracts, etc), the distinguishing feature of zEVM is that

- contracts on zEVM can be called from external chains
- contracts on zEVM can generate outbound transaction on external chains

These two additional features make the zEVM a general purpose programmable platform that supports the notion of *cross-chain transactions* that alter states in different chains *atomically* and in *a single step.*

4.4.1. Challenges in General Purpose Cross-Chain Transaction

There are two key challenges in designing a general-purpose cross-chain transaction platform: asynchrony and atomicity.

The first challenge is that communication between chains is necessarily via message passing and inherently asynchronous between heterogeneous chains. This means unlike smart contracts on a single chain (such as EVM), querying or changing the state of another chain is asynchronous. This precludes the common convenient synchronous function calls from cross-chain smart contracts. The cross-chain smart contract programming model thus is best considered as a finite state machine, where state change is triggered by the messages (observations) from external chains. The app contract thus will be structured as distributed event-drive state machine triggered by messages. This is quite a more complicated programming model from the synchronous model of single chain smart contract.

The second challenge is the atomicity of cross-chain transactions. As cross-chain transaction involves altering states on multiple chains, if one part of the state change fails, all previous state changes need to be reverted. Blockchains reverts are powerful mechanisms to maintain atomicity, but no blockchain is built with consideration such that revert is conditional on what happened on another blockchain. To maintain cross-chain transaction atomicity, any cross-chain solution must adequately handle reverts, otherwise cross-chain applications will be too onerous to reason about and build.

In this paper we explore a viewpoint of hybrid UTXO and account-based approach, playing to the strengths of each. Essentially, we use UTXO to represent and track external blockchain transactions, and use account-based smart contracts for logic and managing shared global states. We treat observed external events as a synthetic UTXO.

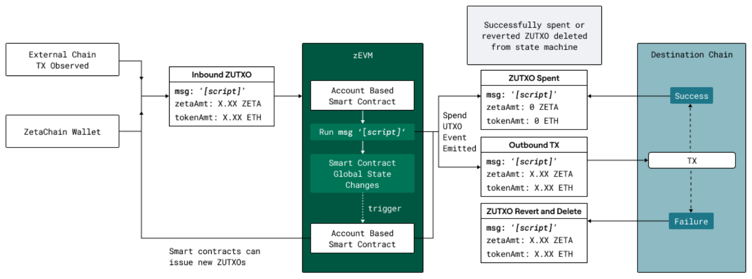

Figure 4. Hybrid UTXO-account flow.

A UTXO includes the amount of ZETA coin (burned), amount of another coin (optional, for example, BTC on the Bitcoin network where it's impossible to issue ZETA coin), and a script msg (roughly equivalent to a message or function call on Ethereum). The smart contract on ZetaChain runs the msg and generates an Event that tries to spend the UTXO on ZetaChain. The Event is then picked up by ZetaClient signers and they will sign a transaction to an external chain. The ZetaChain Virtual Machine and ZetaClient will validate certain invariants, one of which is that the output ZETA must be equal to the input ZETA in the UTXO. Once the outbound transaction is confirmed and observed, the UTXO is marked as spent and deleted from the state machine. If the outbound transaction fails (insufficient gas, etc.), the UTXO is marked as revert and refunds of ZETA and/or associated coins are refunded on the source chain. When the refund is confirmed then the UTXO is deleted from the state machine. See Figure 4 for an illustration.

We use the synthetic UTXO model for its accountability, simplicity, and scalability while avoiding the key limitation of UTXO which is the expressiveness of its scripting, and awkwardness in certain important applications (one TX per block in AMM).

4.4.2. Mechanism 1: Cross-Chain Message Passing

In the Cross-Chain Message Passing (CCMP) mechanism, the ZetaChain and zEVM serve only as a relayer of messages (cross-chain contract calls) and value (multi-chain token ZETA). The general programmability of zEVM is not necessary (except in specialized cases help handling converting gas) for this mechanism.

The ZetaChain CCMP style handles reverts by creating a "revert" transaction that undoes the committed state changes that needs to be reversed. The responsibility of the "revert" transaction is shared among the ZetaChain protocol and app contract; the protocol is responsible to initiate the revert transaction if needed, and refund the value (in ZETA token only, as its supply is controlled by the ZetaChain protocol). The app contract is responsible to do app level reversing the state; for example, in a cross-chain swap app, reverting a committed swap tx from user means swapping ZETA back to the asset that user put in, and send back to the user.

The workflow of a CCMP style cross-chain transaction works like this.

1. An end user interacts with a Contract C1 on Chain A.
2. The interaction leaves an event or transaction memo, with user specified [`chainID`, `contractAddress`, `message`]. (the `message` is arbitrarily encoded application data in binary format.
3. ZetaChain observers (in `zetaclient`) pick up this event/memo and report to `zetacore`, which verifies the *inbound transaction*.
4. `zetacore` modifies the CCTX (cross-chain tx) state variable with `OutboundTxParams` which instructs the TSS signers (in `zetaclient`) to build, sign, and broadcast external transaction.
5. The `zetaclient` TSS signers observe the `OutboundTxParams` in the CCTX, and build outbound tx, enter into a TSS keysign ceremony to sign the transaction, and then broadcast the signed transaction to the external blockchains. For CCMP, the *outbound transaction* is mainly calling the user specified contract with specified addresses and parameters.
6. The CCTX structure also tracks the stages/status of the cross-chain transaction.
7. Once the broadcasted transaction is included in one of the blocks (said to be "mined" or "confirmed"), `zetaclients` will report such confirmation to `zetacore`, which will update the CCTX status.
8. If the "confirmed" outbound transaction was successful, the CCTX status becomes `OutboundMined`, and the CCTX is considered in its terminal state and will not be updated anymore. **This CCTX processing is completed**.
9. If the "confirmed" outbound transaction is failure (e.g. revert on Ethereum chains), then the CCTX will updates it status to `PendingRevert` if possible, or `Aborted` if revert is not possible. **The CCTX processing is completed if it goes to `Aborted` status**.
10. If the new status is "PendingRevert", a second `OutboundTxParams` should be already in the CCTX, which instructs the `zetaclients` to create a "Revert" outbound tx to the incoming chain & contract, allowing the incoming contract to implement a application level revert function to cleanup contract state.
11. The `zetaclients` will build the revert transaction, enter into TSS keysign ceremony to sign the transaction, and broadcast to the incoming blockchain (Chain A in this case).
12. Once the revert transaction is "confirmed" on Chain A, the `zetaclients` will report the transaction status to `zetacore`.
13. If the revert transaction is successful, the CCTX status becomes `Reverted`, and **the CCTX processing is completed**.
14. If the revert transaction is failure, the CCTX status becomes `Aborted`, and **the CCTX processing is completed**.

To start a CCMP, one initiates a transaction that directly or indirectly calls the `Connector.send()` function on the supported smart contract chains:

```
interface ZetaInterfaces {
  struct SendInput {
      uint256 destinationChainId;
      bytes destinationAddress;
      uint256 destinationGasLimit;
      bytes message;
      uint256 zetaValueAndGas;
      bytes zetaParams;
  }
}

interface ZetaConnector {
    function send(ZetaInterfaces.SendInput calldata input) external;
}
```

As can be seen from the interface, the sending party (could be an user EOA directly, or through an application contract indirectly) specifies the destination chain, the destination contract on the destination chain, and the parameters to call a specific function of that contract (`message`).

On the receiving side (the destination chain), the contract specified on the sending side will be called; specifically, the following function must be present in the receiving contract with exactly the signature:

```
interface ZetaInterfaces {
  struct ZetaMessage {
      bytes zetaTxSenderAddress;
      uint256 sourceChainId;
      address destinationAddress;
      uint256 zetaValue;
      bytes message;
  }
}

interface ZetaReceiver {
  function onZetaMessage(ZetaInterfaces.ZetaMessage calldata zetaMessage) external;
}
```

The `onZetaMessage()` function will be called by the ZetaChain protocol with input `ZetaMessage` struct filled in with appropriate values. Notably, the `zetaTxSenderAddress` will be the sending party address (could be EOA or the contract that calls `Connector.send()`); the `zetaValue` will be the ZETA token sent (burnt) from the sender side, less some transaction fees; message will be the message that user specified.

The application contract will receive `zetaValue` amount of ZETA token, and can implement business logic in this `onZetaMessage()` function. If this `onZetaMessage()` is successful, the this message passing CCTX is completed; if it failed (reverted), then 9th step above will be invoked, and the `zetaTxSenderAddress` will be interpreted as a contract, and it must implement the following function `onZetaRevert()`.

```
interface ZetaInterfaces {
  struct ZetaRevert {
      address zetaTxSenderAddress;
      uint256 sourceChainId;
      bytes destinationAddress;
      uint256 destinationChainId;
      /// @dev Equals to: zetaValueAndGas - ZetaChain gas fees -
      /// destination chain gas fees - source chain revert tx gas fees
      uint256 remainingZetaValue;
      bytes message;
  }
}

interface ZetaReceiver {
  function onZetaRevert(ZetaInterfaces.ZetaRevert calldata zetaRevert) external;
}
```

Again, the `onZetaRevert()` function of contract at `zetaTxSenderAddress` address will be invoked with ZetaChain filled struct `ZetaRevert`. The remaining ZETA token will be refunded back to this contract, and this contract should implement application level revert since the cross-chain contract call failed. Such revert might include refunding user, cleanup its state, etc.

4.4.3. Mechanism 2: Omni-Chain Smart Contract

Note that in the Mechanism 1 (CCMP) style, each participating blockchain needs to support smart contract (this precludes Bitcoin, Dogecoin, etc), and the dApp needs to deploy at least one contract to each chain. The application state and logic is scattered around all those application contracts in a distributed manner. Synchronizing the state and communicate between contracts on different chain becomes expensive, slow, and complicates the handling of reverts.

To reduce the complexity and surface area between different chains, ZetaChain introduces Mechanisms 2, also called omni-chain smart contract. The distinguishing features of omni-chain smart contract are that omni-chain smart contract

- is arbitrarily programmable, like a regular smart contract on zEVM
- manages foreign assets on external chains directly
- can be called *from external chains*

The most important foreign assets on external chain are fungible tokens, such as native gas assets (Ether, BNB, Matic, or Bitcoin) and regular ERC20 tokens (USDT, USDC, etc).

On zEVM, a smart contract can directly custody these foreign assets and manipulate them as if they are native zEVM assets. On external chains, these fungible tokens are controlled by the TSS address; internal to zEVM, they are represented as ZRC20–a ERC20 compatible contract but with interfaces to handle deposit and withdraw–which act as the ledger to keep track of the native assets that ZetaChain holds.

Any smart contract on zEVM can become omni-chain via interacting with these ZRC20 contracts and implement a simple interface to be callable from external chains:

```
interface zContract {
    function onCrossChainCall(
        address zrc20,
        uint256 amount,
        bytes calldata message
    ) external;
}
```

Calling a omni-chain smart contract from external chains:

1. A user sends native asset to the TSS address on Chain A (for ERC20, the user needs to send to a `ERC20Custody` contract controlled by the TSS address), with a memo specifying [`zEVMContractAddress`, `message`].
2. The observers in `zetaclients` observe the incoming omni-chain call and report to `zetacore`.
3. `zetacore` invokes the `depositAndCall()` function of the `SystemContract`, which in turn calls the `onCrossChainCall()` function of the specified address `zEVMContractAddress`. The ZetaChain protocol (more specifically the `fungible` module) fills in the appropriate parameters in the `onCrossChainCall()` call:
    - `zrc20` will be filled with the ZRC20 contract address that manages the foreign coin that the user sent in step 1;
    - `amount` will be filled with the amount of the foreign coin that the user sent in step 1;
    - `message` will be the `message` in the memo of the transaction that user sent in step 1;
4. The omni-chain smart contract will be called in the following way

```
zContract(zEVMContractAddress).onCrossChainCall(zrc20,amount,message)
```

5. The app contract should implement its business logic in the zEVMContractAddress function, with entry point being `onCrossChainCall()` function. **This omni-chain smart contract interaction is completed** if the execution does not revert and there is no output of external asset.
6. If the zEVM contract execution failed (reverted), then a CCTX is created to revert the inbound tx; namely sending the foreign coin back to the user (the same coin, with same amount as user sent in step 1, less applicable fees).
7. If there is an output of `onCrossChainCall` (for example, it calls some ZRC20 `withdraw()`), then a separate CCTX will be created to instruct and track transferring foreign asset to user specified address on external chains. These withdrawals are simple token transfers and should not fail (in exceptional cases such as the destination address is blacklisted by USDT, the CCTX will transition into `Aborted` state and no revert or refund will be attempted.

Note 1: if the app contract does not need to be called from external chains, then it does not need to implement the `zContract` interface.

Note 2: As demonstrated above, the capability of omni-chain smart contract depends on omni-chain primitive contracts such as ZRC20, so it's limited to what omni-chain primitives are provided. ZetaChain provides ZRC20 that manages all fungible foreign tokens, and may introduce further asset types such as non-fungible token, non-transferable token, digital identity, etc.

Note 3: the omni-chain smart contract is deployed only on zEVM, with all the logic and states in a single place. Moreover, the app contract does not need to handle reverts, because all reverts are handled by ZetaChain protocol. Therefore omni-chain smart contract addresses both of the two major challenges in general cross-chain transaction; namely asynchrony and scatter of state and logic into multiple contracts on multiple chains, communicated only via messages. As a result, omni-chain smart contract dApps are much easier to build than message passing ones.

Note 4: omni-chain smart contract does not require deploying app smart contract on external chains, therefore can support non-smart contract chains such as Bitcoin network. Omni-chain cross-chain apps can be initiated on Bitcoin and can also settle on Bitcoin.

4.4.4. Omnichain Smart Contracts vs. Messaging

While both mechanisms can support many types of applications, they offer fairly significant differences in the architecture those applications would adopt.

More complicated dApps may prefer Omnichain Smart Contracts because the logic & state is in a single place, whereas with messaging, you must broadcast messages and

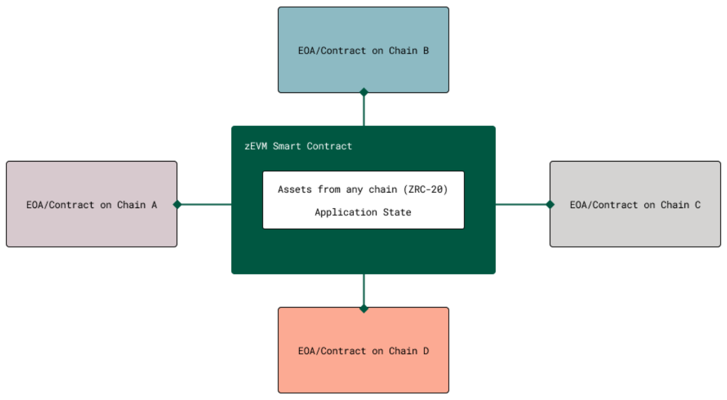

Figure 5. Omnichain smart contract-based application. Note that there is a single contract that receives input, writes output, maintains state, and orchestrates external assets for the application. The number of external transactions required for an omnichain dApp increases only based on the required outbound transactions, like withdrawing assets to an external chain's address

state sync across many contracts on different chains, which can lead to more attack surface and more gas fees (each message requires additional gas to be paid, and the number of messages you must send to maintain a full state sync scales). In other words, Omnichain Smart Contracts behave, for developers, as if all assets were on one chain (see Figure 5).

Common applications like Uniswap V2/V3, Curve, Aave, Compound, and so on that have been audited and battle-tested on Ethereum/EVM can easily be deployed and built on top of in ZetaChain's Omnichain Smart Contracts. One can extend these applications by adding in compatibility with ZRC-20, but those changes are minimal and the majority of logic may remain the same, and users may interact with these applications in single-step transactions just as they would on Ethereum (or by calling them from external chains). On the other hand, with messaging, in many situations (especially those that are more complex), a developer must recreate the logic in a completely different, asynchronous messaging and state-sync system; messaging cannot leverage existing work in the same way.

ZRC-20 can easily support Bitcoin/Cardano/XRP which do not have capability or efficiency to support general purpose smart contracts for applications like swapping, lending, etc. Messaging cannot work for these chains, because messaging requires smart contracts on any connected chain.

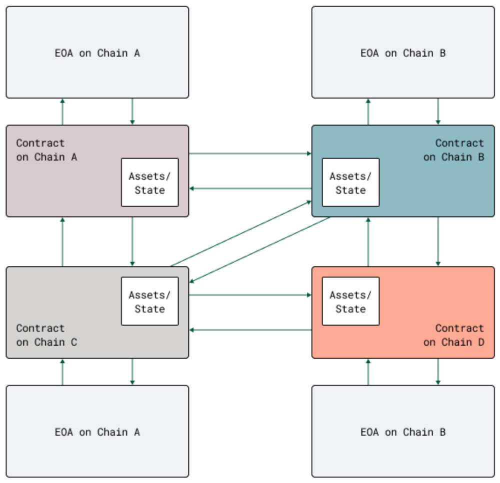

Figure 6. Messaging-based application. Note that for contracts to stay in sync across connected chains, the number of messages required increases exponentially with the number of chains involved.

Messaging generally makes sense in simpler use cases between 2 or just a few chains, or where state should heavily be based in one chain, and sent or interacted with from other chains. Application-specific bridges, for example, where the goal is simply to get data/value into one chain, could make sense to build with messaging. Applications that must utilize contracts on external chains may also need a messaging-based component. For more complex applications, the number of messages (and thus gas/transactions) required to synchronize state across multiple chains can increase exponentially with the number of chains involved (see Figure 6). For example, managing a vault or lending protocol with assets across many chains could be difficult to manage with just messaging.

Message passing style logic and state are distributed on asynchronous chains which adds significant complexity to maintaining cross-chain transaction atomicity, and forces dApps to program in an event (message) driven way that is generally harder to do than synchronously in a single chain. Omnichain smart contracts on the other hand offer the novel ability to develop multichain applications in a more synchronous, atomic environment as if they were on one chain.

4.4.5. Fees & Gas

To prevent spam and ensure fair and efficient use of the blockchain resources (compute and storage), the user must pay proper fees for processing the cross-chain transaction.

Unlike transaction on a single chain, a cross-chain transaction naturally might involve several different gas assets and need to pay more than one types of tokens for gas fees. This is rather inconvenient, and may add undue operational cost or risk to operate the cross-chain solution. For example, if one invokes a contract on Ethereum from BSC chain, the user needs to pay both BNB and Ether as gas fees; but how can the user pay Ether on BNB? Do they need to acquire "wrapped" Ether on BSC? Which version of the wrapped Ether? Who wraps and unwrap the Ether?

Alternatively, one might just ask the user to pay in a single asset (for example, only BNB), and then some off-chain service converts the BNB into Ether to reimburse the protocol which needs to pay Ether for the outbound tx processing. This is quite an operational burden, and runs counter to the autonomous nature of sovereign blockchain that does not need centralized operator.

ZetaChain completely automates the gas handling and conversion on-chain, and with market force to maintain proper conversion rate. Also, the conversion of different gas assets are synchronous with the CCTX itself so the settlement is as fast as possible. The way ZetaChain does it is to rely on ZRC20 and their AMM pools on zEVM (currently Uniswap v2 pools). All gas asset has a corresponding ZRC20 which pairs with ZETA (native gas token on zEVM) on zEVM.

Let us consider the two cases when the user need to pay gas fees in coins they might not have:

- In message passing, the user pays a single asset (ZETA token) for all gas fees. The ZetaChain protocol converts proper amounts of ZETA into outbound chain gas asset ZRC20 synchronously and use the balance to pay outbound transaction gas fees.
- In omni-chain smart contract ZRC20, when a user (or a smart contract) wishes to withdraw the foreign asset, the user will need to pay the outbound gas fee. The withdrawing smart contract can acquire the outbound chain gas asset ZRC20 from the internal AMM pools on zEVM to pay gas synchronously.

In either case, the multi-gas handling of ZetaChain is sound (which means that the protocol always has enough gas asset to pay outbound tx gas fees), and the conversion rate is determined by market force. As zEVM ZRC20 assets are easily withdrawn to external chain with on-chain contracts, the markets on zEVM are connected with other markets therefore we can expect market forces to maintain price parities.

5. ZETA Token

ZETA token is a multi-chain utility token that are essential in the funtions of the ZetaChain blockchain and its cross-chain infrastructure.

1. Securing the DPoS of the ZetaChain conensus via staking/delegation/slashing.
2. Anti-spam and ensuring fair and efficient use of blockchain resources such as compute and storage
3. Universal gas asset used to pay gas fees on multiple chains
4. Represent value that can transfer from one blockchain to another

6. Use Cases & Applications

In this section we discuss some sample applications of ZetaChain. These examples are not anywhere near comprehensive, since the general smart contract and interoperability capabilities of ZetaChain provide a platform for virtually unlimited creativity in terms of omnichain application-building.

6.1. Smart Contract Managed External Assets

A powerful feature of smart contracts is that smart contracts can hold any assets that a normal account can hold, and are able to receive and spend that asset according to programmed logic. However, important blockchains like Bitcoin, Dogecoin, Monero, etc., do not have general enough smart contract capability to support useful applications such as AMM exchanges, collateralized borrowing/lending markets with pools, and the like. There is currently no way to involve native Bitcoin (without wrapping) in arbitrary logic in a decentralized and permissionless manner. The cross-chain smart contract capability of ZetaChain can hold and use assets on external chains directly, therefore enabling smart contract managed native Bitcoin on ZetaChain, among other native assets such as ETH, ERC20, Algorand ASAs, etc. Furthermore, through ZetaChain smart contracts and with message passing, cross-chain dApps can be easily composed with smart contracts on the participating chains, with ZetaChain smart contracts managing native Bitcoin vaults.

Let us look at an example in some detail. The mechanism for ZetaChain smart contracts to manage BTC on Bitcoin is as follows. The initialization of smart contract requests `KeyGen` to generate a TSS key which acts as the address of a Bitcoin vault. The ZetaClient will monitor the TSS address and upon identifying incoming transactions to the TSS vault, it parses the data from the Bitcoin transaction in OP_RETURN and invokes the `zetaProcess` function with the parsed data on the smart contract. The smart contract takes actions accordingly (such as credit to certain accounts, sending out another asset according to AMM pricing, etc.). To send out Bitcoin from the smart contract, the smart contract emits a specific `Event` that the ZetaClient will pick up and sign & broadcast to Bitcoin network. The smart contract must also implement a function `zetaExternalTxConfirm` which will be invoked when the outbound external chain transaction is mined.

6.2. Cross-chain AMM Exchanges

ZetaChain can enable true cross-chain AMM decentralized exchanges, built on top of smart contracts. There are two ways of constructing an AMM DEX on ZetaChain: message passing and native ZetaChain smart contracts. The key difference is whether the pool is managed by an external smart contract or native ZetaChain smart contract. With message passing, the asset pool is managed by smart contracts on external chains; with the native ZetaChain smart contract approach, the pool is managed by ZetaChain smart contracts through a TSS account.

Specifically, in message passing, the assets are managed by smart contracts on external chains, paired with a ZETA coin. A swap of asset X on chain A for asset Y on chain B can be accomplished by: 1) swap X for ZETA on chain A using smart contract managed pool and AMM; 2) pass message, together with the ZETA coin from chain A to chain B; 3) chain B smart contract managed pool (Y/ZETA) swaps ZETA coin for Y.

With native ZetaChain smart contracts, the ZetaChain TSS account holds all the native assets on external chains, which can be managed by ZetaChain contracts directly. The ZetaChain smart contract implements AMM logic that determines pricing, swap, liquidity providers, and fees.

In the message passing approach, the dApp states and logic are spread across all the external chains; ZetaChain only acts as a message verifier and relayer. The advantages in this approach is that existing infrastructure can be reused (for example, on EVM chains Uniswap contracts can be reused to manage pool X/ZETA), and the dApp needs only to handle the cross-chain messaging to implement conditional execution. On the other hand, in the native ZetaChain smart contract approach, the logic and state of the dApp lives on ZetaChain, a single platform with a unified interface to interact with external chains. The advantages in this approach are the ease of dApp development (minimal development efforts in accommodating new chains), and flexibility (no longer constrained to chain idiosyncrasies and message-passing cross-chain interaction). Additional benefits are that it relies on smart contracts on external chains minimally, so complex logic can work on not only smart-contract chains but also UTXO chains like Bitcoin.

6.3. Cross-chain message passing with value/data

The ability to reliably and securely pass messages from one chain to another can enable powerful cross-chain applications, even without native ZetaChain smart contracts. The message passing functionality consists of communication endpoints on all external chains. The ZetaChain validators serve as a byzantine fault tolerant notary that attests the validity of events/transactions on chain A to chain B, and as a relayer of messages. Chain B's smart contract only needs to whitelist the TSS address of ZetaChain in order to trust that ZetaChain has verified the events on chain A. This al-

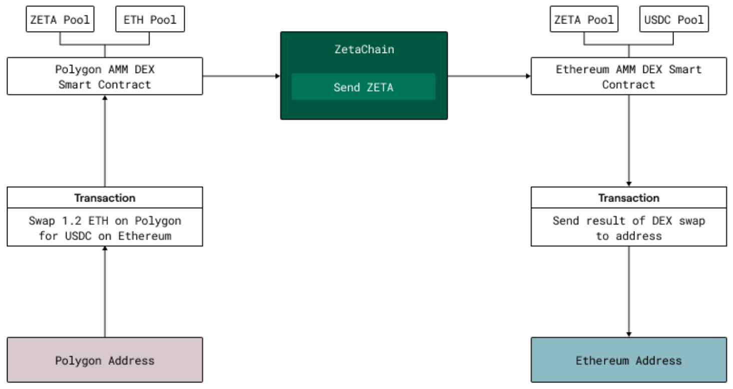
ZETA Pool
ETH Pool
Polygon AMM DEX Smart Contract
ZetaChain
Send ZETA
ZETA Pool
USDC Pool
Ethereum AMM DEX Smart Contract
Transaction
Swap 1.2 ETH on Polygon for USDC on Ethereum
Transaction
Send result of DEX swap to address
Polygon Address
Ethereum Address

Figure 7. DEX built with ZetaChain message passing. Leveraging external chain smart contract DEXs, one can build a cross-chain swap by sending messages with ZETA.

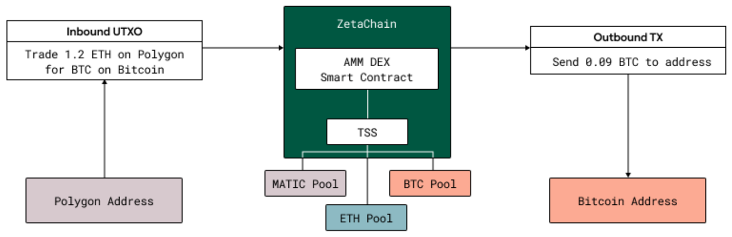

Figure 8. DEX built with ZetaChain Smart Contracts. Since ZetaChain TSS can manage external chain pools with its smart contracts, DEX can even support non-smart-contract chains and assets where transactions are simple and single-step.

lows conditional execution on chain B's contract depending on transactions/messages from chain A, which opens a wide range of cross-chain dApps, such as AMM DEXs, NFT, etc. (see more below). An important and convenient feature of the ZetaChain system is that the messages can be attached with value in the form of the ZETA coin (natively cross-chain), which considerably simplifies dApps which require moving the value cross- chain.

The messaging service of ZetaChain consists primarily of interface contracts on the connected chains. To access the message passing service, a dApp needs to deploy a smart contract on both the source chain and destination chain. On the source chain, the sending smart contract can invoke a `zeta.MessageSend` function with the following information: sending address, destination chain id, destination contract address, ZETA coin to transfer, gas limit on destination chain, contract message for destination transaction (binary or JSON encoded payload), and transaction index. The sending contract must implement a `zetaMessageRevert` function, which will be called by ZetaChain when the destination message delivery and processing of a transaction fails (for example, due to out of gas, out of funds, invalid message, etc.). Upon failure, the ZetaChain system will refund the ZETA coin to the sending address (less gas fees), and invoke the dApp contract `zetaMessageRevert` function which is supposed to revert application actions (unlocking a locked NFT, for example). On the destination chain, the dApp contract must implement a function `zetaMessageReceive` which takes the same parameters as the sending `zeta.MessageSend`, and can perform application logic (such as minting an NFT that has been locked on the source chain). The destination contract will also receive a ZETA coin (less gas fee), which can be used as a value transfer cross-chain.

Message passing can enable a variety of important applications such cross-chain DEX, borrowing/lending, multi-chain NFT, etc.

6.4. Multi-chain NFT

Non-fungible Token (NFT) is an emerging concept that has found use in art collection, gaming, event tickets, and many other applications. In contrast to fungible tokens such as ETH, BTC, or ERC-20 tokens, each NFT is unique and not interchangable with another NFT in the same collection. This non-fungibility can be essential in applications such as art, real-estate, etc. On Ethereum, for example, the most common NFT standards are ERC-721 and ERC-1155. In ERC-721, an NFT is basically a tuple `(contractAddress, tokenId)`. The smart contract that issues the NFTs keeps track of the owners of each NFT in a map `owner=>tokenId`. The NFT can be transferred from one owner to another, and each NFT owner can be queried.

In a multi-chain NFT world, where the same collection of NFTs are issued on multiple chains (such as Ethereum, Flow, Solana), and one NFT can transfer to another chain, a challenge in the bridge model is the knowing the provenance of a given NFT – who is the owner of a given NFT now that the NFT could be on one of multiple chains

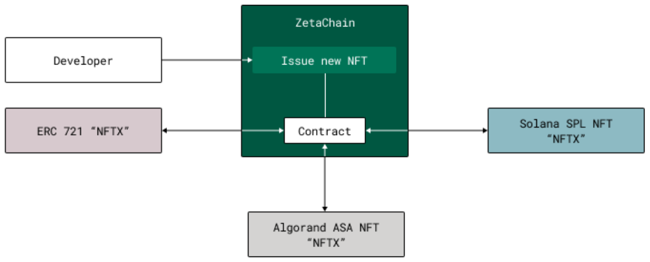

Figure 9. Multi-chain NFT. With a decentralized issuing authority (ZetaChain TSS), one can have an NFT that is easily sent between chains, where ownership and current location are easily verifiable.

and where are the records of the transactions of the transfers? This problem can be solved by ZetaChain smart contracts which facilitate cross-chain ownership transfers of NFTs. It can work as follows. Each chain will have an escrow smart contract controlled by the ZetaChain key. To transfer an NFT to another chain, one transfers the NFT to the escrow, pays transaction fee in ZETA coin, and ZetaChain will mint the NFT on the destination chain. The smart contract on ZetaChain keeps track of the owner and blockchain where the NFT is at any given time. While there have been experimental cross-chain NFT bridges, having a decentralized issuing authority allows an NFT to be natively cross-chain, making it simpler and feasible to create, verify, and exchange NFTs cross-chain.

6.5. Other Use Cases

These are just a few other potential use cases of ZetaChain. Given ZetaChain is a general smart contract platform, you can also imagine that any application you deploy on a single blockchain/smart contract platform can be expanded to operate across all connected chains.

6.5.1. Universal Payments

A system that lets users/EOAs send payments from/to any asset on any chain. This can help vendors and customers have a decentralized, universal, and accessible payments route that doesnt require users to have a hyper-specific set of assets on a specific chain.

6.5.2. Universal Identity and Assets

Identity system, name service, or Soul Bound Tokens that can serve as identities across all chains. With omnichain capabilities, identities can interact with other chains agnostically and in a future-proof manner as ZetaChain adds support for more chains. Users need not have individual identities/domains per chain, and can utilize their assets (gaming, collectibles, fungible tokens, etc.) from all chains from a single place.

6.5.3. Multi-chain, Multi-signature vaults

Securely custody and manage assets on multiple chains with a multi-sig that involves accounts and/or messages from many chains.

6.5.4. Omnichain Account Abstraction or Smart Contract Wallets

Smart contract wallets that can manage transactions to/from all chains, allowing things like gasless transactions, complex/multi-transactions, etc. that involve multiple chains. This could be imagined as an EIP-4337 but with omnichain capabilities.

6.5.5. Omnichain DeFi

DEXs, lending/borrowing, perps, and so on can support seamless 1-step trades and transactions that unify liquidity across chains. Leveraging omnichain smart contracts, one can significantly reduce common complexity and concerns of slippage, race-conditions, MEV that are involved with transacting fungible tokens that are often involved in todays cross-chain applications. Financial applications spanning many chains can be built with the same logic as if they were all on one chain.

6.5.6. Omnichain DAOs

Decentralized Autonomous Organizations and governance tooling that lets groups of people orchestrate activity, governance, and asset management in a chain-agnostic manner.

7. Security

7.1. Decentralization

The ZetaChain system is designed to not have a single point of failure, primarily through decentralization.

ZetaChain is decentralized architecturally and infrastructurally. Decentralization is an effective way to be fault tolerant, resist attacks and collusions. The ZetaChain nodes are run by individuals or organizations without permission. No single point of failure in ZetaChain node (ZetaCore, ZetaClient) affects the system.

On the other hand, to effect changes in external chains, ZetaChain must act as a single entity to sign messages, therefore raising the issue of centralized signing key. ZetaChain utilizes GG20 leaderless Threshold Signature Scheme (TSS) which does the key generation and key sign in a distributed, decentralized way. No single ZetaChain node or other individual ever has access to the complete private key at any point in time. Effectively, the ZetaChain node (the signer in ZetaClient, to be specific) has equal "vote" in signing outbound transactions, like in an m/n multisig.

To strike a balance between decentralization and coordination, certain aspects of ZetaChain are not fully decentralized, or designed to evolve into more decentralized gradually. For example, the software is currently developed by a central entity, which means the system is susceptible to software bugs from a single source. To defend against bugs ZetaChain employs multi-level blanket protection, to be discussed in more detail below.

7.2. Securing Inbound and Outbound Transactions

The ZetaCore takes in events from the observers in the ZetaClients. The ZetaClients monitor events on external chains via a variety of sources–node as service providers such as Infura, their (validator operator) own instance of full node, or full node run by the developers and partners. The observed event (as an inbound transaction to ZetaChain) must reach consensus on the ZetaCore to trigger state changes in the ZetaCore.

The state change in ZetaCore causes the signers of ZetaClient to prepare, sign, and broadcast transactions to external chains. ZetaChain's consensus mechanism ensures that the transaction is agreed upon; the TSS key ensures that only super majority of ZetaClients can sign.

All the inbound/outbound transactions and decisions made (through state changes) are recorded in the ZetaChain blocks which are available, immutable, verifiable, and completely transparent.

7.3. Comprehensive Defense Against Arbitrary Minting

Since the only native value that can move cross-chain through ZetaChain is the ZETA token, and ZetaChain effectively only manages transferring ZETA token from chain A to chain B, it's possible to offer comprehensive protection against the only way to steal value from Zetachain: invalid minting that inflates the total supply of ZETA across chains.

We offer comprehensive protection against minting without commensurate burning as follows:

ZetaChain nodes will check total supply across chains before initiating the minting of ZETA token. This protects against software bugs or vulnerability in the ZetaChain node software. The token contracts on the chains (except on Ethereum, where a locking contract will assume the role) checks total supply of ZETA across chains before minting. The total supply of ZETA is provided by Chainlink and posted on each connected chain. This protection ensures that no one can arbitrarily mint and that the total supply of ZETA remains fixed across chains. It should be noted that the two comprehensive defenses, while providing strong protection against software bugs and stealing from ZetaChain (including every holder of the ZETA token), they do not eliminate exploits. For example, if the attacker gains control of 2/3 validators, or the attacker is able to exploit a bug in the software, he is able to redirect a legitimate mint from another user to his wallet. However in this worst case scenarios the impact is likely to be contained as the attacker can only steal from active users at that specific time, and the system would be promptly stopped once noticed by users

In summary: the funds at risk in the worst-case scenario is only the ZETA amount that is being moved cross-chain at the time of the exploit. Funds at rest are never at risk.

7.4. What Happens When External Chains are Attacked

If the external chains connected by ZetaChain are being attacked (such as 51% attack), which can result in the following violations: 1) double spend leading to inflated supply of ZETA token; 2) censorship; 3) reversion leading to loss of atomicity of cross-chain transaction, as the source part might be no longer existent; 4) hard fork, chain split; and more. The design of ZetaChain can mitigate a few of these cases, or contain the damage from unlimited spreading. For example, an external chain causing unlimited mint (by repeatedly reverting and paying) cannot happen because of the total supply check of ZetaChain. By extension, the dApps that use ZETA coin for all cross-chain value transfer are also protected from unlimited inflation. For other external chains that are being exploited, the ZetaChain should go into an emergency halt to assess the situation. The recovery will be coordinated by stakeholders and governance mechanisms.

8. Conclusion

In this whitepaper we survey the landscape of cross-chain interoperability. While bridging is the main solution today and the focus of many emerging projects, ZetaChain explores a more ambitious and general approach: native cross-chain smart contracts that directly interact with nearly any external blockchain. No wrapping around assets are needed to transfer values cross-chain and no bridges are needed for every pair of blockchains. The ZetaChain smart contract can custody assets on an external chain directly, and manages assets according to predetermined arbitrary logic. Every external chain interaction is settled on external chains directly. As such, ZetaChain provides the most general platform for decentralized cross-chain applications to build on with connections to almost any existing or future blockchain and/or L2/rollup, with access to the whole supply of native assets on those chains.

References

[1] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. "A Survey on Blockchain Interoperability: Past, Present, and Future Trends." *ACM Computing Surveys (CSUR)* 54 (8). ACM New York, NY: 141. 2021.

[2] "Chain Key Cryptography: The Scientific Breakthrough Behind the Internet Computer." https://medium.com/dfinity/chain-key-technology-one-public-key-for-the-internet-computer-6a3644901e28.

[3] "Connext Documentation." https://docs.connext.network/.

[4] "Ethereum/BTCRelay." github.com/ethereum/btcrelay.

[5] "ETH-NEAR Rainbow Bridge." https://near.org/blog/eth-near-rainbow-bridge/.

[6] "FUSION Whitepaper." https://fusion.org/en.

[7] Rosario Gennaro, and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup." In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 11791194. CCS 18. Association for Computing Machinery, New York, NY, USA. 2018. doi:10.1145/3243734.3243859.

[8] Rosario Gennaro, and Steven Goldfeder. "One Round Threshold ECDSA with Identifiable Abort." *IACR Cryptol. ePrint Arch.* 2020: 540. 2020.

[9] "Hop: Send Tokens Across Rollups." https://hop.exchange/whitepaper.pdf.

[10] Jae Kwon, and Ethan Buchman. "Cosmos: A Network of Distributed Ledgers." *URL Https://cosmos.network/whitepaper.* 2016.

[11] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, and Yih-Chun Hu. “Hyperservice: Interoperability and Programmability across Heterogeneous Blockchains.” In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 549566. 2019.

[12] “Multi-Party Threshold Signature Scheme.” https://github.com/binance-chain/tss-lib.

[13] Caleb Banister Ryan Zarick Bryan Pellegrino. “LayerZero: Trustless Omnichain Interoperability Protocol.” *URL https://coinweb.io/files/Coinweb-Whitepaper.pdf*. 2021.

[14] “Synapse Protocol Documentation.” https://docs.synapseprotocol.com/.

[15] THORChain.org. “Decentralized Liquidity Network.” URL https://github.com/thorchain/Resources/blob/master/Whitepapers/THORChain-Whitepaper-May2020.pdf. 2020.

[16] *URL Https://chainflip.io/*. “Chainflip.” 2021.

[17] *URL Https://multichain.org/*. “Multichain (previously Anyswap).” 2021.

[18] *URL Https://sifchain.finance/*. “Sifchain.” 2021.

[19] *URL Https://wormholenetwork.com/en/about/*. “Wormhole.” 2020.

[20] *URL Https://www.quant.network/*. “Quant Network.” 2021.