TL;DR

- The default crypto UX today is for users to always know which network they are interacting with. However, users of the internet don't have to know which cloud provider they are interacting with. Bringing this approach to blockchains is what we call Chain Abstraction.
- This article introduces the CAKE framework i.e. Chain Abstraction Key Elements. It is composed of four layers: Applications, Permissions, Solving, and Settlement, which collectively facilitate seamless cross-chain operations for users.
- Achieving Chain Abstraction requires the use of a complex set of technologies to provide reliable, cost efficient, secure, fast, and private execution.
- We define the cross-chain tradeoff space in chain abstraction as a trilema and propose six designs, which each offering unique advantages.
- In order to successfully make the leap to a chain abstraction future, it is imperative we as an industry define and adopt a common standard for messaging between the layers of the CAKE. A great standard is the icing on the cake.

Introduction

In 2020, the Ethereum network transitioned to a

rollup centric roadmap

for scaling. Four years since that decision more than 50 rollups (L2s) are already live in production. While rollups provide much needed horizontal scaling for the EVM blockspace, it has

totally ruined the user experience

.

Users should neither care, nor know, which rollup they are interacting with. Crypto users knowing which rollup (Optimism or Base) they are interacting with is equivalent to web2 users knowing which cloud provider (AWS or GCP) they are interacting with. Chain Abstraction is a vision where chain information is abstracted away from the user. The user only connects their wallet to a dApp and signs for the intended operation, the details of making sure that the user has correct balance on the target chain and and then executing the intended operation happens behind the scenes.

Over the course of this article, we will observe that Chain Abstraction is a truly multi-disciplinary problem. Involving interactions with the Application Layer, Permission Layer, Solver Layer and Settlement Layer. We introduce the Chain Abstraction Key Elements (CAKE) framework and then delve deeper into the design trade-offs of chain abstraction systems.

Introducing the CAKE Framework

In a chain abstracted world a user goes to a dApps website, connects their wallet, signs the intended operation and waits for eventual settlement. All the complexity of acquiring the required assets to the target chain and the final settlement gets abstracted away from the user, happening in infrastructure layers of the CAKE. There are three infrastructure layers of the CAKE:

Achieving Chain Abstraction means combining the above three infrastructure layers into a unified product. A key insight while combining these layers is the difference between transferring information vs transferring value. Transferring information between chains should be lossless and thus needs reliance on the most secure pathways. Suppose a user is trying to vote

Yes

on a governance vote from one chain to another, they don’t want their vote to convert to a Maybe. On the other hand transferring value can be lossy based on users preference. A sophisticated third-party can be leveraged to give the user faster, cheaper or guaranteed transfer of value. Note, 95% of ethereum blockspace (weighted by fees paid to validators) is consumed for transferring value.

Key Design Decisions

The above three layers, introduce key design decisions which need to be taken by a CAF. They are related to who controls power over execution of the intent, what information should be revealed to solvers and what are the settlement pathways available to solvers. Let's look at each of them in detail.

Permission Layer

The permission layer holds the private key for the user and signs messages on their behalf, which are then executed on-chain as

transactions. A CAF needs to support signing schemes and transaction payloads for all the target chains it wants to support. For example, a wallet supporting the ECDSA signing scheme and EVM transaction standard will be limited to Ethereum, its L2s, and its side-chains (e.g., Metamask wallet). On the other hand, a wallet supporting both EVM and SVM (Solana VM) will be able to support both ecosystems (e.g., Phantom wallet). It's important to note that the same mnemonic can be used to generate wallets on both EVM and SVM chains.

A single multi-chain transaction consists of several sub-transactions that need to be executed in the correct order. These sub-transactions must be executed on multiple chains, each with its own time-varying fees and nonce. How the coordination and settlement of these sub-transactions takes place is a crucial design decision for permission layer.

Solver Layer

Once a user posts their intent the solving layer involves returning a

fee

and

confirmation time

to the user. This problem is closely related to designing an Order Flow Auction and has been written in detail

here

. A CAF can either leverage in-protocol paths to execute a users intent or leverage sophisticated third-parties aka solvers to provide improved UX to the user by compromising on some security guarantees. The next two design decisions arise when we bring solvers into a CAF framework, and are related to information.

An intent consists of two types of extractable values (EV):

EV_ordering

and

EV_signal

. EV_ordering is a value specific to the blockchain, typically extracted by entities that execute user orders like block-builders or validators. On the other hand, EV_signal represents the value accessible to any entity that observes the order before it is officially recorded on the blockchain.

Different user intents have varying distributions between EV_ordering and EV_signal. For example, an intent to swap coins on a DEX usually has high EV_ordering but low EV_signal. Conversely, an incoming hack transaction will have a higher component of EV_signal since front-running it will return significantly more value than executing it. It's important to note that EV_signal can sometimes be negative, such as in the case of trades from Market Makers, where entities executing these orders can experience losses due to a market makers better understanding of future market conditions.

When someone has the ability to observe a user's intent ahead of time, they can engage in front-running, which leads to value leakage. Additionally, the potential for EV_signal to be negative creates a competitive environment among solvers, causing them to submit lower bids and resulting in further value leakage (aka adverse selection). Ultimately, leakage impacts the user by either increasing fees or providing less favourable prices. Note, low fees or price improvement are two sides of the same coin and will be used interchangeably during the remainder of the article.

Information Sharing

There are 3 methods to sharing information with solvers:

In an auction with partial information on incoming orders, a solver may try to estimate the missing information by either: 1) making a bid that depends on obscured data, or 2) spamming bids. Both of these approaches raise concerns about latency races, which can lead to centralization and scalability problems due to excessive simulation resource consumption.

Solver List

The CAF also needs to decide how many and which bidders are allowed to participate in the auction. Broadly, the options are the following:

- Open access

: Barriers to entry for the ability to participate are as low as possible. This is similar to a public mempool and leaks both EV_signal and EV_ordering.

- Gated access

: There is some gatekeeping on the ability to execute an order, either through a whitelist, a reputation system, a fee, or a seat auction. The gatekeeping mechanism needs to ensure that solvers in the system don’t capture EV_signal. Examples are 1inch Auction, Cowswap Auctions and Uniswap X auctions. The competition to win orders captures EV_ordering for the user whereas the gating mechanism can capture the EV_signal for the order generator (Wallet, dApps).

- Exclusive access

: Exclusive access is a special case of the seated solver auction where only one solver is selected each time period. Since no information is leaked to other solvers there is no adverse selection and front-running discount. The orderflow originator captures the expected value of EV_signal and EV_ordering, since there is no competition the user can only get execution and no price-improvement. Some examples of these auctions are the Robinhood and DFlow auctions.

Settlement Layer

Once a wallet signs a set of transactions, they need to be executed on the blockchain. Cross-chain transactions convert the settlement process from atomic to asynchronous. While the initial transactions are being executed and confirmed, the state on the target chain can change, potentially leading to transaction failure. This subsection will study the trade-offs between the cost of security, confirmation time, and execution guarantee.

It is important to note that executing the intended transaction on the target chain depends on the transaction inclusion mechanics of the target chain. Including the ability to censor a transaction and the fee mechanism of the target chain, among other factors. We believe that the choice of the target chain is a decision for the dApp and will consider it beyond the scope of this article.

Cross-Chain Oracle

Two blockchains with distinct states and consensus mechanisms require an intermediary, such as an Oracle, to facilitate the transfer of information between them. Oracles serve as relays for information between chains. This includes verifying situations such as a user locking funds in an escrow account for a lock and mint bridge, or confirming a user's token balance on the origin chain for participation in governance voting on the target chain.

Oracles transfer information among chains at the speed of the slowest chain. This is necessary to manage reorg-risk, as the Oracle needs to wait for consensus on the origin chain. Let's consider a scenario where a user wants to bridge USDC from the origin chain to the target chain. To do this, the user locks their funds in an escrow. However, if the Oracle doesn't wait for enough confirmations and proceeds to mint tokens for the user on the target chain, a problem can occur. In the event of a reorg, if the user overwrites their escrow transaction, the Oracle would have double spend.

There are two types of oracles:

Bridging Tokens

In a multi-chain world, user token and fee balances are spread across all the networks. Before every cross-chain operation the user needs to bridge funds from the origin chain to the target chain. Currently there are

34 active

bridges with a combined TVL of $7.7B and bridging

volume of $8.6B

in the last 30 days.

Bridging tokens is a case of value transfer. This creates an opportunity to utilize specialized third parties who excel in capital management and are willing to assume reorg risk, reducing the cost and time required for user transactions.

There are 2 types of bridges:

Note, Lock and mint bridges create new tokens on the target chain, which may be wrapped versions of the required tokens. For example, USDC.e instead of USDC or axlUSD instead of USDC.

Over time, liquidity bridges have become more capital efficient. Solver networks (Uniswap X) are more efficient than lending protocols (Across), which in turn are more efficient than AMM bridges (Hop protocol).

In both types of bridges there is a liquidity cost which needs to be paid by the user. In Lock and Mint bridges the liquidity cost is while swapping from the wrapped token to the desired token (USDC.e to USDC) on the target chain, whereas in Liquidity Bridges the liquidity cost is while swapping from the token on the origin chain to the token on the target chain.

Cross-Chain Trilemma

The above 5 design decisions give rise to the cross-chain trilemma. A CAF has to choose 2 properties between Execution Guarantee, Low Fees and Execution Speed.

The Six Pieces Of CAKE

To write this article we studied more than 20 different designs from teams both explicitly and implicitly working on Chain Abstraction. In this section we discuss six independent CA implementations which we believe have inherent efficiencies and product market fit. These designs have the potential to compose with each other if built right.

One key takeaway from this exercise is that we need a common standard for expressing cross-chain intents. Each of the teams are

working on their own methods and protocols for encoding user intents. Unifying towards a standard will improve user understanding of the message they are signing, make it easier for solvers and oracles to understand these intents and simplify the integration with wallets.

Token Anointed Bridges

There is a special case of lock and mint bridge which does not pay the liquidity cost also called a burn and mint bridge (eg. USDC CCTP). The token team anoints a canonical token address on each chain while the bridge has the authority to mint the token i.e. the token which the user needs.

If you squint hard enough, a burn and mint bridge is similar to a cross-chain transfer at the speed of enough block-confirmations.

xERC20

is one such standard to anoint canonical tokens and their authorized bridges on target chains. A token anointed bridge is an example of an in-protocol path i.e. it compromises on speed for execution guarantee and low fees, e.g. CCTP takes 20 minutes to execute a transfer.

Ecosystem Aligned Bridge

An

ecosystem-aligned bridge

enables the transfer of arbitrary messages between chains within the same ecosystem. It falls under the category of in-protocol paths, prioritizing execution guarantee and low fees over speed. Examples include Cosmos IBC, Polygon AggLayer, and Optimism Superchain.

Three years ago, the Cosmos ecosystem faced similar challenges to what the Ethereum is facing today. Liquidity was fragmented across chains, each chain had its own fee token, and managing multi-chain accounts was cumbersome. The Cosmos ecosystem addressed these issues by implementing in-protocol message passing bridges through IBC, resulting in seamless multi-chain accounts and cross-chain transfers.

The cosmos ecosystem comprises of independent chains having sovereign security and fast finality, making the in-protocol path for cross-chain messaging very fast. On the flip-side the rollup ecosystem depends on expiry of the challenge period (Optimistic Rollups) or commiting zk-proofs (Validity Rollups) for finality. In-protocol paths for message passing across ecosystems will be slow due to these finality constraints.

Solver Price Competition

A

Solver price competition

involves sharing order information with all solvers. Solvers aim to incorporate the expected value (EV) generated by the intention of the order and provide it to users. The selection of the winning solver in the system is based on maximizing user price improvement. However, this design carries the risk of non-execution and requires additional mechanisms to ensure the reliable inclusion of orders. Examples of such mechanisms include Uniswap X, Bungee, and Jumper.

Wallet Coordinated Messages

Wallet coordinated messaging

utilize capabilities provided by AA or policy-based wallets to offer a cross-chain experience that is compatible with any intent type. It serves as the ultimate CA aggregator, redirecting user intents among various CA designs to address specific intents. Examples include Avocado wallet, Near Account Aggregator, and Metamask Portfolio.

Note, over the last decade, the crypto-ecosystem has learnt that the relationship between a user and their wallet is very sticky. I personally feel a mortal dread whenever I think about migrating my mnemonic from Metamask to another wallet. This is also the reason why even after 2.5 years and backing from Vitalik Buterin himself EIP-4337 has gained

minimal adoption

. Although newer versions of wallet protocols might provide the user with better price (account abstraction) or improved ease of use (policy-based wallets), migrating the user from their current wallets is an uphill task.

Solver Speed Competition

The Solver speed competition allows users to express their intentions for specific cross-chain transitions for high execution guarantees. It does not assist users with minimizing fees, but instead offers a reliable channel for including complex transactions. The first solver to execute the intent based on block-builder fees or inclusion speed wins the intent.

The design aims to achieve a high inclusion rate by maximizing the EV captured by solvers. However, it comes at the cost of centralization, as it relies on sophisticated capital management on the Ethereum mainnet or low-latency execution on L2s.

Exclusive Batch Auctions

An

exclusive batch auction

holds an auction for the exclusive rights to execute all the order flow in a time window to a single solver. Since other solvers cannot see the orders, they place the bid based on the predicted market volatility and their average execution quality. Exclusive batch auctions depend on a

backstop price

in order to assure good user prices and therefore can’t be used for price improvement. Sending all the order flow to a single bidder eliminates information leakage and improves execution guarantees.

Conclusion

Chain Abstraction Frameworks (CAFs) promise to give users seamless cross-chain interaction. In this article we studied designs in-production and in-development by several teams who are explicitly or implicitly trying to solve for Chain Abstraction. We believe this will be the year of CAFs and expect significant competition happening between different designs and their implementations in the next 6-12 months.

Cross-chain value transfers will be routed through a combination of token-anointed bridges for low fees and Solver Speed or Price Competitions for speed and execution. While information transfers will be routed through a combination of ecosystem aligned message bridges which will aim to minimize the cost to users, and to wallet controlled platforms who will maximize speed. Final implementations will cluster around these six distinct designs as they each serve independent needs and benefit from efficiencies existing in different corners of the tradeoff matrix.

One key takeaway from this exercise is that we need a common standard for expressing cross-chain intents. Several teams are working on their individual protocols for encoding user intents causing duplicated work. Unifying towards a standard will improve user understanding of the message they are signing, make it easier for solvers and oracles to work with intents and simplify the integration with wallets.