Xythum Protocol

A Trustless Decentralized Aggregation Layer For BlockChains

5th June 2024

Jayendra Madaram

NGIT

Abstract

The meteoric rise of blockchain technology and decentralized finance (DeFi) has unlocked vast potential, with cryptocurrency market capitalization and trading volumes soaring. However, despite a decade of progress, blockchains largely operate in silos, hindering liquidity flow. Web3 demands a paradigm shift beyond single execution environments, necessitating a purely decentralized aggregation layer.

We present to you Xythum Protocol, a truly decentralized aggregation layer for a variety of blockchains that offers you, your private execution environment, access to limitless liquidity across any corner of web3. Xythum leverages a trustless Secure Multi-Party

Computation (SMPC) network to facilitate trades without compromising sensitive transaction details until settlement.

Disclaimer: This paper is the first release of "Xythum" preliminary conceptual design, hence over the course of development there might be a lot of iterations to this paper, and a lot can be modified but the core concepts would remain the same.

Introduction

The emergence of Blockchain brought a new way of finances into the world, which is more trustless, Open, and truly decentralized. Blockchains are backed by strong cryptographic principles making the distributed ledger secure. Bitcoin, Ethereum, and Solana are the most widely discussed blockchains in today's world and each of them serves different purposes. As of writing this paper, the total market capital of the existing blockchain ecosystem is over 2.5 Trillion US dollars with per day Volume crossing 90 billion US dollars. With the recent Buzz around crypto ETFs and the enormous growing utility around the whole ecosystem, the whole industry is subjected to reach new ultimate heights. But Still lot of people in space tend to choose centralized services(custodial) as their primary preference for dealing with web3. centralized service architecture is against the whole point of decentralization and self-custody. Well, it is true over a period of time from “FTX to MT. Gox” there has been a lot of misuse of custody of assets by Centralized exchanges. But there is a ray of hope, Uniswap, a truly decentralized Exchange operating with Automated Market Maker style architecture was a revolution in space. It can be said that without Uniswap there won't be any well-known Defi products out there.

In Reality Uniswap's success is only limited to a single execution environment. whereas web3 is not limited to a single module. So far BlockChains have been executing in either Monolithic style or Modular Style which adds huge friction to liquidity on different chains to flow into other chains. Since the start of decentralized finance, there has always been a talk of **Interoperability.** There were many proposals on this topic but none of them were decentralized enough. This is the reason why almost 80% of hacks/heists happen on crypto bridges. Most crypto bridges are not truly decentralized enough hence causing a Single point of failure or getting Hacked often. In the Context of Rollups which are anchored to its Layer 1s are also limited to access to liquidity from its sibling rollups and child rollups, which again brings a huge need for a layer that acts as a shared liquidity layer in which all chains interact and access liquidity along with context sharing. Talking about our next concern in web3 liquidity flow is the lack of privacy. In traditional finance markets we have exchanges that go by the name “dark pools” It is estimated that over 20% of the whole trade on Wall Street happens through dark pools. A dark pool is a privately organized financial forum or exchange for trading securities. Dark pools allow institutional investors to trade without exposure until after the trade has been executed and reported. Well, there is a huge need for a secure dark pool architecture in web3 space that can be used by HNI’s to protect their transactions from getting front-runner and having slippage issues, Well, Here we present you “**Xythum PROTOCOL**” a truly decentralized, trustless aggregation layer for all blockchains that opens the door to Boundless liquidity. Xythum allows its traders to execute transactions in a dark pool environment. The rest of the paper is divided into three segments, we start by explaining how Xythum works and lives up to its promises, followed by the security model supporting Xythum, and finally what steps will be taken to encourage the community/enthusiasts to be a part of the system.

Note: for the rest of the paper we consider having the context of only two primary chains BITCOIN and ETHEREUM. but the same principles can be applied to any other chain.

MOTIVATION

- - CrossChain Trades
- - Private Execution
- - Interoperability
- - Decentralized Bridging

DEEP DIVE

Before understanding how Xythum achieves true decentralization we must explore how today's major bridges bridge assets. Most of the current bridges are built upon resolvers, executors, and sequencers which are majorly maintained by core protocol members. They tend to be prone to a variety of known attacks on distributed systems. several protocols use (k-of-n) multi-sig bloating up signature callData for verification also limiting participants. Speaking techniques followed by these systems are usually "Lock-n-mint" or "Atomic Swaps".

"Lock-n-mint"

trade LifeCycle starts with the User initiating an Order on the source Chain, usually depositing into a deposit channel pre-determined by the bridge, during the

initiation nodes will be actively watching over the deposit into the deposit channel. once the deposit has passed enough confirmation on the source chain. a node initiates a mint transaction on the destination Chain. Well, the mint transaction creates a new synthetic token representing a 1:1 value for the token locked on the source chain. this newly minted synthetic token can be used on the destination chain as bridged liquidity.
How about we need a system where we don't need synthetic tokens but rather we intend to swap ASSET A on CHAIN X to ASSET B on CHAIN Y. This can be accomplished by introducing htlcs

Say, Alice wants to trade 1 A on CHAIN X for 10 B on CHAIN Y which she thinks is a valid market price, and Bob agrees to do the trade. they could do the following atomic swap for a secure trustless swap

Steps involved in Cross Atomic Swap:

1. Alice creates random secret s and computes secretHash $sh = \mathrm{H}(s)$.
2. Now she creates an order on Chain X with a condition, when a secret for sh is submitted 1A should be transferred to bob address on Chain A
3. Looking at the Open Order created by Alice, Bob picks up sh and initiates a Similar Order on CHAIN Y with a similar condition as Alice, when someone reveals a secret in this Order 10 B is sent to Alice's address on Chain Y.
4. Once Alice confirms the Order on CHAIN Y is initiated with the proper amount she can reveal the secret she decided. and claim B asset.
5. Once the secret s is revealed on Chain Y Bob can capture the secret and reveal it on CHAIN X and claim his funds
6. If due to any circumstances either party decides not to reveal a secret, the corresponding party can refund their initialized asset after waiting for the order to expire.

Note: H(x) refers to the cryptographic One-way-hash function which is supported by both blockchains

Bridges can still swap for different assets by using on-chain liquidity from the destination Chain using the Following “Lock-n-mint” strategy.

Say, Alice wants to trade 1 A for 10 B which she thinks is a valid market price.

1. Alice transfers the required amount (1 A) to the Entrypoint address (deposit Channel) designated by the bridge.
2. Nodes watch at deposit events and catch the required metadata to execute the rest of the trade.
3. Bridge Nodes mint new synthetic assets Say 1 oA on CHAIN X where 1 oA == 1 A (1:1).
4. Now bridge can use existing on-chain liquidity on CHAIN Y swap 1 oA for 10B and send it back to the Alice address on CHAIN Y.
5. At any point in Time, Alice or any user can bridge their 1 oA back to 1A at a 1:1 ratio.

Xythum uses both modes of execution cleverly and strategically depending upon the trade. Let's also look at how exactly Xythum is truly decentralized. Xythum Protocol operates on a group of nodes, where no single node can themselves alone execute protocol trades. The entirety of Xythum Protocol relies on a cryptographic scheme called threshold Signatures where in a n group of participants. we can use only k out of n participants to create signatures at the protocol level. Xythum more specifically uses Schnorr Signature Schemes which are most preferable in threshold signatures. Traditional well-known blockchains like Bitcoin, Ethereum, and Solana rely on Elliptic Curve Cryptography but over the course of time, Schnorr signature have proved their resilience to all modern cryptographic attacks offering the same security as ECDSA. Schnorr Signature can be 40% less Gas-effective and faster to verify. But most of all the main reason why Xythum uses Schnorr over ECDSA is its ability to effectively aggregate signatures and perform threshold signatures. Schnorr signatures are linear in nature which gives an advantage over ECDSA which aren't linear making it challenging to form threshold signatures.

Even though Schnorr is not natively supported on Ethereum Xythum tweaks ecrecover precompiled contract to perform Schnoor signature, while same time from TAPROOT upgrade on Bitcoin, it natively supports Schnorr signature verification. Xythum uses a more advanced Schnorr Threshold Signature scheme FROST. Hence k of n nodes produce a Schnorr signature for a specific Tx and k Txs can be aggregated to form a resultant signature which looks exactly signed by the privkey corresponding to group PubKey. It is true that n can vary over time, As participants join and leave the system and this might affect resultant Signatures, Xythum operates on an EPOCH basis. Each Epoch Lasts for 2 weeks and n stays constant for 2 weeks time, refer to Node onBoarding Section for further reading.

Each Epoch has two phases of execution as described Below.

Distributed Key Generation

This is the initial/final action taken in an Epoch by nodes willing to participate.

Steps Followed in DKG for secure setup.

1. Each participant P_i ∈ { 1, …. , n } generates a secret a_i0.
2. Σ a_i0 is privkey for the whole group and it is not exposed to other participants
3. Every participant generates Proof of Knowledge of a_i0

4. In order to share one's secret a user follows a well-known secret-sharing algorithm “Shamir Secret Sharing” where instead of sharing a secret. participant shares coordinates of a polynomial whose y value at x = 0 is a_i0.
5. assuming polynomial being k degree, for such a system k out of n participants can perform “LaGrange’s interpolation” on their coordinates and re-construct polynomial revealing a_i0.
6. More specifically nodes use VSS (verifiable secret sharing) where other nodes can verify that they received valid shares without disclosing the share or secret.
7. This results in O(n^2) network calls but this is one time process per Epoch
8. Once all participants have their shares of all polynomials everyone can generate Group Public Key (Y) and individual participant PubKey (Y_i)
9. Participants privKey = Sum(F_i(index)) | index is participant position

Signature Aggregator

SA (Signature Aggregator) is a semi-trusted service running to share information. SA never contributes to signature generation at the participant level. anyone spins up their own SA and nodes will accept it until it broadcasts an improper message.

Roles of Signature Aggregator

- Build Transaction Object.
- Track n Nodes.
- Emit the Latest State to Nodes.
- Verify Signatures submitted.
- Aggregate Signature submitted from Nodes

Nonce Generation

To perform a Threshold Schnorr Signature as described in FROST. Nodes are subjected to generate a commitment to nonce which will be used to generate Signatures. Signature Aggregator requests for pre-computed commitments to nonces which will be used by respective nodes. Upon using all nonces SA tends to flush previous nonces and request for new nonces.

Signature Generation

Participants follow the same Signing as the traditional Schnoor signing and use their Lagrange Coefficient in order to indicate their share of group Privkey. Once these signatures are aggregated and linear Summation of the signature leads to the final signature whose verification results in the group pub key.

Key Resharing

After Every Epoch we re-perform DKG with new N participants and the Pubkey is generated and updated on Ethereum as the owner of assets at the same time. on Bitcoin combining all previous inputs of the previous pubKey to the new pubKey.

Dark Pools On Xythum

Xythum uses the previously described “Shamir secret sharing” algorithm to share secret orders to k nodes and neither of the nodes will have knowledge of the complete order, until the whole order is executed, and for nodes executing the trade they won't even have the context of who the sender is. Xythum further strengthens order privacy by incorporating Zero-Knowledge Proofs (ZKPs). This powerful cryptographic tool allows nodes to verify the validity of a transaction without revealing any sensitive details about the order itself. For instance, nodes can confirm they possess a valid share without disclosing the specific content of that share.

Xythum Life Cycle

Say a user ALICE wants to convert her 1 BTC on Bitcoin to x ETH on Arbitrum and later convert her x/2 ETH back to Y BTC on the Lightning Network.

Setup Phase

- The protocol will be put on halt. and locked from accepting incoming Transactions
- All N Nodes in the protocol agree on a central Signature Aggregator (SA).
- All N perform FROST Distributed Key Generation and compute their private Shares.
- After Key Generation SA requests for commitments and Shamir shares of the participants in order to compute the “Group Public Key” and Individual “Participant PubKey”
- “SA” Starts with the accumulation of Nonce commitments that will be used for each transaction from all participants.
- Once the key generation is complete and if the current Epoch is not “Genesis Epoch”, Then previous Epoch shares are used for one last time in order to assign a new “Group Pub Key” Signer and Owner of funds from the previous Epoch.
- “SA” declares completion of the Setup phase and opens up the lock held on protocol, Making the system available for accepting incoming transactions.

Transaction Life Cycle

ALICE INITIATE

- Alice can now create an Order on where
    - Src Chain → Bitcoin
    - from Amount → 1 Btc
    - RouteThroughDarkPool → true
- SA picks up Alice's Order, chooses one of the "Pay-To-Taproot" addresses (deposit Channel) and notifies Alice with address and Order Creation.
  Note: On Bitcoin since using a single address for many transactions is not possible (only 21 ancestors and descendants transactions per address in a single block to prevent DOS) Xythum uses multiple deposit Channels to choose the right deposit channel based on its Load Balancing algorithm.
- Xythum deposit channels are special because they use Schnorr Signature verification instead of ECDSA, and only a threshold Signature from (k-of-n) nodes can sign a transaction.
- Alice deposits the desired amount into the deposit channel.
- Once the deposit transaction has passed enough confirmations on the Bitcoin chain. Asset is considered successfully locked SA would Start the Mint process.

MINT PROCESS

- SA starts to sample α nodes randomly out of n nodes such that $k <= α <= n$. where k is the signature threshold and notifies them to begin the signing process.
- SA then creates a Schnorr transaction payload which includes callData on the Arbitrum chain indicating mint new "1 oBtc" (obtc is an exact peg of real Btc 1:1 but on Arbitrum Chain).
- SA constructs a serial order of nodes and the Nonce commitments that will be used.

- This entire callData with its metadata is sent to all α nodes, which respond with appropriate Schnorr Signatures.
- SA checks the validity of signatures and uptime of individual nodes along the process
    - Malicious nodes will be slashed/punished for wrong signatures.
- SA aggregates all signatures and forms the final Signature. nodes are incentivized to participate in signature generation
- Additionally, a fee decided by protocol(x bips of total input amount) is charged per order and is collected as a receiving token.
- SA submits aggregate Signature to Xythum Entrypoint contract on arbitrum which validates the signature with group PubKey.
- Upon successful verification new oBtc is minted on the destination chain.

Onchain Swaps

- once oBtc is freshly minted on the resultant Chain. Entrypoint uses onChain liquidity providers like uniswap or liquidity aggregators like 1Inch and swaps “oBtc” to the resulting requested token X ETH.
- These X ETH can be used by Alice on arbitrum natively.
- When Alice wishes to get her BTC back, she could convert her X Eth back to “oBtc” using Onchain liquidity.
- In order to bridge back to Bitcoin
- Alice burns (sends her “oBtc” to null address 0x0) with metadata consisting of Alice receiving address on the Bitcoin chain.
- SA watches for burn events and accumulates metaData and waits for confirmations on arbitrum.
- After enough confirmations are received, SA will start the process of “Release” on Bitcoin.

- Which would send an equivalent amount of “Y obtc” to the destination address described by Alice from the locked Assets received earlier.

System Attitude

Xythum Protocol provides following properties:

1. Traders can experience limitless cross chain swaps without going through the hassle of underlying Frost based Bridging.
2. Trader doesn't have to be online until order completion.
3. Nodes executing Traders order never has any informational advantage and traders identity is private.
4. BlockChains can build adapters which would allow cross chain messaging (context sharing) among blockchains
5. Total liquidity of Xythum cannot be estimated at any point of time to a decent level.
6. Depending on traffic/user customisation trades are executed in batches on chain.

Tokenomics and Incentives

Xythum Protocol follows several strategies to make the aggregation layer totally unbreakable. Xythum Protocol has its own “Xythum” Token which powers whole protocols and works as fuel for the ecosystem. It is planned to maintain Xythum Token liquidity on all EVM chains but governance only on the Ethereum Mainnet. Tokenomics and Incentives are described as Follows.

Xythum TOKEN

- Follows ERC 20 standard and can be bridged to any EVM chain using the Xythum protocol with 18 decimals.
- 1% of the transfer amount is collected as a development fee.
- 2% of all transfer amount is collected by all participants for a specific Epoch

STAKING

An open network like the Xythum Protocol necessitates a robust mechanism to deter Sybil attacks and ensure Byzantine Fault Tolerance (BFT). While allowing anyone to join as a node participant fosters decentralization, it also introduces potential vulnerabilities like Denial-of-Service (DoS) attacks or malicious actors accumulating excessive voting power. One approach to solve this could be for participants to be able to stake any Asset in order to join the platform. Staked assets should have real-world value and the amount to stake has to be enough such that it is economically impossible for a malicious party to gain over 50% of the network, Since 50% is an attack vector of Xythum Protocol as described in the **Security Model** Section. Xythum protocol chooses the Xythum Token as its Stake token, And the `Stake Amount` is adjusted based on Xythum governance and the real-world value of the Xythum Token. Staked tokens also yield more Xythum tokens over time.

SLASHING and REPUTATION

Currently, the protocol can be subjected to malicious signature submission, fault tolerance, and denial of signing. To guard protocol from such attacks and, at the same time to incentivize participants from honest behavior there must be a strong Punishment/Reward model deployed into the system, and here is how Xythum achieves this.
There are two types of Slashing mechanisms going to be used,

- Partial Slashing: a certain percentage of the total Stake amount is being burned, and such percentage is decided by protocol Governance. following are scenarios where a participant would be subjected to partial slash
    - Submission of improper commitment while Key Generation
    - Going offline or refusing to Sign a valid transaction signed by a majority of protocol participants.
- Full Slash: The total Stake of participants is slashed and distributed among protocol Participants.
    - This occurs when SA recognizes the Attack vector to grow to a sufficiently large

Along with Slashing, Xythum also deploys a viable reputation management system, where all honest participants and active signature submissions are rewarded with a virtual score of reputation. Node runners can use these reputations to rank themselves at the top of the Xythum's leaderboard, where in protocol considers these top honest parties for htlc swaps as the primary choice.

RE-BALANCE BONUS

Since Xythum is truly multichain and follows a "lock-n-mint" scheme as most of its core, there might be a case where users are subject to dry up liquidity on a single Chain and shift all liquidity to other chains. In order to avoid this, Xythum users are incentivized to rebalance the liquidity with less fee to nil fee and higher fee in the opposite direction, such that motivating users and resolvers to trade in opposite directions and fill the liquidity on the opposite chain.

FEES

To incentivize nodes to actively participate in protocol and decentralize the ecosystem they can collect a fee per trade basis. Every trade happening in Xythum is subjected to have a protocol fee of 0.1 to 1% fee in destination asset per order. Upon successful execution of a trade, all α participants chosen by the aggregator are rewarded with a fee socially. encouraging participants to honestly/actively take part in the protocol.

ARBITRAGE

Xythum opens a wide range of opportunities for Arbitrageurs, The dynamic nature of asset prices on Xythum presents a compelling opportunity for arbitrageurs – savvy traders who exploit price discrepancies across markets. By capitalizing on these fluctuations, arbitrageurs help ensure that Xythum's synthetic asset prices closely reflect real-world values. Xythum is open for arbitrage and will be deployed with powerful indexers for searchers to give insights about arbitrage opportunities.

LENDING / LIQUIDITY PROVIDING

Well in the beginning Xythum synthetic tokens are of no value since they are not available as Onchain liquidity. Xythum assumes this can be overcome over a period of time when the protocol will be used often, Xythum incentivizes users to provide liquidity for Xythum synthetic tokens on all possible chains by a time-to-time airdrop of Xythum tokens, This encourages users to provide liquidity for Xythum synthetic tokens over which in return establishes routes for unlimited defi experience. Xythum initially kicks off deploying several utility-based products for its synthetic tokens like lending protocols and perpetual contracts.

GOVERNANCE

Xythum can be a complex protocol with a lot of constraints adding and making it difficult to make rational decisions in its development. To make things truly decentralized Xythum comes up with its own Xythum Decentralized Autonomous Organization (DAO) where significant Holders of a significant amount of Xythum tokens become voting members, wielding the power to influence the protocol's future. This empowers the community to participate in crucial decisions, such as proposing protocol upgrades or advocating for development strategies that accelerate Xythum's growth.

System Assumptions

I. Strong Cryptographic Primitives: We assume that the cryptographic primitives used in our system, such as hash functions, digital signatures, and encryption schemes, are computationally secure and cannot be broken within practical limits by adversaries with reasonable computational resources.

II. Truly Random Number Generation: We assume the existence of a reliable source of true randomness for various cryptographic operations and protocols within our system, such as key generation, nonce creation, and random sampling.

III. Secure Underlying Blockchains: We assume that the underlying blockchains involved in our cross-chain interoperability solution maintain their respective consensus rules, immutability properties, and provide a consistent and secure view of their respective ledgers, same goes for platform built on blockchains too

IV. Honest Majority: We assume that a majority of the nodes participating in our system are honest and follow the protocol correctly, with their combined financial and computational power exceeding a predetermined FROST threshold.

V. Rational Participants: We assume that participants in our system act rationally and will not participate if there is no financial incentive to do so. Additionally, we assume that participants will attempt to maximize their own profit, and therefore, we do not assume that a participant will act honestly if they can maximize their profit by acting maliciously.

VI. Secure Infrastructure: We assume that our system's infrastructure, including the setup process, communication channels, and existing blockchain infrastructures, cannot be tampered with or compromised by adversaries. We assume cryptographic and computational primitives are deterministic and ideal on all blockchains.

Adversary Assumptions

I. Limited Adversarial Power: We assume that adversaries have limited financial and computational powers, making it infeasible for them to control or significantly influence the majority of nodes in our system.

II. Censorship Attempts: We assume that adversaries may attempt to censor or prevent certain transactions or cross-chain interactions from being processed by our system, either through network-level attacks or by leveraging their controlled nodes.

III. Oracle Integrity: We assume the existence of a trusted third-party entity that performs cross-chain computations and data transfers with utmost honesty and correctness, albeit with constrained computational capabilities. Concretely, an adversary cannot manipulate or compromise the integrity of the computations and data transfers facilitated by this trusted third-party. All cross-chain platforms relying on such a trusted third-party need to make this adversarial assumption.

Security Model

Ensuring robust security is a paramount objective in the design of the Xythum Protocol, a decentralized trading platform for cryptocurrencies. The protocol's security architecture is modeled on the real vs. ideal world paradigm, a widely adopted approach in cryptography and distributed systems.

In the ideal world scenario, a trusted and incorruptible entity acts as an intermediary, facilitating trades between parties while preserving the confidentiality of their orders and identities. Traders submit their buy and sell orders to this trusted entity, which matches compatible orders and executes the trades, swapping the respective cryptocurrencies between the traders.

The real-world implementation of the Xythum Protocol aims to replicate the security guarantees of this ideal world through a decentralized and trustless architecture. Instead of a single trusted intermediary, the protocol employs a network of nodes that collectively perform order matching and settlement without revealing trade details.

SIGNATURE AGGREGATE POISONING

The Xythum Protocol employs a Signature Aggregator, which can be any participant or a semi-trusted third-party service. The primary responsibilities of the Signature Aggregator include indexing nodes, performing aggregate actions, and submitting orders on-chain. Crucially, the Signature Aggregator itself does not possess the capability to cause potential damage to the system at any point. However, if the Aggregator becomes compromised or colludes with malicious actors, it can potentially lead to system outages. In such scenarios, the protocol necessitates a temporary halt until a new setup is established, and a new Signature Aggregator is chosen.

The protocol's design incorporates safeguards to mitigate the risks associated with a compromised Signature Aggregator. These measures include:

1. Periodic rotations: The Signature Aggregator role is periodically rotated among a pool of vetted participants or services, reducing the window of potential compromise, One way is to conduct Auctions for Signature Aggregate Role.

2. Decentralized verification: While the Aggregator submits orders on-chain, the validity and integrity of these orders are independently verified by the network's nodes, preventing any potential manipulation.

3. Redundancy and failover: The protocol supports multiple Signature Aggregators, enabling seamless failover and continued operation in case one Aggregator becomes unavailable or compromised.

4. Transparency and accountability: The actions of the Signature Aggregator are publicly auditable on the blockchain, promoting transparency and enabling the detection of any anomalous behavior.

HTLC ATTACKS

The Xythum Protocol facilitates cross-chain interoperability through various
mechanisms, including classic atomic swaps, in addition to the lock-n-mint approach. While atomic swaps are inherently atomic and secure in theory, they can be subject to potential risks at the application layer. One key assumption made by the Xythum Protocol at the client-side application layer is that the generation of random preimages is not influenced by any adversarial entity. This assumption also includes atomic swaps that leverage the same cryptographic primitives across all chains involved in the swap.

To mitigate the risks associated with atomic swaps, the Xythum Protocol employs a time-lock agreement mechanism between makers and takers. The protocol ensures that an order is only processed if there is sufficient time-lock agreement between the counterparties, preventing potential attacks that could arise from time-lock discrepancies.

Attacks

In this section we will go through attacks on systems and impact of attack vectors along with defenses.

- **Value Extraction**

  Xythum follows the FROST threshold scheme which would allow k-of-n. Hence k has to be chosen after practical study such that k shouldn't be too large which would limit System to perform parallel computing or sharding at the same time k should be too small where an adversarial could easily get into the order execution sample.

For every order execution Xythum samples a group of nodes truly random, followed by splitting order among the nodes now in order to interpolate order threshold k of partitioned nodes are meant to be colluded. But in the real world as the system grows financial bonds for onboarding nodes and computational resources would shoot up if an adversarial tries to gain any information over an order. In order for an adversarial to do this he has to have nodes count x where
$K < x < n$. Assuming k would always be above 50% of n. It would take enormous resources of an adversarial to collude with other parties or perform a sybil attack on the system. Yet the Xythum engine random sample should include only all of his nodes. Even with one honest party in the sample adversarial could never interpolate the order. Any information leaks discovered by governance can lead to loss of bond.

- **Fault Tolerance And DDOS**

The Xythum Protocol is designed to operate in a decentralized and distributed environment, where the reliability and availability of individual nodes cannot be guaranteed. To ensure the system's resilience against various faults and attacks, the protocol incorporates robust fault tolerance mechanisms and periodic epoch transitions.

One of the primary threats the protocol addresses is the scenario where a significant number of nodes become unreachable, corrupted, or subjected to Distributed Denial of Service (DDoS) attacks. In such cases, the protocol's Engine component, responsible for orchestrating the overall operation, detects the potential threat and initiates a transition into maintenance mode.

During maintenance mode, the protocol temporarily suspends regular operations and awaits the start of the next epoch. An epoch represents a fixed period of time during which the protocol operates with a specific set of participating nodes and security parameters.

At the beginning of each new epoch, the protocol performs a Fresh Random Oracle-based Secure Threshold Distributed Key Generation (FROST DKG) ceremony. This ceremony involves the following steps:

- ○ **Peer Discovery and Vetting**: The protocol discovers and vetts a new set of peer nodes to participate in the upcoming epoch. This process involves a combination of reputation scoring, stake-based incentives, and distributed consensus mechanisms to identify reliable and trustworthy nodes.
- ○ **Threshold Adjustment**: Based on the number and quality of the discovered peers, the protocol adjusts the threshold equation used for secure multi-party computation (MPC) and distributed key generation (DKG). This adjustment ensures that the system maintains an appropriate level of fault tolerance and security guarantees, even in the face of potential node failures or compromises.
- ○ **FROST DKG Ceremony**: The protocol initiates a new FROST DKG ceremony with the selected peer nodes. This process enables the distributed generation of fresh cryptographic keys and shared secrets, which are used for secure communication, order matching, and settlement during the upcoming epoch.

- **Sybil Attack**

To counter Sybil attacks, the protocol mandates that all nodes and traders commit a bond, or stake, to register their identities. This bond requirement is designed to leverage the Adversarial Assumption, which states that adversaries have limited financial resources. By enforcing a bond, the protocol ensures that an adversary cannot feasibly forge a large number of identities.

For malicious nodes, the bond serves as a deterrent against registering an excessive number of nodes and acquiring a significant number of order shares during the order distribution process. The bond amount is carefully calibrated to strike a balance – high enough to discourage adversarial behavior but low enough to allow honest nodes to participate without excessive barriers.

Furthermore, the bond amount is dynamically adjusted to account for fluctuations in the value of the bonded cryptocurrency, ensuring a globally consistent and fair requirement for all nodes.

In the case of malicious traders, the bond mechanism is extended to discourage the submission of a large number of false orders. Traders are required to submit orders that reference their registered bond, establishing a linear relationship between the bond amount and the maximum number of open orders they can place.

For example, if a trader submits a bond of B and is allowed M open orders, they could alternatively submit a bond of B/2 and be permitted M/2 open orders. This direct correlation between the bond and the order limit serves as an additional deterrent against false order submissions, as traders with malicious intent would need to commit increasingly larger bonds to execute their attacks.

ROADMAP

As of writing this paper, the Xythum protocol is subjected to building all its tech piece by piece in phases and launching accordingly. Every release of Xythum consists of 2 pre-phases of launch before Mainnet Launch.

- TestNet launch: Every version would have a tesnet launch for its users to try out the next rolling version and write back to protocol developers.

- HellNet Launch: an environment for developers, bug bounty hunters, and adversaries to stress test the next rolling version.

Kickoff With Ordinals and BRC-20

There has been a recent buzz about Brc-20 and ordinals on Bitcoin, But in no time it has accumulated more than 3 USD Billion market capital since the writing of this paper. In short, Ordinals is a way of numbering Bitcoin sats, Sats is an indivisible unit of Bitcoin. numbering stats along with being able to inscribe data into sats using `Segwit` and `TapRoot` opens up wide opportunities in the Bitcoin ecosystem. It allows marking Sats as rare bringing the Non-fungibility of tokens to Bitcoin, ever since ordinals theory became a widely accepted standard by the community there have been a lot of NFT collections popping up on Bitcoin. The same goes for BRC-20 tokens, these tokens just function like ERC-20 fungible tokens on EVM chains,

Yet having such huge potential and market Capital, There hasn't been any truly decentralized solution for bridging Brc20 tokens to EVM or any EVM tokens to Brc20 Tokens. Xythum Bridge aims to solve this in its first rolling version.

Bridging Bitcoin To EVM

Bitcoin is today's most valued Currency in the whole web3 ecosystem. However, its limited functionality hinders its full potential within the ever-evolving DeFi landscape. Here's where Bitcoin bridging comes in, but current solutions fall short. Popular bridges like WBTC rely on centralized custodians like BitGo. This negates the very essence of decentralization that underpins Bitcoin and DeFi. A single entity controls billions of dollars in bridged assets, introducing a central point of failure and potential censorship. Xythum takes this as its main objective to bridge Bitcoin effectively, enabling EVM chains to access liquidity from the Bitcoin Chain.

Adding Bitcoin Forks to Protocol

Yet it's been over a decade there has been quite interesting debate continuing to scale Bitcoin, And here we are with a lot of Bitcoin Code forks and Bitcoin Layer 2’s which tend to make Bitcoin cheaper, faster, more efficient, and secure. Dogecoin, Litecoin, zcash, and Bitcoin Cash to name a few, it is so surprising to see the encouragement these chains receive from the community and they grew into all-time highs with Billions of volume flowing through chains every day. Despite having such huge potential liquidity in these chains is limited to a single execution environment only. Hence after the Bitcoin Bridging version, Xythum concentrates on onboarding bitcoin forks and establishing limitless liquidity.

Adding Bitcoin Forks to the protocol can be challenging because they don't support Schnorr Signature nor taproot scripts like Bitcoin does, hence protocol might opt for Fast Ecdsa Threshold Signing.

Aggregating Rollups and EVM Chains

Well just like we covered Bitcoin Layer 2’s, the Ethereum ecosystem also has a lot of Layer 2 and Layer 3 implementations trying to scale Ethereum making it cheaper, faster, and more efficient. These layer 2’s are termed rollups Arbitrum, Optimism, Scroll, Polygon Zkevm, and Mina to name a few. With evolving Ethereum ecosystems these rollups reach newer and newer heights every day in trading volume (52 USD billion dollars per day) as of writing this paper. The Ethereum ecosystem has also chains with code forks namely polygon POS, Avalanche, and fantom to name a few.

Well, almost all of these chains have their own bridge anchored to Ethereum Mainnet But lack a key aggregation layer where any chain can interact with any other chain in the Ethereum and Bitcoin ecosystem and Xythum is here to solve it once and for all.

Resolver Swaps

TBD ..

Dark Pool Guard Up

After onboarding and expanding its ecosystem with different blockchains, the Xythum protocol will implement Dark Pool into its order-matching algorithm. Dark pools are quite essential parts of finance markets where they constitute 15 - 30% of the whole wallet street trade every day. High net worth individuals (HNI’s) are subjected to huge slippage issues which can be vulnerable to trade effortlessly when they trade in Open markets. usually, Huge trades when traded in an open market are exploited by participants by sandwiching huge trades (putting a buy and sell order just before and after the Huge trade) causing a drastic fall in the price of the asset after the Buy. This is still a huge problem in the blockchain space too. to overcome this Xythum deploys an efficient order matching scheme where until the order is completely filled none of the participants will get to know actually volume, amount, and sender of the order, Order will be split into fragments based on Shamir's secret sharing and matched based on shares distributed across participants, None of the participants less than k (threshold) will be able to reconstruct the order. Once the order is settled everyone will be notified about the order and allowed to backrun it for profits.

Scaling

All actions taken after the Dark Pool upgrade would only be much focused on scaling Xythum to process orders faster and efficiently without compromising security. Here are two approaches Xythum would look into for its next upgrade.

- **Sharding**

  One of the best ways to do faster transactions might be parallel execution

  execution of orders and sharding can be the best choice to implement it. Xythum will opt for sharding when the number of nodes participating in the next epoch exceeds more than the secure Node limit. sharding includes having n Groups of computations where instead of single group pubKey there can be multiple groups and multiple pubKeys representing assets under groups.

- **Multi-Processing:**

  In order to achieve the Multiprocessing protocol has to lower its k-value

  such that SA (Signature aggregate) can pick multiple α such that it can form multiple sign operations happening at the same time, without making (n-α) participants stagnant and efficiently using the protocol to its limits.

Dime Wallet Integration

Context: Self-custodian Dime Wallet is going to be a child product under Xythum which aims to onboard the next billion web2 users in their style, Dime Wallet empowers users with familiar login methods like usernames and passwords or OAuth, all without compromising the security of the wallet anywhere. Dime wallet is Built upon Account abstraction EIP 4337 and Zero-knowledge proofs, which open wide opportunities for building wallets whose transactions can be signed by password strings and which can be changed over time on user demand, this lets users build

- recover wallets even after losing private Keys,
- allow third-party services to pay gas for users,

- Bundle transactions for cheaper gas efficiency
- session and programmable wallets

This level of convenience, built on a bedrock of security, makes Dime Wallet the perfect bridge for Web2 users to explore the vast potential of the decentralized world.

Summary

The Xythum Protocol tackles the challenge of connecting blockchains by proposing a decentralized network for seamless cross-chain trading. This allows users to swap assets between different blockchains, like Bitcoin and Ethereum while offering privacy features like dark pools to protect trade details. The protocol utilizes advanced cryptography and operates in phases, with the initial focus on bridging tokens within the Ethereum ecosystem. As it progresses, the plan is to integrate Bitcoin, other blockchains, and even Layer 2 solutions, ultimately aiming to create a highly interoperable and scalable environment for decentralized finance.

References

1. Shamir, A. (1979) How to Share a Secret. Communications of the ACM, 22, 612-613. http://dx.doi.org/10.1145/359168.359176
2. Laura Savu. SIGNCRYPTION SCHEME BASED ON SCHNORR DIGITAL SIGNATURE https://arxiv.org/ftp/arxiv/papers/1202/1202.1663.pdf
3. Chelsea Komlo, & Ian Goldberg. (2020). FROST: Flexible Round-Optimized Schnorr Threshold Signatures.
4. Petkus, M. (2019). Why and How zk-SNARK Works. *ArXiv*. /abs/1906.07221

5. BIP-0340 Schnorr Signatures on Bitcoin https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki
6. Casey Rodarmor. (2022) Ordinals BIP https://github.com/ordinals/ord/blob/master/bip.mediawiki
7. Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, & Michael Bæksvang Østergård. (2020). Fast Threshold ECDSA with Honest Majority. .
8. Vitalik Buterin (@vbuterin), Yoav Weiss (@yoavw), Dror Tirosh (@drortirosh), Shahaf Nacson (@shahafn), Alex Forshtat (@forshtat), Kristof Gazso (@kristofgazso), Tjaden Hess (@tjade273), "ERC-4337: Account Abstraction Using Alt Mempool [DRAFT]," *Ethereum Improvement Proposals*, no. 4337, September 2021. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-4337.