Xythum Labs

Trustless Decentralized Aggregation Layer For BlockChains

Jayendra Madaram

1st December, 2024

First Revision

STATUS : DRAFT

Xythum Labs.

Hyderabad, India

https://xythum.io

hello@xythum.io

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.

Xythum introduces a groundbreaking decentralized aggregation layer that transcends traditional blockchain boundaries. Leveraging advanced cryptographic primitives, including Secure Multi-Party Computation (SMPC) with FROST threshold signatures based on Schnorr signatures, Fully Homomorphic Encryption (FHE), and zero-knowledge proofs (ZKP), Xythum delivers an unprecedented level of privacy and efficiency.

Our novel architecture implements a sophisticated dark pool mechanism powered by Trusted Execution Environment (TEE) with Intel SGX hardware enclaves, ensuring complete transaction privacy. By preventing potential MEV and protecting sensitive trade details, Xythum offers traders a secure, anonymous execution environment across multiple blockchain networks.

The platform provides universal interoperability, enabling seamless liquidity access across diverse blockchain ecosystems. Through innovative cryptographic techniques, Xythum eliminates traditional barriers, creating a trustless, high-performance aggregation layer that redefines cross-chain interactions.

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.

Contents

0.1 Introduction . . . 3
0.2 MOTIVATION . . . 4
0.3 DEEP DIVE . . . 4
0.4 Background. Background . . . 6
  0.4.1 Schnorr Signature . . . 6
  0.4.2 Threshold Signature Scheme . . . 7
  0.4.3 Shamir key sharing . . . 7
  0.4.4 Additive Secret Sharing . . . 8
0.5 Distributed Key Generation . . . 8
0.6 Signature Aggregator . . . 9
  0.6.1 Roles of Signature Aggregator . . . 9
0.7 Nonce and Signature Generation . . . 9

0.7.1 Preprocess Stage. 10
0.7.2 Signing And Signature Aggregation 10
0.8 Key Rotation and Epochs 11
0.9 Key Generation and Transfer Protocol 12
0.9.1 Pre-Epoch Preparation 12
0.9.2 Cross-Chain Asset Ownership Transfer 12
0.9.3 System Components 13
0.10 SYNC NET (Xythum's StateChain) 14
0.11 Dark Pools On Xythum 15
0.12 Xythum Life Cycle (Routing Orders Through Dark Pools) 15
0.12.1 Setup Phase 16
0.12.2 Transaction Life Cycle 16
0.12.3 Hardware Enclave (Tee and Sgx) 18
0.12.4 Dark-Accounts. Dark-Accounts 19
0.13 System Performance And Scalability 21
0.14 System Attitude 24
0.15 Tokenomics and Incentives 24
0.15.1 Xythum TOKEN 24
0.15.2 STAKING 25
0.15.3 SLASHING and REPUTATION 25
0.15.4 RE-BALANCE BONUS 26
0.15.5 FEES 26
0.15.6 ARBITRAGE 26
0.15.7 LENDING / LIQUIDITY PROVIDING 26
0.15.8 GOVERNANCE 26
0.16 System Assumptions 27
0.17 Adversary Assumptions 27
0.18 Security Model 28
0.18.1 SIGNATURE AGGREGATE POISONING 28
0.18.2 HTLC ATTACKS 29
0.19 Attacks 29
0.19.1 Value Extraction 30
0.19.2 Fault Tolerance And DDOS 30
0.19.3 Sybil Attack 31
0.20 ROADMAP 31
0.21 Summary 33
0.22 References 34

0.1 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**" 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.

0.2 MOTIVATION

- CrossChain Trades and Universal Interoperability
- Private Execution
- Scalable, Fast and Cheap Cross chain Communication
- Asset and Context Transfer
- Decentralized Bridging

0.3 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 signatures 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 pre-compiled 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.

FROST can either be used as a two-round protocol where the signer sends and receives a total of two messages, or it can be optimized as a (non-broadcast) single-round signing protocol with a preprocessing phase.FROST achieves efficiency gains without the optimistic case of participant misbehavior. Because the preprocessing round can be performed separately from the signing round, the signing operation can be performed asynchronously; once the preprocessing round is completed, the signer only needs to receive and eventually reply to a message to create a signature.

FROST achieves its efficiency improvement in part by allowing the protocol to abort in the presence of a misbehaving participant (who is then identified and excluded from future operations) - a reasonable model for practical deployment scenarios. During the course of the protocol, if a misbehaving participant provides malformed values, honest parties can identify and exclude the misbehaving participant and re-run the protocol. the flexible design of FROST allows it to support many practical use cases of threshold signatures. In addition, while some threshold schemes require all participants to be active during the signature operation and refer to the threshold property of the protocol as simply a security property, FROST allows any threshold number of participants to generate valid signatures. Thus, FROST can support use cases where a subset of participants (or participating devices) can remain offline, a property that is often required for security in practice.

0.4 Background. Background

0.4.1 Schnorr Signature

Typically, signatures generated by a threshold signature operation should ideally be indistinguishable from those generated by individual participants, thereby maintaining backward compatibility with existing systems and preventing privacy breaches of individual signers' identities. Signatures generated by the FROST signature operation are indistinguishable from Schnorr signatures and can therefore be verified using the standard Schnorr verification operation. For this purpose, we now describe the Schnorr signature and verification operations in the single-signer setup .

Signing. Signing To sign a message, M :

- Choose a random k from the allowed set.
- Let $r = g^k$.

- Let $e = H(r \parallel M)$, where $\parallel$ denotes concatenation and r is represented as a bit string.
- Let $s = k + xe.$

The signature is the pair, $(s, e).$
Note that, $s, e \in \mathbb{Z}/q\mathbb{Z}; ifq < 2256q < 2^{256}$, then the signature representation can fit into 64 bytes.

Verification. Verification

- Let $r_v = g^s y^e$
- Let $e_v = H(r_v \parallel M)$.

If $e_v = e$ then the signature is verified.

0.4.2 Threshold Signature Scheme

A threshold signature scheme is a cryptographic fundamental used to facilitate common ownership of a private key by a group of participants such that a threshold number of participants must cooperate in issuing a signature that can be verified by a single public key. Threshold signatures are useful in a range of situations where a distributed root of trust needs to be established between a set of equally trusted parties. Similar to signature operations in a single-party environment, some implementations of threshold signature schemes need to perform signature operations at large scale and under heavy load. For example, threshold signatures can be used by a group of signers to verify financial transactions in cryptocurrencies, or to sign a network consensus generated by a group of trusted institutions. In both examples, as the number of signatories or signing operations increases, the number of communication rounds required between participants to generate a joint signature becomes a performance bottleneck, in addition to the increased load experienced by each signer. This problem is further exacerbated when signers utilize network-limited devices or unreliable networks for transmission, or when protocols are desired that allow signers to participate in signature operations asynchronously. Therefore, optimizing the network overhead of the signature operation is highly beneficial for practical applications of threshold signatures.

0.4.3 Shamir key sharing

Many threshold schemes are built on Shamir secret sharing, a (t,n) threshold signature scheme that relies on Lagrangian interpolation to recover secrets. In Shamir secret sharing, a trusted central dealer distributes a secret s to n participants in such a way that any cooperative subset of t participants can recover the secret. To distribute this secret, the dealer first randomly chooses t-1 coefficients a1,... , at-1, and uses the randomly chosen values as coefficients to define a polynomial of degree t-1 f (x) = s + SUM (ai xi, i = 1... . t-1), where f (0) = s. Subsequently, the secret share of each participant Pi is (i, f (i)), and the dealer is trusted to honestly assign to each participant P1,... , Pn. , Pn. To reconstruct the secret, at least t participants perform Lagrangian interpolation to reconstruct the polynomial and thus find the value s = f(0). However, groups with fewer than t participants cannot reconstruct the secret because at least t points are needed to reconstruct the polynomial of degree t-1.

0.4.4 Additive Secret Sharing

Similar to the single signature described above, FROST requires the generation of a random number k for each signature operation. However, in the threshold signature, it is supposed that each participant is involved in the generation of k, but does not know its result. Although Shamir secret sharing and its derived structure require sharing on the secret polynomial f, where f(0)=s, the additive secret sharing scheme allows t participants to jointly compute a shared secret s, with each participant Pi contributing a value si, such that the resulting shared secret is s=SUM(si, i=1.. . t), i.e., the sum of each participant's shares. Thus, this t-out-of-t secret sharing can be done non-interactively; each participant directly chooses its own si. where si is the additive secret sharing of s. Then s is the sum of si, and then (si)/(Li) is the Shamir secret sharing of the same s, where Li is the Lagrange factor. In FROST, participants use this technique non-interactively in the signature operation to generate a one-time secret nonce which is the Shamir secret shared among all t signature participants.

Each Epoch has two phases of execution as described Below.

0.5 Distributed Key Generation

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

Unlike threshold schemes such as Shamir secret sharing, which rely on trusted dealers, Distributed Key Generation (DKG) ensures that each participant contributes equally to the generation of shared secrets. At the end of the protocol run, all participants share a joint public key Y, but each participant holds only a share of the corresponding secret, so that no group of participants smaller than the threshold knows about it. Distributed key generation (DKG) supports this threat model by making each participant contribute equally to the generation of the shared secret. At the end of the protocol run, all participants share a joint public key Y, but each participant holds only a share of the corresponding secret s, such that no set of participants smaller than the threshold knows s.

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(nˆ2) 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.

0.6 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.

0.6.1 Roles of Signature Aggregator

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

0.7 Nonce and Signature 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.

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.

0.7.1 Preprocess Stage.

1.Create an empty list Li For each Participant. then, for 1 <= j <= Q, do the following

- The single-use nonces sample (dij, eij) <-$- Zq∗ x Zq∗
- Derive the commitment shares (Dij, Eij) = (g^{dij}^, g^{eij}^)
- Append (Dij, Eij) to Li. store ((dij, Dij), (eij, Eij)) for later use in signature operations
- Each Party publishes (i, Li) to a predefined location, specified by the implementer.

0.7.2 Signing And Signature Aggregation

Let SA denote the signature aggregator (who himself can be one of the t signature participants), S be the set of participants selected for this signature operation, B = < (i, Dij, Eij) for i in S> denote the ordered list of participant indexes corresponding to each participant Pi, and Li be the set of available commitment values for Pi announced in the preprocessing phase. Each identifier i is coupled to the jth commitment (Dij, Eij) published by Pi that will be used for this particular signature operation. Let H1, H2 be hash functions whose outputs are in Zq∗.

1. SA first obtains the next available commitment from Li for each participant Pi in S and constructs B. 2.
2. For each i in S, SA sends a tuple (m, B) to Pi.
3. After receiving (m, B), each Pi first verifies the message m and then checks Dp j, Ep j in G∗ for each commitment in B. If either check fails, it aborts.
4. Then, each Pi computes the set of bound values rp = H1(p, m, B), p in S. Each Pi then derives the set of commitments R = PROD(Dpj ∗ (Epj)^{rp}, p in S), and the challenge c = H2(m, R).
5. Each Pi computes their response by computing zi = dij + (eij ∗ ri) + Li ∗ si ∗ c, using their long-term secret shared si, and using S to determine Li.
6. Each Pi safely removes ((dij, Dij),(eij, Eij)) from their local storage and returns zi to SA.
7. The signature aggregator SA performs the following steps

(a) Derive ri = H1(i,m,B), Ri = Dij ∗ (Eij)ˆ{ri} for i in S, followed by R = PROD(Ri, i in S), c = H2(m,R)

(b) by checking that gˆ{zi} = ? = Ri ∗ {Yi}ˆ{c ∗ Li} for each signature sharing zi, i in S to verify the validity of each response. If not equal, first identify and report misbehaving participants and then abort. Otherwise, continue

(c) Compute the response of the group z = SUM(zi, i in S)

(d) Publish the signature SIG = (z, c) with the message m.

SA finally checks whether the zi reported by each participant matches with their committed shares (Dij, Eij) and their public key shares Yi . If each participant has issued the correct zi, then the sum of the zi values together with c constitutes the Schnorr signature on m. This signature will be correctly verified by a verifier who does not know that FROST was used to generate the signature, and who verifies it with Y as the public key using the standard one-sided Schnorr verification equation (Section 2.4). Handling transient pending shares. Since each nonce and commitment share generated in the preprocessing phase described in the preprocessing algorithm can be used at most once, the participant removes these values after using them in the signing operation, as shown in step 5 of the signing algorithm. An accidental reuse of (dij, eij) leads to the exposure of the participant's long-term secret si, so the participant must securely remove them and defend against snapshot rollback attacks as any Schnorr signature implementation would. However, if the SA chooses to reuse a commitment set (dij, eij) during the signing protocol, doing so simply causes the participant Pi to abort the protocol and therefore does not increase the SA's power.

FROST improves the state of the art for Schnorr threshold signatures by defining a signature protocol that can be optimized for (non-broadcast) single-round operations with a preprocessing phase. Unlike many previous Schnorr threshold schemes, FROST remains secure against known forgery attacks without limiting the concurrency of signature operations. Threshold signatures allow a private key being split into shares given to multiple participants, allowing a subgroup of them to generate a signature that can be verified by the group public key, as if it were signed by the original unsplit private key. It has many applications such as allowing multiple entities to manage a cryptocurrency wallet in a safer and more resilient manner.

0.8 Key Rotation and Epochs

An epoch in the Xythum network represents a discrete time interval—precisely one week—during which a specific distributed key governs cryptographic operations. This temporal segmentation serves multiple critical functions:

1. **Dynamic Network Composition**: Each epoch provides an opportunity for network topology reconfiguration. Nodes can join, verify their credentials, and be incorporated into the network's cryptographic fabric.

2. **Continuous Security Rotation**: By regularly regenerating cryptographic keys, the network mitigates risks associated with long-term key exposure. This approach significantly reduces the potential attack surface for sophisticated cryptanalytic attempts.
3. **Cross-Chain Synchronization**: The epoch mechanism provides a standardized temporal framework for coordinating cryptographic operations across diverse blockchain environments.

0.9 Key Generation and Transfer Protocol

The key generation process is a meticulously orchestrated cryptographic dance involving multiple participants:

0.9.1 Pre-Epoch Preparation

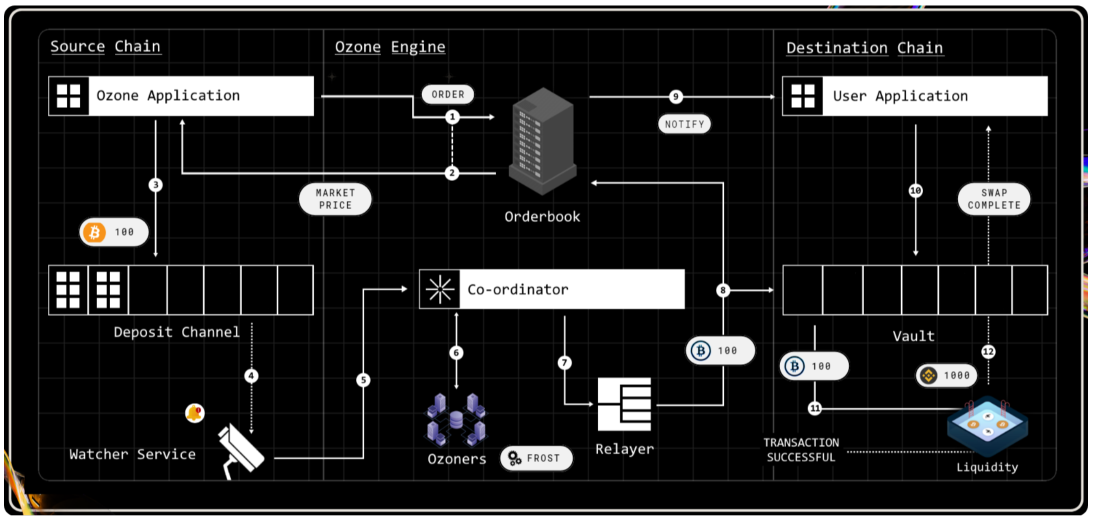
At the conclusion of each epoch, all registered network nodes are signaled to initiate the Distributed Key Generation (DKG) protocol. This process is not merely a technical procedure but a comprehensive network recalibration mechanism. The protocol evaluates multiple critical parameters:

- Verification of newly registered nodes
- Assessment of existing node performance
- Validation of stake commitments
- Cryptographic credential verification

0.9.2 Cross-Chain Asset Ownership Transfer

The first critical operation of each new epoch involves a synchronized, cryptographically secure asset ownership transfer. This process varies slightly across different blockchain architectures, On Ethereum and EVM-Compatible Chains The gateway contract's authority address is dynamically updated which requires a collective, threshold-signed transaction from the new epoch's node set. On UTXO-Based Chains (Bitcoin and Derivatives). Key rotation occurs by Consolidation of all unspent transaction outputs, Then Generation of a new aggregated public key, Finally Cryptographic transfer of asset ownership to the newly generated distributed public key

The conclusion of each epoch is marked by a sophisticated, cryptographically signed reward distribution transaction that serves as a critical mechanism for maintaining network health and participant motivation. This innovative approach ensures rewards are allocated proportionally based on node performance, creating a transparent and verifiable compensation model that directly incentivizes high-quality network participation. By implementing a precise reward distribution mechanism, Xythum cultivates continued engagement from network nodes, encouraging consistent performance, maintaining network security, and providing a clear economic incentive for participants to contribute their computational resources and maintain the integrity of the decentralized infrastructure. The cryptographic signing of the reward transaction further enhances trust, as it provides an immutable and auditable record of all compensation, thereby preventing potential manipulation and ensuring fair treatment of all network contributors.



0.9.3 System Components

At the heart of the Xythum ecosystem lies a robust and innovative protocol designed to facilitate truly decentralized, cross-chain interoperability. This protocol, often referred to as the "Xythum Core" or "Ozone Engine," leverages a combination of consensus mechanisms and secure multi-party computation (MPC) algorithms to overcome the challenges of blockchain fragmentation and enable seamless asset transfers across diverse distributed ledger environments. Let's explore these critical elements in detail

The Xythum Orderbook: The Xythum Orderbook is a crucial component responsible for managing user orders and facilitating the cross-chain transfer process. It consists of two main subcomponents:
a. Order Registry: This service registers user orders at the beginning of the cross-chain communication workflow. The Order Registry is tasked with reporting accurate market prices, providing secure deposit channels on the source chain where users lock their assets, and maintaining the order book itself.
b. Prover: The Prover service plays a pivotal role in enabling the platform's dark pool feature. It is responsible for generating zero-knowledge (ZK) proofs that demonstrate the validity of user orders without exposing any sensitive details about the order's origin or destination. These proofs are essential for preserving user privacy and preventing potential attacks, such as Miner Extractable Value (MEV) exploitation.

The Coordinator Service: The Co-ordinator service acts as the orchestration manager for the cluster of Xythum nodes, coordinating the distributed bridging process. It is also responsible for shard management, ensuring the efficient and scalable operation of the network. When a user locks their assets on the source chain, the Xythum Watchtowers signal the Co-ordinator service. The Co-ordinator then captures the ZK proof about the proof of funding on the source chain, as well as the Fully Homomorphic Encryption (FHE) encrypted order string, which serves as an "invoice" to be used on the destination chain for asset minting or release.

The Distributed Bridging Process: The distributed bridging process is a critical aspect of the Xythum protocol, facilitating the secure and decentralized transfer of assets across blockchain networks. Once the Coordinator receives the necessary proofs and order details, it initiates the distributed bridging process: a. Proof Validation: The Co-ordinator service first validates the order proofs against the latest blockchain state to ensure their validity. b. Threshold Signature Generation: The validated proofs are then distributed to the Xythum nodes, which collaborate to generate a threshold signature using the FROST (Flexible Round-Optimized Schnorr Threshold) signature scheme. This process ensures that no single node can independently derive the complete private key, maintaining the decentralized and trustless nature of the network. c. Signature Aggregation: The Co-ordinator aggregates the individual threshold signatures produced by the Xythum nodes and passes the final, consolidated signature to the user or relayer service. This signature can then be used to unlock the minted assets on the destination chain.

Defi Integrations: Xythum provides seamless integrations with a variety of decentralized finance (DeFi) applications, allowing users to easily convert their minted assets into any desired cryptocurrency or token. This integration expands the utility and liquidity of the cross-chain assets transferred through the Xythum protocol, enabling users to participate in a wide range of DeFi activities.

0.10 SYNC NET (Xythum's StateChain)

SYNC-NET, a custom-designed, distributed ledger system that enables all Xythum nodes to maintain a synchronized, consistent view of the system state. This innovative approach to state management is crucial for ensuring the integrity, fault tolerance, and scalability of the Xythum network. SYNC-NET is architected as a UTXO-like distributed ledger, where each node participates in the maintenance and validation of the global state. This design choice offers several key benefits:

By structuring SYNC-NET as an append-only ledger, Xythum creates an immutable record of all transactions and state changes. This historical data can be reliably audited and referenced, providing a tamper-resistant foundation for the protocol's operations. The UTXO model employed by SYNC-NET inherently guards against double-spending attempts, as each transaction input must reference a previously unspent output. SYNC-NET's role in the Xythum protocol extends beyond simply maintaining a shared ledger. It also serves as a key enabler for the platform's sharding capabilities. By recording shard-specific details, such as shard membership, state transitions, and cross-shard interactions, SYNC-NET provides a unified, verifiable view of the Xythum network's complex and dynamic structure.

The SYNC-NET ledger is composed of multiple types of transactions, each serving a specific purpose:
Intrinsic Transactions: These transactions represent the creation of new orders within

the Xythum ecosystem. Each order is recorded as an intrinsic transaction, which must be validated and confirmed by the Xythum nodes before any bridging operations can commence.
Extrinsic Transactions: Once a cross-chain asset transfer is successfully completed, an extrinsic transaction is added to the SYNC-NET ledger. This transaction represents the successful execution of the bridging process
System Transactions: SYNC-NET also records various system-level operations, such as network adjustments, fee updates, and governance changes. These transactions ensure that all protocol-level modifications are transparently recorded and can be audited by the network participants.
Epoch Transactions: At the conclusion of each epoch, SYNC-NET logs the details of the completed time period, including the newly generated distributed public key, node participation, and other relevant metadata. This historical record is crucial for maintaining the continuous security and integrity of the Xythum network.

0.11 Dark Pools On Xythum

The Xythum protocol takes privacy and security to new heights through its innovative implementation of dark pool functionality. By leveraging the power of Fully Homomorphic Encryption (FHE) and zero-knowledge (ZK) proofs, Xythum ensures that none of the node operators ever have complete knowledge of the order details, including the order origin, destination addresses, and the order amounts.

When a user initiates a cross-chain asset transfer, their order details are encrypted using FHE. This means that the order information, such as the asset amounts and the source and destination addresses, are completely obfuscated from the Xythum nodes. However, the nodes can still perform essential operations on the encrypted data, such as verifying the validity of the order.

To further strengthen the privacy guarantees, the Xythum node binaries are exclusively run within Trusted Execution Environments (TEEs), specifically utilizing Intel SGX hardware enclaves. This ensures that the node operators cannot tamper with the binary or extract any sensitive information, even if they have physical access to the hardware. On account-based blockchain networks like Ethereum, where on-chain ZK verification is possible, Xythum takes an additional step to enhance privacy. It implements an innovative **commitment tree**-based wallet system, where the orders and user wallets are completely decoupled. The user's wallet balances are maintained off-chain, providing an extra layer of privacy and preventing any potential information leakage.

0.12 Xythum Life Cycle (Routing Orders Through Dark Pools)

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.

0.12.1 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.

0.12.2 Transaction Life Cycle

ALICE INITIATE

- Alice can now create an Order on where
  - Src Chain → Bitcoin
  - from Amount → 1 Btc
  - RouteThroughDarkPool → true
- Alice, Order is registered with Orderbook and Orderbook checks for one of its available deposit channel “Pay-To-Taproot" addresses. and responds to Alice with a deposit Channel address.
- These deposit Channel addresses are unique, since every deposit channel address has a nullifier encoded in its Tapscript. (This nullifier is further used in computing zk proofs).

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 and the Watcher Service would Start the Mint process.
- First Step of Xythum's Darkpool Order routing starts here. Xythum zk prover now has to produce 2 proofs in order for Nodes to check Proof of funding and signing encrypted messages.
- **SPV Proof:** One of main Aim of Darkpool is to hide the amount and source Origin of the sender or initiator in our case Alice, In order to accomplish this we use zk-proof to generate a spv proof that there was a tx occurred in a certain block, and tx has enough amount mentioned as per Order and finally a nullifier issued by xythum Core is only funded.
- **FHE amount:** Second kind of proof is encrypting the amount and producing a proof such that the encrypted amount is valid and derived from funding amount, price point and fee deducted.

These proofs are further used in to agree on order validity yet not disclosing any information about order.

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.
- Now its nodes turn to validate the Order and provide their end Schnorr signatures. Firstly a Node would verify SPV proof and validate it in SyncNet. This makes sure that there was a valid funding to one of our deposit channels in a certain block and this proof was never used before.
- Secondly Node verifies amount proof and OrderHash Proof. such that no extra amount is being minted more than funded Amount.
- Up on all validations Node signs the Schnorr signature and submit back to SA
- 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.
- SA also adds an extrinsic tx settling the Order in SyncNet (StateChain level).
- Now it's up to the user to submit the signature and tx onChain or ask Relayer to do it on behalf of the user.
- 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.

0.12.3 Hardware Enclave (Tee and Sgx)

Trusted Execution Environments, such as Intel SGX, provide a hardware-based security solution that isolates specific code and data in a protected memory region, known as an "enclave." This enclave is designed to be impenetrable, even from the host operating system or other privileged software running on the same hardware.

At the core of the Xythum protocol lies a robust security architecture that leverages the power of Trusted Execution Environments (TEEs) to safeguard the integrity and confidentiality of critical system operations. Specifically, Xythum mandates that all participating nodes run their underlying blockchain infrastructure within the confines of a secure hardware enclave, leveraging the Intel Software Guard Extensions (SGX) technology.

The rationale behind this design decision is twofold: first, it ensures that the node operators cannot tamper with or mutate the Xythum binary, and second, it guarantees that the nodes are running the latest, approved version of the software, eliminating the possibility of outdated or compromised code execution. When Xythum nodes operate within a TEE, they can be certain that their critical functions, such as key management, cryptographic operations, and state transitions, are being executed in a secure and tamper-resistant environment. Any attempts to modify the node's behavior or extract sensitive information from the enclave would be detected and thwarted.

In addition to runtime security, Xythum also mandates that all data stored by the nodes, including transaction histories, order details, and system state, must be encrypted at rest within the TEE. This requirement ensures that even if an attacker manages to access the node's storage media, the sensitive information would remain completely inaccessible and unreadable. The encryption of data at rest is achieved through the use of hardware-backed key management within the TEE. The node's encryption keys are generated and stored securely within the enclave, never exposing them to the outside world. This design effectively eliminates the risk of key compromise, further reinforcing the overall security of the Xythum network.

To ensure the integrity of the Xythum binary, the protocol employs a secure boot process that verifies the authenticity and integrity of the software before allowing it to execute. This process involves the use of cryptographic signatures and a trusted chain of trust, anchored in the hardware-based security of the TEE.

0.12.4 Dark-Accounts. Dark-Accounts

Account based chains and smart contract compatible chains like Ethereum and Solana, provide a wide range of defi experience possible. Hence they need extra privacy at xythum we offer Dark-Accounts on chains where zk verification is possible. DarkAccounts take user privacy to next level, Dark Accounts Completely obscure user balances, order details, and transaction histories from external observation. Enable seamless asset representation and trading across multiple blockchain networks.

Each Xythum Dark account is a cryptographically secure structure defined as:

$$W := (B, O, F, K, r) \in F2MB + 8MO + 5MF + 9$$

Where:
B: A list of balances, each specifying a token type and its associated quantity.
O: A list of outstanding orders, defined by parameters such as token pairs, order type, size, and price.

F: A list of relayer fee approvals, including tokens used for fees and associated parameters.
K: A hierarchy of public keys for access control, including keys for root operations, order matching, settlement, and wallet viewing.
r: A secret random value that protects the wallet's contents against brute force and inference attacks.

Cryptographic Commitment
To ensure privacy and consistency, every wallet is committed to a global append-only Merkle tree maintained by Xythum's smart contracts. The commitment $C(W)$ is computed as:

$$C(W) := H(H(B) \,||\, H(O) \,||\, H(F) \,||\, H(K) \,||\, r)$$

Where H represents a cryptographic hash function. This commitment prevents tampering and ensures that the wallet's state is globally verifiable without revealing its content.

Operations The wallet supports various operations using a commit-reveal mechanism:

1. **Deposit** & **Withdrawal:** Users can move funds in and out of the Xythum protocol while maintaining privacy through zero-knowledge proofs.
2. **Order Management:** Users can add or cancel orders within their wallet, ensuring complete off-chain privacy until a match is found.
3. **Trade Settlement:** Matched orders result in encrypted outputs (notes), which users or their delegated relayers can redeem securely.

Privacy Enhancements To maintain privacy:

- Wallet updates include new randomness r, preventing correlation with previous states.
- Nullifiers ensure that wallet commitments and notes cannot be reused, securing the protocol against double-spending attacks.
- Off-chain storage options further reduce gas costs while preserving state integrity.

Wallet Commitments on Smart Contracts The Xythum smart contracts maintain a global append-only Merkle tree where wallet commitments are stored. This structure enables users to:

- **Verify Wallet State:** A wallet is valid if its commitment exists in the tree and its nullifiers have not been used.
- **Prove Valid Updates:** Updates to a wallet (e.g., deposits, withdrawals, or order modifications) require the submission of a zero-knowledge proof (ZKP) demonstrating that the new wallet state adheres to protocol rules.

The Merkle tree structure ensures scalability and efficiency:

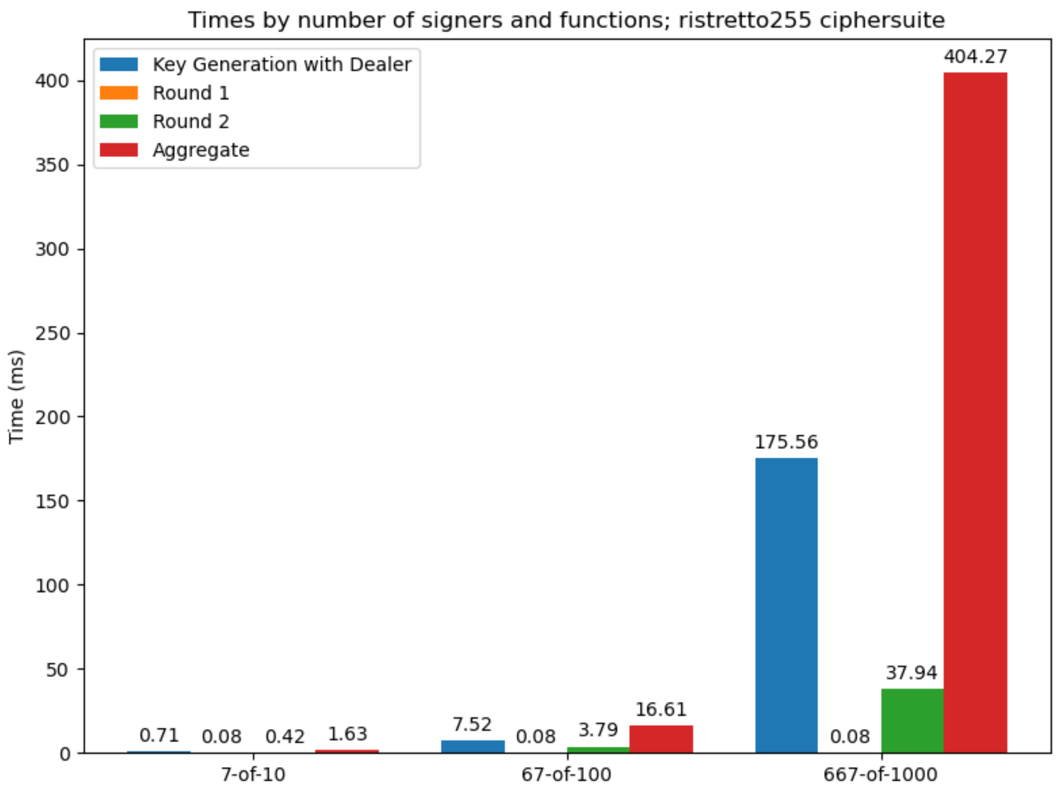
- **Privacy:** Wallets are represented by cryptographic commitments rather than plaintext data, concealing user balances and orders.
- **Atomicity:** Wallet updates, including order creation or settlement, are atomic operations, ensuring consistency even in adversarial conditions.

At a onChain Account level for a user balances and Orders are completely detached and kept off chain, by a user, Hence a Order would never be revealed onChain when it is created Order commitment is broadcasted on chain which doesn't reveal anything about order and user balance. Order details will be broadcasted within the network and xythum nodes can verify zkproof of Order validity and know nothing more about the owner of Order. when user wants to exit the system they would broadcast the final commitment of wallet settlement and cashout with extrinsic payout.

0.13 System Performance And Scalability

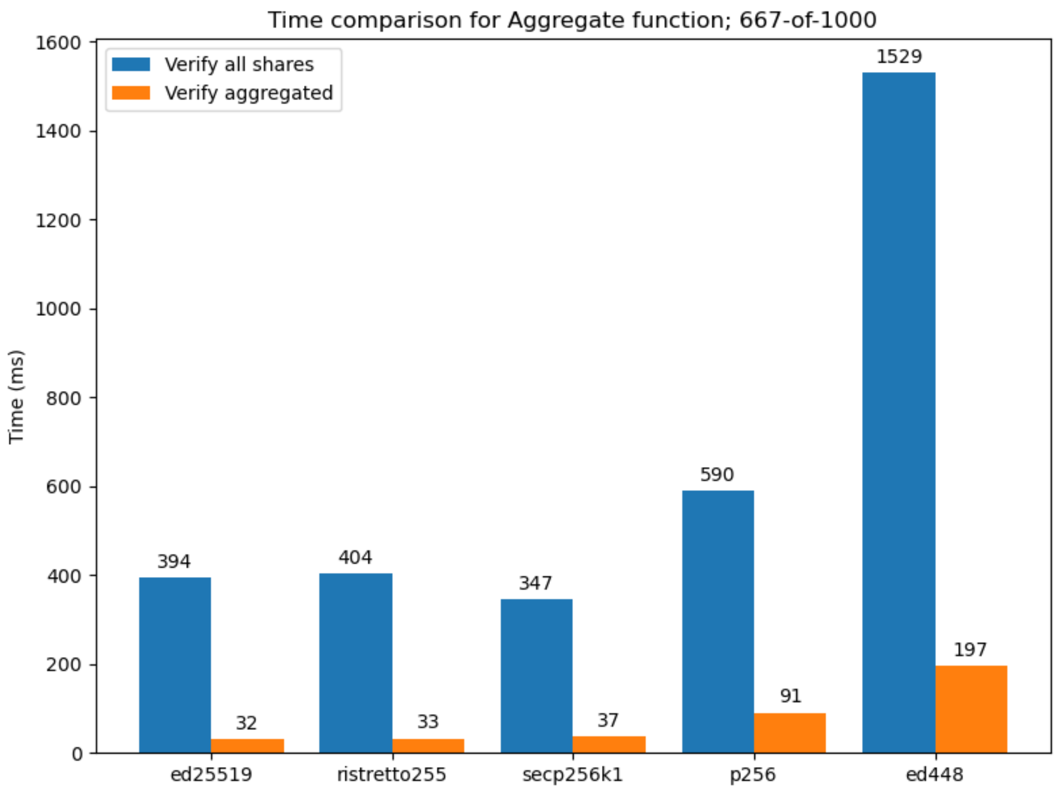
A crucial metric in the cross chain space, Speed of execution is really a finite metric that needs to be accounted for while validating a protocol. existing solutions in cross chain space have a reputation for being slow and stagnant. Having low bridging speed indicates that ownership of assets is held by bridges for a longer time and this is not ideal for any cross chain service. Keeping Speed and Cost in FROST was chosen as best shot for the protocol. Frost is known for its robustness. Here are few metrics on FROST

The FROST scheme can be split into steps. The first one is Key Generation, which only needs to be done once, while the rest are carried out each time the group wishes to generate a new signature. In Round 1, the participant generates commitments which are broadcast to all other participants via a Coordinator. In Round 2, using these commitments and their respective key shares generated during Key Generation, they produce a signature share which is sent to the Coordinator. Finally, the Coordinator carries out the final step, Aggregate, which produces the final signatures from all the signatures shares received.



It was expected that the timings would increase with the larger number of participants (with the exception of Round 1, which does not depend on that number), but the Aggregate timings appeared too high, surpassing 400ms for the 667-of-1000 case (which may not seem much but it's unusual for a signing procedure).

Here's the Aggregate performance comparison for all ciphersuites:



FROST proves to be the best choice for inter blockchain communication, Not just assets with Frost at its Core Xythum also support Context transfer (Inter smart contract calls) with Frost. Xythum will be further working on Frost variations and optimizations.

Apart from Frost there are several Factors which make Xythum Core scalable and efficient of all.

Threshold and Aggregated Signatures: Xythum threshold schnorr signatures scale pretty well. An aggregated threshold signature of 64 bytes is enough to represent security validation of entire thousands of nodes in Xythum. no callData bloating on blockchains, Simple signature.

Taproot and schnorr signatures: Taproot fork was recently activated on Bitcoin, yet all taproot addresses are 10% cheaper than segwit choices, Taproot and schnorr signatures take 40% less time than ECDSA in verification. and much faster in computing signatures.

Batching and Rbf: Xythum always opts for batching the orders for optimal Gas savings for the user and a for the protocol. users can still opt for high speed executions.

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

0.14 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.

0.15 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.

0.15.1 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

0.15.2 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.

0.15.3 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.

0.15.4 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.

0.15.5 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.

0.15.6 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.

0.15.7 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.

0.15.8 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.

0.16 System Assumptions

1. 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.

1. 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.

1. 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

1. 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.

1. 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.

1. 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.

0.17 Adversary Assumptions

1. 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.

1. 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.

1. 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.

0.18 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.

0.18.1 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.

1. 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.

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

1. 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.

0.18.2 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.

0.19 Attacks

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

0.19.1 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.

0.19.2 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.

0.19.3 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.

0.20 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.

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.

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.

0.21 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.

0.22 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.