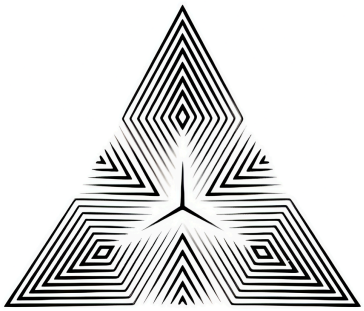


Xythum Labs

An MEV-Resistant EVM Chain with Threshold Encrypted Mempools

Jayendra Madaram
jayendra@xythum.io

Yashwant D.
yash@xythum.io

Protocol Specification, v0.1

www.xythum.io

Abstract

The increasing prevalence of Miner Extractable Value (MEV) poses a significant threat to fair transaction execution within blockchain networks. MEV refers to the additional profit that block producers (validators, sequencers) can extract by reordering, inserting, or censoring transactions within a block. This leads to an unfair trading environment, high slippage for users, and systemic inefficiencies across decentralized finance (DeFi) applications. Current approaches to mitigating MEV, such as priority gas auctions and private mempools, provide only partial solutions and are often limited by trust assumptions or centralization risks.

In this paper, we present an MEV-resistant Layer 2 solution built on the Binance Smart Chain (BSC). Our protocol leverages a Secure Multi-Party Computation (SMPC) network with a threshold encryption scheme using silent setup, ensuring that transaction details remain encrypted until a block is committed. The protocol introduces a group public key system where users encrypt transactions before submission, preventing sequencers from accessing transaction details before finalization. Additionally, users submit zero-knowledge proofs (ZKPs) alongside encrypted transactions to validate transaction integrity, including signature correctness, account balance verification, nonce validity, and gas fee sufficiency.

The Layer 2 operates with epoch-based sequencing, where an epoch leader collects encrypted transactions and assembles a block without decrypting individual transactions. Only after finalization do SMPC parties jointly decrypt the block, ensuring that transaction ordering remains resistant to front-running and other forms of MEV extraction.

Our approach achieves strong censorship resistance, eliminates information asymmetry, and ensures secure transaction execution without relying on centralized trusted parties. Through silent threshold encryption, we remove the need for an interactive Distributed Key Generation (DKG) process, significantly improving scalability and usability. Our work provides a robust framework for confidential, verifiable, and fair transaction execution on Layer 2 blockchains, setting a new standard for MEV resistance.

1 Motivation

MEV, or Miner Extractable Value, has emerged as a significant economic and technical challenge in blockchain ecosystems. MEV occurs when block producers whether miners, validators, or sequencers manipulate transaction ordering to maximize personal profits. This is especially prevalent in decentralized exchanges (DEXs), liquidations, and NFT minting, where transaction order can determine profit margins.

Common MEV strategies include:

- **Front-running**: Block producers insert their transactions before a user's, profiting from price movements.
- **Back-running**: Transactions are placed immediately after a user's to benefit from the price shift their transaction causes.
- **Sandwich attacks**: A user's trade is surrounded by buy and sell transactions from an attacker, leading to unfair slippage.
- **Censorship**: Validators refuse to include certain transactions, either due to bribes or competitive incentives.

1.a Why Existing Solutions Fail

Several existing MEV mitigation strategies have been proposed, but they fall short in critical areas:

1. Priority Gas Auctions (PGA)
   - Users bid higher gas fees to prioritize their transactions.
   - However, this often increases gas costs without solving MEV, as miners can still reorder transactions.
2. Private & Encrypted Mempools
   - Flashbots and MEV-Boost introduced off-chain private transaction relay networks.
   - These reduce frontrunning but require users to trust centralized relayers.
3. Fair Sequencing Services (FSS)
   - Solutions like Anoma and Espresso use cryptographic fair ordering.
   - However, these are resource intensive and still have sequencer trust assumptions.
4. Threshold Cryptography Based Solutions

- Some proposals use threshold decryption, but they rely on expensive interactive DKG setups. and per epoch DKG setups making real world implementation unfeasible.

Our motivation stems from the urgent need to preserve transaction confidentiality while ensuring fair and verifiable ordering. Existing mitigation techniques often require trusted intermediaries or impose prohibitive computational overhead due to interactive key generation protocols. In contrast, our protocol draws inspiration from the "Threshold Encryption with Silent Setup" paradigm. This approach eliminates the need for an interactive Distributed Key Generation (DKG) phase by enabling each participant to generate its key pair independently. The resultant group public key is derived deterministically from these individual contributions, allowing for a non-interactive, scalable setup that is particularly well suited for dynamic and large scale environments like those found on BNB Chain. This innovation is pivotal for creating "darkpools" encrypted mempools where transaction ordering is randomized and tamper resistant. The use of constant-size ciphertexts not only defends against size guessing attacks but also facilitates seamless integration with the EVM. Furthermore, the incorporation of zk proofs serves as a regulatory layer to prevent transaction spam while providing users with cryptographic guarantees regarding transaction validity (e.g., correct signatures, sufficient gas, valid nonces, and account balances).

Unlike existing approaches that may rely on complex and potentially vulnerable trusted setups or introduce significant performance overhead, our silent setup threshold encryption scheme offers a practical and efficient path to MEV resistance. It allows for the creation of a system where:

- **Transaction content remains confidential**: Sequencers cannot see the details of transactions before committing to a block, eliminating the ability to front-run, sandwich, or otherwise exploit user transactions based on mempool visibility.

- **Order-fairness is enforced**: The encrypted nature of the mempool ensures that transaction ordering is not influenced by transaction content, promoting a fairer and more predictable transaction execution environment.

- **Scalability and Efficiency are maintained**: By operating as a Layer 2, our solution inherits the scalability benefits of off-chain processing while

the underlying silent setup threshold encryption scheme is designed for concrete efficiency and scalability to large committees.

- **Silent Setup**: Utilizing a threshold encryption scheme with a silent setup eliminates the need for expensive and complex distributed key generation protocols. This enables asynchronous setup, multiverse support, dynamic threshold capabilities, and importantly, allows for scaling to larger validator committees compared to traditional threshold encryption schemes. By allowing parties to non-interactively derive a shared public key from independently generated keys, this primitive eliminates the need for complex DKG protocols. Committees can scale dynamically new nodes join by simply publishing a public key, while departed nodes are automatically excluded. Crucially, encrypted transactions remain secure even if up to **t-1** nodes are compromised, a property unattainable with traditional threshold schemes requiring synchronous setup.

2 Background & Preliminaries

In this section, we introduce the mathematical and cryptographic tools that underpin our MEV-resistant Layer 2 protocol. We begin by reviewing pairing-based cryptography and bilinear groups, then describe threshold encryption with silent setup, and finally detail the polynomial commitment and witness encryption frameworks that enable our silent threshold signature (STS) construction.

2.a Bilinear Groups and Pairings

Let p be a prime and F_p the finite field with p elements. We consider cyclic groups G and G_T of prime order p, with generators $g \in G$ and $g_T \in G_T$. A bilinear pairing is a map

$$e : G \times G \to G_T$$

satisfying the following properties for all $a, b \in F_p$ and $u, v \in \{G\}$:

- Bilinearity:

$$e\big(u^a, v^b\big) = e(u, v)^{ab}.$$

- Non-degeneracy:

$$e(g, g) \neq 1_{G_T}$$

- Computability: There exists an efficient algorithm to compute $e(u, v)$

In our implementation, we work over the bn256 curve, where the groups G and G_T are instantiated with efficient arithmetic.

2.b Threshold Encryption with Silent Setup

A traditional threshold encryption scheme allows a ciphertext c to be decrypted only if at least t out of n parties collaborate. Formally, a threshold encryption scheme is a tuple of algorithms

$$(\text{Setup}, \text{Enc}, \text{Dec}, \text{Combine})$$

with the following properties:

- **Key Generation**: Each party $i \in [n]$ independently samples a key pair.

$$(\text{sk}_i, \text{pk}_i) \text{ where } \text{pk}_i = g^{\text{sk}_i}$$

- **Silent Setup**: Define a deterministic function

$$f : G^n \to G, \quad \text{such that} \quad \text{ek}, \text{ak} = f(\text{pk}_1, ..., \text{pk}_n)$$

This encryption key ek is the sole public parameter needed for encryption.

> ak is eventually used in later decryption phase along with parties individual decryption shares.

- **Encryption**: Given a message $m \in F_p$ and a ciphertext-specific threshold t the encryption algorithm computes

$$c = \mathrm{Enc}(\mathrm{ek}, m, t)$$

where c is internally padded to have 2^n in length, preventing any party from size guessing attack.

- **Partial Decryption**: For each party, compute

$$\sigma_i = \mathrm{Dec}(\mathrm{sk}_i, c)$$

- **Decryption Aggregation**: Given a set $B \subset [n]$ with $|B| > t$, the original message is recovered as

$$m = \mathrm{Combine}\Big(\{\sigma_i\}_{\{i \in B\}}, \mathrm{ak}, c\Big)$$

Our silent threshold encryption (STE) scheme achieves non-interactivity by eliminating the costly interactive Distributed Key Generation (DKG) process.

2.c KZG Polynomial Commitments

To efficiently commit to vectors (e.g., public key vectors or indicator vectors) in our STS, we employ the KZG polynomial commitment scheme. Given a polynomial

$$P(x) = \sum_{\{i=0\}}^{\{d\}} a_i x^i, \quad a_i \in F_p,$$

and a secret $\tau in F_p$ from the common reference string (CRS), the commitment is computed as

$$\mathrm{Com}(P) = [P(\tau)] = \prod_{\{i=0\}}^{\{d\}} \left[\tau^i\right]^{a_i}$$

where each $[\tau^i]$ is precomputed in the CRS. Verification of an evaluation

Verification of an evaluation $P(x_0)$ at a point x_0 involves checking a pairing equation of the form

$$e\big(\mathrm{Com}(P)\cdot g^{-P(x_0)}, g\big) \stackrel{?}{=} e\big(g^{Q(x_0)}, g^{\tau - x_0}\big)$$

where $Q(x)$ is the quotient polynomial obtained by dividing $P(x) - P(x_0)$ by $x - x_0$.

2.d Signature Based Witness Encryption

Witness encryption allows an encryptor to hide a message mmm under a statement S with an associated witness w such that decryption is possible if and only if w is known. In our setting, we use a variant that builds on signature verification.

Let $H : \{0,1\}^* \longrightarrow F_p$ be a hash function (modeled as a random oracle) and consider a signature σ on a tag. In the case of a BLS signature, verification is performed by checking

$$e(g, \sigma) \stackrel{?}{=} e(\mathrm{pk}, H(\mathrm{tag})).$$

The witness encryption scheme then constructs a ciphertext for m as

$$c = (c_1, c_2) = (\alpha \cdot g, \alpha \cdot (\mathrm{pk} \bigcirc H(\mathrm{tag})) + m)$$

where $\alpha \in F_p$ is chosen uniformly at random and $\bigcirc$ denotes the pairing operation. Given a valid signature σ, decryption recovers the message via

$$m = c_2 - (c_1 \bigcirc \sigma)$$

2.e Silent Threshold Signatures (STS)

To build our encryption scheme, we compile a silent threshold signature (STS) with linear verification. Each party iii generates a key pair as in Section 3.2 and publishes additional auxiliary values

$$\big\{\big[\mathrm{sk}_i \cdot \tau^j\big]\big\}_{\{j=1\}}^{\{d\}}$$

which are used for linear verification. Given a challenge $\gamma \in G$, each party computes a partial signature

$$\sigma_i = \gamma \cdot \mathrm{sk}_i$$

For a subset $B \subseteq [n]$ with $|B| \geq t$ an aggregator computes the aggregate signature

$$\sigma^* = \sum_{\{i \in B\}} \sigma_i$$

and the aggregated public key

$$\mathrm{aPK} = \sum_{\{i \in B\}} \mathrm{pk}_i$$

Verification then requires that

$$e(g, \sigma^*) \stackrel{?}{=} e(\mathrm{aPK}, H(\mathrm{tag}))$$

which is a linear equation in the group elements. To ensure correct aggregation and to certify that the subset B is authorized (i.e., $|B|$ *get*) we use KZG commitments and associated proofs. In particular, if we represent the indicator vector $\mathrm{B} = (b_1, b_2, l..., b_n)$ as a polynomial

$$B(x) = \sum_{\{i=0\}}^{\{n-1\}} b_i x^i$$

then the commitment is given by

$$\mathrm{Com}(B) = [B(\tau)]$$

The aggregation is certified by proving a polynomial identity (a generalized sumcheck) of the form

$$\mathrm{SK}(x) \cdot B(x) = \mathrm{aSK} + Q_{x(x)} \cdot x + Q_{Z(x)} \cdot Z(x)$$

where $\mathrm{SK}(x)$ is derived from the secret keys sk_i and $Z(x)$

is the vanishing polynomial. The verification of this identity uses pairing checks:

$$e([\mathrm{SK}(\tau)], [B(\tau)]) \stackrel{?}{=} e(\mathrm{aPK}, [1]) \cdot e\Big(\Big[Q_{x(\tau)}\Big], [\tau]\Big) \cdot e\Big(\Big[Q_{Z(\tau)}\Big], [Z(\tau)]\Big)$$

refer [hinTS] which describes in depth To ensure that the aggregation is honest and that the set B indeed represents at least t parties, the aggregator must provide two succinct proofs, π_1 and π_2

- Proof π_1 Honest Aggregation of Public Keys (The goal of π_1 is to prove that the aggregated public key aPK is exactly the inner product of the full public key vector $(\mathrm{pk}_1, \mathrm{pk}_2, ..., \mathrm{pk}_n)$ with the indicator vector B.

$$\pi_1 = \Big(\Big[Q_{x(\tau)}\Big], \Big[Q_{Z(\tau)}\Big]\Big)$$

- Proof π_2: Authorization of the Signer Set (The proof π_2 is designed to ensure that the committed vector B is "authorized" in the sense that it represents at least t non-zero entries. In other words, we need to verify that:

$$\sum_{\{i=1\}}^{\{n\}} b_i get$$

$$\pi_2 = \Big(\Big[\hat{B}(\tau)\Big], \texttt{text}\{(\text{dummy party opening proof})\}\Big)$$

2.f Final Aggregated Verification

Once the proofs π_1 and π_2 are produced, the final signature (which, in our construction, is later compiled into an encryption scheme) consists of:

- Aggregated public key: aPK
- Aggregated signature: σ^*
- Commitment to the signer set: $[B(\tau)]$
- The proofs: $\Big[Q_{x(\tau)}\Big], \Big[Q_{Z(\tau)}\Big], \Big[\hat{Q}_{x(\tau)}\Big], \Big[\hat{B}(\tau)\Big], \Big[\hat{B}(\tau)\Big], [Q_0(\tau)]$

The entire system is then verified via the following set of linear pairing equations:

(Sumcheck): $[\mathrm{SK}(\tau)]_1 \bigcirc [B(\tau)]_2 \stackrel{?}{=} [1]_2 \bigcirc \mathrm{aPK} + [Z(\tau)]_2 \bigcirc \left[Q_{Z(\tau)}\right]_1 + [\tau]_2 \bigcirc \left[Q_{x(\tau)}\right]_1$

(Degree-check 1): $[\tau]_{\{2\}} \bigcirc \left[Q_{x(\tau)}\right]_{\{1\}} \stackrel{?}{=} [1]_2 \bigcirc \left[\hat{Q}_{x(\tau)}\right]_{\{1\}}$

(Signature Verification): $[\gamma]_2 \bigcirc \mathrm{aPK} \stackrel{?}{=} [1]_{\{1\}} \bigcirc \sigma^*$

(Degree-check 2): $[\tau^t]_{\{1\}} \bigcirc [B(\tau)]_{\{2\}} \stackrel{?}{=} [1]_{\{2\}} \bigcirc \left[\hat{B}(\tau)\right]_{\{1\}}$

(Dummy Party Check): $[1]_{\{1\}} \bigcirc [B(\tau)]_{\{2\}} \stackrel{?}{=} [\tau - 1]_{\{2\}} \bigcirc [Q_0(\tau)]_{\{1\}} + 1$

| Note: Read [HinTS] for theorem and derivations

2.g Zero-Knowledge Proofs and Hash Functions

To ensure that encrypted transactions satisfy system rules (e.g., valid signatures, sufficient balances, correct nonces) without revealing sensitive details, we integrate zero-knowledge (ZK) proofs. Let

$$\mathrm{zk} : \mathrm{NP} \longrightarrow \{0, 1\}$$

be a proof system such that for any statement x and witness w the relation

$$\mathrm{zk}(x, w) = 1$$

holds if and only if w is a valid witness for x In our protocol, the ZK circuit verifies algebraic constraints such as:

- $H : \{0,1\}^* \longrightarrow F_p$ (modeled as a random oracle),
- Pairing checks of the form $e(g, \sigma) = e(\mathrm{pk}, H(\mathrm{tag}))$
- Polynomial degree and evaluation constraints from the KZG commitments.

All such operations are expressed as algebraic equations over F_p, ensuring that our verification and encryption processes remain within the framework of succinct mathematical representations.

3 Protocol Overview

3.a Explanation of the Main Actors

Our protocol security and performance of our system rely on several distinct actor roles, each employing cryptographic primitives such as silent threshold encryption and zero-knowledge proofs. In this section, we explain the main actors and their responsibilities.

3.a.a Users (Browser Wallets / Applications)

Role and Goals:

- Privacy & MEV-Resistance: Users are end clients who create standard EVM transactions but require protection against MEV (e.g., front-running, sandwich attacks).

- Encryption & Proof Generation: Users encrypt their transaction tx using an encryption key ek that is deterministically derived from the aggregated public keys of the SMPC nodes. Formally, if tx is the plaintext transaction and Enc is our threshold encryption function, then the ciphertext is:

$$\mathrm{ct} = \mathrm{Enc}(\mathrm{ek}, \mathrm{tx})$$

- ZK Proof Attachment: Before submission, users generate an aggregate zero-knowledge (ZK) proof $\pi_{\{\mathrm{agg}\}}$ that asserts several properties:
  - $\pi_{\{\mathrm{enc}\}}$: The ciphertext ct is correctly produced under ek.
  - $\pi_{\{\mathrm{tx}\}}$: The transaction is well-formed (correct structure, size within global limits).
  - $\pi_{\{\mathrm{state}\}}$: The sender's balance is sufficient, as verified by the current Ethereum state root R (i.e., using a relation $f(R, \mathrm{balance}) = 1$)
  - $\pi_{\{\mathrm{gas}\}}$: The gas limit and nonce are valid (computed from the state root and transaction sequence).
  - $\pi_{\{\mathrm{sig}\}}$: And finally encrypted txobject signature is valid.

3.a.b SMPC Nodes (Xymers)

Role and Responsibilities:

- **Core Block Functions**: SMPC nodes form a decentralized committee responsible for:
  - **Block Production**: Collecting encrypted transactions and batching them into blocks.
  - **Encrypted Mempool** Maintenance: Managing a "darkpool" where transactions remain encrypted until block finalization.
  - **Threshold Decryption**: Each node i holds a secret sk_i and computes a partial decryption:

$$\sigma_i = \text{Dec}(\text{sk}_i, \text{ct})$$

$$m = \text{Combine}\Big(\{\sigma_i\}_{\{i \in B\}}\Big)$$

- **Dynamic Membership**: Thanks to our silent threshold encryption scheme, nodes can join or leave non-interactively. Membership is secured via staking x BNB on the mainnet in a stake-manager contract. A node's identity is authenticated by verifying that its address appears in the on-chain registry. Stake would be dynamically updated by DAO such that it makes economical infeasible for any party to put system in jeopardy.

- **Secure Communication**: SMPC nodes use secure peer-to-peer channels (built on Noise protocols and libp2p libraries) to exchange messages. authentication is validated against user stake address on L1 (BNB chain)

After Every resize in the protocol, either by slashing, interactively participants leaving the system or joining the system Current Epoch Leader would run a consensus and update the groupPubkey and proofs for aggregation, For every pubkey update Gateway contract on L1 (BNB chain) is updated accordingly representing new grouppubkey which users can now query to encrypt txs. The system is Byzantine fault tolerant by design, with the threshold t dynamically adjusting to remove misbehaving nodes (slashing) and to maintain a secure quorum.

3.a.c Epoch Leader (Block Proposer)

Roles and Responsibilities:

- Elected Leadership: For each epoch, a block proposer is elected using a distributed randomness mechanism. This mechanism leverages techniques such as a VRF-based scheme (similar to Ethereum 2.0's RANDAO+VDF) ensuring unbiased and unpredictable selection.

- Block Assembly: The elected leader (or block proposer) is responsible for:
  - Batching pre-validated encrypted transactions (whose associated π_{agg} proofs have been verified off-chain or by SMPC nodes).
  - Constructing the block header and including necessary metadata.

- Threshold Selection for Decryption: After block formation, the leader randomly selects a subset of SMPC nodes (satisfying |B|≥t) to collaboratively perform the threshold decryption. The decryption key is computed from the aggregated partial decryptions. $|B| > 1$.

- Accountability: The leader is held accountable via slashing mechanisms if block production fails due to high latency or misbehavior, ensuring system robustness.

In practice all smpc nodes would be maintaining a validated set of txs beforehand by processing zk proofs for encrypted transactions as soon as they enter mempool (parallelly processed), upon election, Leader would propose his prebuilt block right-away and commit it to the chain. If all threshold of SMPC nodes are able to decrypt the block and change state of chain, then block is finalized in single slot. If nodes weren't able to decrypt the block respected parties will be punished and key resizing would occur in next epoch.

3.a.d Proof-Production Service

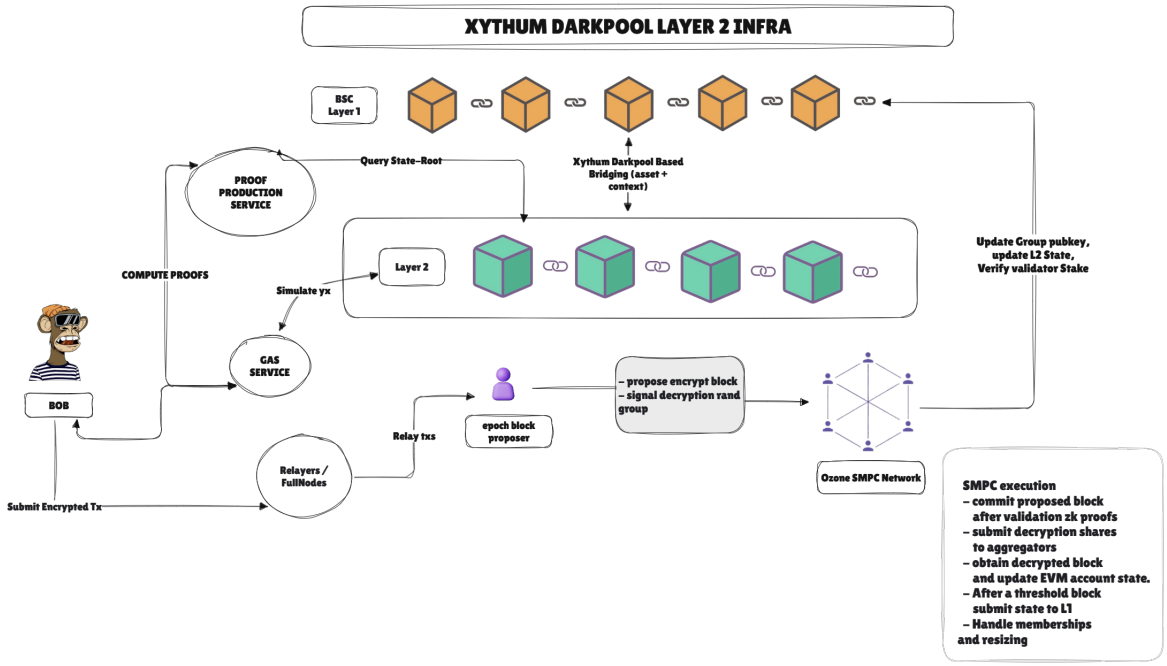
Roles and Responsibilities:

- Off-chain ZK Computation: This service is an off-chain infrastructure component that accelerates the generation of the ZK proofs (π_{agg}) attached by users.

- Hardware Acceleration: Utilizing specialized hardware (e.g., GPUs, FPGAs) and optimized cryptographic libraries, the service reduces the computational burden on clients.

- Service Model: Typically run by third-party providers under the umbrella of the “Xythum Treasure” initiative, these services integrate seamlessly via API endpoints and are designed to be decentralized and permissionless.

4 Transaction Lifecycle



The operation of our MEV-resistant Layer 2 protocol begins long before any individual transaction is submitted. At the foundational level, SMPC nodes (referred to as “Xymers”) must first join the network. To ensure economic security and deter malicious behavior, each prospective SMPC node is required to stake a predetermined amount of BNB. This staking requirement is dynamically set by the Xythum DAO, ensuring that the staked value is sufficiently high so that any attempt to subvert the system would result in a substantial economic loss via slashing. Validators, who also function as node runners in the SMPC network, must stake the DAO determined amount on the Xythum Gateway contract deployed on the BNB mainnet. The same private key and public key pair used for staking is later employed in the cryptographic protocols for silent setup and decryption, thereby serving as the node’s persistent identity across both layers.

Once a node's membership is confirmed via the stake-manager contract, it immediately becomes part of the SMPC network. At this point, all nodes engage in a silent setup protocol. During silent setup, each node i independently generates a key pair $\left(\mathrm{sk}_i, \mathrm{pk}_i = g^{\{\mathrm{sk}_i\}}\right)$ and broadcasts its public key. The collective set of public keys $\{\mathrm{pk}_i\}_{\{i=1\}}^{\{n\}}$ is then used to deterministically compute the group public key, which encapsulates both the encryption key and the aggregation key needed for subsequent operations. Mathematically, the group encryption key ek is computed as a deterministic function and this value, together with its associated aggregation key, is updated on the L1 (BNB chain) Xythum Gateway contract. This ensures that all participants in the system can query a consistent and publicly verifiable encryption parameter.

The network then enters an epoch cycle. Every 32 epochs, the SMPC committee members engage in a decentralized random selection process utilizing a combination of RANDAO and verifiable random functions (VRFs) to elect the epoch leader. This leader, who acts concurrently as the block proposer, threshold group manager, and decryption share aggregator, is responsible for orchestrating block production and decryption throughout the epoch. The election process is carefully designed so that the selection is both unpredictable and unbiased, drawing inspiration from protocols such as those used in Ethereum 2.0 and Solana. Once the leader is determined, the epoch begins with that leader assuming its roles for the next 32 epochs.

At this point, the transaction lifecycle enters the user interaction phase. Consider a scenario in which a user wishes to swap 1000 USDC for DAI on a decentralized exchange (such as Uniswap) hosted on our Layer 2. The user constructs a transaction object in the standard EVM format that includes fields such as the sender address, target contract address, call data, gas limit, gas price, nonce, and signature. Despite the transaction being formatted as any other EVM transaction, the subsequent steps diverge substantially to protect it from MEV attacks. The user first encrypts the transaction using the group encryption key ek obtained from the L1 (BNB chain) Xythum Gateway contract. Formally, let tx denote the binary representation of the transaction; then, the user computes

$$\mathrm{ct} = \mathrm{Enc}(\mathrm{ek}, \mathrm{tx})$$

where the encryption mechanism is a witness-based encryption scheme derived from the silent threshold signature construction. In our setting, the witness encryption is expressed as:

$$\mathrm{ct} = \mathrm{ct}_1, \mathrm{ct}_2) = \left(\alpha \cdot g,, \alpha \mathrm{pk} \bigcirc \dot{H}(\mathrm{tag}) + \mathrm{tx}\right)$$

with $\alpha \in F_p$ chosen uniformly at random, H a hash function modeled as a random oracle, and tag a deterministic replay tag ensuring non-repetition of transaction nonces. To mitigate potential size-guessing attacks which could otherwise be exploited for MEV extraction the transaction binary is padded to a fixed length (for example, a power-of-two size 2^n) before encryption.

Simultaneously, the user must generate a comprehensive set of zero-knowledge (zk) proofs that attest to various aspects of the transaction's validity. The zk proofs serve as cryptographic guarantees that the encrypted transaction will not fail during execution. They encapsulate multiple validations:

- A proof that the encryption was performed correctly with the valid ek.
- A circuit check that the transaction is properly constructed and does not exceed a predefined maximum size.
- A verification that the sender's balance is sufficient to cover the transaction (this is achieved by proving, via a Merkle Patricia Trie or through a dedicated zkDB service, that the balance exceeds (gasPrice × gasLimit).
- A gas limit validation proof, which may also incorporate an attestation generated by a third-party proof-production service (using techniques akin to Private Information Retrieval for enhanced privacy).
- A nonce validity proof, ensuring that the nonce is correctly derived from the Ethereum state root.
- A signature proof, which confirms that the transaction's signature matches the sender's public key.

  Each of these proofs is generated using a combination of on-device computation and, where necessary, off-chain computation via the Proof-Production Service. In cases where the computational load is prohibitive, the user may request the SMPC node itself to perform the heavy zk computations. The resulting proofs are then aggregated into a single composite proof π_{agg} that accompanies the encrypted transaction payload. Once the encryption and proof generation are complete, the user broadcasts the encrypted transaction ct along with the aggregate proof π_{agg} to the

network's encrypted mempool. This mempool, maintained by the SMPC nodes in a "darkpool" fashion, ensures that the transaction data remains confidential until block finalization.

Upon receiving transactions, the epoch leader (block proposer) begins the process of transaction selection and validation. As transactions arrive, the leader performs an initial verification of the attached zk proof π_{agg} using a GROTH16-based verifier. Only transactions that pass this cryptographic check are admitted into the candidate block. In the event that the epoch leader fails to respond or produces an invalid block, the protocol's slashing mechanism is triggered, penalizing any misbehaving party. After a block is assembled, the epoch leader calculates the block metadata, including a commitment to the block's contents. Subsequently, the leader randomly selects a subset of SMPC nodes specifically, at least $t + k$ nodes, where k is a safety margin to provide their decryption shares. Each SMPC node computes its decryption share for the block's ciphertext by performing a partial decryption:

$$\sigma_i = \text{Dec}(\text{sk}_i, \text{ct})$$

where Dec denotes the partial decryption function derived from our threshold encryption scheme. These shares are then transmitted back to the epoch leader. Provided that the number of valid decryption shares meets the threshold t (with tolerance for up to k failures), the leader aggregates these shares using the function:

$$\text{tx} = \text{Combine}\Big(\{\sigma_i\}_{\{i \in B\}}, \text{ak}, \text{ct}\Big)$$

thereby reconstructing the plaintext transactions.

Once the decryption is complete, the epoch leader publishes the final decryption key (or equivalently, the aggregate decryption result) into a designated witness field in the block. This publication is performed in a manner that ensures all network participants can immediately access and use the key to decrypt the transactions within the block. At this point, every node in the SMPC network updates its local state by processing the decrypted transactions, and the new state is then synchronized across the system. Every 32 blocks (or epochs), the protocol consolidates the latest state and publishes a commitment to L1 (BNB chain) via the Xythum Gateway contract. This on-chain commitment provides an immutable record of the Layer 2 state

transition and facilitates finality by leveraging the security guarantees of the BNB Chain. Throughout the entire transaction lifecycle, the confidentiality of user transactions is maintained by ensuring that no sensitive information is ever revealed before the block is finalized. The combination of silent threshold encryption, zero-knowledge proofs, and secure multi-party computation guarantees that transaction data remains hidden in the encrypted mempool until after the decryption process is complete. Moreover, by embedding cryptographic checks such as ensuring the correct aggregation of public keys and the validity of zk proofs the protocol achieves robust security against MEV exploitation and other forms of adversarial behavior.

This end-to-end process leverages advanced cryptographic techniques and distributed systems principles to ensure that every transaction is secure, private, and immune to manipulation thus delivering a truly MEV-resistant environment for decentralized finance on our Layer 2 protocol.

5 Bonus content

L2 would have its own Native bridging and context transfer solution to anychain and schnorr signature compatibility (including bitcoin, sui, aptos, solana, evm chains) refer [Xythum-Whitepaper]

- decentralized and censorship resistant layer
- darkpool based order routing
- robust onRamp and offramp assets to L2

6 Scalability Considerations

STE Performance

- Setup time: No interactive setup required
- Encryption: 7ms per transaction
- Decryption: $<$ 1s for partial decryptions and $<$ 200 - 700 ms for aggregation with network size = 1024
- Communication overhead: O(n) in all phases

Validator Requirements

- quad-core CPU (8+ cores preferred), 8 - 32 GB RAM, and 2 TB NVMe SSD storage.

Network: High-bandwidth (100 Mbps+), low-latency internet with built-in DDoS protection. Software & OS: Secure Linux (e.g., Ubuntu LTS) running dedicated validator software with bn256 and P2P protocols.

7 System Assumptions

The system assumptions they we adopt are not fundamentally different and are only more provable secure than that of any other rollup stack.

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.

2. 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.

3. 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 BLS threshold.

4. 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.

5. 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.

8 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. This can be even enhanced by requiring the network validators to stake or provide a proof of stake on POS beacon chain as a part of entry requirements.

2. Censorship Attempts: We assume that adversaries may attempt to censor or prevent certain transactions from being processed by our system, either through network-level attacks or by leveraging their controlled nodes which is to the most extent mitigated due to the encrypted nature of the transactions.

9 Security Model

9.a Value Extraction

Xythum follows the novel threshold witness based encryption 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.

9.b 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 (SETUP) 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 (SETUP). 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.

9.c 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

10 System Attitude

Xythum Protocol provides following properties:

- Users can experience limitless EVM execution swaps without going through the hassle of underlying **Threshold Encryption**.
- User doesn't have to be online until order completion.
- Nodes executing users order never has any informational advantage and traders identity is private.
- Depending on traffic/user customization trades are executed in batches on chain.

11 Staking

An open network like 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 its Token as Stake token and the Stake amount is adjusted based on governance and the real world value of the Xythum Token. Staked tokens also yield more Xythum tokens over time.

12 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.

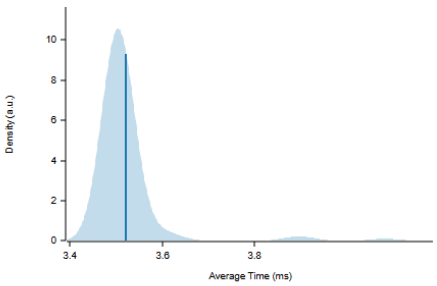
- Partial Slashing: A certain percentage of the total Stake amount is being burned, and such percentage is decided by protocol Governance. Following are scenarios where a participant would be subjected to partial slash.
  - Submission of improper commitment while Key Generation
  - Going offline or refusing to Sign a valid transaction signed by a majority of protocol participants.

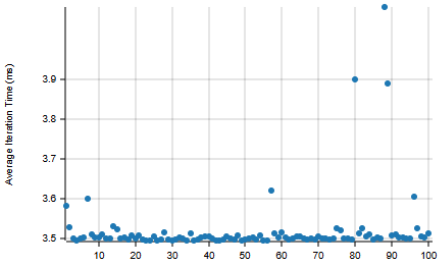
- Full Slash: The total Stake of participants is slashed and distributed among protocol Participants.
  - This occurs when SA recognizes the Attack vector to grow to a sufficiently large

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.

13 Benchmarks

encrypt





Additional Statistics:

	Lower bound	Estimate	Upper bound
R^2	0.0001012	0.0001040	0.0000994
Mean	3.5081 ms	3.5216 ms	3.5392 ms
Std. Dev.	21.469 µs	81.379 µs	120.50 µs
Median	3.5006 ms	3.5020 ms	3.5042 ms
MAD	4.0852 µs	5.8420 µs	8.0126 µs

Additional Plots:

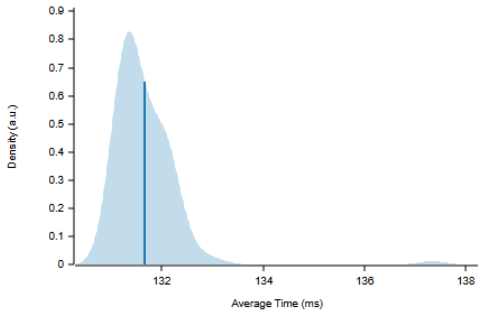
- Typical
- Mean
- Std. Dev.
- Median
- MAD

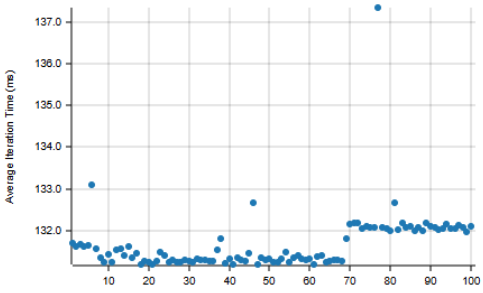
Understanding this report:

The plot on the left displays the average time per iteration for this benchmark. The shaded region shows the estimated probability of an iteration taking a certain amount of time, while the line shows the mean. Click on the plot for a larger view showing the outliers.

The plot on the right shows the average time per iteration for the samples. Each point represents one sample.

decrypt/1024





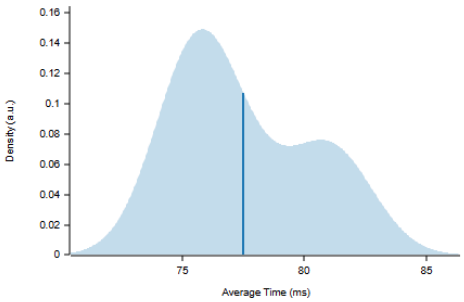
Additional Statistics:

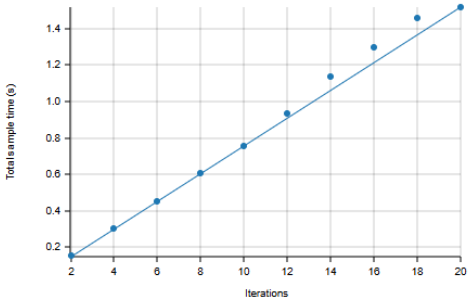
	Lower bound	Estimate	Upper bound
R^2	0.0035281	0.0036274	0.0034624
Mean	131.55 ms	131.67 ms	131.83 ms
Std. Dev.	362.71 µs	703.43 µs	1.0714 ms
Median	131.34 ms	131.42 ms	131.61 ms
MAD	149.16 µs	278.10 µs	537.33 µs

Additional Plots:

- Typical
- Mean
- Std. Dev.
- Median
- MAD

Lagrange setup/1024





Bibliography

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, EUROCRYPT 2005, volume 3494 of LNCS, pages 440–456. Springer, Heidelberg, May 2005. 13

[GJM+24] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. hints: Threshold signatures with silent setup. IEEE S&P 2024, 2024. https://eprint. iacr.org/2023/567. 3, 4, 7, 9, 10, 12, 14, 18, 25, 26, 29

[BGJ+23] Leemon Baird, Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. Threshold signatures in the multiverse. IEEE S&P 2023, 2023. https://eprint.iacr.org/2023/063. 3, 25

Eip-2537: Precompile for bls12-381 curve operations. Available: https://eips.ethereum.org/EIPS/eip-2537., February 2020. Ethereum Improvement Proposals, no. 2537,.