# ZEROBASE: At the Intersection of Trusted Computing and Verifiable Privacy

## I. Introduction

In recent years, as data sovereignty, privacy protection, and system transparency have become central themes in global policy and industry discourse, the demand for a technical foundation that enables ”controllable yet non-intrusive” infrastructure has become a critical challenge for the architectures of digital finance and the sharing economy. In traditional paradigms, on-chain execution provides verifiability but suffers from performance bottlenecks and limited privacy guarantees, making it ill-suited for high-frequency trading, complex strategies, and sensitive data processing. Off-chain systems, while more efficient, sacrifice verifiability and trustlessness, resulting in increased trust costs and structural risks for users.

ZEROBASE proposes a hybrid paradigm that integrates off-chain Trusted Execution Environments (TEEs) with on-chain Zero-Knowledge Proofs (ZKPs), achieving both privacy in execution and verifiability in outcomes—without reliance on centralized intermediaries [1]–[3]. This architecture leverages three foundational technologies—zero-knowledge proofs, hardware-based circuit-level encryption (e.g., AMD SEV-SNP), and programmable liquidity engines—to construct a highly available, modular, and sustainable privacy-preserving execution network. It provides unified, accessible, and trustworthy infrastructure for developers, asset providers, and end-users alike.

Therefore, ZEROBASE is not intended as an isolated protocol but as a generalized execution platform capable of supporting complex strategy deployment, resource assetization, and trusted off-chain computation—serving as a systemic anchor and universal trust layer for diverse application scenarios.

### *A. Vision and Mission: Restructuring Trust in Financial Execution*

The mission of ZEROBASE is to address two long-standing challenges in on-chain ecosystems: (1) the inability of users to verify complex strategies, leading to widespread ”black-box execution”; and (2) the inherent tension between privacy preservation and system transparency, which forces users and developers to trade off efficiency against trust.

We argue that a truly sustainable execution network must satisfy three essential conditions: preserving confidential space for strategists, defining clear risk boundaries for capital participants, and minimizing integration and composability costs for builders.

To this end, ZEROBASE aims to establish a philosophy of structural transparency—a system built not on trust assumptions, but on mathematical guarantees, architectural isolation, and standardized interfaces, thereby enabling a trustworthy multi-party collaboration framework.

Our vision can be summarized in four core pillars:

- Verifiable Strategy Execution: Support for complex strategies (e.g., hedge funds, algorithmic trading) to run privately off-chain, while exposing key risk metrics such as leverage ratios and value-at-risk (VaR) intervals via zero-knowledge range proofs—achieving “auditable privacy” without blind trust.
- Resource Assetization: Enable bandwidth, storage, GPU cycles, and other shareable resources to participate in DeFi mechanisms—collateralization, lending, atomic swaps—by generating on-chain attestations through trusted execution and ZK proofs, thereby unlocking their on-chain liquidity potential.
- Composable User Experience: All core modules (e.g., circuits, zkStaking, ZK browser) adopt standardized, nested-callable interfaces. Developers can easily compose and reuse capabilities to rapidly construct composite applications with minimal friction.
- Frictionless Ecosystem Collaboration: Through well-defined interface specifications and stable tooling, ZK applications can interoperate without requiring additional consensus layers, transitioning from isolated protocols to a structurally collaborative ecosystem.

Guided by these principles, ZEROBASE is committed to building a decentralized execution ecosystem that is structurally sound, clearly bounded, and collaboration-efficient—a platform that demands neither trust concessions nor privacy sacrifices.

### *B. Technical Philosophy and Design Principles: Reconstructing Trust Boundaries between Mathematics and Hardware*

ZEROBASE is not merely an incremental upgrade to existing paradigms—it represents a foundational reconfiguration of the trust model itself. It is a systemic logic that unites mathematical formalism, cryptographic mechanisms, and trusted hardware into a cohesive operational framework—where “trustlessness” becomes the default assumption.

This structurally transparent execution network is governed by three core design principles:

- Minimal Disclosure. Information is no longer disclosed as “raw data,” but as cryptographic “proofs.” Whether representing strategy risk, yield performance, or liquidity status, ZEROBASE employs range proofs or structured ZKPs to express system states, ensuring that users can establish trust without accessing underlying data. For example, statements like “leverage does not exceed 2x” or “daily yield falls between 0.1% and 0.3%” are presented as verifiable outputs, rather than plaintext disclosures.

- Trust Minimization. Off-chain execution in ZEROBASE operates within AMD SEV-SNP's confidential computing environment, with remote attestation verifying the integrity of the runtime. On-chain verification is performed via formally constructed ZK circuits. As a result, the system eliminates dependence on any single trusted party and ensures logical correctness through cryptographic primitives and architectural compartmentalization—achieving end-to-end structural trustlessness from hardware root-of-trust to governance mechanisms.
- Composable Proofs. In the ZEROBASE architecture, proofs function as the "intermediate language" for inter-module cooperation. Each strategy module outputs a unified state digest (e.g., risk intervals, performance metrics, solvency indicators [4]) that can be directly consumed by DeFi protocols, liquidation engines, or collateralized lending structures. This abstraction—"proof as interface"—dramatically reduces verification costs, clearly defines module boundaries, and fosters innovation across chains and execution domains.

ZEROBASE thus embraces a novel ethical framework: privacy is usable, trust is computable, and structure is composable. A truly trustworthy system requires no justification—it proves itself through its own architecture.

### *C. Market Landscape and Core Challenges: Fragmented Privacy Solutions and Absent Systemic Coordination*

Amidst rapid advances in cryptographic finance, distributed computing, and modular protocols, the greatest challenge today is not the absence of technological tools—but the lack of an integrated architecture that harmonizes them. Users demand privacy, capital demands verifiability, systems rely on off-chain execution, and resources seek on-chain monetization—yet existing architectures typically satisfy only one of these needs, often at the expense of the others.

This has led to several structural dilemmas:

- Users require privacy, yet protocol design mandates data transparency;
- Strategies need secrecy to preserve alpha, yet participants demand clear exposure metrics;
- System performance depends on off-chain execution, but verification logic is constrained by on-chain infrastructure;
- Resources are shareable but not composable, making "off-chain value" difficult to tokenize.

These tensions highlight the absence of structural coordination as a systemic bottleneck. From opaque operations on centralized platforms to the indiscriminate public exposure of on-chain data; from fragmented mining yield structures to inconsistent privacy standards—the broader crypto ecosystem lacks a unified hub that integrates execution, verification, and incentive mechanisms.

The Trust-Minimized Execution Network (TMEN) proposed by ZEROBASE is not tailored to a specific application. Rather, it is a foundational capability layer designed to make off-chain execution verifiable on-chain, to interconnect ZKP and TEE systems, and to unify resources and strategies in a composable structure. Its core value lies not in expanding application coverage, but in offering:

- A reusable bridge between off-chain and on-chain environments;
- An interface architecture that accommodates diverse proof systems and hardware models;
- A modular framework that abstracts assets, strategies, and resources into standardized I/O formats;
- A unified platform that synchronizes execution, verification, and incentive flows.

Built atop this interface layer, developers can freely build complex interactions, users can engage with confidence, resource providers can monetize participation, and the system as a whole gains evolutionary resilience through architectural alignment.

## II. System Architecture

ZEROBASE seeks to construct a trust-minimized and privacy-verifiable execution network that serves as a structural bridge of trust for off-chain computation, addressing the prevailing reliability gap between on-chain and off-chain systems. This network is built upon the coordinated operation of three core technological components:

- Trusted Execution Environments (TEE) for off-chain privacy-preserving and integrity-guaranteed computation;
- Zero-Knowledge Proofs (ZKP) on-chain to enable result verifiability without revealing sensitive data;
- A unified proof mediation layer (Proof Mesh) for cross-layer and cross-protocol standardization and interoperability.

This tripartite architecture not only ensures trust and privacy for off-chain computation, but also supports strategy abstraction, modular composition, and verifiable state transitions—forming a scalable foundation for decentralized finance (DeFi), shared computing, and AI computation.

### *A. Three-Layer Architectural Design: TEE, ZKP, and Proof Mesh*

The ZEROBASE execution network is organized into three architectural layers, each fulfilling distinct functional responsibilities and interfacing through standardized protocols. This layered design enhances system modularity, maintainability, and extensibility.

*1) Trusted Execution Layer (TEE Layer):* At the core of the execution network lies the TEE Layer, responsible for all sensitive off-chain computations, including strategy execution, resource scheduling, and data processing. Each strategy engine, transaction node, or resource provider operates within a hardware-isolated trusted execution environment, such as AMD SEV-SNP, Intel TDX, or ARM CCA [5]–[7]. TEE technologies enforce memory encryption and CPU-level access control to isolate the runtime environment and protect against external inspection or tampering.

To prevent malicious nodes from impersonating trusted computing agents, the system employs Remote Attestation mechanisms [5]–[7]. Each node must submit hardware-signed

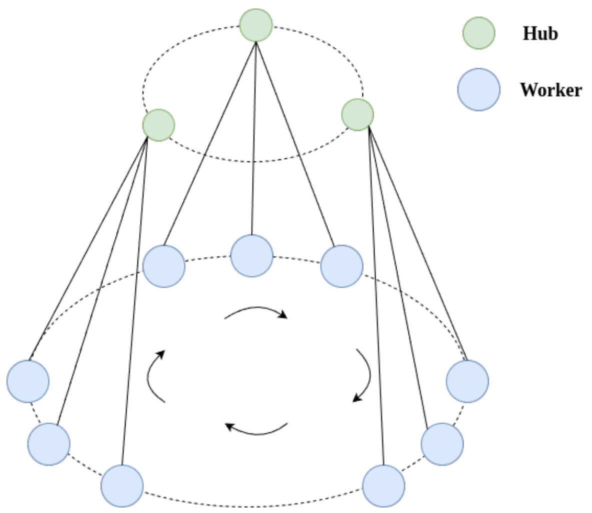

Fig. 1: Network Architecture

attestations during registration or execution, proving the authenticity, integrity, and consistency of its execution environment. These attestations form the basis for a decentralized reputation model that balances openness with security across the ZEROBASE network.

Logic executed within the TEE is subject to rigorous audit and encapsulation, functioning externally as a "black box." It accepts encrypted inputs from on-chain or off-chain sources, performs private computations internally, and produces both a final output and a proof digest for on-chain verification. Throughout this process, user data, strategy logic, and resource access patterns remain encrypted—ensuring full execution privacy.

*2) Zero-Knowledge Proof Layer (ZKP Layer):* Following execution within the TEE, results must be verifiably conveyed to on-chain validators. This task is handled by the ZKP Layer, which transforms key execution semantics into cryptographic zero-knowledge proofs. Serving as the system's "trust extension mechanism," the ZKP Layer enables each node to use standardized circuit templates to convert inputs, state transitions, and outputs into verifiable zk-SNARK or zk-STARK proofs [1], [8]–[11]. To promote generalizability and developer efficiency, ZEROBASE offers a standardized circuit library encompassing multiple commonly-used proof types [12], [13], including:

- Range Proofs: Verifying that a numerical value lies within a specified interval [12];
- Validity Proofs: Confirming that a computation was performed on valid inputs and states;
- State Transition Proofs: Verifying that a state change from $s_0$ to $s_1$ adheres to predefined logical constraints;
- Merkle Path Proofs: Proving inclusion of a specific element in an off-chain Merkle-based state tree.

The current implementation integrates major proof systems such as PLONK, Groth16, and STARKs to optimize for low verification cost and compact proof sizes. For long-term resilience against quantum threats, ZEROBASE also envisions incorporating lattice-based zero-knowledge systems, including constructions based on LWE and SIS [14], [15], to combine cryptographic hardness with programmability and structural flexibility.

*3) Proof Mediation Layer (Proof Mesh Layer):* The Proof Mesh serves as the structural adaptor between TEE-based off-chain execution and on-chain integration, representing one of ZEROBASE's most novel architectural components. It performs several critical roles that enable high-efficiency, high-generalizability, and high-composability coordination across layers.

First, the Proof Mesh standardizes all ZKPs produced by the TEE Layer. This involves formatting (e.g., unified field names and proof structures), semantic annotations (e.g., proof type, purpose, timestamp), and protocol version control. These standards allow on-chain smart contracts to interface with off-chain proofs via uniform APIs without needing to interpret implementation-specific logic—significantly lowering integration overhead.

Second, the Proof Mesh enables advanced proof-level functionalities, such as:

- Proof Aggregation. Compressing multiple independent proofs into a single proof to reduce on-chain verification overhead [9], [12], [16];
- Recursive Proofs: Embedding one proof within another, allowing hierarchical modeling of complex off-chain processes [9], [10], [30];
- Cross-Protocol Invocation: Facilitating the reuse and invocation of proofs across different modules and DApps, forming a "proof-as-a-service" data layer [17]–[19].

In addition, the Proof Mesh introduces version control and event subscription mechanisms, enabling support for "off-chain computation logs" and "on-chain event triggers." For example, a DApp can subscribe to the publication of specific types of execution proofs and trigger automated logic in response to newly generated proofs.

Through the coordinated operation of these three architectural layers—secure execution (TEE), verifiable translation (ZKP), and interoperable mediation (Proof Mesh)—ZEROBASE establishes a complete trust-preserving execution loop. Each off-chain computation is encapsulated into a standardized, succinct, and verifiable "proof package," ready for dissemination and invocation on-chain.

This design embodies a new execution paradigm where "results become interfaces, and proofs define structure," transforming off-chain execution from a black-box process into a first-class citizen within the on-chain economic system—composable, callable, and trust-minimized by construction.

## III. Network Topology and Node Management

To achieve both scalability and resilience against adversarial behavior, the ZEROBASE network adopts a hierarchical routing architecture, whereby Worker nodes are organized into subnets managed by Hub nodes. Each Hub is responsible for a subset of Worker nodes, and the routing structure is built on a Kademlia-like topology, where node identifiers (Node IDs) are derived by hashing the node's IP address and port number using SHA-1 (160-bit), which also serves as the basis

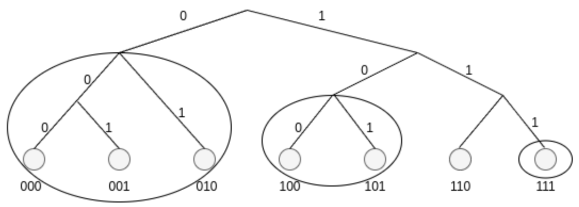

Fig. 2: Node Distance

for logical distance calculation and routing table construction [20].

### A. Node Identification and Logical Distance Metrics

Every node in the network—whether a Hub or a Worker—generates a unique 160-bit Node ID upon joining the system by hashing its IP address and port number:

$$\text{ID = SHA1(IP, Port)}$$

The logical distance between any two nodes is defined as the bitwise XOR of their Node IDs. For instance, given two nodes with IDs: $X = 1101, Y = 1000$.This XOR-based metric does not reflect physical proximity but is used solely for constructing logical routing paths. When a Worker joins the network, it computes its logical distance to all existing Hubs and binds to the nearest Hub, which is then responsible for maintaining and managing its state.

### B. Node Assignment and Distance-Based Layering

Consider a network with three Hub nodes: $\text{Hub}_0$, $\text{Hub}_1$, and $\text{Hub}_2$, with corresponding Node IDs $X$, $Y$, and $Z$. A new Worker node with ID $T$ joins the network. The Worker computes its XOR distance to each Hub:

$$D(T, X) = T \otimes X \quad D(T, Y) = T \otimes Y \quad D(T, Z) = T \otimes Z$$

Suppose

$$X = 0001 \quad Y = 0010 \quad Z = 0100 \quad T = 1000$$

Then

$$T \otimes X = 1001 \quad T \otimes Y = 1010 \quad T \otimes Z = 1100$$

Since the node ID is a 160-bit binary string, each Hub needs to maintain 160 lists (K-buckets) to store other Hub information, such as IP, port, and ID.

Consider the following scenario: Hub1's ID is 101. If the first digits of Hub2's ID are the same as Hub1's ID, and only the last digit is different, that is, 100, the XOR of the two Hub IDs is $101 \oplus 100 = 001$, that is, the distance is 1; for Hub2, Hub1 will put it into "K-Bucket 1".

Consider the IDs of some Hubs, where all the first digits are the same and the second-to-last digits are different. There are two such Hubs, whose IDs are 111 and 110, respectively. The XOR values of these Hubs with the ID of Hub1 are $101 \oplus 111 = 010$ and $101 \oplus 110 = 011$, that is, the distance ranges are 2 and 3. For these two Hubs, Hub1 will put them into "K-Bucket 2".

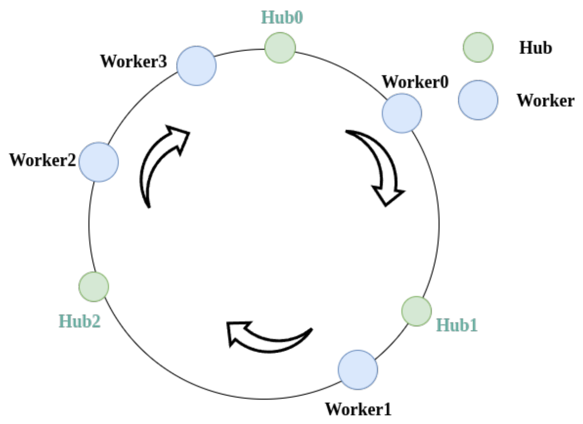

Fig. 3: Node Assign

Consider the IDs of some Hubs, where all the first digits are the same and the third digit from the last is different. There are only four such Hubs, and their IDs are: 000, 001, 010, 011, and the XOR values of these Hubs with the ID of Hub1 are $101 \oplus 000 = 101$, $101 \oplus 001 = 100$, $101 \oplus 010 = 111$, $101 \oplus 011 = 110$, that is, the distance ranges are 4, 5, 6, and 7; Hub1 will put these four Hubs into "K-Bucket 3". As shown in the figure below, for Hub with ID 101, it adds Hub with ID $0 \times \times$ to "K-Bucket 3", Hub with ID $11\times$ to "K-Bucket 2", and Hub with ID 101 to "K-Bucket 1".

If the first digits of a Hub's ID are the same, but the digits from the last digit are different, there are only $2^{i-1}$ such Hubs, and their distances from Hub1 are in the range of $[2^{i-1}, 2^{i}-1]$. For Hub1, these Hubs will be placed in "K-Bucket i".

TABLE I: Bucket

| K-Bucket | Distance Interval |
|---|---|
| K-Bucket 1 | $[2^0, 2^1 - 1]$ |
| K-Bucket 2 | $[2^1, 2^2 - 1]$ |
| K-Bucket 3 | $[2^2, 2^3 - 1]$ |
| $\vdots$ | $\vdots$ |
| K-Bucket 160 | $[2^{159}, 2^{160} - 1]$ |

As $i$ increases, there are $2^{i-1}$ Hub information in K-Bucket i, which will bring great storage overhead to the Hub. Therefore, it is stipulated that each K-Bucket only records the information of K Hubs, where K is an adjustable constant parameter. Since the ID length is 160 bits, each Hub maintains information of up to $160 \times K$ (K is usually 20) other Hubs. In addition, the position of the Hubs stored in each K-bucket is arranged in the order of the last time they were seen, with the earliest accessed at the head and the latest accessed at the tail.

### C. Instruction Set for Hub Nodes

Communication between Hub nodes is facilitated through a concise set of protocol-level commands. The core instruction set includes:

- PING: Used to probe the liveness of a target Hub node. Essential for maintaining the health of K-Buckets.

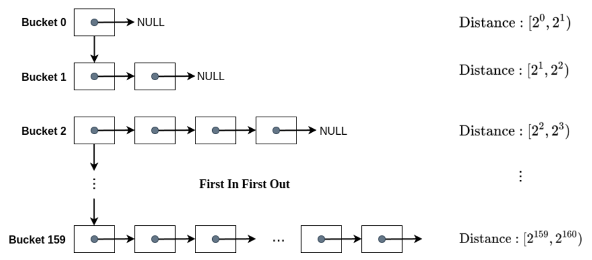

Fig. 4: K-Bucket

- ADD_WORKER: Instructs the recipient Hub to store state and routing information for a specified Worker node.
- FIND_HUB: Requests the target Hub to return detailed routing information (IP, Port, Node ID) for the K Hubs closest (in logical distance) to a specified target ID. If the optimal K-Bucket does not contain enough entries, results are merged from multiple buckets until K entries are obtained. If fewer than K are known, all available entries are returned.
- FIND_WORKER: Similar in operation to FIND_HUB, but targets a specific Worker node, returning its unique identity information.

### *D. Routing Table Maintenance: K-Bucket Update Mechanism*

ZEROBASE employs a Kademlia-inspired mechanism for maintaining the integrity and freshness of routing tables. Each Hub maintains 160 K-Buckets, each corresponding to a distinct range of logical distances. The system supports three primary modes of updating K-Buckets:

- Active Hub Discovery: A Hub proactively issues FIND_HUB requests to refresh the entries in its K-Buckets.
- Passive Hub Discovery: Upon receiving RPC requests such as FIND_HUB or FIND_WORKER from other Hubs, the recipient updates its corresponding K-Bucket with the sender's information.
- Liveness Check and Failure Detection: Periodic PING messages are sent to peers in K-Buckets to determine their availability. Non-responsive nodes are subsequently purged from the table.

Upon receiving an RPC message from another Hub y, Hub x executes the following update procedure:

- Distance Calculation: Compute the logical distance $d(x, y) = x \oplus y$,, where $x$ and $y$ are the 160-bit Node IDs.
- Bucket Selection: Identify the corresponding K-Bucket based on the value of d(x,y).
- Update Logic
  - If y's IP is already recorded in the selected K-Bucket, move its entry to the tail of the bucket (indicating recent activity).
  - If y's IP is not recorded and the bucket contains fewer than k entries, append (IP, Port, Node ID) of y to the tail.
  - If the bucket is full:
    - ∗ Select the oldest entry z at the head of the bucket and issue a PING.
    - ∗ If z fails to respond, remove z and append y's entry to the tail.
    - ∗ If z responds, retain z by moving it to the tail and discard y.

This update mechanism implements an efficient least recently seen replacement policy, allowing active, long-lived nodes to persist in the routing table, thereby enhancing network stability and reducing maintenance overhead. By prioritizing persistent nodes, the likelihood that entries remain online in subsequent intervals increases, thereby reducing churn-related inefficiencies.

Additionally, this mechanism contributes to resilience against denial-of-service (DoS) attacks, since new entries can only replace existing ones if the latter are verified to be offline, thereby mitigating routing table flooding via malicious node injection.

To prevent K-Bucket staleness, the system periodically selects a random Hub from each K-Bucket to perform a PING check if no updates have occurred within a predefined timeout window. This ensures responsiveness without incurring significant overhead.

### *E. Node Admission and Recursive Discovery Mechanisms*

Maintaining robust network connectivity and efficient task assignment in ZEROBASE relies heavily on dynamic node onboarding and recursive node localization algorithms. This section details the procedures for Worker registration, Hub node admission and departure, and recursive lookup protocols for both node types.

*1) Worker Node Admission Protocol:* Worker nodes are the computational backbone of the execution layer, responsible for off-chain computation and proof generation. When a new Worker node attempts to join the network, the following admission process is initiated:

- Proximity Discovery: A Hub calculates the logical XOR distance between the Worker's Node ID and all known Hubs, identifying the K closest Hubs.
- Registration Broadcast: The initiating Hub sends ADD_WORKER commands to these K Hubs, informing them of the new Worker's existence.
- State Recording: Each recipient Hub updates its local state table, adding the Worker to its distributed index.
- Periodic Re-advertisement: Each of the K Hubs periodically (e.g., hourly) re-broadcasts Worker information to ensure discoverability.
- Expiration Policy: Worker records automatically expire 24 hours after publication to prevent outdated or invalid references.

This protocol enables fault-tolerant, decentralized indexing of Worker nodes while ensuring discoverability and consistency across the network.

*2) Hub Node Admission and Departure Protocol:* To join the network, a new Hub node x must connect to at least one existing bootstrap node y. The admission proceeds as follows:

- Bootstrap Contact: Node x initiates a connection to bootstrap node y and adds y to its initial K-Bucket.
- Initial Lookup: Hub x sends a FIND_HUB request to y, using x's own Node ID as the target.
- Neighbor Discovery: Node y replies with the K Hubs closest to x, which x then adds to its own routing table.
- Iterative Expansion: Hub x recursively issues FIND_HUB requests to the newly discovered Hubs until its routing table reaches sufficient coverage.

This process guarantees rapid integration of new nodes into the network's logical topology.

Hub departures are handled implicitly through passive expiration. Neighboring Hubs detect failed nodes via periodic PINGs and automatically remove them from their K-Buckets. This self-healing property aligns with the "Sleepy Model" of asynchronous consensus [21], allowing nodes to temporarily go offline without compromising the system's security or liveness guarantees.

*3) Recursive Hub Lookup Algorithm:* To locate a Hub with target Node ID y, Hub x initiates a recursive lookup as follows:

- Distance Computation: Calculate the logical distance $d(x, y) = x \oplus y$.
- Bucket Selection: Determine the appropriate K-Bucket by computing $\lfloor \log d \rfloor$ and select up to $\alpha$ candidate Hubs.
- Initial Query: Send FIND_HUB to these $\alpha$ candidates.
- Recursive Expansion:
  - If any queried node has ID y, it responds with its own metadata.
  - Otherwise, each queried Hub measures its own distance to y and returns $\alpha$ closer candidates from its routing table.
  - Hub x repeats this process until it collects K Hubs closest to y, or until no closer candidates can be found.

This recursive approach ensures convergence even if the exact target does not exist in the network, and provides robust pathfinding through the XOR metric. The parameter $\alpha = 3$ is chosen to balance search parallelism with messaging overhead.

## *F. Recursive Worker Lookup Protocol*

To discover a specific Worker node, a recursive lookup is initiated, analogous to Hub resolution. For instance:

- Let $\text{Hub}_1$ have ID = 00000110, and it aims to locate a Worker with Node ID = 00010000.
- The XOR distance is $00000110 \oplus 00010000 = 00010110$, i.e., decimal 22, which falls within the distance range of K-Bucket 5 (i.e., $[24, 25)[2^4, 2^5)[24, 25)$).
- $\text{Hub}_1$ examines its K-Bucket 5:
  - If the Worker is managed by a known Hub (e.g., $\text{Hub}_2$), $\text{Hub}_1$ directly queries $\text{Hub}_5$.
  - If not, $\text{Hub}_1$ selects a candidate $\text{Hub}_3$ from the bucket and instructs it to continue the search.
  - If $\text{Hub}_3$ lacks the desired Worker, it returns a closer $\text{Hub}_4$, and the search continues recursively.

This recursive convergence ensures resilient and efficient Worker localization, even under frequent topology changes or partial routing failures.

## *G. Threat Models and Security Analysis in Distributed Networks*

In the context of the open-access and highly decentralized architecture of the ZEROBASE network, the system is inherently exposed to a wide range of adversarial threats. The permissionless nature of node participation and the dynamic evolution of routing state significantly heighten the attack surface. This section identifies and analyzes four critical attack vectors that pose substantial risks to network stability and integrity: Sybil Attacks, Eclipse Attacks, Churn Attacks, and Adversarial Routing Attacks.

*1) Sybil Attacks:* A Sybil attack refers to a scenario in which a malicious Hub node illegitimately assumes multiple identities within the network, with each instance referred to as a Sybil Hub [22], [23]. The attacker leverages these false identities to gain disproportionate influence over the network. The implications of such an attack include:

- Injection of Fake Hubs: In a permissionless network, attackers may issue repeated join requests to elicit neighbor lists from responding nodes. This enables the attacker to map out large portions of the network topology, facilitating further targeted disruption.
- Routing Table Manipulation: Routing in ZEROBASE relies on timely inter-Hub announcements to maintain accurate routing tables. By impersonating multiple Hubs, a Sybil attacker can infiltrate the routing tables of honest nodes and mislead their routing decisions. In extreme cases, this behavior may escalate into an Eclipse attack.
- Fake Resource Publication: Malicious nodes may register fabricated Worker nodes and advertise them to the network. Upon receiving computational tasks, these fake Workers drop requests silently. A proliferation of such bogus Workers degrades the system's proof-generation efficiency and reduces overall throughput.

*2) Eclipse Attacks:* An Eclipse attack occurs when an attacker successfully populates a victim Hub's K-Buckets with malicious nodes, thereby isolating the victim from the legitimate network [24], [25]. To execute such an attack, the adversary must first establish a sufficient number of Sybil Hubs and then strategically position them into the routing table of the target Hub. The impact of Eclipse attacks includes:

- Data Inaccessibility: The victim is unable to communicate with legitimate nodes, leading to failed or delayed data queries.
- Routing Degradation: Polluted routing tables may cause recursive loops or redirections to invalid nodes, severely impairing query efficiency.
- Network Partitioning: When multiple Hubs are simultaneously compromised, the network may fragment into isolated subnetworks, undermining both decentralization and fault tolerance.

*3) Churn Attacks:* Churn attacks exploit the natural tolerance of distributed networks to dynamic node participation. In such attacks, adversaries intentionally induce high rates of node join and leave events to disrupt the system's operational stability. Though distributed systems are generally resilient to organic churn, artificially elevated churn levels can lead to the following consequences:

- Degraded Network Performance: Constant reconfiguration of routing tables and rebalancing of data increase latency and reduce query success rates.
- Reduced Worker Availability: Frequent node churn causes Workers to disappear or become unreachable, interrupting proof generation and task execution.
- Diminished Hub Reliability: Legitimate Hubs may appear unstable due to frequent route inconsistencies or update failures, diminishing their trust scores and participation.

*4) Adversarial Routing Attacks:* Adversarial routing attacks involve the deliberate manipulation of routing paths by a malicious entity to intercept, misdirect, or suppress network traffic. These attacks target the integrity and availability of routing operations, and their effects include:

- Routing Table Corruption: By injecting falsified routing information or exploiting vulnerabilities in route propagation, adversaries can steer the routing behavior of legitimate nodes.
- Path Hijacking and Traffic Control: The attacker may redirect traffic through compromised nodes, enabling eavesdropping, tampering, or traffic analysis.
- Response Forgery and Data Poisoning: During routing or query resolution, attackers may forge or alter response messages, leading to failed queries or data inconsistencies.
- Communication Disruption: By obstructing connectivity between selected nodes and the broader network, adversaries can induce partial network outages or create partitioned subgraphs.

## IV. Privacy-Preserving Computation

The ZEROBASE architecture realizes a native privacy-preserving execution environment through tightly coordinated layers of TEE + ZKP + Proof Mesh, enabling sensitive off-chain data to be processed under trusted conditions while publishing verifiable results in zero-knowledge. This ensures confidentiality, verifiability, and immutability throughout the computation lifecycle. This privacy-computation framework not only enforces data security but also encapsulates a reusable set of mechanisms and algorithmic constructs.

### *A. Data Confidentiality Mechanisms: End-to-End Encryption & Enclosed Execution*

ZEROBASE's privacy framework safeguards data across its full lifecycle, maintaining encryption especially during untrusted off-chain processing:

- On-chain Input Encryption. Prior to on-chain submission or dispatch to a Worker, all user data is encrypted, denoted as Enc(data), via symmetric encryption combined with session key wrapping, ensuring that only the target Worker's TEE can decrypt and process it. This protocol preserves confidentiality during transit, thwarting man-in-the-middle exposures.
- TEE Enclosed Execution Model. Within the Worker node's local Trusted Execution Environment, the encrypted data is decrypted and processed in isolation [26] . The execution pipeline comprises:
  - Decryption of input data strictly within the TEE boundary;
  - Execution of the private model or strategy logic in a sealed environment;
  - Output generation, re-encrypting results before exposing them;
  - Zero-Knowledge Proof (ZKP) generation, enabling verifiable correctness without revealing inputs or internal states.
- Intermediate states are purged, and no residual traces are left, ensuring both data confidentiality and resistance to tampering.
- Cryptographic Hardware Protections. ZEROBASE leverages hardware-backed isolation technologies such as AMD SEV, Intel TDX, and ARM CCA to enforce encryption of memory and logical I/O barriers [13–15], ensuring data confidentiality even during computation. Future extensibility includes optional Fully Homomorphic Encryption (FHE) to support end-to-end private computation without exposing plaintext at any stage [27].

This architecture's use of encryption, secure execution enclaves, and hardware isolation reflects an evolution from predecessors that decoupled verification and privacy within blockchain systems [28], [29], promoting a decentralized, minimally-trusted privacy infrastructure.

### *B. Verifiability Mechanisms: ZKP Generation from Execution Context*

To support on-chain verification of privately computed results, ZEROBASE introduces a formal modeling framework that translates off-chain computation into zero-knowledge constraints:

- Context-Aware Circuit Templates. ZEROBASE supplies a library of standardized ZKP circuit templates tailored to various private computation tasks:
  - Range Checks (range_check): Assert numerical values fall within prescribed intervals.
  - Model Inference Proofs (model_inference): Verify the correctness of model output given an input.
  - Data Transformation Checks (data_transformation): Confirm validity of state transitions.
- Each template maps a specific task type to an appropriate circuit construction.
- Constraint System Assembly:Upon completion of TEE execution, essential elements—input, state, intermediate variables, and output—are extracted to construct an R1CS or AIR constraint system. Efficient proof systems (e.g., Groth16, PLONK) are then instantiated to generate zk-SNARK or zk-STARK proofs. The public inputs covering the state and context enable on-chain smart contract verification of correctness without exposing sensitive information.

- Algorithmic Optimizations:
  - Witness Compression: Reduces proof size by pruning redundant witness components.
  - Recursive Proofs: Enable merging or nesting of multiple proof stages to streamline on-chain verification.
  - Circuit Parameterization: Tailors templates to specific application requirements, aiding rapid deployment and efficiency.

### *C. Composability Mechanisms: Encapsulation within the Proof Mesh*

The Proof Mesh layer acts not only as a transport intermediary but also as an orchestrator of composable privacy operations, as detailed below:

- Modular Task Abstraction: Each privacy computation is encapsulated as a self-contained module: Task := (InputEnc, StrategyHash, ZKP_Circuit_ID, OutputDigest, Proof $\pi$)
  - InputEnc: Encrypted input payload [30]
  - StrategyHash: Digest of the execution logic
  - ZKP_Circuit_ID: Identifier for the applicable circuit template
  - OutputDigest: Hash summarizing the result
  - Proof $\pi$: ZK proof certifying correct execution
- Inter-Module Proof Reuse:
  - Cross-module chaining: Outputs of one task can serve as inputs for another, supporting sequences of private computations.
  - Off-chain closed-loop flows: Users can trigger private model inferences, followed by further private computation or contract interactions.
  - Cross-DApp privacy interoperability: Different applications can share validated proofs, fostering a decentralized privacy-computation network.
- Versioning & Context Packaging. Each task carries execution metadata—e.g., user ID hash, timestamp, and strategy hash—and forms part of a proof chain with traceability. Subscribers can monitor proof publication events or task status, supporting transparent, composable orchestration. This embraces recent advances in privacy-aware domain-specific languages that embed safety guarantees at the abstraction layer [31].

### *D. Security Foundations & Attack-Resistance Strategies*

ZEROBASE embeds multi-layered security constructs across its execution, proof, and mediation layers, establishing defense-in-depth against adversarial threats, passive snooping, and tampering.

#### *1) TEE Layer Protections:*

- Remote Attestation. Prior to execution, each Worker exchanges hardware-signed attestations with the network, ensuring only genuine TEEs participate [23], [32].
- Encrypted Storage & Isolated I/O. Data remains encrypted within TEE storage, with all in/out channels logically isolated to resist side-channel and memory snooping attacks. Hardware-enforced safeguards (e.g., AMD SEV, Intel TDX) underpin this confidentiality.
- Strategy-Fingerprint Binding. Each execution task references a StrategyHash bound to the proof context to validate that only vetted computation logic was executed, preventing code tampering or malicious substitution.

#### *2) ZKP Layer Protections:*

- Data Confidentiality by Construction. The proof systems completely conceal input data while validating correctness. Optional differential privacy mechanisms can be employed to resist statistical inference from proof leaks [33].
- Anti-Replay Protections. Proof bundles include unique task identifiers and timestamps to prevent replay or context substitution attacks.
- Circuit Auditing & Registry. All ZKP circuit templates are subject to mandatory security audits and registration to ensure correctness, absence of backdoors, and logical consistency.

#### *3) Proof Mesh Layer Protections:*

- Multi-Signature Validation. Every proof package includes cryptographic signatures and metadata annotations—indicating task provenance, execution lineage, and node identity—to enable verifiers to authenticate proof integrity.
- Linked Proof Chain for Traceability. Proofs are chained by including previous output hashes as public inputs to subsequent tasks. This constructs an auditable directed acyclic structure that resists partial tampering or dislocation attacks.
- Zero-Trace Mode. For high-security tasks, metadata can be encrypted or suppressed so that proof publication leaves no observable traces, mitigating metadata analysis and profiling.

Through these layered security mechanisms—spanning hardware-based trust, cryptographic soundness, and orchestrated mediation—ZEROBASE delivers a robust privacy-computation framework. It is intrinsically resistant to manipulation, eavesdropping, and structural vulnerability, empowering trustworthy execution in applications ranging from DeFi to privacy-preserving AI inference.

## V. User Engagement Pathways and Application Architecture

The ZEROBASE network is designed to accommodate multiple role types, each interacting with the trust-minimized execution framework via explicit input/output interfaces. These roles are integrated within the system's Trusted Execution, Proof Generation, and Liquidity modules, forming a modular, reusable, and composable participation paradigm.

### *A. Participation Pathways*

Resource Contribution Pathway. Individuals possessing idle resources—such as bandwidth, compute power, or legacy GPUs—can contribute to the network via resource aggregators using a ZEROBASE client or embedded SDK. Tasks are scheduled across multiple platforms and executed within SEV-SNP protected TEEs. Upon completion, a verifiable Resource Usage Proof is generated and translated into a structured on-chain credential using zero-knowledge circuits. This credential can be staked, lent, or otherwise deployed within financial strategies, forming a nexus between ”off-chain resource $\rightarrow$ verifiable state $\rightarrow$ composable asset.”

Strategy Execution Pathway. Asset managers or quant teams with proprietary trading models can encapsulate their logic in TEE-compatible execution images. After remote attestation registers their environment, the strategy runs off-chain against live market data, consistently outputting zero-knowledge interval proofs denoting risk metrics and return distributions. On-chain verifiers assess this evidence's integrity and interact via a zkStaking module. This pathway achieves the design goal of ”invisible strategy, yet transparent risk.”

Protocol Integration Pathway. External DeFi or Web3 protocols can incorporate ZEROBASE via standardized interfaces—e.g., zkRouter or zkReport. Integrators can query proof-of-state from strategy pools or resource tokens to calibrate liquidation thresholds, determine collateral ratios, or orchestrate cross-protocol logic. These interactions do not require off-chain coordination or trusted intermediaries; the entire exchange is driven by structured zero-knowledge proofs and on-chain function hooks, delivering high reusability and low trust assumptions.

### *B. Illustrative Use Cases*

- Strategy Execution Example. A hedge fund packages a delta-neutral trading algorithm as a SEV VM executable. Periodically, the system issues proofs of leverage bounds and drawdown risk on zkStaking, visible to users without disclosing strategy details. Investors can choose participation based on zero-knowledge attestations, enabling confidential tokenized fundraising.
- Resource Utilization Example. An individual connects spare compute capacity to a ZEROBASE aggregator for parallel task execution. The system computes in TEE, emits zero-knowledge earnings proofs, and mints on-chain resource tokens. These tokens can be staked, transferred, or nested into other protocols, effectively financializing hitherto idle resources.
- Cross-Protocol Composition Example. A DeFi lending platform employs zkRouter to query a user's solvency position within a strategy pool. On-chain verification of proof updates loan-to-value ratios without trusting off-chain oracles or inter-protocol messaging.
- Infrastructure Integration Example. A Web2 data provider serializes its external API (e.g., credit ratings or voting results) into a SEV-enclosed execution node. Upon execution and zero-knowledge attestation, a credible on-chain data summary is generated for smart contracts—bridging Web2 outputs into Web3 trust environments.

By standardizing interface contracts, proof formats, and participant abstractions, ZEROBASE establishes a composable execution medium that supports multiple engagement modalities. Participants can fluidly shift roles—for instance, resource providers evolving into strategy node operators, or integrators specifying new proof-based state requirements. The result is an emergent participant topology that is evolutionary and self-organizing.

## VI. Conclusion

In a landscape where transparency and privacy conflict, where off-chain execution struggles to align with on-chain verification, and where latent resources remain untapped, ZEROBASE offers a structural remedy: embedding trust into the architecture and restoring validation authority to users and protocols.

Leveraging trusted execution, zero-knowledge proofs, and composable liquidity structures, ZEROBASE delivers a system that requires no human interpretation. Strategies remain undisclosed but risks can be audited; resources need neither custodial intermediaries nor opaque contracts; capital need not rely on platform guarantees, but can be orchestrated via structural rules. This is the essence of structural transparency and trust-minimized collaboration.

ZEROBASE is not designed to supplant existing systems or to predefine a fixed end-product class. Instead, it is built as a foundational substrate—a trust-anchored core for developers to reuse, ecosystems to synergize, and execution agents to run.We invite strategy providers to use it to build unique yet verifiable financial products; resource owners to expose real-world compute capacity on-chain; and developers to treat it as a compositional engine, embedding new modules, expanding new use cases, and inventing new structures. We have unveiled the initial version of this architecture—and it will, in time, belong to everyone.

## References

[1] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Scalable zero knowledge via cycles of elliptic curves,” *Algorithmica*, vol. 79, no. 4, pp. 1102–1160, 2017.

[2] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous distributed e-cash from bitcoin,” in *2013 IEEE symposium on security and privacy*. IEEE, 2013, pp. 397–411.

[3] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth, “Succinct non-interactive arguments via linear interactive proofs,” *Journal of Cryptology*, vol. 35, no. 3, p. 15, 2022.

[4] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, “Spongent: A lightweight hash function,” in *International workshop on cryptographic hardware and embedded systems*. Springer, 2011, pp. 312–325.

[5] Intel Corporation, “Intel® Trust Domain Extensions (TDX),” https://www.intel.com, 2025, [Online].

[6] AMD, “AMD SEV-SNP: Secure Encrypted Virtualization – Secure Nested Paging,” https://www.amd.com, 2025, [Online].

[7] Arm Ltd., “Arm Confidential Compute Architecture (CCA),” https://www.arm.com, 2025, [Online].

[8] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.

[9] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, “Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge,” *Cryptology ePrint Archive*, 2019.

[10] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Scalable, transparent, and post-quantum secure computational integrity,” *Cryptology ePrint Archive*, 2018.

[11] M. Fischlin, “Communication-efficient non-interactive proofs of knowledge with online extractors,” in *Annual International Cryptology Conference*. Springer, 2005, pp. 152–168.

[12] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short proofs for confidential transactions and more,” in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 315–334.

[13] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2001, vol. 2.

[14] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, “Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 71–101.

[15] S. Gorbunov, V. Vaikuntanathan, and H. Wee, “Attribute-based encryption for circuits,” *Journal of the ACM (JACM)*, vol. 62, no. 6, pp. 1–33, 2015.

[16] A. Kothapalli, S. Setty, and I. Tzialla, “Nova: Recursive zero-knowledge arguments from folding schemes,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.

[17] F. Castillo, J. Heiss, S. Werner, and S. Tai, “Trusted compute units: A framework for chained verifiable computations,” *arXiv preprint arXiv:2504.15717*, 2025.

[18] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town crier: An authenticated data feed for smart contracts,” in *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, 2016, pp. 270–282.

[19] M. K. Reiter and A. D. Rubin, “Crowds: anonymity for web transactions,” *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[20] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 53–65.

[21] R. Pass and E. Shi, “The sleepy model of consensus,” in *International conference on the theory and application of cryptology and information security*. Springer, 2017, pp. 380–409.

[22] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, “Sok: Research perspectives and challenges for bitcoin and cryptocurrencies,” in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 104–121.

[23] M. Castro, B. Liskov *et al.*, “Practical byzantine fault tolerance,” in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[24] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[25] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds,” in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 199–212.

[26] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, “Iron: functional encryption using intel sgx,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 765–782.

[27] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[28] G. Zyskind, O. Nathan *et al.*, “Decentralizing privacy: Using blockchain to protect personal data,” in *2015 IEEE security and privacy workshops*. IEEE, 2015, pp. 180–184.

[29] C. Lin, D. He, X. Huang, M. K. Khan, and K.-K. R. Choo, “Dcap: A secure and efficient decentralized conditional anonymous payment system based on blockchain,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2440–2452, 2020.

[30] N. Döttling, S. Garg, M. Hajiabadi, D. Masny, and D. Wichs, “Two-round oblivious transfer from cdh or lpn,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 768–797.

[31] E. Lobo-Vesga, A. Russo, and M. Gaboardi, “A programming language for data privacy with accuracy estimations,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 43, no. 2, pp. 1–42, 2021.

[32] O. Goldreich, “Secure multi-party computation,” *Manuscript. Preliminary version*, vol. 78, no. 110, pp. 1–108, 1998.

[33] C. Dwork, A. Roth *et al.*, “The algorithmic foundations of differential privacy,” *Foundations and trends® in theoretical computer science*, vol. 9, no. 3–4, pp. 211–407, 2014.