# ZEROBASE: 신뢰 컴퓨팅과 검증 가능한 프라이버시의 교차점에서

## I. 소개

최근 몇 년간 데이터 주권, 프라이버시 보호 및 시스템 투명성이 세계 정책과 산업 담론의 핵심 주제가 되면서, "통제 가능하지만 침해적이지 않은(controllable yet non-intrusive)" 인프라를 가능하게 하는 기술적 기반에 대한 수요가 디지털 금융과 공유경제 아키텍처의 중요한 과제가 되었다. 기존 패러다임에서 온체인(on-chain) 실행은 검증 가능성을 제공하지만 성능 병목과 제한된 프라이버시 보장으로 인해 고빈도 거래, 복잡한 전략, 민감한 데이터 처리에 적합하지 않다. 반면 오프체인(off-chain) 시스템은 효율성이 뛰어나지만 검증 가능성과 무신뢰성(trustlessness)을 희생하여 사용자에게 신뢰 비용과 구조적 리스크를 증가시킨다.

ZEROBASE는 오프체인의 신뢰 실행 환경(TEEs)과 온체인의 영지식 증명(ZKPs)을 결합한 하이브리드 패러다임을 제안합니다.이를 통해 중앙화된 중개자 없이 실행 단계의 프라이버시와 결과의 검증 가능성을 동시에 달성합니다. [1]–[3]. 이 아키텍처는 세 가지 핵심 기술(영지식 증명, 하드웨어 기반 회로 수준 암호화(예: AMD SEV-SNP), 프로그래밍가능한 유동성 엔진)을 활용하여 고가용성, 모듈성 및 지속 가능한 프라이버시 보호 실행 네트워크를 구축한다. 이는 개발자, 자산 제공자, 최종 사용자 모두에게 통합적이고 접근 가능하며 신뢰할 수 있는 인프라를 제공한다.

따라서 ZEROBASE는 독립적인 프로토콜로 설계된 것이 아니라, 복잡한 전략 배포, 자원 자산화 및 신뢰 가능한 오프체인 계산을 지원하는 일반화된 실행 플랫폼으로, 다양한 응용 시나리오의 시스템적 기반이자 보편적 신뢰 계층 역할을 한다.

### *A.* 비전과 미션: 금융 실행에서의 신뢰 구조 재편

ZEROBASE의 미션은 온체인 생태계가 오랫동안 직면해 온 두 가지 문제를 해결하는 것이다: (1) 사용자가 복잡한 전략을 검증할 수 없어 만연한 "블랙박스 실행" 현상과, (2) 프라이버시 보호와 시스템 투명성 간의 본질적인 긴장 관계로 인해 사용자와 개발자가 효율성과 신뢰 중 하나를 포기해야 하는 문제이다.

우리는 진정으로 지속 가능한 실행 네트워크가 전략가들에게 기밀 공간을 보장하고, 자본 참여자들에게 명확한 위험 경계를 정의하며, 개발자들의 통합 및 조합 비용을 최소화하는 세 가지 필수 조건을 충족해야 한다고 주장한다.

이를 위해 ZEROBASE는 신뢰 가정이 아닌 수학적 보장, 아키텍처적 격리 및 표준화된 인터페이스를 기반으로 하는 구조적 투명성 철학을 확립하여 신뢰 가능한 다자 협력 프레임워크를 구현하는 것을 목표로 한다.

우리의 비전은 네 가지 핵심 기둥으로 요약할 수 있다:

- 검증 가능한 전략 실행: 헤지펀드, 알고리즘 트레이딩과 같은 복잡한 전략을 오프체인에서 프라이빗하게 실행하면서 레버리지 비율과 VaR(위험가치) 구간 같은 주요 위험 지표를 영지식 범위 증명을 통해 노출하여 "검증 가능한 프라이버시"를 달성한다.
- 자원의 자산화: 신뢰 실행 환경과 ZK 증명을 통해 온체인 인증을 생성하여 대역폭, 저장 공간, GPU 사이클과 같은 공유 가능한 자원을 DeFi 메커니즘(담보화, 대출, 원자적 스왑)에 참여시켜 온체인 유동성 잠재력을 발휘할 수 있게 한다.
- 구성 가능한 사용자 경험: 모든 핵심 모듈(회로, zkStaking, ZK 브라우저 등)이 표준화된 중첩 호출 인터페이스를 채택하여 개발자가 기능을 쉽게 구성하고 재사용하여 복합 애플리케이션을 최소한의 마찰로 신속하게 구축할 수 있게 한다.
- 마찰 없는 생태계 협력: 명확히 정의된 인터페이스 명세와 안정적인 툴링을 통해 추가적인 합의 계층 없이도 ZK 애플리케이션들이 상호 운용 가능하도록 지원하여 고립된 프로토콜에서 구조적으로 협력하는 생태계로 전환한다.

이러한 원칙을 기반으로 ZEROBASE는 신뢰를 양보하거나 프라이버시를 희생하지 않는, 구조적으로 견고하고 명확히 경계 지어지며 협력 효율적인 탈중앙화된 실행 생태계를 구축하는 데 전념하고 있다.

### *B.* 기술 철학과 설계 원칙: 수학과 하드웨어 간의 신뢰 경계 재구성

ZEROBASE는 기존 패러다임의 단순한 점진적 업그레이드가 아니라, 신뢰 모델 자체를 근본적으로 재구성하는 것을 목표로 한다. 이는 수학적 형식주의, 암호학적 메커니즘 및 신뢰 가능한 하드웨어를 통합한 체계적 논리로서, "신뢰 없음(trustlessness)"이 기본 전제가 되는 운영 프레임워크이다.

이러한 구조적으로 투명한 실행 네트워크는 세 가지 핵심 설계 원칙을 바탕으로 구축된다:

- 최소 공개. 정보는 더 이상 "원시 데이터"가 아니라 암호학적 "증명" 형태로 공개된다. ZEROBASE는 전략 위험, 수익 성과, 유동성 상태 등 시스템 상태를 범위 증명이나 구조화된 영지식 증명(ZKP)을 통해 표현하여, 사용자가 기초 데이터를 접근하지 않고도 신뢰를 형성할 수 있게 한다. 예를 들어 "레버리지가 2배를 초과하지 않는다" 또는 "일일 수익률이 0.1%에서 0.3% 사이"와 같은 진술이 평문 공개가 아닌 검증 가능한 출력으로 제공된다.
- 신뢰 최소화. ZEROBASE의 오프체인 실행은 AMD SEV-SNP의 기밀 컴퓨팅 환경에서 수행되며, 원격 증명을 통해 실행 환경의 무결성을 검증한다. 온체인 검증은 공식적으로 구성된 ZK 회로를 통해 이루어진다. 이를 통해 시스템은 단일 신뢰 주체에 대한 의존성을 제거하고, 암호학적 프리미티브와 구조적 구획화를 통해 논리적 정확성을 보장하여, 하드웨어 신뢰 근원부터 거버넌스 메커니즘까지 전방위적인 구조적 신뢰 없음을 달성한다.
- 구성 가능한 증명. ZEROBASE 아키텍처에서 증명은 모듈 간 협력을 위한 "중간 언어" 역할을 합니다. 각 전략 모듈은 DeFi 프로토콜, 청산 엔진 또는 담보 대출 구조에서 직접 사용할 수 있는 통합 상태 요약(예: 위험 구간, 성과 지표, 지급여력 지표 [4])을 출력합니다. 이러한 추상화, 즉 "인터페이스로서의 증명"은 검증 비용을 획기적으로 줄이고, 모듈 경계를 명확하게 정의하며, 체인 및 실행 영역 전반에 걸쳐 혁신을 촉진합니다.

따라서 ZEROBASE는 새로운 윤리적 프레임워크를 수용한다: 프라이버시는 사용 가능하며, 신뢰는 계산 가능하고, 구조는 구성 가능하다. 진정으로 신뢰할 수 있는 시스템은 별도의 정당화를 필요로 하지 않으며, 자체 아키텍처를 통해 그 신뢰성을 입증한다.

### C. 시장 환경과 핵심 과제: 파편화된 프라이버시 솔루션과 시스템적 조정 부재

암호학 기반 금융, 분산 컴퓨팅 및 모듈형 프로토콜이 급속히 발전하고 있지만, 현재 가장 큰 과제는 기술적 수단의 부족이 아니라 이를 통합적으로 조율하는 아키텍처의 부재이다. 사용자는 프라이버시를 요구하고, 자본은 검증 가능성을 원하며, 시스템은 오프체인 실행에 의존하고, 자원은 온체인 수익화를 추구한다. 하지만 기존 아키텍처는 일반적으로 이러한 요구 중 한 가지 정도만 충족할 뿐, 다른 요구들을 희생시켜야 하는 경우가 많다.

이는 다음과 같은 여러 구조적 딜레마를 초래하였다:

- 사용자는 프라이버시를 원하지만 프로토콜 설계상 데이터는 투명하게 공개되어야 한다.
- 전략은 알파(alpha)를 유지하기 위해 비밀성을 필요로 하지만, 참여자들은 명확한 리스크 지표 공개를 요구한다.
- 시스템 성능은 오프체인 실행에 의존하지만, 검증 로직은 온체인 인프라에 제약된다.
- 자원은 공유 가능하지만 조합이 어려워, "오프체인 가치"를 토큰화하기 어렵다.

이러한 긴장 관계는 구조적 조정의 부재가 전체 시스템의 병목이 되었음을 보여준다. 중앙화 플랫폼의 불투명한 운영에서부터 온체인 데이터의 무차별적 공개, 파편화된 마이닝 수익 구조에서 일관되지 않은 프라이버시 표준까지, 암호화폐 생태계 전반에는 실행, 검증 및 인센티브 메커니즘을 통합하는 중심 허브가 존재하지 않는다.

ZEROBASE가 제안하는 신뢰 최소화 실행 네트워크(Trust-Minimized Execution Network, TMEN)는 특정 애플리케이션에 국한된 것이 아니라, 온체인에서 오프체인 실행을 검증 가능하게 만들고 ZKP 및 TEE 시스템을 연결하며 자원과 전략을 조합 가능한 구조로 통합하기 위해 설계된 기본 역량 계층이다. TMEN의 핵심 가치는 애플리케이션 범위를 단순히 확장하는 데 있는 것이 아니라, 다음과 같은 가치를 제공하는 데 있다:

- 오프체인과 온체인 환경 간에 재사용 가능한 브리지 제공
- 다양한 증명 시스템과 하드웨어 모델을 수용하는 인터페이스 아키텍처
- 자산, 전략, 자원을 표준화된 입출력(I/O) 형식으로 추상화하는 모듈형 프레임워크
- 실행, 검증 및 인센티브 흐름을 동기화하는 통합 플랫폼

이러한 인터페이스 계층 위에서 개발자는 자유롭게 복잡한 상호작용을 구축할 수 있으며, 사용자는 신뢰를 바탕으로 참여할 수 있고, 자원 제공자는 참여를 통해 수익을 얻을 수 있다. 시스템 전체는 구조적 정렬을 통해 진화적 탄력성을 확보하게 된다.

## II. 시스템 아키텍처

ZEROBASE는 체인 밖 연산을 위한 신뢰 최소화 및 프라이버시 검증 가능한 실행 네트워크를 구축하고자 하며, 이는 온체인과 오프체인 시스템 간의 신뢰 격차 문제를 해결하고 신뢰의 구조적 연결 고리 역할을 합니다. 이 네트워크는 세 가지 핵심 기술 구성 요소의 조정된 작동을 기반으로 구축됩니다:

### A. 3계층 아키텍처 설계: *TEE, ZKP,* 그리고 *Proof Mesh*

ZEROBASE 실행 네트워크는 세 개의 아키텍처 계층으로 구성되며, 각 계층은 고유한 기능적 역할을 수행하고 표준화된 프로토콜을 통해 상호 연결됩니다. 이러한 계층 구조는 시스템의 모듈성, 유지 관리성, 확장성을 향상시킵니다.

*1*) 신뢰 실행 계층 (*TEE* 계층): 실행 네트워크의 핵심에는 TEE 계층이 있으며, 이는 전략 실행, 자원 스케줄링, 데이터 처리 등 모든 민감한 오프체인 연산을 담당합니다. 각 전략 엔진, 트랜잭션 노드 또는 자원 제공자는 AMD SEV-SNP, Intel TDX, ARM CCA [5]–[7]와 같은 하드웨어 격리 신뢰 실행 환경 내에서 작동합니다. TEE 기술은 메모리 암호화와 CPU 수준 접근 제어를 통해 실행 환경을 격리하고 외부 감시 또는 조작으로부터 보호합니다.

악의적인 노드가 신뢰된 컴퓨팅 에이전트를 사칭하지 못하도록 하기 위해 시스템은 원격 인증(Remote Attestation) 메커니즘을 적용합니다 [5]–[7]. 각 노드는 등록 또는 실행 시 하드웨어 서명된 인증서를 제출해야 하며, 이를 통해 실행 환경의 진정성, 무결성, 일관성을 증명합니다. 이러한 인증은 ZEROBASE 네트워크 전반에 걸쳐 개방성과 보안을 균형 있게 유지하는 탈중앙화된 평판 모델의 기반을 형성합니다.

TEE 내에서 실행되는 로직은 엄격한 감사를 받고 캡슐화되며, 외부에서는 "블랙박스"로 작동합니다. 이는 온체인 또는 오프체인 소스로부터 암호화된 입력을 수신하고 내부에서 프라이버시 보호 연산을 수행한 뒤, 최종 출력과 온체인 검증을 위한 증명 요약을 생성합니다. 이 과정 전반에서 사용자 데이터, 전략 로직, 자원 접근 패턴은 암호화된 상태로 유지되어 완전한 실행 프라이버시를 보장합니다.

*2*) *Zero-Knowledge Proof Layer* (*ZKP Layer*): TEE에서 연산이 수행된 후, 결과는 온체인 검증자에게 검증 가능한 형태로 전달되어야 합니다. 이 작업은 ZKP 계층이 담당하며, 핵심 실행 의미를 암호학적 제로 지식 증명으로 변환합니다. 시스템의 "신뢰 확장 메커니즘" 역할을 수행하는 ZKP 계층은 각 노드가 표준화된 회로 템플릿을 사용하여 입력, 상태 전이, 출력을 검증 가능한 zk-SNARK 또는 zk-STARK 증명으로 변환할 수 있게 합니다 [1], [8]–[11].

일반화 가능성과 개발자 효율성을 촉진하기 위해, ZEROBASE는 여러 일반적으로 사용되는 증명 유형을 포함하는 표준화된 회로 라이브러리를 제공합니다 [12], [13], 다음과 같은 항목들을 포함합니다:

- 범위 증명(Range Proofs): 숫자 값이 지정된 구간 내에 있는지 검증 [12];
- 유효성 증명(Validity Proofs): 연산이 유효한 입력 및 상태에 기반하여 수행되었는지 확인;
- 상태 전이 증명(State Transition Proofs): 상태 $s_0$에서 $s_1$로의 변경이 사전 정의된 논리 제약 조건을 충족하는지 검증;
- 머클 경로 증명(Merkle Path Proofs): 특정 요소가 오프체인 머클 기반 상태 트리에 포함되어 있는지 증명.

현재 구현에는 PLONK, Groth16, STARKs 등의 주요 증명 시스템이 통합되어 있으며, 낮은 검증 비용과 간결한 증명 크기를 최적화합니다. 장기적으로는 양자 위협에 대한 복원력을 확보하기 위해, LWE 및 SIS 기반 구성 [14], [15]을 포

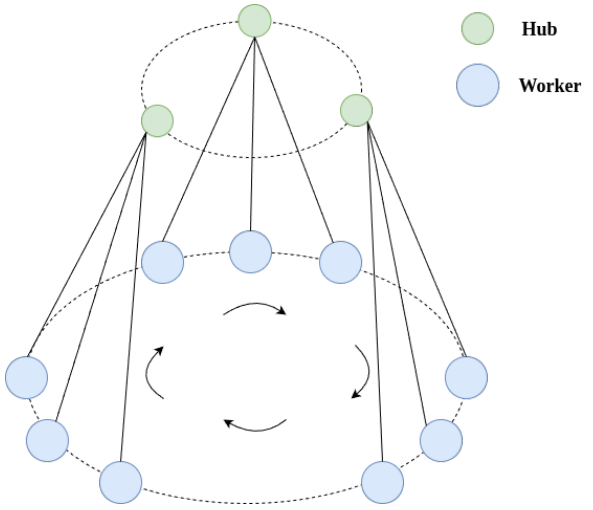


그림 1: 네트워크 아키텍처

함한 격자 기반 제로 지식 시스템의 도입도 구상하고 있으며, 암호학적 강도와 프로그래머블 구조적 유연성을 결합하고자 합니다.

*3*) 증명 중재 계층 (*Proof Mesh* 계층): Proof Mesh는 TEE 기반 오프체인 실행과 온체인 통합 사이를 연결하는 구조적 어댑터 역할을 하며, ZEROBASE의 가장 혁신적인 아키텍처 구성 요소 중 하나입니다. 이는 계층 간의 고효율, 고범용성, 고조합성 조정을 가능하게 하는 여러 핵심 역할을 수행합니다.

첫째, Proof Mesh는 TEE 계층에서 생성된 모든 ZKP를 표준화합니다. 여기에는 형식 지정(예: 통일된 필드 명칭 및 증명 구조), 의미 주석(예: 증명 유형, 목적, 타임스탬프), 프로토콜 버전 제어가 포함됩니다. 이러한 표준 덕분에 온체인 스마트 계약은 구현별 로직을 해석할 필요 없이 균일한 API를 통해 오프체인 증명과 상호작용할 수 있어 통합 비용이 크게 절감됩니다.

둘째, Proof Mesh는 다음과 같은 고급 증명 수준 기능을 지원합니다:

- 증명 집계(Proof Aggregation): 여러 개의 독립적인 증명을 하나의 증명으로 압축하여 온체인 검증 비용을 줄임 [9], [12], [16];
- 재귀 증명(Recursive Proofs): 하나의 증명을 다른 증명 안에 삽입하여 복잡한 오프체인 프로세스를 계층적으로 모델링 가능하게 함 [9], [10], [30];
- 교차 프로토콜 호출(Cross-Protocol Invocation): 서로 다른 모듈 및 DApp 간에 증명을 재사용 및 호출 가능하게 하여 "서비스로서의 증명" 데이터 계층을 형성 [17]–[19].

추가로, Proof Mesh는 버전 관리 및 이벤트 구독 메커니즘을 도입하여 "오프체인 연산 로그" 및 "온체인 이벤트 트리거"를 지원합니다. 예를 들어, DApp은 특정 유형의 실행 증명 게시를 구독하고 새로 생성된 증명에 반응하여 자동 로직을 실행할 수 있습니다.

이러한 세 계층 아키텍처—안전한 실행(TEE), 검증 가능한 변환(ZKP), 상호 운용 중재(Proof Mesh)의 조정된 작동을 통해 ZEROBASE는 완전한 신뢰 보존 실행 루프를 수립합니다. 각 오프체인 연산은 표준화되고 간결하며 검증 가능한 "증명 패키지"로 캡슐화되어 온체인 상에서 전파 및 호출 준비가 완료됩니다.

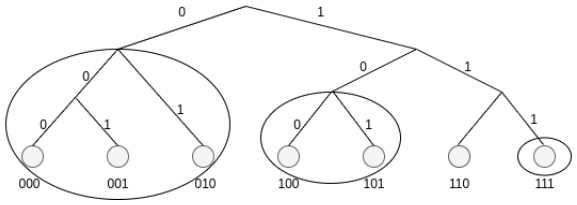


그림 2: 노드 거리

이 설계는 "결과가 인터페이스가 되고, 증명이 구조를 정의한다"는 새로운 실행 패러다임을 구현하며, 오프체인 실행을 블랙박스에서 벗어나 온체인 경제 시스템 내에서 조합 가능하고 호출 가능하며 구조적으로 신뢰 최소화된 일급 구성 요소로 탈바꿈시킵니다.

## III. 네트워크 토폴로지 및 노드 관리

확장성과 적대적 행위에 대한 복원력을 동시에 달성하기 위해, ZEROBASE 네트워크는 계층적 라우팅 아키텍처를 채택하며, 이에 따라 워커(Worker) 노드들은 허브(Hub) 노드에 의해 관리되는 서브넷으로 구성됩니다. 각 허브는 일부 워커 노드를 담당하며, 라우팅 구조는 Kademlia 유사 토폴로지에 기반합니다. 이때 노드 식별자(Node ID)는 노드의 IP 주소와 포트 번호를 SHA-1(160비트) 해시로 변환하여 생성되며, 이는 논리적 거리 계산 및 라우팅 테이블 구성의 기준으로도 사용됩니다 [20].

### *A.* 노드 식별 및 논리적 거리 메트릭

네트워크 내의 모든 노드—허브든 워커든—시스템에 참여할 때 자신의 IP 주소와 포트 번호를 해시하여 고유한 160비트 노드 ID를 생성합니다:

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

두 노드 간의 논리적 거리는 이들의 Node ID를 비트 단위 XOR 연산하여 정의됩니다. 예를 들어, 두 노드의 ID가 각각 $X = 1101, Y = 1000$이라면 논리적 거리는 XOR 결과에 의해 결정됩니다. 이 XOR 기반 메트릭은 물리적 거리와는 무관하며, 오직 논리적 라우팅 경로를 구성하는 데에만 사용됩니다. 워커 노드가 네트워크에 참여할 때, 기존 허브들과의 논리적 거리를 계산한 후 가장 가까운 허브에 연결되며, 해당 허브가 이후 이 워커 노드의 상태를 유지 및 관리하는 책임을 집니다.

### *B.* 노드 할당 및 거리 기반 계층화

세 개의 허브 노드—$\text{Hub}_0$, $\text{Hub}_1$, $\text{Hub}_2$—가 존재하는 네트워크를 가정해 봅니다. 이들의 Node ID는 각각 $X$, $Y$, $Z$입니다. 새로운 워커(Worker) 노드가 ID $T$를 가지고 네트워크에 참여합니다. 이 워커는 각 허브에 대한 XOR 기반 논리적 거리를 계산합니다:

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

가정하자

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

그러면

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

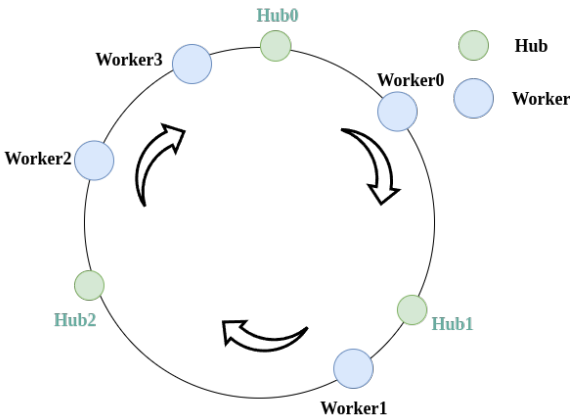


그림 3: 노드 할당

노드 ID는 160비트 이진 문자열이므로, 각 허브는 다른 허브들의 IP, 포트, ID와 같은 정보를 저장하기 위해 160개의 리스트(K-버킷)를 유지해야 합니다.

다음과 같은 시나리오를 고려해 봅니다: Hub1의 ID가 101이라고 가정합니다. 만약 Hub2의 ID의 앞자리 숫자가 Hub1과 같고 마지막 자리만 다르다면, 예를 들어 ID가 100이라면, 두 Hub ID의 XOR 값은 $101 \oplus 100 = 001$이며, 즉 거리 값은 1입니다. 이 경우, Hub1은 Hub2를 "K-Bucket 1"에 저장합니다.

또 다른 예로, 허브들의 ID 중 앞자리는 모두 동일하지만 끝에서 두 번째 자리가 다른 경우를 고려해 봅니다. 예를 들어 ID가 각각 111과 110인 두 허브가 있다고 가정합니다. 이 두 허브와 Hub1(ID: 101) 간의 XOR 값은 각각 $101 \oplus 111 = 010$ 및 $101 \oplus 110 = 011$이며, 거리는 각각 2와 3입니다. 이러한 경우, Hub1은 이 두 허브를 "K-Bucket 2"에 저장하게 됩니다.

허브들의 ID 중에서 앞자리는 모두 동일하지만 끝에서 세 번째 자리가 다른 경우를 고려해 봅니다. 이러한 허브는 총 네 개뿐이며, 그들의 ID는 다음과 같습니다: 000, 001, 010, 011. 이 허브들과 Hub1의 ID 간의 XOR 값은 각각 다음과 같습니다: $101 \oplus 000 = 101$, $101 \oplus 001 = 100$, $101 \oplus 010 = 111$, $101 \oplus 011 = 110$. 즉, 거리 범위는 4, 5, 6, 7입니다. Hub1은 이 네 개의 허브를 "K-Bucket 3"에 저장합니다. 아래 그림에서와 같이, ID가 101인 허브는 ID가 $0 \times \times$인 허브를 "K-Bucket 3"에, ID가 $11\times$인 허브를 "K-Bucket 2"에, 그리고 ID가 101인 허브를 "K-Bucket 1"에 각각 추가합니다.

허브의 ID 앞자리는 동일하지만 끝에서 $i$번째 자리가 다른 경우, 가능한 허브 수는 $2^{i-1}$개에 불과하며, 이들의 Hub1과의 논리적 거리는 $[2^{i-1}, 2^i - 1]$ 범위 내에 존재합니다. Hub1은 이러한 허브들을 "K-Bucket i"에 저장하게 됩니다.

TABLE I: 버킷

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

$i$가 증가함에 따라, 각 K-Bucket $i$에는 $2^{i-1}$개의 허브 정보가 포함될 수 있으며, 이는 허브에 큰 저장 오버헤드를 유발할 수 있습니다. 따라서 각 K-Bucket은 최대 $K$개의 허브 정보만을 기록하도록 규정되어 있으며, 여기서 $K$는 조정 가능한 상수 파라미터입니다. ID 길이가 160비트이므로, 각 허브는 최대 $160 \times K$개의(보통 $K$는 20으로 설정됨) 다른 허브 정보를 유지하게 됩니다. 또한, 각 K-Bucket에 저장된 허브의 위치는 마지막으로 관측된 시점을 기준으로 정렬되며, 가장 오래된 허브가 리스트 앞부분에, 가장 최근에 접근된 허브가 리스트 뒷부분에 위치합니다.

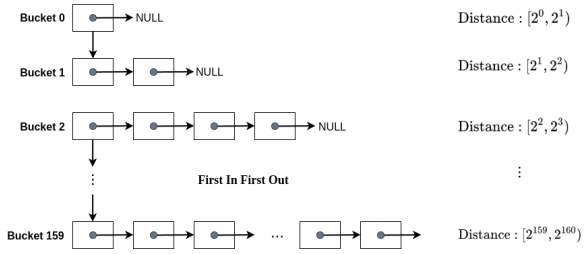


그림 4: K-버킷

### C. 허브 노드를 위한 명령어 집합

허브 노드 간의 통신은 간결한 프로토콜 수준의 명령어 집합을 통해 이루어집니다. 주요 명령어 집합은 다음과 같습니다:

- PING: 대상 허브 노드의 생존 여부를 확인하는 데 사용됩니다. 이는 K-Bucket의 건강 상태를 유지하는 데 필수적입니다.
- ADD_WORKER: 수신 측 허브에게 특정 워커 노드의 상태 및 라우팅 정보를 저장하라고 지시합니다.
- FIND_HUB: 대상 허브에게 지정된 타깃 ID와 논리적 거리상 가장 가까운 $K$개의 허브에 대한 상세 라우팅 정보(IP, Port, Node ID)를 요청합니다. 최적의 K-Bucket에 충분한 엔트리가 없는 경우, 여러 버킷에서 결과를 병합하여 $K$개의 결과가 확보될 때까지 반환됩니다. 만약 알려진 항목이 $K$개 미만인 경우, 가능한 모든 엔트리가 반환됩니다.
- FIND_WORKER: 동작 방식은 FIND_HUB와 유사하지만, 특정 워커 노드를 대상으로 하며 해당 워커의 고유 식별 정보를 반환합니다.

### D. 라우팅 테이블 유지관리: $K$-버킷 업데이트 메커니즘

ZEROBASE는 라우팅 테이블의 무결성과 최신성을 유지하기 위해 Kademlia 기반의 메커니즘을 사용합니다. 각 허브는 논리적 거리 범위별로 160개의 K-Bucket을 유지하며, K-Bucket 업데이트는 다음 세 가지 주요 방식으로 수행됩니다:

- 능동적 허브 탐색(Active Hub Discovery): 허브가 주기적으로 FIND_HUB 요청을 발행하여 K-Bucket을 갱신합니다.
- 수동적 허브 탐색(Passive Hub Discovery): 다른 허브로부터 FIND_HUB 또는 FIND_WORKER RPC 요청을 수신할 경우, 수신자는 발신자의 정보를 해당 K-Bucket에 업데이트합니다.
- 생존 확인 및 장애 탐지(Liveness Check and Failure Detection): K-Bucket 내 피어에게 주기적으로 PING 메시지를 보내 가용성을 확인하고, 응답하지 않는 노드는 테이블에서 제거합니다.

다른 허브 y로부터 RPC 메시지를 수신한 허브 x는 다음과 같은 절차로 업데이트를 수행합니다:

- 거리 계산(Distance Calculation): $x$와 $y$가 160비트 Node ID일 때, 논리적 거리 $d(x, y) = x \oplus y$를 계산합니다.
- 버킷 선택(Bucket Selection): 계산된 $d(x, y)$ 값에 따라 해당하는 K-Bucket을 식별합니다.
- 업데이트 로직(Update Logic)
  - 선택된 K-Bucket에 y의 IP가 이미 존재하면, 해당 엔트리를 버킷의 끝으로 이동시켜 최근 활동을 표시합니다.
  - y의 IP가 존재하지 않고 버킷이 아직 K개 미만이면, y의 (IP, Port, Node ID)를 버킷 끝에 추가합니다.
  - 버킷이 가득 찬 경우:
    * 버킷 맨 앞의 가장 오래된 항목 z를 선택하여 PING을 발송합니다.
    * z가 응답하지 않으면 z를 제거하고 y의 정보를 끝에 추가합니다.
    * z가 응답하면 z를 끝으로 이동시키고 y는 폐기합니다.

이 업데이트 메커니즘은 최근에 관측되지 않은 노드를 우선 교체하는 정책을 구현하며, 활동 중인 장기 노드가 라우팅 테이블에 지속적으로 유지되도록 하여 네트워크 안정성과 유지 비용을 개선합니다. 이러한 방식은 지속 가능한 노드를 우선시함으로써 이후 기간 동안 온라인 상태 유지 확률을 높이고, 네트워크 churn에 따른 비효율을 줄입니다.

또한 이 메커니즘은 거부 서비스 공격(DoS)에 대한 복원력을 제공하며, 새로운 항목은 기존 항목이 오프라인 상태임이 확인될 경우에만 교체될 수 있어 악의적 노드에 의한 라우팅 테이블 홍수 공격을 완화할 수 있습니다.

K-Bucket의 구버전화(staleness)를 방지하기 위해, 시스템은 각 K-Bucket에서 주기적으로 무작위 허브를 선택하여 PING 체크를 수행하며, 이때 사전 정의된 타임아웃 내 업데이트가 없는 경우에만 수행됩니다. 이를 통해 과도한 오버헤드 없이 응답성을 유지할 수 있습니다.

### *E.* 노드 가입 및 재귀적 탐색 메커니즘

ZEROBASE에서 견고한 네트워크 연결성과 효율적인 작업 할당을 유지하려면, 동적 노드 온보딩과 재귀적 노드 탐색 알고리즘이 핵심적입니다. 이 섹션은 워커 등록, 허브 노드의 참여 및 탈퇴, 그리고 두 노드 유형에 대한 재귀적 조회 프로토콜을 상세히 설명합니다.

*1) Worker Node Admission Protocol:* 워커 노드는 실행 계층의 연산 백본이며, 오프체인 계산 및 증명 생성을 담당합니다. 새로운 워커 노드가 네트워크에 참여하려고 할 때 다음과 같은 등록 절차가 시작됩니다:

- 근접성 탐색(Proximity Discovery): 허브는 워커의 Node ID와 자신이 알고 있는 모든 허브와의 XOR 기반 논리적 거리를 계산하고, 가장 가까운 K개의 허브를 식별합니다.
- 등록 브로드캐스트(Registration Broadcast): 초기 허브는 이들 K개의 허브에 ADD_WORKER 명령을 전송하여 새로운 워커의 존재를 알립니다.
- 상태 기록(State Recording): 각 수신 허브는 로컬 상태 테이블을 갱신하고 해당 워커를 분산 인덱스에 추가합니다.
- 주기적 재광고(Periodic Re-advertisement): 각 K 허브는 워커 정보를 주기적으로(예: 매 시간) 다시 브로드캐스트하여 검색 가능성을 유지합니다.
- 만료 정책(Expiration Policy): 워커 정보는 게시 후 24시간이 지나면 자동으로 만료되어 오래되거나 무효한 참조를 방지합니다.

이 프로토콜은 장애 허용성을 갖춘 탈중앙화된 워커 인덱싱을 가능하게 하며, 네트워크 전반에 걸친 검색 가능성과 일관성을 보장합니다.

*2)* 허브 노드 가입 및 탈퇴 프로토콜: 네트워크에 참여하려면 새로운 허브 노드 $x$는 최소한 하나의 기존 부트스트랩 노드 $y$에 연결해야 합니다. 참여 절차는 다음과 같습니다:

- 부트스트랩 연결(Bootstrap Contact): 노드 $x$는 부트스트랩 노드 $y$에 연결을 시도하고 $y$를 초기 K-Bucket에 추가합니다.
- 초기 조회(Initial Lookup): 허브 $x$는 자신의 Node ID를 타깃으로 하여 $y$에게 FIND_HUB 요청을 보냅니다.
- 이웃 탐색(Neighbor Discovery): 노드 $y$는 $x$에 가장 가까운 $K$개의 허브를 반환하고, $x$는 이를 자신의 라우팅 테이블에 추가합니다.
- 반복 확장(Iterative Expansion): 허브 $x$는 새로 발견된 허브들에게 FIND_HUB 요청을 재귀적으로 발행하여 라우팅 테이블의 커버리지를 확장합니다.

이 프로세스를 통해 새로운 노드는 빠르게 네트워크의 논리적 토폴로지에 통합됩니다.

허브의 탈퇴는 암시적으로 수동 만료를 통해 처리됩니다. 인접 허브들은 주기적인 PING을 통해 실패한 노드를 감지하고 자동으로 K-Bucket에서 제거합니다. 이러한 자기 복구 속성은 비동기 합의의 "Sleepy Model" [21]에 부합하며, 노드가 일시적으로 오프라인 상태가 되어도 시스템의 보안성과 생존성 보장을 유지할 수 있도록 합니다.

*3) Recursive Hub Lookup Algorithm:* 타깃 Node ID $y$를 가진 허브를 찾기 위해, 허브 $x$는 다음과 같은 방식으로 재귀적 조회를 시작합니다:

- 거리 계산(Distance Computation): 논리적 거리 $d(x, y) = x \oplus y$를 계산합니다.
- 버킷 선택(Bucket Selection): $\lfloor \log d \rfloor$를 계산하여 적절한 K-Bucket을 결정하고, 최대 $\alpha$개의 후보 허브를 선택합니다.
- 초기 질의(Initial Query): 이들 $\alpha$개의 후보에게 FIND_HUB를 전송합니다.
- 재귀적 확장(Recursive Expansion):
  - 조회된 노드 중 하나가 ID $y$를 가지고 있다면, 해당 노드는 자신의 메타데이터를 응답합니다.
  - 그렇지 않다면, 각 응답 허브는 자신과 $y$ 사이의 거리를 측정하고, 라우팅 테이블에서 더 가까운 $\alpha$개의 후보 허브를 반환합니다.
  - 허브 $x$는 이 과정을 반복하여 $y$에 가장 가까운 $K$개의 허브를 수집하거나, 더 이상 가까운 후보가 발견되지 않을 때까지 계속합니다.

이러한 재귀적 접근은 타깃이 네트워크에 정확히 존재하지 않더라도 수렴을 보장하며, XOR 기반 메트릭을 통한 견고한 경로 탐색을 가능하게 합니다. 파라미터 $\alpha = 3$은 검색 병렬성과 메시지 오버헤드 사이의 균형을 고려하여 선택되었습니다.

### F. 재귀적 워커 조회 프로토콜

특정 워커 노드를 찾기 위해 허브 검색과 유사한 재귀적 조회가 수행됩니다. 예를 들어:

- $\text{Hub}_1$의 ID가 00000110이고, Node ID가 00010000인 워커를 찾으려 한다고 가정합니다.
- XOR 거리는 $00000110 \oplus 00010000 = 00010110$, 즉 10진수 22이며, 이는 K-Bucket 5의 거리 범위(즉 $[2^4, 2^5)$ 또는 $[16, 32)$)에 속합니다.
- $\text{Hub}_1$은 자신의 K-Bucket 5를 검사합니다:
  - 해당 워커가 이미 알려진 허브(예: $\text{Hub}_2$)에 의해 관리되고 있다면, $\text{Hub}_1$은 $\text{Hub}_2$에 직접 질의합니다.
  - 그렇지 않다면, $\text{Hub}_1$은 해당 버킷에서 후보 허브 $\text{Hub}_3$을 선택하고 검색을 계속하도록 요청합니다.
  - $\text{Hub}_3$이 원하는 워커를 가지고 있지 않다면, 더 가까운 $\text{Hub}_4$를 반환하고 검색은 재귀적으로 계속됩니다.

이러한 재귀적 수렴은 토폴로지가 자주 변경되거나 일부 라우팅 실패가 발생해도 안정적이고 효율적인 워커 탐색을 보장합니다.

### G. 분산 네트워크의 위협 모델 및 보안 분석

ZEROBASE 네트워크의 개방형 접근성과 고도로 탈중앙화된 구조 특성상, 시스템은 다양한 유형의 적대적 위협에 노출됩니다. 노드 참여가 무허가 방식으로 이뤄지고 라우팅 상태가 동적으로 변화함에 따라 공격 표면이 크게 확장됩니다. 본 절에서는 네트워크 안정성과 무결성에 실질적 위험을 초래할 수 있는 네 가지 주요 공격 벡터—Sybil 공격, Eclipse 공격, Churn 공격, 적대적 라우팅 공격—을 식별하고 분석합니다.

*1) Sybil Attacks:* Sybil 공격은 악의적인 허브 노드가 네트워크 내에서 여러 개의 신원을 불법적으로 생성하고 이를 통해 과도한 영향력을 행사하려는 시도입니다 [22], [23]. 공격자는 이러한 가짜 신원을 활용하여 네트워크를 교란할 수 있습니다. 주요 영향은 다음과 같습니다:

- 허위 허브 삽입: 무허가 네트워크에서는 공격자가 반복적인 가입 요청을 통해 응답 노드로부터 이웃 목록을 획득할 수 있으며, 이를 통해 네트워크 토폴로지를 파악하고 후속 공격을 준비할 수 있습니다.
- 라우팅 테이블 조작: ZEROBASE는 정확한 라우팅 테이블 유지를 위해 허브 간 공지가 필수적입니다. Sybil 공격자는 여러 허브를 가장함으로써 라우팅 테이블에 침투하고 라우팅 경로를 왜곡시킬 수 있으며, 심한 경우 Eclipse 공격으로 발전할 수 있습니다.
- 허위 리소스 게시: 악성 노드는 존재하지 않는 워커 노드를 등록하고 이를 네트워크에 광고할 수 있습니다. 이러한 워커는 작업 요청을 수신해도 아무런 처리를 하지 않고 무시하며, 이러한 가짜 워커의 증가는 증명 생성 효율성과 전체 처리량을 저하시킵니다.

*2) Eclipse Attacks:* Eclipse 공격은 공격자가 피해 허브의 K-Bucket을 악의적인 노드로 채워 넣음으로써, 피해 노드를 정당한 네트워크로부터 고립시키는 행위입니다 [24], [25]. 이 공격을 실행하기 위해, 공격자는 먼저 충분한 수의 Sybil 허브를 구축한 뒤, 이를 타깃 허브의 라우팅 테이블에 전략적으로 삽입해야 합니다. Eclipse 공격의 영향은 다음과 같습니다:

- 데이터 접근 불가: 피해 노드는 정당한 노드와 통신할 수 없어 데이터 질의가 실패하거나 지연됩니다.
- 라우팅 성능 저하: 오염된 라우팅 테이블은 재귀 루프를 유발하거나 유효하지 않은 노드로 리디렉션되어 질의 효율을 심각하게 저하시킵니다.
- 네트워크 분할: 다수의 허브가 동시에 공격당할 경우, 네트워크는 고립된 서브넷으로 분열될 수 있으며, 이는 탈중앙성과 장애 허용성을 위협합니다.

*3) Churn Attacks:* Churn 공격은 분산 네트워크가 동적인 노드 참여에 대해 자연스럽게 허용적인 특성을 악용합니다. 이러한 공격에서 공격자는 의도적으로 노드의 빈번한 참여 및 이탈을 유도하여 시스템의 운영 안정성을 방해합니다. 분산 시스템은 일반적으로 자연스러운 churn에 대해 견고하지만, 인위적으로 증가된 churn 수준은 다음과 같은 결과를 초래할 수 있습니다:

- 네트워크 성능 저하: 라우팅 테이블의 지속적인 재구성과 데이터 리밸런싱이 지연을 증가시키고 질의 성공률을 감소시킵니다.
- 워커 가용성 감소: 빈번한 노드 churn은 워커 노드가 사라지거나 접근 불가능해지게 하여 증명 생성 및 작업 수행을 중단시킵니다.
- 허브 신뢰도 하락: 경로 불일치나 업데이트 실패가 자주 발생하면 정당한 허브조차 불안정한 것으로 간주되어 신뢰 점수와 참여도가 저하됩니다.

*4) Adversarial Routing Attacks:* 적대적 라우팅 공격은 악의적 주체가 라우팅 경로를 고의적으로 조작하여 네트워크 트래픽을 가로채거나 오도하거나 차단하는 행위를 말합니다. 이러한 공격은 라우팅 작동의 무결성과 가용성을 겨냥하며, 그 영향은 다음과 같습니다:

- 라우팅 테이블 손상: 위조된 라우팅 정보 삽입 또는 라우팅 전파 과정의 취약점을 악용하여 정당한 노드의 라우팅 동작을 교란시킬 수 있습니다.
- 경로 탈취 및 트래픽 제어: 공격자는 트래픽을 손상된 노드로 리디렉션시켜 도청, 조작 또는 트래픽 분석을 수행할 수 있습니다.
- 응답 위조 및 데이터 오염: 라우팅 또는 질의 해결 중, 공격자는 응답 메시지를 위조하거나 변조하여 질의 실패나 데이터 불일치를 유도할 수 있습니다.
- 통신 방해: 특정 노드와 네트워크 간 연결을 차단함으로써 부분적인 네트워크 장애 또는 분리된 서브그래프를 생성할 수 있습니다.

## IV. 프라이버시 보호 연산

ZEROBASE 아키텍처는 TEE + ZKP + Proof Mesh의 긴밀하게 조정된 계층을 통해 네이티브 프라이버시 보호 실행 환경을 실현합니다. 이를 통해 민감한 오프체인 데이터는 신뢰할 수 있는 조건 하에서 처리되며, 검증 가능한 결과는 제로 지식 상태로 게시됩니다. 이는 연산 수명 주기 전반에 걸쳐 기밀성, 검증 가능성, 불변성을 보장합니다. 이 프라이버시-컴퓨팅 프레임워크는 데이터 보안을 강화할 뿐 아니라, 재사용 가능한 메커니즘과 알고리즘 구조도 함께 캡슐화합니다.

### A. 데이터 기밀성 메커니즘: 종단 간 암호화 & 격리된 실행

ZEROBASE의 프라이버시 프레임워크는 데이터의 전체 수명 주기에 걸쳐 보안을 유지하며, 특히 신뢰할 수 없는 오프체인 환경에서도 암호화를 유지합니다:

- 온체인 입력 암호화(On-chain Input Encryption): 온체인에 제출되기 전 또는 워커에게 전달되기 전에, 모든

사용자 데이터는 대칭 키 암호화와 세션 키 래핑을 결합한 방식으로 Enc(data) 형태로 암호화됩니다. 이로 인해 오직 대상 워커의 TEE만이 해당 데이터를 복호화하고 처리할 수 있습니다. 이 프로토콜은 전송 중 기밀성을 보호하며 중간자 공격을 방지합니다.

- TEE 격리 실행 모델(TEE Enclosed Execution Model): 워커 노드의 로컬 신뢰 실행 환경(TEE) 내에서 암호화된 데이터가 복호화 및 격리 실행됩니다 [26]. 실행 파이프라인은 다음을 포함합니다:
  - TEE 내부에서만 입력 데이터를 복호화;
  - 밀폐된 환경에서 개인 모델 또는 전략 로직을 실행;
  - 결과를 생성하고 노출 전 재암호화;
  - 입력 또는 내부 상태를 공개하지 않고도 검증 가능한 Zero-Knowledge Proof(ZKP)를 생성.
- 중간 상태는 모두 삭제되며, 어떠한 잔여 흔적도 남지 않아 데이터 기밀성과 조작 방지성을 보장합니다.
- 암호화 하드웨어 보호(Cryptographic Hardware Protections): ZEROBASE는 AMD SEV, Intel TDX, ARM CCA 등 하드웨어 기반 격리 기술을 활용하여 메모리 암호화 및 논리 I/O 장벽을 구현합니다 [13–15]. 이를 통해 계산 중에도 데이터 기밀성이 유지됩니다. 향후 확장성으로는 완전 동형 암호(FHE)를 도입하여 어떤 단계에서도 평문을 노출하지 않고도 완전한 프라이버시 연산을 지원할 수 있습니다 [27].

이러한 아키텍처의 암호화, 안전한 실행 인클레이브, 하드웨어 격리 기술 활용은 블록체인 시스템에서 검증과 프라이버시를 분리했던 선행 연구 [28], [29]와 달리, 탈중앙화되고 최소 신뢰 기반의 프라이버시 인프라를 촉진합니다。

### *B.* 검증 가능성 메커니즘: 실행 컨텍스트로부터의 *ZKP* 생성

프라이빗하게 수행된 결과를 온체인에서 검증할 수 있도록 하기 위해, ZEROBASE는 오프체인 연산을 제로 지식 제약 조건으로 변환하는 형식적 모델링 프레임워크를 도입합니다:

- 컨텍스트 인식 회로 템플릿(Context-Aware Circuit Templates): ZEROBASE는 다양한 프라이버시 연산 작업에 맞춘 표준화된 ZKP 회로 템플릿 라이브러리를 제공합니다:
  - 범위 검사(range_check): 수치 값이 지정된 구간 내에 있는지 검증합니다.
  - 모델 추론 증명(model_inference): 주어진 입력에 대해 모델 출력의 정확성을 검증합니다.
  - 데이터 변환 검사(data_transformation): 상태 전이의 유효성을 확인합니다.
- 각 템플릿은 특정 작업 유형을 해당 회로 구조에 매핑합니다.
- 제약 시스템 조립(Constraint System Assembly): TEE 실행이 완료되면 입력, 상태, 중간 변수, 출력 등 핵심 요소를 추출하여 R1CS 또는 AIR 제약 시스템을 구성합니다. 이후 효율적인 증명 시스템(Groth16, PLONK 등)을 사용하여 zk-SNARK 또는 zk-STARK 증명을 생성합니다. 상태 및 컨텍스트를 포함하는 공개 입력은 민감한 정보를 노출하지 않고 온체인 스마트 계약에 의한 정확성 검증을 가능하게 합니다.
- 알고리즘 최적화(Algorithmic Optimizations):
  - 증인 압축(Witness Compression): 중복된 증인 구성 요소를 제거하여 증명 크기를 줄입니다.
  - 재귀 증명(Recursive Proofs): 여러 증명 단계를 병합하거나 중첩함으로써 온체인 검증을 간소화합니다.
  - 회로 파라미터화(Circuit Parameterization): 특정 애플리케이션 요구사항에 맞춰 템플릿을 조정하여 빠른 배포 및 효율성을 지원합니다.

### *C.* 조합성 메커니즘: *Proof Mesh* 내 캡슐화

Proof Mesh 계층은 단순한 전송 중개자 역할을 넘어, 조합 가능한 프라이버시 연산을 조율하는 역할도 수행합니다. 그 주요 메커니즘은 다음과 같습니다:

- 모듈형 작업 추상화(Modular Task Abstraction): 각 프라이버시 연산은 독립적인 모듈로 캡슐화됩니다: Task := (InputEnc, StrategyHash, ZKP_Circuit_ID, OutputDigest, Proof $\pi$)
  - InputEnc: 암호화된 입력 페이로드 [30]
  - StrategyHash: 실행 로직의 요약 해시
  - ZKP_Circuit_ID: 해당 회로 템플릿의 식별자
  - OutputDigest: 결과를 요약한 해시 값
  - Proof $\pi$: 정확한 실행을 증명하는 ZK 증명
- 모듈 간 증명 재사용(Inter-Module Proof Reuse):
  - 모듈 간 연결: 하나의 작업 결과를 다른 작업의 입력으로 활용하여 프라이버시 연산 시퀀스를 구성할 수 있습니다.
  - 오프체인 폐쇄 루프 흐름: 사용자가 프라이빗 모델 추론을 트리거한 후 추가 프라이빗 연산이나 계약 호출을 이어서 수행할 수 있습니다.
  - DApp 간 프라이버시 상호 운용성: 서로 다른 애플리케이션이 검증된 증명을 공유함으로써 탈중앙화된 프라이버시 연산 네트워크를 구축할 수 있습니다.
- 버전 관리 및 컨텍스트 패키징(Versioning & Context Packaging): 각 작업은 사용자 ID 해시, 타임스탬프, 전략 해시 등의 실행 메타데이터를 포함하며, 추적 가능한 증명 체인의 일부를 형성합니다. 구독자는 증명 게시 이벤트나 작업 상태를 모니터링할 수 있어 투명하고 조합 가능한 실행 흐름을 지원합니다. 이는 추상화 계층에서 안전성을 보장하는 최신 프라이버시 지향 도메인 특화 언어의 발전 방향을 반영합니다 [31].

### *D.* 보안 기반 & 공격 저항 전략

ZEROBASE는 실행, 증명, 중재 계층 전반에 걸쳐 다층적인 보안 구조를 내재화하여, 적대적 위협, 수동 감청, 변조로부터 방어하는 심층 방어 체계를 구축합니다.

*1) TEE Layer Protections:*

- 원격 인증(Remote Attestation): 실행 전에 각 워커는 하드웨어 서명된 인증 정보를 네트워크와 교환하여, 진정한 TEE만이 참여할 수 있도록 보장합니다 [23], [32].
- 암호화 저장소 및 I/O 격리: 데이터는 TEE 내부 저장소에서 암호화된 상태로 유지되며, 입출력 채널은 모두 논리적으로 격리되어 사이드 채널 공격 및 메모리 감청으로부터 방어됩니다. 이는 AMD SEV, Intel TDX와 같은 하드웨어 기반 보호 기술에 의해 구현됩니다.
- 전략 지문 연동(Strategy-Fingerprint Binding): 각 실행 작업은 검증된 전략 해시(StrategyHash)에 바인딩되어, 승인된 연산 로직만 실행되었음을 보장하고 악성 코드 대체나 조작을 방지합니다.

*2) ZKP Layer Protections:*

- 구조적으로 구현된 데이터 기밀성: 증명 시스템은 입력 데이터를 완전히 은닉한 채 정확성을 검증하며, 증명 유출로부터의 통계적 추론을 방지하기 위해 차등 프라이버시 메커니즘도 선택적으로 적용할 수 있습니다 [33].
- 재사용 방지 보호: 각 증명 번들은 고유 작업 식별자와 타임스탬프를 포함하여, 재생 공격 및 문맥 대체 공격을 방지합니다.
- 회로 감사 및 등록: 모든 ZKP 회로 템플릿은 필수 보안 감사를 거치고 등록되어야 하며, 이를 통해 백도어 부재 및 논리적 일관성을 보장합니다.

*3) Proof Mesh Layer Protections:*

- 다중 서명 검증(Multi-Signature Validation): 각 증명 패키지는 암호학적 서명 및 메타데이터를 포함하며, 작업 출처, 실행 계보, 노드 ID 등을 표시하여 검증자가 증명의 무결성을 확인할 수 있도록 합니다.
- 연결된 증명 체인(Linked Proof Chain): 각 증명은 이전 작업의 출력 해시를 후속 작업의 공개 입력에 포함시켜 체이닝되며, 이를 통해 부분적인 조작이나 분리 공격에 저항하는 감사 가능한 DAG 구조를 형성합니다.
- 제로 트레이스 모드(Zero-Trace Mode): 고보안 작업의 경우, 메타데이터를 암호화하거나 생략함으로써 증명 게시 시 관측 가능한 흔적을 남기지 않으며, 메타데이터 분석 및 프로파일링을 방지합니다.

이러한 하드웨어 기반 신뢰, 암호학적 완전성, 계층 간 조율을 결합한 보안 메커니즘을 통해 ZEROBASE는 조작, 도청, 구조적 취약성에 강한 견고한 프라이버시 연산 프레임워크를 제공합니다. 이는 DeFi부터 프라이버시 보호 AI 추론에 이르기까지 신뢰 가능한 실행을 실현합니다.

## V. 사용자 참여 경로 및 애플리케이션 아키텍처

ZEROBASE 네트워크는 다양한 역할 유형을 수용하도록 설계되었으며, 각 역할은 신뢰 최소화 실행 프레임워크와 명시적인 입출력 인터페이스를 통해 상호작용합니다. 이러한 역할은 신뢰 실행, 증명 생성, 유동성 모듈 내에 통합되어 모듈화되고 재사용 가능하며 조합 가능한 참여 패러다임을 구성합니다.

### *A.* 참여 경로

리소스 기여 경로(Resource Contribution Pathway): 대역폭, 컴퓨팅 파워, 구형 GPU와 같은 유휴 자원을 보유한 사용자는 ZEROBASE 클라이언트 또는 SDK가 내장된 리소스 집계기를 통해 네트워크에 기여할 수 있습니다. 작업은 다양한 플랫폼에 스케줄링되며, SEV-SNP 보호된 TEE 내에서 실행됩니다. 완료 후에는 검증 가능한 리소스 사용 증명이 생성되며, 이는 제로 지식 회로를 통해 구조화된 온체인 증명 자격 증명으로 변환됩니다. 이 자격 증명은 스테이킹, 대출, 혹은 다양한 금융 전략에 활용될 수 있으며, "오프체인 자원 → 검증 가능한 상태 → 조합 가능한 자산"이라는 연결고리를 형성합니다.

전략 실행 경로(Strategy Execution Pathway): 독점 트레이딩 모델을 보유한 자산운용사 또는 퀀트 팀은 TEE 호환 실행 이미지로 로직을 캡슐화할 수 있습니다. 원격 인증을 통해 실행 환경이 등록되면, 해당 전략은 실시간 시장 데이터를 기반으로 오프체인에서 실행되며, 리스크 지표 및 수익 분포를 나타내는 제로 지식 간격 증명을 지속적으로 출력합니다. 온체인 검증자는 이 증명의 무결성을 평가하고 zkStaking 모듈을 통해 상호작용합니다. 이 경로는 "보이지 않는 전략, 그러나 투명한 리스크"라는 설계 목표를 실현합니다.

프로토콜 통합 경로(Protocol Integration Pathway): 외부 DeFi 또는 Web3 프로토콜은 zkRouter 또는 zkReport와 같은 표준화된 인터페이스를 통해 ZEROBASE를 통합할 수 있습니다. 통합자는 전략 풀이나 리소스 토큰의 상태 증명을 쿼리하여 담보 비율 조정, 청산 임계값 설정, 혹은 프로토콜 간 로직 조정을 수행할 수 있습니다. 이러한 상호작용은 오프체인 조정이나 신뢰된 중개자 없이도 가능하며, 전 과정은 구조화된 제로 지식 증명 및 온체인 후크를 통해 구동되어 높은 재사용성과 낮은 신뢰 가정으로 작동합니다.

### *B.* 사례 중심 활용 예시

- 전략 실행 예시: 한 헤지펀드는 델타-뉴트럴 트레이딩 알고리즘을 SEV VM 실행 파일로 패키징합니다. 시스템은 주기적으로 레버리지 한도 및 손실 리스크에 대한 증명을 zkStaking 상에 게시하며, 전략 내용을 공개하지 않고도 투자자에게 노출됩니다. 투자자는 이러한 제로 지식 보증을 기반으로 참여 여부를 결정할 수 있어 기밀 토큰화 펀딩이 가능합니다.
- 리소스 활용 예시: 한 개인이 남는 컴퓨팅 자원을 ZEROBASE 집계기에 연결하여 병렬 작업 실행에 참여합니다. TEE에서 연산이 수행되고, 제로 지식 수익 증명이 발행되며, 온체인 리소스 토큰이 민팅됩니다. 이 토큰은 스테이킹, 전송, 다른 프로토콜 내 중첩이 가능하여 유휴 자원의 금융화를 실현합니다.
- 프로토콜 간 조합 예시: 한 DeFi 대출 플랫폼이 zkRouter를 사용해 사용자의 전략 풀 내 지급 능력 상태를 쿼리합니다. 증명이 온체인에서 검증되며, 외부 오라클이나 프로토콜 간 메시지 없이도 LTV 비율이 업데이트됩니다.
- 인프라 통합 예시: Web2 데이터 제공자가 외부 API(예: 신용 등급 또는 투표 결과)를 SEV로 격리된 실행 노드에 직렬화합니다. 실행 및 제로 지식 증명 후, 스마트 계약을 위한 신뢰할 수 있는 온체인 데이터 요약이 생성되어 Web2 출력을 Web3 신뢰 환경으로 연결합니다.

인터페이스 계약, 증명 형식, 참여자 추상화를 표준화함으로써 ZEROBASE는 다양한 참여 방식을 지원하는 조합 가능한 실행 매체를 수립합니다. 참여자는 유연하게 역할을 전환할 수 있으며, 예를 들어 리소스 제공자가 전략 노드 운영자로 발전하거나, 통합자가 새로운 상태 기반 증명 요건을 정의할 수 있습니다. 그 결과, 진화 가능하고 자기 조직화되는 참여자 토폴로지가 자연스럽게 형성됩니다。

## VI. 결론

투명성과 프라이버시가 충돌하고, 오프체인 실행이 온체인 검증과의 정렬에 어려움을 겪으며, 잠재된 자원이 활용되지 못하는 현실 속에서, ZEROBASE는 구조적 해결책을 제시합니다: 아키텍처에 신뢰를 내재화하고, 검증 권한을 사용자와 프로토콜에게 되돌려줍니다.

신뢰 실행(TEE), 제로 지식 증명(ZKP), 조합 가능한 유동성 구조를 활용하여, ZEROBASE는 인간의 해석을 요구하지 않는 시스템을 제공합니다. 전략은 공개되지 않지만 리스크는 감사를 통해 확인할 수 있고, 자원은 수탁자나 불투명한 계약 없이 활용될 수 있으며, 자본은 플랫폼 보증이 아닌 구조적 규칙을 통해 조율됩니다. 이것이 바로 구조적 투명성과 신뢰 최소화 협업의 본질입니다.

ZEROBASE는 기존 시스템을 대체하거나 고정된 완성 제품을 정의하기 위해 설계된 것이 아닙니다. 대신, 개발자가 재사용하고, 생태계가 시너지를 이루며, 실행 주체가 운용할 수 있는 신뢰 기반의 핵심 기저층으로 구축되었습니다. 우리는 전략 제공자에게 ZEROBASE를 사용하여 고유하지만 검증 가능한 금융 상품을 구축할 것을, 자원 보유자에게는 현실 세계의 컴퓨팅 자원을 온체인에서 노출할 것을, 그리고 개발자에게는 새로운 모듈을 삽입하고, 새로운 유스케이스를 확장하며, 새로운 구조를 설계할 수 있는 조합 엔진으로 활용할 것을 권장합니다. 우리는 이 아키텍처의 초기 버전을 공개했으며, 이는 결국 모두의 것이 될 것입니다。

## 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.