

하이퍼레인(Hyperlane)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다

소개

프로토콜 개요

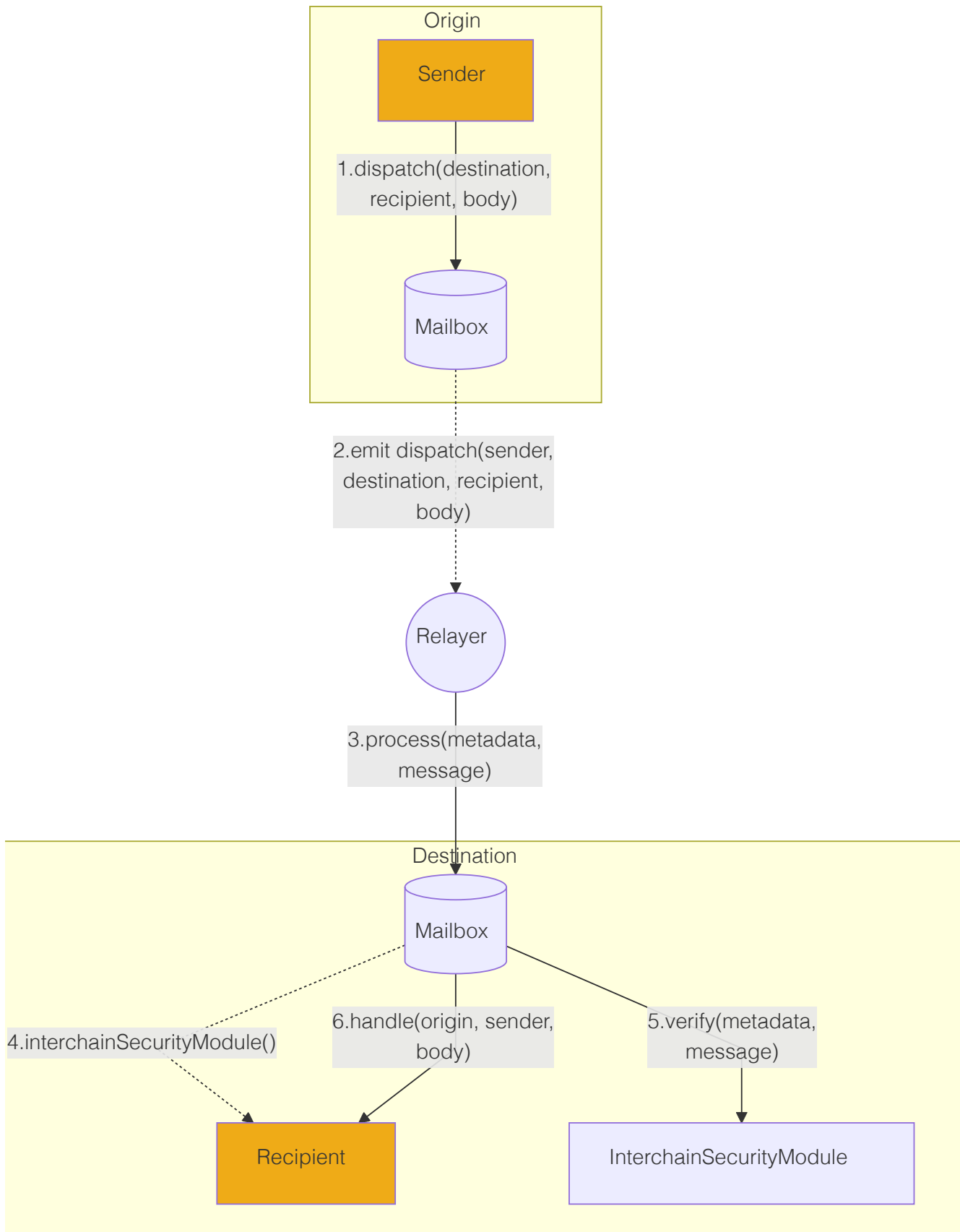
하이퍼레인의 크로스체인 메시징 프로토콜 및 핵심 구성요소 개요

하이퍼레인은 블록체인 간 임의의 데이터를 전송할 수 있도록 허용하는 최초의 [비허가형 상호운용성] 레이어다.

개발자는 하이퍼레인을 사용하여 토큰을 이동하거나, 함수 호출을 실행하거나, 그 외 다양한 작업을 수행할 수 있다. 이를 통해 모든 블록체인 사용자가 접근할 수 있는 인터체인 애플리케이션을 구축할 수 있다.

사용자는 [Mailbox] 스마트 컨트랙트를 통해 하이퍼레인 프로토콜과 상호작용하며, 이를 통해 인터체인 메시지를 송수신할 수 있는 온체인 메시징 인터페이스를 이용한다.

하이퍼레인은 모듈식 보안 접근방식을 채택하고 있으며, 애플리케이션이 다양한 [인터체인 보안 모듈] (Interchain Security Modules, ISM) 중에서 선택하고 구성할 수 있도록 한다. 각 애플리케이션은 자신이 하이퍼레인 메시징 인터페이스와 통합되는 과정에서 적용할 보안 모델을 커스터마이징하기 위해 특정 ISM을 지정할 수 있다.



프로토콜

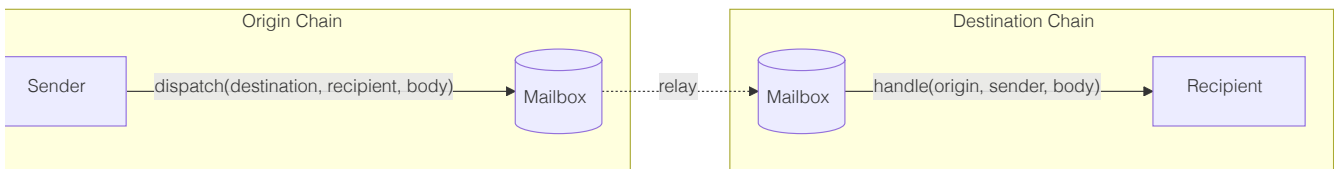
일반 메시지 전송

메일박스 (Mailbox)

하이퍼레인에서 크로스체인 메시지를 송수신하기 위한 핵심 스마트 컨트랙트 인터페이스

하이퍼레인의 Mailbox 스마트 컨트랙트는 인터체인 메시지를 송수신하기 위한 온체인 API를 제공한다. 하이퍼레인이 지원하는 모든 체인에는 각각 하나의 Mailbox 컨트랙트가 배포되어 있다.

이 Mailbox 네트워크는 블록체인 간 연결 구조를 형성하며, 개발자들은 이를 활용해 인터체인 애플리케이션을 구축하거나 기존 애플리케이션에 인터체인 기능을 추가할 수 있다.



- 인터체인 메시지를 **전송**하려면 `dispatch` 함수를 호출한다.
- 인터체인 메시지를 **수신**하려면 `handle` 함수를 구현한다.

인터페이스

IMailbox 인터페이스는 두 가지 상태 변경 함수인 `dispatch()`와 `process()`를 제공한다. 이 함수들은 각각 메시지를 전송하고 수신하는 데 사용된다.

```
// SPDX-License-Identifier: MIT OR Apache-2.0
pragma solidity >=0.8.0;

import {IInterchainSecurityModule} from
"./IInterchainSecurityModule.sol";
import {IPostDispatchHook} from "./hooks/IPostDispatchHook.sol";

interface IMailbox {
    // ===== Events =====
    /**
     * @notice Emitted when a new message is dispatched via Hyperlane
     * @param sender The address that dispatched the message
     * @param destination The destination domain of the message
     * @param recipient The message recipient address on `destination`
     */
}
```

```

    * @param message Raw bytes of message
    */
event Dispatch(
    address indexed sender,
    uint32 indexed destination,
    bytes32 indexed recipient,
    bytes message
);

/**
 * @notice Emitted when a new message is dispatched via Hyperlane
 * @param messageId The unique message identifier
 */
event DispatchId(bytes32 indexed messageId);

/**
 * @notice Emitted when a Hyperlane message is processed
 * @param messageId The unique message identifier
 */
event ProcessId(bytes32 indexed messageId);

/**
 * @notice Emitted when a Hyperlane message is delivered
 * @param origin The origin domain of the message
 * @param sender The message sender address on `origin`
 * @param recipient The address that handled the message
 */
event Process(
    uint32 indexed origin,
    bytes32 indexed sender,
    address indexed recipient
);

function localDomain() external view returns (uint32);

function delivered(bytes32 messageId) external view returns
(bool);

function defaultIsm() external view returns
(IInterchainSecurityModule);

function defaultHook() external view returns (IPostDispatchHook);

function requiredHook() external view returns (IPostDispatchHook);

```

```

function latestDispatchedId() external view returns (bytes32);

function dispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata messageBody
) external payable returns (bytes32 messageId);

function quoteDispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata messageBody
) external view returns (uint256 fee);

function dispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata body,
    bytes calldata defaultHookMetadata
) external payable returns (bytes32 messageId);

function quoteDispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata messageBody,
    bytes calldata defaultHookMetadata
) external view returns (uint256 fee);

function dispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata body,
    bytes calldata customHookMetadata,
    IPostDispatchHook customHook
) external payable returns (bytes32 messageId);

function quoteDispatch(
    uint32 destinationDomain,
    bytes32 recipientAddress,
    bytes calldata messageBody,
    bytes calldata customHookMetadata,
    IPostDispatchHook customHook
) external view returns (uint256 fee);

function process(

```

```

        bytes calldata metadata,
        bytes calldata message
    ) external payable;

    function recipientIsM(
        address recipient
    ) external view returns (IInterchainSecurityModule module);
}

```

메시지 헤더

Mailbox는 메시지 본문 앞에 다음 필드들을 포함하는 헤더를 추가한다:

필드	설명
version	Mailbox 컨트랙트의 버전
nonce	특정 Mailbox에서 전송된 각 메시지의 고유 식별자
origin	원본 체인의 도메인
sender	원본 체인에서의 송신자 주소
destination	목적지 체인의 도메인
recipient	목적지 체인에서의 수신자 주소

메시지 인코딩에 대한 자세한 내용은 [Message 라이브러리]를 참고하라.

고유성 (Uniqueness)

nonce는 특정 Mailbox에서 전송되는 각 메시지에 대해 단조 증가하는 정수다. 동일한 메시지라도 서로 구분될 수 있도록, 메시지가 전송될 때마다 nonce가 증가한다.

```
function delivered(bytes32 messageId) external view returns (bool);
```

messageId는 전역적으로 고유한 메시지 식별자로, dispatch 호출 시 반환된다. 이는 헤더를 포함한 전체 메시지에 대해 keccak256 해시를 계산하여 생성된다.

재전송 방지 (Replay Protection)

Mailbox는 이미 전달된 messageId 값을 매핑으로 관리하여 재전송 공격(replay attack)을 방지한다. 이미 전달된 messageId를 가진 메시지가 수신되면 해당 메시지는 거부된다.

디스패치 (Dispatch)

인터체인 메시지를 전송하기 위해 개발자는 `Mailbox.dispatch()`를 호출한다.

이 함수는 메시지 내용, 목적지 체인 ID, 그리고 수신자 주소를 매개변수로 받는다. 각 메시지는 Mailbox에 저장된 증분 머클 트리(incremental merkle tree)에 리프(leaf)로 삽입된다.

하이퍼레인의 지분증명(Proof of Stake) 프로토콜은 이 머클 트리를 사용하여 부정 행위(fraud) 증명을 검증한다.

프로세스 (Process)

인터체인 메시지를 전달하기 위해 릴레이어(relayer)는 `Mailbox.process()`를 호출한다.

이 함수는 전달할 메시지와 릴레이어가 지정할 수 있는 임의의 메타데이터를 매개변수로 받는다.

Mailbox는 메시지와 메타데이터를 수신자의 인터체인 보안 모듈(ISM)로 전달하여 검증을 수행한다. ISM이 메시지를 성공적으로 검증하면, Mailbox는 `recipient.handle()`을 호출하여 해당 메시지를 수신자에게 전달한다.

하이퍼레인 메시지 인코딩에 대한 자세한 내용은 [Message.sol](#)을 참고하라.

전송 수수료 (Transfer Fees)

하이퍼레인 크로스체인 전송 시 원본 수수료와 인터체인 수수료 이해하기

수수료 추정 (Fee Estimation)

하이퍼레인 GMP를 사용하여 크로스체인 전송을 수행할 때는 두 가지 수수료가 필요하다:

1. **원본 수수료 (Origin Fee)** – 원본 체인에서 트랜잭션을 제출하기 위한 가스 비용
2. **[인터체인 수수료 (Interchain Fee)]** – 메시지를 목적지 체인에서 전달하고 실행하기 위한 가스 비용(릴레이어 수수료 포함)

이 두 수수료는 모두 원본 체인에서 지정된 IGP(Interchain Gas Paymaster)에 선불로 지불된다.

이 수수료 구조는 다음을 보장한다:

1. 사용자가 원본 트랜잭션을 실행할 수 있을 만큼 충분한 자금을 보유한다.
2. 크로스체인 메시지가 목적지 체인에서 실행될 수 있을 만큼 충분한 가스를 확보하며, 메시지 전달 비용을 지불할 수 있다.
3. 전송의 총비용이 실행 전에 사용자에게 투명하게 제공된다.

수수료 토큰 고려사항 (Fee Token Considerations)

크로스체인 전송 시, 수수료에 사용되는 토큰은 원본 체인 유형과 IGP에 따라 다르다:

1. **EVM 체인**
 - 원본 수수료와 인터체인 수수료 모두 네이티브 가스 토큰(예: ETH)을 사용한다.
2. **Sealevel (Solana) 체인**
 - 원본 수수료와 인터체인 수수료 모두 네이티브 가스 토큰(예: SOL)을 사용한다.
3. **Cosmos 체인**
 - 커스텀 토큰을 지원하며, 수수료는 IGP의 `denom` 파라미터를 통해 지정된다.
 - 수수료는 `denom` 파라미터를 사용하여 정의된다.

전송 수수료 및 계산 방식에 대한 자세한 내용은 다음 문서를 참고하라:

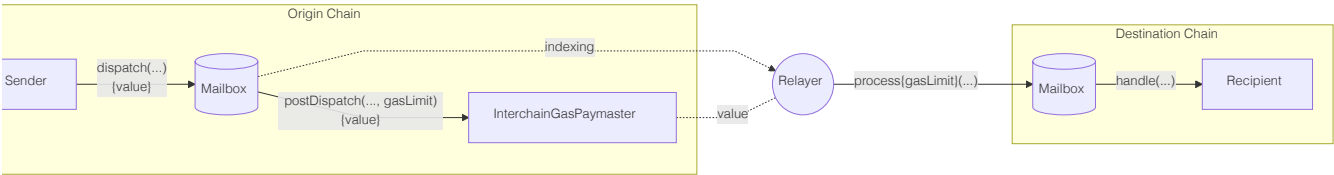
- [인터체인 가스 지불 (Interchain Gas Payment)] – 체인 간 가스 지불 및 릴레이어 수수료 처리 방식
- [Post Dispatch Hooks] – 메시지 전송 이후 동작을 실행하기 위한 프레임워크

인터체인 가스 지불 (Interchain Gas Payments)

하이퍼레인의 `InterchainGasPaymaster`를 사용하여 크로스체인 메시지 전달 시 릴레이어에게 수료를 지불하는 방법

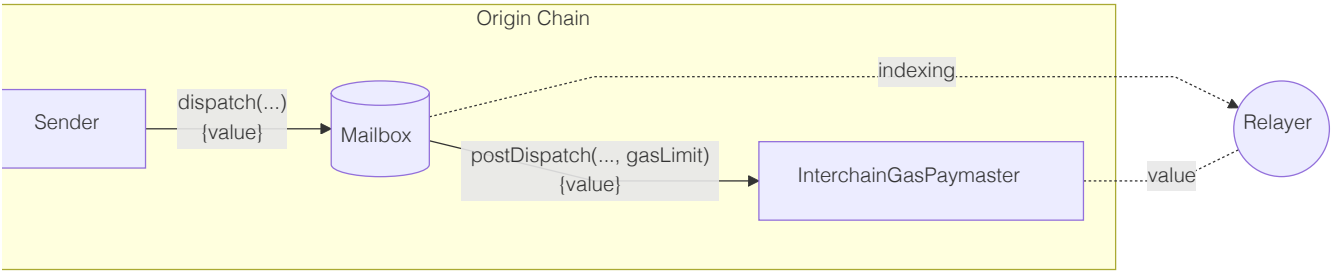
성공적인 인터체인 메시지를 위해서는 두 개의 트랜잭션이 필요하다. 하나는 원본 체인에서 메시지를 전송하기 위한 트랜잭션이며, 다른 하나는 목적지 체인에서 메시지를 전달하기 위한 트랜잭션이다.

편의를 위해 하이퍼레인은 원본 체인에 온체인 인터페이스를 제공한다.
 이 인터페이스를 통해 메시지 송신자는 목적지 체인에서 메시지를 전달하기 위한 수수료를 [릴레이어]에게 지불할 수 있다.
 이 지불은 **인터체인 가스 지불 (interchain gas payment)**이라 불린다.

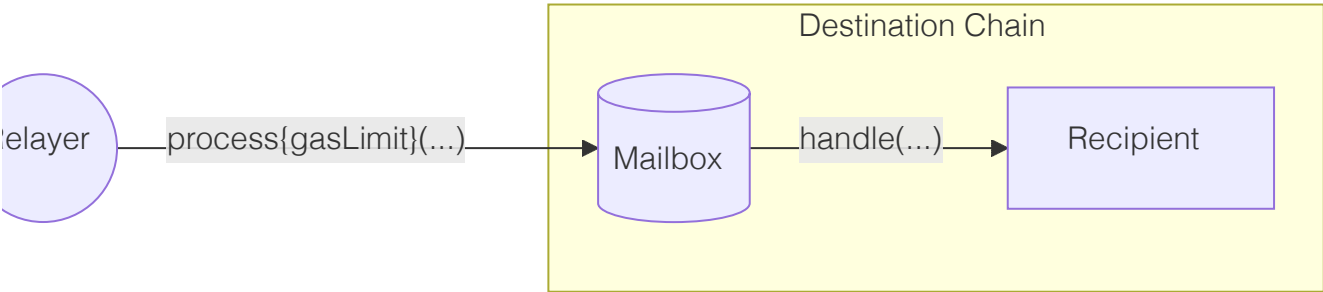


인터체인 가스 지불 프로세스 (Interchain Gas Payment Process)

[메시지를 전달]하기 위해서는, 인코딩된 메시지와 ISM 메타데이터를 포함하여 Mailbox의 process 함수를 호출하는 트랜잭션이 목적지 체인에 포함되어야 한다.



편의를 위해 릴레이어는 전송된 메시지를 모니터링하며, 원본 체인에서 충분한 수수료를 수령한 경우 메시지 송신자를 대신해 process 트랜잭션을 제출한다.
 이것이 바로 **인터체인 가스 지불 (interchain gas payment)**이다.



메시지가 임의의 코드 실행을 트리거할 수 있기 때문에, 릴레이어는 메시지 전송 시 적절한 요금을 부과하기 위해 handle 호출에 gasLimit을 설정해야 한다.

InterchainGasPaymaster

인터체인 가스 지불은 `InterchainGasPayerMaster(IGP)` 스마트 컨트랙트를 통해 수행된다.

이 컨트랙트는 [InterchainGasPayerMaster 인터페이스](#)를 제공하며, 메시지 송신자가 원본 체인의 네이티브 토큰으로 릴레이어에게 목적지 체인에서 메시지를 전달하기 위한 비용을 지불할 수 있도록 한다.

각 `InterchainGasPayerMaster` 컨트랙트는 정확히 하나의 릴레이어에 대응한다.

Abacus Works의 릴레이어 주소는 [addresses](#)에서 확인할 수 있다.

인터체인 가스 지불 계산 (Interchain Gas Payment Calculation)

인터체인 가스 지불은 다음과 같이 계산된다:

$$destinationTxCost = destinationGasPrice * gasLimit \quad (1)$$

$$exchangeRate = \frac{originGasTokenPrice}{destinationGasTokenPrice} \quad (2)$$

$$originFee = exchangeRate * destinationTxCost \quad (3)$$

`gasLimit`은 메시지 송신자가 설정하며, `exchangeRate`는 릴레이어가 설정한다.

신뢰 가정 (Trust Assumptions)

인터체인 가스 지불은 메시지 송신자와 릴레이어 간의 사회적 계약(social contract)에 기반한다.

즉, 릴레이어는 해당 메시지를 실제로 전달하지 않고도 인터체인 가스 지불을 수령할 수 있다.

따라서 인터체인 가스 지불을 수행할 때는 신뢰할 수 있고 평판이 좋은 릴레이어와 연결된 IGP 컨트랙트를 사용하는 것이 권장된다.

정직한 릴레이어 구현체는 실제로 얼마나 많은 토큰이 지불되었는지와 관계없이, 자신이 관리하는 IGP로 성공적으로 이루어진 모든 가스 지불을 이행한다.

릴레이어는 [가스 오라클](#)을 최신 상태로 유지함으로써, “공정한” 가격이 지불된 경우에만 가스 지불이 성공하도록 보장할 수 있다.

하이퍼레인 프로토콜은 릴레이어에 대한 신뢰 가정을 포함하지 않으며, 어떤 상황에서도 악의적인 릴레이어가 메시지를 검열하거나 위조할 수 없다.

릴레이어가 할 수 있는 최악의 행위는 메시지를 전달하지 않고 수수료만 수령하는 것이다.

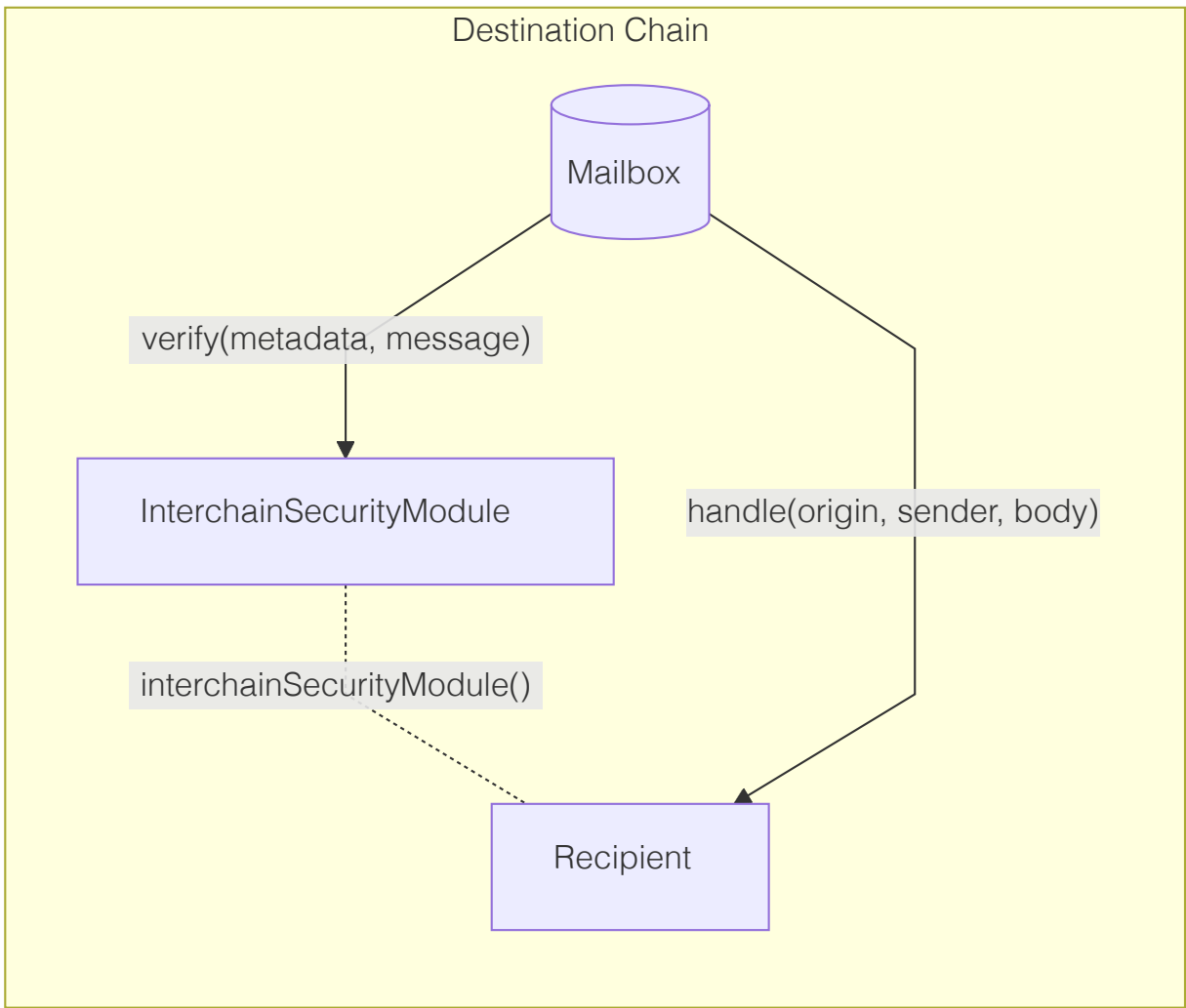
추가 학습 (Learn More)

- 가스 한도 계산, 벤치마크 권장사항, 가스 오라클, 목적지 가스 설정 등에 대한 자세한 내용은 [Interchain Gas Payment Details](#)를 참고하라.
- 릴레이어 설정 및 구현 가이드는 [Relayer](#) 섹션을 참고하라.

모듈형 보안 (Modular Security)

ISM 개요 (ISM Overview)

하이퍼레인은 인터체인 보안 모듈(Interchain Security Modules, ISM)에 의해 보호된다. ISM은 목적지 체인에서 전달되는 인터체인 메시지가 실제로 원본 체인에서 존재했는지를 검증하는 역할을 하는 스마트 컨트랙트다.



핵심 개념 (Core Concepts)

기본 ISM vs 커스텀 ISM (Default vs Custom ISM)

개발자는 Mailbox의 기본 ISM을 사용하거나, 애플리케이션 전용 ISM을 직접 지정할 수 있다.

애플리케이션 전용 ISM은 다음과 같은 형태로 구성될 수 있다:

- **구성형 (Configured):** 사전 구축된 ISM을 사용하되, 사용자 정의 파라미터로 설정
- **조합형 (Composed):** 여러 ISM을 결합하여 보안 "레고"처럼 구성
- **맞춤형 (Customized):** 특정 요구사항에 맞춰 완전히 새로운 ISM을 직접 개발

"기본 ISM(Default ISM)"은 Mailbox 컨트랙트에 사전 설정된 보안 모듈(즉, Multisig ISM)을 의미하며,
애플리케이션이 별도의 커스텀 ISM을 지정하지 않을 경우 사용된다.

구성 (Configure)

하이퍼레인은 여러 사전 구축된 ISM을 정의한다.

개발자는 이러한 컨트랙트를 "기성품(off-the-shelf)" 형태로 배포한 후, 자체 파라미터로 구성할 수 있다.

예를 들어, 인터체인 보안에 대한 주권을 강화하고자 하는 애플리케이션 개발자는 자신의 커뮤니티에서 선출된 검증자(validators)를 기반으로 구성된 Multisig ISM을 배포할 수 있다.

조합 (Compose)

ISM은 “보안 레고(Security Legos)”처럼 작동한다.

개발자는 다양한 ISM을 조합하여 자신에게 가장 적합한 보안 모델을 구성할 수 있다.

예를 들어, 추가적인 보안을 원한다면,

하이퍼레인 커뮤니티의 검증자로 구성된 Multisig ISM 및

Wormhole 검증자 세트의 정족수(quorum)가 메시지를 검증했음을 확인하는 Wormhole ISM 양쪽의 검증을 요구하는

Aggregation ISM을 배포할 수 있다.

맞춤 (Customize)

ISM은 완전히 커스터마이징 가능하다.

개발자는 자신의 애플리케이션 요구사항에 맞춰 직접 ISM을 작성할 수 있다.

예를 들어, 메시지의 중요도나 특성에 따라 보안 모델을 조정하는 ISM을 구축할 수 있다.

높은 가치이면서 빈도가 낮은 메시지(예: 거버넌스)는 지연(latency)과 가스 비용보다 안전성을 우선하는 보안 모델로 검증할 수 있다.

반면, 낮은 가치이면서 빈도가 높은 메시지는 안전성보다 지연과 가스 효율성을 우선하는 모델로 검증할 수 있다.

나만의 ISM 만들기 방법 알아보기

시퀀스 다이어그램 (Sequence Diagram)

체인 간 메시지가 전송될 때, 그 진위(authenticity)를 보장하기 위한 검증 과정이 수행된다.
이 문서는 해당 검증 과정이 어떻게 작동하는지를 설명한다.

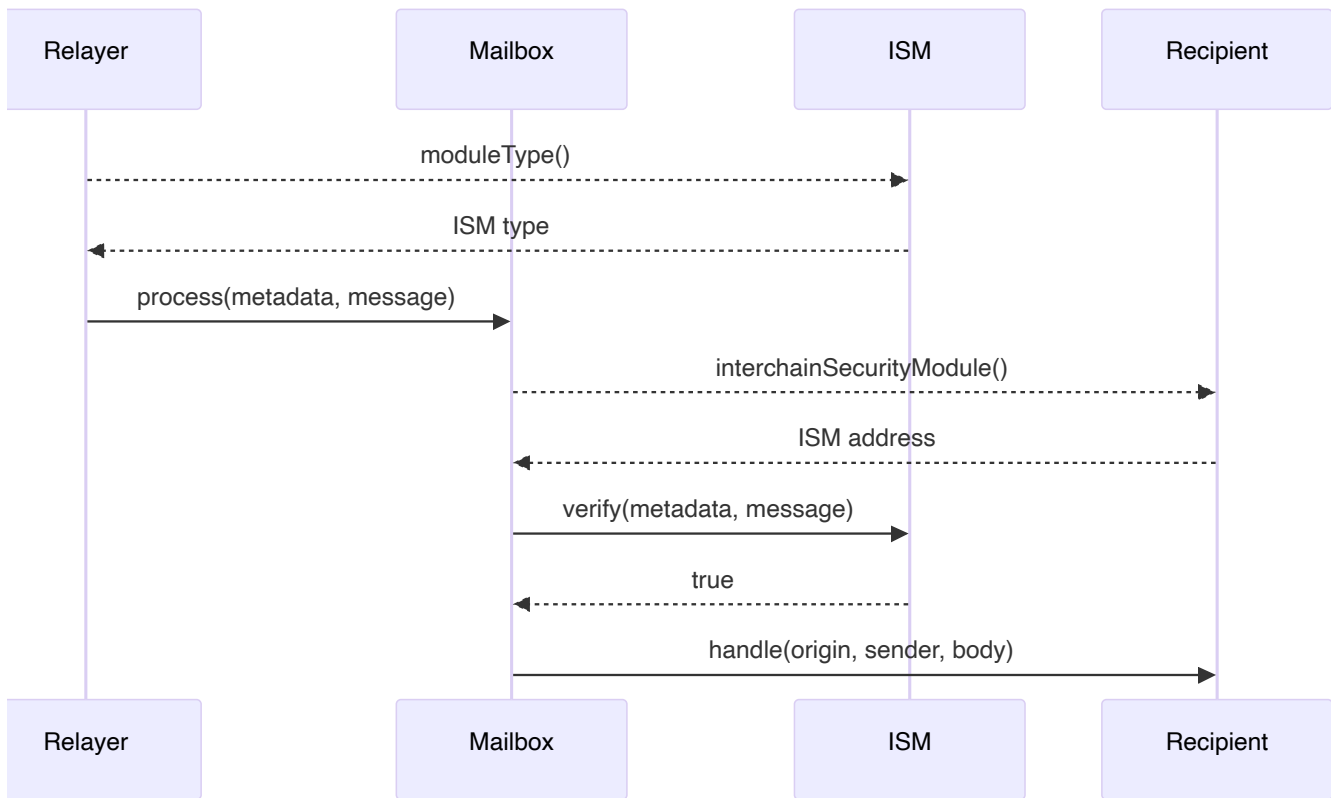
개요 (Overview)

검증 과정에는 다음 세 가지 주요 구성요소가 포함된다:

1. **Mailbox 컨트랙트** – 메시지 처리를 위한 주요 진입 지점
2. **ISM (Interchain Security Module)** – 보안 검증을 담당하는 컨트랙트
3. **수신자(Recipient) 컨트랙트** – 메시지를 실제로 수신하는 애플리케이션

시퀀스 다이어그램 (Sequence Diagram)

- 수신자가 `ISpecifiesInterchainSecurityModule`을 구현하지 않거나 `recipient.interchainSecurityModule()`이 `address(0)`을 반환하는 경우, Mailbox에 구성된 기본 ISM이 메시지 검증에 사용된다.
- 명확성을 위해 이 부분은 시퀀스 다이어그램에서 생략되어 있다.



작동 방식 (How It Works)

1. 유형 식별 (Type Identification)

- 릴레이어는 ISM의 `moduleType()`을 호출하여 그 유형을 확인한다.
- 해당 유형(예: `MULTISIG`, `ROUTING`, `AGGREGATION`)에 따라 릴레이어는 수집해야 할 검증 증거의 종류를 결정한다.
- Multisig ISM의 경우, 검증자들로부터 암호학적 서명을 수집해야 한다.

2. 메시지 제출 (Message Submission)

- 릴레이어는 `Mailbox.process()`를 호출하며 다음 두 가지를 함께 제공한다:
 - 원본 메시지
 - 수집된 검증 메타데이터

3. 검증 과정 (Verification Process)

- Mailbox는 수신자 컨트랙트의 `interchainSecurityModule()`을 호출하여 어떤 ISM이 지정되어 있는지 확인한다.
- ISM이 지정되어 있지 않은 경우, Mailbox는 기본 ISM을 사용한다.
- Mailbox는 메시지와 메타데이터를 ISM의 `verify()` 함수로 전달한다.
- ISM은 자체 보안 로직(예: 서명 검증, 임계값 확인 등)을 실행한다.

4. 메시지 전달 (Message Delivery)

- 검증이 성공적으로 완료되어 `verify()`가 `true`를 반환하면, Mailbox는 `recipient.handle()`을 호출하여 메시지를 전달한다.
- 검증이 실패할 경우, 트랜잭션은 되돌려지고 해당 메시지는 거부된다.

ISM 마켓플레이스 (ISM Marketplace)

ISM 마켓플레이스는 개발자가 메시지 검증 방식을 커스터마이징할 수 있도록 지원하는 인터체인 보안 모듈(ISM)의 레지스트리다.

애플리케이션 팀은 보안 수준, 지연 시간(latency), 비용 요건에 따라 ISM을 선택할 수 있다. 예를 들어, 높은 가치의 메시지에는 강력한 검증을 적용하고, 일반적인 트랜잭션에는 더 가벼운 ISM을 사용할 수 있다.

이 마켓플레이스에는 Abacus Works에서 개발한 표준 ISM과, 파트너 팀 및 커뮤니티 기여자들이 개발한 커뮤니티 ISM이 모두 포함되어 있다.

표준 ISM (Standard ISMs)

표준 ISM은 Abacus Works에서 개발 및 유지관리된다.

ISM 이름	유형	단계	설명
<u>Aggregation ISM (Static)</u>	Composition	In Production	여러 ISM을 결합하여 모든 또는 k-of-n 하위 세트가 검증에 동의해야 메시지가 유효하다고 판단한다.
<u>Aggregation ISM (Storage)</u>	Composition	In Production	여러 ISM을 결합하며, 모든 또는 k-of-n 하위 세트의 동의를 필요로 한다. 모듈 세트 및 임계값은 상태에 저장되며, 배포 후에도 업데이트할 수 있다.
<u>Routing ISM</u>	Composition	In Production	원본 도메인에 따라 메시지를 서로 다른 ISM으로 라우팅한다.
Amount Routing ISM	Composition	In Development	전송 금액에 따라 메시지를 서로 다른 ISM으로 라우팅한다.
<u>Fallback Routing ISM</u>	Composition	In Production	라우팅 맵에 원본 도메인이 구성되어 있지 않을 경우, 백업 ISM 경로를 제공한다.
ICA Routing ISM	Composition	In Production	메시지를 인터체인 계정(Interchain Account, ICA)을 통해 라우팅한다.
<u>Pausable ISM</u>	Composition	In Production	비상 상황에서 메시지 흐름을 일시 중단할 수 있도록 한다.
Rate Limited ISM	Composition	In Development	일정 시간 간격 내에서 HWR를 통해 전송할 수 있는 양을 제한하기 위해 토큰 버킷(rate limiter)을 사용한다. HWR 애플리케이션/메시지 인코딩에만 적용된다.
<u>Merkle Root Multisig ISM</u>	Security	In Production	Merkle 루트에 대해 k-of-n 서명을 검증하고, 이 루트 내 메시지 ID의 Merkle 증명을 사용한다.
<u>Merkle Root Multisig ISM (Storage)</u>	Security	In Production	Merkle 루트에 대해 k-of-n 서명을 검증하고, 메시지 ID의 Merkle 증명을 사용한다. 모듈 세트와 임계값은 상태에 저장되며, 배포 후 업데이트 가능하다.
<u>Message ID Multisig ISM</u>	Security	In Production	메시지 ID에 대해 k-of-n 서명을 검증한다. 서명은 체크포인트 구조에서 가져오며, 관련 ISM 간에 공유된다.

<u>Message ID Multisig ISM (Storage)</u>	Security	In Production	메시지 ID에 대해 k-of-n 서명을 검증한다. 체크포인트 구조에서 가져온 서명이 관련 ISM들 간에 공유되며, 모듈 세트와 임계값은 상태에 저장되어 업데이트 가능하다.
Weighted Merkle Root Multisig ISM	Security	In Development	Merkle Root Multisig을 확장한 가중치 투표 시스템으로, 각 서명자의 영향력이 스테이크에 따라 달라진다.
Weighted Message ID Multisig ISM	Security	In Development	Message ID Multisig 기반의 가중치 투표 시스템이다.
<u>Offchain Lookup ISM</u>	Security	In Production	외부 데이터 소스를 활용한 검증 로직을 구축하기 위한 추상화 계층으로 작동한다.
<u>Chainlink CCIP ISM</u>	Security	In Production	Chainlink CCIP을 활용하여 메시지를 검증한다.
<u>Arbitrum L2 to L1 ISM</u>	Security	In Development	Arbitrum의 네이티브 증명 시스템을 이용해 Arbitrum L2에서 Ethereum L1으로의 메시지를 검증한다.
<u>OP Stack ISM</u>	Security	In Production	OP Stack 롤업에서 오는 크로스체인 메시지를 OP Stack의 네이티브 보안 모델로 검증한다.
<u>Noop ISM / Test ISM</u>	Security	In Production	테스트용 ISM.
<u>Trusted Relayer ISM</u>	Security	In Production	신뢰할 수 있는 릴레이어가 Mailbox에 메시지를 전달하는 방식에 의존한다.

- Static 변형은 바이트코드 상에서 불변(immutable)이다.
- Storage 변형은 상태(state) 내에서 모듈 세트 및 임계값을 변경할 수 있다.

커뮤니티 ISM (Community ISMs)

커뮤니티 ISM은 Abacus Works 외부의 기여자들이 개발 및 유지관리한다.

ISM 이름	유형	단계
Axelar ISM	Security	In Development
Celestia ZK ISM	Security	In Development
Noble ISM	Security	In Development
Polygon zkEVM V2 ISM	Security	In Development
Succinct Lightclient ISM	Security	In Development
Timelock ISM	Security	In Development
Wormhole ISM	Security	In Development

나만의 ISM 만들기 (Create Your Own ISM)

개발자는 애플리케이션 내에서 `ISpecifiesInterchainSecurityModule` 인터페이스를 구현함으로써 기본 ISM을 지정하거나 재정의할 수 있다.

ISM이 지정되지 않았거나, 지정된 ISM이 null 주소(`address(0)`)인 경우, 목적지 체인의 Mailbox에 설정된 기본 ISM이 사용된다.

ISM 인터페이스 (ISM Interface)

ISM은 `InterchainSecurityModule` 인터페이스를 반드시 구현해야 한다. 이 구현체는 애플리케이션의 요구사항에 따라 구성(`configure`), 조합(`compose`), 또는 커스터마이징(`customize`)할 수 있다.

특히, 이 인터페이스는 `handle()`을 구현하는 동일한 스마트 컨트랙트 내에서 구현되어야 한다.

해당 인터페이스는 두 개의 함수로 구성된다: `verify`와 `moduleType`.

```
// SPDX-License-Identifier: MIT OR Apache-2.0
pragma solidity >=0.8.0;
```

```

enum ModuleType {
    UNUSED,
    ROUTING,
    AGGREGATION,
    LEGACY_MULTISIG,
    MERKLE_ROOT_MULTISIG,
    MESSAGE_ID_MULTISIG,
    NULL, // used with relayer carrying no metadata
    CCIP_READ,
    ARB_L2_TO_L1,
    POLYGON_POS,
    OP_STACK
}

interface InterchainSecurityModule {
    /\*\*
    _ @notice Returns an enum that represents the type of security model
    _ encoded by this ISM.
    _ @dev Relayers infer how to fetch and format metadata.
    _/
    function moduleType() external view returns (uint8);

    /**
     * @notice Defines a security model responsible for verifying
interchain
     * messages based on the provided metadata.
     * @param _metadata Off-chain metadata provided by a relayer,
specific to
     * the security model encoded by the module (e.g. validator
signatures)
     * @param _message Hyperlane encoded interchain message
     * @return True if the message was verified
     */
    function verify(
        bytes calldata _metadata,
        bytes calldata _message
    ) external returns (bool);
}

```

Verify

```
/**
 * @notice Defines a security model responsible for verifying interchain
 * messages based on the provided metadata.
 * @param _metadata Off-chain metadata provided by a relay, specific
 * to
 * the security model encoded by the module (e.g. validator signatures)
 * @param _message Hyperlane encoded interchain message
 * @return True if the message was verified
 */
function verify(
    bytes calldata _metadata,
    bytes calldata _message
) external returns (bool);
```

ISM이 반드시 구현해야 하는 주요 함수는 `verify()`이다.

Mailbox는 메시지를 수신자에게 전달하기 전에 `IInterchainSecurityModule.verify()`를 호출한다.

만약 `verify()`가 되돌아가거나(`revert`) `false`를 반환하면, 해당 메시지는 전달되지 않는다.

매개변수 (Parameters)

- `_metadata`: 릴레이어가 제공하는 임의의 바이트 데이터로 구성된다.
일반적으로 이 데이터는 ISM에 특화되어 있다.
예를 들어, Multisig ISM의 `_metadata`에는 검증자 서명이 포함되어야 한다.
- `_message`: 검증 대상인 하이퍼레인 메시지로 구성된다.
ISM은 이를 사용하여 검증 대상 메시지의 세부 정보를 확인할 수 있다.
예를 들어, Multisig ISM은 메시지의 원본 체인에 따라 검증자 세트를 변경할 수 있다.

`verify()`에 전달되는 하이퍼레인 메시지의 형식에 대한 자세한 내용은 Message.sol 라이브러리를 참고하라.

모듈 유형 (Module Type)

ISM이 구현해야 하는 두 번째 함수는 `moduleType()`이다.

```
function moduleType() external view returns (uint8);
```

이 함수는 릴레이어에게 `_metadata`에 어떤 정보를 포함해야 하는지를 알리기 위해 사용된다.
ISM은 지원되는 모듈 유형 중 하나를 반드시 반환해야 한다.

```
enum ModuleType {
    UNUSED,
    ROUTING,
    AGGREGATION,
    LEGACY_MULTISIG,
    MERKLE_ROOT_MULTISIG,
    MESSAGE_ID_MULTISIG,
    NULL, // used with relayer carrying no metadata
    CCIP_READ,
    ARB_L2_T0_L1,
    POLYGON_POS,
    OP_STACK
}
```

- 모든 ISM 컨트랙트는 moduleType이 정의된 ISM 인터페이스를 반드시 구현해야 한다. 릴레이어는 이 moduleType 값을 기반으로 분기하여 해당 ISM에 필요한 메타데이터를 결정한다. 모듈 유형과 그 메타데이터 형식에 대한 자세한 내용은 [프로토콜](#)을 참고하라.
- 사용 가능한 모듈 유형 및 각 메타데이터에 대한 자세한 내용은 본 섹션에 포함된 ISM 문서(예: [Multisig ISM](#))를 참고하라.

경제적 보안 (Economic Security)

개요 (Overview)

하이퍼레인이 경제적 보안을 보장하는 방법

하이퍼레인은 부정 행위를 “비용이 따르는 행위”로 만드는 것을 목표로 한다.

스테이킹과 슬래싱 기능은 현재 설계 중이며 아직 구현되지 않았다.

이 페이지는 정보 제공 목적으로만 공개된 것이며, 설계가 성숙해짐에 따라 세부 내용은 변경될 수 있다.

스테이킹(staking)과 슬래싱(slashing)은 검증자 서명에 의존하는 ISM — 예를 들어 [Multisig ISM](#) 및 [Optimistic ISM](#) — 에 경제적 보안을 추가할 수 있게 한다.

하이퍼레인 검증자들은 선택적으로 하이퍼레인의 스테이킹 프로토콜에 참여할 수 있다.

이후 해당 검증자가 인터체인 메시지를 위조하거나 검열하려 시도할 경우, 그들의 스테이크 및 위임받은 스테이크가 슬래싱(벌금으로 소각)될 수 있다.

인터체인 보안 모듈(Interchain Security Modules)은 이러한 검증자들의 서명에 의존할 때, 스테이킹과 슬래싱을 통해 강화된 경제적 보안의 이점을 얻는다.

검증 가능한 사기 증명 (Verifiable Fraud Proofs)

다른 많은 인터체인 통신 프로토콜과 달리, 하이퍼레인의 슬래싱 프로토콜은 **검증 가능한 사기 증명(verifiable fraud proofs)**을 사용한다.

이는 신뢰된 제3자의 개입 없이도 하이퍼레인 프로토콜이 검증자가 위조된 체크포인트(checkpoint)에 서명했는지를 직접 검증할 수 있음을 의미한다.

이것이 가능한 이유는, 검증자들이 스테이킹한 자산이 그들이 서명으로 증명하는 상태(즉, Mailbox 머클 루트)가 존재하는 **동일한 체인**에 있기 때문이다. 슬래싱 스마트 컨트랙트는 검증자의 서명을 최신 루트와 비교하고, 복잡한 머클 트리 연산을 수행하여 검증자가 서명한 체크포인트가 위조되었는지를 확인할 수 있다.

반면, 많은 인터체인 통신 프로토콜에서는 검증자의 스테이크가 인터체인 메시지가 발생한 체인과 **다른 체인**에 존재하는 것이 일반적이다.

이 경우, 부정행위를 한 검증자의 스테이크를 슬래싱하려면 동일한 메시지 전달 프로토콜이 해당 스테이크가 존재하는 체인으로 메시지를 전달해야 한다. 문제가 보이죠? 부정행위가 발생한 동일한 검증자 세트가 그 부정행위의 증거를 전달하는 역할을 하게 되며, 이는 명백한 신뢰 리스크를 초래한다.

하이퍼레인은 이러한 가능성을 원천적으로 차단한다. 따라서 하이퍼레인 검증자들은 자신이 검증하는 원본 체인(origin chain)에 스테이크를 유지해야 한다. 이로 인해 사기 증명은 신뢰 없이도 검증이 가능하며, 슬래싱 대상이 되는 스테이크와 부정행위 기록이 동일한 환경에 존재하므로 사기 증명 절차에서 오류가 발생할 여지가 없다.

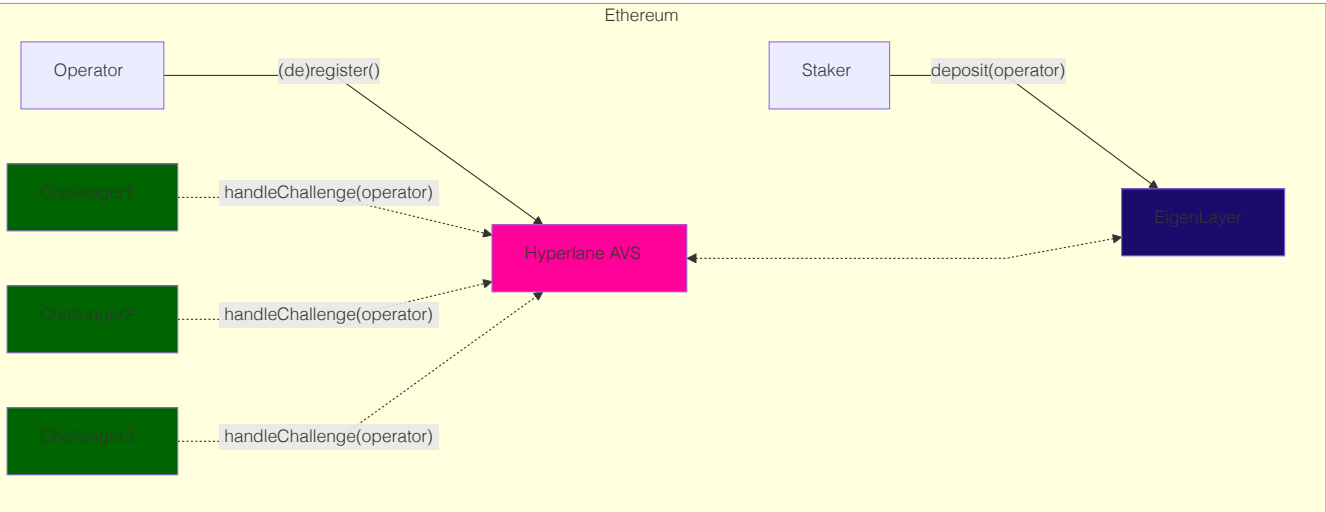
하이퍼레인 AVS (Hyperlane AVS)

EigenLayer를 기반으로 구축된 하이퍼레인의 AVS 모듈

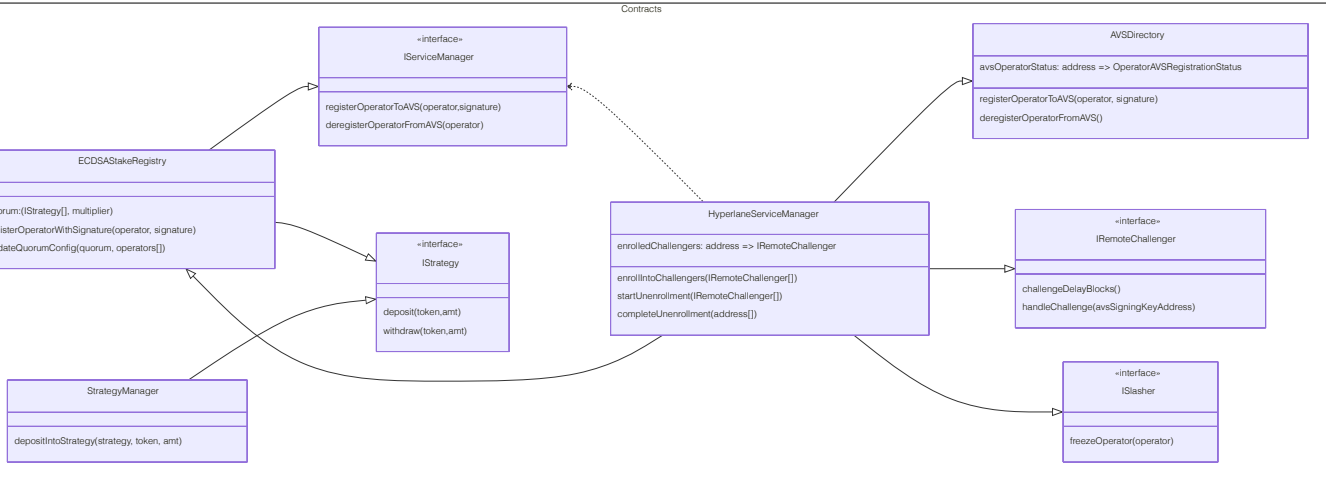
이 가이드는 이더리움 리스테이킹(restaking) 프로토콜인 EigenLayer 위에 구축된 하이퍼레인의 AVS(Actively Validated Service) 모듈에 대해 설명한다.

현재 하이퍼레인은 보안성을 유지하기 위해 신뢰할 수 있는 검증자 그룹이 선택되는 **Proof of Authority(POA)** 모델을 사용하고 있다. 그러나 검증자가 악의적이거나 실수로 잘못된 서명을 하더라도, 그들은 경제적 이해관계를 보유하지 않기 때문에 아무런 불이익을 받지 않는다.

AVS 모듈은 바로 이러한 격차를 해소하기 위해 설계되었다.



아키텍처 (Architecture)



위의 클래스 다이어그램은 현재 AVS 모듈 컨트랙트의 아키텍처를 설명한다. 특히, `IRemoteChallenger` 인터페이스를 통해 지원되는 "metaAVS" 디자인 패턴을 보여준다.

IRemoteChallenger

하이퍼레인에서 운영자(Operators)는 자신이 선택한 체인(들)에서 자유롭게 운영할 수 있다. 이로 인해 AVS 모듈은 어떤 체인이나 챌린저(challenger)에도 적응할 수 있도록 설계되었다. 그러나 이러한 유연성은 메인넷(Mainnet)에서 슬래싱에 대한 ‘보편적으로 합의된 진실의 원천 (canonical source of truth)’이 존재하지 않을 수 있음을 의미한다. 이는 사기 행위(fraud)가 원본 체인에서만 증명될 수 있기 때문에, 그 정보를 스테이크가 존재하는 체인으로 전달할 수 있는 메커니즘이 필요하기 때문이다.

하이퍼레인은 **비허가형 상호운용성(permissionless interoperability)**의 철학을 따르며, 애플리케이션 개발자가 자신만의 진실 원천(source of truth)을 정의하고 자신의 애플리케이션에 대한 경제적 보안 보장을 직접 설정할 수 있어야 한다고 믿는다. 이에겐 챌린지 조건 및 슬래싱 윈도우를 프로그래밍할 수 있는 기능이 포함된다.

이러한 수준의 커스터마이징을 가능하게 하기 위해

IRemoteChallenger 인터페이스가 도입되었다.

handleChallenge 함수는 사기 서명이 탐지된 체인에서

L1로 전달된 `ism.verify()`가 성공적으로 실행된 경우에만 호출된다.

위에서 언급한 바와 같이, 이러한 기반 ISM에 대해 강력한 보안 설정을 구성하는 것이 중요하다.

```
interface IRemoteChallenger {
    /// @notice Returns the number of blocks that must be mined before a
    challenge can be handled
    /// @return The number of blocks that must be mined before a
    challenge can be handled
    function challengeDelayBlocks() external view returns (uint256);

    /// @notice Handles a challenge for an operator
    /// @param operator The address of the operator
    function handleChallenge(address operator) external;
}
```

이 구성은 변경 불가능(immutable)해야 하며,

모든 AVS 운영자가 이를 조회하고 자유롭게 참여(opt-in)할 수 있어야 한다.

handleChallenge 함수는 예를 들어 Arbitrum과 같은 원본 체인에서

챌린지가 어떻게 이더리움 메인넷(AVS 컨트랙트와 IRemoteChallenger가 존재하는 곳)으로 전달되는지를

명시적으로 정의한다.

Arbitrum의 경우, 이는 롤업의 네이티브 브리지(bridge)를 통해 이루어질 수 있으며,

다른 L1의 경우 위원회(committee) 기반의 솔루션을 사용할 수도 있다.

우리의 인터페이스는 챌린저 구현에 구애받지 않도록 설계되어 있으며,

이로 인해 다양한 설계 환경에서 유연하게 적용할 수 있다.

이 접근 방식은 L1에서 `ism.verify()`가 성공적으로 실행된 후

`handleChallenge()`를 호출함으로써 기존의 hook-ISM 구성을 재활용할 수 있도록 한다.

롤업의 네이티브 브리지를 사용하는 경우,

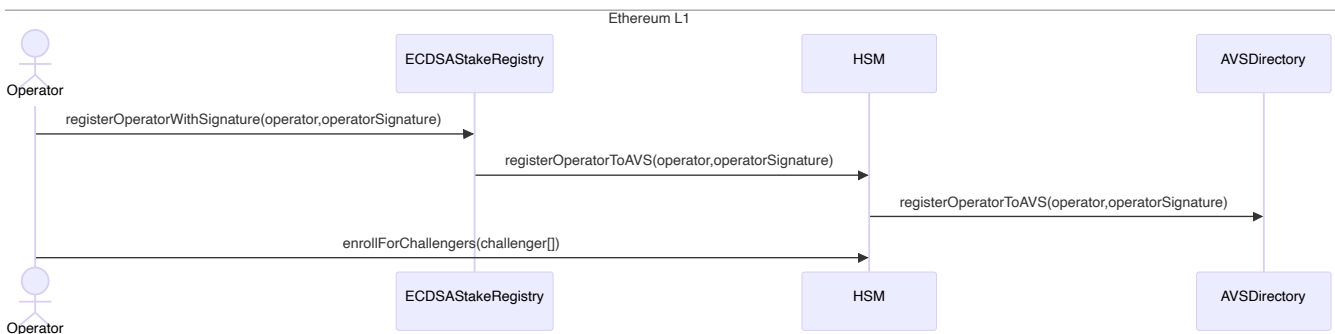
합리적인 `challengeDelayBlocks` 값은 약 1주일보다 약간 긴 기간으로 설정하는 것이 적절하다.

운영자는 여러 IRemoteChallenger 컨트랙트를 검토하여
각각의 리스크와 보상 구조를 평가해야 한다.
관심이 있는 경우, HyperlaneServiceManager 컨트랙트를 통해
하나 이상의 챌린저에 직접 등록(enroll)할 수 있다.

IRemoteChallenger 구현체는 아직 프로덕션 환경에서 활성화되어 있지 않다.

등록 워크플로 (Workflow for Registering)

사전 조건: 운영자는 CLI를 통해 EigenLayer 운영자로 등록되어 있어야 한다.



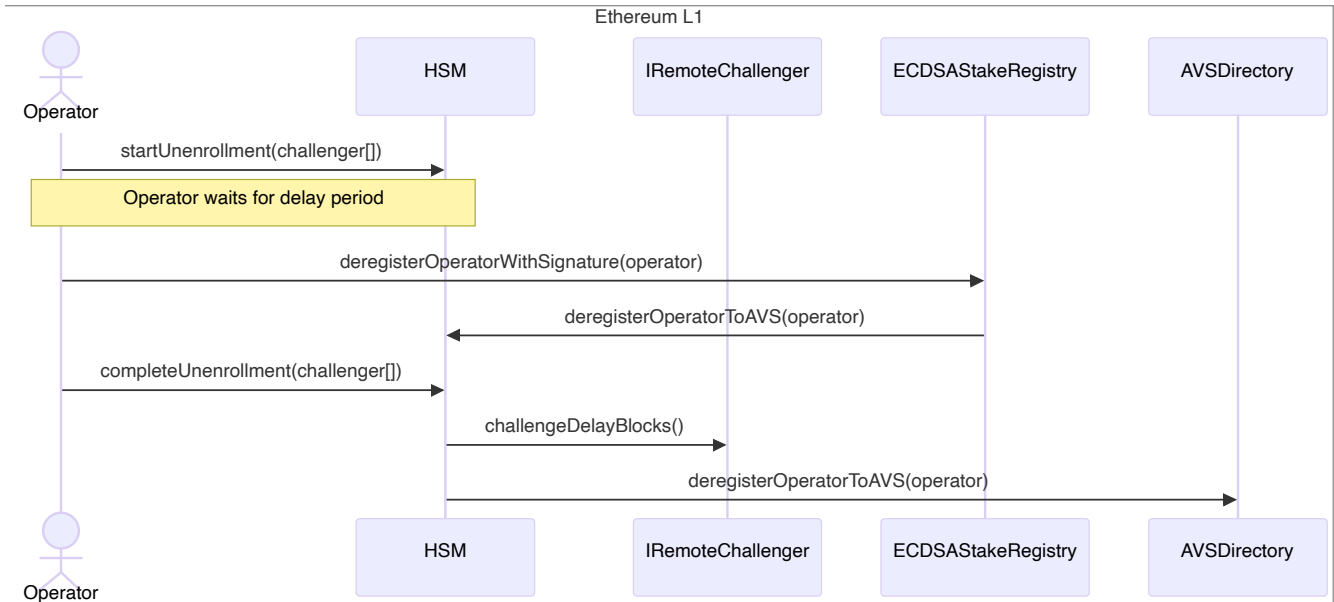
운영자는 퍼미션리스 슬래싱(permissionless slashing)을 통한 경제적 보안을 확보하기 위해
특정 챌린저(Challenger)에 등록(enroll)해야 한다.

운영자는 각 리모트 챌린저(Remote Challenger)를 직접 검토하고,
위험 수준에 따라 자신이 참여하고자 하는 챌린저를 자유롭게 선택할 수 있다.

이는 또한 중앙화된 승인(permissioning)이나 화이트리스트(whitelisting) 절차가 없음을 의미하며,
따라서 서로 다른 롤업 스택에 새로운 챌린저를 추가하는 과정에서
병목 현상이 발생하지 않는다.

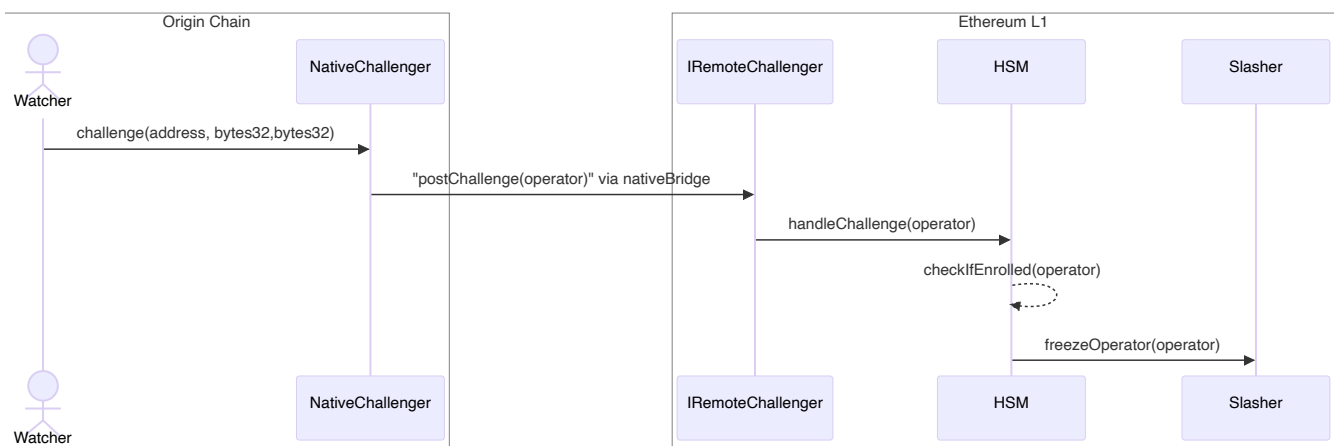
운영자는 등록 해제 지연 블록 수(unenrollment delay blocks)가 지난 후
등록을 해제(unenroll)할 수 있다.

등록 해제 워크플로 (Workflow for Deregistering)



운영자는 자신이 등록된 모든 챌린저에서 등록 해제(unenroll)를 완료한 이후에만 운영자 등록을 해제(deregister)할 수 있다. 각 챌린저는 고유한 지연 기간(delay period)을 가질 수 있으며, 이는 운영자가 어떤 챌린저에서도 챌린지 기간이 끝나기 전에 스테이크를 인출하지 못하도록 보장하기 위함이다.

스테이킹 워크플로 (Workflow for Staking)

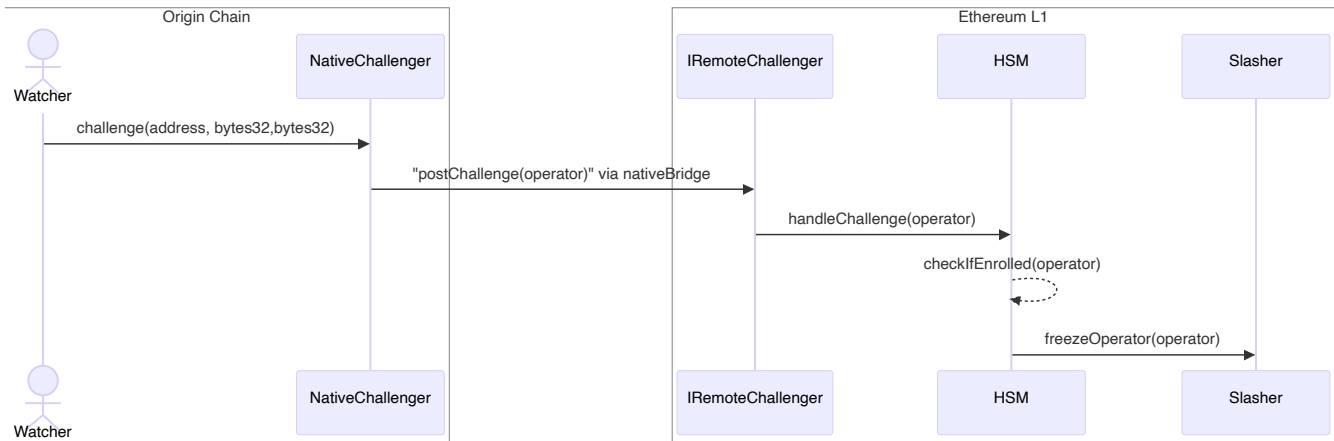


EigenLayer의 설계에서는 슬래싱이 활성화되기 전까지 스테이커의 자금이 AVS에 직접 접근되지 않도록 보호된다. 이를 통해 자금이 조기 위험에 노출되는 일이 발생하지 않는다.

또한 이 설계는 사용자가 하이퍼레인 AVS만 개별적으로 선택하여 참여(opt-in)하는 것을 허용하지 않는다.

스테이킹은 **StrategyManager**를 통해 관리되며, 특정 운영자(operator)를 대상으로 위임(delegate)되는 구조로 작동한다.

슬래싱 워크플로 (Workflow for Slashing)



슬래싱 메커니즘은 퍼미션리스(permissionless)하고 유연하게 설계되어 있다.
IRemoteChallenger 인터페이스는 다양한 형태의 슬래싱 메커니즘 구현을 허용한다.

예를 들어, 원본 체인(origin chain)의 네이티브 챌린저(native challenger)가 postChallenge 함수를 호출하면,
이는 HyperlaneServiceManager 컨트랙트의 handleChallenge 함수를 호출하게 된다.
HyperlaneServiceManager 컨트랙트는 해당 운영자가 챌린저에 등록되어 있는지를 확인한 후,
Slasher 컨트랙트의 freezeOperator 함수를 호출하여
운영자의 스테이크를 동결(freeze)한다.

챌린저(Challenger) 및 슬래셔(Slasher) 컨트랙트는 아직 프로덕션 환경에서 활성화되지 않았다.
이 시퀀스 다이어그램은 최종 구현에 따라 변경될 수 있다.

컨트랙트 배포 (Contract Deployment)

Ethereum

이름	프록시(Proxy)	구현(Implementation)
ECDSAStakeRegistry	0x272CF0BB70D3B4f79414E0823B426d2EaFd48910	0xa11b1a385287b5167ecca6a38f63abeab981d589
HyperlaneServiceManager	0xe8E59c6C8B56F2c178f63BCFC4ce5e5e2359c8fc	0x3127e69517ec2268dd5745c97194d387a4fc0c45

Holesky

이름	프록시(Proxy)	구현(Implementation)
ECDSAStakeRegistry	0xFfa913705484C9BAea32Ffe9945BeA099A1DFF72	0x628bc518ed1e0e8c6cbcd574eba0ee29e7f6943e
HyperlaneServiceManager	0xc76E477437065093D353b7d56c81ff54D167B0Ab	0xa3ab7e6ce24e6293bd5320a53329ef2f4de73fca

하이퍼레인 에이전트 (Hyperlane Agents)

개요 (Overview)

하이퍼레인 프로토콜은 크로스체인 메시지 전달의 **전송 계층(transport layer)**과 **보안 계층(security layer)**을 분리한다. 배포를 실행하기 위해 하이퍼레인은 온체인 활동을 관찰하고, 프로토콜의 전송 또는 보안 측면을 수행하는 오프체인 에이전트에 의존한다.

이러한 에이전트들은 **Rust**로 구현되어 있으며, **Docker 이미지 및 바이너리 형태**로 배포된다.

1. **릴레이어(Relayers)**는 프로토콜의 메시지 전송을 담당한다. 이들은 [IInterchainSecurityModule 인터페이스](#)에 필요한 오프체인 보안 메타데이터를 집계하고, 메시지를 수신자에게 전달한다.
2. **검증자(Validators)**는 [Multisig ISM](#) 또는 [하이퍼레인 AVS](#)의 일부로서 프로토콜의 보안 계층을 담당한다. 이들은 [Mailbox](#) 메시지의 유효성을 검증하고, 서명을 릴레이어가 사용할 수 있도록 공개한다.

검증자 (Validators)

검증자는 하이퍼레인 프로토콜의 보안 계층을 구성하는 핵심 요소로서, [Multisig ISM](#) 또는 [경제적 보안 모듈\(Economic Security Module\)](#)의 일부로 작동한다. 다른 유형의 ISM을 사용하는 경우 검증자는 필요하지 않다.

검증자는 [Mailbox](#)에서 새 메시지가 디스패치될 때마다 머클 루트(또는 체크포인트)에 서명함으로써 메시지를 인증(attest)한다. 이 서명은 누구나(일반적으로 릴레이어)가 조회하여 메시지와 함께 목적지 체인으로 제출할 수 있다.

하이퍼레인은 다른 많은 프로토콜과 달리, **고정된 검증자 세트(enshrined validator set)**를 두지 않는다. 수신 컨트랙트가 자신을 포함한 [Multisig ISM](#)을 지정하기만 하면, 누구나 자유롭게 자신의 검증자를 운영할 수 있다.

검증자의 체크포인트는 모두 공개된다. 이는 누구나 서명을 가져와 `Mailbox.process()`를 호출하여 메시지를 전달할 수 있도록 하여, 전송 계층이 **퍼미션리스(permissionless)** 하게 유지되도록 보장한다. 릴레이어는 단지 메시지 송신자를 대신해 편의 서비스를 제공할 뿐이다.

검증자는 프로토콜의 안전성을 위해 반드시 **최종화된 체크포인트(finalized checkpoint)**에만 서명해야 한다.

리오그(reorg)로 인해 무효화될 수 있는 메시지에 서명할 경우, 경제적 보안 모듈에서 슬래싱이 발생한다.

검증자 에이전트는 단일 체인에 연결되어 새 메시지를 모니터링하며, 여러 체인을 검증하려면 여러 인스턴스를 실행해야 한다.

하이퍼레인은 누구나 손쉽게 실행할 수 있는 Rust 기반의 바이너리 형태로 검증자 프로그램을 개발 중이다. 운영 측면에서 검증자는 이 바이너리를 실행하고, 검증 대상 체인에 대한 RPC 공급자 또는 노드 접근 권한을 보유해야 한다. 하이퍼레인은 각 체인의 기존 노드 운영자 커뮤니티가 대부분의 검증자로 참여하기를 기대하고 있다.

검증자를 운영하고 싶은가? [검증자 가이드](#)를 참고하라.

자세한 설명 (In more detail)

검증자(Validator) 에이전트는 두 종류의 키를 사용한다. 하나는 자신이 검증 중인 블록체인에서 트랜잭션에 서명하기 위한 키이며, 다른 하나는 이더리움과 호환되는 ECDSA 곡선에서 체크포인트에 서명하기 위한 키다. 체크포인트는 공개되므로, 검증자가 처음 가동될 때는 온체인 announce 트랜잭션을 통해 자신의 체크포인트 위치를 공표해야 한다. 이는 검증자 에이전트가 제출하는 유일한 트랜잭션이며, 그 외에는 블록체인별 트랜잭션 서명 키를 전혀 사용하지 않는다. 또한 블록체인별 키를 설정하지 않기 위해 새로운 검증자를 수동으로 공표하는 옵션도 있다. 검증자는 누구나 공표할 수 있다.

[Multisig ISM](#)을 사용하는 메시지의 경우, 검증자는 `MerkleTreeHook.latestCheckpoint()`를 호출하여 현재 머클 루트를 읽는다. 검증자는 `MerkleTreeHook.latestCheckpoint()` 함수를 사용해 새로운 트랜잭션을 인덱싱해야 하는 시점을 판단한다. 이 폴링 메커니즘 덕분에 검증자는 전체 트리를 백필(backfill)할 필요 없이 새 메시지에 대한 서명을 바로 시작할 수 있다.

Multisig ISM 선행 조건 (Prerequisites)

원본 체인에서 전송된 메시지를 검증하려면, 목적지 체인의 수신자가 지정한 [Multisig ISM](#)에 검증자의 체크포인트 서명 키가 등록되어 있어야 한다. 이는 ISM 컨트랙트가 배포될 때만 가능하지만, 어떤 구현의 [StaticThresholdAddressSetFactory](#)에서 `deploy(validatorAddresses, threshold)` 트랜잭션을 보내는 간단한 방식으로 처리할 수 있다.

AVS 운영자 선행 조건 (AVS Operator Prerequisites)

EigenLayer AVS에서 검증자를 운영하며 [경제적 보안](#)을 제공하려면, 검증자는 AVS 운영자로 등록해야 한다. 자세한 내용은 [AVS 운영자 가이드](#)를 참고하라.

아키텍처 (Architecture)

검증자는 두 가지 주요 구성 요소로 이루어진다. 원본 체인을 위한 인덱서(Indexer)와 서명된 머클 루트를 게시하는 체크포인트 서브미터(Checkpoint Submitter)다.

인덱서 (The Indexer)

검증자가 인덱싱하는 이벤트 타입은 [Mailbox](#)가 새 메시지를 디스패치할 때마다 발생하는 머클 트리 훅 (Merkle Tree Hook) [컨트랙트](#)의 `InsertedIntoTree` 단 하나다. 이 과정은 RPC를 통해 이루어지며, 에이전트는 삽입 이벤트를 기반으로 메모리 내 머클 트리를 구성한다. 이 이벤트들은 빠른 시작을 위해 로컬 RocksDB에도 캐시된다. 새 메시지가 디스패치될 때마다 체크포인트 서브미터는 메모리 내 트리에서 체크포인트를 생성하고, 이를 처리 큐에 추가한다.

체크포인트 서브미터 (The Checkpoint Submitter)

체크포인트 서브미터는 ECDSA 키를 사용하여 체크포인트에 서명하고, 이를 고가용성 스토리지에 게시한다. 체크포인트의 스키마는 다음과 같다:

```
pub struct Checkpoint {  
    /// The merkle tree hook address  
    pub merkle_tree_hook_address: H256,  
    /// The mailbox / merkle tree hook domain  
    pub mailbox_domain: u32,  
    /// The checkpointed root  
    pub root: H256,  
    /// The index of the checkpoint  
    pub index: u32,  
}
```

현재 에이전트 구현이 지원하는 체크포인트 스토리지는 AWS S3와 Google Cloud Storage다. Azure Blob Storage 기여는 아직 [진행 중](#)이다.

라이브니스 영향 (Liveness implications)

Multisig ISM에 설정된 임계값(threshold)에 따라 검증자(Validator)의 다운타임은 프로토콜의 라이브니스를 중단시킬 수 있다. 서브미터(submitter)는 오래된 체크포인트보다 새로운 체크포인트를 우선시하여, 릴레이어가 아직 전달되지 않았을 가능성이 더 높은 최신 메시지의 메타데이터를 먼저 가져올 수 있도록 한다.

기본 ISM 검증자 (Default ISM Validators)

검증자 수는 인터체인 보안 모듈(ISM) 구성에 따라 달라진다. 기본 ISM은 특정 검증자 세트를 사용하며, 임계값, 운영자, 주소 등 검증자 세부 정보는 [여기](#)에서 확인할 수 있다.

릴레이어 (Relayer)

릴레이어는 원본 체인에서 목적지 체인으로 메시지를 전달함으로써 하이퍼레인 프로토콜의 전송 계층을 운영한다. 수신자가 지정한 [인터체인 보안 모듈\(ISM\)](#)에 따라 [Multisig ISM](#)의 검증자 서명, 머클 증명, 영지식 증명 등 메시지의 보안을 입증하기 위한 메타데이터가 필요할 수 있으므로, 릴레이어는 전달을 시도하기 전에 해당 ISM에 특화된 메타데이터를 수집한다.

릴레이어는 메타데이터 수집과 메시지 제출을 주기적으로 재시도한다. 이는 전달 비용 지불 미이행, 릴레이어 잔액 부족, 목적지 체인의 일시적 수수료 급등, 검증자 다운타임과 같은 장애 상황에 대한 복원력을 높이기 위해 중요하다.

릴레이어 인센티브 (Relayer Incentives)

메시지 송신자는 원본 체인에서 전달 비용을 지불하며, 릴레이어 운영자는 목적지 체인 계정으로 수익을 재균형(rebalance)하여 전달 트랜잭션 가스비가 항상 지불될 수 있도록 해야 한다. 릴레이어는 프로토콜로부터 직접적인 토큰 인센티브를 받지 않지만, 운영자는 핵심 서비스를 제공하는 대가로 자체 수수료 구조를 구성할 수 있다. 자세한 내용은 [인터체인 가스 지불\(IGP\)](#)를 참고하라.

릴레이어 운영 (Operating a Relayer)

릴레이어는 처리할 메시지를 쉽게 구성할 수 있다. 현재 릴레이어는 다음을 지원한다:

1. 송신자/수신자 화이트리스트
2. 송신자/수신자 블랙리스트
3. 목적지 체인 메시지 처리를 위해 원본 체인에서의 [지불 수락](#) 기능

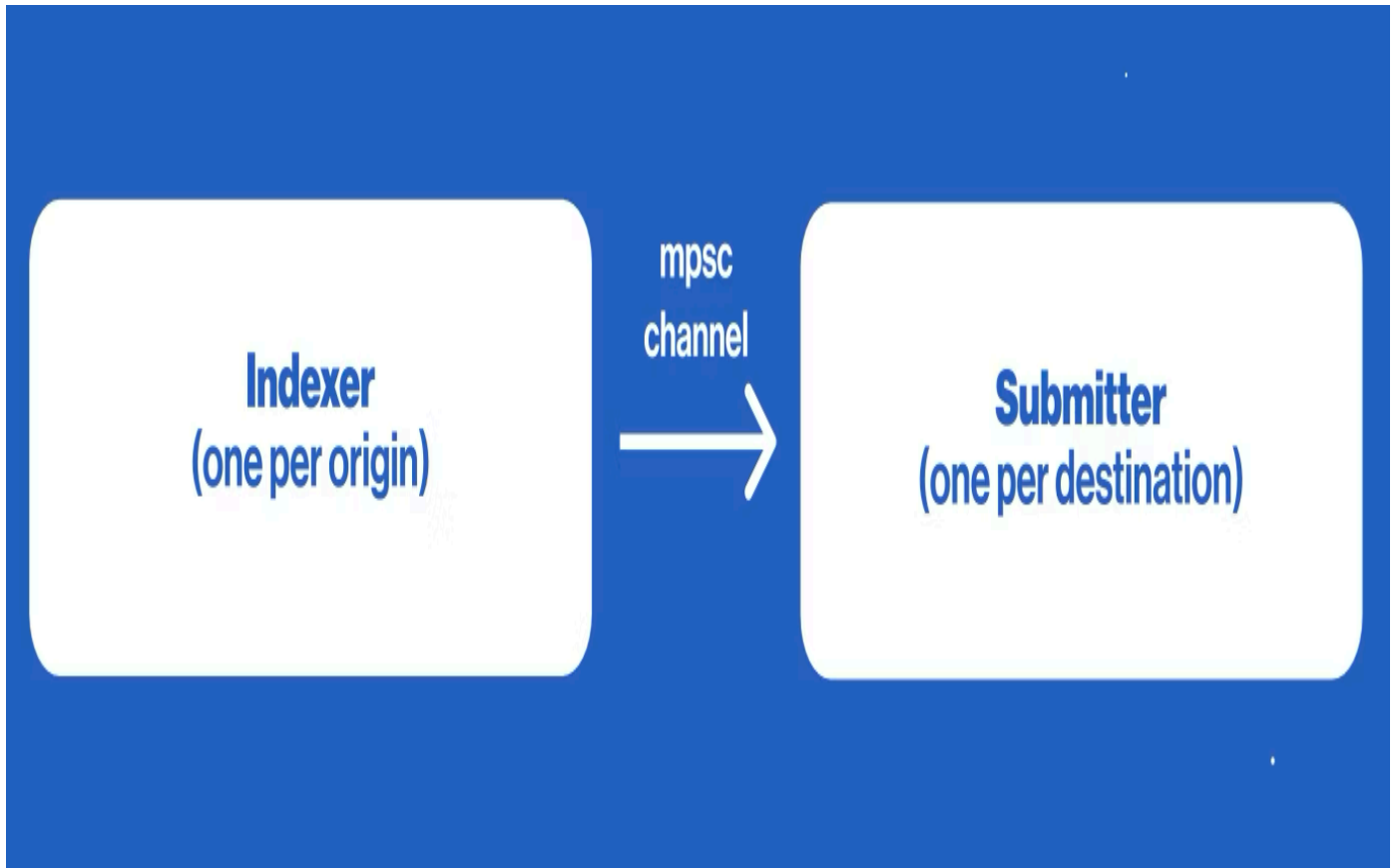
퍼미션리스하게 누구나 운영할 수 있지만, 어떤 릴레이어에게 전달 비용을 지불할지는 메시지 송신자가 선택한다. [Mailbox](#) 통합은 보통 동일한 릴레이어에게 항상 지불하도록 기본 설정되어 있으나, 송신자는 이를 오프트아웃할 수 있다. 편의를 위해 하이퍼레인은 Rust 바이너리로 구현된 오픈소스이자 설정 가능한 릴레이어 에이전트를 운영하며, 직접 운영하려면 [여기](#)에서 오픈소스 바이너리를 확인할 수 있다.

자체 릴레이어를 운영하려면 [IGP 컨트랙트](#)를 배포하고 그 안에서 토큰 환율과 가스 가격을 유지하여 메시지 전달 수수료를 정확히 청구해야 한다.

릴레이어를 운영하고 싶은가? [Relayer 가이드](#)를 따르라.

자세한 설명 (In more detail)

릴레이어는 원본(origin)마다 인덱서(Indexer), 목적지(destination)마다 서브미터(Submitter)의 두 가지 주요 작업으로 구성된다.



인덱서 (The Indexer)

새로운 메시지와 과거 메시지는 모두 RPC를 사용해 Mailbox 컨트랙트 이벤트를 쿼리함으로써 인덱싱된다. 또한 가스 지불도 인덱싱하여 송신자가 전달 비용을 지불했는지 확인하고, 일부 ISM은 추가적인 원본 체인 데이터가 필요하다. 예컨대 Multisig ISM은 어떤 검증자 서명이 어떤 메시지에 대응하는지 릴레이어에게 알리기 위해 Merkle Tree Hook 컨트랙트에 의존한다. 이벤트 유형(메시지, IGP, 머클 트리 삽입)마다 독립적인 인덱싱 태스크가 생성된다.

인덱서는 캐시 및 서브미터 태스크와의 통신 수단으로 로컬 데이터베이스(RocksDB)에 기록하며, 서브미터는 주기적으로 데이터베이스를 폴링해 새로운 메시지 디스패치가 있었는지 확인한다.

릴레이어는 오래된 메시지보다 새로운 메시지를 우선한다. 이는 오래된 메시지가 이미 전달되었을 가능성이 높다는 가정에 기반한다. 따라서 (재)시작 시 두 개의 포인터가 생성되는데, 블록체인 팁에서 시작하는 전방 포인터와 후방 포인터이며, 둘 다 메시지의 단조 증가하는 `nonce` 필드를 사용해 순회한다. 다른 이벤트 인덱싱에도 유사한 접근을 사용하지만 일부는 시퀀스 번호가 없어 잘못된 RPC 응답을 받으면 누락될 수 있다(이는 빈번히 발생한다). 이런 이유로 메시지 인덱싱 태스크는 이벤트를 발견한 트랜잭션 ID를 다른 인덱싱 태스크로 전달하여, 해당 이벤트가 `Mailbox.dispatch()` 호출과 동일한 트랜잭션 내에서 발생한 한 어떤 하이퍼레인 이벤트도 놓치지 않도록 보장한다.

서브미터 (The Submitter)

메시지는 아래에 설명된 네 개의 독립적인 태스크 단계를 거치며 서브미터에서 처리된다.

서브미터 태스크

설명

Message Processor

로컬 데이터베이스를 폴링(polling)하여 아직 전달되지 않은 메시지가 있는지 확인하고, 이를 Prepare 단계의 큐로 보낸다.

Prepare Task

큐에서 메시지를 꺼내 가스비가 지불되었는지 확인하고, 필요한 메타데이터를 가져오며, 메시지 전달 트랜잭션을 시뮬레이션한다. 시뮬레이션이 성공하면 메시지를 Submit 단계의 큐로 보내고, 실패하면 다시 Prepare 큐로 되돌린다.

Submit Task

큐에서 메시지를 꺼내 온체인으로 전달 트랜잭션을 보낸다. 가능한 경우 전달 트랜잭션은 배치(batch) 처리된다. 브로드캐스트 오류가 없으면 메시지를 Confirm 단계의 큐로 보내고, 오류가 있으면 메시지를 다시 Prepare 큐로 되돌린다.

Confirm Task

트랜잭션 최종화를 기다리며, 체인 리오그(reorg)가 발생하거나 트랜잭션이 되돌려질 경우 메시지를 다시 Prepare 큐로 되돌린다.

메시지는 위에서 언급한 메타데이터와 함께 목적지 체인에서 `Mailbox.process()`를 호출함으로써 수신자에게 전달된다.

메시지의 재시도 횟수(retry count)는 지수형 백오프(exponential backoff) 전략에 따라 다음 재전송 시점을 결정한다. 현재 릴레이어가 메시지 처리를 중단하는 고정된 최대 재시도 횟수는 존재하지 않는다. 그러나 이는 무한 재시도를 보장하지 않으며, 운영자는 이를 서비스 수준 계약(SLA)으로 간주해서는 안 된다.

재시도 전략 (Retry Strategy)

전달 시도가 실패하면 릴레이어는 적응형 백오프(adaptive backoff) 전략을 사용하여 재시도를 수행한다. 이 전략은 최종성(finality)이나 처리 시간이 더 긴 ISM을 지원하도록 최적화되어 있으며, 일시적인 문제로부터 복구할 수 있을 만큼 충분히 긴 기간 동안 재시도가 지속되도록 설계되어 있다.

재시도 횟수 범위	재시도 간 지연 시간
1	5초
2	10초
3	30초
4	1분
5-24	3분
25-39	5분에서 시작, 재시도마다 1.5분씩 증가
40-44	30분
45-49	1시간
50+	점진적으로 몇 시간에서 며칠 단위로 증가

구현 내용은 [여기](#)에서 확인할 수 있다.

참고: 위의 값들은 대략적인 수치이며 변경될 수 있다.

HYPER 및 프로토콜 경제 (HYPER and Protocol Economics)

하이퍼레인 프로토콜 경제 (Hyperlane Protocol Economics)

하이퍼레인은 크로스체인 메시지 전송을 가능하게 하는 퍼미션리스 상호운용성(interoperability) 프로토콜이다.

HYPER는 하이퍼레인 생태계 내 인센티브를 조정하고 커뮤니티를 강화하기 위해 도입된 네이티브 토큰이다.

이 토큰은 스테이킹을 통해 크로스체인 통신의 보안을 확보하고, 프로토콜 활동에 기여하는 네트워크 참여자들에게 보상을 제공한다.

HYPER 토큰 개요 (HYPER Token Overview)

HYPER

HYPER는 하이퍼레인 프로토콜의 네이티브 토큰이다. 이 토큰은 Ethereum 메인넷, Base, OP Mainnet, Arbitrum One, BSC 등 여러 네트워크에 존재한다. HYPER는 스테이킹을 통해 하이퍼레인 프로토콜의 보안을 강화하는 데 사용된다. 검증자(Validator)는 HYPER 스테이킹을 기반으로 기본 메시지 보안을 제공하도록 선택된다.

토큰	네트워크	컨트랙트 주소
HYPER	Ethereum Mainnet	0x93A2Db22B7c736B341C32Ff666307F4a9ED910F5
HYPER	Arbitrum One	0xC9d23ED2ADB0f551369946BD377f8644cE1ca5c4
HYPER	Base	0xC9d23ED2ADB0f551369946BD377f8644cE1ca5c4
HYPER	OP Mainnet	0x9923DB8d7FBAC2E69E87fAd19b886C81cd74979
HYPER	BSC	0xC9d23ED2ADB0f551369946BD377f8644cE1ca5c4

stHYPER

stHYPER는 사용자가 [Symbiotic Vault](#)에 HYPER를 예치(lock)할 때 발행되는 유동 스테이킹 토큰 (liquid staking token)이다. stHYPER는 Ethereum Mainnet과 BSC에서 사용 가능하다. 이 토큰은 사용자의 스테이킹된 HYPER를 나타내는 영수증 역할을 하며, stHYPER가 존재하는 네트워크에서는 보상을 청구할 수 있다.

토큰	네트워크	컨트랙트 주소
stHYPER	Ethereum Mainnet	0xE1F23869776c82f691d9Cb34597Ab1830Fb0De58
stHYPER	BSC	0x6E9804a08092D8ba4E69DaCF422Df12459F2599E

HYPER vs. stHYPER

토큰	설명	존재 네트워크	보상
HYPER	하이퍼레인 프로토콜의 네이티브 토큰	Ethereum, Base, OP Mainnet, Arbitrum One, BSC	-
stHYPER	Symbiotic Vault에서 발행되는 유동 스테이킹 토큰	Ethereum, BSC	 예

스테이킹 (Staking)

스테이킹은 참여자들이 하이퍼레인 프로토콜의 경제적 보안을 강화하고 보상을 얻는 방법이다. 사용자는 Ethereum에서 Symbiotic Vault에 HYPER를 예치함으로써 스테이킹을 수행하며, 이 과정에서 자신의 스테이킹 포지션을 나타내는 유동 스테이킹 토큰인 stHYPER가 생성된다.

Symbiotic의 HYPER Vault: HYPER Vault는 다른 프로토콜이나 네트워크로 리스테이킹 (restake)하지 않는다. 오직 Symbiotic 상의 하이퍼레인 네트워크로만 위임되며, 이로써 외부 슬래싱 위험을 최소화한다.

스테이킹 작동 방식 (How Staking Works)

- **스테이킹 플로우 (Staking Flow):** HYPER 보유자는 Symbiotic 앱에서 HYPER Vault를 선택하고 HYPER 토큰을 예치(lock)하여 스테이킹을 시작할 수 있다.
- **에포크 길이 (Epoch Length):** 보상은 알고리즘적으로 계산되어 매 에포크마다 자동으로 분배된다. 현재 에포크는 약 1개월로 설정되어 있으며, 이는 기본 인터체인 보안 모듈(ISM) 검증자 세트 체인에서의 최대 정산 지연을 반영한다.
- **언스테이킹 플로우 (Unstaking Flow):** 언스테이킹을 진행하려면 아래 단계를 따라야 한다.
 1. Symbiotic 사용자 인터페이스에서 언스테이킹(unstake) 액션을 시작한다.
 2. 에포크 시작 및 종료 시점은 프로토콜에서 정의되므로, 언스테이킹을 시작한 후 인출이 가능해질 때까지 약 한 에포크(약 30~60일, 최대 약 0.99 + 1 에포크)를 기다려야 한다.
 3. 에포크 종료 후 UI의 스테이크 컨트랙트에서 수동으로 출금을 완료한다.

경제적 보안 할당 (Economic Security Allocation)

Symbiotic의 HYPER Vault에 예치된 HYPER는 경제적 보안 할당 알고리즘(Economic Security Allocation Algorithm)에 따라 검증자(Validators)에게 위임된다. 이 알고리즘은 각 도메인(체인)의 경제 활동에 비례하여 경제적 보안을 분배하는 것을 목표로 한다. 즉, 경제 활동이 활발한 도메인은 그만큼 더 많은 HYPER 스테이킹이 해당 도메인의 검증자 세트에 위임된다.

이 검증자 세트는 각 도메인의 기본 인터체인 보안 모듈(Interchain Security Module, ISM)을 보호하는 역할을 한다. ISM은 목적지 체인에서 전달되는 메시지가 실제로 원본 체인에서 전송된 것임을 검증하는 스마트 컨트랙트 모듈이다. ISM 내 검증자들은 메시지가 처리되기 전에 그 유효성과 신뢰성을 검증한다.

이 할당 알고리즘은 도메인별 할당 비율(allocation percentage)을 산출하며, 이는 총 스테이킹된 HYPER를 각 도메인의 검증자 세트에 분배하는 데 사용된다. *이 과정에 대한 구체적인 내용은 추후 공개될 예정이다.*

하이퍼레인 검증자 선정 및 등록 (Hyperlane Validator Selection and Registration)

하이퍼레인의 ISM 모델은 완전한 보안 커스터마이징을 가능하게 하지만, 각 Mailbox에는 기본 ISM(Default ISM)이 존재한다. 이 ISM에 선택된 검증자(그리고 해당 검증자만)가 검증자 보상(Validator Rewards)을 받게 된다.

검증자 선정 과정 (Selection Process)

예비 검증자(또는 Symbiotic 네트워크 내 운영자, Operator)는 [Hyperlane Validator 설문조사](#)를 완료해야 한다. 하이퍼레인 프로토콜의 검증자는 특정 원본 체인의 메시지를 인증(attest)하는 역할을 담당한다. 따라서 지원자는 운영 관행 및 업타임(uptime) 등 여러 가중 기준에 따라 평가된다.

평가 기준에는 다음 항목들이 포함되며, 이에 국한되지 않는다:

- **운영 모범 사례 (Operating Best Practices)**
 - 체인 노드 운영 (예: RPC 설정 및 관리 등)
 - 운영 보안 관행 (모니터링/알림, 비밀 관리, 엔지니어 온콜 로테이션 등)
- **운영 이력 (Operating History)**
 - 과거 업타임 기록
 - 전체 운영 이력
 - 운영 중인 하이퍼레인 검증자 수
- **경제적 보안 (Economic Security)**
 - 하이퍼레인 Symbiotic Vault 내 HYPER 자가 스테이킹 비율

검증자는 운영 이력 및 성과를 기준으로 지속적으로 평가되며, 이 결과는 기본 ISM 세트 내 위치를 유지할 수 있는지 여부를 결정한다.

참고: 검증자는 하이퍼레인의 기본 ISM 세트에 선정되어야만 보상을 받을 수 있다.

Symbiotic 운영자 등록 및 옵트인 (Registering as a Symbiotic Operator and Opting In)

기본 ISM에 선정된 후 검증자는 Symbiotic에서 운영자(Operator)로 등록해야 위임(Stake)과 보상을 받을 수 있다.

운영자 등록 (Operator Registration)

Symbiotic의 [CLI](#)를 통해 운영자로 등록하는 단계는 Symbiotic 문서를 참고하라:

- [운영자 등록 가이드](#)
- [운영자용 CLI 명령어 가이드](#)

메타데이터 제출 (Metadata Submission)

등록을 완료한 후, [Symbiotic 메인넷 메타데이터 저장소](#)에 운영자/네트워크 정보 및 아이콘을 포함한 Pull Request를 제출해야 한다. 이때 등록 시 사용한 주소와 동일한 주소를 사용해야 한다.

메타데이터는 Symbiotic 문서에 명시된 여러 요구사항을 충족해야 한다. 요구사항을 충족하지 못하는 Pull Request는 승인될 수 없다.

메타데이터는 이후 수정할 수 있지만, 변경 사항은 Symbiotic 팀의 검토 후에만 적용된다.

하이퍼레인 네트워크 및 Vault 옵트인 (Opting into Hyperlane's Network and Vault)

운영자가 위임 및 보상을 받기 전에 다음 두 가지 옵트인 절차를 완료해야 한다:

1. Symbiotic에서 [하이퍼레인 네트워크](#)에 옵트인:
 1. [CLI 옵트인 명령어](#)를 사용한다.
 2. [CLI 검증 명령어](#)를 통해 옵트인을 확인한다.
2. Symbiotic에서 [HYPER Vault](#)에 옵트인:
 1. [CLI 옵트인 명령어](#)를 사용한다.
 2. [CLI 검증 명령어](#)를 통해 옵트인을 확인한다.

하이퍼레인 기본 ISM 참여 (Joining Hyperlane's Default ISM)

검증자가 Hyperlane Validator 설문을 제출하고 Symbiotic 운영자 등록 및 옵트인을 완료하면, 기본 ISM 검증자 포지션이 열릴 때 선택될 수 있다. 기본 ISM 포지션은 다음의 경우에 열릴 수 있다:

- 하이퍼레인이 새로운 체인으로 확장될 때
- 기존 체인에서 다음과 같은 이유로 검증자 포지션이 비워질 때
 1. 검증자가 교체(rotated out)될 경우
 2. 보안 요구사항에 따라 검증자 포지션이 확장될 경우

공유 커뮤니케이션 채널을 통해 상호 관심이 확인되면, 검증자는 배포를 진행할 수 있다(원한다면 조기 배포도 가능). 자세한 내용은 [Run Validators](#)를 참고하라.

검증자 보상(Validator Rewards) 청구는 [Validator Rewards](#) 섹션을 참고하라.

슬래싱 (Slashing)

슬래싱은 하이퍼레인 프로토콜 내에서 정직한 참여를 유도하기 위해 설계된 메커니즘이다. 하이퍼레인 프로토콜의 보안을 위해 HYPHER를 스테이킹한 참여자는 보안 위반이 발생할 경우 슬래싱 위험을 감수한다. 예를 들어, 검증자 정족수(quorum)가 정식 체인 상태에 포함되지 않은 메시지에 서명하는 등 안전성 위반(safety violation)을 초래할 경우, 해당 체인에 위임된 HYPHER의 일부가 슬래싱될 수 있다. 즉, 스테이커의 스테이킹된 HYPHER가 감액될 수 있다. 슬래싱은 검증자의 사기 행위가 증명(proven fraud)될 때 발생한다.

사기 및 온체인 사기 귀속 (Fraud and Onchain Fraud Attribution)

하이퍼레인의 기본 ISM 세트 내 모든 체인에서는 스마트 컨트랙트를 통해 사기(fraud)가 검증된다. 이러한 컨트랙트는 [AttributeCheckpointFraud](#) 컨트랙트를 활용해 사기 귀속 시스템(fraud attribution system)을 구현하며, 특정 유형의 부정행위를 정의한다.

[AttributeCheckpointFraud](#) 컨트랙트는 위반 유형에 따라 특정 attribute 함수를 사용하여 검증자(validator)에게 사기를 귀속시킨다. 이 귀속 결과는 이후 HYPHER Vault 내 슬래싱 로직을 트리거하는 데 사용된다.

사기 유형 (Fraud Types)

[AttributeCheckpointFraud](#) 컨트랙트는 네 가지 사기 유형을 정의한다:

1. **Whitelist:** 검증자가 잘못된 머클 트리 구현을 사용함.
2. **Premature:** 검증자가 너무 일찍 체크포인트에 서명 및 제출함.
3. **Message ID:** 체크포인트가 실제 메시지와 일치하지 않는 잘못된 메시지 ID를 포함함.
4. **Root:** 체크포인트 내 머클 루트가 유효하지 않음.

슬래싱 페널티 및 절차 (Slashing Penalties and Process)

슬래싱 페널티는 모든 HYPHER 스테이커에게 사회화(socialized)되어 적용된다. 예를 들어, 전체 HYPHER의 5%가 슬래싱된다면, 모든 스테이커는 스테이킹된 HYPHER의 5%를 잃게 된다.

초기에는 슬래싱이 하이퍼레인 재단(Hyperlane Foundation)과 보안위원회(Security Council)의 감독하에 이루어지며, 이들이 슬래싱 결정을 승인하거나 거부할 수 있다. 시간이 지나면 이러한 과정은 트러스트리스 ISM(trustless ISM)과 같은 모듈을 통한 온체인 집행으로 전환될 예정이다.

보상 (Rewards)

보상은 프로토콜이 인센티브를 정렬하고(active participation) 참여를 유지하는 핵심 요소다. 하이퍼레인 프로토콜은 네트워크 보안 유지, 메시지 전송, 검증자 운영 등 기여도에 따라 보상을 분배한다.

세 가지 주요 수혜자 그룹은 다음과 같다:

수혜자(Recipient)	보상 유형(Reward)
스테이커(Stakers) (stHYPER 보유자)	스테이커 보상(Staker Rewards)
검증자(Validators) (기본 ISM 검증자 세트)	검증자 보상(Validator Rewards)
사용자(Users) (크로스체인 메시지 전송 트랜잭션을 서명한 EOA)	확장 보상(Expansion Rewards)

스테이커 보상 (Staker Rewards)

프로토콜은 Symbiotic Vault에 HYPER를 스테이킹한 참여자에게 보상을 분배한다. 이 보상은 하이퍼레인의 경제적 보안에 기여한 사용자를 장려하기 위한 것이다.

- **자격(Eligibility):** stHYPER를 보유한 모든 사용자는 보상을 받을 자격이 있다.
- **분배(Distribution):** 보상은 에포크(약 30일)마다 고정 비율로 배분된다.
- **보상 계산(Reward Calculation):** 보상은 스테이킹된 총액과 유지 기간에 비례하여 계산된다. 검증자 커미션(Validator Commission)이 차감된 후 스테이커는 자신의 지분 비율에 따라 보상을 받는다.

이 보상 구조는 하이퍼레인 프로토콜의 경제적 보안을 지속적으로 강화하도록 사용자를 유인한다.

검증자 보상 (Validator Rewards)

프로토콜은 기본 인터체인 보안 모듈(ISM) 검증자 세트의 참여자에게 보상을 분배한다. 이들은 크로스체인 메시지를 검증하여 프로토콜의 보안을 유지한다. 검증자 보상은 스테이킹 보상의 일부로 커미션 형태로 지급된다.

- **자격(Eligibility):** 기본 ISM 검증자 세트에 속한 검증자.
- **분배(Distribution):** 보상은 Symbiotic UI를 통해 분배된다.
 - 검증자는 `claimRewards()` 함수를 통해 메르클 증명(Merkle proof)을 제출해 보상을 청구할 수 있다. 이 증명은 [Symbiotic CLI](#)를 통해 획득할 수 있다. 자세한 내용은 [Symbiotic Operator Rewards](#)를 참고하라.

◦ 검증자 보상은 등록 시 지정된 운영자 키(Operator key)에 직접 할당된다.

- **ISM 참여 자격 요건:** 검증자는 운영 보안, 과거 성과, HYPER 스테이킹 규모 등의 기준에 따라 평가된다. 자세한 내용은 검증자 선정(Validator Selection) 섹션을 참고하라.

참고: 애플리케이션 전용 ISM(Application-specific ISM) 등 다른 ISM의 검증자는 하이퍼레인 프로토콜로부터 직접 보상을 받지 않는다.

이 보상 구조는 하이퍼레인 검증자 운영을 장려한다.

확장 보상 (Expansion Rewards)

프로토콜은 크로스체인 메시지를 전송하며 프로토콜 활동을 증가시키는 사용자에게 보상을 분배한다. 사용자는 크로스체인 메시지를 전송하는 트랜잭션에 서명한 외부 소유 계정(EOA)을 의미한다.

- **자격(Eligibility):** 확장 보상은 일정 기준을 초과하는 수수료를 발생시킨 모든 하이퍼레인 사용자 및 애플리케이션에 선형적으로 분배된다. 즉, 사용자 또는 애플리케이션이 전체 초과 수수료 대비 차지하는 비율에 따라 stHYPER가 분배된다.
- **작동 방식(How it works):** 확장 보상은 프로토콜 사용량을 기준으로 분기마다 과거 활동에 대한 보상으로 지급된다.
- **보상 청구(Claiming rewards):** 자격이 있는 주소는 사용자 인터페이스(UI)를 통해 보상을 청구할 수 있다.
- **지급 주기(Frequency):** 보상은 3개월마다 분배된다.

이 보상 구조는 프로토콜의 실제 사용과 채택을 촉진하기 위한 것이다.

스트릭 보너스: 하이퍼스트릭 (Streak Multiplier: HyperStreak)

하이퍼레인 보상 체계는 장기적인 프로토콜 정렬을 장려하기 위해 새로운 인센티브 모델인 **HyperStreak**을 도입한다.

HyperStreak란 무엇인가?

HyperStreak는 사용자가 stHYPER를 일정 기간 연속적으로 보유(stake and hold)할 경우 추가 보상을 제공하는 시스템이다. 사용자가 보유 기간 동안 자신의 스트릭(streak)을 유지할수록 더 높은 보상을 받을 수 있다. 토큰을 미리 잠그지 않아도 되며, HyperStreak는 사용자가 얼마나 오래 stHYPER를 유지했는지를 추적함으로써 지속적인 참여를 유도한다.

작동 방식 (How it works)

- **자격(Eligibility):** HyperStreak를 쌓으려면 stHYPER를 보유해야 한다.
- **스트릭 축적(Streak Accumulation):** 사용자가 stHYPER를 이동하거나 출금하지 않고 매일 유지할 때마다 스트릭이 하루씩 증가한다. 출금 또는 전송 시 스트릭이 초기화된다(단, 유예 기간 정책이 적용될 수 있음).

- **스트릭 레벨(Streak Levels):** HyperStreak은 4단계 레벨로 구성되며, 스트릭을 오래 유지할수록 더 높은 레벨(최대 레벨 4)에 도달한다. 각 레벨은 보상 부스트를 제공한다.
- **레벨 할당(Level Assignment):** 시스템은 최대 스트릭 대비 비율에 따라 동적으로 레벨을 부여한다. 출시 후 첫 90일 동안 연속적으로 stHYPER를 유지한 사용자는 최고 레벨(레벨 4)에 도달한다. 초기에는 출시 당일 스테이킹을 시작하고 보유를 유지한 사용자가 레벨 4 자격을 얻는다.

보상 분배 및 할당 (Reward Distribution & Allocation)

보상은 분기별 확장 보상(Expansion Rewards) 풀에서 가중치 시스템에 따라 분배된다. 각 사용자의 보상 비율은 스트릭 레벨에 따라 결정된다. 즉, 더 긴 스트릭을 가진 사용자가 더 큰 비중을 차지하며, 이는 스트릭이 없는 사용자에게 돌아가는 보상 비중을 줄인다.

레벨	B (보너스 비율)	가중치(Weight)
레벨 4	0.6	1.60
레벨 3	0.4	1.40
레벨 2	0.3	1.30
레벨 1	0.2	1.20
스트릭 없음	0	1.00

총 보상 풀 R이 주어졌을 때, 각 사용자의 보상은 다음과 같이 계산된다:

$$\text{User Reward} = \left(\frac{\text{User's Weight}}{\sum_{i=1}^N \text{User}_i\text{'s Weight}} \right) \times R$$

(4)

HyperStreak 메커니즘은 지속적인 참여를 장려하며, 초기부터 참여한 사용자에게 추가 보상을 제공한다. 이는 인센티브 정렬을 통해 하이퍼레인 프로토콜의 장기적 지속 가능성을 지원하도록 설계되었다.

추가 자료 (Additional Resources)

- [하이퍼레인 재단 블로그 게시물](#)

자료 (Resources)

용어집 (Glossary)

에이전트 (Agent)

하이퍼레인 에이전트는 온체인 하이퍼레인 스마트 컨트랙트 상태를 읽고 쓰는 오프체인 주체다. 예시 에이전트로는 검증자(Validators)와 릴레이어(Relayer)가 있다.

Aggregation ISM

Aggregation ISM은 다수의 ISM로부터 보안을 집계(aggregate)하는 인터체인 보안 모듈 유형으로, 특정 인터체인 메시지에 대해 n개 중 m개의 ISM 검증을 요구한다.

체크포인트 (Checkpoint)

체크포인트는 특정 시점의 Mailbox 증분 머클 트리 상태에 대응하는 (merkle root, index) 쌍을 의미한다. 검증자의 체크포인트 서명은 Multisig ISM에서 사용된다.

담보 체인 (Collateral chain)

Hyperlane Warp Route에서, 담보 토큰이 워프 라우트에 예치되는 체인을 의미한다.

담보 토큰 (Collateral token)

Hyperlane Warp Route에서, 원격 합성 체인에 래핑 토큰을 생성하기 위해 담보 체인에 예치되는 토큰을 의미한다.

기본 ISM (Default ISM)

수신자가 자체 ISM을 지정하지 않은 경우 수신 메시지를 검증하는 데 사용되는 인터체인 보안 모듈이다.

도메인 (Domain)

특정 체인을 식별하는 고유 식별자로, 하이퍼레인 프로토콜이 메시지의 출처와 목적지를 판단하는 데 사용된다. EVM 체인 ID와 같을 수 있으나 항상 그렇지 않다. 알려진 하이퍼레인 도메인 ID 목록은 domains를 참조하라.

인터체인 계정 (Interchain accounts, ICA)

사용자가 인터체인 스마트 컨트랙트 호출을 수행할 수 있도록 하는 [미들웨어](#) 스마트 컨트랙트다. 예를 들어, DAO는 인터체인 계정을 통해 원격 체인에서 컨트랙트를 소유할 수 있다. 자세한 내용은 [Interchain Accounts](#)를 참고하라.

인터체인 가스 페이마스터 (Interchain gas paymaster)

[릴레이어](#)가 배포하는 스마트 컨트랙트로, 원본 체인에서 목적지 체인의 메시지 전달을 위한 지불을 수락한다. 자세한 내용은 [interchain gas payments](#)를 참고하라.

인터체인 쿼리 (Interchain queries, IQS)

사용자가 인터체인 뷰 호출을 수행할 수 있도록 하는 [미들웨어](#) 스마트 컨트랙트다. 예를 들어, 스마트 컨트랙트는 인터체인 쿼리를 사용해 원격 체인에서 오라클 환율이나 토큰 잔액을 조회할 수 있다.

인터체인 보안 모듈 (Interchain Security Module, ISM)

[인터체인 보안 모듈](#)(ISMs)은 하이퍼레인의 인터체인 [Mailbox](#)에 보안을 제공하는 스마트 컨트랙트다. ISM은 목적지 체인으로 전달되는 인터체인 메시지가 [실제로](#) 원본 체인에서 전송되었음을 검증한다.

Mailbox

하이퍼레인의 가장 중요한 스마트 컨트랙트로 볼 수 있는 [mailbox](#)는 개발자가 인터체인 메시지를 [전송](#)하고 [수신](#)할 수 있는 인터페이스를 노출한다.

미들웨어 (Middleware)

메시지를 송수신하고 개발자용 API를 노출하는 스마트 컨트랙트다. 개발자는 [mailbox](#)와 직접 상호작용하는 대신 이 API를 사용하는 것이 권장된다. 예시 미들웨어에는 [interchain-accounts-ica](#)와 [interchain-queries-iqs](#)가 있다.

Multisig ISM

[Multisig ISM](#)은 특정 인터체인 메시지를 검증하기 위해 n명 중 m명의 [검증자](#) 서명을 사용하는 [인터체인 보안 모듈](#)의 한 유형이다.

퍼미션리스 상호운용성 (Permissionless Interoperability)

하이퍼레인은 누구나 어떤 체인(레이어 1, 롤업, 앱체인 등)에도 배포할 수 있으며, 그 체인이 하이퍼레인이 배포된 다른 모든 체인과 원활히 통신하도록 한다. 하이퍼레인 프로토콜의 개방형 설계를 퍼미션리스 상호운용성(PI)이라 부른다. 하이퍼레인 코어 팀이 아닌 기관이 배포한 체인을 ‘PI 체인’이라고 한다.

릴레이어 (Relayer)

릴레이어는 원본 체인에서 목적지 체인으로 메시지를 전달하는 하이퍼레인 에이전트다. 릴레이어는 신뢰할 필요가 없으며, 누구나 릴레이어를 운영할 수 있다.

Routing ISM

Routing ISM은 전달되는 메시지에 따라 서로 다른 ISM에 위임하는 인터체인 보성 모듈 유형이다. 예를 들어, 메시지의 원본 체인에 따라 다른 ISM을 사용할 수 있다.

합성 체인 (Synthetic chain)

Hyperlane Warp Route에서, 래핑 토큰이 발행되는 체인을 의미한다. 이 래핑 토큰은 담보 체인에 잠겨 있는 담보 토큰으로 담보화된다.

검증자 (Validator)

검증자는 원본 체인에서 전송된 메시지를 인증(attest)하는 하이퍼레인 에이전트다. Multisig ISM은 수신 메시지 보안을 제공하기 위해 검증자 서명을 사용한다.

Hyperlane Warp Route

Hyperlane Warp Routes는 토큰 브리징 개념에 대한 하이퍼레인의 접근으로, 어떤 ERC20 유사 자산 이든 하이퍼레인을 통해 퍼미션 없이 어느 체인으로든 전송할 수 있게 한다. Hyperlane Warp Route 배포 방법을 참고하라.

자주 묻는 질문 (Frequently Asked Questions)

일반 질문 (General Questions)

하이퍼레인이란 무엇인가? 하이퍼레인은 모듈러 블록체인 스택을 위해 구축된 최초의 범용(pervasive) 퍼미션리스 상호운용성 프레임워크다. 누구나 [하이퍼레인을 배포](#)하여 레이어 1, 롤업, 앱체인 등 어떤 블록체인 환경에서든 다른 하이퍼레인 배포 체인과 원활히 통신할 수 있다.

하이퍼레인을 사용하면 블록체인 개발자에게 어떤 이점이 있는가? 하이퍼레인은 블록체인 및 dApp 개발자 모두에게 이점을 제공한다. * 블록체인 개발자에게 하이퍼레인 배포는 하이퍼레인이 배포된 다른 연결된 블록체인의 사용자, 자산, 가치 있는 상태에 대한 접근성을 확장한다. * dApp 개발자에게 하이퍼레인은 여러 블록체인 간 원활한 연결을 제공하여, 개발자는 인터체인 네트워크 효과와 유동성을 창출하고 사용자는 선호하는 체인에서 애플리케이션과 상호작용할 수 있다.

Abacus Works란 무엇인가? Abacus Works는 하이퍼레인 프로젝트에 참여하는 법인 명칭이다. 이전 게시물과 저장소에서 Abacus에 대한 언급을 볼 수 있다. 하이퍼레인은 2022년 가을 Abacus에서 Hyperlane으로 리브랜딩했다.

하이퍼레인으로 무엇을 구축할 수 있는가? 하이퍼레인의 [메시징 인터페이스](#)는 dApp 개발자가 서로 다른 체인의 스마트 컨트랙트 간 임의의 바이트를 전송할 수 있게 한다. 이를 통해 여러 체인에 걸친 인터체인 애플리케이션을 만들 수 있다. 영감을 얻기 위해 하이퍼레인 위에 구축된 사전 제작 애플리케이션을 살펴보자: *

Hyperlane Warp Routes (HWR): 사용자가 토큰을 한 체인에서 다른 체인으로 이동할 수 있게 한다.

* **Interchain Accounts:** 사용자가 인터체인 함수 호출을 수행할 수 있게 한다. * **Interchain**

Queries: 사용자가 인터체인 뷰 호출을 수행할 수 있게 한다.

개발자인데, 우리 팀이 하이퍼레인으로 어떻게 빌드할 수 있나? 이 FAQ를 보고 있다면 이미 문서를 찾은 것으로, 훌륭한 시작점이다. 구현 중 의문이 생길 수 있으며, 우리는 기꺼이 도와드린다. 하이퍼레인 커뮤니티는 [Discord](#)에서 활발하다. 개발자 피드백은 제품 로드맵에 직접적인 영향을 준다.

기술 질문 (Technical Questions)

메시징 (Messaging)

하이퍼레인으로 메시지를 보내면 무엇이 일어나는가? 자세한 내용은 [send](#)와 [receive](#) 페이지를 참조하라. 요약하면: 1. 애플리케이션이 원본 체인의 `dispatch()` 함수를 호출하여 메시지를 Mailbox의 머클 트리에 삽입한다. 2. 하이퍼레인 릴레이어가 디스패치된 메시지를 관찰하고 수신자의 인터체인 보안 모듈(ISM)을 위한 메타데이터를 구성한다. 3. 릴레이어가 `Mailbox.process()`를 호출하여 메시지를 수신자에게 전달한다. 4. Mailbox가 수신자의 ISM으로 메시지를 검증한 뒤 `recipient.handle()`을 호출하여 메시지를 전달한다.

보낸 메시지의 상태를 어떻게 확인할 수 있나? [Hyperlane Message Explorer](#)의 검색창에 주소나 트랜잭션 해시를 붙여 넣으면 해당 메시지의 상태와 이력을 확인할 수 있다. 메시지 전송이 실패한 경우 원인을 추적할 수 있는 오류 메시지가 표시된다. 자세한 내용은 [Explorer Debugging](#)을 참고하라.

하이퍼레인에서 인터체인 메시지를 보낸다는 것이 실시간 채팅·문자 전송을 의미하나? 아니다. 하이퍼레인은 서로 다른 체인의 스마트 컨트랙트가 상호작용할 수 있도록 설계된 프로토콜이다. 여기서 메시지 전달은 사람 간 P2P 또는 텍스트 메시지가 아니라 스마트 컨트랙트 간 임의 바이트 전송을 의미한다. 채팅 사용 사례에는 [XMTP](#), [Push](#), [Orbis](#)를 고려하라.

하이퍼레인은 토큰 브리지인가? 정확히는 아니다. 하이퍼레인은 블록체인 간 통신을 가능하게 하는 범용 메시지 전달(GMP) 프로토콜이다. 토큰 브리지는 하이퍼레인 위에 구축될 수 있는 많은 애플리케이션 유형 중 하나일 뿐이다.

배포 (Deployment)

하이퍼레인은 어디에 배포되어 있나? 알려진 배포 목록은 [컨트랙트 주소](#) 페이지에서 확인할 수 있다.

고급 주제 (Advanced Topics)

하이퍼레인은 어떻게 보안이 보장되나? 하이퍼레인은 인터체인 보안 모듈(ISM)로 특징지어지는 모듈러 보안 스택으로 보호된다. 개발자는 사전 구축 ISM을 구성하거나 서로 조합하거나, 애플리케이션 요구에 맞춰 커스텀 ISM을 만들 수 있다. 모듈러 보안 접근은 하이퍼레인이 최신 보안 모델 발전을 지속적으로 반영하도록 한다.

하이퍼레인은 검증자가 몇 명이며 누가 운영하나? 검증자 수는 인터체인 보안 모듈(ISM) 구성에 따라 달라진다. 기본 ISM은 특정 검증자 세트를 사용하며, 임계값, 운영자, 주소 등의 세부 정보는 [여기](#)에서 확인할 수 있다.

메시지 무결성은 어떻게 보장하나? 하이퍼레인의 메시지 보안은 ISM 구성에 의해 좌우된다. 기본적으로 메시지는 기본 ISM의 검증자 세트에 의존하며, 정족수(quorum) 서명을 요구한다. 극히 드물게 검증자 손상이 발생하면 메시지 보안에 영향이 있을 수 있지만, 개발자는 자체 ISM을 구성하여 애플리케이션 요구에 맞춘 더 강한 보안 모델을 채택할 수 있다.

목적지 체인에서 트랜잭션이 실행되었는지 원본 체인으로 확인이 돌아오나? 현재는 원본 체인으로 자동 확인이 돌아오지 않는다. 다만 목적지 체인의 `handle` 함수를 커스터마이징하여 이 기능을 추가할 수 있다. 자세한 내용은 [handle 함수](#)를 참고하라.

왜 아웃바운드 메시지 ID를 머클 트리에 저장하나? 아웃바운드 메시지 ID를 머클 트리에 저장하면 메시지를 효율적으로 추적·검증할 수 있다. 이를 통해 검증자는 새로 생성된 메시지를 정확히 감지하고 서명할 수 있다.

검증자가 왜 `MerkleTreeHook.latestCheckpoint()`를 호출하나? 검증자는 `MerkleTreeHook.latestCheckpoint()` 함수를 사용해 새로운 트랜잭션을 인덱싱해야 하는 시점을 판단한다. 이 폴링 메커니즘 덕분에 검증자는 전체 트리를 백필할 필요 없이 즉시 새 메시지 서명을 시작할 수 있다.

커뮤니티 & 기여 (Community & Contributions)

하이퍼레인 커뮤니티에 어떻게 참여하나? [Discord](#)에 참여하거나 [Twitter](#)를 팔로우하라. 인터체인의 미래를 논의하는 개발자와 열성적인 커뮤니티를 만날 수 있다.

하이퍼레인에서 일하고 싶다. 채용 공고는 어디에서 보나? [여기](#)에서 오픈 포지션을 확인하라.

이 문서 개선에 어떻게 기여할 수 있나? [docs 저장소](#)에 직접 PR을 보내 수정할 수 있다.

지원 또는 질문은 어디에서 할 수 있나? [하이퍼레인 Discord](#)에서 지원을 받거나 질문할 수 있다. 커뮤니티와 팀이 적극적으로 도와준다.

감사(Audits)

하이퍼레인 2025년 2분기 보안 감사 (Hyperlane Q2 2025 Security Audit)

[원문\(Original\)](#)

HyperToken 감사 (HyperToken Audit)

[원문\(Original\)](#)

Yield Routes 감사 (Yield Routes Audit)

[원문\(Original\)](#)

Compound Staker Rewards 감사 (Compound Staker Rewards Audit)

[원문\(Original\)](#)

USDT 감사 (USDT Audit)

[원문\(Original\)](#)

CCIP 감사 [v5.12.0](#)

[원문\(Original\)](#)

Hook 및 ISM 감사 (Hook and ISM Audit)

[미러\(Mirror\)](#)

프로토콜 v3 감사 (Protocol v3 Audit)

[원문\(Original\)](#)

[미러\(Mirror\)](#)

프로토콜 v2 감사 (Protocol v2 Audit)

[원문\(Original\)](#)

[미러\(Mirror\)](#)

프로토콜 v1 감사 (Protocol v1 Audit)

[원문\(Original\)](#)

[미러\(Mirror\)](#)

Sealevel 프로토콜 감사 (Sealevel Protocol Audit)

[원문\(Original\)](#)

[미러\(Mirror\)](#)

Sealevel 감사 - Sherlock (Sealevel Audit - Sherlock)

[원문\(Original\)](#)

CosmWasm 프로토콜 감사 (CosmWasm Protocol Audit)

[원문\(Original\)](#)

[미러\(Mirror\)](#)

Starknet 프로토콜 감사 (Starknet Protocol Audit)

[원문\(Original\)](#)