

미나(Mina)

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.

미나 사용하기 (Using Mina)

지갑 설치하기 (Install a Wallet)

미나를 송금하고 수신할 수 있는 지갑에 대해 알아보자.

미나 커뮤니티는 다양한 형태의 지갑을 제작했다. 현재 Chrome용 Auro Wallet과 MinaPortal은 zkApp과의 상호작용을 지원하며, 이들 중 어떤 지갑을 사용하더라도 MINA를 송금하거나 수신할 수 있다.

- [Auro Wallet \(Chrome, Firefox, iOS, & Android\)](#)
- [Clorio Wallet \(Windows, MacOS, Linux, 온라인\)](#)
- [Ledger Mina Protocol 지갑](#) 사용 방법은 [Ledger 하드웨어 지갑 문서](#)를 참고하라.
- [MinaPortal, MetaMask Snap](#) 자세한 내용은 [MinaPortal 위키](#)를 참고하라.
- [Pallad \(Chrome 및 Brave 사이드 패널\)](#)

참고: 스팸 방지를 위해 미나 네트워크는 계정 생성 시 한 번만 부과되는 1 MINA의 수수료를 청구한다. 이 수수료는 첫 번째 수신 거래에서 자동으로 차감된다.

노드 운영자를 위한 지갑 (Wallet for Node Operators)

미나 CLI는 MINA를 보관, 송금, 수신할 수 있는 기능을 제공한다. 그러나 이 방법은 미나 노드 운영자만을 위한 것이며, 일반 사용자가 지갑으로 사용하는 것은 **권장하지 않는다**. 자세한 내용은 [미나 CLI 문서](#)를 참고하라.

다음 단계 (Next Steps)

이제 지갑을 설정하는 방법을 배웠다면, [MINA를 송수신하는 방법](#)을 배워보자.

Ledger 하드웨어 지갑 (Ledger Hardware Wallet)

Ledger는 미나를 Nano S 및 Nano X 하드웨어 지갑에서 지원한다. 하드웨어 장치에 미나 애플리케이션을 설치해 자금을 보관하고 미나 블록체인과 상호작용할 수 있다.

정보: 현재 Ledger 지갑을 사용해 zkApp 트랜잭션에 서명하는 것은 아직 불가능하다.

미나 앱이 지원하는 작업은 다음과 같다:

- 키 쌍 생성
- 결제 트랜잭션 서명
- 위임 트랜잭션 서명

미나 앱이 **지원하지 않는** 작업은 다음과 같다:

- zkApp 트랜잭션 서명
- 기본 LedgerLive 지갑 통합 (대신 서드파티 지갑을 사용할 수 있음)

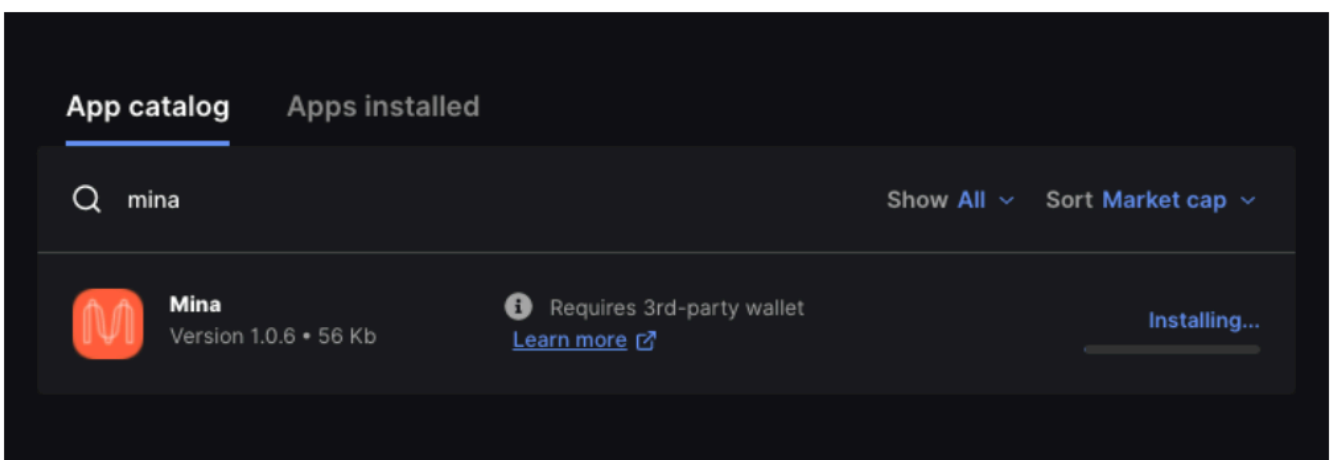
아래 지침에 따라 미나 앱을 Ledger Nano S 또는 Nano X에 다운로드할 수 있다.

미나 앱 설치 (Installing the Mina app)

시작하기 전에 아직 설치하지 않았다면 [Ledger Live](#)를 다운로드하라.

Ledger 펌웨어가 미나를 지원하는 최신 버전인지 확인하라.

1. Ledger 장치를 연결하고 잠금을 해제한다.
2. Ledger Live에서 **Manager**를 연다.
3. 장치에서 관리자 접근을 허용한다.
4. 앱 카탈로그에서 **Mina**를 검색한다.
5. **Install** 버튼을 클릭한다. 장치에 *Processing...* 메시지가 표시된다.
6. 다운로드가 완료되면 Ledger Live에 *Installed*가 표시된다.



팁: 하드웨어 지갑 관련 주의 사항:

- 비밀 니모닉 문구를 반드시 백업해라.
- Ledger를 최신 상태로 유지해라.

문제 해결 (Troubleshooting)

트랜잭션 서명이 너무 오래 걸리는 이유는 무엇인가?

미나는 새로운 암호학을 사용하고 있으며 Ledger는 해당 연산에 대한 하드웨어 가속을 지원하지 않기 때문에, 미나 앱으로 서명할 때 다른 지갑보다 시간이 더 오래 걸릴 수 있다. 향후 Ledger가 이 암호학을 지원하길 기대한다.

펌웨어 업데이트 옵션이 보이지 않는 이유는 무엇인가?

Nano X 펌웨어에서 최신 버전으로 업그레이드하는 옵션이 표시되지 않는다면, 장치가 업데이트 준비 단계에 있는 것이다. 1~2일 정도 기다렸다가 옵션이 표시되는지 확인하라. 여전히 보이지 않는다면 Ledger 지원팀에 문의하라.

다음 단계 (Next Steps)

축하한다. 이제 미나의 Ledger 앱을 설치했다. Ledger를 사용해 미나에서 지원하는 Ledger 지갑에 연결할 수 있다:

- [Aurowallet.com](https://aurowallet.com)
- [Clor.io](https://clor.io)

Ledger에 대한 도움이 필요하거나 질문이 있다면 미나 프로토콜 디스코드의 [#ledger-hardware](#) 채널에 참여하라.

송금 및 수신 방법 (How to Send & Receive)

MINA를 송금하고 수신하는 방법을 알아보자.

사전 준비 (Prerequisites)

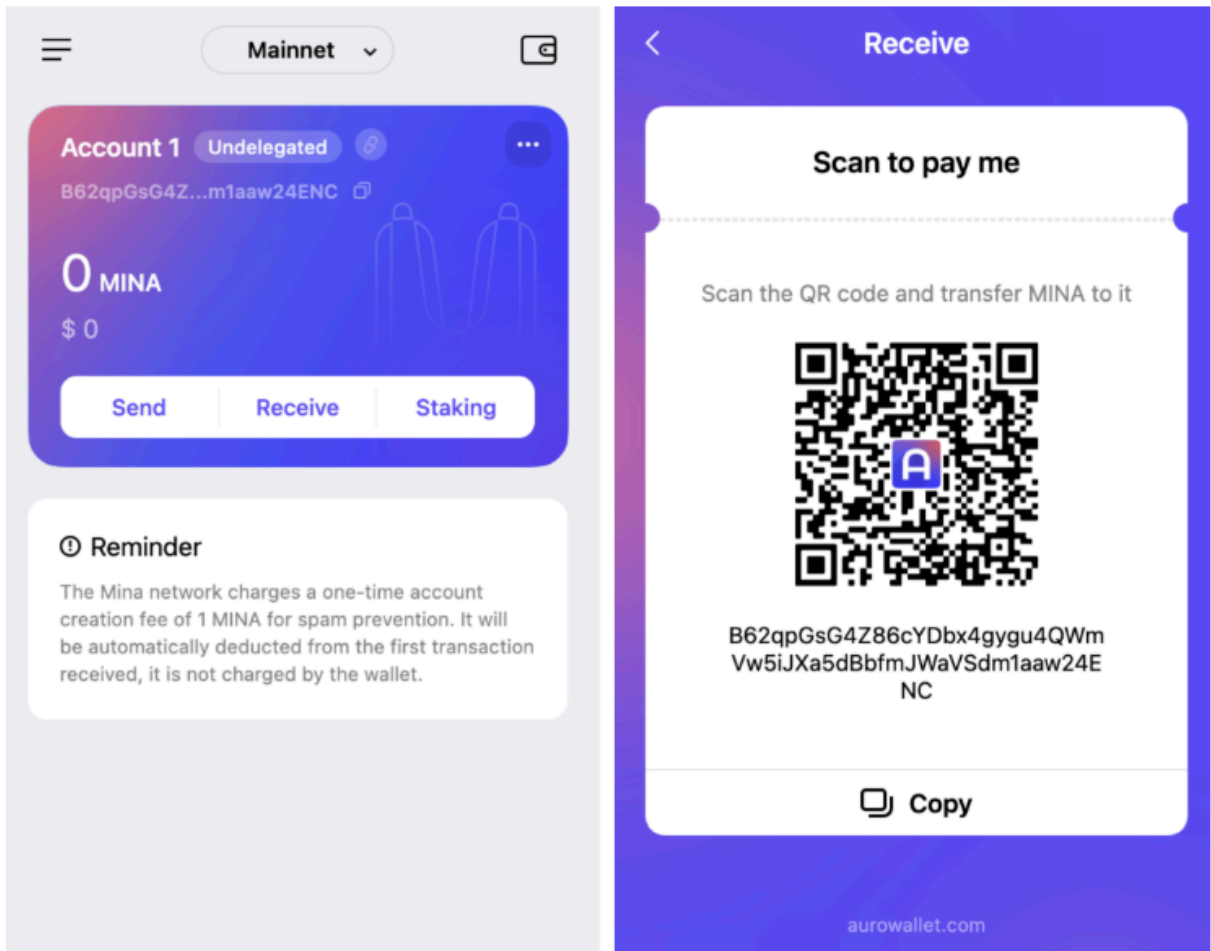
먼저, [미나 지갑을 설치했는지](#) 확인하라.

MINA 수신하기 (Receiving MINA)

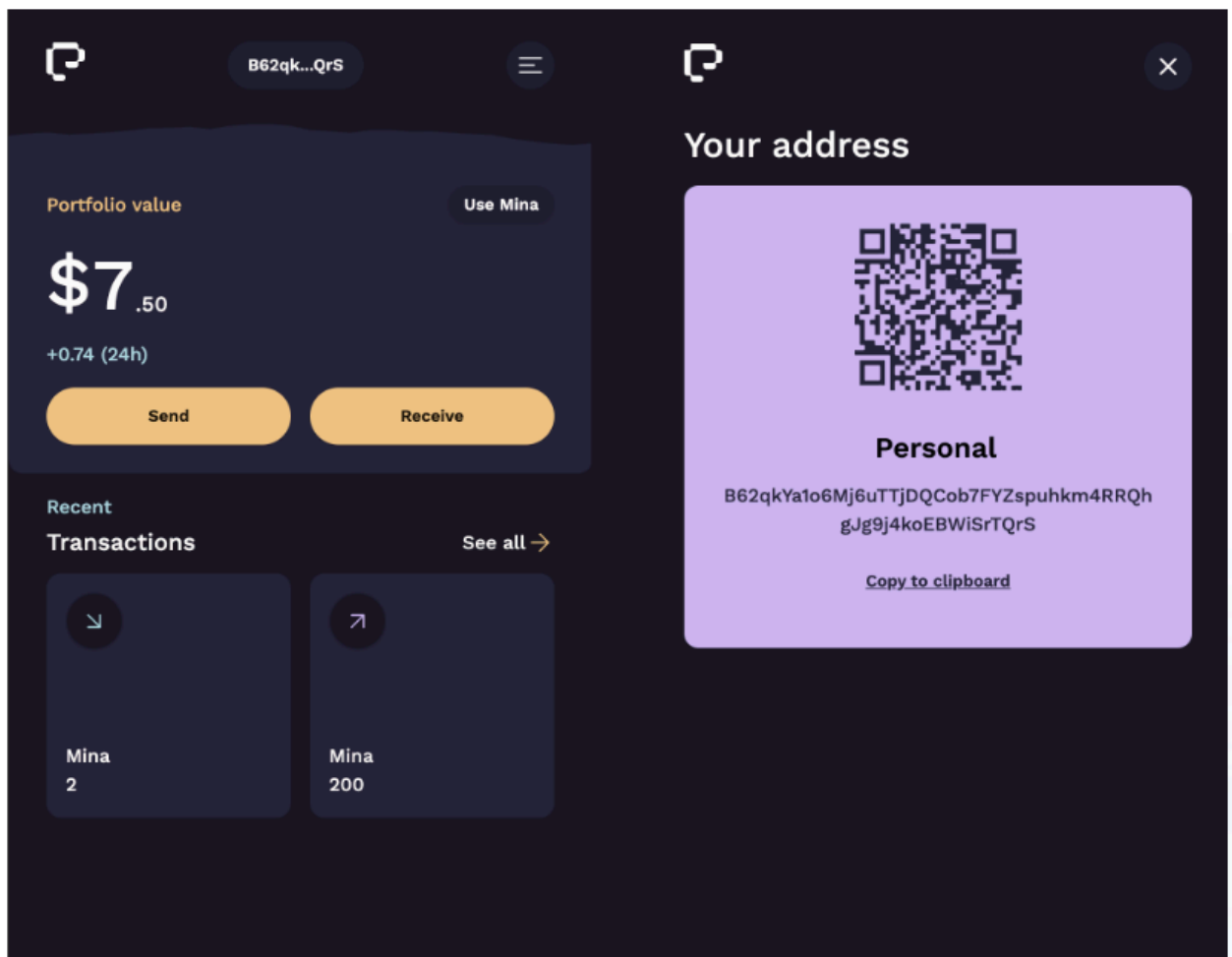
MINA를 수신하려면 송신자에게 미나 계정의 고유 주소를 제공해야 한다.

다음은 일반적인 지갑에서 주소를 찾는 방법이다:

- [Auro](#) – 지갑을 열고 *Receive*를 탭한 뒤 *Copy*를 눌러 주소 또는 QR 코드를 공유한다.



- **Clorio Wallet** – 지갑을 열고 상단을 탭해 주소를 복사한다.
- **Ledger 하드웨어 지갑** – Auro Wallet을 통해 Ledger 미나 계정을 생성한 뒤, 위의 Auro 지갑 수신 절차를 따라라.
- **Pallad** – 지갑을 열고 *Receive*를 탭한 뒤 *Copy to clipboard*를 눌러 주소를 공유한다. 또한 QR 코드도 표시된다.



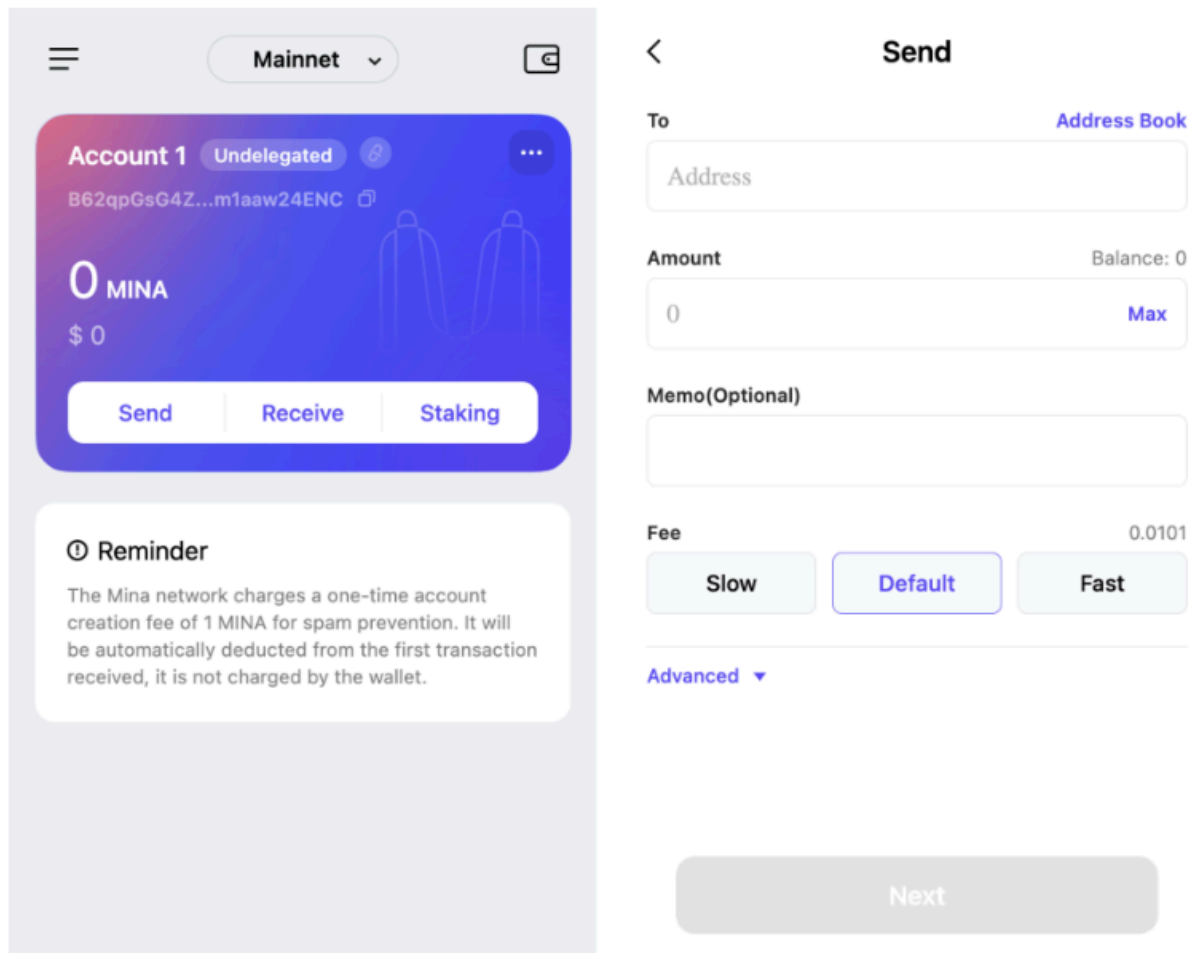
참고: 스팸 방지를 위해 미나 네트워크는 계정 생성 시 한 번만 부과되는 1 MINA의 수수료를 청구한다. 이 수수료는 첫 번째 수신 거래에서 자동으로 차감된다.

MINA 송금하기 (Sending MINA)

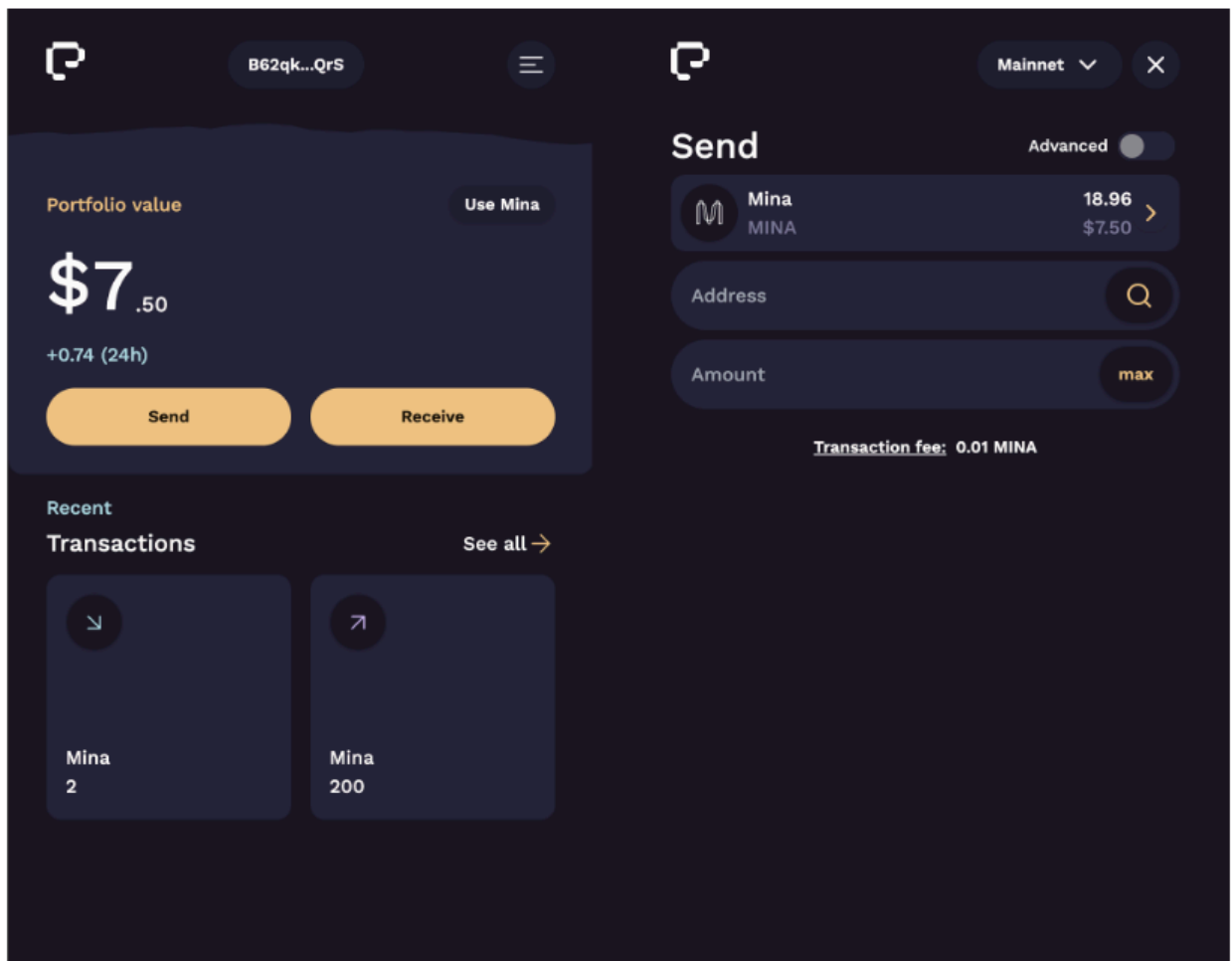
MINA를 송금할 때는 항상 수신자의 미나 주소가 올바른지 **두 번 확인하라**.

미나 블록체인에서의 트랜잭션은 **최종적이며 되돌릴 수 없다**.

- **Auro** – 지갑을 열고 *Send*를 탭한 후 주소, 금액, 수수료를 입력하고 *Next*와 *Confirm*을 차례로 탭한다.



- **Clorio Wallet** – 지갑을 열고 *Send TX*를 탭한다. 주소, 메모, 금액, 수수료를 입력하고 *Preview*를 탭한 후 nonce를 입력하고 *Confirm*을 누른다.
- **Ledger 하드웨어 지갑** – Auro Wallet을 통해 Ledger 미나 계정을 생성한 뒤, 위의 Auro 지갑 송금 절차를 따라라.
- **Pallad** – 지갑을 열고 *Send*를 탭한다. 주소와 금액을 입력하고 *Next*를 누른다. 지출 비밀번호를 입력하고 제출해 트랜잭션을 확인한다.

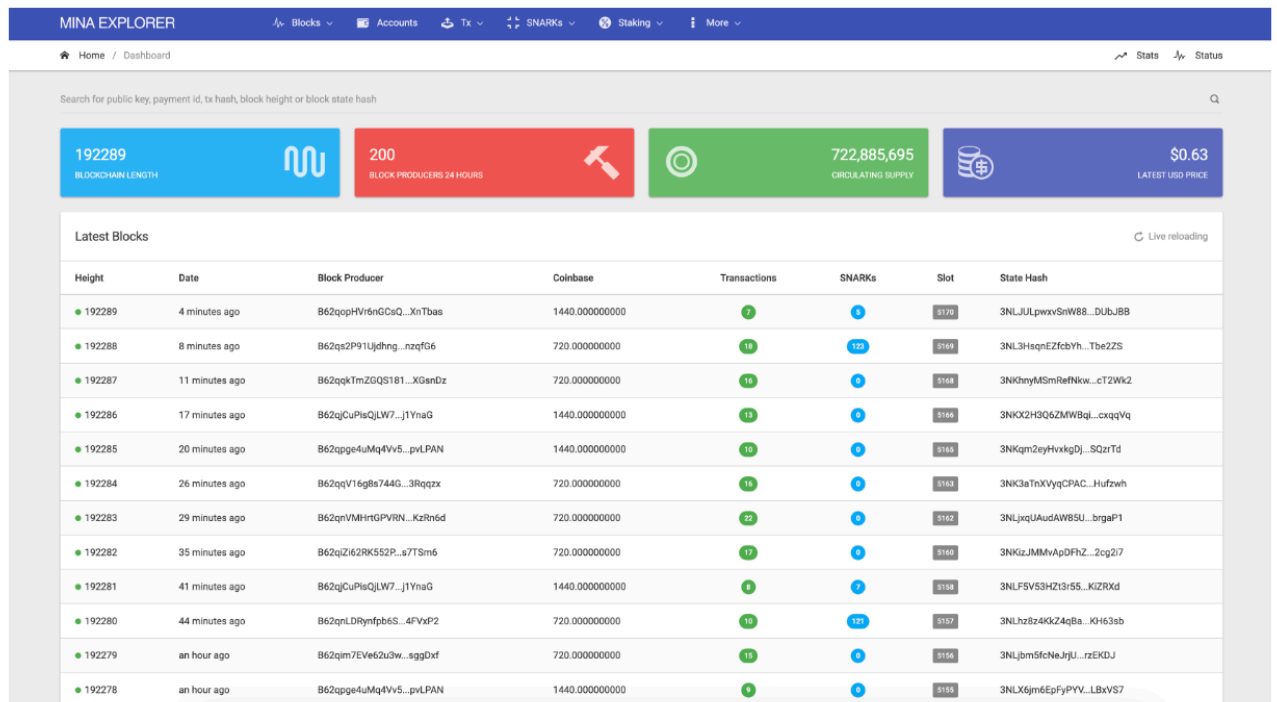


참고: 미나와 같은 블록체인에서 트랜잭션을 보낼 때는 송신자가 트랜잭션 수수료를 포함해야 한다. 대부분의 지갑은 권장 수수료를 자동으로 계산한다. 현재 네트워크 거래량을 기반으로 한 제안 수수료는 커뮤니티가 만든 [Gas Station 웹사이트](#)에서 확인할 수 있다.

블록체인 탐색기에서 트랜잭션 확인하기 (Viewing your transaction on a blockchain explorer)

커뮤니티에서 운영하는 블록체인 탐색기를 통해 자신의 트랜잭션을 확인할 수 있다:

- [MinaScan](#)
- [Minataur](#)
- [MinaSearch](#) (Granola에서 개발 중)



위임하기 (How to Delegate)

MINA를 위임하고 스테이킹 보상을 받는 방법을 알아보자.

자신의 노드를 직접 운영하는 대신, 검증자(블록 생성자)에게 MINA를 위임하여 스테이킹 보상을 받을 수 있다.

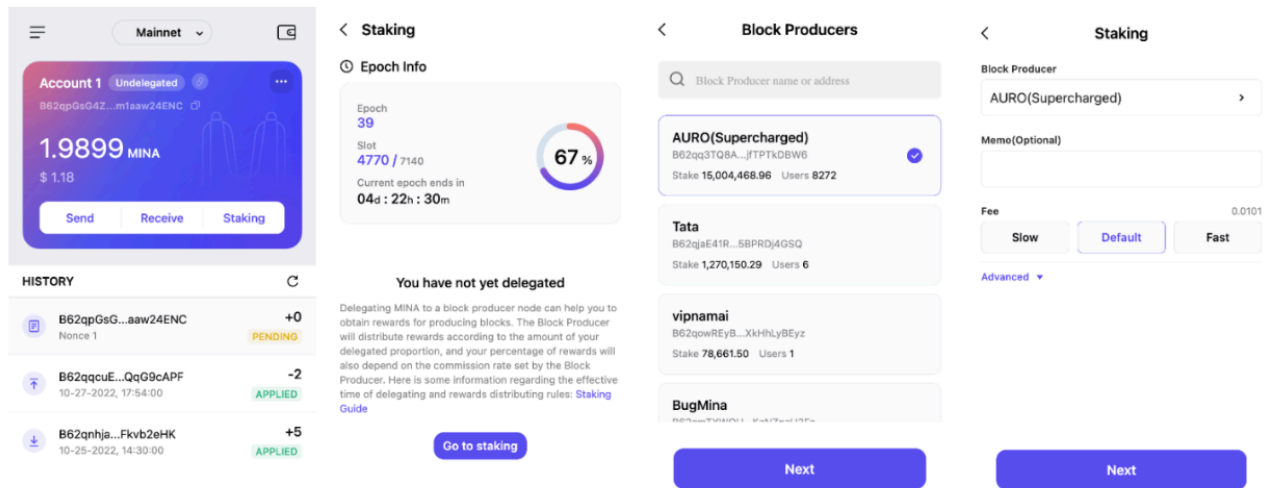
사전 준비 (Prerequisites)

먼저, 자금이 있는 미나 지갑을 설치했는지 확인하라.

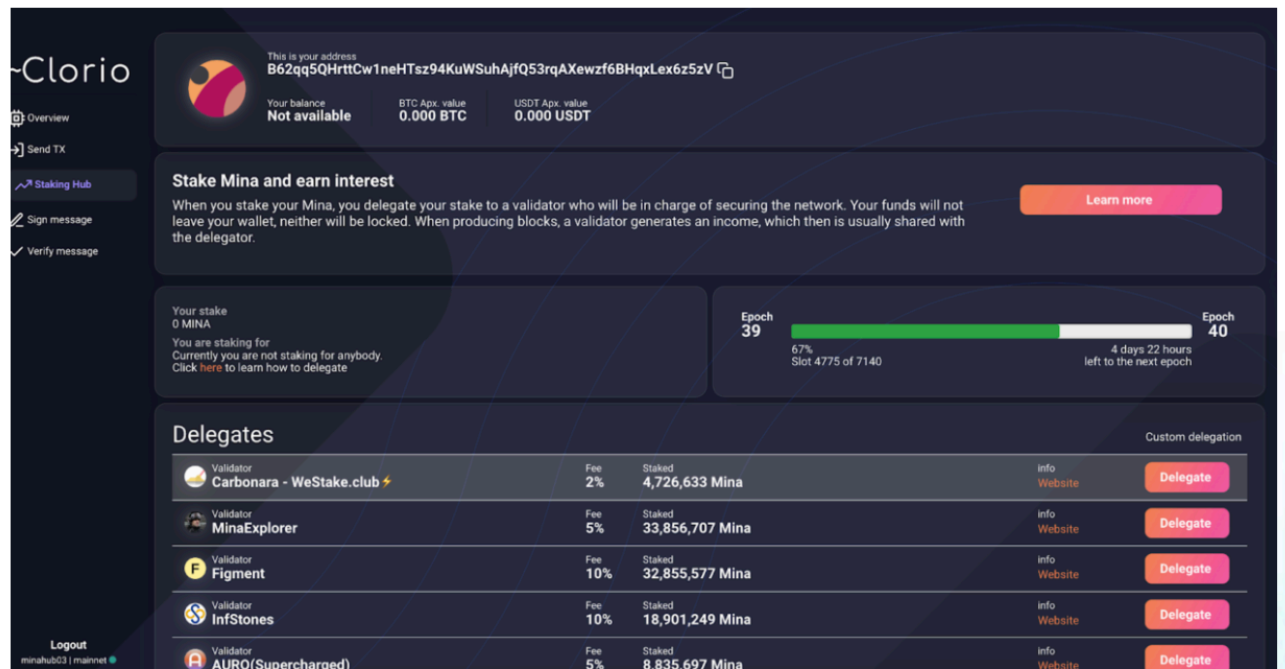
MINA 위임하기 (Delegating MINA)

다음은 일반적인 지갑에서 MINA를 위임하는 방법이다:

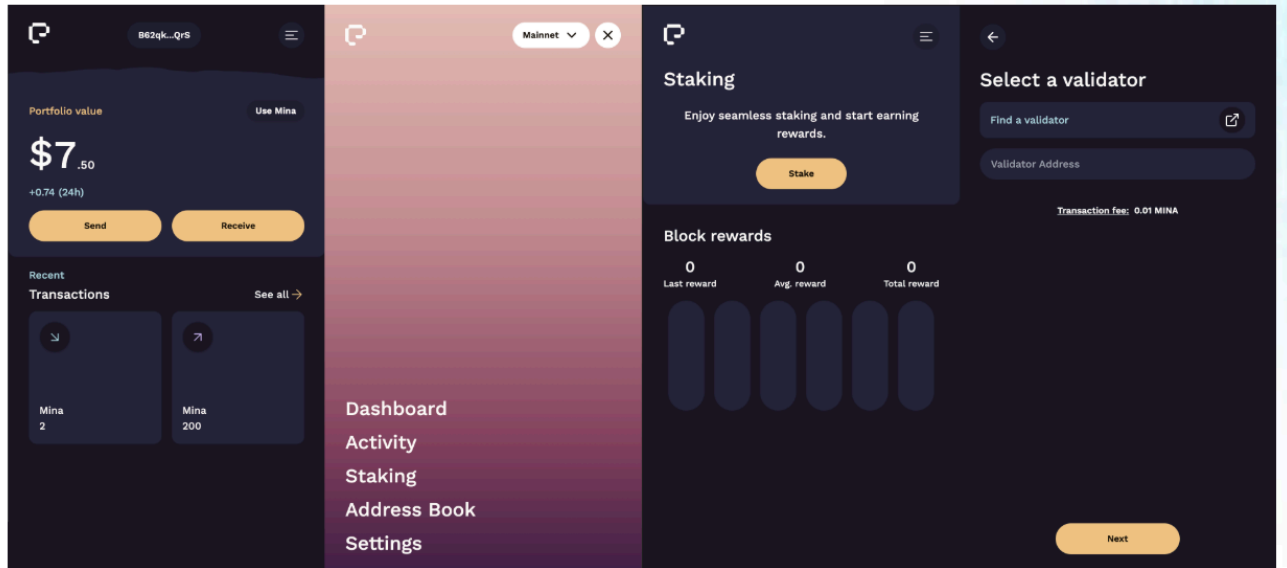
- **Auro** – 지갑을 열고 *Staking*을 탭한 뒤 *Go to Staking*을 선택한다. 블록 생성자를 선택하고 트랜잭션 세부 정보를 확인한다.



- **Clorio Wallet** – 지갑을 열고 *Staking Hub*를 탭한 뒤 검증자를 선택하고 확인한다. 트랜잭션 수수료와 암호구문(passphrase) 또는 개인 키를 입력한다.



- **Ledger 하드웨어 지갑** – Auro Wallet을 통해 Ledger 미나 계정을 생성한 뒤, 위의 Auro 지갑에서 MINA 스테이킹 절차를 따라라.
- **Pallad** – 지갑을 열고 우측 상단의 *Menu* 버튼을 탭한다. 사이드바 메뉴에서 *Staking*을 선택한다. *Stake*를 클릭하고 검증자의 공개 키를 붙여넣거나 *Find a validator*를 사용한다. *Next*를 탭하고 지출 비밀번호를 입력한 후 제출해 트랜잭션을 확인한다.



참고: 미나와 같은 블록체인에서 트랜잭션을 전송할 때 송신자는 반드시 트랜잭션 수수료를 포함해야 한다. 대부분의 지갑은 권장 수수료를 자동으로 계산한다. 현재 네트워크 거래량을 기반으로 한 제안 수수료는 커뮤니티가 만든 [Gas Station 웹사이트](#)에서 확인할 수 있다.

zkApp 사용하기 (How to Use a zkApp)

영지식 스마트 컨트랙트(zero-knowledge smart contract)와 상호작용하는 방법을 알아보자.

정보: 블록당 zkApp 트랜잭션의 최대 개수는 현재 24개로 제한되어 있다. 이 제한은 메인넷 업그레이드 이후 점차 해제될 예정이다.

사전 준비 (Prerequisites)

1. zkApp과 호환되는 [미나 지갑](#)을 설치한다.
2. 트랜잭션 수수료를 지불할 수 있도록 지갑에 MINA가 있는지 확인한다.

참고: 미나 커뮤니티는 다양한 지갑을 제작했지만, 현재 zkApp 상호작용을 지원하는 지갑은 [Chrome용 Auro Wallet](#)뿐이다.

사용 방법 (Instructions)

1. 웹 브라우저에서 zkApp을 방문한다. 예: [mycoolzkapp.com](#)
2. 의도된 방식으로 zkApp과 상호작용한다. 예를 들어 게임에서 움직임을 입력하거나 나이를 입력하는 등의 작업을 수행한다.
3. 확인 버튼을 클릭해 트랜잭션을 미나 네트워크로 전송한다.
4. 브라우저 확장 지갑에서 트랜잭션을 승인한다.
5. 완료! 미나 네트워크가 트랜잭션을 처리하며, 블록에 포함되면 zkApp의 온체인 상태가 업데이트된

다.

zkApp 트랜잭션 확인하기 (Check the zkApp transaction)

zkApp 트랜잭션이 성공적으로 처리되었는지 확인하려면, 지갑에 표시되는 트랜잭션 해시를 클릭해 블록 탐색기에서 확인한다.

미나 프로토콜 (Mina Protocol)

소개 (Introduction)

미나 프로토콜은 진정한 탈중앙화, 확장성, 보안을 실현하기 위해 설계된 레이어 1 프로토콜이다.

미나는 우아한 해결책을 제시한다. 블록체인을 일정한 크기의 암호학적 증명으로 대체하는 것이다. 이를 통해 각 사용자가 다운로드해야 하는 데이터의 양이 획기적으로 줄어든다. 미나 참여자는 전체 블록체인을 시간의 시작부터 검증하는 대신, 재귀적 영지식 증명(zk-SNARKs)을 사용해 네트워크와 트랜잭션을 완전하게 검증할 수 있다. 노드는 전체 체인을 저장하는 대신 작은 증명만 저장하면 된다. 크기가 일정하므로, 미나는 사용자가 늘어나고 트랜잭션 데이터가 축적되어도 접근성이 유지된다.

미나 프로토콜 네트워크 (The Mina Protocol)

미나 프로토콜에는 세 가지 공개 네트워크가 있다:

1. **mainnet** – 운영용 네트워크
2. **devnet** – 메인넷과 동일한 소프트웨어 버전을 기반으로 한 테스트 네트워크
3. **berkeley** – 새로운 기능이 시험되는 개발 네트워크

다음 GraphQL 쿼리를 사용해 네트워크의 식별자를 확인할 수 있다:

```
query MyQuery {  
  networkID  
}
```

이 섹션에서는 미나 프로토콜의 작동 방식을 설명한다.

- [지분증명 \(Proof Of Stake\)](#)
- [블록의 구성 \(What's in a Block\)](#)
- [블록 생성자 \(Block Producers\)](#)
- [SNARK 작업자 \(SNARK Workers\)](#)
- [스캔 상태 \(Scan State\)](#)
- [타임락 계정 \(Time-Locked Accounts\)](#)
- [결제 전송 \(Sending a Payment\)](#)
- [결제의 생애주기 \(Lifecycle of a Payment\)](#)

노드 운영자 (Node Operators)

노드 운영자는 미나 네트워크에서 미나 노드를 실행하는 방법을 설명한다.

미나 노드는 네트워크 내에서 서로 다른 역할을 수행한다.

노드 개발자 (Node Developers)

노드 개발자는 개발자가 미나 노드를 개선하거나 새로운 기능을 추가하는 방법을 설명한다.

거래소 운영자 (Exchange Operators)

거래소 운영자는 미나 네트워크에 연결하고 MINA 토큰에 대한 접근을 제공하는 방법을 설명한다.

지분증명 (Proof of Stake)

미나에 구현된 지분증명(Proof of Stake) 합의 메커니즘은 [Ouroboros Praos](#) 프로토콜의 확장 버전으로, 미나의 간결한 블록체인 구조에 맞게 약간 수정되었다. 이 문서는 Ouroboros 지분증명이 작동하는 방식에 대한 개요를 제공하며, 미나에 적용된 변경점과 추가 사항을 자세히 다룬다. Ouroboros 프로토콜의 전체 설명은 다음 원본 논문을 참고하라: [원본 논문](#), [Praos 논문](#).

Praos, Genesis, 그리고 미나에 대한 참고 (A note on Praos, Genesis, and Mina)

형식적으로, 미나의 Ouroboros 구현은 Praos의 확장판이다. 그러나 Praos를 확장한 새로운 논문 **Ouroboros Genesis**가 존재한다. 이 확장은 긴 포크(long fork) 공격에 대한 취약점을 해결하지만, 논문에서 설명된 방식으로서는 간결한 블록체인 프로토콜에서 구현할 수 없다. 이에 따라 미나는 긴 포크 공격을 방지하기 위한 새로운 간결한 방법을 도입했다.

Praos에 대한 추가 사항 (Additions to Praos)

에폭 원장 최적화 (Epoch Ledger Optimization)

Ouroboros에서는 노드가 VRF 평가를 수행하기 위해 이전 에폭의 시작 시점에 원장을 보존해야 한다. 이는 단순히 VRF 출력만으로 블록이 정당하게 제안되었는지를 판단할 수 없기 때문이다. 블록이 해당 노드에 의해 정당하게 제안되었음을 확인하려면, VRF 출력이 제안자의 지분 비율에 따라 결정된 임계값보다 작아야 한다. 하지만 미나와 같은 간결한 프로토콜에서는 과거의 원장을 보관하거나 재구성하는 것이 쉽지 않다. 비 간결형 블록체인과 달리, 노드는 과거의 임의 지점에서 데이터를 요청해 필요한 정보를 재구성할 수 없다. 따라서 이를 단순히 구현하면, 전체 원장의 세 복사본을 항상 유지해야 하고 최소 두 개의 전체 에폭이 지나야 해당 정보를 사용할 수 있게 된다. 하지만 적절한 설계를 통해 훨씬 효율적으로 구현할 수 있다.

Ouroboros와 미나의 차이는 블록 승자(block winners) 증명 방식에 있다. Ouroboros에서는 각 노드가 블록을 수신할 때 제안자가 실제로 블록을 획득했는지 확인해야 하므로, 해당 공개키의 잔액을 직접 조회해야 한다. 반면 미나에서는 VRF 평가의 정확성을 SNARK 내에서 증명하므로, 다른 노드는 SNARK 검증만으로 블록이 정당한 승자에 의해 제안되었음을 확인할 수 있다. 제안자가 직접 이를 증명하므로, 자신과 자신에게 위임된 계정의 계정 기록(account record) 및 머클 경로만 저장하면 충분하다. 또한 Ouroboros는 최종성(finality)이 두 에폭 이내에 도달함을 보장하므로, 제안자는 이 정보를 최종 확정 이후에만 캡처해 보관 데이터를 줄일 수 있다. 이 정보는 SNARK로 전달되어 다음을 증명한다: 1) 제공된 공개키에 대한

VRF 평가가 정확함, 2) 해당 공개키의 계정이 에폭 원장에 존재하며, 해당 잔액이 VRF 출력보다 큰 VRF 임계값을 형성함 (머클 경로를 통해 에폭 원장의 루트를 증명). 이는 여전히 제안자가 필요한 정보를 저장하기 위해 일정 시간 온라인 상태를 유지해야 한다는 문제를 완전히 해결하지는 못하지만, 다른 접근 가능성을 열어준다. 예를 들어, 제안자는 특정 에폭 원장에서 자신의 계정 기록과 머클 경로를 요청할 수 있다. 현재 구현에서는 이를 저장하거나 제공하는 노드가 없지만, 향후 이러한 정보를 유료로 제공해 제안자가 더 빨리 온라인으로 전환할 수 있게 하는 서비스가 생길 가능성이 있다.

블록의 구성 (What's in a Block?)

블록은 네트워크의 상태를 확장하는 트랜잭션과 합의 정보의 집합이다. 미나의 블록에는 현재 네트워크 상태가 완전히 유효함을 증명하는 증거가 포함된다. 미나의 블록은 다음 요소들로 구성된다:

- [프로토콜 상태 \(Protocol state\)](#)
- [프로토콜 상태 증명 \(Protocol state proof\)](#)
- [스테이징 원장 차이 \(Staged ledger diff\)](#)
- [델타 전이 체인 증명 \(Delta transition chain proof\)](#)
- 현재 프로토콜 버전
- 제안된 프로토콜 버전

노드가 피어로부터 블록을 수신하면, 먼저 이를 검증한 후 기존 상태에 적용하고 노드의 전이 프런티어 (transition frontier)에 추가한다. [합의 규칙](#)에 따라 블록체인의 길이가 증가하면, 노드는 최적의 팁 (best tip)을 업데이트하고 프런티어의 루트를 이동시켜 전이 프런티어에 \$k\$개의 블록만 유지한다.

팁: 미나에서 블록은 “전이(transition)”와 동의어다.

피어로부터 수신된 전이는 외부 전이(external transition)라 부르며, 로컬에서 생성되고 적용된 전이는 내부 전이(internal transition)라 부른다.

프로토콜 상태 (Protocol State)

프로토콜 상태는 이전 프로토콜 상태 해시와 그 본문(body)으로 구성되며, 본문은 다음을 포함한다:

- [제네시스 상태 해시 \(Genesis state hash\)](#)
- [블록체인 상태 \(Blockchain state\)](#)
- [합의 상태 \(Consensus state\)](#)
- [합의 상수 \(Consensus constants\)](#)

각 블록은 이전 블록의 프로토콜 상태 해시를 포함하며, 이를 통해 블록들이 연결되어 변경 불가능한 체인을 형성한다. 프로토콜 상태 해시는 이전 상태와 본문의 해시를 결합하여 계산되며, 각 블록의 고유 식별자로 작용한다.

제네시스 상태 해시 (Genesis State Hash)

제네시스 상태 해시는 제네시스 프로토콜 상태에 대한 프로토콜 상태 해시이다.

블록체인 상태 (Blockchain State)

블록체인 상태는 다음 요소들로 구성된다:

- 스테이징 원장 해시 (Staged ledger hash)
- 제네시스 원장 해시 (Genesis ledger hash)
- 원장 증명 명세 (Ledger proof statement)
- 타임스탬프 (Timestamp)
- 본문 참조 (Body reference)

원장 증명 명세 (Ledger Proof Statement)

원장 증명 명세는 다음 요소들로 구성된다:

- SNARK된 원장 해시 (Snarked ledger hash)
- 서명된 금액 (Signed amount)
- 보류 중인 코인베이스 스택 (Pending coinbase stack)
- 수수료 초과분 (Fee excess)
- SOK 다이제스트 (Sok digest)
- 로컬 상태 (Local state)

합의 상태 (Consensus State)

합의 상태는 다음 요소들로 구성된다:

- 블록체인 길이 (Blockchain length)
- 에폭 카운트 (Epoch count)
- 최소 윈도우 밀도 (Min window density)
- 서브 윈도우 밀도 (Sub window density)
- 마지막 VRF 출력 (Last VRF output)
- 총 통화량 (Total currency)
- 현재 글로벌 슬롯 (Current global slot)
- 제네시스 이후 글로벌 슬롯 (Global slot since genesis)
- 스테이킹 에폭 데이터 (Staking epoch data)
- 다음 에폭 데이터 (Next epoch data)
- 동일 체크포인트 윈도우 내 조상 존재 여부 (Has ancestor in same checkpoint window)
- 블록 스테이크 승자 (Block stake winner)
- 블록 생성자 (Block creator)
- 코인베이스 수신자 (Coinbase receiver)
- 코인베이스 슈퍼차지 여부 (Supercharge coinbase)

합의 상수 (Consensus Constants)

합의 상수는 합의 파라미터를 정의한다:

- k
- delta
- slots_per_sub_window
- slots_per_window

- sub_windows_per_window
- slots_per_epoch
- grace period slots
- grace period end
- checkpoint window slots per year
- checkpoint window size in slots
- block window duration ms
- slot duration ms
- epoch duration
- delta duration
- genesis state timestamp

프로토콜 상태 증명 (Protocol State Proof)

프로토콜 상태 증명은 블록 생성자가 생성한 새로운 프로토콜 상태가 유효함을 검증하는 블록체인 증명이다. 재귀 SNARK를 사용하기 때문에, 이 증명은 체인의 전체 이력이 유효함을 함께 증명한다.

스테이징 원장 차이 (Staged Ledger Diff)

블록 생성자가 블록 생성 슬롯을 획득하면, 트랜잭션 풀과 SNARK 풀에서 필요한 트랜잭션과 SNARK 작업을 선택한다. 이후 블록체인의 다음 상태를 제안하기 위해 스테이징 원장의 차이(diff)를 생성한다. 이 차이는 다음 요소들로 구성된다:

- 블록에 포함된 트랜잭션
- 이전 트랜잭션에 대해 SNARK 작업자가 생성한 SNARK 증명 목록
- 보류 중인 코인베이스 (Pending coinbase)

스테이징 원장은 트랜잭션, 결제, 코인베이스, 그리고 SNARK 증명이 아직 생성되지 않은 수수료 결제 내역을 포함하는 대기 계정 데이터베이스로 볼 수 있다. 스테이징 원장은 계정 상태(원장)와 SNARK 증명이 없는 트랜잭션을 위한 트랜잭션 큐로 구성되며, 이는 스캔 상태 (Scan State)라 불린다.

델타 전이 체인 증명 (Delta Transition Chain Proof)

네트워크 전반에 새로 생성된 블록을 전파할 때, 불리한 네트워크 상태를 고려해 일정한 네트워크 지연이 허용된다. 델타 전이 체인 증명은 블록이 허용된 슬롯 시간 내에 생성되었음을 증명한다.

예시 블록 (Example Block)

```
{
  "external_transition": {
    "protocol_state": {
      "previous_state_hash":
      "3NLKJLNbD7rBAbGdjZz3tfNBPYxUJJJaLmwCP9jMKR65KSz4RKV6b",
      "body": {
```

```

    "genesis_state_hash":
"3NLxYrjb7zmHdoFgBrubCN8ijM8v7eT8kvLiPLc9DHt3M8XrDDEG",
    "blockchain_state": {
        "staged_ledger_hash": {
            "non_snark": {
                "ledger_hash":
"jxV4SS44wHUVrGEucCsfXLisZyUC5QddsiokGH3kz5xm2hJWZ25",
                "aux_hash":
"UmosfM82dH5xzqdcKXgA1JoAvJ5tLxch2wsty4sXmiEPKnPTPq",
                "pending_coinbase_aux":
"WLo8mDN6oBUTSyBkFCy7Fky7Na5fN4R6oGq4HMf3YoHCAj4cwY"
            },
            "pending_coinbase_hash":
"2mze7iXKwA9JAqVDC1MVvgWfJDgvgbSexKtuShdkgqMfv1tjATQQ"
        },
        "ledger_proof_statement": {
            "connecting_ledger_right":
"jwbhUympjiAFLPi7w3Cg9GEEncACEpPxrTp45XZt3BDA5UNV8S",
            "sok_digest": null,
            "target": {
                "local_state": {
                    "full_transaction_commitment":
"0x0000000000000000000000000000000000000000000000000000000000000000",
                    "call_stack":
"0x0000000000000000000000000000000000000000000000000000000000000000",
                    "token_id":
"wSHV2S4qX9jFsLjQo8r1BsMLH2ZRKsZx6EJd1sbozGPieEC4Jf",
                    "excess": {
                        "sgn": [
                            "Pos"
                        ],
                        "magnitude": "0"
                    },
                    "success": true,
                    "stack_frame":
"0x0641662E94D68EC970D0AFC059D02729BBF4A2CD88C548CCD9FB1E26E570C66C",
                    "will_succeed": true,
                    "account_update_index": "0",
                    "supply_increase": {
                        "sgn": [
                            "Pos"
                        ],
                        "magnitude": "0"
                    }
                },
            }
        },
    }

```

```

        "ledger":
"jw6bz2wud1N6itRUHZ5ypo3267stk4UgzkiuWtAMPRZo9g4Udyd",
        "failure_status_tbl": [],
        "transaction_commitment":
"0x0000000000000000000000000000000000000000000000000000000000000000",
        },
        "pending_coinbase_stack": {
            "state": {
                "init":
"4Yx5U3t3EYQycZ91yj4478bHkLwGkhDHnPbCY9TxgUk69SQityej",
                "curr":
"4Yx5U3t3EYQycZ91yj4478bHkLwGkhDHnPbCY9TxgUk69SQityej"
            },
            "data":
"4QNrZFBTDQCPfEZqBZsaPYx8qdaNFv1nebUyCUsQW9QUJqyuD3un"
        },
        "first_pass_ledger":
"jwbhUympjiAFLPi7w3Cg9GEeNCfACEpPxrTp45XZt3BDA5UNV8S",
        "second_pass_ledger":
"jwbhUympjiAFLPi7w3Cg9GEeNCfACEpPxrTp45XZt3BDA5UNV8S"
    },
    "fee_excess": [
        {
            "token":
"wSHV2S4qX9jFsLjQo8r1BsMLH2ZRKsZx6EJd1sbozGPieEC4Jf",
            "amount": {
                "sgn": [
                    "Pos"
                ],
                "magnitude": "0"
            }
        },
        {
            "amount": {
                "sgn": [
                    "Pos"
                ],
                "magnitude": "0"
            },
            "token":
"wSHV2S4qX9jFsLjQo8r1BsMLH2ZRKsZx6EJd1sbozGPieEC4Jf"
        }
    ],
    "source": {

```

```

        "second_pass_ledger":
"jwbhUympjiAFLPi7w3Cg9GEeNCfACEpPxrTp45XZt3BDA5UNV8S",
        "local_state": {
            "account_update_index": "0",
            "supply_increase": {
                "sgn": [
                    "Pos"
                ],
                "magnitude": "0"
            },
            "stack_frame":
"0x0641662E94D68EC970D0AFC059D02729BBF4A2CD88C548CCD9FB1E26E570C66C",
            "ledger":
"jw6bz2wud1N6itRUHZ5ypo3267stk4UgzkiuWtAMPRZo9g4Udyd",
            "transaction_commitment":
"0x0000000000000000000000000000000000000000000000000000000000000000",
            "token_id":
"wSHV2S4qX9jFsLjQo8r1BsMLH2ZRKsZx6EJd1sbozGPieEC4Jf",
            "excess": {
                "sgn": [
                    "Pos"
                ],
                "magnitude": "0"
            },
            "call_stack":
"0x0000000000000000000000000000000000000000000000000000000000000000",
            "will_succeed": true,
            "full_transaction_commitment":
"0x0000000000000000000000000000000000000000000000000000000000000000",
            "failure_status_tbl": [],
            "success": true
        },
        "first_pass_ledger":
"jwbhUympjiAFLPi7w3Cg9GEeNCfACEpPxrTp45XZt3BDA5UNV8S",
        "pending_coinbase_stack": {
            "data":
"4QNrZFBTDQCPfEZqBZsaPYx8qdaNFv1nebUyCUsQW9QUJqyuD3un",
            "state": {
                "curr":
"4Yx5U3t3EYQycZ91yj4478bHkLwGkhDHnPbCY9TxgUk69SQityej",
                "init":
"4Yx5U3t3EYQycZ91yj4478bHkLwGkhDHnPbCY9TxgUk69SQityej"
            }
        }
    },

```

```

    "supply_increase": {
      "sgn": [
        "Pos"
      ],
      "magnitude": "0"
    },
    "connecting_ledger_left":
"jwbhUympjiAFLPi7w3Cg9GEeNCfACEpPxrTp45XZt3BDA5UNV8S"
  },
  "body_reference":
"b94b2580ca80f27c9655289579a0d71df0b7604dfa7c404e6c309cccf7730d2f",
  "snarked_ledger_hash":
"jx9171AbMApHNG1guAcKct1E6nyUFweA7M4ZPCjBZpgNNrE21Nj",
  "genesis_ledger_hash":
"jxX6VJ84HaafrKozFRA4qjnni4aPXqXC2H5vQLKSryNpKTXuz1R",
  "snarked_next_available_token": "2",
  "timestamp": "1611691710000"
},
"consensus_state": {
  "blockchain_length": "3852",
  "epoch_count": "1",
  "min_window_density": "1",
  "sub_window_densities": [
    "3",
    "1",
    "3",
    "1",
    "4",
    "2",
    "1",
    "2",
    "2",
    "4",
    "5"
  ],
  "last_vrf_output":
"g_1vrXSXLhvn1e4Ap1Ey5e8yh3PFMJT0vZyhZLlTBAA=",
  "total_currency": "16725580000001000",
  "curr_global_slot": {
    "slot_number": "12978",
    "slots_per_epoch": "7140"
  },
  "global_slot_since_genesis": "12978",
  "staking_epoch_data": {
    "ledger": {

```

```

        "hash":
"jxX6VJ84HaafrKozFRA4qjnni4aPXqXC2H5vQLKSryNpKTXuz1R",
        "total_currency": "165950000000001000"
    },
    "seed":
"2vb1Mjvydod6sEwn7qpbejKCfRqugMgyG3MHXXRKcAkwQLRs9fj8",
    "start_checkpoint":
"3NK2tkzqqK5spR2sZ7tujjqPksL45M3UUrcA4WhCkeiPtnugyE2x",
    "lock_checkpoint":
"3NK5G8Xqn1Prh3XoTyZ2tqntJC6X2nVwruv5mEJCL3GaTk7jKUNo",
    "epoch_length": "1769"
},
    "next_epoch_data": {
        "ledger": {
            "hash":
"jx7XXjRfJj2mGXmiHQmpm6ZgTxz14udpugyFtw4DefJFpie7apN",
            "total_currency": "166537000000001000"
        },
        "seed":
"2vavBR2GfJWvWkpC7yGJQFnts18nHaFjdVEr84r1Y9DQXvnJRhmd",
        "start_checkpoint":
"3NLdAqxtBRYxYbCWMXxGu6j1hGDrpQwGkBDF9QvGxmtpziXQDADu",
        "lock_checkpoint":
"3NL4Eis1pS1yrPdfCbiJcpCCYsHuXY3ZgEzHojPnFWfMK9gKmhZh",
        "epoch_length": "2084"
    },
    "has_ancestor_in_same_checkpoint_window": true,
    "block_stake_winner":
"B62qpBrUYW8SHcKTFWLbHKD7d3FqYFvGRBaWRLQCgsr3V9pwsPSd7Ms",
    "block_creator":
"B62qpBrUYW8SHcKTFWLbHKD7d3FqYFvGRBaWRLQCgsr3V9pwsPSd7Ms",
    "coinbase_receiver":
"B62qpBrUYW8SHcKTFWLbHKD7d3FqYFvGRBaWRLQCgsr3V9pwsPSd7Ms",
    "supercharge_coinbase": true
},
    "constants": {
        "k": "290",
        "slots_per_epoch": "7140",
        "slots_per_sub_window": "7",
        "delta": "0",
        "genesis_state_timestamp": "1609355670000"
    }
}
},
    "protocol_state_proof": "<opaque>",

```

```
    "staged_ledger_diff": "<opaque>",
    "delta_transition_chain_proof": "<opaque>",
    "current_protocol_version": "1.1.0",
    "proposed_protocol_version": "<None>"
  }
}
```

블록 생성자 (Block Producers)

미나에서 블록 생성자의 역할은 합의(consensus)를 달성하고 블록체인의 보안을 제공하는 것이다. 블록 생성자는 네트워크에서 전파된 최신 트랜잭션과, 체인의 현재 상태가 유효함을 증명하는 블록체인 증명을 포함하는 새로운 블록을 생성할 책임이 있다.

미나에서는 누구나 블록 생성자가 될 수 있다. 참여자 수에는 제한이 없으며, 블록 생성 확률은 스테이킹된 자금의 크기에 비례한다. 스테이킹된 자금은 락업되지 않으며 슬래싱(slash)의 대상이 되지 않는다.

스테이킹된 자금과 필요한 블록체인 증명을 생성하는 대가로, 정식 체인(canonical chain)에 포함된 블록은 코인베이스와 트랜잭션 수수료의 형태로 보상을 받는다. 단, 필요한 트랜잭션 SNARK 작업을 구입하는데 지불된 수수료는 제외된다.

블록을 성공적으로 생성하기 위해서는 블록 생성자가 현재 블록체인의 상태를 보유해야 한다. 또한 슬롯 시간 내에 블록체인 SNARK를 생성할 수 있는 충분한 연산 자원과, 생성된 블록을 네트워크 합의 파라미터에서 정의된 허용 지연 내에 피어에게 브로드캐스트할 수 있는 네트워크 연결이 필요하다.

블록 생성자 선정 (Select a Block Producer)

각 슬롯에서 블록을 생성할 수 있는 기회는 검증 가능한 난수 함수(VRF, Verifiable Random Function)에 의해 결정된다. 이 함수는 복권과 유사하게 작동한다. 각 블록 생성자는 각 슬롯마다 독립적으로 VRF를 실행하며, 출력값이 자신의 스테이크 비율에 따라 정해진 임계값보다 크면 해당 슬롯에서 블록을 생성할 기회를 얻게 된다.

이 과정은 비공개로 수행되며, 개인 키 보유자만 VRF 출력을 계산할 수 있고 언제 자신이 블록을 생성할 차례인지 알 수 있다. 이 선택 방식은 특정 슬롯의 블록 생성자를 공격(예: DoS나 표적 공격)할 수 없도록 해 보안을 강화한다. 따라서 동일한 슬롯에 여러 생성자가 선택될 수 있으며, 여러 생성자가 같은 슬롯에서 유효한 블록을 생성할 경우 단기 포크(short-range fork)가 발생한다. 이때 합의 규칙에 따라 가장 긴 체인이 선택된다.

스테이크 분포는 현재 에폭 - 2의 마지막 블록에서 SNARK된 원장을 기준으로 결정된다. 따라서 새로 획득했거나 위임된 스테이크는 반영되기까지 지연이 있다. 예를 들어 현재 에폭이 10이라면, 스테이킹 분포는 8번째 에폭의 마지막 블록의 SNARK된 원장을 기준으로 결정된다.

VRF의 출력을 로그에서 확인하려면 Checking VRF evaluations 항목을 찾아보면 된다.

블록 생성 (Generating a Block)

블록 생성자가 특정 슬롯에서 블록을 생성하도록 선택되면 다음 과정을 수행한다:

- 자신의 전이 프런티어(transition frontier, 로컬 블록 저장소)에서 현재 최적의 팁(best tip)을 선택해 새로운 블록을 구축한다.
- 트랜잭션 풀과 SNARK 풀에서 트랜잭션 및 필요한 SNARK 작업을 선택한다.

블록 생성자는 블록에 추가하는 트랜잭션 수와 동일한 수의 SNARK 작업을 최소한 구입해야 한다. 또한 사용자 트랜잭션 외에 블록 생성 보상으로 코인베이스 트랜잭션을 추가하고, SNARK 작업자에게 지불할 수수료 전송(fee transfer)도 포함해야 한다.

- 블록체인의 다음 상태를 생성한다.
 - 계정 원장(account ledger)과 스캔 상태(아직 증명되지 않은 트랜잭션 큐)를 포함하는 스테이징 원장의 diff를 만든다.
 - 이 diff를 기존 스테이징 원장에 적용해 새로운 상태를 생성한다.
- 새로운 상태가 유효함을 증명하는 블록체인 증명을 생성한다.
 - 이 SNARK는 이전 프로토콜 상태 증명도 함께 검증한다.
- 네트워크 합의 파라미터에서 정의된 허용 네트워크 지연 내에 블록이 생성되었음을 증명하는 델타 전이 체인 증명(delta transition chain proof)을 생성한다.
- 새로 생성된 상태를 로컬에 적용하고 기존 전이 프런티어에 추가한다.
- 생성된 블록을 피어에게 브로드캐스트(외부 전이 호출, external transition)한다.

지분 위임 (Stake Delegation)

위임된 자금은 송금할 수 없으며, 언제든지 원래 계정으로 지분을 다시 위임(re-delegation)함으로써 위임을 해제할 수 있다.

SNARK 작업자 (SNARK Workers)

대부분의 프로토콜은 하나의 주요 노드 운영자 그룹(예: 채굴자, 검증자, 블록 생성자)만 존재하지만, 미나에는 두 번째 그룹인 **SNARK 작업자(SNARK Workers)**가 있다.

SNARK 작업자는 네트워크 내 트랜잭션의 SNARK 증명을 생성(snarking)하는 역할을 담당하므로, 미나 네트워크의 건전성을 유지하는 데 필수적이다. 이들은 트랜잭션의 SNARK 증명을 생성함으로써 미나 블록체인의 간결성을 유지한다.

이 섹션에서는 SNARK 작업자가 필요한 이유, 경제적 인센티브 구조, 그리고 SNARK 작업의 운영 세부 사항을 다룬다.

참고: 이 문서에서는 zk-SNARK의 이론적 배경은 다루지 않는다. SNARK의 깊은 지식은 필수는 아니지만, 기본 원리와 활용 목적을 이해하면 도움이 된다. 자세히 알고 싶다면 [What are zk-SNARKs?](#) 입문서를 참고하라.

SNARK 작업자가 필요한 이유 (Why SNARK Workers?)

미나의 가장 큰 특징은 간결한 블록체인(succinct blockchain)이다.

각 블록 프로듀서는 네트워크에 새 블록을 제안할 때 반드시 해당 블록과 함께 zk-SNARK를 포함해야 한다. 이 덕분에 노드는 최종화된 과거 데이터를 모두 삭제하고 SNARK만 유지할 수 있다. 미나 프로토콜에 익숙하지 않다면 [이 영상](#)을 참고하라.

하지만 미나의 블록 생성자가 블록에 대한 SNARK 증명을 생성하는 것만으로는 충분하지 않다. 트랜잭션 자체도 SNARK 증명이 필요하다. 왜냐하면 블록체인 SNARK는 블록에 포함된 개별 트랜잭션의 유효성을 직접 증명하지 않기 때문이다.

예를 들어, 현재 블록체인의 헤드 상태 해시가 a6f8792226...이고, 새 블록의 상태 해시가 ff6f2d8ef4...인 경우를 생각해보자. 이 블록에는 블록 생성자가 선택한 트랜잭션과 관련 메타데이터가 포함되어 있다. 또한 다음 명제를 검증하는 SNARK가 함께 제공된다:

“상태 해시 `offd2f84f`를 가진 블록이 존재하며, 이는 상태 해시 `a6f8792226`을 가진 이전 최적 틱을 확장한다.”

이 문장은 새로운 블록에 포함된 트랜잭션의 유효성에 대해서는 아무것도 말하지 않는다. 따라서 이를 그대로 신뢰한다면 악의적인 블록 프로듀서에 의해 속을 수도 있다. 운 좋게도 우리는 원본 블록을 받아 각 트랜잭션의 유효성을 직접 검증할 수 있지만, 모든 노드가 각 블록을 재검증하지 않고도 상태를 신뢰할 수 있으려면 트랜잭션에 대한 별도의 SNARK가 필요하다.

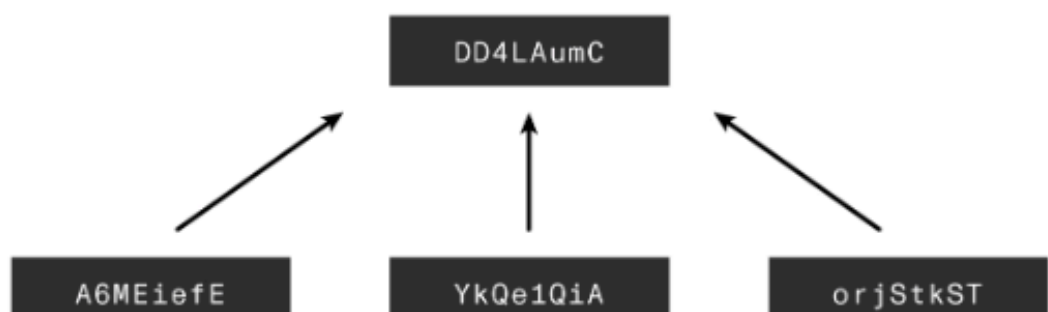
트랜잭션 SNARK 생성 (Snarking Transactions)

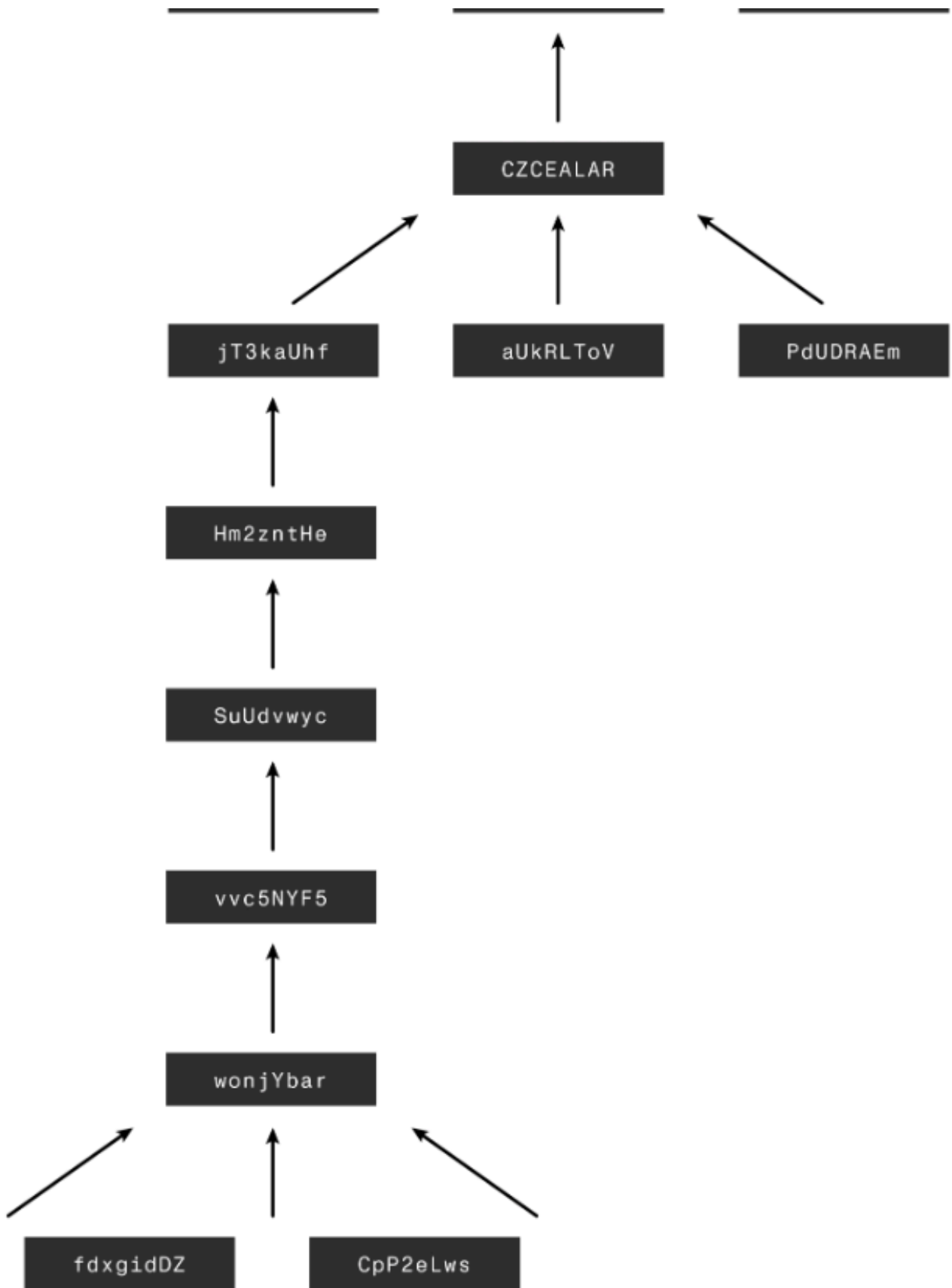
모든 노드가 트랜잭션 재실행 없이 미나 블록체인의 상태를 검증하려면, 블록체인 SNARK만으로는 충분하지 않으며, 트랜잭션도 유효함을 증명해야 한다. 따라서 각 트랜잭션에 대해 SNARK를 생성하고 이를 결합하는 접근이 필요하다.

하지만 SNARK 증명 생성은 계산 비용이 매우 높다. 각 트랜잭션마다 순차적으로 SNARK를 생성한다면 처리량은 매우 낮아지고 블록 시간이 급격히 길어진다. 게다가 실제 환경에서는 트랜잭션이 비동기적으로 도착하기 때문에 언제 다음 증명을 수행해야 할지 예측하기 어렵다.

다행히 SNARK의 두 가지 중요한 성질을 활용할 수 있다:

1. **증명 병합 가능성** — 두 SNARK 증명은 하나의 병합된 증명(merged proof)으로 결합할 수 있다.
2. **병합의 결합성** — 병합 순서에 관계없이 결과는 동일하다.





이 두 속성 덕분에 병렬 처리가 가능해진다. 증명들을 병합할 수 있고 순서가 중요하지 않다면, SNARK 증명은 병렬로 생성될 수 있다. 먼저 완료된 증명은 이후 생성 중인 증명들과 결합할 수 있다. 이는 마치 이진 트리 구조처럼 볼 수 있으며, 하단의 리프 노드는 개별 트랜잭션 증명이고, 상위 노드로 올라가며 병합 증명들이 쌓인다. 루트 노드는 모든 트랜잭션을 적용한 상태 갱신을 나타낸다.

또한 SNARK 증명들이 서로 독립적이므로 병렬성을 활용할 수 있다는 점은, 누구나 이 작업에 참여할 수 있음을 의미한다. 즉, 분산된 작업 풀이 허가 없이 운영될 수 있다. 여분의 연산 자원을 가진 누구나 SNARK 작업자로 참여해, SNARK가 필요한 트랜잭션을 관찰하고 계산 자원을 제공할 수 있다. 물론, 그들은 작업 대가를 **스나켓플레이스(snarketplace)** 라 불리는 시장에서 보상받는다.

참고: SNARK 작업 체계의 발전 과정을 더 자세히 알고 싶다면, [High Throughput with Slow Snarks](#). 영상을 추천한다. 함수형 프로그래밍이나 앞서 설명한 트리 구조인 스캔 상태(scan state)의 세부 구현에 관심이 있다면, [이 영상](#)에서도 기술적 내용을 다루고 있다.

스나켓플레이스 (The Snarketplace)

SNARK 작업의 핵심 역학 관계는 다음과 같다:

블록 생성자는 자신의 블록 보상으로 SNARK 작업자로부터 SNARK 작업을 구입한다.

프로토콜은 SNARK의 가격 책정에 관여하지 않으며, SNARK 작업자가 SNARK를 생성해도 프로토콜 수준의 보상은 없다. 인센티브는 전적으로 피어 투 피어(P2P) 구조로 형성되며, 공개 시장인 **스나켓플레이스(snarketplace)** 에서 동적으로 결정된다.

그렇다면 왜 블록 생성자가 SNARK를 구입해야 할까? 앞서 설명했듯이, 미나 블록체인의 상태가 유효함을 확실히 하려면 모든 트랜잭션이 SNARK 증명되어야 한다. 하지만 SNARK 생성 속도보다 트랜잭션이 빠르게 추가된다면, 미처 처리되지 못한 작업이 쌓이게 된다. 이를 방지하기 위해서는 SNARK 생성 속도가 트랜잭션 추가 속도와 비슷해야 한다.

블록 생성자는 블록에 트랜잭션을 포함함으로써 트랜잭션 수수료와 코인베이스 보상을 얻기 때문에, 이 트랜잭션을 상쇄하기 위해 동일 수량의 SNARK 작업을 구입해야 한다. 이로써 SNARK 작업에 대한 수요가 형성된다. 반면 SNARK 작업자는 자신들의 SNARK 작업을 가능한 한 높은 가격에 판매하길 원한다. 이 두 집단이 시장의 양쪽 역할을 수행하며, 시간이 지나면서 SNARK 작업의 시장 가격이 형성되고 균형을 이룬다.

SNARK 작업 가격 책정 방법 (How to Price SNARK Work)

스나켓플레이스(snarketplace)는 **수요와 공급**의 기본 법칙에 따라 동적으로 균형을 이루게 될 것으로 예상된다. 각 SNARK 작업은 서로 다른 트랜잭션에 적용되지만, 큰 관점에서 보면 SNARK 작업은 거의 **동질적인 상품(commodity)** 으로 볼 수 있다. 즉, 어떤 SNARK 작업자가 작업을 수행하든 결과물은 동일하다는 의미다. 다만, 몇 가지 미묘한 차이가 있으므로 가격 책정 전략에 대한 휴리스틱을 고려하는 것이 좋다:

- 시장 가격이 X라면, 운영비를 감안했을 때 수익이 남는 한 X 이하(예: $X \times 0.9$) 가격으로 SNARK 작업을 판매하는 것이 효과적일 수 있다.
- 블록 생성자는 같은 SNARK 작업자로부터 더 많은 양의 SNARK 작업을 구매할 유인이 있다. 그 이유는 블록에 포함해야 하는 **수수료 전송(fee transfer)** 트랜잭션이 하나만 필요하기 때문이다.
 - 기본적으로 블록 프로듀서는 SNARK 워커에게 **수수료 전송(fee transfer)** 이라는 특수한 트랜잭션 형태로 보상한다. 블록 프로듀서의 목표는 이러한 수수료 전송의 개수를 최소화하는

것이다. 왜냐하면 각각의 수수료 전송은 별도의 트랜잭션으로 블록에 포함되어야 하고, 그만큼 더 많은 SNARK 작업이 필요하기 때문이다. 따라서 가장 이상적인 시나리오는 하나의 SNARK 작업자로부터 여러 SNARK 작업을 번들로 구매하는 것이다.

- 일부 SNARK 작업은 다른 작업보다 우선적으로 처리될 필요가 있다. 왜냐하면 특정 작업을 완료하면 전체 작업 트리의 메모리를 해제할 수 있기 때문이다 (자세한 내용은 위의 영상을 참고하라). 이를 위해 다양한 작업 선택 방식이 존재한다. 현재 기본적으로 지원되는 세 가지 방식은 순차적 (sequential), 무작위(random), 무작위 오프셋을 가진 순차적(sequential with random offset)이다. 다만, 이 방식들은 동적 시장의 장점을 충분히 활용하지 못하므로, 이는 미나 커뮤니티가 개선할 수 있는 여지로 남아 있다.

SNARK 및 가격 관련 데이터는 모두 공개되어 있으므로, 여러 방법으로 시장을 분석할 수 있다. 예를 들어 [GraphQL API](#)를 사용할 수 있으며, CLI나 SNARK 메모폴을 추적하는 맞춤형 솔루션을 구축할 수도 있다.

스나켓플레이스의 시장 역학에 대한 보다 구체적인 분석이 곧 공개될 예정이다.
또한 경제적 맥락을 보다 자세히 다루는 화이트페이퍼도 곧 발표될 것이다.

참고: [SNARKs 및 SNARK 워커 FAQ](#)

스캔 상태 (Scan State)

스캔 상태(scan state)는 트랜잭션 SNARK 생성 과정을 블록 생성자로부터 분리해 SNARK 작업자가 담당할 수 있도록 하는 데이터 구조이다.

블록 생성자는 트랜잭션 SNARK를 직접 생성할 필요가 없기 때문에, 트랜잭션 처리량이 증가하더라도 블록 생성 시간은 일정하게 유지된다. 이 데이터 구조 덕분에 트랜잭션 SNARK 증명 생성이 병렬화되어 여러 SNARK 작업자가 동시에 작업을 수행할 수 있다.

스캔 상태는 여러 개의 이진 트리(binary tree)로 구성된 포리스트(forest) 구조로, 각 트리의 노드는 SNARK 작업자가 수행해야 하는 개별 작업(job)을 나타낸다. 스캔 상태는 주기적으로 트리의 최상단에서 하나의 증명을 반환하며, 이는 트리 하단의 모든 트랜잭션이 올바르게 처리되었음을 증명한다.

블록 생성자는 자신이 생성하는 블록체인 SNARK 내에 이 원장 증명을 포함시켜, 체인의 현재 상태가 유효함과 함께 SNARK된 원장에 포함된 모든 트랜잭션의 유효성을 입증한다.

결과적으로 트랜잭션 처리량이 늘어나더라도 블록 생성 시간은 일정하게 유지될 수 있다. 스캔 상태는 원하는 트랜잭션 처리량에 맞춰 동적으로 조정될 수 있다.

팁: 모든 슬롯이 채워지고 필요한 증명이 완료된 정상 상태에서는, 각 블록마다 하나의 원장 증명이 생성된다.

트랜잭션 포함 (Including Transactions)

블록을 구성할 때, 블록 생성자는 스캔 상태 상수(scan state constants)에서 정의된 최대 트랜잭션 수까지 트랜잭션을 포함시킬 수 있다. 블록 생성자는 사용 가능한 트랜잭션 수수료를 가져가거나, 트랜잭션을 포함시켜 코인베이스 보상을 받을 수 있다. 추가된 각 트랜잭션은 새로운 기본 작업(base job)으로 변환되어 스캔 상태에 추가된다.

추가된 트랜잭션마다 블록 생성자는 스캔 상태에 이미 존재하는 작업(job) 시퀀스에 해당하는 동일한 양의 완료된 SNARK 작업을 포함해야 한다. 이 완료된 작업들이 스캔 상태에 추가되면, 루트 노드를 제외한 경우 새 병합 작업(merge job)을 생성하고, 루트 노드의 경우 최종 증명(proof)이 결과로 반환된다.

블록 생성자는 이 작업을 직접 수행하는 대신, SNARK 풀(snarketplace)에 등록된 입찰(bid)을 통해 원하는 SNARK 작업자로부터 완료된 작업을 구매할 수 있다.

스캔 상태 상수 (Scan State Constants)

다음 상수들은 스캔 상태의 구조와 동작을 정의한다:

- transaction_capacity_log_2
- work_delay

transaction_capacity_log_2 상수는 블록에 포함될 수 있는 최대 트랜잭션 수를 정의한다.

$$\text{max_no_of_transactions} = 2^{\{\text{transaction_capacity_log_2}\}}$$

work_delay는 SNARK 작업자가 SNARK 작업을 완료할 수 있도록 충분한 시간을 보장한다. 완료된 증명이 없으면 블록 생성자는 어떤 트랜잭션도 포함시킬 수 없다. work_delay 값에 따라 스캔 상태에 존재할 수 있는 최대 트리 수가 다음과 같이 정의된다:

$$\text{max_number_of_trees} = (\text{transaction_capacity_log_2} + 1) * (\text{work_delay} + 1) + 1$$

블록당 포함될 수 있는 최대 증명(proof) 수는 다음과 같이 정의된다:

$$\text{max_number_of_proofs} = 2^{\{\text{transaction_capacity_log_2} + 1\}} - 1$$

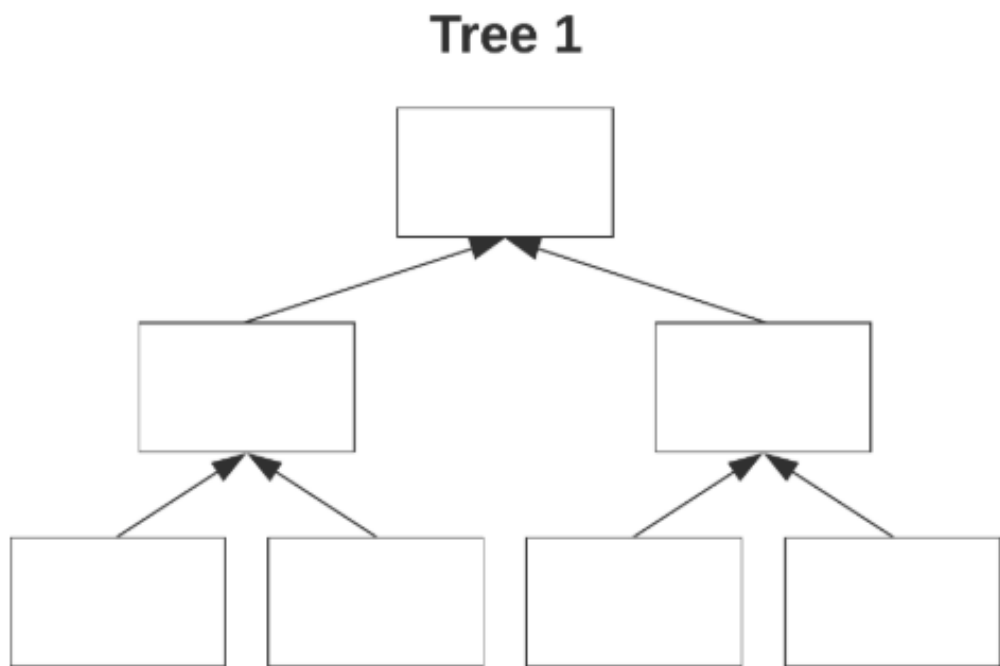
이러한 스캔 상태 제약 조건은 다음을 보장한다:

- 블록당 단 하나의 증명만 생성될 수 있다.
- 자식 증명을 추가한 후 업데이트되는 병합 노드(merge node)는 항상 비어 있다.

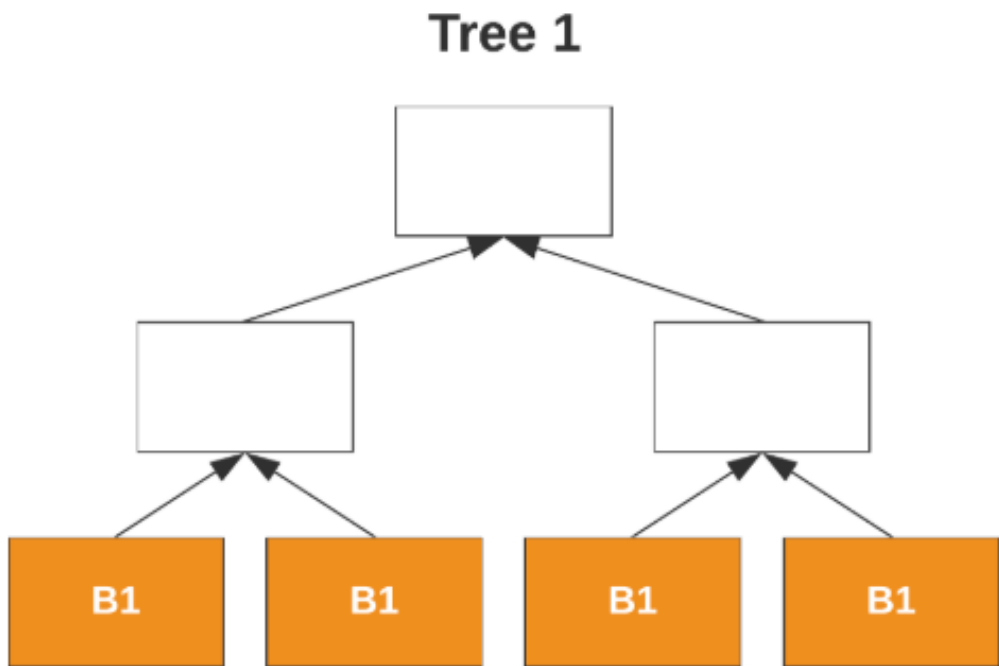
최대 트랜잭션 수는 고정될 수 있지만, 이 수치는 트랜잭션 처리량에 따라 동적으로 조정될 수 있다. 따라서 스캔 상태는 트랜잭션 증명 지연(latency)이 로그 형태로 증가하는 대가를 치르더라도, 사실상 무제한의 트랜잭션 처리량을 감당할 수 있다.

예시 (Example)

max_no_of_transactions = 4, work_delay = 1인 스캔 상태를 가정하자. 따라서 최대 작업량은 7이며, 최대 7개의 트리가 존재할 수 있다.
제네시스(genesis) 시점에는 스캔 상태가 비어 있다.

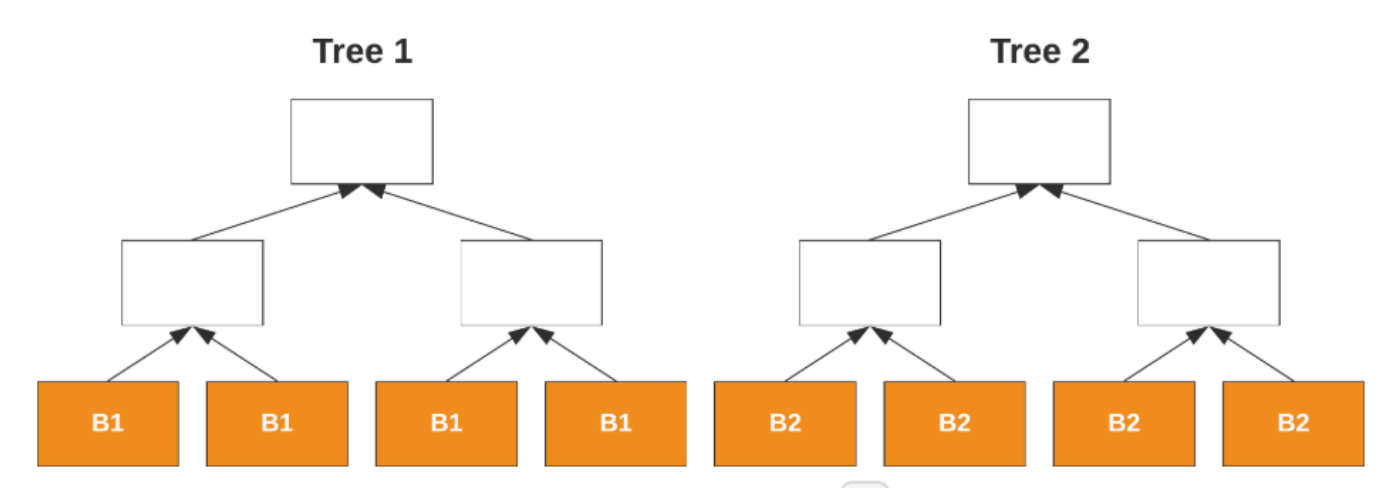


블록 1:
블록 생성자는 네 개의 트랜잭션을 스캔 상태에 포함하며, B1로 표시된 첫 번째 트리의 하단을 채운다.



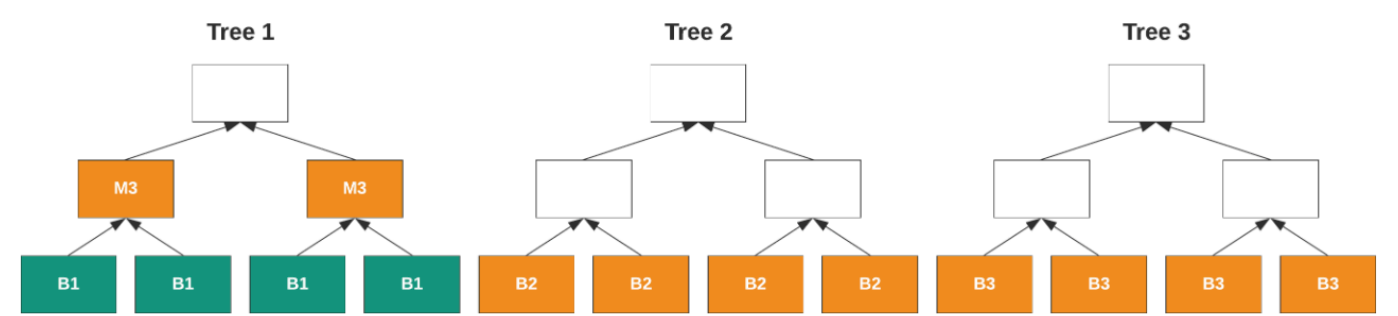
블록 2:

두 번째 블록에서 블록 생성자는 네 개의 트랜잭션(B2)을 추가한다. 이 트랜잭션들은 두 번째 트리의 하단을 채운다. `work_delay`가 1블록이기 때문에, 아직 증명이 필요하지 않다.



블록 3:

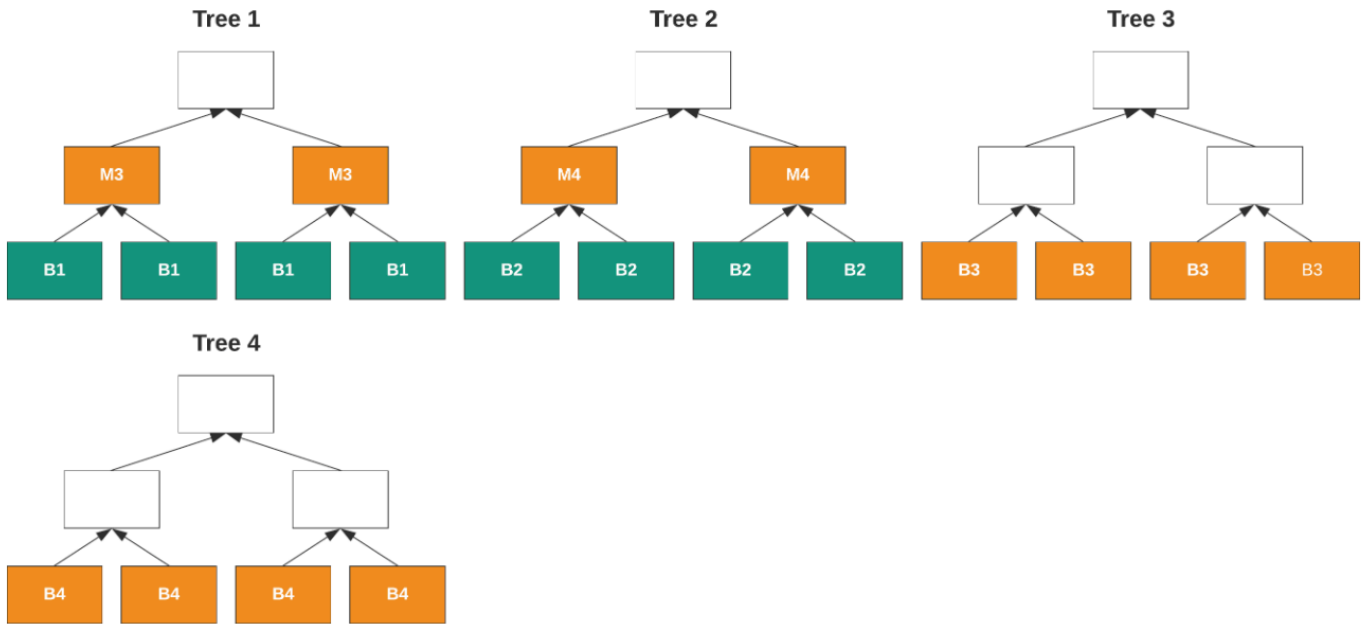
세 번째 블록에서 블록 생성자는 네 개의 B3 트랜잭션을 세 번째 트리에 추가하고, 첫 번째 트리에 대한 네 개의 증명을 포함해야 한다. 이 완료된 기본 증명(base proof)을 포함한 결과로 두 개의 새로운 M3 병합 작업이 생성된다.



팁: B 또는 M은 각각 기본 작업(Base job)과 병합 작업(Merge job)을 의미하며, 숫자는 스캔 상태에 추가된 순서를 나타낸다.

블록 4:

네 번째 블록에서 블록 생성자는 네 개의 트랜잭션(B4)을 네 번째 트리의 하단에 추가한다. 그리고 블록 2에서 추가된 작업에 해당하는 네 개의 증명을 포함해야 한다. 이 결과로 두 개의 M4 병합 작업이 새로 생성된다.

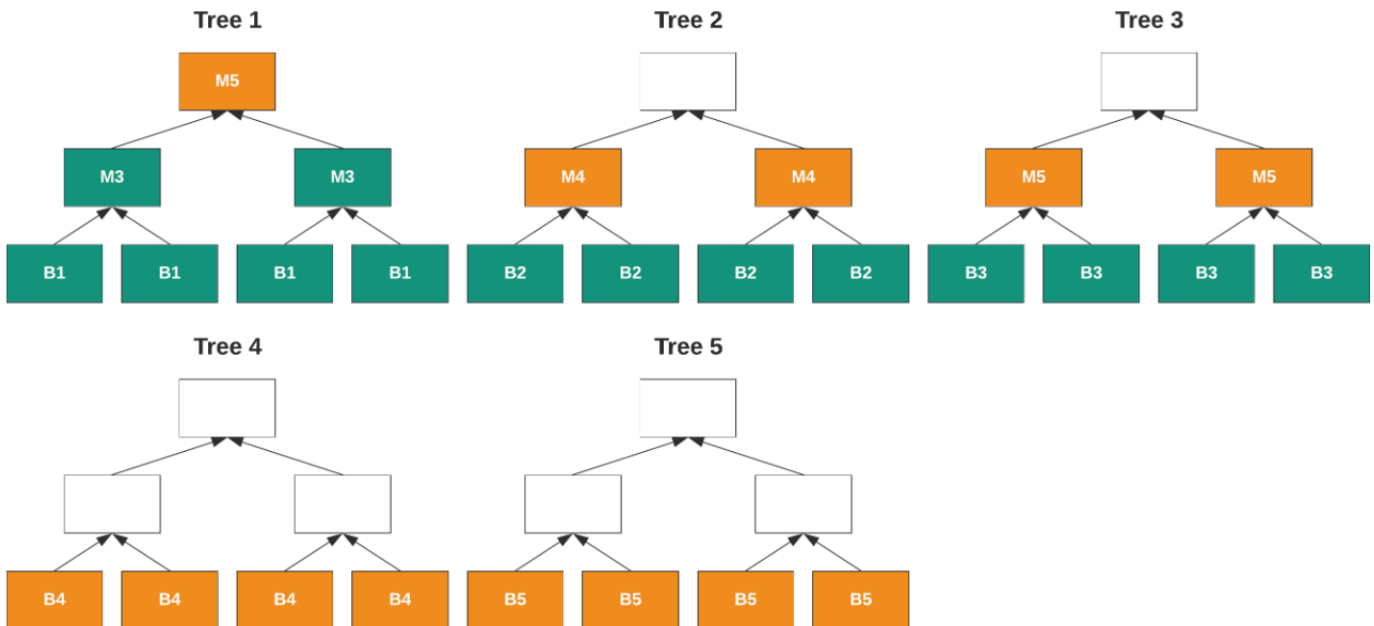


팁: 주황색으로 표시된 모든 보류 중인 작업은 SNARK 작업자가 수행해야 하는 작업이다. SNARK 작업자는 완료된 작업을 SNARK 풀에 제출한다. 여러 SNARK 작업자가 동일한 작업을 수행할 수 있지만, SNARK 풀에는 가장 낮은 수수료를 제시한 작업만 남아 블록 생성자가 구매할 수 있다.

블록 5:

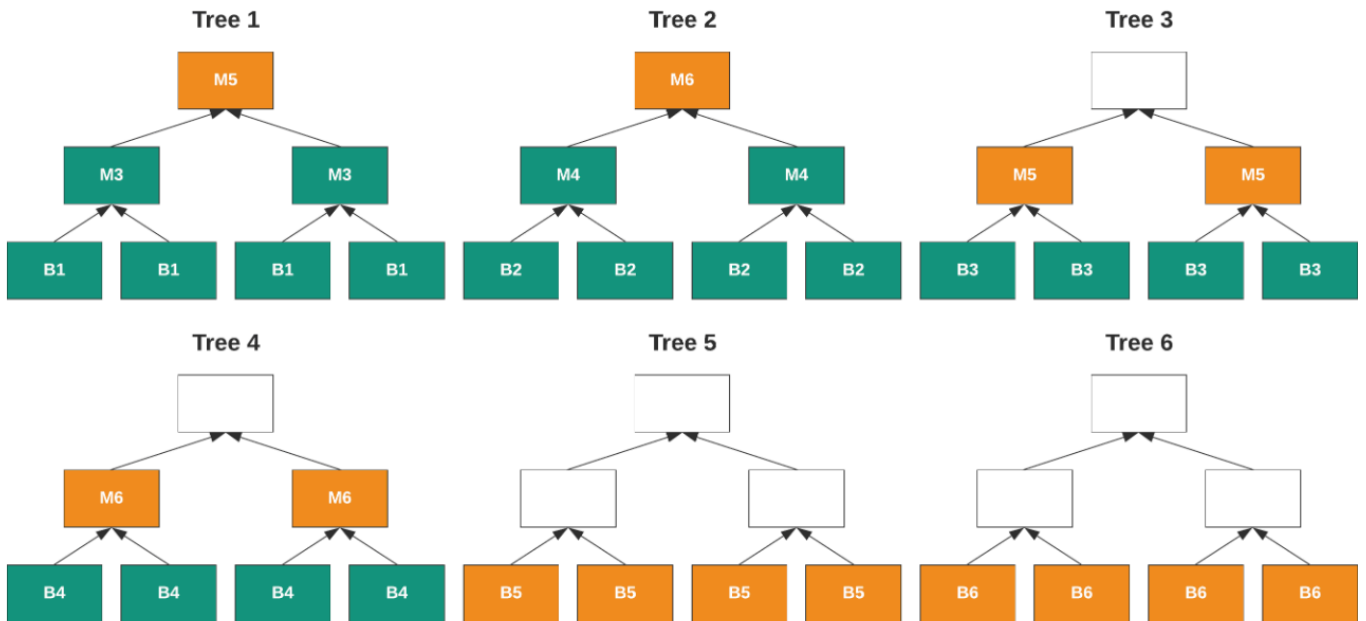
다섯 번째 블록에서는 다섯 번째 트리(B5)의 하단을 채우기 위해 네 개의 트랜잭션을 추가하고, 여섯 개의 증명(B3, M3)을 포함해야 한다.

이 M3 병합 작업 결과로 첫 번째 트리의 최종 보류 병합 작업(M5)이 생성된다.

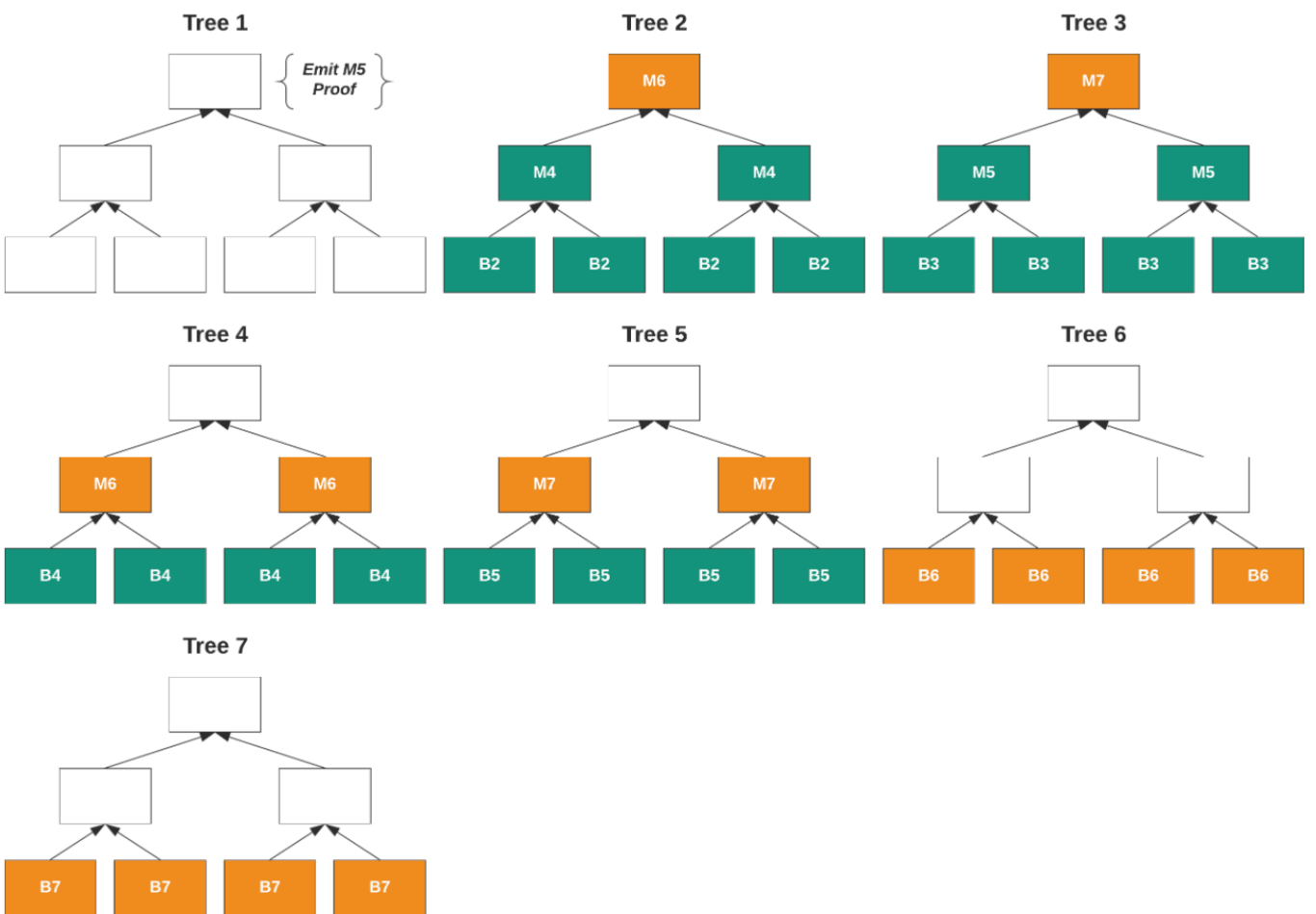


블록 6:

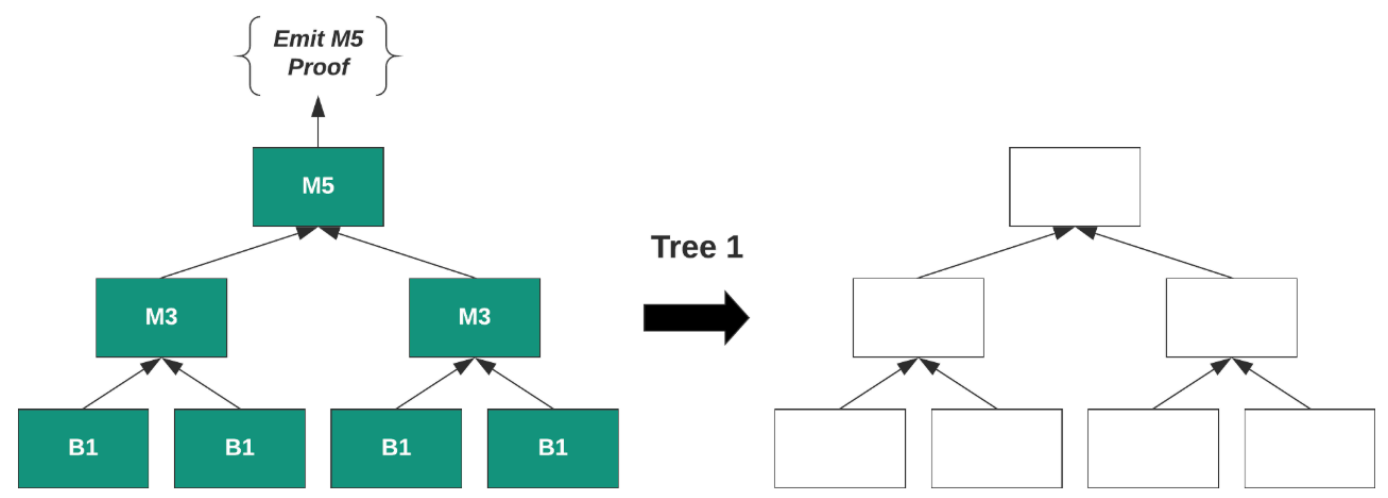
여섯 번째 블록에서 블록 생성자는 네 개의 트랜잭션(B6)을 추가해 여섯 번째 트리의 하단을 채운다. 이 블록에는 여섯 개의 증명(B4, M4)이 포함되며, 세 개의 새로운 병합 작업(M6)이 생성된다.



블록 7:
 일곱 번째 블록에서 블록 생성자는 네 개의 트랜잭션(B7)을 추가해 일곱 번째 트리의 하단을 채운다. 스캔 상태 상수에 따라 최대 트리 수는 7개이다. 이 블록에는 7개의 증명(B5, M5)이 포함되며, 이 증명으로 인해 세 개의 새로운 병합 작업(M7)이 생성되고, 동시에 최상단 M5 증명이 스캔 상태에서 방출(emitted)된다.

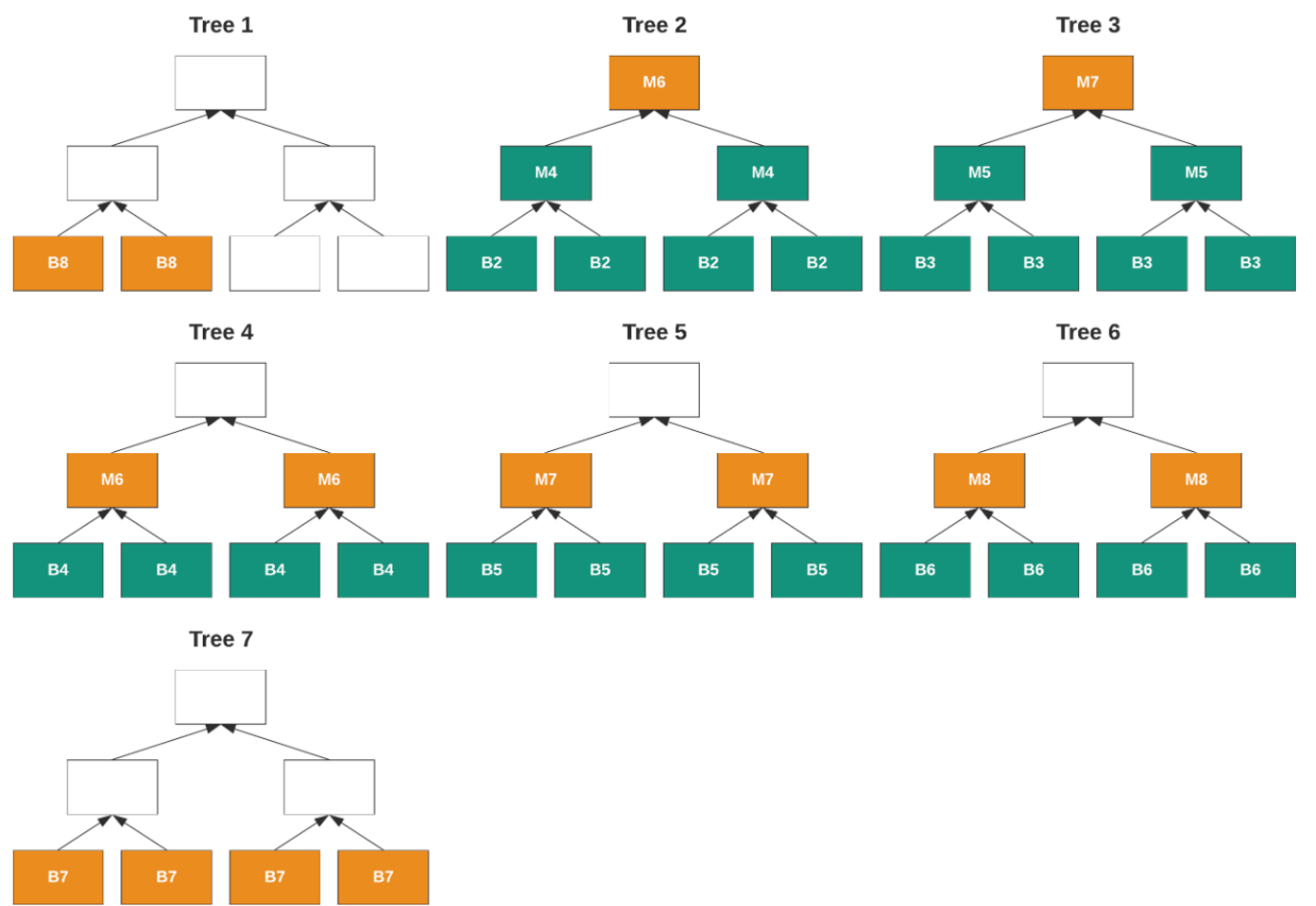


방출된 첫 번째 트리의 증명은 블록 1에 추가된 트랜잭션에 해당하는 원장 증명이다. 트리의 내용은 제거되어 새로운 트랜잭션을 위한 공간이 확보된다.



블록 8:

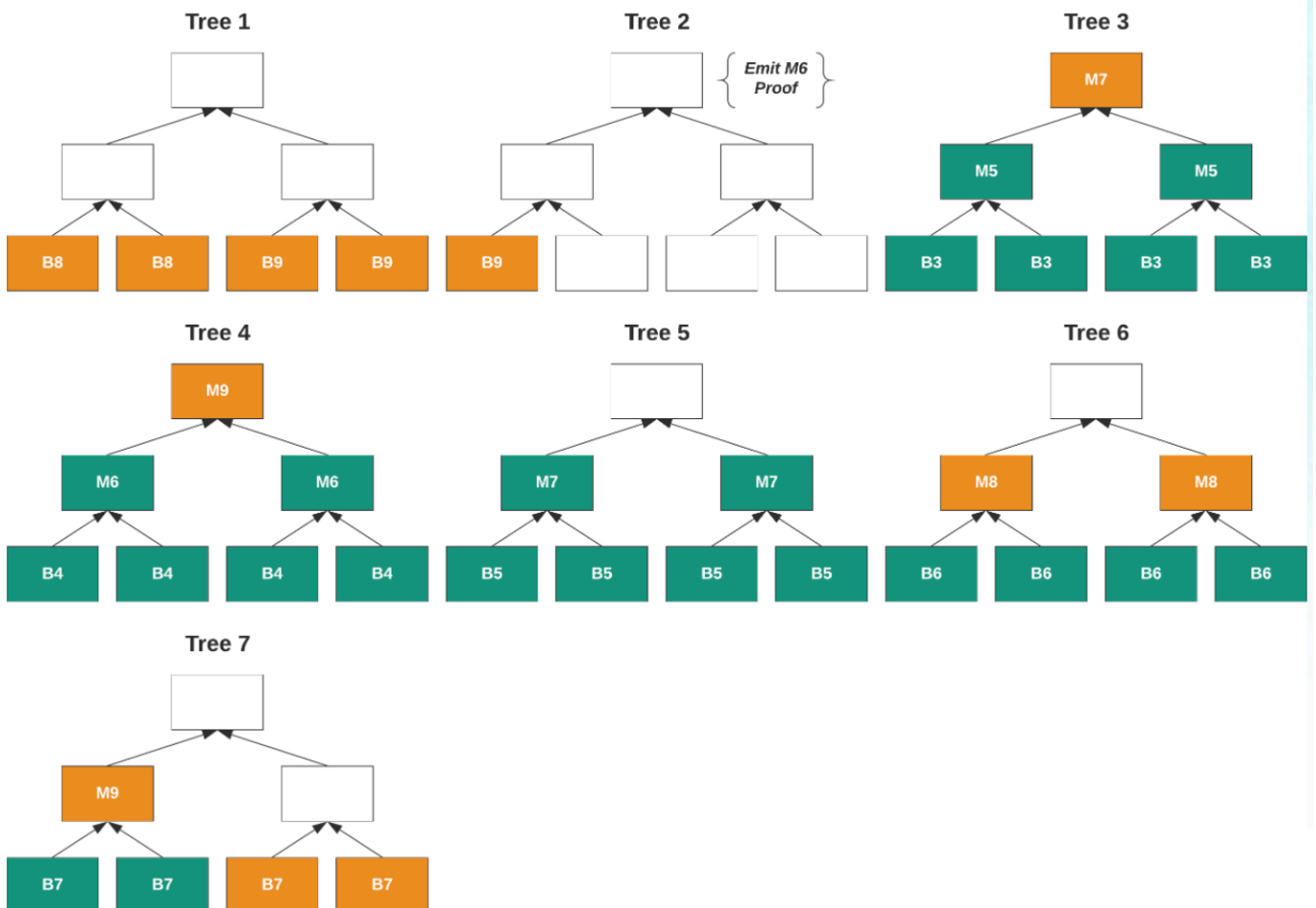
여덟 번째 블록에서 블록 생성자는 두 개의 트랜잭션(B8)을 추가하고, 네 개의 증명(B6)을 포함한다. 이 증명들로 인해 두 개의 새로운 병합 작업(M8)이 생성된다. 단, 트랜잭션이 두 개 추가되었더라도 필요한 증명은 네 개뿐이다.



참: SNARK 작업은 일반적으로 두 개의 *workId*를 포함하는 작업 패키지로 묶인다. 단, 트리의 최종 루트 증명은 예외다. 트랜잭션당 필요한 SNARK 작업은 두 개의 증명으로 계산되어, 포함된 트랜잭션 수와 구매해야 할 SNARK 작업 수의 균형이 유지된다.

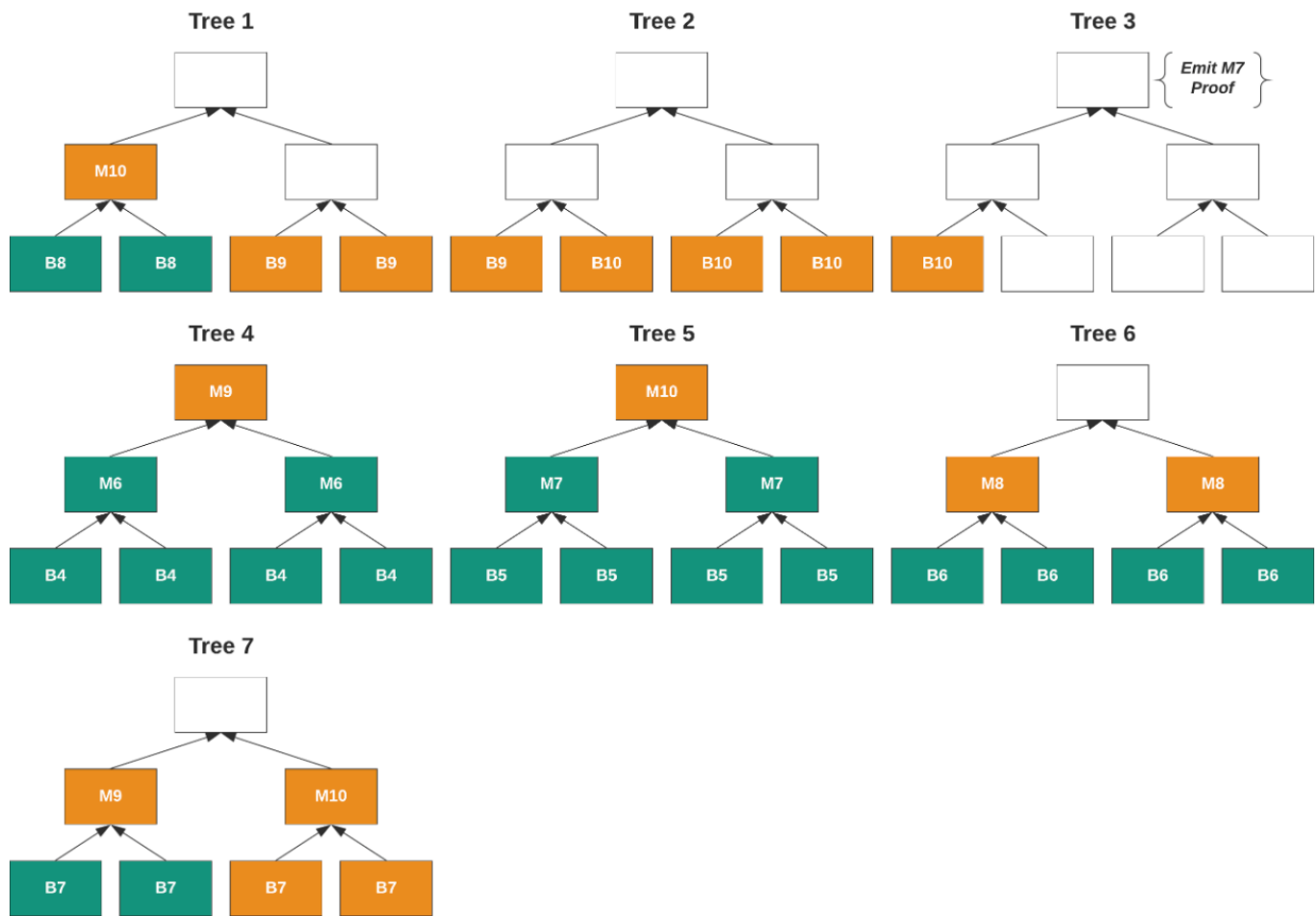
블록 9:

아홉 번째 블록에서 블록 생성자는 세 개의 트랜잭션(B9)을 추가한다. 현재 비어 있는 트리의 슬롯을 채우기 위해 세 개의 증명(M6)이 필요하다. 이전 블록에서 네 개의 증명이 추가되었으므로, 총 7개의 작업 중 세 개만 더 수행하면 된다. 이때 두 번째 트리의 M6 증명이 원장 증명으로 반환된다. 세 번째 B9 트랜잭션은 이제 비워진 트리에 들어가고, 두 개의 B7 증명이 추가된다.



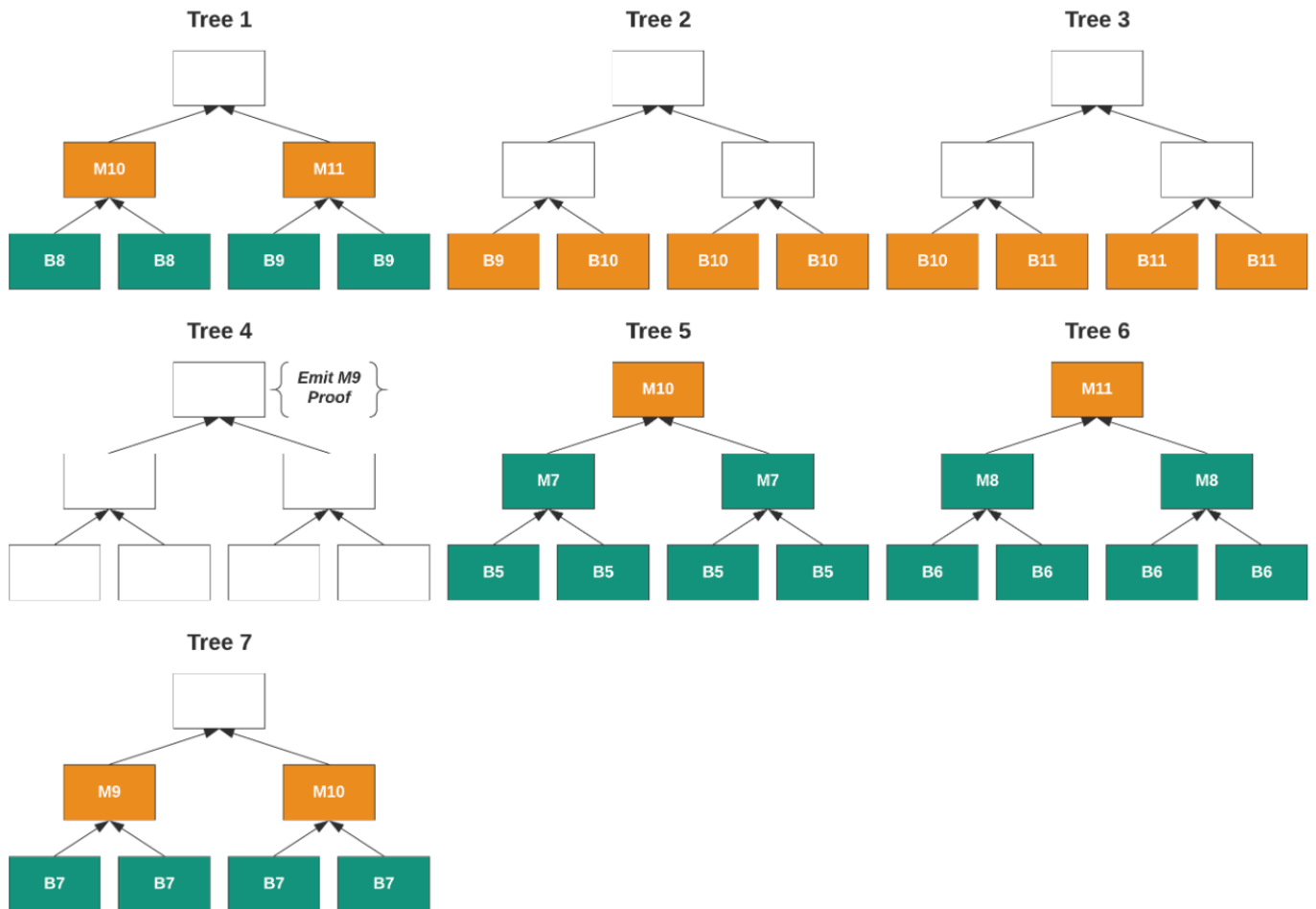
블록 10:

열 번째 블록에서 블록 생성자는 네 개의 트랜잭션을 추가하며, 그 결과 일곱 개의 증명(B7, M7, 두 개의 B8)을 포함한다.



블록 11:

열한 번째 블록에서 블록 생성자는 세 개의 트랜잭션(B11)을 추가하고, 다섯 개의 증명(B9, B9, M8, M8, M9)을 순서대로 완료한다. 또한 네 번째 트리에서 M9 원장 증명이 반환된다.



답: 스캔 상태의 내용을 확인하려면 `mina advanced snark-job-list` 명령어를 실행하라.

SNARK 풀과의 통합 (Integration with the SNARK Pool)

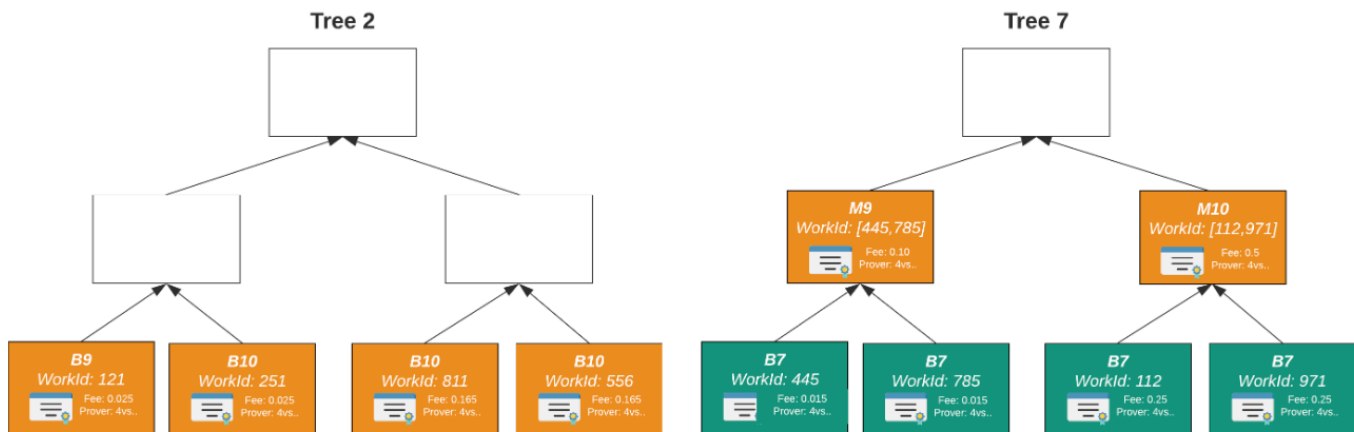
스캔 상태에 새로 추가된 작업은 SNARK 작업자가 수행해야 하는 보류 작업이다.

SNARK 작업자는 필요한 트랜잭션 SNARK를 생성하고, 완료된 작업에 대한 입찰(bid)을 제출한다. 노드가 해당 작업을 수신하고 유효성을 검증하면, SNARK 작업자는 유효하고 가장 낮은 수수료를 가진 작업을 로컬 SNARK 풀에 추가한다. 이 작업은 네트워크의 다른 피어들에게도 전파된다(gossip).

답: 여러 SNARK 작업자가 동일한 작업을 완료할 수 있지만, SNARK 풀에는 가장 낮은 수수료를 제시한 작업만 포함된다.

블록 생성자가 트랜잭션을 추가할 때 그에 상응하는 완료된 증명을 블록에 포함시키려면, SNARK 풀에서 해당 작업을 구매해야 한다. 이전 예시를 이어가면, 다음 블록(12)에서 블록 생성자가 세 개의 트랜잭션(코인베이스, 사용자 결제, SNARK 작업자에게 지급할 수수료 전송)을 추가하려면 세 개의 완료된 SNARK 작업을 구매해야 한다. 이는 일곱 번째 트리의 여섯 개 증명(B9, B10, M9, M10)에 해당하며, 각 SNARK 작업은 두 개의 `workId`를 포함한다.

블록이 생성되는 동안, SNARK 풀에는 완료된 작업과 필요한 작업에 대한 최적의 입찰가(0.025, 0.165, 0.1, 0.5)가 포함될 수 있다.



블록 생성자는 트랜잭션을 선택하기 전에 사용 가능한 SNARK 작업의 가격을 고려한다.

- 블록 생성자가 추가하는 첫 번째 트랜잭션은 코인베이스 트랜잭션으로, 코인베이스 보상을 받는다.
- 트랜잭션 수수료가 해당 트랜잭션을 포함하는 데 필요한 SNARK 작업 수수료를 충당하지 못하면, 해당 트랜잭션은 블록에 포함되지 않는다.

블록 생성자는 경제적으로 이득이 없는 SNARK 작업을 절대 구매하지 않는다.

필요한 순서의 완료된 SNARK 작업이 구매 가능한 상태가 아니라면, 해당 트랜잭션들은 블록에 포함되지 않는다. 이 경우 빈 블록이 생성될 수 있으며, 코인베이스 트랜잭션조차 추가되지 못할 경우 블록 생성자는 보상을 받지 못한다.

현재 SNARK 풀을 확인하려면 다음을 사용할 수 있다:

- [GraphQL API](#)
- Mina CLI: `mina advanced snark-pool` 명령어

타임락 계정 (Time-Locked Accounts)

타임락 계정은 블록 높이에 따라 결정되는 최소 잔액 이하로 잔액이 줄어드는 결제를 허용하지 않는다.

타임락을 생성하려면 새 계정을 만들 때 해당 설정을 지정해야 한다. 이 작업은 네트워크 시작 시점의 제네시스 원장(genesis ledger)에서만 가능하다. 이 섹션에서는 타임락의 메커니즘을 살펴보고, 타임락 계정과 상호작용하는 방법을 알아본다.

팁: 현재 릴리스에서는 타임락 계정의 값이 가입 순서에 따라 할당되었다.

타임락 이해하기 (Understanding Time-Locks)

타임락은 다음 필드들로 구성된다: `initial_minimum_balance`, `cliff` 시간, `vesting_period` 시간, 그리고 `vesting_increment`.

계정에 타임락이 있더라도, 계정에 충분한 자금이 있다면 사용할 수 있다. 타임락된 자금의 양은 네트워크 시작 시점에서 `initial_minimum_balance`로 설정된다. 네트워크가 `cliff`와 동일한 블록 높이에 도달하면, 타임락된 금액은 이후 매 `vesting_period`마다 `vesting_increment` 만큼 감소하기 시작한다.

이 프로세스에 대한 더 기술적인 설명은 [RFC-0025](#)를 참고하라. 해당 문서에는 이 메커니즘에 대한 심층적인 개요가 포함되어 있다.

유동 잔액 세부사항 (Liquid Balance Details)

특정 시점에서 베스팅 계정(vesting account)의 유동 잔액을 계산하고자 한다면, 이는 다음 함수에 의해 결정된다. (이 함수는 계정의 잠금된 부분을 계산한다.)

```
(*
 * uint32 global_slot -- the "clock" it starts at 0 at the genesis
block and ticks up every 3minutes.
 * uint32 cliff_time -- the slot where the cliff is (similar to startup
equity vesting)
 * uint32 cliff_amount -- the amount that unlocks at the cliff
 * amount vesting_increment -- unlock this amount every "period"
 * uint32 vesting_period -- the period that we increment the unlocked
amount
 * balance initial_minimum_balance -- the total locked amount until the
cliff
*)
let min_balance_at_slot ~global_slot ~cliff_time ~cliff_amount
~vesting_period
  ~vesting_increment ~initial_minimum_balance =
  let open Unsigned in
  if Global_slot.(global_slot < cliff_time) then initial_minimum_balance
  else
    match Balance.(initial_minimum_balance - cliff_amount) with
    | None ->
      Balance.zero
    | Some min_balance_past_cliff -> (
      (* take advantage of fact that global slots are uint32's *)
      let num_periods =
        UInt32.(
          Infix.((global_slot - cliff_time) / vesting_period)
          |> to_int64 |> UInt64.of_int64)
      in
      let vesting_decrement =
        UInt64.Infix.(num_periods * Amount.to_uint64
vesting_increment)
      |> Amount.of_uint64
```

```
in
match Balance.(min_balance_past_cliff - vesting_decrement) with
| None ->
    Balance.zero
| Some amt ->
    amt )
```

타임락 계정 생성하기 (Creating a Time-Locked Account)

현재 릴리스 기준으로, 타임락 계정을 생성할 수 있는 유일한 방법은 제네시스 원장(genesis ledger)을 통해서다. 향후 릴리스에서는 `mina client` 명령어나 GraphQL API를 통해 새로운 타임락 계정을 생성할 수 있는 기능이 추가될 예정이다.

결제의 생애 주기 (Lifecycle of a Payment)

미나에서는 결제가 검증되어 완료로 간주되기까지 여러 단계를 거친다.

이 문서는 단일 결제가 처리되는 과정을 단순화하여 개요 형태로 설명하며, 미나의 결제 메커니즘을 이해하는 데 도움을 주기 위한 것이다. 이는 기술적 세부사항을 모두 다루는 문서가 아니라, 사용자를 위한 간단한 절차 설명이다.

가십 프로토콜 (Gossip Protocol)

미나는 가십 프로토콜(gossip protocol)을 사용하여, 메시지가 네트워크의 모든 노드에 신뢰성 있게 그리고 신속히 전달되도록 보장한다.

결제 (Payments)

결제(payment)는 한 계정에서 다른 계정으로 가치를 전송하는 트랜잭션의 한 유형이며, 송신자가 결제 처리를 위해 지불할 의사가 있는 수수료를 포함한다.

이 시나리오에서는 송신자 Bob이 수신자 Alice에게 일정량의 MINA를 송금하는 과정을 다룬다.

단계 1: 결제 생성 – Bob이 송금 클릭 (Step 1: To Create a Payment, Bob Clicks Send)

네트워크의 누구나 결제를 생성하여 미나 네트워크에 전파할 수 있다. 결제는 개인 키로 암호 서명되어 송신자의 계정이 검증될 수 있다. 결제는 네트워크의 피어에게 전송되어 처리되며, 피어가 이를 수신하면 해당 결제는 피어의 로컬 메모리 내 트랜잭션 풀(transaction_pool)에 저장된다. 이 풀에는 피어가 네트워크에서 수신한 모든 트랜잭션이 임시로 저장된다.

단계 2: 블록 생성 – Bob의 결제가 처리 대기열에 등록됨 (Step 2: To Produce a Block, Bob's Payment Gets Put in a Todo List)

네트워크의 특정 슬롯에 대해 블록 생성자 노드가 선택된다. 활성 블록 생성자는 사용 가능한 SNARK 증명을 기반으로 처리 대기 중인 결제들을 선택하고, 이를 전이 프런티어(transition frontier)라 불리는 처리 목록에 유지한다. 블록 생성자는 이러한 블록을 구축함으로써 MINA 보상을 받는다. 블록 생성자는 이전 블록과 비교해 전이 블록의 구조를 정의하는 SNARK를 생성하지만, 이 단계에서는 새로운 결제를 아직 검증하지 않는다. 생성자는 이 정보를 SNARK 작업자에게 전송하여 SNARK 처리를 수행하도록 한다.

단계 3: SNARK 트랜잭션 검증 – Bob의 결제가 SNARK로 서명됨 (Step 3: To Prove a SNARK Transaction, Bob’s Payment Gets SNARK-Signed)

네트워크의 SNARK 작업자 노드들은 새로운 전이 블록의 각 단계에 대해 SNARK 계산을 수행한다. 각 결제에 대한 개별 증명과 인접한 결제 간의 병합 증명이 생성된다. 결국 모든 결제가 검증된다. SNARK 작업자는 이러한 증명을 생성함으로써 블록 생성자로부터 보상을 받을 수 있다. 이 증명들은 네트워크를 통해 전파된다.

단계 4: 결제 검증 – Alice와 Bob의 계정에 잔액 반영 (Step 4: To Verify a Payment, Alice and Bob’s Accounts Show the Result of the Transfer)

전체 블록이 증명되면, 블록 생성자는 전이 블록의 확인 메시지를 네트워크에 전파한다. 이후 네트워크의 각 노드는 해당 변경 사항을 로컬 계정 잔액에 반영하여 업데이트한다.

단계 5: 결제 확신도 확보 – Alice가 결제 완료를 신뢰함 (Step 5: To Achieve a Payment Confidence Level, Alice Is Confident the Transfer Is Complete)

후속 블록이 추가될 때마다 수신자는 결제가 실제로 완료되었고, 네트워크가 해당 블록에 대해 합의(consensus)에 도달했음을 더 높은 신뢰도로 확신할 수 있다. 대부분의 블록체인과 마찬가지로, 결제는 일정 수의 블록이 추가된 후 *트랜잭션 최종성(transaction finality)* 이 확보되었다고 간주된다.

예를 들어, 비트코인 네트워크에서는 공격자가 전체 해시레이트의 10% 이상을 확보하지 못한다고 가정할 때, **6개의 블록(약 60분)** 이후에 트랜잭션이 확정된다.

미나의 경우, 슬롯 지속 시간이 3분이며 90%의 정직한 스테이크를 가정할 때, 다음 표는 블록 단위의 예상 최종성, 해당 블록 수를 생성하는 평균 시간, 그리고 결제가 확정될 확률(신뢰도)을 보여준다.

최종성 (블록 단위)	평균 최종성 시간	결제 확정 신뢰도 (%)
8	33분	98.6709
15	60분	99.6231
23	1시간 32분	99.9695
30	2시간	99.9968
38	2시간 32분	100

평균 시간은 에폭당 슬롯 채움 비율(현재 75%)에 기반한 합의 상수(consensus constants)를 바탕으로 계산된다.

결제 확정을 위해 권장되는 대기 시간은 **15 블록**이며, 이는 **99.6%의 확률**로 결제가 되돌려지지 않음을 보장한다.

실패 시나리오 (Failure Scenarios)

결제가 실패할 수 있는 여러 이유가 존재한다.

트랜잭션이 네트워크에 의해 거부된 경우 (Transaction Is Not Accepted by the Network)

피어 노드에 전파된 트랜잭션이 수락되지 않는 이유는 다음과 같다:

- 트랜잭션 자체가 유효하지 않다. 예를 들어, 송신자 계정이 존재하지 않거나, 잔액이 부족하거나, 서명이 계정과 일치하지 않거나, 트랜잭션의 nonce가 올바르게 증가되지 않았을 수 있다.
- 네트워크 내의 악의적인 노드들이 특정 송신자에 대한 서비스 거부(DoS)를 시도할 수 있다. 그러나 이러한 행동은 강력히 비인센티브화되어 있으며, 단 하나의 정직한 노드만 있어도 이러한 공격을 방지할 수 있다.

트랜잭션이 블록에 포함되지 않은 경우 (Transaction Is Not Included in a Block)

트랜잭션이 유효하고 네트워크가 정직하다면, 해당 트랜잭션은 대부분의 경우 블록에 포함된다. 그러나 다음과 같은 경우 트랜잭션이 트랜잭션 풀에서 제거될 수 있다:

- 트랜잭션 풀이 용량(max_txpool_size)에 도달한 경우, 가장 낮은 수수료를 가진 트랜잭션이 제거되어 메모리에서 삭제된다. 이런 경우, 송신자는 당시 시장 상황에 맞춰 더 높은 수수료를 설정하여 트랜잭션을 다시 전송해야 한다.

미나 보안 (Mina Security)

미나 프로토콜은 ZK(영지식 증명) 기반으로 설계되어, 누구나 빠르게 네트워크를 동기화하고 검증할 수 있도록 한다. 이를 통해 참여도, 탈중앙화, 검열 저항성, 네트워크 보안이 기하급수적으로 향상된다.

그러나 미나의 보안은 여기서 그치지 않는다. 미나를 안전하게 유지하기 위한 다양한 보안 조치를 아래 자료에서 확인할 수 있다.

보안 감사 (Audits)

프로토콜 (Protocol)

- [2024년 8월 27일 o1js](#) — Veridise
- [2023년 12월 12일 Pickles](#) — Least Authority
- [2023년 8월 28일 트랜잭션 로직 및 트랜잭션 풀](#) — Least Authority
- [2022년 10월 16일 미나 코드베이스 및 생태계 프로젝트](#) — Mo Ashouri
- [2022년 2월 22일 미나 클라이언트 SDK, 서명 라이브러리 및 기본 구성요소](#) — NCC Group
- [2020년 12월 미나 프로토콜 및 스테이킹 경제 모델](#) — Gauntlet Network
- [2020년 5월 14일 코다 프로토콜](#) — NCC Group

도구 (Tools)

- [2022년 2월 4일 미나 Ledger 애플리케이션](#) — Least Authority
- [2021년 9월 28일 StakingPower Wallet](#) — Least Authority
- [2021년 8월 9일 Auro Wallet](#) — Least Authority
- [2021년 7월 16일 Clor.io Wallet](#) — Least Authority

감사 기관 (Auditors)

- [Veridise](#)
- [Least Authority](#)
- [NCC Group](#)
- [Gauntlet Network](#)

현재 커뮤니티 보안 프로그램 (Current Community Security Programs)

- [버그 바운티 프로그램](#)
- [Testworld Mission 2.0](#)

학습 자료 (Learn)

- [블록체인의 보안 결함 해결 — 접근 가능한 노트](#)
- [Ouroboros Samisika란 무엇인가?](#)
- [Ouroboros Samasika가 미나의 탈중앙화 목표를 어떻게 실현하는가](#)
- [미나 메인넷 런칭 — 인터넷 프라이버시와 데이터 보안의 새로운 시대](#)

