

# 스택스(Stacks)

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.

## 시작하기

license CC0-1.0

## 시작하기

### 스택스: 핵심 요약

스택스는 비트코인의 코어를 수정하지 않으면서 스마트 컨트랙트 기능을 비트코인에 제공하는 대표적인 비트코인 L2입니다.

스택스는 다음 세 가지 핵심 구성 요소를 통해 이를 실현하며, 나머지 문서에서 더 자세히 살펴봅니다:

### Proof of Transfer

Proof of Transfer(PoX)는 스택스 체인의 블록 생성 메커니즘입니다. 본질적으로 PoW의 블록 생성 패턴을 프로그래밍적으로 재현하려고 시도합니다. 스택스 채굴자는 새로운 스택스 블록을 채굴할 기회를 얻기 위해 BTC를 지출합니다. 내부적으로 이 블록 생성 메커니즘은 스택스 블록을 비트코인 블록에 앵커링하여, 스택스 블록을 뒤집는 것이 비트코인 블록을 뒤집는 것만큼 어렵게 만듭니다. 이는 큰 주장이고, 나카모토 블록 생성 섹션에서 더 자세히 풀이합니다.

### Clarity

Clarity는 스택스가 사용하는 스마트 컨트랙트 언어입니다. 개발자가 안전하고 보안성 높은 스마트 컨트랙트를 더 쉽게 작성할 수 있도록 바닥부터 설계되었습니다. 또한 스택스와 비트코인을 위해 목적에 맞게 만들어졌기 때문에 비트코인 상태를 읽는 내장 함수가 있으며, 이를 통해 Clarity에서 비트코인 상태를 사용해 동작을 수행할 수 있습니다. 예를 들어, 특정 비트코인 트랜잭션이 발생했는지 확인한 뒤에야 Clarity에서 민팅 함수를 실행하도록 체크를 설정할 수 있는데, 이는 세 번째 구성 요소인 sBTC에서 실제로 일어나는 일이기도 합니다.

## sBTC

sBTC는 스택스 레이어의 신뢰 최소화된 양방향 비트코인 페그입니다. sBTC는 스택스를 통해 비트코인을 프로그래머블하게 만들고 비트코인에 완전한 스마트 컨트랙트 기능을 제공하는 핵심입니다. sBTC는 연합체가 아니라, 가능한 한 낮은 상대방 위험으로 비트코인에 스마트 컨트랙트 기능을 도입하는 오픈 네트워크, 탈중앙 양방향 페그 솔루션으로 동작합니다. 이 문서에는 sBTC를 설명하는 전용 섹션이 있습니다.

## AI 기반 시맨틱 검색

특정 내용을 찾고 있나요? 본 문서는 AI 기반 시맨틱 검색과 통합되어 있으며, Cmd/Ctrl + K를 눌러 검색 창을 열고 원하는 것을 질문할 수 있습니다.

## 다음은?

### 스택스 알아보기

스택스가 정확히 어떻게 동작하는지 더 알아보고 싶다면 왼쪽 내비게이션의 "Concepts" 섹션으로 이동하세요. "What is Stacks" 페이지가 학습 여정의 최적 출발점입니다. 여기서 스택스가 어떻게 작동하는지, 다양한 구성 요소를 심층적으로 학습할 수 있습니다.

### 스택스 Dapp 빌드하기

바로 빌드를 시작하고 싶은 개발자인가요? 퀵스타트 튜토리얼이 최적의 출발점입니다. 단 30분 안에 스택스 위에서 빌드하기 위해 알아야 할 핵심을 소개합니다. 이후에는 나머지 가이드 & 튜토리얼에서 디파이 앱, 크라우드펀딩, 컬렉터블 등 다양한 사용 사례를 빌드하는 방법을 학습하세요.

### 스택스 노드 실행하기

스택스 노드를 실행하고 싶다면, 팔로워 노드와 마이너 노드 중 선택할 수 있습니다. 테스트넷과 메인넷 모두에서 두 유형을 실행하는 방법에 대한 가이드가 "Run a Node" 섹션에 준비되어 있습니다.

### 서이너(Signer) 실행하기

서이너는 스택스 생태계의 핵심 구성 요소로, 새로운 스택스 블록과 sBTC 트랜잭션을 검증하고 추가하는 역할을 맡습니다. 서이너 실행에 전용 섹션이 있습니다.

### STX 스테킹(Stacking)

스테킹은 스택스와 Proof of Transfer 합의 메커니즘의 핵심 구성 요소입니다. 단독 스테킹, 풀 스테킹, 서이너 운영 여부에 따라 다양한 방식이 있으며, 어떤 상황이든 과정을 안내하는 [스테킹 섹션](#)이 준비되어 있습니다.

## 더 깊이 참여하기

스택스 생태계에서 커리어를 성장시키고 싶나요? 자신의 프로젝트를 시작하고 [Code for STX](#) 프로그램에 제출해, 매달 프로젝트 작업만으로 STX를 받을 수 있습니다. 더 큰 도전을 원한다면, 검증된 헌신적인 스택스 빌더들의 엑셀러레이션 커뮤니티인 Clarity Collective에도 지원해 보세요.

다음으로, 스택스가 정확히 무엇이며 어떻게 동작하는지 살펴봅시다.

# 스택스 101

스택스는 블록체인 세계에서 매우 독특한 기술 모델을 갖고 있습니다. 이 섹션은 스택스가 어떻게 작동하는지 이해하는 데 필수적인 구성 요소를 높은 수준에서 개관하는 데 도움이 됩니다.

스택스가 무엇이며 철학적·기술적 관점에서 어떻게 동작하는지 기본을 다루고, 이어지는 섹션에서 더 상세히 파고들 수 있습니다.

먼저 스택스가 정확히 무엇인지 개요를 살펴봅시다.

## 스택스란 무엇인가요?

### 스택스란 무엇인가요?

2010년에 [사토시가 비트코인의 일반화를 어떻게 구상했는지](#)를 보면 스택스의 목표와 정신을 엿볼 수 있습니다:

"...비트코인과 CPU 파워를 공유하되 완전히 별도의 네트워크와 별도의 블록체인이 되는 것... 전 세계 모든 네트워크가 결합된 CPU 파워를 공유하여 전체 강도를 높이는 것."

이는 스택스 설계 결정의 주요 주제입니다. 비트코인 세계에서는 다소 모순적일 수 있지만, 스택스 네트워크는 비트코인 L2인 동시에 자체 토큰을 보유합니다.

이는 연합체에 의존하지 않고 탈중앙화를 유지하기 위한 의도적이고 핵심적인 설계 결정입니다.

이해가 어렵거나 회의적일 수 있지만, 문서를 진행하면서 이러한 아이디어를 더 깊이 탐구할 것입니다.

## 스택스와 블록체인의 목적

새로운 블록체인 기술을 평가할 때, 원래의 의도와 목적을 훼손하지 않는 것이 중요합니다. 비트코인으로 돌아가 보면, 본래 다음과 같이 설계되었습니다:

- 탈중앙화
- 불변성
- 보안성

아마도 블록체인 트릴레마—탈중앙화, 확장성, 보안성의 균형 문제—에 대해 들어봤을 것입니다.

스택스는 체인을 레이어로 분리하는 접근으로 이 트릴레마를 해결합니다.

가장 아래에는 기초 레이어인 비트코인이 있습니다.

비트코인은 가장 탈중앙화되고, 가장 안전하며, 가장 불변적인 블록체인 네트워크입니다. 하지만 몇 가지 트레이드오프가 따릅니다.

비트코인은 다른 네트워크에 비해 매우 느립니다. 약 10분마다 새 블록이 생성되므로, Solana처럼 속도를 위해 설계된 네트워크와 비교하면 처리량이 미미합니다.

비트코인은 또한 "지루"합니다. 이더리움은 비트코인 이후 등장해, 비트코인이 돈에 대해 했던 것을 소프트웨어에 적용하려 했습니다. 이더리움의 목표는 일종의 탈중앙 슈퍼컴퓨터로서, 스마트 컨트랙트(블록체인에 기록되는 코드)를 위한 글로벌 컴퓨팅 환경이 되는 것입니다.

비트코인은 또한 확장성이 좋지 않습니다. 모든 새 블록이 네트워크의 모든 노드에 전파되어야 하므로, 비트코인은 네트워크에서 가장 느린 노드만큼만 빨리 동작할 수 있습니다. 이제는 Cosmos와 같은 모듈러 블록체인 네트워크가 등장하여, 누구나 자신의 블록체인 네트워크를 쉽게 띄울 수 있도록 설계되고 있습니다.

많은 신규 프로토콜은 이러한 특성을 부정적으로 보고 제거하려 하지만, 스택스 커뮤니티는 다르게 접근합니다.

## 스택스의 방식

스택스는 바닥에 기초 결제 레이어(비트코인)를 두고, 그 위에 레이어를 추가해 확장성과 기능을 제공하는 레이어드 접근을 취합니다.

L2에는 다양한 유형과 구성 방식이 있으며, 각기 서로 다른 트레이드오프를 갖고 확장성이나 기능성의 목표를 달성합니다.

이 레이어드 접근을 통해, 우리는 이더리움 같은 체인과 동일한 기능을 비트코인 위에 구축할 수 있습니다.

즉, 스택스는 독자 토큰 등 고유한 특성을 지닌 비트코인 L2로서, 모든 트랜잭션의 이력 원장을 유지하고 자체 보안 예산(비트코인의 보안 예산에 더해—이는 다음 섹션에서 자세히 다룸)으로 운영하기 위한 인센티브 메커니즘을 제공합니다.

이는 라이트닝과 같은 다른 비트코인 레이어와 스택스를 구분하는 요소 중 하나입니다.

라이트닝은 비트코인에 추가 기능을 더하지 않으며, 단지 기존 기능의 확장과 가속을 돕습니다. 라이트닝은 영속적 상태를 유지하지 않는(ephemeral) 특성 때문에 데이터를 추적하고 상태를 유지해야 하는 스마트 컨트랙트 등에는 적합하지 않습니다.

이에 반해 스택스는 비트코인에 추가 기능을 더하면서도, 궁극적으로 비트코인으로 결제(settle)됩니다(다음 섹션에서 다룰 내용).

이 접근의 장점은 관심사의 분리를 유지해 비트코인을 단순하고 견고하게 유지하면서, 기능성과 속도를 위한 레이어를 추가할 수 있다는 점입니다. 설령 상위 레이어가 침해되더라도 기초 레이어에는 영향이 없습니다.

이는 전 세계적 탈중앙화된 화폐(비트코인)와 그 위에 구축되는 탈중앙 경제(스택스)를 논할 때 중요한 속성입니다.

STX 토큰은 정직한 블록 생성을 인센티브하기 위한 별도의 토큰입니다. 이는 페깅된 비트코인을 나타내지 않으며(그 목적을 위한 별도의 비트코인 페그는 [sBTC](#)입니다), 비트코인 커뮤니티 일부의 반감을 살 수 있지만 여러 장점이 있습니다.

스택스 체인에 토큰을 도입함으로써, 채굴자가 정직하게 스택스 블록을 생성하도록 추가 경제적 인센티브를 제공합니다.

이 토큰은 체인 성장을 위한 수단으로서 추가 인센티브를 제공해, 블록 생성을 순수한 이타심에만 의존하지 않고, 빌더·토큰 보유자·투자자 모두에게 동시에 인센티브를 제공할 수 있게 합니다.

2017년의 ICO 사기는 많은 사람들에게 불신을 남겨, 새로운 토큰을 가진 블록체인 프로젝트에 대한 회의론을 키웠습니다.

하지만 문제는 이들 프로젝트 대부분이 실제 가치가 없었고, 어떤 가치에도 앵커링되지 않았으며 실질적 유틸리티를 제공하지 못했다는 점입니다.

스택스와 같은 프로젝트는 **탈중앙화 방식**으로 비트코인을 활용하고 생산적인 자산으로 만드는 수단이라는 실질적 유틸리티를 제공합니다. 현재 비트코인을 생산적으로 만드는 유일한 방법은 커스터디 서비스를 이용하거나, 이더리움의 wBTC 같은 방식으로 비트코인 체인 밖으로 옮기는 것입니다.

스택스는 결국 비트코인 체인으로 결제되면서도 이를 가능케 합니다.

또한 스택스는 비트코인을 기초 결제 레이어로 활용하는 탈중앙·검열저항 소프트웨어를 구축할 수 있게 합니다. 궁극적으로는 돈으로서의 비트코인을 활용하는 금융 시스템과 탈중앙 소프트웨어 제품의 네트워크를 구축하는 것이 목표입니다.

이제 스택스가 비트코인과 정확히 어떻게 연결되는지 살펴봅시다.

# 비트코인 연결

앞 섹션에서 스택스가 비트코인의 코어를 수정하지 않으면서 스마트 컨트랙트 기능을 비트코인에 제공한다고 설명하고, 체인의 동작을 간략히 소개했습니다.

이는 큰 약속인데, 스택스는 이를 실제로 어떻게 달성할까요? 그리고 스택스를 다른 비트코인 레이어나 이더리움 같은 다른 블록체인과 구분 짓는 요소는 무엇일까요?

스택스가 해결하려는 문제와 그 해결 방식을 기술적 상세에 들어가기 전에 높은 수준에서 개관하는 것이 중요합니다. 문서를 진행하며 일부 주제는 더 깊게 파고들겠지만, 전체 그림을 이해하는 데 도움이 됩니다.

이 주제는 다소 토끼굴처럼 깊고 이 섹션도 꽤 길지만, 스택스가 해결하려는 문제와 그 해결 방식을 심층적으로 이해할 수 있을 것입니다.

그럼 시작해 봅시다.

## 스택스는 비트코인 L2인가요?

스택스는 스마트 컨트랙트를 위한 비트코인 레이어입니다. 레이어-1(L1), 레이어-2(L2), 혹은 사이드체인으로의 분류는 사용되는 정의에 따라 달라집니다. 그럼에도 일반적으로 L1 체인은 (a) 자체 보안 예산을 보유하고, (b) 다른 어떤 L1 체인의 도움 없이도 생존할 수 있다는 점에서 주권적입니다. L2 체인은 일반적으로 자체 보안 예산을 가지지 않고 기반 L1 체인의 보안을 공유하며, 기반 L1 체인 없이는 존재할 수 없습니다. L2가 사용할 수 있는 설계 메커니즘은 매우 다양하며, [다른 비트코인 레이어들 속의 스택스](#) 섹션에서 그 여러가지와 스택스의 비교를 다룹니다.

2021년 초의 스택스 초기 릴리스는 비트코인 L1과 분리된 보안 예산을 가지고 있었습니다. 스택스 레이어가 비트코인 L1 없이 동작할 수는 없었지만, 프로젝트에 참여한 개발자들은 이를 기존 분류에 깔끔하게 들어맞지 않는 별개의 시스템으로 설명했고, 때때로 레이어 1.5라는 용어를 사용하기도 했습니다(예: [이 Decrypt 기사](#)).

다가오는 계획된 릴리스인 Nakamoto 릴리스에서는 스택스가 더 이상 비트코인과 분리된 보안 예산을 갖지 않게 됩니다. 대신 **비트코인 해시파워의 100%**가 스택스 레이어의 파이널리티를 결정합니다. 다음 업그레이드 이후 스택스 블록/트랜잭션을 리오그(reorg)하려면 공격자는 비트코인 L1 자체를 리오그해야 하며 (이는 수행하기 매우 어려워 비트코인 레이어에 훌륭한 보안 속성을 제공합니다), 자세한 내용은 [스택스 백서](#)를 참고하세요.

[Ethereum에서 사용하는 L2의 정의](#)와 다른 최신 생태계의 정의는 다르며, 오직 L1의 보안과 L1 채굴자만으로 자산을 인출할 수 있는 능력에 초점을 둡니다. 그 정의에 따르면, sBTC 인출 여부를 페그-아웃 서이너 집합이 결정한다는 점에서 스택스 레이어는 명확한 L2가 아닙니다. 비트코인 L1의 변경 없이는 비트코인이 그러한 검증을 지원할 수 없습니다(미래에 가능해질 수도 있습니다). 또한 Ethereum의 L2 정의는 비트코인 L2에 깔끔하게 적용되지 않는데, 비트코인의 경우 L2에서는 새로운 자산이 L2에서 발행되고 L1에서는

발행되지 않기 때문입니다(오직 BTC만이 L1 자산). 따라서 자산 인출의 보안을 기준으로 한 정의는 직접 적용되기 어렵고, 어차피 자산이 L2에서 정의·사용되며 비트코인 L1로 “인출”하지 않는 경우가 많습니다(BTC 자체는 예외). 오히려 더 중요한 것은 “비트코인에 대한 결제(settlement)” — 즉, 컨트랙트 데이터와 상태가 비트코인 해시파워 100%에 의해 보안되는가 — 입니다.

또한 비트코인 위의 L2는 기능성과 확장성 모두를 확장하는 추가 목적을 수행해야 하므로, L1의 기능에 따라 L2가 달성하는 목표는 근본적으로 다를 수 있다는 점을 기억하세요.

사용자와 개발자들은 이해의 단순성 때문에 자연스럽게 스택스를 비트코인 L2라고 부릅니다. 스택스 레이어에는 다음과 같은 특성들이 있어 비트코인 L2라는 개념을 뒷받침합니다:

1. 위에서 논의한 비트코인 파이널리티 — 비트코인 해시파워 100%가 블록 순서와 트랜잭션 파이널리티를 결정합니다.
2. 스택스 합의는 비트코인 L1에서 실행되며, 스택스 L2는 비트코인 L1 없이는 동작하거나 생존할 수 없습니다.
3. 다가오는 탈중앙 비트코인 페그인 sBTC가 도입되면, 스택스 레이어의 경제 대부분은 경제 단위로 BTC를 사용할 가능성이 큼니다. 대부분의 사용자는 지갑과 앱에서 단순히 비트코인을 사용하고, 이후 BTC를 비트코인 L1로 페그-아웃할 것으로 예상됩니다.
4. 스택스의 모든 데이터와 트랜잭션은 모든 비트코인 블록마다 자동으로 해시되어 비트코인 L1에 영구 저장됩니다. 누구나 비트코인 L1의 해당 해시를 확인하여 스택스의 어떤 데이터가 유효한지 검증할 수 있습니다. 이러한 L1의 해시 압축 저장은 롤업과 어느 정도 유사합니다(다른 차이점도 존재). 자세한 과정은 [블록 프로덕션](#) 섹션을 참고하세요.
5. 스택스 레이어의 컨트랙트는 비트코인 L1 트랜잭션을 읽고 이에 반응할 수 있습니다. 스택스 레이어의 자산은 단순히 비트코인 L1 트랜잭션을 통해 이동될 수 있습니다.

위의 모든 상세에도 불구하고, 왜 일부 사람들은 스택스가 비트코인 L2가 아니라고 생각할까요? 이 질문이 자주 제기되는 이유는 몇 가지가 있습니다:

1. (2021년 초 릴리스된) 스택스의 초기 버전은 분리된 보안 예산을 가졌으며, Nakamoto 릴리스에서 비트코인 해시파워 100%를 따르는 방식으로 변경되었습니다. 여전히 초기 스택스 버전에 대해 이야기하는 오래된 자료와 블로그 글들이 떠돌고 있으며, 시간이 지나면서 업데이트될 가능성이 큼니다.
2. Ethereum의 L2 정의에 따르면, 사용자는 L1 트랜잭션만으로 그리고 오직 L1 보안에 의존하여 기초 레이어 자산을 인출할 수 있어야 합니다(예: 라이트닝에는 해당). 이 정의는 비트코인 L2에 깔끔히 적용되지 않는데, 자산이 비트코인 L1이 아니라 L2에서 정의되기 때문입니다. 이 문제가 중요한 유일한 자산은 비트코인 L1에서 오는 페그드 BTC이며, 다른 모든 자산은 어차피 L2 네이티브입니다. 다가오는 스택스 릴리스에서는 사용자가 비트코인 L1 트랜잭션만 전송해 BTC를 인출할 수 있지만, 비트코인 L1은 그러한 복잡한 트랜잭션을 검증할 수 없고 페그-아웃 서이너 다수가 페그-아웃 요청에 서명해야 합니다. 이상적인 세계에서는 비트코인 채굴자가 그러한 트랜잭션을 검증할 수 있겠지만, 이는 비트코인 L1의 변경을 필요로 합니다. 따라서 스택스의 설계는 비트코인 L1의 변경 없이 배포할 수 있고 탈중앙화된 방법을 최적화합니다. 미래에 비트코인 L1 변경이 가능해진다면 스택스 레이어 보안도 그로부터 이익을 얻을 수 있습니다.

3. 비트코인 커뮤니티 구성원들은 일반적으로 주장에 대해 회의적이며, 허위 마케팅 주장에 경계심을 갖고 있습니다. 이는 비트코인 생태계에 전반적으로 건강하며 면역 체계를 강화합니다. 일부 구성원은 기술적 세부와 근거를 충분히 읽기 전까지 스택스를 비트코인 레이어 또는 L2로 보는 것에 회의적일 수 있습니다. 이 주제에 관한 좋은 [트위터 스레드](#)도 있습니다.

그렇다면 왜 스택스에 ‘사이드체인’이라는 용어를 사용하지 않을까요? 비트코인에서 사이드체인은 일반적으로 비트코인 L1과 다른 보안 예산(비트코인 채굴자 일부가 사이드체인에 참여 — 비트코인 파이널리티 100%를 따르지 않음), 사이드체인에서 자체 합의(비트코인 L1에서 실행되는 것과 대조), 그리고 데이터/해시를 비트코인 L1에 게시하지 않는 특성을 가집니다. 스택스 레이어는 합의가 비트코인 L1에서 실행되고, 비트코인 파이널리티를 따르며, 데이터/해시를 L1에 게시하기 때문에 이 정의에 깔끔히 들어맞지 않습니다.

### 스택스 레이어는 롤업과 함께 동작할 수 있나요?

예! 스택스 레이어에 롤업을 통합하기 위한 활발한 연구개발이 이미 진행 중입니다. 스택스 레이어와 소버린 롤업 모두에서 기술적으로 까다로운 부분은 BTC를 스택스 레이어나 소버린 롤업으로 들고 빼는 방법입니다. 탈중앙화된 BTC 페그인 [sBTC](#)는 스택스 레이어와 소버린 롤업 모두에 적용됩니다. 비트코인 L1을 변경하지 않고서는, 서이더의 탈중앙 개방형 멤버십 그룹을 사용하는 sBTC와 유사한 설계가 비트코인 레이어 간 BTC를 들고 빼는 가장 신뢰 최소화된 방법입니다. 유효성 롤업을 가능케 하는 필요한 비트코인 L1 업그레이드(즉, 비트코인 L1이 레이어로부터의 BTC 인출을 강제할 수 있음)가 이뤄지면, 스택스 레이어도 그 이점을 얻도록 업그레이드할 수 있습니다.

소버린 롤업에는 sBTC 같은 신뢰 최소화 자산이 필요하기 때문에, sBTC가 출시되면 이러한 소버린 롤업 배포는 더욱 흥미로워집니다. 스택스 레이어는 소버린 롤업에서 사용할 신뢰 최소화 BTC 자산을 위한 탈중앙 서이더 그룹을 잠재적으로 제공할 수 있으며, DA는 예컨대 오디널즈(Ordinals)처럼 비트코인 L1에서 직접 제공될 수 있습니다.

### 왜 스택스에는 토큰이 필요한가요?

이는 “블록체인에 토큰이 필요한가”라는 크립토·비트코인 세계의 중심 철학적 논의로 이어집니다.

우선 토큰이 존재하는 근본 이유를 살펴봅시다: 블록체인의 유지보수와 발전을 위한 자금 조달입니다.

비트코인은 토큰입니다. 이는 채굴자가 체인에 새 블록을 추가하도록 인센티브를 제공하는 암호화폐입니다. 비트코인의 경우 채굴 보상은 사전에 정의된 일정에 따라 주어지며, 채굴 보상이 끝나면 체인은 트랜잭션 수수료만으로 생존해야 합니다.

블록체인의 목적은 체인에서 발생한 모든 트랜잭션의 영구한 역사 기록을 유지하는 것입니다. 블록체인은 기본적으로 원장입니다. 토큰은 체인을 보안·유지하기 위한 인센티브 메커니즘으로 사용됩니다.

이 때문에 라이트닝 같은 네트워크와 다른 P2P 네트워크는 토큰이 필요 없습니다. 이들은 역사 기록을 유지할 필요가 없습니다. 라이트닝 같은 채널 기반 솔루션은 사용자들이 서로 2-of-2 멀티시그를 열도록 의존합니다. 채널이 닫히면 상태는 사라집니다. 글로벌 금융 시스템을 유지하려는 시스템에서는 그 유지보수가 올바르게 인센티브화되는 것이 중요합니다.

이 개념을 스택스와 그 목표의 맥락에서 살펴봅시다. 스택스는 비트코인에 스마트 컨트랙트 기능을 제공하여, 비트코인 위에 탈중앙 경제를 구축하기 위한 프로그래밍 레일을 제공하고자 합니다.

많은 비트코인 커뮤니티 구성원들은 새로운 토큰에 회의적이며, 그럴 만한 이유가 있습니다. 수많은 프로젝트가 토큰을 억지로 도입하지만, 실제로 토큰이 필요 없는 경우가 많습니다. 스택스 프로젝트는 토큰 없이도 비트코인 L1에서 앱과 프로토콜을 오랫동안 구축해 온 비트코인 빌더들이 시작했습니다(예: 2015년 비트코인 L1에서 출시된 BNS는 비트코인 L1의 OP\_RETURN을 사용하는 최대 규모 프로토콜 중 하나였습니다). 그렇다면 왜 많은 비트코인 빌더들이 스택스 L2에 별도 토큰을 도입하기로 했을까요? 아주 좋은 질문입니다! 자세히 살펴보죠.

스택스 토큰(STX)은 주로 두 가지 목적에 사용됩니다(자세한 내용은 [스택스 백서](#) 참조):

1. 스택스 L2 채굴자에 대한 인센티브
2. 페그-아웃 서이너에 대한 인센티브

토큰을 제거하는 유일한 방법은 스택스를 Liquid처럼 연합형 네트워크로 구축하는 것입니다. 연합체에서는 사전에 선택된 기업 집단이 채굴과 블록 생성을 통제하며, 페그-아웃 트랜잭션에도 사전에 선택된 기업 집단을 신뢰해야 합니다.

스택스 개발자들은 개방적이고 퍼미션리스한 시스템을 설계하고자 했습니다. 탈중앙 채굴 과정을 가능케 하는 유일한 방법은 인센티브입니다. 위에서 언급했듯이, 이는 비트코인이 동작하는 방식과 동일합니다. 새로 발행되는 BTC가 새 블록 채굴 인센티브로 사용되며, 전 세계 누구나 채굴자가 될 수 있습니다. BTC를 가진 누구든 스택스 L2 체인을 채굴할 수 있으며, 이는 개방적이고 퍼미션리스합니다.

마찬가지로 sBTC의 설계에서는 서이너 그룹이 연합과 달리 개방적이고 퍼미션리스합니다. 이 서이너들은 페그-아웃 요청에 대해 프로토콜을 올바르게 따르도록 경제적 인센티브를 가집니다. 연합체에서는 사용자가 연합 멤버들을 맹목적으로 신뢰해, 연합에서 자신의 BTC를 꺼내 비트코인 L1로 돌려받아야 합니다. 스택스 개발자들은 비트코인 L1에서 스택스 L2로, 그리고 다시 되돌리는 과정을 개방적·퍼미션리스·탈중앙 방식으로 구현하고자 했습니다. 이는 경제적 인센티브—즉, 토큰의 필요—를 통해 가능해집니다.

더 많은 비트코인 레이어가 등장함에 따라, 연합형 vs 개방형 네트워크 설계에는 미묘한 차이가 있습니다. Botanix의 Spiderchain 같은 일부 프로토콜은 개방형 네트워크를 제공하지만 인센티브 메커니즘은 다릅니다. 이러한 내용은 [다른 레이어들 속의 스택스](#) 섹션에서 자세히 다룹니다.

이 두 가지 이유 외에도, STX는 트랜잭션의 가스비 지불에도 사용됩니다. 그러나 다가오는 sBTC 페그가 라이브되면 스택스 L2의 경제 대부분은 비트코인 표준을 따르게 되어, 경제 단위로 BTC를 사용하게 될 것으로 예상됩니다. 사용자는 대체로 지갑과 앱에서 비트코인만 다루게 되며(가스비는 백그라운드에서 아토믹 스왑으로 BTC로 지불 가능), 이는 자연스러운 흐름입니다. 중요한 점은, 스택스 L2에서 채굴 인센티브로 BTC를 사용할 수 없다는 것입니다. 프로토콜에 의해 새로 발행되는 자산—즉, 토큰—을 통해서만 탈중앙 블록 생성을 인센티브화할 수 있기 때문입니다(비트코인 자체가 작동하는 방식과 유사).

## 스택스와 비트코인의 공생 관계

스택스와 비트코인은 상호 보완적입니다. 스택스는 비트코인의 극도의 탈중앙성, PoW 합의 메커니즘, 그리고 암호화폐로서의 가치를 활용합니다.

하지만 스택스는 추가적인 사용 사례를 열어 비트코인의 가치를 시간이 지남에 따라 높임으로써 비트코인을 보완하기도 합니다. 이는 또한 코인베이스 보상이 사라진 이후, 트랜잭션 수수료만으로 비트코인이 유지되어야 하는 미래의 지속 가능성 문제를 해결하는 데 도움을 줍니다.

비트코인을 오직 가치 저장 수단으로만 본다면, 각 트랜잭션의 경제적 밀도(즉, 교환되는 가치의 크기)는 최소화될 것입니다. 그러나 비트코인이 전체 탈중앙 경제의 기초가 된다면, 그러한 트랜잭션은 훨씬 더 가치 있게 되어 트랜잭션 수수료가 증가합니다. 이는 코인베이스 보상이 줄어드는 상황에서도 채굴자가 네트워크 보안을 계속 담당하도록 하는 핵심 인센티브입니다.

## 비트코인 상태 읽기

스택스 체인이 비트코인과 연결되어 “초능력”을 발휘하는 이유는 위에서 논한 프로토콜 수준의 연결뿐 아니라, 비트코인을 프로그래밍 수준에서 활용할 수 있기 때문이기도 합니다.

여기서 Clarity가 등장합니다. Clarity는 스택스의 스마트 컨트랙트 언어이며, 여기서 논하는 많은 기능을 실제로 구현하는 수단입니다.

Clarity의 자주 언급되는 특징 중 하나는 비트코인 체인의 상태에 접근할 수 있다는 점인데, 실제로는 어떻게 가능할까요? 스택스의 PoX 메커니즘 덕분에 모든 스택스 블록은 비트코인 블록에 연결되며, get-burn-block-info? 함수를 통해 비트코인 블록 헤더 해시를 조회할 수 있습니다.

이 함수는 비트코인 블록 높이를 입력받아 해당 헤더 해시를 반환합니다. 또한 burn-block-height Clarity 키워드는 현재 비트코인 체인의 블록 높이를 제공합니다.

다만 get-burn-block-info?는 해당 높이의 비트코인 블록이 이미 처리되었고 스택스 체인 출시 이후 생성된 경우에만 데이터를 반환합니다. 따라서 비트코인에서 어떤 일이 발생했는지 평가하려면 최소 한 블록 이후까지 기다려야 합니다.

이는 Clarity 컨트랙트가 비트코인의 프로그래밍 레이어 역할을 할 수 있게 되는 1단계입니다. BTC 트랜잭션이 시작되면, 가장 먼저 Clarity 컨트랙트가 그 사실을 인지해야 합니다. 이는 위에서 논의한 Clarity 함수와 BTC 라이브러리를 활용해 수동으로 수행할 수 있으며, 이는 Catamaran Swaps에서 하는 방식입니다.

이 과정은 2.1에서 추가된 get-burn-block-info? 같은 Clarity 함수들로 더 쉬워졌다는 점에 유의하세요.

혹은(우리 dapp에 약간의 중앙화 비용을 치르면서) Hiro의 chainhooks 같은 이벤트 기반 아키텍처를 사용해 자동화할 수도 있습니다. 이를 통해 특정 BTC 트랜잭션이 시작될 때 Clarity 컨트랙트 호출을 자동으로 트리거할 수 있습니다.

이것이 스택스를 사용해 비트코인 디앱을 구축하는 첫 번째 구성 요소, 즉 비트코인 체인에 대한 읽기 접근입니다.

다음으로, 스택스의 블록 생성 메커니즘인 Proof of Transfer를 살펴보며 스택스가 정확히 어떻게 “비트코인 위에 구축”되는지 좀 더 깊이 파고들겠습니다.

# Proof of Transfer

앞선 섹션에서는 스택스의 비전과 정신을 살펴보았고, 비트코인을 수정하지 않으면서 그 위에서 기능을 확장하는 방법에 대해 이야기했습니다.

이번 섹션에서는 이를 가능하게 하는 블록 생성 메커니즘, Proof of Transfer를 살펴보겠습니다.

여기서는 Proof of Transfer의 개념적 개요를 다룹니다. 블록 생성이 기술적으로 정확히 어떻게 일어나는지 자세한 내용은 [Block Production](#) 섹션을 확인하세요.

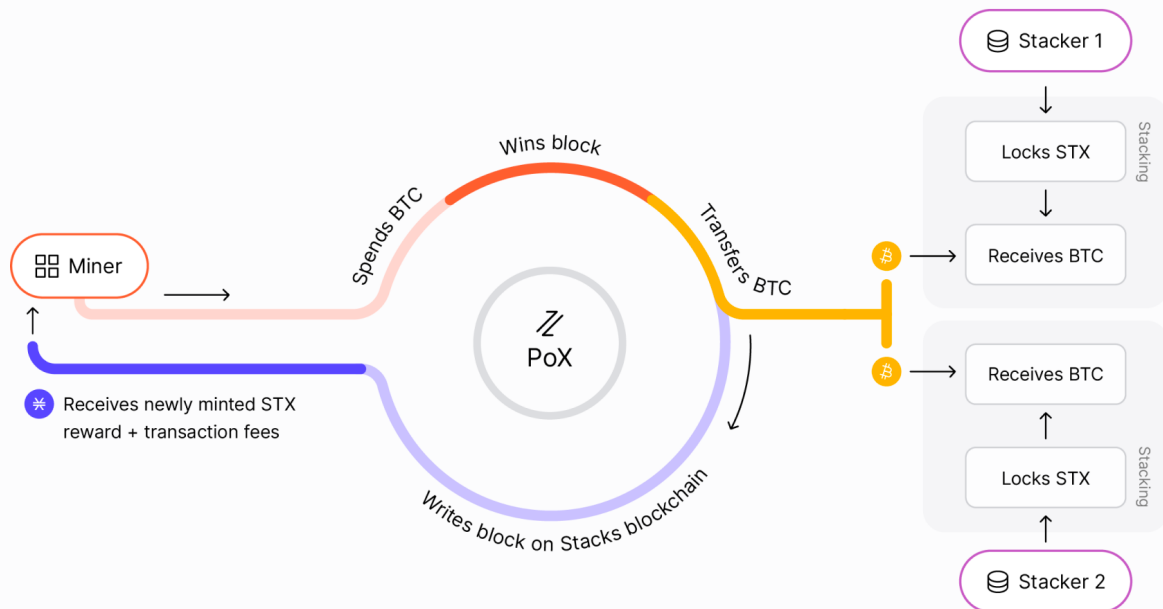
블록체인의 합의 알고리즘은 블록체인을 보호하기 위해 연산 자원 또는 재정적 자원을 요구합니다. 탈중앙 합의의 일반적 관행은 단일 악의적 행위자가 네트워크를 공격할 만큼 충분한 연산력 또는 지분을 확보하는 것을 사실상 불가능하게 만드는 것입니다.

현대 블록체인에서 널리 쓰이는 합의 메커니즘에는 연산 자원을 투입하는 작업증명(Proof of Work)과, 재정 자원을 투입하는 지분증명(Proof of Stake)이 있습니다.

Proof of Burn은 상대적으로 덜 사용되지만, 채굴자가 연산 자원의 대용으로 작업증명 암호화폐를 ‘소각’(파괴)하며 경쟁하는 또 다른 합의 메커니즘입니다.

Proof of Transfer(PoX)는 Proof of Burn 메커니즘의 확장입니다. PoX는 기존 블록체인의 작업증명 암호화폐(이 경우 비트코인)를 사용해 새로운 블록체인의 보안을 확보합니다. 다만 Proof of Burn과 달리 암호화폐를 소각하는 대신, 채굴자가 커밋한 암호화폐를 네트워크의 다른 참여자(이 경우 Stacker)에게 **이전**합니다.

## Mechanism



이 방식은 네트워크 참여자가 PoX 암호화폐 네트워크를 보호하는 동시에 기초 암호화폐(BTC)로 보상을 받을 수 있게 합니다. 따라서 PoX 블록체인은 자신이 선택한 PoW 체인에 앵커링됩니다. 스택스는 비트코인을 앵커 체인으로 사용합니다.



## Participants



### Miners



- Miners transfer BTC to stackers for the chance to win the block reward and transaction fees for a Stacks block.

| Miner 1        | Miner 2        | Miner 3        |
|----------------|----------------|----------------|
| 0.3 BTC<br>30% | 0.5 BTC<br>50% | 0.2 BTC<br>20% |

- The winning miner is elected by a weighted random function. Their chance of winning is proportional to the amount of BTC they transferred compared to the other miners.



- Once elected, the winning miner (block leader) earns the right to stream microblocks and commit a new block to the Stacks blockchain.



- The miner receives newly minted STX as block reward, plus transaction fees.



### Stackers



- Stackers lock their STX holdings for one or multiple cycles.



- Depending on the amount of STX locked, they can choose between stacking independently or collectively by pooling with other stackers.



- Stackers provide a BTC address to receive their rewards. The rewards are available as soon as they are transferred to the address, and they are proportional to the amount of STX stacked.



- When the amount of cycles committed ends, their STX holdings unlock.

## 왜 비트코인인가?

스택스가 합의를 뒷받침할 블록체인으로 비트코인을 선택한 이유는 여러 가지가 있습니다. 2009년에 출시된 가장 오래된 블록체인 프로토콜이며, 암호화폐 커뮤니티 밖에서도 자산으로 인정받았습니다. 지난 10년간 BTC는 모든 암호화폐 중 가장 높은 시가총액을 유지해 왔습니다.

비트코인은 단순성과 안정성을 지향하고 시간의 시험을 견뎌 왔습니다. 네트워크에 영향을 미치거나 공격하는 것은 잠재적 공격자에게 실행 불가능하거나 비현실적입니다. 대중의 관심을 사로잡은 몇 안 되는 암호화폐 중 하나이기도 합니다. 비트코인은 가정용어가 되었고, 정부·대기업·기성 금융기관에서 자산으로 인정받습니다. 마지막으로, 비트코인은 신뢰할 수 있는 가치 저장 수단으로 널리 간주되며, PoX 합의 메커니즘을 지원하는 광범위한 인프라를 제공합니다.

SIP-001은 스택스를 보호하기 위해 비트코인을 선택한 이유의 전체 목록을 제공합니다.

참고로, SIP는 Stacks Improvement Proposal의 약자로, 커뮤니티 구성원이 네트워크 변경에 합의하는 절차입니다. SIP를 자세히 읽어보는 것은 구현 수준에서 스택스에 익숙해지는 훌륭한 방법입니다. 모든 SIP는 문서의 SIPs 섹션에서 확인할 수 있습니다.

## 비트코인 자본 열기(Unlocking Bitcoin capital)

앞선 섹션에서 스택스가 비트코인 위에 탈중앙 경제를 구축할 수 있게 해 주며, 그 핵심 요소로 PoX가 작동한다고 이야기했습니다.

그 이유는 두 가지입니다. 첫째, 여기서 다룬 PoX 채굴 과정의 일환으로, 각 스택스 블록의 해시가 비트코인 트랜잭션의 OP\_RETURN 오피코드에 기록됩니다. 익숙하지 않다면, OP\_RETURN 오피코드는 비트코인 트랜잭션에 최대 40바이트의 임의 데이터를 저장할 수 있게 해 줍니다.

이 Stack Exchange 답변은 이 오피코드의 배경과 역사에 대한 훌륭한 개요를 제공합니다.

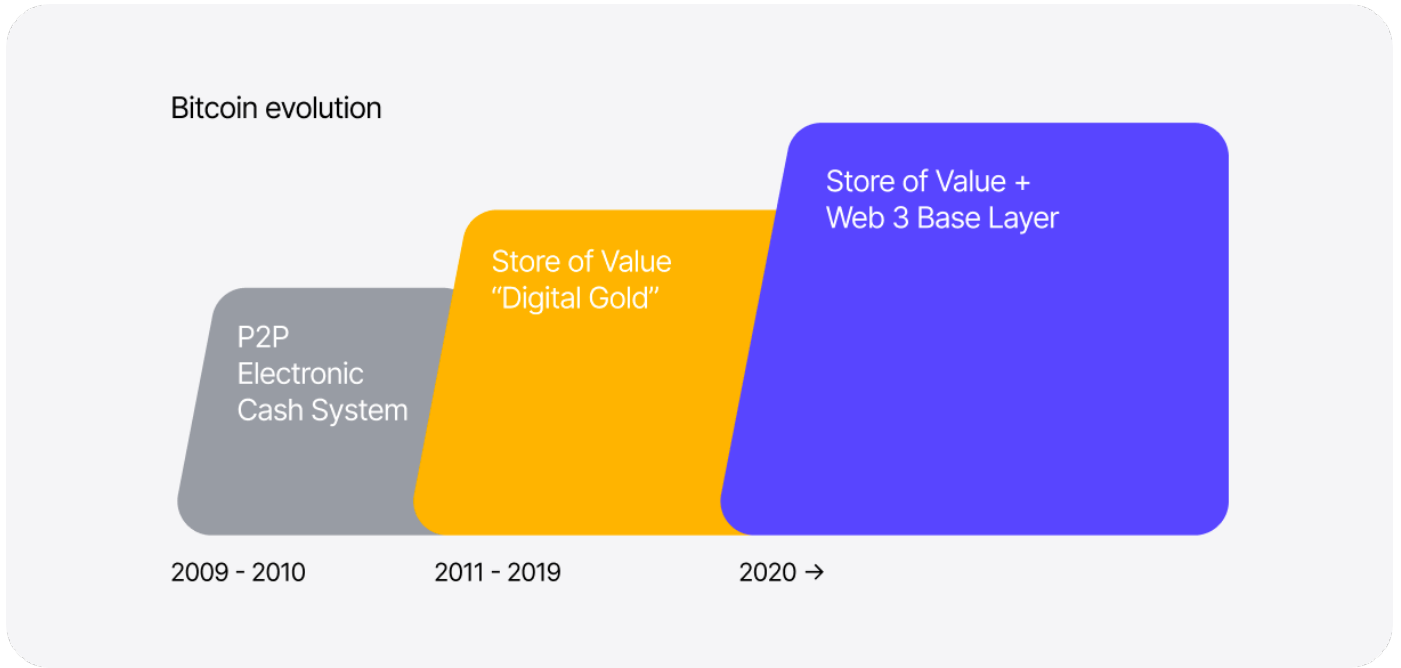
이것이 스택스가 비트코인의 보안을 상속받는 첫 번째 방식입니다. 즉, 스택스의 히스토리는 블록 단위로 비트코인 체인에 앵커링됩니다. 누구든 머클 루트를 사용해 이러한 해시를 검증하여 히스토리가 정확한지 확인할 수 있습니다.

추가로 Nakamoto 업그레이드 이후 스택스는 자체적으로 포크하지 않습니다. 프로토콜 수준에서 채굴자는 마지막으로 채굴된 스택스 블록 위에만 빌드해야 하므로, **스택스는 비트코인의 100% 해시레이트에 더해, 스택스 채굴자의 보안 예산으로도 보호됩니다.** 이 과정은 Block Production 섹션에서 더 자세히 다룹니다.

또한 PoX 과정의 일부로 각 스택스 블록은 자신이 어떤 비트코인 블록에 앵커링되었는지도 알고 있습니다. 스택스의 스마트 컨트랙트 언어인 Clarity에는 이 데이터를 읽기 위한 내장 함수가 있으며, 예를 들어 get-block-info는 여러 필드 가운데 해당 스택스 블록에 대응하는 비트코인 헤더의 해시를 제공하는 burnchain-header-hash를 반환합니다.

이를 통해 체인을 모니터링하고 특정 트랜잭션이 발생했는지 여부를 검증하여 Clarity 컨트랙트에서 특정 동작을 트리거하는 등 매우 흥미로운 작업을 수행할 수 있습니다. 이 방식은 Catamaran Swaps에서 실제로 사용되고 있으며, Zest와 같은 다른 흥미로운 프로젝트가 이 기능을 확장하려고 시도하고 있습니다.

이 모든 것의 궁극적인 목표는 웹3의 비전을 실현하는 것입니다. 즉, 비트코인을 결제 레이어로 삼고, 비트코인을 기초 탈중앙 화폐로 사용하여 탈중앙 경제를 구축하고 진정한 사용자 자산-데이터 소유권을 가능하게 하는 것입니다.



## Proof of Transfer 컨트랙트와 기술 세부

Proof of Transfer 기능은 [Clarity 스마트 컨트랙트](#)를 통해 Stacks 체인에 구현됩니다. 이 컨트랙트의 개요는 문서에서 확인할 수 있습니다.

스태킹과 Proof of Transfer의 원래 설계는 관련 SIP인 [SIP-007](#)을 읽어보세요. 또한 [Hiro의 API](#)를 활용해 관련 컨트랙트 주소를 포함한 Proof of Transfer 세부 정보를 얻을 수 있습니다.

## 다른 레이어 속 스택스

최근 생태계 전반에서 시장이 마침내 이 아이디어에 각성함에 따라 새로운 “비트코인 레이어”가 봇물 터지듯 등장하고 있습니다.

그러나 모든 비트코인 레이어가 동일하게 만들어진 것은 아닙니다. 이들 프로젝트 중 상당수는 유행에 편승한 베이퍼웨어이지만, 다양한 접근법을 통해 비트코인 위에 구축하고 비트코인 경제를 성장시키려는 성실한 노력을 기울이는 여러 프로젝트도 존재합니다.

[Bitcoin Layers 프로젝트](#)는 이러한 다양한 레이어를 학습하기 시작하기에 훌륭한 장소입니다. 더불어, 여기서는 가장 유망한 비트코인 L2 솔루션들과 스택스를 비교하여, 여러분이 모두를 이해하고 어떤 것을 사용할지에 대해 정보에 입각한 결정을 내릴 수 있도록 정리했습니다.

## 비트코인 레이어란 무엇인가?

특히 웹3처럼 새롭게 진화하는 생태계, 그리고 그보다도 더 새롭고 빠르게 진화하는 하위 생태계인 비트코인 레이어에서 용어를 정의하는 것은 중요합니다.

본 문서와 비교의 목적상, 다음과 같은 정의를 사용할 수 있습니다: 비트코인 레이어란, 확장성, 기능성, 혹은 그 둘 다를 향상시키기 위해 비트코인과 나란히 혹은 그 위에 구축된 별도의 분산 컴퓨팅 시스템입니다.

이 정의는 의도적으로 일반적이며, L2, 사이드체인, 연합형, 개방형 네트워크 등 다양한 프로젝트를 포괄합니다.

## 기술적 고려 vs 경제적 고려

블록체인, 특히 레이어 2 시스템을 설계할 때는 기술적 요인과 경제적 요인을 모두 고려해야 함을 이해하는 것이 중요합니다. 블록체인 시스템의 핵심 구성 요소가 “돈”인 이상, 시스템은 기술적으로 견고하면서도 경제적으로 효율적이어야 하며, 이 둘을 달성하는 동안 탈중앙성도 유지해야 합니다.

BTC를 L1에서 L2로 브릿지하는 “신뢰 기반 브릿지”를 만드는 것은 사소할 수 있지만, 그 경우 일반적으로 블록체인 기술의 목적 — 퍼미션리스하고 신뢰 최소화된 시스템 구축 — 을 훼손하게 됩니다.

동시에, 네트워크를 운영하는 탈중앙 행위자들의 경제적 인센티브를 고려하지 않는 훌륭한 기술적 해법은 장기적 채택과 생존 가능성 면에서 지속 가능한 경로를 갖지 못합니다.

이 균형 때문에 스택스는 현재의 설계를 선택했습니다. 즉, 비트코인 L2의 기술적 특징(보안 상속, 신뢰 최소화된 BTC 페그)을 달성하는 동시에, 생태계 참여자들이 장기적으로 시스템을 유지할 수 있도록 경제적 인센티브를 조화시키려는 것입니다.

이의 한 예로, Galaxy는 최근 이 주제에 대해 연구를 수행하며, 비트코인 롤업이 “월간 L2 트랜잭션 수수료에서 약 190만~963만 달러의 수익을 창출할 필요가 있다”고 결론지었습니다. 이는 상당한 규모이며, 비트코인 레이어 설계 시 기술적·경제적 요인을 모두 고려해야 함을 다시 한 번 부각합니다.

## 대중적인 비트코인 레이어 비교

### 라이트닝

라이트닝은 아마 가장 잘 알려진 비트코인 레이어이며, 주로 확장성 문제 해결을 목적으로 설계되었습니다. 라이트닝은 비트코인과 별개의 P2P 네트워크로 동작해, 참여자가 메인체인에서 라이트닝으로 BTC를 옮겨 라이트닝에서 여러 트랜잭션을 수행한 뒤, 최종 결과를 BTC 체인으로 보내 파이널라이즈할 수 있게 합니다.

이는 실제로 스택스가 해결하려는 문제와는 완전히 별개의 문제입니다. 라이트닝이 비트코인의 기존 기능을 훨씬 더 확장 가능하게 만드는 반면, 스택스는 지금은 할 수 없는 일을 가능하게 하기 위해 비트코인의 기능을 확장하고자 합니다.

본질적으로 라이트닝은 휘발성(에페메럴)으로, 상태 관리가 없습니다. 라이트닝 네트워크에서 일어난 일의 지속적 기록은 존재하지 않고, 오직 현재 채널만 있을 뿐입니다. 사용자가 채널을 닫고 그들의 트랜잭션이 비트코인 체인에 다시 기록되면, 그 기록은 사라집니다.

완전 표현력 있는 스마트 컨트랙트의 핵심 구성 요소는, 체인에서 일어난 모든 트랜잭션의 영구한 역사 기록을 유지한다는 점입니다.

비트코인은 현재 이를 수행하지만, 스크립팅 언어는 매우 제한적입니다. 따라서 라이트닝이 기존 비트코인 기능을 더 빠르게 수행하도록 하는 반면, 스택스는 새로운 기능을 추가하는 것을 목표로 합니다.

## RSK

스택스와 마찬가지로 RSK도 비트코인에 추가 기능을 제공하려 하지만, 그 접근 방식은 스택스와 다릅니다.

RSK는 머지 마이닝 체인으로, 비트코인과 동시에 채굴됩니다. 스택스는 자체 채굴자와 채굴 과정을 갖고 있으며, 토큰 가치의 함수로서 자체 경제적 가치와 보안을 보유합니다(아래에서 더 자세히 다룹니다).

여기에 대해 여러 관점이 있을 수 있습니다. RSK는 머지 마이닝이므로, 비트코인 채굴자가 RSK 블록도 채굴하며, RSK는 자체 토큰이 없습니다.

RSK는 비트코인 채굴자들의 옵트인에 의해서만 존재할 수 있고, 채굴 보상은 트랜잭션 볼륨에 크게 의존합니다.

이는 또한 별도 토큰 보유의 비용과 편익에 대한 더 넓은 논의를 열어 주며, 이는 아래 롤업을 논할 때 일부 다루겠습니다.

RSK는 EVM 호환이며, 스택스는 Clarity와 Clarity VM을 사용합니다.

## Liquid

Liquid는 비트코인으로 더 진보한 금융 기능을 여는 데 초점을 둔 연합형 네트워크입니다. 연합형이라는 특성상 Liquid는 개방형 네트워크가 아니며, 따라서 탈중앙적이지도 않습니다.

Liquid의 합의 메커니즘은 15명의 기능자(functionary)에 의해 관리되며, 이들이 트랜잭션 처리와 검증을 담당합니다. Liquid는 범용 애플리케이션을 지원하지 않고, 오직 금융 애플리케이션에만 초점을 둡니다.

다른 관점을 원한다면, Hiro가 스택스와 다른 비트코인 프로젝트를 비교한 훌륭한 글을 참조하세요.

## 비트코인 롤업

롤업은 탈중앙 애플리케이션을 확장하기 위한 흥미로운 발전입니다. 롤업에는 여러 유형이 있으며, 대체로 ZK 롤업과 옵티미스틱 롤업으로 나뉘지만, 다른 분류도 존재합니다(개요는 여기 참조).

롤업은 일반적으로 비트코인이나 이더리움 같은 레이어-1 블록체인 위에서 동작하는 레이어-2(L2) 기술로 간주됩니다. 롤업의 핵심 측면은 신뢰 없는(trustless) 특성으로, L1 체인에서 실행되는 로직이 롤업에서 일어난 일이 유효했는지를 판단할 수 있어야 한다는 점입니다. 이는 모든 유형의 롤업에 적용되지는 않으며, 정확한 정의에는 다소 모호함이 있습니다. 예컨대 소버린 롤업은 합의가 아니라 데이터 가용성(DA)에만 기반 L1을 사용합니다.

대부분의 이더리움 롤업은 이더리움 L1을 데이터 가용성 레이어이자 합의에 모두 사용합니다. 즉, 롤업 트랜잭션의 유효성은 이더리움 L1에서 실행되는 로직에 의해 결정됩니다. Celestia 같은 최신 시스템은 더 모듈러한 접근을 취해 DA와 합의를 분리하고 있습니다. DA를 분리하면 비트코인처럼 더 성숙하고 내구성 있는 체인을 DA로 사용할 수 있다는 흥미로운 측면이 생깁니다. 아래 이미지는 비트코인에서 가능한 사이드체인

과 두 가지 유형의 롤업을 대략 비교합니다(John Light이 [트위터](#)에 게시):

|                                       | Sidechain | Sovereign rollup | Validity rollup |
|---------------------------------------|-----------|------------------|-----------------|
| Trustless bridge                      | ✗         | ✗                | ✓               |
| Custom execution environment          | ✓         | ✓                | ✓               |
| Bitcoin-grade data availability       | 👤         | ✓                | ✓               |
| Bitcoin-grade double-spend resistance | 👤         | ✓                | ✓               |

이 이미지는 대략적으로, 오늘날 개발자가 비트코인 위에 소버린 롤업을 구축할 수 있지만, 롤업으로 BTC를 입출금하기 위해서는 “신뢰 기반” 설정이 필요함을 의미합니다. 실제로 사람들은 이미 이를 진행 중입니다 — 최근 [Rollkit 발표](#)를 참고하세요. 유효성 롤업(비트코인 L1이 롤업에서의 BTC 출금을 강제하는 모델)을 구축하려면 비트코인 L1의 수정이 필요합니다. 자세한 내용은 [이 개요](#)를 참조하세요.

여기서 중요한 미묘한 차이는, 첫 섹션에 링크된 Galaxy 연구 보고서에서 논의된 바와 같이 비트코인에서 롤업을 효과적으로 운영하는 데 필요한 비용입니다.

이제 스택스가 어떻게 동작하는지, 그리고 다른 레이어들 사이에서 어떤 위치를 차지하는지에 대해 탄탄한 이해를 갖게 되었습니다. 이제 일부 기술적 구현 세부를 파고들어 스택스가 실제로 어떻게 동작하는지 살펴봅시다.

## 재점적 인센티브 및 보안 예산

스택스 체인을 리오그하려면, 누군가는 현재 스테킹된 STX의 최소 70%를 장악하고 동시에 비트코인 자체에 대한 51% 공격을 수행해야 합니다. 이를 시가로 취득한다고 가정하면, 이 글을 쓰는 시점 기준으로 스테킹된 STX만 해도 거의 미화 10억 달러에 달하는 지출이 필요합니다.

여기에 더해, 스택스가 포크를 허용하지 않음으로써 비트코인 파이널리티를 달성하는 방식 때문에, 스택스의 보안 예산은 비트코인 채굴 파워의 51%에 도달합니다. 즉, 체인 상태를 되돌리려면 비트코인 체인 상태 역시 되돌려야 하기 때문입니다.

Stacker는 이제 블록에 서명하여 스택스 체인에 블록을 추가할 수 있는 새 권한을 갖습니다. 그러나 일부는 서명을 거부해 어떤 블록도 70% 서명 임계치를 달성하지 못하게 만들 수 있습니다. 이는 우연히 발생할 수 있으나, 경제적으로 합리적인 행태는 아닙니다—체인을 너무 오래 정지시키면 그들의 STX 가치는 하락하고, 재스테킹하거나 STX를 청산하거나 PoX를 활성화해 BTC를 벌 수도 없습니다. 또한 블록이 확정되지 않으면 채굴자들은 채굴을 중단하고, 그 결과 지속적인 PoX 지급도 사라집니다.

이 모든 것이 어떻게 동작하는지에 대한 기술적 세부는 [Block Production](#) 섹션에서 논의합니다.

# 네트워크 기초(Network Fundamentals)

이제 스택스가 무엇이며 어떻게 동작하는지에 대한 높은 수준의 이해를 갖추었으니, 스택스 비트코인 L2를 구성하는 모든 컴포넌트의 세부로 들어가 보겠습니다.

네트워크가 실제로 어떻게 구조화되어 있는지와 이를 구성하는 컴포넌트의 기초부터 시작하겠습니다.

## 토큰

스택스(STX) 토큰은 스택스 블록체인의 네이티브 토큰입니다. 가장 작은 단위는 1 마이크로-STX이며, 1,000,000 마이크로-STX가 1 STX를 이룹니다.

STX 수량은 정수(8바이트 길이)로 저장되어야 하며, 마이크로-STX 단위의 값을 나타냅니다.

## 수수료

수수료는 스택스 블록체인에서 채굴자가 트랜잭션을 확정하도록 인센티브를 제공합니다. 수수료는 추정 수수료율과 원시 트랜잭션의 바이트 크기에 기반해 계산됩니다. 수수료율은 시장에 의해 결정되는 변수입니다. 테스트넷에서는 1 마이크로-STX로 설정되어 있습니다.

수수료 추정치는 API의 [GET /v2/fees/transfer](#) 엔드포인트를 통해 얻을 수 있습니다.

이 예시는 외부 도구인 [Hiro의 스택스 API](#)를 사용합니다. 자체 노드를 실행하거나 노드에 연결하길 원한다면, 네이티브 [스택스 API](#)도 사용할 수 있습니다.

```
# 메인넷의 경우 `testnet`을 `mainnet`으로 교체
curl 'https://api.testnet.hiro.so/v2/fees/transfer'
```

API는 수수료를 정수로 응답합니다:

1

[The Stacks Transactions JS library](#)는 다음 작업에 대한 수수료 추정을 지원합니다:

- 토큰 전송(`estimateTransfer`)
- 컨트랙트 배포(`estimateContractDeploy`)
- 읽기 전용이 아닌 컨트랙트 호출(`estimateContractFunctionCall`)

JavaScript 이외의 언어로 구현이 필요하다면, [이 레퍼런스 구현](#)을 확인하세요.

## 논스

모든 계정에는 해당 계정으로 처리된 트랜잭션 수를 나타내는 [논스 속성](#)이 있습니다. 논스는 일회성 코드로, 새 계정의 경우 0에서 시작하여 트랜잭션마다 1씩 증가합니다.

논스는 모든 트랜잭션에 추가되어, 트랜잭션이 순서대로 처리되고 중복 처리되지 않도록 식별을 돕습니다.

합의 메커니즘은 또한 두 가지 방식으로 트랜잭션의 “재생(replay)”을 방지합니다. 첫째, 노드는 새로운 트랜잭션에서 지출 조건을 만족시키기 위해 미사용 트랜잭션 출력(UTXO)을 조회합니다. 둘째, 노드 간에 전송되는 메시지는 시퀀스 번호를 검토합니다.

새 토큰 전송 트랜잭션을 구성할 때에는, 해당 계정의 최신 논스를 조회하여 설정해야 합니다.

API에는 [논스 처리를 단순화](#)하는 엔드포인트가 제공됩니다.

## 쿼리

스택스 네트워크의 상세 정보는 [Stacks Blockchain API](#)를 사용해 조회할 수 있습니다.

## 상태 체크

[status checker](#)는 스택스 블록체인의 상태를 빠르게 검토할 수 있는 사용자 인터페이스를 제공합니다.

## 네트워크 정보

네트워크 정보는 `GET /v2/info` 엔드포인트로 가져올 수 있습니다:

```
# 메인넷의 경우 `testnet`을 `mainnet`으로 변경
curl 'https://api.testnet.hiro.so/v2/info'
```

# 메인넷과 테스트넷

스택스에는 서로 다른 목적을 위한 메인넷과 몇 가지 테스트넷이 있습니다. 메인넷과 테스트넷은 완전히 별개의 네트워크이며, 토큰은 상호 간에 전송할 수 없습니다.

## 메인넷

스택스 메인넷은 비트코인 메인넷과 직접 연결되어 있으며, 토큰에 실제 화폐 가치가 있는 네트워크입니다. 이는 프로덕션 네트워크로서 그렇게 취급되어야 합니다.

메인넷 활동은 [Hiro의 블록 탐색기](#)에서 확인할 수 있습니다.

## Primary와 Nakamoto 테스트넷

Primary Testnet과 Nakamoto Testnet 간에는 주목할 만한 차이가 있습니다. 아래 표는 필요에 가장 부합하는 네트워크를 선택하는 데 도움을 줍니다.

| Attributes            | Nakamoto Testnet                                                                                                                                      | Primary Testnet                                                                                                                                                    |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Stacking Cycle Length | 3 days                                                                                                                                                | 1 week                                                                                                                                                             |
| Description           | 최신 기능을 담은 릴리스 후보 (Release Candidate)가 더 자주 적용되는 최전선(bleeding edge) 환경.                                                                                | 메인넷 직전 단계로, 안정 릴리스만 적용.                                                                                                                                            |
| Usage Recommendations | <ul style="list-style-type: none"> <li>빈번한 리셋을 개의치 않고 출시 직후의 최신 기능을 테스트하고 싶다면 사용</li> <li>다양한 stacking-signer 시나리오를 빠른 피드백 루프로 시험하고 싶다면 사용</li> </ul> | <ul style="list-style-type: none"> <li>더 안정적인 릴리스를 선호하고 잦은 리셋/업데이트를 원치 않는다면 사용</li> <li>최신 기능을 가장 먼저 테스트할 필요가 없다면 사용</li> <li>더 긴 Stacking 주기를 선호한다면 사용</li> </ul> |
| Lifespan              | Nakamoto Testnet은 sBTC가 메인넷에 출시될 때까지 유지됩니다.                                                                                                           | Primary Testnet은 영구적으로 유지·관리됩니다.                                                                                                                                   |
| Explorer              | <a href="#">Nakamoto Explorer</a>                                                                                                                     | <a href="#">Primary Explorer</a>                                                                                                                                   |

## Primary 테스트넷 관련 중요 정보:

- 코어 개발자들이 BTC Regtest Explorer를 작업 중입니다. 그동안 Wallet, Explorer, API에서 BTC 트랜잭션으로 연결되는 링크는 아무 곳에도 도달하지 않습니다. 이는 예상된 동작이며 추후 해결될 예정입니다. 모든 STX 트랜잭션은 [Explorer](#)에서 추적할 수 있습니다.
- Signer 온보딩을 시작하고, 컨트랙트를 배포하며, 앱을 테스트할 수 있습니다. 이전 테스트넷의 모든 기능이 사용 가능합니다.
- 이전 테스트넷 데이터는 보관(아카이브)되며 2024년 6월 말까지 계속 [열람 가능](#)합니다.
- Faucet 및 tSTX:**
  - [Faucet 주소](#)와 한도는 동일하게 유지됩니다.
  - Primary Testnet에서 Signer 온보딩을 위해 현재 일일 한도보다 많은 tSTX가 필요하다면, 생태계 내 메인 연락 창구에 문의해 주세요.

## 테스트넷

테스트넷은 스택스 메인넷과 분리된 블록체인으로, 스테이징 환경에 해당합니다. 이는 개발자가 앱, 스마트 컨트랙트, 또는 프로토콜 변경 사항을 프로덕션과 유사한 환경에서 테스트하는 데 사용하는 네트워크입니다.

테스트넷은 메인넷과 거의 동일한 속도로 블록을 생성합니다. 평균적으로 약 10분마다 1블록이 생성됩니다. 스택스 테스트넷은 리셋되는 일이 거의 없습니다.

## Faucets

테스트넷 Faucet은 무료 스택스 토큰(STX)을 제공하여 테스트에 활용할 수 있도록 합니다. 이 토큰은 메인넷의 STX와 동일하지 않으며 가치도 없습니다. 테스트넷 STX를 얻는 방법은 몇 가지가 있습니다.

## Hiro

[Hiro Explorer Sandbox](#)에서 STX를 받을 수 있으며, 또는 [API](#)를 사용할 수도 있습니다.

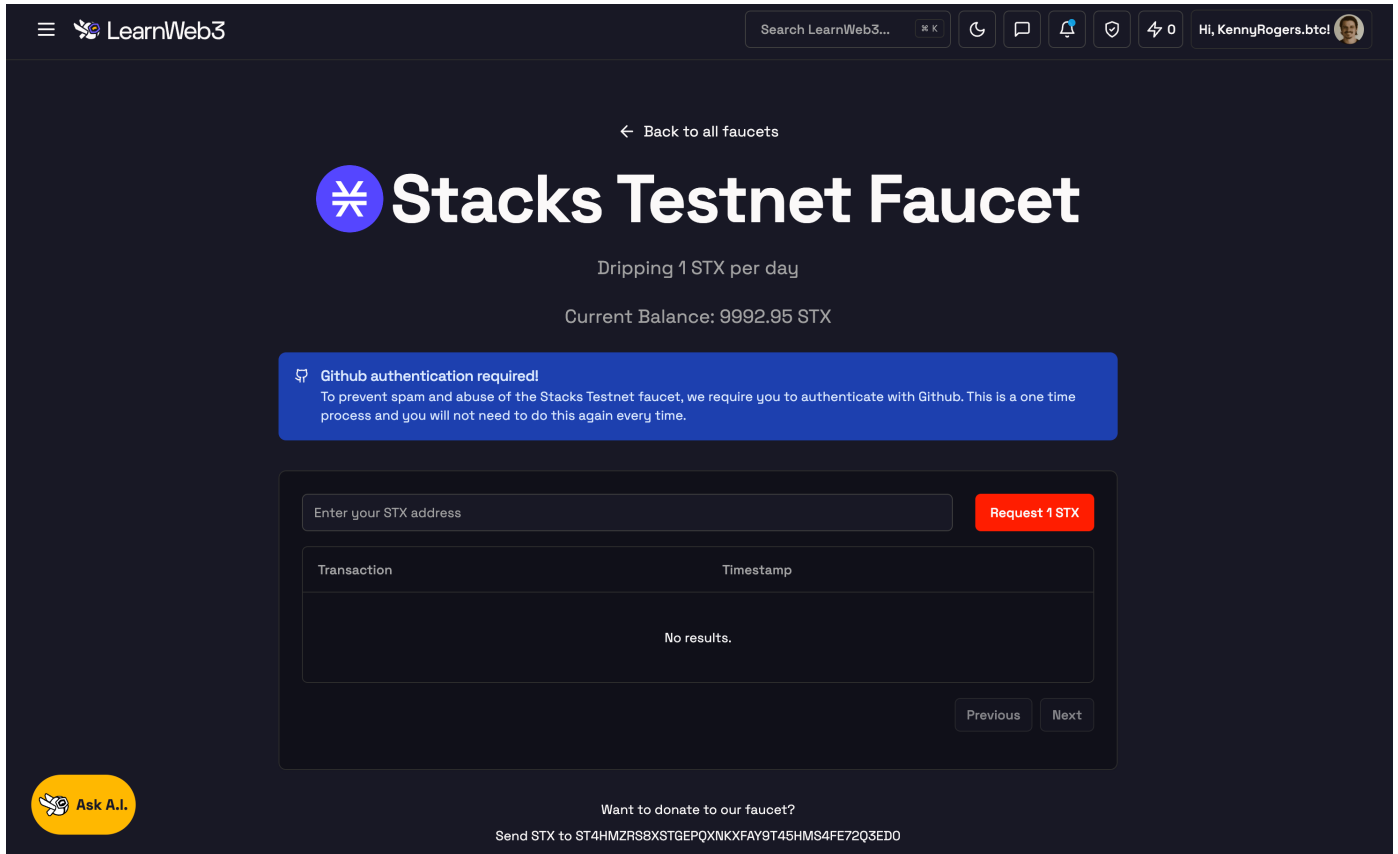
Explorer Sandbox 내에서 STX 토큰을 받으려면, 왼쪽의 "Faucet" 탭으로 이동해 "Request STX" 버튼을 클릭하세요.

I want to stack 버튼을 클릭해 Stacking을 직접 체험해 볼 수도 있습니다.

Explorer Sandbox를 사용하려면 스택스 지갑으로 로그인해야 합니다.

## LearnWeb3

또는 [LearnWeb3 faucet](#)을 사용할 수도 있습니다.



## Testnet API

테스트넷용 호스팅 스택스 Blockchain API의 베이스 URL은 다음과 같습니다:

```
https://api.testnet.hiro.so/
```

# 계정(Accounts)

## 개요

스택스는 비트코인과 같은 UTXO 모델이 아닌, 이더리움과 더 유사한 계정 기반 모델을 사용합니다. UTXO 모델에서 네트워크는 원장처럼 동작하며, 각 UTXO는 현금 지폐에 비유할 수 있습니다.

계정 기반 모델에서는 각 계정이 잔액과 연계되며, 그 잔액은 더해지거나 차감될 수 있습니다.

스택스 계정은 스택스(STX) 토큰과 같은 자산을 보유하는 주체입니다. 계정은 주소, 개인 키, nonce (nonce), 그리고 하나 이상의 자산 잔액을 가집니다.

스택스에서 사용되는 암호학적 서명 알고리즘은 secp256k1입니다.

또한 Ed25519는 VRF(Verifiable Random Function)에 한해서만 사용됩니다.

계정 소유자의 동작 없이 자산이 계정을 떠날 수 없습니다.  
자산(및 계정의 잔액)에 대한 모든 변경은 해당하는 트랜잭션을 필요로 합니다.

트랜잭션 유형은 반드시 토큰 전송일 필요는 없습니다. 컨트랙트 배포 및 컨트랙트 호출 트랜잭션도 계정의 잔액을 변경할 수 있습니다.

## 생성(Creation)

계정은 24개 단어의 니모닉 구문에서 생성됩니다. 이는 흔히 **시드 구문(seed phrase)** 이라고도 불립니다. 시드 구문은 스택스 계정에 접근하는 수단을 제공합니다.

시드 구문을 분실하면, 해당 계정에 대한 접근은 복구할 수 없습니다. 어떤 사람이나 조직도 분실된 시드 구문을 되찾아 줄 수 없습니다.

새로운 스택스 계정을 생성하는 가장 쉬운 방법은 스택스 CLI를 사용하는 것입니다:

```
# CLI 전역 설치
npm install --global @stacks/cli

# 새 계정을 생성하고 세부 정보를 새 파일에 저장
# '-t' 옵션은 테스트넷 계정을 생성합니다
stx make_keychain -t > cli_keychain.json
```

자세한 내용은 Stacks CLI reference를 확인하세요.

| Field              | Description                                                  |
|--------------------|--------------------------------------------------------------|
| mnemonic           | 계정 접근에 사용되는 24단어 시드 구문. <u>BIP39</u> 표준을 사용해 256비트 엔트로피로 생성됨 |
| keyInfo.privateKey | 계정의 개인 키. 토큰 전송에 필요하며 흔히 senderKey라고 불림                      |
| keyInfo.address    | 계정의 스택스 주소                                                   |
| keyInfo.btcAddress | 계정에 대응하는 BTC 주소                                              |
| keyInfo.wif        | 압축 형식의 btcAddress 개인 키                                       |
| keyInfo.index      | 계정의 논스. 0에서 시작                                               |

새로운 개인 키가 생성될 때마다 자동으로 새 계정이 생성됩니다.  
따라서 스택스 블록체인에서 계정을 수동으로 인스턴스화할 필요는 없습니다.

주소는 공개 키의 [SHA256](#) 해시를 취한 후 [RIPEMD-160](#) 해시를 적용하여 생성됩니다.  
BTC 주소는 [Base58Check](#)로 인코딩되며, 스택스 주소는 [c32check](#)를 사용합니다.  
공개 키로부터 주소를 유도하는 과정은 인터넷 연결 없이도 가능하며, 예를 들어 [c32check](#)의 `c32addressDecode` 메서드를 사용할 수 있습니다.

---

CLI 대신 [Stacks Transactions JS](#) 라이브러리를 사용할 수도 있습니다:

마지막으로, [Leather](#), [Xverse](#), [Asigna](#)와 같은 스택스 지원 지갑을 사용해 새로운 계정을 생성할 수도 있습니다.

# 인증(Authentication)

## 개요

이 가이드는 스택스 블록체인에서 인증이 어떻게 수행되는지 설명합니다.

인증은 사용자가 자격 증명과 개인 정보를 완전한 통제하에 유지하면서 앱에 본인을 증명할 수 있도록 합니다.

여러분의 앱에 등록된 사용자는 이후 [Bitcoin Name System](#)을 지원하는 다른 앱에도 인증할 수 있으며, 그 반대도 가능합니다.

---

## 웹2 vs 웹3 인증

웹2 세상에서는 사용자 이름과 비밀번호로 인증하는 데 익숙할 것입니다. 이때 사용자의 정보는 데이터베이스에 저장되고, 입력된 비밀번호는 해시되어 저장된 해시와 비교됩니다. 일치하면 사용자가 로그인됩니다.

웹3 인증은 조금 다르게 동작합니다. 웹3의 핵심 철학 중 하나는 데이터 소유권이며, 이는 사용자가 인증을 포함한 자신의 데이터에 대한 통제권을 가진다는 의미입니다.

비트코인 및 스택스 세계에서 인증은 암호학을 사용하여 개인 키, 공개 키, 주소를 생성합니다.

여기서는 기본적인 부분을 다루지만, 더 깊이 학습하고 싶다면 [Learn Me a Bitcoin](#)을 참고하면 좋습니다.

스택스 계정을 생성하려면, 앞서 다룬 24단어 니모닉 구문에서 개인 키를 생성합니다. 이 개인 키는 일방향 해시 함수로 공개 키를 생성하는 데 사용됩니다. 즉, 개인 키에서 공개 키를 도출할 수 있지만 그 반대는 불가능합니다.

사용자의 개인 키는 보안의 핵심이며 인증에 사용됩니다. 개인 키를 절대 잃어버리지 마세요.

예를 들어 [Leather](#) 같은 지갑 앱을 사용해 StackingDAO 같은 dapp에 인증하려는 경우, 지갑은 제 개인 키를 사용해 제가 해당 주소의 소유자임을 증명합니다.

지갑은 이후 제 주소와 공개 키를 서명과 함께 앱에 전달합니다.

서명은 개인 키를 실제로 드러내지 않고도 제가 개인 키를 소유하고 있음을 증명하는 메커니즘입니다. 이 방식으로 스택스 인증을 지원하는 모든 앱에 동일한 지갑과 주소로 로그인할 수 있습니다.

웹3 인증에 대한 개념적 입문은 Third Web의 [primer](#)를 참고하세요.

좀 더 실용적인 입문은 [퀵스타트 튜토리얼](#)과 [Hiro의 Stacks.js 문서](#)를 살펴보세요.

---

## 작동 방식

스택스의 인증 흐름은 중앙화 로그인 서비스(OAuth 등)에서 사용하는 전형적인 클라이언트-서버 흐름과 유사합니다. 그러나 스택스에서는 인증 흐름이 전적으로 클라이언트 측에서 이루어집니다.

여기서는 과정 이해를 돕기 위한 설명이며, 대부분의 기능은 지갑과 사용하는 JS 라이브러리가 처리합니다. 자세한 정보는 [Stacks.js 문서](#)를 참고하세요.

[Leather 지갑](#) 같은 앱과 인증자는 인증 과정에서 두 개의 토큰을 주고받으며 통신합니다.

요청 앱은 인증자에게 `authRequest` 토큰을 보냅니다. 사용자가 인증을 승인하면, 인증자는 앱에 `authResponse` 토큰으로 응답합니다.

이러한 토큰은 [JSON Web Token\(JWT\) 표준](#)을 기반으로 하며, 비트코인 및 다수의 암호화폐에서 사용하는 `secp256k1` 곡선을 추가 지원합니다. 토큰은 URL 쿼리 문자열을 통해 전달됩니다.

사용자가 앱 인증을 선택하면, 앱은 동일한 이름의 매개변수를 사용해 URL 쿼리 문자열로 `authRequest` 토큰을 인증자에게 보냅니다:

```
https://wallet.hiro.so/...?authRequest=j902120cn829n1jnvoa...
```

인증자는 요청을 수신하면 *일시적 트랜짓 키(ephemeral transit key)* 를 사용해 앱을 위한 `authResponse` 토큰을 생성합니다.

일시적 트랜짓 키는 해당 앱 인스턴스에만 사용되며, 이 경우 `authRequest`에 서명하는 데 활용됩니다.

앱은 요청 생성 시 일시적 트랜짓 키를 저장합니다. 트랜짓 키의 공개 키 부분은 `authRequest` 토큰에 포함됩니다. 인증자는 이 공개 키를 사용해 *앱 개인 키*를 암호화하여 `authResponse`로 반환합니다.

인증자는 사용자의 *identity address private key* 와 앱의 도메인을 조합하여 앱 개인 키를 생성합니다. 앱 개인 키의 용도는 세 가지입니다:

1. 앱이 사용자의 Gaia 허브 내 스토리지 버킷에 접근할 수 있도록 자격 증명을 생성
2. 사용자의 Gaia 스토리지에 앱을 위해 저장되는 파일의 종단간 암호화에 사용

### 3. 앱이 기타 암호 연산을 수행하는 데 사용할 수 있는 암호학적 비밀로 기능

마지막으로, 앱 개인 키는 결정적(deterministic)입니다. 즉, 특정 스택스 주소와 도메인에 대해 항상 동일한 개인 키가 생성됩니다.

---

## 키 페어

스택스 인증은 공개 키 암호 방식 전반, 특히 `secp256k1` 곡선을 사용하는 ECDSA를 폭넓게 활용합니다.

다음 섹션에서는 사용되는 세 가지 공개-개인 키 쌍에 대해, 생성 방법, 사용 위치, 개인 키가 누구에게 공개 되는지를 설명합니다.

### Transit private key

트랜짓 개인 키는 인증 과정에서 인증자가 앱으로 전달해야 하는 비밀을 암호화하는 데 사용되는 일시적 키입니다. 이는 인증 응답 시작 시 앱에 의해 무작위로 생성됩니다.

트랜짓 개인 키에 대응하는 공개 키는 인증 요청 토큰의 `public_keys` 키에 단일 요소 배열로 저장됩니다. 인증자는 이 공개 키를 사용해 앱 개인 키 같은 비밀 데이터를 암호화하고, 사용자가 앱에 로그인할 때 이를 앱으로 다시 전송합니다. 트랜짓 개인 키는 앱 인증 요청에 서명합니다.

### Identity address private key

Identity address private key는 사용자의 키체인 구문에서 파생되며, 사용자가 앱에 로그인할 때 선택한 스택스 사용자 이름의 개인 키입니다. 이는 사용자가 소유하는 비밀로, 인증자의 사용자 인스턴스를 절대 벗어나지 않습니다.

이 개인 키는 사용자가 해당 앱에 로그인하는 것을 승인했음을 나타내기 위해 앱의 인증 응답 토큰에 서명합니다.

### App private key

App private key는 사용자의 identity address private key에서 `domain_name`을 입력값으로 하여 생성되는 앱 전용 개인 키입니다.

앱 개인 키는 인증자에 의해 트랜짓 공개 키로 암호화되어 매 인증 시 앱과 안전하게 공유됩니다. 트랜짓 키는 클라이언트 측에만 저장되므로, 서버나 인터넷 제공자가 앱 개인 키를 엿볼 수 있는 중간자 공격을 방지할 수 있습니다.

# 비트코인 네임 서비스(Bitcoin Name System)

Bitcoin Name System(BNS)는 중앙화된 제어 지점 없이 스택스 사용자 이름을 오프체인 상태에 바인딩하는 네트워크 시스템입니다.

스택스 V1 블록체인은 1차 네임 연산(first-order name operations)을 통해 BNS를 구현했습니다. 스택스 V2에서는 BNS가 제네시스 블록에 로드된 스마트 컨트랙트를 통해 구현됩니다.

BNS의 이름은 세 가지 속성을 가집니다:

- **이름은 전역적으로 고유합니다.** 프로토콜은 이름 충돌을 허용하지 않으며, 모든 정상 노드는 동일한 이름을 동일한 상태로 해석합니다.
- **이름은 사람이 이해할 수 있습니다.** 각 이름은 생성자가 선택합니다.
- **이름은 강하게 소유됩니다.** 이름의 소유자만 그 이름이 해석되는 상태를 변경할 수 있습니다. 구체적으로, 이름은 하나 이상의 ECDSA 개인 키에 의해 소유됩니다.

스택스 블록체인은 각 노드의 BNS 뷰가 전 세계의 다른 모든 노드와 동기화되도록 보장합니다. 따라서 한 노드에서의 쿼리는 다른 노드에서도 동일합니다. 스택스 블록체인 노드는 이름 소유자가 자신의 이름에 최대 40Kb의 오프체인 상태를 바인딩할 수 있도록 허용하며, 이는 P2P 네트워크를 통해 모든 다른 스택스 블록체인 노드로 복제됩니다.

개발자에게 가장 큰 의미는, BNS에서 이름 상태 읽기는 빠르고 저렴하지만, 이름 상태 쓰기는 느리고 비용이 많이 든다는 점입니다. 이름 등록과 수정에는 블록체인에 트랜잭션을 전송해야 하며, 충분히 확정되기 전까지 BNS 노드가 이를 처리하지 않기 때문입니다. 사용자와 개발자는 BNS 트랜잭션을 보내기 위해 필요한 암호 화폐(STX)를 확보하고 사용해야 합니다.

---

## 이름 시스템의 동기

우리는 일상에서 이름 시스템에 의존하며, 이는 다양한 애플리케이션에서 중요한 역할을 합니다.

예를 들어, 소셜 미디어에서 친구를 찾을 때 해당 플랫폼의 이름 시스템을 사용해 그들의 이름을 프로필로 해석합니다.

웹사이트를 찾을 때는 DNS를 통해 호스트 이름을 호스트의 IP 주소로 해석합니다.

Git 브랜치를 체크아웃할 때는 Git 클라이언트를 사용해 브랜치 이름을 커밋 해시로 해석합니다.

PGP 키서버에서 누군가의 PGP 키를 찾을 때는 키 ID를 공개 키로 해석합니다.

그렇다면 이름에 대해 우리가 바라는 성질은 무엇일까요? BNS에서는 이름이 전역적으로 고유하고, 사람이 이해 가능하며, 강하게 소유됩니다. 그러나 위 예시들을 보면 각 경우는 이 중 두 가지 속성만 만족합니다. 이는 그 유용성을 제한합니다.

- DNS와 소셜 미디어에서는 이름이 전역적으로 고유하고 사람이 읽을 수 있지만, 강하게 소유되지 않습니다. 시스템 운영자가 최종 권한을 가집니다.
  - **문제:** 클라이언트는 특정 이름이 무엇으로 해석되는지 시스템이 올바른 선택을 한다는 점을 신뢰해야 합니다.
- Git에서는 브랜치 이름이 사람이 이해할 수 있고 강하게 소유되지만, 전역적으로 고유하지는 않습니다.
  - **문제:** 이름이 충돌 상태를 참조할 수 있으므로, 개발자는 모호성을 해결할 다른 메커니즘을 찾아야 합니다.
- PGP에서는 이름이 키 ID입니다. 전역적으로 고유하고 암호학적으로 소유되지만, 사람이 읽을 수 없습니다.
  - **문제:** 의미적 정보를 담지 않으므로 사용자가 기억하기 어렵습니다.

BNS 이름은 이 세 가지 속성을 모두 가지며, 이러한 문제를 해결합니다. 따라서 네트워크 애플리케이션을 구축하는 강력한 도구가 됩니다.

BNS를 통해 다음을 포함해 다양한 작업을 할 수 있습니다:

- 탈취될 수 없는 호스트 이름을 가진 도메인 이름 서비스 구축
- 피싱 공격으로 훔칠 수 없는 사용자 이름을 가진 소셜 미디어 플랫폼 구축
- 저장소 브랜치가 충돌하지 않는 버전 관리 시스템 구축
- 사용자가 서로의 키를 쉽게 발견하고 기억할 수 있는 공개 키 인프라 구축

---

## BNS의 조직 구조

BNS 이름은 전역 네임 계층 구조에 따라 조직됩니다. 네임과 관련된 세 가지 레이어가 있습니다:

- **네임스페이스:** 계층의 최상위 이름입니다. DNS 최상위 도메인과 유사합니다. 기존 BNS 네임스페이스에는 `.id`, `.podcast`, `.helloworld` 등이 있습니다. 다른 모든 이름은 정확히 하나의 네임스페이스에 속합니다. 누구나 네임스페이스를 만들 수 있지만, 지속되려면 *컨차*되어야 하며, 그래야 누구나 그 안에 이름을 등록할 수 있습니다. 네임스페이스는 창작자가 소유하지 않습니다.
- **BNS 이름:** 블록체인에 직접 기록된 이름입니다. 이 이름의 소유권과 상태는 블록체인 트랜잭션을 전송해 제어됩니다. 예: `verified.podcast`, `muneeb.id`. 네임스페이스가 존재한다면 누구나 BNS 이름을 만들 수 있습니다.
- **BNS 서브도메인:** 오프체인에 저장되지만 블록체인에 앵커링된 이름입니다. 상태와 소유권은 P2P 네트워크 데이터에 있습니다. 서브도메인은 별도의 개인 키로 소유되지만, 반드시 상위 BNS 이름 소유자가 그 상태를 브로드캐스트해야 합니다. 예: `jude.personal.id`, `podsaveamerica.verified.podcast`. 네임스페이스 및 BNS 이름과 달리, 서브도메인의 상태는 블록체인 합의 규칙의 일부가 아닙니다.

아래는 이러한 이름 객체 간의 유사성과 차이를 요약한 기능 비교 매트릭스입니다:

| Feature                                | Namespaces | BNS names | BNS Subdomains |
|----------------------------------------|------------|-----------|----------------|
| Globally unique                        | X          | X         | X              |
| Human-meaningful                       | X          | X         | X              |
| Owned by a private key                 |            | X         | X              |
| Anyone can create                      | X          | X         | [1]            |
| Owner can update                       |            | X         | [1]            |
| State hosted on-chain                  | X          | X         |                |
| State hosted off-chain                 |            | X         | X              |
| Behavior controlled by consensus rules | X          | X         |                |
| May have an expiration date            |            | X         |                |

[1] BNS 이름 소유자의 협력이 필요

## 네임스페이스(Namespaces)

네임스페이스는 BNS의 최상위 이름 객체입니다.

네임스페이스는 그 안의 이름에 대해 몇 가지 속성을 제어합니다:

- 등록 비용
- 갱신 주기
- 등록 수수료 수령자(있다면)
- 초기 이름 시드를 제공할 수 있는 권한자

현재 가장 큰 네임스페이스는 `.id`입니다. `.id` 네임스페이스의 이름은 사용자 아이덴티티 해석을 위해 사용 됩니다. 짧은 `.id` 이름은 긴 이름보다 비싸며, 소유자는 2년마다 갱신해야 합니다. 등록 수수료는 특정인에게 지급되지 않고 “블랙홀” 주소로 전송되어 사용 불가능하게 됩니다(의도는 ID 스쿼팅 방지).

DNS와 달리, 누구나 네임스페이스를 생성하고 그 속성을 설정할 수 있습니다. 네임스페이스는 선착순으로 생성되며, 한 번 생성되면 영구적으로 유지됩니다.

그러나 네임스페이스 생성은 무료가 아닙니다. 생성자는 암호화폐를 소각해야 하며, 네임스페이스가 짧을수록 더 많은 암호화폐를 소각해야 합니다. 예: Blockstack PBC는 2015년에 `.id` 네임스페이스를 생성하기 위해 40 BTC를 소각했습니다(트랜잭션 `5f00b8e609821edd6f3369ee4ee86e03ea34b890e242236cdb66ef6c9c6a1b281`).

네임스페이스는 1~19자 길이일 수 있으며, `a-z`, `0-9`, `-`, `_` 문자로 구성됩니다.

---

## 서브도메인(Subdomains)

BNS 이름은 강하게 소유되는데, 이는 소유자의 개인 키로 유효한 트랜잭션을 생성해 해당 이름의 존 파일 해시와 소유권을 갱신할 수 있기 때문입니다. 그러나 이는 블록체인 트랜잭션 비용을 지불해야 하며, 등록 및 연산 속도를 블록체인의 트랜잭션 처리량에 제한하게 됩니다.

BNS는 이를 서브도메인으로 극복합니다. **BNS 서브도메인**은 상태와 소유자가 블록체인 밖에 저장되지만, 존재와 운영 기록이 블록체인에 앵커링되는 이름입니다. 서브도메인도 전역적으로 고유하고, 강하게 소유되며, 사람이 읽을 수 있습니다. BNS는 그들에게 독자적 상태와 공개 키를 부여합니다. 온체인 이름과 달리, 서브도메인은 배치로 브로드캐스트되므로 저렴하게 생성·관리할 수 있습니다. 하나의 블록체인 트랜잭션으로 최대 120개의 서브도메인 연산을 전송할 수 있습니다.

이는 서브도메인 기록을 BNS 이름 존 파일에 저장함으로써 달성됩니다. 온체인 이름 소유자는 서브도메인 연산을 DNS 존 파일의 TXT 레코드로 인코딩해 브로드캐스트합니다. 존 파일은 `NAME_UPDATE` 트랜잭션으로 새로운 존 파일 해시를 설정하고 복제됩니다. 이렇게 되면 서브도메인 연산도 모두 복제되어 온체인 트랜잭션에 앵커링됩니다. BNS 노드의 합의 규칙은 **유효한** `NAME_UPDATE` 트랜잭션에서의 **유효한** 서브도메인 연산만 저장되도록 보장합니다.

## 서브도메인 생성 및 관리

온체인 이름과 달리, 서브도메인 소유자는 자신의 서브도메인 연산을 브로드캐스트하기 위해 온체인 이름 소유자의 도움을 받아야 합니다. 구체적으로:

- 서브도메인 생성 트랜잭션은 해당 접미사를 공유하는 온체인 이름의 소유자만 처리할 수 있습니다. 예: `.res_publica.id`로 끝나는 서브도메인 이름을 생성하려면 `res_publica.id` 소유자만 서브도메인 생성 트랜잭션을 브로드캐스트할 수 있습니다.
- 서브도메인 이전 트랜잭션은 이를 생성한 온체인 이름 소유자만 브로드캐스트할 수 있습니다. 예: `cicero.res_publica.id` 소유자는 자신의 공개 키를 변경하려면 `res_publica.id` 소유자에게 서브도메인 이전 트랜잭션을 브로드캐스트해 달라고 요청해야 합니다.
- 서브도메인 생성 또는 이전을 보내려면, 온체인 이름 소유자의 모든 존 파일(zone file)이 Atlas 네트워크에 존재해야 합니다. 이를 통해 BNS 노드는 새로운 존 파일을 처리할 때 충돌하는 서브도메인 생성·이전 연산의 **부재**를 증명할 수 있습니다.

- 서브도메인 업데이트 트랜잭션은 *어떤* 온체인 이름 소유자라도 브로드캐스트할 수 있지만, 서브도메인 소유자는 협력해 줄 소유자를 찾아야 합니다. 예: `verified.podcast` 소유자는 `cicero.res_publica.id` 소유자가 생성한 서브도메인 업데이트 트랜잭션을 브로드캐스트할 수 있습니다.

즉, 서브도메인을 생성하려면, 서브도메인 소유자가 원하는 이름에 대한 서브도메인 생성 연산을 만든 뒤 이를 온체인 이름 소유자에게 전달해야 합니다.

일단 생성된 후에는 서브도메인 소유자는 어떤 온체인 이름 소유자라도 이용해 서브도메인 업데이트 연산을 브로드캐스트할 수 있습니다. 이를 위해 서브도메인 소유자가 필요한 연산을 생성·서명하고, 온체인 이름 소유자에게 전달하면, 그가 다른 서브도메인 연산과 함께 DNS 존 파일에 패키징해 네트워크에 브로드캐스트합니다.

서브도메인 소유자가 자신의 서브도메인 주소를 변경하려면, 서브도메인 이전 연산에 서명한 후 해당 서브도메인을 생성한 온체인 이름 소유자에게 전달해야 합니다. 이후 이는 존 파일에 패키징되어 브로드캐스트됩니다.

---

## 서브도메인 등록자(Registrar)

서브도메인 이름은 저렴하기 때문에, 개발자가 애플리케이션을 위해 서브도메인 등록자를 운영하고 싶어 할 수 있습니다. 예를 들어, `personal.id`는 사용자가 비트코인을 소비하지 않고도 사용자 이름을 등록할 수 있도록 사용됩니다.

개발자가 서브도메인 연산을 브로드캐스트할 수 있도록 [BNS 서브도메인 등록자](#)의 참조 구현을 제공합니다. 사용자는 여전히 자신의 서브도메인 이름을 소유하며, 등록자는 단지 특정 애플리케이션 맥락에서 이를 등록·관리할 수 있는 편리한 방법을 제공할 뿐입니다.

---

## BNS와 DID 표준

BNS 이름은 [Decentralized Identity Foundation](#)의 탈중앙 식별자(DID) 프로토콜 사양을 준수합니다.

BNS의 각 이름에는 DID가 연결되어 있습니다. BNS의 DID 형식은 다음과 같습니다:

`did:stack:v0:{address}-{index}`

Where:

- `{address}`는 온체인 공개 키 해시(예: 비트코인 주소)입니다.
- `{index}`는 해당 주소가 생성한 *n*번째 이름을 의미합니다.

예: `personal.id`의 DID는 `did:stack:v0:1dARRtzHPAFRNE7Yup2Md9w18XEQAAtLiV-0`입니다. 이는 `personal.id`가 `1dARRtzHPAFRNE7Yup2Md9w18XEQAAtLiV`에 의해 최초로 생성된 이름이었기 때문입니다.

또 다른 예로, `jude.id`의 DID는

`did:stack:v0:16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg-1`입니다. 이는 주소 `16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg`가 과거에 `abcdefgh123456.id`라는 이름을 먼저 생성했기 때문에, 이 이름이 두 번째 생성된 것임을 나타냅니다.

DID의 목적은 공개 키에 대한 영구적인 식별자를 제공하는 것입니다. 공개 키는 변경될 수 있지만 DID는 변하지 않습니다.

스택스 블록체인은 공개 키 해석에 DID를 사용하는 다른 시스템과의 호환성을 위해 자체 DID 메서드를 구현했습니다. DID가 해석되려면 다음 조건이 모두 충족되어야 합니다:

- 이름이 존재해야 합니다.
- 이름의 존 파일 해시는 올바른 DNS 존 파일의 해시여야 합니다.
- DNS 존 파일이 스택스 노드 데이터에 존재해야 합니다.
- DNS 존 파일에 서명된 JWT를 가리키는 URI 리소스 레코드가 있어야 합니다.
- JWT에 포함된 공개 키는 해당 이름을 소유한 주소의 해시와 일치해야 합니다.

모든 이름이 공개 키로 해석되는 DID를 갖는 것은 아닙니다. 그러나 표준 도구로 생성된 이름은 DID를 가집니다.

RESTful API가 개발 중입니다.

---

## 서브도메인을 위한 DID 인코딩

BNS의 모든 이름과 서브도메인은 DID를 가집니다. 서브도메인의 경우 인코딩이 약간 달라 소프트웨어가 어떤 코드 경로를 사용할지 구분할 수 있습니다.

- 온체인 BNS 이름의 경우 `{address}`는 해당 이름을 소유한 비트코인 주소와 동일합니다. 현재 버전 바이트 0과 5가 모두 지원됩니다(즉, 1 또는 3으로 시작하는 `p2pkh`, `p2sh` 주소).
- 오프체인 BNS 서브도메인의 경우 `{address}`는 단일 개인 키 소유 서브도메인에는 버전 바이트 63, `m-of-n` 개인 키 집합 소유 서브도메인에는 버전 바이트 50을 가집니다. 즉, 서브도메인 DID 주소는 각각 S 또는 M으로 시작합니다.

서브도메인의 DID에 사용되는 `{index}`는 BNS 이름 DID의 `{index}`와는 별개입니다. 동일한 주소가 이름과 서브도메인을 모두 생성했더라도 마찬가지입니다.

예: `abcdefgh123456.id`의 DID는

`did:stack:v0:16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg-0`이며, 이는 해당 주소가 최초로 생성한 이름이었기 때문입니다.

하지만 같은 주소가 첫 번째 서브도메인 `jude.statism.id`도 생성했습니다. 그 DID는 `did:stack:v0:SSXMcDiCZ7yFSQSUj7mWzmDcdwYhq97p2i-0`입니다. 여기서 `SSXMcDiCZ7yFSQSUj7mWzmDcdwYhq97p2i` 주소는 `16EMaNw3pkn3v6f2BgnSSs53zAKH4Q8YJg`와 동일한 공개 키 해시를 인코딩합니다. 두 문자열의 유일한 차이는 첫 번째는 버전 바이트 0으로 `base58check` 인코딩된 것이고, 두 번째는 버전 바이트 63으로 인코딩된 것입니다.

---

# SIPs

## Stacks Improvement Proposals (SIPs)

스택스 개선 제안(SIP)은 스택스 블록체인의 구현을 설명하고 개선을 제안하기 위한 것입니다.

[SIP-000](#)에서 SIP 작성 및 승인 절차가 정의되어 있습니다.

SIP에는 기능이나 표준의 간결한 기술적 명세와 그 근거가 포함되어야 합니다. SIP는 새로운 기능을 제안하고, 시스템 전반의 이슈에 대해 커뮤니티 의견을 수집하며, 설계 결정을 문서화하는 주요 수단이 됩니다.

SIP는 [스택스 Community Governance 조직](#)의 [stacksgov/sips](https://stacksgov/sips) 저장소에 보관됩니다.

스택스 커뮤니티 누구나 SIP를 제출할 수 있습니다.

주간 [커뮤니티 SIP](#) 콜을 캘린더에 추가할 수 있습니다.

SIP 미팅 콜은 녹화되어 [여기](#)에서 확인할 수 있으며, 회의에 대한 더 자세한 정보는 [여기](#)에서 볼 수 있습니다.

---

## 승인된 SIP 목록

- ✓ [SIP 000: Improvement Proposal Process](#)
- ✓ [SIP 001: Burn Election](#)
- ✓ [SIP 002: Clarity, a language for predictable smart contracts](#)
- ✓ [SIP 003: Peer Network](#)
- ✓ [SIP 004: Cryptographic Commitment to Materialized Views](#)
- ✓ [SIP 005: Blocks, Transactions, and Accounts](#)
- ✓ [SIP 006: Clarity Execution Cost Assessment](#)
- ✓ [SIP 007: Stacking Consensus](#)
- ✓ [SIP 008: Clarity Parsing and Analysis Cost Assessment](#)
- ✓ [SIP 009: Standard Trait Definition for Non-Fungible Tokens](#)

- ✓ [SIP 010: Standard Trait Definition for Fungible Tokens](#)
  - ✓ [SIP 012: Burn Height Selection for a Network Upgrade to Introduce New Cost-Limits](#)
  - ✓ [SIP 013: Standard Trait Definition for Semi-Fungible Tokens](#)
  - ✓ [SIP-015: Stacks Upgrade of Proof-of-Transfer and Clarity](#)
  - ✓ [SIP-016: Metadata for Tokens](#)
  - ✓ [SIP-018: Signed Structured Data](#)
  - ✓ [SIP-019: Notifications for Token Metadata Updates](#)
  - ✓ [SIP-020: Bitwise Operations in Clarity](#)
  - ✓ [SIP-022: Emergency Fix to PoX Stacking Increases](#)
  - ✓ [SIP-023: Emergency Fix to Trait Invocation Behavior](#)
  - ✓ [SIP-024: Emergency Fix to Data Validation and Serialization Behavior](#)
- 

## 참여 방법

SIP 프로세스에 참여할 수 있는 방법은 여러 가지가 있습니다:

- [여기](#)에 나열된 **주간 SIP 미팅 콜**에 참여하세요.
- **SIP 에디터**: SIP 에디터는 작성자가 SIP를 올바르게 작성하고 절차를 따르도록 돕습니다. 초안을 Accepted 상태로 발전시켜 심층 검토 준비를 돕습니다. SIP 에디터가 되고 싶다면, 이름과 이메일을 포함한 이슈를 열어 리스트에 추가되도록 요청하세요.
- **CAB(Consideration Advisory Board) 참여**: SIP는 하나 이상의 검토 분야에 속합니다. 전체 목록은 [이 GitHub 디렉토리](#)에 있습니다. 현재는 다양성, 경제, 윤리, 거버넌스, 기술입니다. CAB 멤버는 자신의 전문성을 활용해 SIP를 심층 검토하고, 실행 가능성을 높이기 위한 피드백을 제공합니다.
- **Steering Committee 참여**: Steering Committee는 CAB를 조직하고, 추천된 SIP를 Activation-in-Progress 상태로 발전시킬지, 최종적으로 승인(Ratified) 또는 거부(Rejected)할지를 결정합니다. SIP 활성화 섹션을 통해 스택스 생태계가 해당 SIP를 승인했는지 여부를 판별합니다. 이 위원회에 참여하려면 Stacks Foundation 이사회의 승인이 필요합니다.

## 블록 생성 (Block Production)

블록 생성은 스택스가 내부적으로 어떻게 동작하는지를 이해하는 데 핵심 개념입니다. 이 섹션에서는 스택스 체인이 운영되기 위해 필요한 세 가지 주요 작업을 설명합니다.

- Mining(채굴)

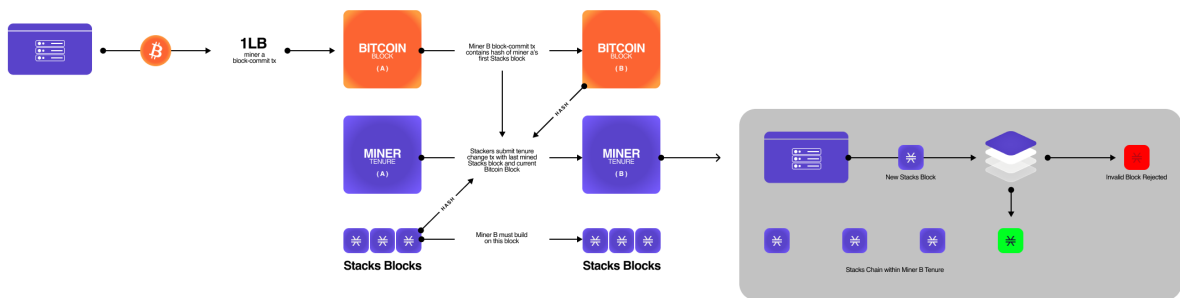
- Signing(서명)
- Stacking(스태킹)

스택스의 블록 생성에는 두 주체가 있습니다: **채굴자(miners)** 와 **스태커(stackers)**. 채굴자는 새로운 블록을 구축·제안하고, 스태커는 해당 블록을 검증하며 sBTC 입·출금에 서명합니다.

스태킹은 스태커가 수행하는 별도의 행위로, 서명을 하기 위한 필수 전제 조건입니다. 다음 세 섹션에서 이 모든 것이 어떻게 맞물려 동작하는지 다룹니다.

Nakamoto 업그레이드 이후 블록 생성의 기술적 세부는 [SIP-021](#)을 꼼꼼히 읽어보세요.

아래 다이어그램은 Nakamoto 규칙하의 블록 생성 과정을 요약합니다. 각 요소는 이어지는 문서에서 자세히 설명합니다.



# Mining(채굴)

이 문서는 채굴 방식에 대한 개념적 가이드입니다. 실제로 자신의 채굴기를 설정하는 절차는 [메인넷](#)과 [테스트넷](#) 가이드를 참고하세요.

## Miner Tenure(채굴자 임기)

이전(Stake Nakamoto 업그레이드 이전)에는 스택스 채굴자가 비트코인 블록과 1:1로 새 스택스 블록을 생성했습니다.

Nakamoto 이후에는 달라졌습니다. **Nakamoto 규칙**에서는 채굴자가 비트코인 블록에 대응하는 **임기(tenure)**를 위해 선출됩니다. 임기 동안 채굴자는 약 10초 간격으로 여러 개의 스택스 블록을 구축·제안하고, 스태커가 이를 승인·부착합니다(다음 섹션 참조).

임기 후보가 되려면 채굴자의 **블록 커밋(block commit)** 이 비트코인 블록에 포함되어야 합니다. 제출 후 커밋을 갱신하려면 비트코인의 **Replace-By-Fee(RBF)** 를 사용할 수 있습니다.

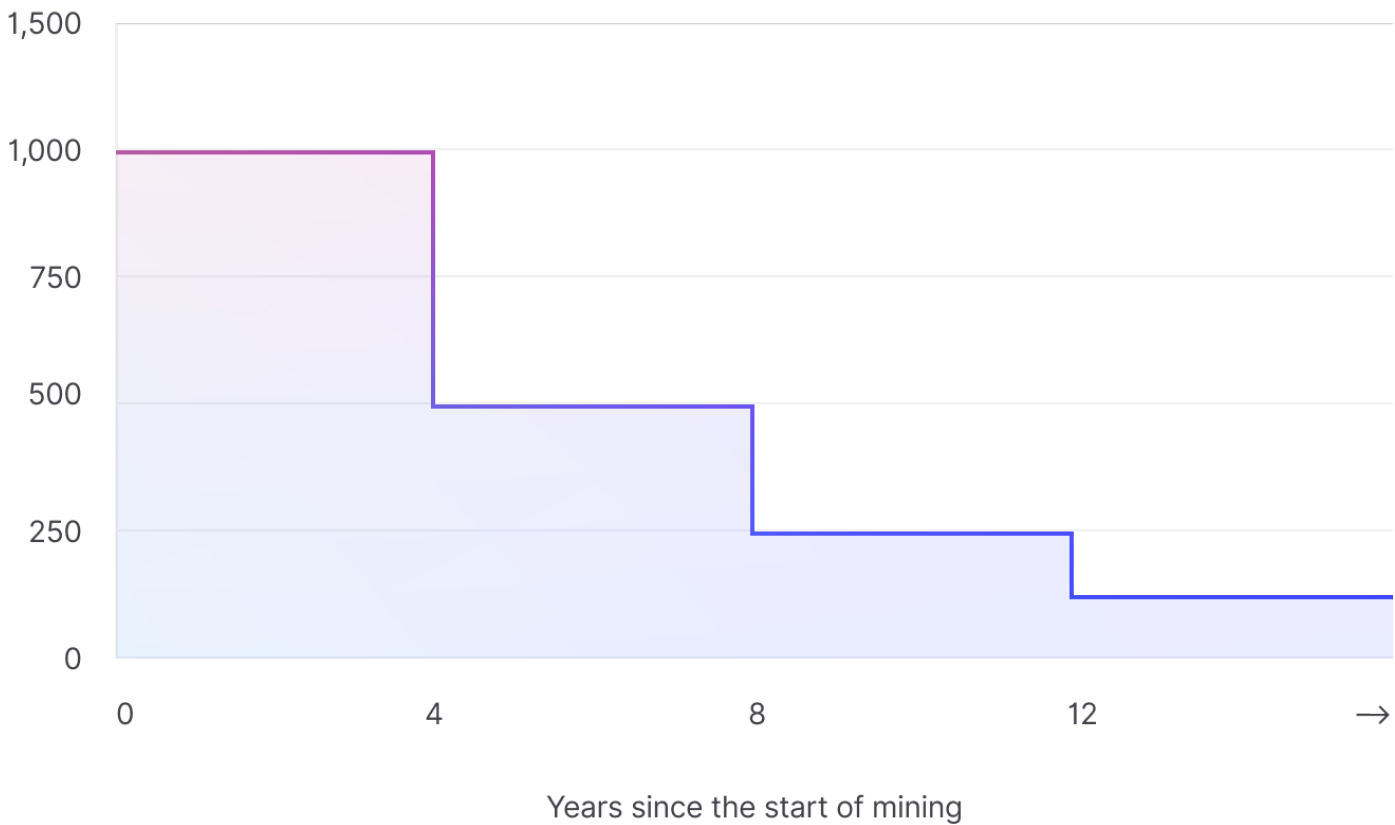
## 코인 기반 보상(Coinbase rewards)

채굴자는 자신이 **획득한 임기**에 대해 코인 기반 보상을 받습니다.

보상량은 다음과 같습니다.

- 첫 4년: 임기당 1000 STX
- 다음 4년: 임기당 500 STX
- 다음 4년: 임기당 250 STX
- 이후 영구: 임기당 125 STX

이 “반감기(halving)”는 비트코인 반감기와 **동기화**됩니다.



## 트랜잭션 수수료

채굴자는 자신이 생성한 블록에 포함된 트랜잭션의 **스택스 수수료**를 받습니다.

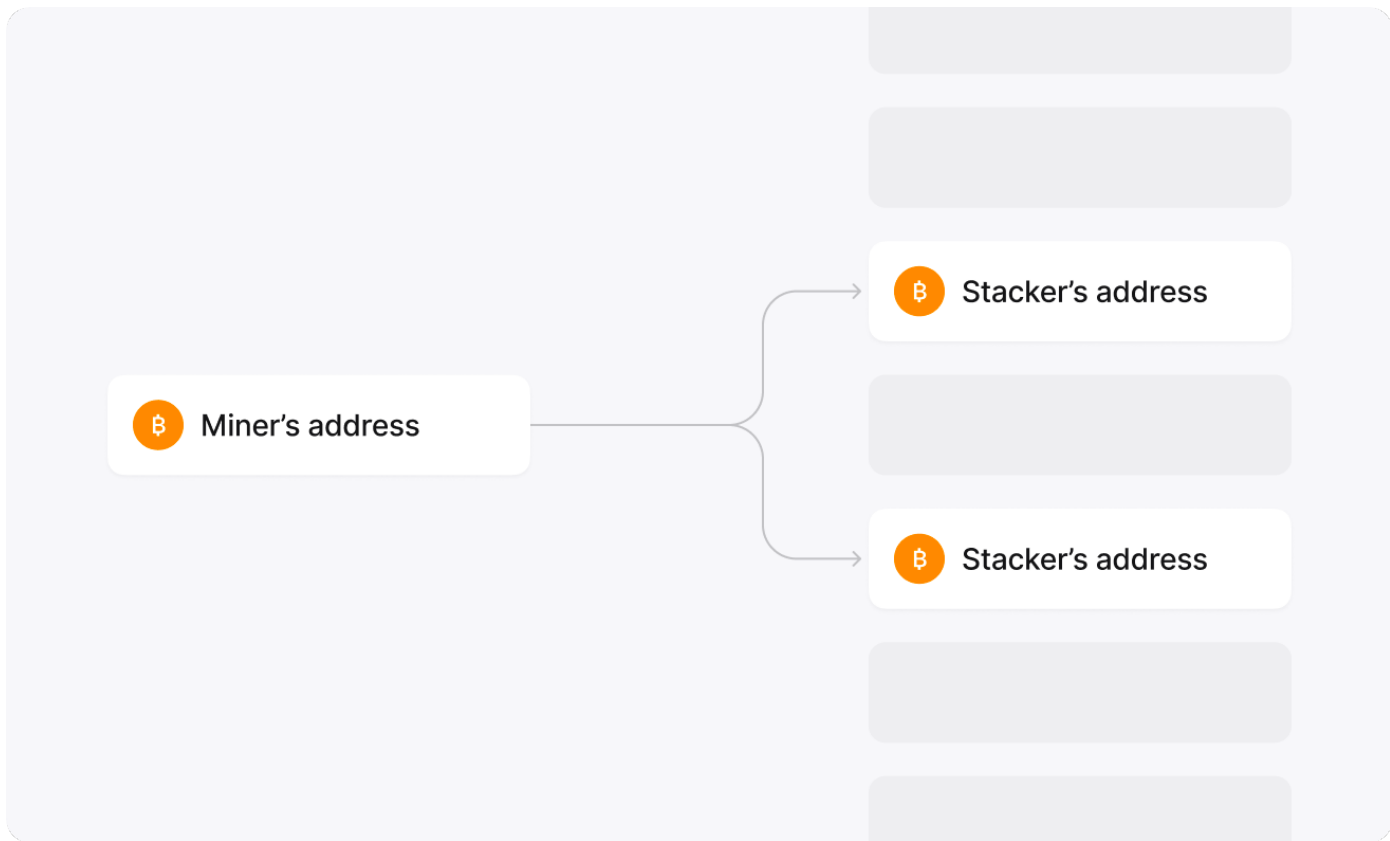
## 보상 성숙(Reward maturity)

블록 보상과 트랜잭션 수수료는 **비트코인 100블록**이 지나야 성숙합니다. 즉, 성공적으로 블록을 채굴한 뒤 **약 24시간** 후에 스택스 계정에서 보상이 확인됩니다.

## Proof-of-Transfer 기반 채굴

채굴자는 매 리더 블록 커밋에서 두 개의 비트코인 주소로 동일 금액의 BTC를 전송(commit)합니다. 이 주소들은 현재 스테킹 참여자(보상 집합)에서 **VRF(검증 가능한 난수함수)**로 선택됩니다. 올바른 두 주소를 계산하려면 스택스 체인을 **모니터링**해야 합니다.

자세한 내용은 Proof of Transfer를 설명한 [SIP-007](#)을 참고하세요.



PoX 채굴은 Proof-of-Burn(PoB)의 변형으로, **소각 주소로 BTC를 보내는 대신**, 스테킹에 참여한 **자격 있는 STX 보유자에게 BTC가 전송**됩니다.

PoX 채굴자는 **자격 있는 STX 보유자에게 BTC를 전송**해야만 새로 발행되는 STX 보상을 받을 수 있습니다.



## Mining Flow



1

### Registration

Miners send consensus data to the network to register.



2

### Commitment

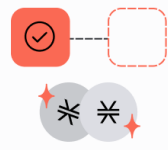
Miners transfer BTC to participate in election.



3

### Election

A VRF chooses one miner to write a new block on the Stacks blockchain.



4

### Assembly

The elected miner writes a stream of microblocks and one anchor block, and collects STX rewards.

채굴자는 **마이닝이 활성화된 스택스 노드**를 실행하여 PoX 메커니즘에 참여합니다. 노드는 다음의 4단계를 통해 올바른 처리와 인센티브를 보장합니다.

- **등록(Registration):** 채굴자는 향후 선출에 참여하기 위해 합의(consensus) 데이터를 네트워크에 전송해 등록합니다.
- **커밋(Commitment):** 등록된 채굴자는 선출에 참여하기 위해 BTC를 전송합니다. 전송된 BTC는 스테킹에 참여하는 STX 보유자 집합에 분배됩니다.
- **선출(Election):** VRF가 새로운 임기의 블록 작성 권한을 가질 채굴자 1인을 선택합니다.
- **조립(Assembly):** 선출된 채굴자는 맴폴에서 트랜잭션을 선택해 새 블록들을 작성하고, 새로 발행된 STX 보상을 수령합니다.

## 다음 블록 채굴 확률

다음 블록을 채굴할 채굴자는 채굴자들이 전송(커밋)하거나 소각한 BTC의 양에 따라 선택됩니다.

채굴자가 다음 블록을 채굴할 확률은 [이 문서](#)에서 설명된 ATC-C(Assumed Total Commitment with Carryforward) MEV 완화 전략을 변형한 방식으로 결정됩니다. 채굴자가 소팅(sortition)에서 승리하여 현재 임기를 부여받을 확률은 현재 소팅에 이르는 블록들에서의 총 블록 커밋 지출을 반영하는 함수로 산출됩니다.

자세한 내용은 [SIP-021의 MEV 섹션](#)을 참고하세요.

프로토콜에서 최소 BTC 커밋을 강제하지는 않지만, 실제로는 **dust(더스트)** 제약이 존재합니다. 즉, 트랜잭션 수수료가 출력값보다 크면 dust로 간주됩니다. dust의 계산 방법은 여러 요인에 따라 달라지며, 일반적으로 출력당 5,500 사토시가 하한선으로 적절하다고 합니다. 스택스 채굴자의 비트코인 트랜잭션에는 PoX를 위해 두 개의 출력이 포함되므로, 블록당 최소 11,000 사토시의 커밋이 권장됩니다.

채굴자가 보낼 BTC 금액을 계산하기 위해서는:

- 다음 날(100 블록 후)의 BTC/STX 가격을 추정해야 합니다.
- 모든 채굴자가 커밋할 BTC 총량을 추정해야 합니다.

스태커(Stacker)는 새로운 블록을 검증·추가하고, 채굴자의 임기를 관리하는 역할을 담당합니다. 다음 섹션에서 이 과정이 비트코인 파이널리티로 이어지는 방법을 설명합니다.

---

## 실제 스택스 채굴

Signal21의 채굴 대시보드를 보면 스택스 네트워크에서의 채굴 데이터(블록당 사용된 BTC, 블록당 획득한 STX, 전체 채굴자 수, 특정 블록에서의 채굴자 수 등)를 확인할 수 있습니다.

많은 사람들이 스택스의 채굴자 수가 적다는 점에 주목합니다. 맥락이 없으면 이는 의문을 불러일으킬 수 있습니다. 하지만 채굴이 어떻게 동작하는지를 이해하면, 이것이 탈중앙성을 저해하지 않는 이유를 알 수 있습니다.

스택스의 채굴자는 L2 시스템의 시퀀서와 유사하게 **새로운 블록을 구성·제안**만 담당합니다. 블록을 체인에 추가하는 것은 스타커의 역할입니다. 그러나 대부분의 Ethereum L2가 단일 중앙화된 시퀀서로 운영되는 것과 달리, 스택스는 누구나 참여할 수 있는 오픈 멤버십 구조로 **항상 최소 4~5명의 채굴자**가 활동합니다.

스택스의 블록 생성 과정에는 채굴자와 스타커 두 주체가 있으며, 이는 상호 보완적인 관계를 이룹니다. 스타커는 채굴자가 체인에 미칠 수 있는 잠재적 파괴력을 크게 줄여줍니다.

- 채굴자는 체인을 재구성(reorg)할 수 없습니다. 최악의 경우 일부 트랜잭션을 누락할 수 있지만, 이를 해결하려면 사용자가 직접 채굴기를 실행하면 됩니다.
- 네트워크에 더 많은 채굴자가 존재하면, 채굴자들이 BTC를 더 많이 수수료로 지불하게 되어 스타커가 받는 보상이 줄어듭니다.

---

## 채굴자가 많아지면 경쟁이 치열해져 더 많은 보상이 있지 않을까?

실제로는 그렇지 않습니다. 채굴자는 경제적으로 합리적으로 행동하며, 무한정 BTC를 투입할 수 없습니다.

채굴자는 코인베이스(1,000 STX)와 수수료 보상을 얻기 위해 PoX 커밋 + 비트코인 수수료를 지불합니다. 채굴자가 많아지면, 각자 승리 확률이 낮아져 지불액도 줄어듭니다. 채굴자가 손해를 보는 수준까지 계속 지불할 수는 없기 때문에, 채굴자가 지출할 수 있는 BTC에는 상한이 있습니다.

지불액이 줄어들수록 비트코인 수수료가 비용에서 차지하는 비중이 커지고, 이는 승리 확률을 더욱 낮춥니다.

예를 들어:

- STX 가격이 1,000 사토시라고 가정합니다.
  - 모든 채굴자가 논리적으로 행동하고 스택스 수수료를 무시한다면, 총 지출은  $1,000 \text{ STX 코인베이스} \times 1,000 \text{ 사토시} = 1,000,000 \text{ 사토시}$  미만일 것입니다.
  - 채굴자가 5명이라면, 약 10,000 사토시(트랜잭션당 2,000 사토시)가 비트코인 수수료로, 990,000 사토시가 PoX에 사용될 수 있습니다.
  - 채굴자가 100명이라면, 약 200,000 사토시가 비트코인 수수료로, 800,000 사토시가 PoX에 사용될 수 있습니다.
- 

이 설계는 자연스러운 경제적 균형을 만듭니다:

1. 충분한 채굴자가 참여해 블록이 안정적으로 생성됨
2. 스택커가 최적의 BTC 보상을 수령함
3. 과도한 경쟁 없이 네트워크의 검열 저항성 유지

이는 의도된 설계입니다. 스택커가 보완적 보안 보증자로서 PoX를 통해 BTC 보상을 받음으로써, 스택스는 채굴자 수가 과도하지 않아도 보안을 확보할 수 있습니다.

다른 체인에서는 채굴자만이 체인의 정당성을 결정하지만, 스택스는 두 주체 구조를 통해 더 강력한 보장을 제공합니다:

- 채굴자는 유효하지 않은 트랜잭션이나 블록을 강제로 추가할 수 없습니다(스택커가 서명하지 않고, 노드도 이를 거부함).
- 채굴자는 단독으로 체인을 재구성할 수 없습니다(체인 파이널리티는 스택커가 통제).
- 블록이 승인되기 전 스택커의 70% 서명 임계치가 요구되어 광범위한 합의가 보장됩니다.

이처럼 채굴자와 스택커의 역할 분리는, 채굴자 수가 적더라도 스택스를 독자적으로 안전하게 만드는 핵심입니다.

---

## 마이크로블록(Microblocks)은 어떻게 되었나?

마이크로블록은 이전 버전의 스택스에서 존재하던 기능으로, 트랜잭션 처리량 향상을 위해 고안되었지만 Nakamoto 이전 구조에서는 제대로 동작하지 않았습니다.

Nakamoto에서는 마이크로블록 대신, 약 10초마다 스택스 블록을 빠르게 생성하는 구조를 사용합니다.

---

# Signing(서명)

스태커는 과거 채굴자가 담당했던 역할 일부를 맡으며 Nakamoto 시스템에서 핵심적 역할을 합니다. 이전에는 채굴자가 블록의 내용을 결정하고, 블록을 체인에 포함할지 여부도 결정했습니다. 이제는 역할이 분리됩니다:

- **채굴자:** 블록의 내용을 결정
- **스태커:** 블록을 체인에 포함할지 여부를 결정

Nakamoto 변경의 주요 복잡성은 이 두 역할을 분리하면서도, 마이닝과 스테킹 모두 오픈 멤버십으로 유지되도록 보장하는 것입니다. **중요한 점은 여전히 누구나 채굴자, 누구나 스테커가 될 수 있다는 점입니다.**

스태커는 채굴자의 블록이 체인에 추가되기 전 반드시 이를 검증하고 승인해야 합니다. 이를 위해 스테커는 먼저 정식 체인 팁(canonical chain tip)에 합의한 뒤, 그 위에 블록을 적용(및 롤백)하여 유효성을 판단합니다. 유효성이 확인되면 블록에 공동 서명하고, 이를 네트워크에 복제합니다. 이 단계에서야 비로소 노드들이 블록을 체인에 추가합니다.

이 방식은 포크 발생을 방지합니다. 채굴자가 오래된 팁 위에 블록을 만들면 스테커가 서명을 거부합니다. 스테커가 정식 팁에 합의하지 못하면 블록은 추가되지 않습니다. 이 경우 체인이 정지할 위험이 있지만, 충분히 다양하고 분산된 스테커 집단이 항상 온라인 상태를 유지하도록 PoX 보상으로 유도함으로써 위험이 완화됩니다.

## 스태커 서명 (Stacker Signing)

[Hiro 블록 탐색기](#)에서 모든 [활성 서명자(active signers)] 목록을 확인할 수 있습니다.

스테킹이 어떻게 동작하는지는 [Stacking](#) 섹션에서, sBTC 서명은 [sBTC](#) 섹션에서 다룹니다. 여기서는 스택스 블록 생성과 관련된 서명 과정을 설명합니다.

스태커가 정식 체인 팁(canonical chain tip)에 합의하고 블록을 추가하기로 동의하는 방식은 PoX에 연결되어 있습니다. 각 보상 주기마다 스테커는 하나 이상의 보상 슬롯을 확보하며, 보상 주기에는 최대 4,000개의 보상 슬롯이 존재합니다. 스테커는 블록에 대해 가중 임계치 서명을 생성함으로써 블록 수용 여부를 투표합니다. 이 서명은 PoX에 잠긴 STX 총량의 상당 비율(임계치)을 반영해야 하며, 각 스테커의 서명 비중은 자신이 보유한 잠긴 STX의 비율에 비례합니다.

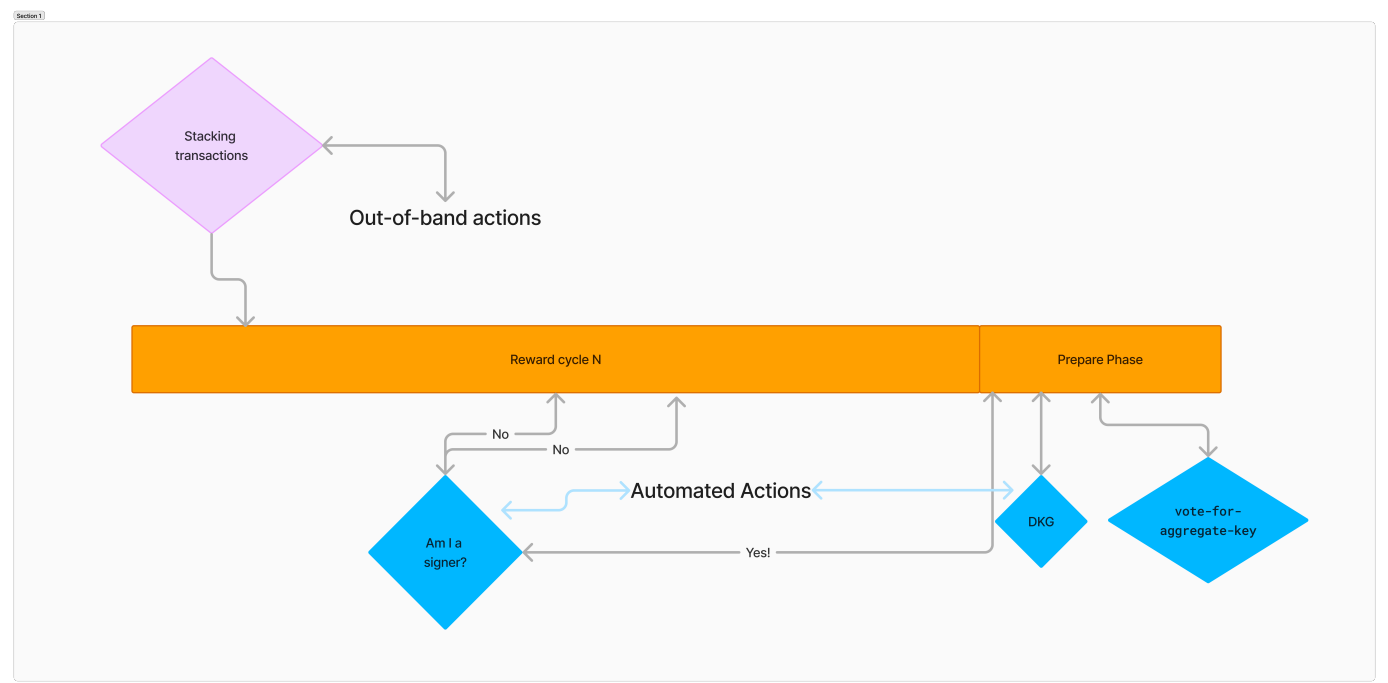
가중 임계치 서명은 [FROST 프로토콜](#)의 변형을 통해 생성된 슈노르(Schnorr) 서명입니다. 각 스테커는 서명 키 쌍을 생성하고, 분산 서명 프로토콜을 통해 노드가 검증할 수 있는 집계 공개 키를 공동으로 생성합니다. 이 프로토콜은 스테커가 확보한 보상 슬롯 수에 비례하여 집계 개인 키의 지분을 할당합니다. 스테커 중 누구도 집계 개인 키 전체를 알지 못하며, 각 스테커는 자신의 지분 키를 사용해 서명 조각을 계산하고, 이 조각들이 결합되어 단일 슈노르 서명이 만들어집니다.

채굴자가 블록을 생성하면, 스테커들은 분산 서명 프로토콜을 실행하여 해당 블록에 대해 단일 슈노르 서명을 공동으로 생성합니다. 이 프로토콜은 최소 X%의 보상 슬롯이 집계 서명에 반영되어야만 성공합니다.

Nakamoto에서는 서명 임계치가 **70%**로 설정되어 있으며, 즉 잠긴 STX의 70% 이상이 블록에 서명해야 스택스 블록체인에 추가될 수 있습니다.

Nakamoto는 WSTS 프로토콜과 FIRE 확장을 사용합니다. 이는 스택커 수에 따라 CPU 및 네트워크 대역폭 복잡도가 증가하는 분산 키 생성 및 서명 알고리즘 쌍을 허용합니다. FIRE 확장은 WSTS가 비잔틴 스택커를 견딜 수 있도록 합니다.

아래는 서명과 스택킹 간의 관계를 보여주는 다이어그램입니다.



## 새로운 블록의 검증 및 추가 (Validating and Appending New Blocks)

채굴자가 새로운 임기에 선출되면, 멤버의 트랜잭션을 모아 새 블록을 구축하기 시작합니다. 이후 채굴자는 블록을 스택커에게 보내 승인을 요청합니다. 스택커는 최소 70%의 정족수(quorum)로 블록을 승인해야 해당 블록이 체인에 추가됩니다.

스택커는 다음 조건에 따라 블록을 승인합니다:

- 블록이 올바르게 구성되었는가?
  - 올바른 버전과 메인넷/테스트넷 플래그를 포함해야 함
  - 헤더에 올바른 수의 이전 스택스 블록이 포함되어 있어야 함
  - 헤더에 현재 임기를 선출한 소팅에서 사용된 비트코인 총 지출이 올바르게 포함되어 있어야 함
  - 헤더에 임기 블록 커밋 트랜잭션을 포함한 비트코인 블록 해시가 일치해야 함\*
  - 헤더에 직전 부모 블록의 올바른 parent block ID가 포함되어 있어야 함\*
  - 트랜잭션 머클 트리 루트가 트랜잭션과 일관성이 있어야 함
  - 모든 트랜잭션이 적용된 후 상태 루트 해시가 MARF tip root 해시와 일치해야 함
  - 블록 헤더에 채굴자의 올바른 ECDSA 서명이 있어야 함

- 블록 헤더에 스택어 집합의 올바른 WSTS 슈노르 서명이 있어야 함
- 마지막 유효 소팅 이후부터(해당 임기의 블록 커밋이 포함된 비트코인 블록 직전까지)의 모든 비트코인 트랜잭션이 스택스 체인 상태에 적용되어야 함\*
- 임기 시작 블록의 경우:
  - 첫 번째 트랜잭션은 `TenureChange`여야 함
  - `TenureChange` 이후 첫 번째 트랜잭션은 `Coinbase`여야 함

\* 표시된 조건들은 스택스가 **비트코인 파이널리티**를 보장하는 핵심입니다. 이 조건을 준수함으로써 채굴자는 올바른 체인 팁 위에 블록을 구축해야만 블록을 추가할 수 있으며, 이로써 블록 이력이 비트코인에 앵커링됩니다.

스태커는 이러한 규칙을 검증함으로써 비트코인 파이널리티를 보장합니다. 이에 대해서는 다음 섹션에서 더 자세히 다룹니다.

## 채굴자 임기 변경 수행 (Conducting Miner Tenure Changes)

블록 생성에서 또 다른 주요 서명 책임은 임기 변경 트랜잭션을 수행하는 것입니다. 앞서 채굴 섹션에서 설명했듯, 채굴자는 비트코인 체인에 `block-commit` 트랜잭션을 제출하여 채굴을 시작합니다. 만약 선출된다면, 스택커가 이를 감지하고 `tenure-change` 트랜잭션을 생성합니다.

이 임기 변경 트랜잭션에는 다음 정보가 포함됩니다:

| Name                           | Description                                                                                                                               | Representation      |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| tenure consensus hash          | 이번 임기의 합의 해시. 해당 채굴자가 선출된 소팅 (sortition)에 해당합니다. 만약 이 채굴자의 임기가 이후 소팅에서도 연장된다면, consensus hash 값은 최초 당첨된 block-commit이 채굴된 소팅과 동일하게 유지됩니다. | 20 bytes            |
| previous tenure consensus hash | 이전 임기의 합의 해시. 이전에 당첨된 block-commit의 소팅에 해당합니다.                                                                                            | 20 bytes            |
| burn view consensus hash       | 기반 번체인(burnchain)의 현재 합의 해시. 마지막으로 관찰된 소팅에 해당합니다.                                                                                         | 20 bytes            |
| previous tenure end            | 이전 임기의 마지막 스택스 블록의 인덱스 블록 해시.                                                                                                             | 32 bytes            |
| previous tenure blocks         | 마지막 소팅 연계 임기 이후 생성된 블록 수.                                                                                                                 | 4 bytes, big-endian |
| cause                          | 임기 변경의 원인을 나타내는 플래그.<br>- 0x00: 소팅이 발생했으며, 새 채굴자가 블록 생성을 시작해야 함.<br>- 0x01: 현재 채굴자가 블록 생성을 계속해야 함. 이 경우 현재 채굴자의 임기 실행 예산이 리셋됨.            | 1 byte              |
| pubkey hash                    | 현재 임기의 ECDSA 공개 키 해시.                                                                                                                     | 20 bytes            |

이 임기 변경 트랜잭션은 새로 선출된 채굴자에게 전송되며, 채굴자는 이를 첫 블록의 **첫 번째 트랜잭션**으로 반드시 포함해야 합니다. 그렇지 않으면 스택커가 해당 블록을 승인하지 않습니다.

이 과정은 새 채굴자가 선출될 때마다 반복됩니다.

자세한 내부 동작은 [SIP-021](#)을 참고하세요.

# 비트코인 파이널리티 (Bitcoin Finality)

**100% 비트코인 파이널리티** 개념은 스택스 설계의 핵심입니다. 이것이 스택스를 진정한 비트코인 L2로 만들며, 비트코인이 지닌 모든 보안을 활용할 수 있게 합니다.

우선 100% 비트코인 파이널리티의 의미를 정의한 후, 스택스가 이를 어떻게 달성하는지 살펴보겠습니다.

**파이널리티(finality)** 는 트랜잭션이 되돌릴 수 없게 되는 지점을 의미합니다. 일단 블록체인이 파이널리티에 도달하면, 원장을 변경하는 것은 사실상 불가능하며, 경제적·연산적으로 엄청난 비용이 필요합니다.

스택스 블록이 100% 비트코인 파이널리티를 가진다는 것은, 그것을 되돌리는 것이 비트코인 트랜잭션을 되돌리는 것만큼 어렵다는 뜻입니다.

그렇다면 스택스는 이를 어떻게 달성할까요?

앞서 설명했듯, 채굴자는 비트코인 블록 하나에 해당하는 임기 동안 스택스 블록을 생성합니다. 채굴자의 블록 커밋 트랜잭션에는 이제 **인덱스 블록 해시**를 포함해야 합니다.

인덱스 블록 해시는 해당 임기의 첫 번째 스택스 블록 해시로, 이전에 수용된 모든 비트코인 트랜잭션의 합의 해시와 블록 자체의 SHA512/256 해시를 포함합니다.

이를 통해 스택스 체인 이력이 비트코인에 앵커링되며, 이전 임기 시작 시점과 모든 인과적으로 연결된 비트코인 상태가 포함됩니다. 이 방식은 비트코인 파이널리티를 보장하고, 포크 방지는 스택커가 담당하도록 하여 채굴자 연결 문제를 해결하며, 노드가 비트코인 리오그 시 어떤 스택스 블록이 영향을 받았는지 식별하고 복구할 수 있게 합니다.

즉, 매 비트코인 블록 N+1 시점에, 임기 N 시작 시의 스택스 체인 상태가 비트코인에 기록됩니다. 따라서 과거 모든 스택커의 키가 탈취되더라도, 비트코인 블록 N+1을 되돌리지 않는 이상 스택스 임기 N의 이력을 되돌릴 수 없습니다.

따라서 스택스 트랜잭션은 자신이 속한 임기가 종료되면(즉, 해당 비트코인 블록 N+1에서) 비트코인 파이널리티를 가집니다. 예를 들어, 내가 스택스 트랜잭션을 전송해 채굴자에게 확정되면, 해당 임기가 끝나는 시점에 그 트랜잭션은 비트코인에 기록되며, 이후 모든 채굴자는 그 체인 팁 위에서만 블록을 생성해야 하므로 이를 되돌리는 것은 해당 비트코인 트랜잭션을 되돌리는 것만큼 어렵습니다.

---

## Nakamoto 트랜잭션과 비트코인 리오그

만약 Nakamoto 트랜잭션이 비트코인 파이널리티를 따른다면, 비트코인 포크가 발생하면 어떻게 될까요?

이를 설명하기 위해 스택스 트랜잭션을 두 가지로 나눌 수 있습니다:

- **비트코인 의존 트랜잭션(Bitcoin-reliant):** 비트코인 상태를 읽는 트랜잭션입니다. 비트코인 포크가 발생하면 이들은 변경됩니다. 예: L1에서 L2로 BTC를 전송한 경우, 비트코인 파이널리티를 기다려야 해당 BTC를 L2에서 사용할 수 있습니다.

- **내부 트랜잭션(Internal)**: 비트코인 상태에 의존하지 않는 트랜잭션입니다. 비트코인 포크가 발생해도 이들은 변경되지 않으며, 스택커가 동일한 순서로 다시 처리하도록 보장할 수 있습니다.

핵심은 다음과 같습니다:

Nakamoto 체제에서 스택스 트랜잭션은 비트코인 포크로 인해 중대한 리오그가 발생하지 않습니다. 드물게 발생하더라도, 리오그된 스택스 트랜잭션은 리오그된 비트코인 트랜잭션과 동일하게 동작합니다. 따라서 장차 내부 트랜잭션은 비트코인 포크의 영향을 전혀 받지 않을 수도 있습니다.

더 알고 싶다면 [비트코인 리오그 문서](#)를 참고하세요.

---

## 비트코인 리오그 (Bitcoin Reorgs)

Nakamoto 체제에서는 비트코인 포크로 인해 스택스 트랜잭션이 심각하게 리오그되는 일은 없습니다. 리오그 자체가 드물고, 리오그된 트랜잭션은 단순히 리오그된 비트코인 트랜잭션과 같은 방식으로 처리됩니다.

핵심은 파이널리티 이해입니다. **Nakamoto 이후 스택스 체인은 스스로 포크하지 않습니다.** 설계상 포크를 허용하지 않으며, 설령 스택커의 31% 이상이 포크를 원치 않아도 포크는 불가능합니다. 발생하더라도 단일 임기 내에서만 가능할 것입니다.

즉, Nakamoto 이후 스택스가 포크하는 유일한 경우는 비트코인 포크가 원인일 때입니다.

Nakamoto에서는 채굴자가 단일 블록을 만드는 권리가 아니라 다수 블록을 만드는 임기(tenure)를 획득합니다. 이 임기 동안 생성되는 모든 스택스 블록은 **스태커 70% 이상**의 서명이 있어야 블록체인에 포함됩니다. 스택커는 비트코인 블록체인을 모니터링하며, 최신 소팅에서 선출된 채굴자의 블록만 서명합니다.

만약 비트코인이 리오그되면, 스택커는 새로운 최적의 비트코인 포크와 직전 공통 소팅 지점으로 되돌아가 해당 임기부터 다시 서명을 시작합니다. 스택커 70%가 합의하면 새로운 단일 최적의 스택스 블록체인이 형성됩니다.

리오그된 임기에서 확정되었던 트랜잭션은 어떻게 될까요? 아무 일도 일어나지 않습니다. 마치 해당 임기가 존재하지 않았던 것처럼 그대로 메모에 남습니다.

**이는 비트코인 트랜잭션 리오그와 1:1 대응됩니다.** 비트코인에서 체인 팁 근처의 트랜잭션은 파이널리티를 갖지 않듯, 스택스에서 임기 팁 근처의 트랜잭션은 파이널리티를 갖지 않습니다.

---

## 트랜잭션 재실행 (Replaying Transactions)

스택스 블록이 포함되려면 서명자 70% 이상이 서명해야 하므로, 이들은 비트코인 포크 전후 체인 상태를 모두 알고 있습니다.

다만 주의할 점이 있습니다. 트랜잭션이 리오그된 비트코인 블록체인과 연관된 경우(예: peg-in)라면, 이는 새 체인에서 무효가 됩니다. 즉, **Taint 분석**을 통해 어떤 트랜잭션이 무효화되었는지, 또 그와 상호작용한 다른 트랜잭션도 무효인지 판단해야 합니다. 이는 연쇄 효과를 유발할 수 있습니다.

Taint 분석과 재실행(enforcement)은 미래에 추가될 수 있습니다. 현재 버전에서는 스택스 블록체인이 비트코인 블록체인에 명시적으로 연결되어, 언제나 하나의 최적 스택스 포크만 존재하도록 합니다. 이는 비트코인 블록체인 동작과 동일하지만, 임기 단위로 확장된 것입니다.

# 스태킹 (Stacking)

## 소개

스태킹은 STX 보유자가 일정 기간 토큰을 **잠가(lock)** 두고 합의에 필수적인 **서명자(signers)**로 참여하는 대가로 **비트코인(BTC)** 보상을 받는 메커니즘입니다. 스택스에서 서명자 개념이 익숙하지 않다면 먼저 [Stackers and Signing](#) 섹션을 참고하세요.

이 문서는 스테킹이 **어떻게 작동하는지**에 대한 **개념적 개요**입니다. 실제로 스택하는 방법은 [스태킹 가이드](#)와, 컨트랙트 동작을 깊게 파는 [컨트랙트 해설](#)을 참고하세요.

pox-4.clar 가 스테킹 컨트랙트입니다. 상태 변경, 솔로 스테킹, 풀 스테킹 등 PoX 활용을 실험하려면 배포된 컨트랙트에서 모든 함수를 확인할 수 있습니다:

- 테스트넷: <https://explorer.hiro.so/txid/0xfba7f786fae1953fa56f4e56aeac053575fd48bf72360523366d739e96613da3?chain=testnet>
- 메인넷: <https://explorer.hiro.so/txid/0xc6d6e6ec82cabb2d7a9f4b85fcc298778d01186cabaee01685537aca390cdb46?chain=mainnet>

## 스태킹 vs 스테이킹

스택스의 스테킹은 개념적으로 PoS 체인의 **스테이킹**과 비슷해 보이지만, 스택스는 **PoS 네트워크가 아니며** 몇 가지 핵심 차이가 있습니다.

### 1) 소각체인 토큰으로 보상 (Yield in burnchain token)

스테이킹은 **잠근 토큰과 같은 토큰**으로 보상을 받습니다. 스테킹은 **STX를 잠그고, 비트코인(BTC)**으로 보상을 받습니다. PoX에서 수익은 **외부(채굴자의 BTC 입금)**에서 오며, PoS의 수익은 **프로토콜 발행 일정**에서 옵니다.

이 때문에 PoS는 네트워크 사용량과 설계에 따라 **발행률 조정**(예: 이더리움의 EIP-1559 및 소각/발행 다 이내믹)이 필요하지만, 스테킹의 **보상 산정**은 **발행률과 분리**되어 있습니다. (참고: 스택스도 별도의 STX 발행 스케줄은 있으나, **스태킹 보상과는 독립**입니다.)

스태커의 BTC 보상은 **채굴자가 커밋한 BTC 총량**과 **네트워크에 잠긴 STX 규모**의 조합으로 결정됩니다.

## 2) 슬래싱 없음 (No slashing)

스태커는 합의에 핵심인 **서명자 역할**을 수행하지만, PoX에는 **슬래싱이 없습니다**. 대신 서명 의무를 다하지 않으면 **잠금 해제 지연/보상 수령 실패**가 발생할 수 있습니다.

스태킹은 PoX 메커니즘의 **내장 동작**이며, 모든 스택스 채굴자가 이를 실행합니다.

 스타킹 기능은 Clarity **스마트 컨트랙트**로 구현되어 있습니다. 자세한 내용은 [컨트랙트 문서](#)를 참고하세요.

## STX 잠금/해제

여기서 말하는 **잠금(lock)**은 **토큰 이동이 아님**에 유의하세요. 잠금은 **비수탁적**이며, STX는 **사용자 지갑에 그대로** 있습니다. 스타킹 트랜잭션이 체결되면 **프로토콜 레벨에서만** 이동 불가가 되며, **잠금 기간 종료 시 자동으로 사용 가능** 상태가 됩니다(별도의 “잠금 해제 트랜잭션”은 없음).

## 스타킹 플로우

스타킹 절차는 아래와 같습니다:

1. 다가오는 리워드 사이클 정보를 API로 조회
2. 해당 스택스 계정의 **참여 자격** 확인
3. **BTC 보상 주소와 잠금 기간** 확정
4. 트랜잭션 브로드캐스트 → STX 잠금 (**다음 리워드 사이클의 prepare phase** 시작 전, 즉 진행 중인 리워드 phase의 **마지막 100 비트코인 블록** 이전에 완료)
5. 리워드 사이클 동안 설정된 BTC 주소로 **보상 전송**
6. 잠금 기간 중 **해제 시점/보상 등 상태 조회**
7. 잠금 기간 종료 시 **자동 해제**(사용 가능)
8. 과거 사이클의 **보상 이력 표시**

리워드 사이클의 목표 길이는 약 **2주(2100 BTC 블록)**입니다. 비트코인 네트워크의 확인 시간 변동에 따라 달라질 수 있습니다.

## 위임 스타킹(Delegation) 플로우

스타킹 방식은 크게 **솔로 스타킹**과 **위임 스타킹**이 있습니다.

솔로 스타킹은 위 플로우를 그대로 따르며, **직접 서명자**를 운영합니다. 솔로 스타킹에는 **최소 필요 STX 수량**이 있으며, 이는 [PoX API의 min\\_threshold\\_ustx](#)로 확인할 수 있습니다.

위임의 경우:

- 토큰 보유자는 **특정 위임자(delegateor)**에게 본인 계정에 대해 **스테킹 권한(최대 한도 포함)**을 부여
- 보유자는 **보상 BTC 주소/권한 만료 블록 높이** 등을 제한적으로 지정 가능
- 위임자는 여러 계정의 STX를 모아 **최소 임계치 달성** 후 다음 리워드 사이클 참여 확정
- 일부 위임 관계에서는 **보유자가 직접 보상 수령**하도록 설정 가능
- 위임 종료는 **만료 규칙** 또는 **권한 철회**로 처리

## 토큰 보유자 참여 조건

모든 STX 보유자가 자동으로 보상을 받는 것은 아닙니다. 다음을 충족해야 합니다.

- 리워드 사이클 시작 **이전**에 참여 커밋
- **최소 STX 수량**(또는 풀을 통한 합산)으로 **리워드 슬롯 확보**
- 지정 기간 동안 STX **잠금**
- 보상 수령을 위한 **지원되는 BTC 주소** 제공
- (서명자를 운영한다면) **서명자 소프트웨어 유지**

스테커 **최소 필요량 산출**은 다음 다이어그램을 참고하세요.

스택스 생태계에는 다양한 **풀/도구**가 있으며, [스택스 공식 페이지](#)에서 옵션을 확인할 수 있습니다.

## PoX 합의에서의 스테킹

스테킹은 PoX의 **내장 기능**이며, 블록체인 상 일련의 동작으로 일어납니다. 전체 구현은 [SIP-007](#)에 정의되어 있습니다.

*참고: SIP-007은 Nakamoto 이전 스테킹을 설명합니다. 기본 기능은 유사하나, Nakamoto 이후 스테커는 서명자 역할(블록 서명)을 추가로 수행합니다. 자세한 사항은 [SIP-021](#) 참고.*

스테킹은 약 **2100 BTC 블록(≈2주)**의 리워드 사이클로 진행되며, **준비(prepare) 100 블록 + 리워드** 두 단계로 나뉩니다.

Nakamoto 이후 스택스는 **포크하지 않으므로**, **PoX 앵커 블록**(다음 사이클의 스테커를 정하는 블록)은 **준비 단계 100 BTC 블록 전에** 항상 결정됩니다(마지막 tenure-start 블록).

PoX 앵커 블록은 **다음 서명자 집합**을 식별합니다. 이들은 100 BTC 블록 동안 **블록 서명을 위한 DKG**(분산 키 생성)를 완료해야 합니다. PoX 컨트랙트는 스테킹/위임 시 **블록 서명 키 등록**을 요구하여, 네트워크가 서명을 검증할 수 있게 합니다.

이 과정은 [서명자 실행](#) 및 [스테킹 오퍼레이션 수행](#)으로 처리합니다.

## 스테킹과 서명(Signing)

두 동작은 구분되지만 **상호** 필요합니다. 서명자는 **STX**를 스테킹해야 하며, 스테킹은 서명자 운영과 결합됩니다. 다만 **솔로/위임**에 따라 방식이 다릅니다.

## 솔로 스테킹

### 1. 직접 서명자 운영

가이드(How to Run a Signer)를 따라 자체 서명자를 운영할 수 있습니다. 약간의 인프라/운영 지식이 필요합니다.

### 2. 다른 서명자와 협업

자체 운영이 어려우면, 타 서명자와 협업하여 **해당 서명을 스테킹 트랜잭션에 포함**할 수 있습니다. 방법은 [Stack STX 가이드](#)에 안내되어 있습니다.

## 위임 스테킹

풀에 위임하는 경우 서명자 운영은 **필요 없습니다**. 실제 스테킹 트랜잭션을 수행하는 **풀 운영자**가 서명자를 운영합니다. 풀 운영자를 위한 전용 [가이드](#)도 있습니다.

## 스테킹 방법과 위치

**솔로, 풀 참여, 거래소 이용, 리퀴드 스테킹** 등 여러 옵션이 있습니다.

옵션 비교는 [스테킹 페이지](#)에서 확인하세요.

실행 방법은 [Stack STX 가이드](#)를 참고하세요.

스테킹 데이터/통계는 다음 도구에서 확인할 수 있습니다:

- [Signal21](#)
- [Stacking Tracker](#)
- [StakingRewards.com](#)
- [Stacking Calendar](#)

# 트랜잭션 (Transactions)

트랜잭션은 스택스 체인의 핵심 구성 요소이며, 사용자가 네트워크와 상호작용하는 주요 수단입니다.

이 섹션에서는 트랜잭션이 어떻게 동작하는지와 함께, **Post Condition**이라는 추가 보안 기능을 소개합니다. 이는 클라이언트 개발자가 악의적인 스마트 컨트랙트와의 상호작용을 방지하기 위해 특정 조건을 강제할 수 있도록 합니다.

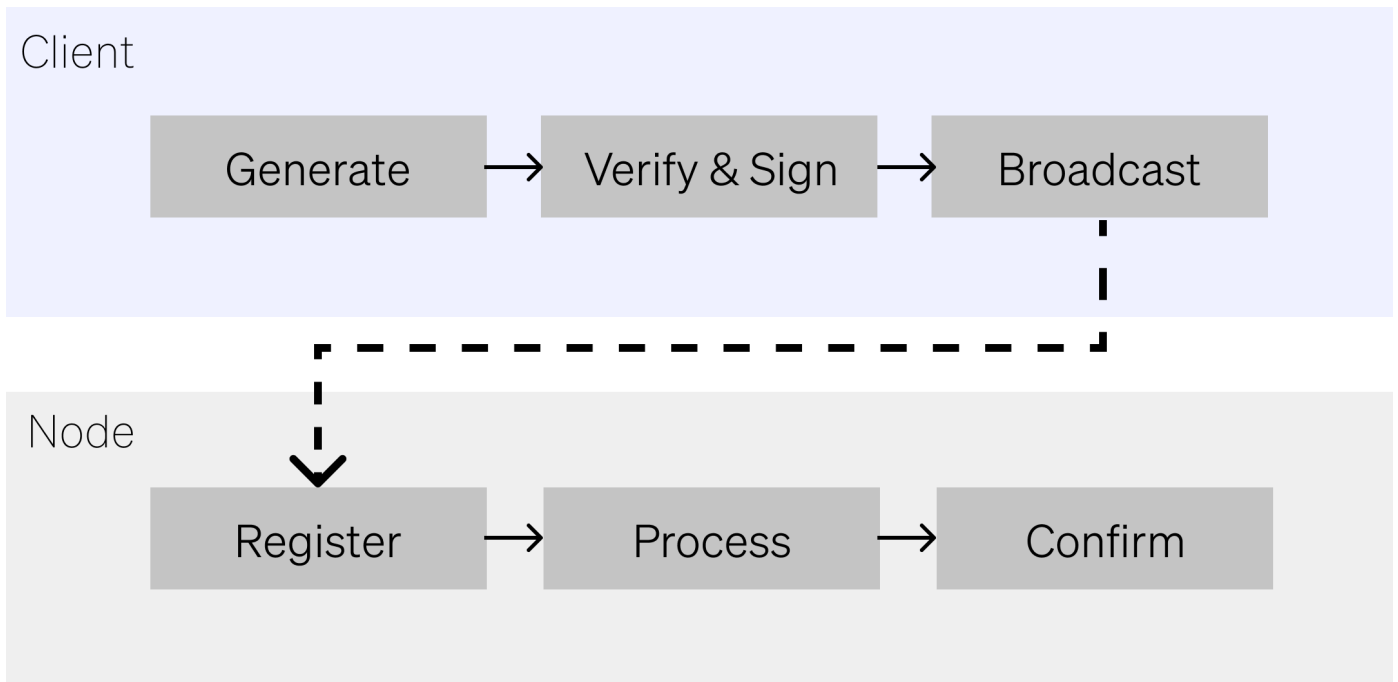
## 트랜잭션 동작 방식

## 소개

트랜잭션은 스택스 블록체인의 **실행 기본 단위**입니다. 각 트랜잭션은 스택스 계정에서 시작되며, 영구적으로 블록체인에 기록됩니다. 이 가이드는 스택스 트랜잭션의 작동 방식을 이해하는 데 도움이 됩니다.

## 라이프사이클

트랜잭션은 스택스 2.0 네트워크에서 최종 확정되기 전 여러 단계를 거칩니다.



- **생성(Generate):** 인코딩 명세에 따라 트랜잭션이 조립됩니다.
- **검증 및 서명(Validate and sign):** 트랜잭션이 올바르게 구성되었는지 확인하고 필요한 서명을 채웁니다.
- **브로드캐스트(Broadcast):** 트랜잭션이 노드로 전송됩니다.
- **등록(Register):** 채굴자가 트랜잭션을 수신, 검증하고 이를 **mempool**(대기 중인 트랜잭션 보관소)에 추가합니다.
- **처리(Process):** 채굴자가 mempool을 검토하고 블록에 포함할 트랜잭션을 선택합니다. 이 단계에서 토큰 전송 시 Post Condition 검증, 스마트 컨트랙트 토큰 발행, 기존 컨트랙트 호출 등이 발생할 수 있습니다.
- **확정(Confirm):** 채굴자가 블록을 제안하고, 스택커가 승인하면 네트워크에 전파되어 최종 확정됩니다.

트랜잭션은 등록 후 세 가지 상태 중 하나를 가집니다: pending, success, failed.

## 트랜잭션 유형

스택스는 다음과 같은 트랜잭션 유형을 지원합니다:

| 유형(Type)                  | 값(Value)                | 설명(Description)                                        |
|---------------------------|-------------------------|--------------------------------------------------------|
| Tenure change             | TenureChange            | 채굴자가 교체되어 새로운 스택스 블록 생성을 맡게 되는 이벤트. 스택커에 의해 수행됩니다.     |
| Tenure change block found | TenureChange-BlockFound | 소팅에 의해 유도되며, 새 채굴자가 블록 생성을 시작하고 기존 채굴자는 중단합니다.         |
| Tenure change extend      | TenureChange-Extend     | 스택커에 의해 유도되며, 현재 채굴자의 실행 예산을 리셋하여 계속 블록을 생성할 수 있게 합니다. |
| Token transfer            | token_transfer          | 자산을 송신자에서 수신자로 전송합니다.                                  |
| Contract deploy           | smart_contract          | 스마트 컨트랙트 배포.                                           |
| Contract call             | contract_call           | 공개된(read-only가 아닌) 컨트랙트 함수 호출.                         |

각 트랜잭션 유형의 예시는 [스택스 Blockchain API 응답 정의](#)에서 확인할 수 있습니다.

읽기 전용 컨트랙트 호출은 트랜잭션이 **필요하지 않습니다**. 자세한 내용은 네트워크 가이드를 참고하세요.

## Post Conditions

Post Condition은 스택스의 가장 독특하고 흥미로운 기능 중 하나입니다.

스택스는 설계 초기부터 보안이 핵심 원칙이었으며, Post Condition은 이러한 철학을 기반으로 탄생했습니다. Clarity(스택스의 스마트 컨트랙트 언어)와 마찬가지로, Post Condition은 사용자가 블록체인 애플리케이션과 상호작용할 때 **예상치 못한 동작을 방지**하도록 특별히 설계되었습니다.

### Post Condition의 동작 방식

Post Condition은 스마트 컨트랙트가 예상치 못한 동작을 하지 않도록 **클라이언트 측에서 설정하는 조건**입니다.

예시로, 사용자가 NFT 마켓플레이스에서 **100 STX로 NFT 구매**를 기대한다고 가정합니다. 이때 프론트엔드 개발자는 Post Condition을 추가하여 실제로 100 STX가 전송되는 경우에만 트랜잭션이 실행되도록 할 수 있습니다. 조건을 충족하지 않으면 트랜잭션은 **수수료만 제외하고 즉시 중단**됩니다.

중요한 점은, Post Condition은 **스마트 컨트랙트 내부에 존재하지 않는다**는 것입니다. 대신, 스마트 컨트랙트 위에 얹혀진 **추가 보안 레이어**이며, 프로토콜 수준에서 강제됩니다.

따라서 사용자가 악의적인 컨트랙트와 상호작용하더라도, 예상하지 못한 동작이 발생하면 트랜잭션이 중단되어 자산이 보호됩니다.

스택스 지갑(예: Hiro 웹 지갑)을 사용할 때, 개발자가 설정한 Post Condition은 지갑에 표시되어 **사용자에게 트랜잭션 실행 결과를 명확히 알립니다**. 만약 컨트랙트 동작이 조건과 일치하지 않으면 트랜잭션은 중단됩니다.

예시는 다음과 같습니다:

# mint

Requested by "Fabulous Frogs" (localhost) using  
localhost:3999

You will keep



fabulous-frogs

ST3Q...9W6H

You will keep or receive FAB or the transaction will  
abort.

You will transfer less than or equal to



50

STX

ST3Q...9W6H

You will transfer at most 50 STX or the transaction  
will abort.

위 예시에서 스마트 컨트랙트가 **NFT 한 개 전송 및 50 STX 차감**을 하지 않으면, 트랜잭션은 **즉시 중단**됩니다.

---

Post Condition의 더 자세한 동작 방식은 [SIP-005](#)와 [Hiro의 Post Condition 튜토리얼](#)에서 확인할 수 있습니다.

# Clarity

Clarity는 스택스에서 사용하는 스마트 컨트랙트 언어입니다. 개발자가 안전하고 보안성 높은 스마트 컨트랙트를 작성할 수 있도록 처음부터 설계되었습니다. Clarity는 스마트 컨트랙트 작성에 최적화된 여러 독특한 기능을 제공하며, [Clarity Crash Course](#) 가이드를 참고하면 학습을 시작하는 데 도움이 됩니다.

Clarity는 **결정가능(decidable)** 스마트 컨트랙트 언어로, 예측 가능성과 보안성을 극대화하도록 설계되었으며 스택스 블록체인에 맞춰 개발되었습니다. 스마트 컨트랙트는 개발자가 블록체인 상에서 핵심 비즈니스 로직을 코드로 구현할 수 있게 합니다.

Clarity의 설계는 Solidity에서 흔히 발생한 보안 취약점 사례에서 교훈을 얻어, 안전성과 보안성을 최우선으로 두고 만들어졌습니다.

이 문서는 Clarity에서 사용할 수 있는 함수와 키워드에 대한 레퍼런스를 주로 다룹니다.

Clarity 학습을 위해 [Clarity of Mind](#) 온라인 교재나 [Clarity Camp](#)와 같은 몰입형 교육 프로그램을 추천합니다.

---

## Clarity의 차별점

다음은 [Clarity of Mind](#)의 발췌 내용입니다:

스마트 컨트랙트 언어는 매년 늘어나고 있습니다. 초보자가 첫 언어를 선택하는 것은 쉽지 않으며, 주로 관심 있는 생태계에 따라 결정됩니다. 모든 언어에는 장단점이 있지만, 여기서는 Clarity를 돋보이게 만드는 특징과 보안·투명성이 중요한 상황에서 Clarity가 왜 최적의 선택이 되는지를 설명합니다.

Clarity의 핵심 원칙 중 하나는 **설계 자체에서의 보안성**입니다. 언어 설계 과정은 스마트 컨트랙트 개발에서 빈번히 발생하는 실수와 취약점 사례를 분석하고 이를 언어 차원에서 예방하는 방향으로 진행되었습니다. 예를 들어, 이더리움의 Parity 버그나 DAO 해킹 사건은 수억 달러 규모의 손실을 야기했으며, 이러한 문제는 언어 설계로 미리 방지할 수 있었습니다.

---

## Clarity는 인터프리터 언어

Clarity 코드는 **그대로 해석되어 체인에 기록**됩니다. Solidity와 같은 언어는 바이트코드로 컴파일된 후 체인에 제출됩니다. 컴파일 언어는 컴파일러 버그로 인한 의도치 않은 바이트코드 생성 위험과, 바이트코드의 **비가독성**이라는 문제를 가집니다. 반대로 Clarity는 작성된 코드 그대로 해석되므로 “보는 것이 실행되는 것”입니다.

## Clarity는 결정가능(Decidable)

결정가능 언어는 코드만으로 프로그램의 동작을 확실히 알 수 있는 성질을 가집니다. 즉, 어떤 입력이 주어지더라도 유한 단계 내에 실행이 종료됨이 보장됩니다. 이를 통해 호출 그래프를 정적으로 완전히 분석할 수 있으며, 실행 전에 정확한 비용을 예측할 수 있습니다. 따라서 실행 도중 “가스 부족” 상황이 발생할 수 없습니다.

## 재진입(Reentrancy) 방지

재진입은 한 컨트랙트가 다른 컨트랙트를 호출한 뒤 다시 원래 컨트랙트로 제어가 돌아오는 상황입니다. 이는 공격자가 잔고 업데이트 전 다중 출금을 유도할 수 있는 취약점입니다. Clarity는 이를 언어 차원에서 **금지**합니다.

## 오버플로/언더플로 방지

수치 연산에서 결과가 저장 범위를 초과(오버플로)하거나 부족(언더플로)하면 컨트랙트가 비정상 상태에 빠지거나 악용될 수 있습니다. Clarity는 이와 같은 상황에서 트랜잭션을 자동으로 중단시킵니다.

## 사용자 정의 토큰 내장 지원

대부분의 스마트 컨트랙트 사용 사례는 토큰 발행입니다. Clarity는 **대체 가능(Fungible)·대체 불가능(Non-Fungible) 토큰 발행 기능을 언어에 내장**하고 있어, 개발자가 별도의 잔고 관리·공급 관리 이벤트를 처리할 필요가 없습니다.

## Post Condition 기반 보안

스택스에서는 트랜잭션에 Post Condition을 붙여 실행 후 체인 상태가 예상대로 변경되었는지 확인할 수 있습니다. 예: 컨트랙트 호출 시 “500 STX가 특정 주소로 전송되어야 한다”는 조건을 붙일 수 있으며, 조건 검증이 실패하면 트랜잭션은 전체가 롤백됩니다.

## 응답 처리 강제

공개 컨트랙트 호출은 반드시 성공/실패 여부를 나타내는 **response**를 반환합니다. 다른 컨트랙트를 호출하는 컨트랙트는 이를 적절히 처리해야 하며, 그렇지 않으면 배포가 불가능합니다. 이는 Solidity의 “무응답(low-level call)” 허용과 대조됩니다.

## 상속 대신 컴포지션

Clarity는 **상속보다 컴포지션**을 채택했습니다. 스마트 컨트랙트는 상속 구조를 가지지 않고, 대신 trait을 정의해 구현합니다. 이를 통해 더 유연한 인터페이스 적합성을 제공하며 복잡한 클래스 계층 문제를 피할 수 있습니다.

## 비트코인 접근성

Clarity 스마트 컨트랙트는 비트코인 베이스 체인의 상태를 읽을 수 있습니다. 즉, **비트코인 트랜잭션을 스마트 컨트랙트 동작의 트리거로 활용**할 수 있습니다. 또한 secp256k1 서명 검증과 키 복구를 위한 함수도 내장되어 있습니다.

---

# 개요 (Overview)

Clarity는 예측 가능성과 보안성을 위해 설계된 **결정가능 언어**이며, 스택스 블록체인에 특화되었습니다. 설계 철학은 Solidity의 취약점에서 교훈을 얻어 보안성을 최우선으로 한 것입니다.

이 문서는 Clarity에서 사용할 수 있는 **함수와 키워드에 대한 레퍼런스**를 제공합니다.

Clarity 학습을 원한다면 [Clarity of Mind](#) 온라인 교재 또는 [Clarity Camp](#)에 참여해 보세요.

# 결정가능성(Decidability)

**언어가 비튜링 완전(Non-Turing Complete)하거나 결정가능(Decidable)하다는 건 무엇인가요?**

Clarity의 보안적 장점을 설명할 때 자주 등장하는 두 용어가 **비튜링 완전**과 **결정가능**입니다. 서로 관련은 있지만 같은 뜻은 아닙니다.

## 비튜링 완전(Non-Turing Complete)

어떤 시스템/언어가 **비튜링 완전**하다는 것은 튜링 기계(계산의 추상 모델)를 모사할 수 없다는 뜻입니다. 즉, 튜링 완전 언어보다 계산 능력이 제한됩니다. 튜링 완전 언어는 임의의 튜링 기계를 모사할 수 있습니다. 비튜링 완전 시스템의 예로는 유한 상태 기계(FSM)나 특정 도메인 전용 언어(Clarity 등)가 있습니다.

비튜링 완전 언어는 모든 알고리즘을 표현할 수 없습니다. 특히 **무한 루프**나 **무제한 재귀**가 필요한 문제는 표현할 수 없습니다. Clarity에서는 이 특성이 언어 차원에서 **무한 루프**와 **재진입(reentrancy)**을 금지하는 근거가 됩니다.

## 결정가능(Decidable)

어떤 **문제(problem)**가 결정가능하다는 것은, 모든 입력에 대해 유한 시간 내에 그 성질을 판정하는 알고리즘이 존재한다는 뜻입니다. 즉, 모든 입력에서 **반드시 종료**하는 튜링 기계로 풀 수 있는 문제입니다. 결정가능성은 “문제”의 성질이고, 튜링 완전성은 “언어/계산 시스템”의 성질입니다.

Clarity가 **결정가능 언어**라는 사실은, 입력이 무엇이든 **행동을 예측하고 증명**하기 쉬워진다는 의미입니다.

---

## 스마트 컨트랙트 개발자의 마인드셋

솔리디티와 클래리티의 보안 속성을 살펴보면, 많은 이슈는 개발자의 “주의 깊은 구현”으로도 **완화(mitigation)**할 수 있습니다. 하지만 사람은 실수합니다. **기능을 유지하면서 인간 오류 가능성을 줄일 수 있다면, 언어/플랫폼 차원에서 줄이는 편이 낫습니다.**

---

## 스마트 컨트랙트는 튜링 완전해야 할까?

스마트 컨트랙트의 활용은 점점 넓어지고 있습니다. 그렇다면 **데이터와 프로그램을 분리**한다는 전통적 원칙을 떠올려 봅시다. 컨트랙트는 데이터여야 할까요, 프로그램이어야 할까요, 혹은 그 사이 어딘가일까요? 프로그램이라면 **어떤 언어로, 어떤 성질을 가져야 할까요?**

**Church-Turing 논제**에 따르면 “계산”의 모든 형식적 개념은 튜링 기계로 포착됩니다. 파이썬, C++, 러스트, 자바, 리스프, 솔리디티 등 대부분의 범용 언어가 **튜링 완전**합니다.

그러나 튜링 완전 언어의 일반 프로그램은 **최악의 경우 검증 불가능**합니다. 프로그램의 올바름을 증명하는 **정확성 증명(proof-of-correctness)**은 일반적으로 **결정불가능**합니다(괴델의 불완전성 정리/결정불가능 결과). 따라서 **모든** 튜링 완전 프로그램을 검증할 수는 없습니다.

또한 Church와 Turing은 **비계산 문제(uncomputable problems)**가 존재함을 보였습니다. 이 문제들은 어떤 컴퓨터로도 풀 수 없습니다.

**Clarity는 비튜링 완전이면서도 표현력이 높습니다.** 따라서 **결정가능하며, 비계산 문제를 표현할 수 없습니다.** 솔리디티에서도 비튜링 완전한 “부분집합”을 정의하려는 시도가 있었지만, 합의된 표준은 드물고 실제 사용도 제한적입니다.

---

## 스마트 컨트랙트에서 ‘결정가능성’의 장점

### 1. 실행 중 가스 부족이 없습니다.

Clarity는 결정가능하므로 호출 그래프를 **완전 정적 분석**할 수 있고, 실행 전 **정확한 비용 추정**이 가능합니다. 반면 솔리디티는 무한/무제한 루프·재귀·동적 호출이 가능해, 사전 비용 추정이 어렵고 실행 중 **가스 고갈**이 발생할 수 있습니다.

### 2. DoS 취약점 감소

솔리디티의 고전적 DoS 사례(예: GovernMental)처럼, 거대한 매핑 삭제/정리 과정이 블록 가스 한도를 초과해 컨트랙트가 멈추는 문제가 있습니다. Clarity는

- 재귀 미지원,
- 이터레이션 가능한 자료형에 ‘상한’ 강제(리스트는 길이 제한 필수, 맵/튜플은 순회 불가)  
같은 언어 제약으로 이런 종류의 DoS를 예방합니다. “임의 크기의 매핑” 같은 패턴은 **반패턴**이며, 사용자 주도 처리 등 **대안적 설계를 유도**합니다.

### 3. 형식 검증(Formal Verification) 용이성

경로 폭발(path explosion)을 완화합니다. **무제한 루프가 불가**하므로 검증 공간이 줄어들고, 수학적 증명이 현실적으로 접근 가능해집니다.

복잡성은 보안의 적입니다. 복잡할수록 버그와 비계산 문제가 늘어납니다. 특히 **불변(immutable)**인 공개 금융 인프라에서는 언어 차원의 제약으로 **복잡성 증가의 악순환**을 끊는 것이 중요합니다.

자세한 언어 설계 배경은 [SIP-002](#)를 참고하세요.

참고: 일반적 스마트 컨트랙트 취약점과 Clarity의 대응은 이 글도 유용합니다.

<https://stacks.org/bringing-clarity-to-8-dangerous-smart-contract-vulnerabilities/>

## 비계산성(Uncomputability)에 대한 직관

비계산성은 결정불가능성과 같은 뿌리를 둡니다. 대표적으로 **정지 문제(Halting Problem)**가 있습니다: 임의의 프로그램 P와 유한 입력 I가 주어졌을 때, P가 I에서 **멈추는지** 판정하는 문제는 **해결 불가능**합니다.

또 다른 예로 **Post 대응 문제(Post Correspondence Problem)**가 있습니다.

문자열 쌍 집합  $P = \{ (x_1, y_1), \dots, (x_n, y_n) \}$ 과 정수  $m > 0$ 가 주어졌을 때, 인덱스열  $(i_1, \dots, i_m)$ 을 골라  $x(i_1) x(i_2) \dots x(i_m) = y(i_1) y(i_2) \dots y(i_m)$  를 만족하게 할 수 있는지 묻습니다. 일반적 해법은 존재하지 않으며 **비계산 문제**입니다. 이런 문제들은 보통 **무정해 루프** 탐색을 유도하며, 바로 이런 패턴을 Clarity가 언어 차원에서 제한합니다.

요점: **프로그램 행동에 관한 많은 질문이 비계산적**이며, 튜링 완전 언어 기반 컨트랙트는 본질적으로 이러한 위험을 내포합니다. Clarity는 이를 구조적으로 배제합니다.

## 레이먼드 스멀리언(R. Smullyan)의 결정불가능성 직관

기사도/악한의 섬(“Knight & Knave”) 비유로 **체계 내부에서 판정 불가능한 진술**이 존재함을 직관적으로 보여줍니다. 이것을 스마트 컨트랙트에 비유하면, **튜링 완전 언어**는 체계 내에서 **결정 불가능한 상태**를 만들 수 있고, 이는 예기치 않은 동작·취약점·자원 소모(가스 고갈 등)로 이어질 수 있습니다.

반면 **비튜링 완전 언어(Clarity)**는 **무한 루프/재귀 등 역설을 유발하는 구성 자체를 불허**하여, 예측 가능성과 자원 사용의 상한을 제공합니다. 고가치/핵심 시스템일수록 이런 **예측 가능성**이 바람직합니다.

# sBTC

## sBTC 소개

sBTC는 스택스 블록체인에서 발행되는 **SIP-010 토큰**으로, 비트코인(BTC)을 **1:1 비율**로 나타냅니다. 이를 통해 비트코인 보유자는 기본 자산인 BTC와의 페그(peg)를 유지하면서 DeFi 애플리케이션 및 기타 스마트 컨트랙트 기능에 참여할 수 있습니다.

## 목적

sBTC의 주된 목적은 스택스 블록체인을 통해 **비트코인을 DeFi로 연결**하는 것입니다. 이를 통해 비트코인 보유자는 BTC의 보안성과 가치를 유지하면서도 스마트 컨트랙트의 풍부한 기능을 활용할 수 있습니다.

## 주요 장점

1. **비트코인 호환성**: 비트코인을 매도하지 않고도 스택스 생태계에 참여 가능
2. **빠른 전환**: BTC ↔ sBTC 이동은 빠르게 완료 (입금 시 3개 블록, 출금 시 6개 블록 이내)
3. **탈중앙화 관리**: 커뮤니티가 선택한 15명의 서명자 세트가 페그 지갑 유지
4. **커뮤니티 거버넌스**: 초기 서명자 집합 선정 등 핵심 의사결정에 커뮤니티 참여

---

## 핵심 개념

sBTC를 이해하기 위해 알아야 할 주요 개념은 다음과 같습니다:

### sBTC

sBTC는 스택스 블록체인 상의 SIP-010 토큰으로, 비트코인 블록체인에서 BTC로 다시 전환될 수 있습니다. **항상 1 sBTC = 1 BTC** 비율이 유지됩니다.

### sBTC UTXO

sBTC UTXO는 비트코인 블록체인 상에서 **sBTC로 페깅된 전체 BTC 잔액**을 보관하는 **단일 미사용 트랜잭션 출력(UTXO)**입니다. 이는 sBTC 서명자 집합이 관리·유지합니다.

### sBTC Signer

sBTC Signer는 스택스의 Nakamoto 서명자와 구분되는 서명자 엔티티입니다. 주요 역할은:

- sBTC 연산 서명
- 스택스 체인 상의 sBTC 컨트랙트와 통신
- sBTC UTXO 관리

## sBTC Signer Set

sBTC 서명자 집합 전체를 의미합니다. 이들은 sBTC UTXO에 대한 민주적 접근 권한을 가지며, 페그 지갑 보안을 유지합니다. 또한 보안을 위해 개인 키를 순환(rotating)할 수 있습니다.

## Emily API

Emily는 sBTC 브릿지를 **지원 및 감독**하는 API로, 사용자와 서명자 사이의 **프로그래매틱 인터페이스** 역할을 합니다.

## SIP-010 토큰

sBTC는 스택스 블록체인의 [SIP-010](#) 표준을 따릅니다. 이를 통해 SIP-010을 지원하는 지갑 및 애플리케이션과 호환됩니다.

---

# 핵심 기능 (Core Features)

sBTC는 스택스와 Bitcoin 간의 **신뢰 최소화 비트코인 브릿지**로서 강력한 기능을 제공합니다:

## 1. 1:1 비트코인 담보

각 sBTC는 페그 지갑에 보관된 동일량의 BTC로 담보됩니다. 따라서 sBTC의 가치는 BTC와 안정적으로 연동됩니다.

## 2. 탈중앙화 관리

sBTC 페그 지갑은 sBTC 서명자 집합에 의해 관리됩니다. 이를 통해 보안성이 강화되고 단일 장애점 (single point of failure)이 제거됩니다.

## 3. 빠른 전환

sBTC는 BTC와 sBTC 간의 빠른 이동을 지원합니다:

- **BTC → sBTC**: 3개 비트코인 블록 내 완료
- **sBTC → BTC**: 6개 비트코인 블록 내 완료

## 4. SIP-010 호환성

sBTC는 스택스 블록체인의 **SIP-010 토큰 표준**을 준수하여, SIP-010 지원 지갑과 애플리케이션에서 호환됩니다.

## 5. 커뮤니티 거버넌스

초기 sBTC 서명자 집합은 **STX 보유량 가중 투표**를 통해 커뮤니티가 결정합니다. 이는 시스템 운영에 커뮤니티의 참여를 보장합니다.

## 6. 서명자 키 회전

sBTC 서명자는 **개인 키를 주기적으로 교체**할 수 있어 장기적 보안을 강화합니다.

## 7. 트랜잭션 수수료 후원

sBTC 트랜잭션은 **sBTC로 수수료 지불**이 가능하도록 후원(sponsorship)을 지원합니다. 따라서 사용자는 STX 대신 sBTC로 수수료를 낼 수 있습니다.

---

이러한 핵심 기능들이 결합되어, sBTC는 **안전하고 효율적이며 사용자 친화적인 브릿지**를 제공하며 비트코인과 스택스 생태계를 연결합니다.

# sBTC 운영(Operations)

이 섹션은 sBTC 시스템의 핵심 오퍼레이션을 다룹니다:

1. 입금(Deposit): BTC → sBTC 전환
2. 출금(Withdrawal): sBTC → BTC 전환

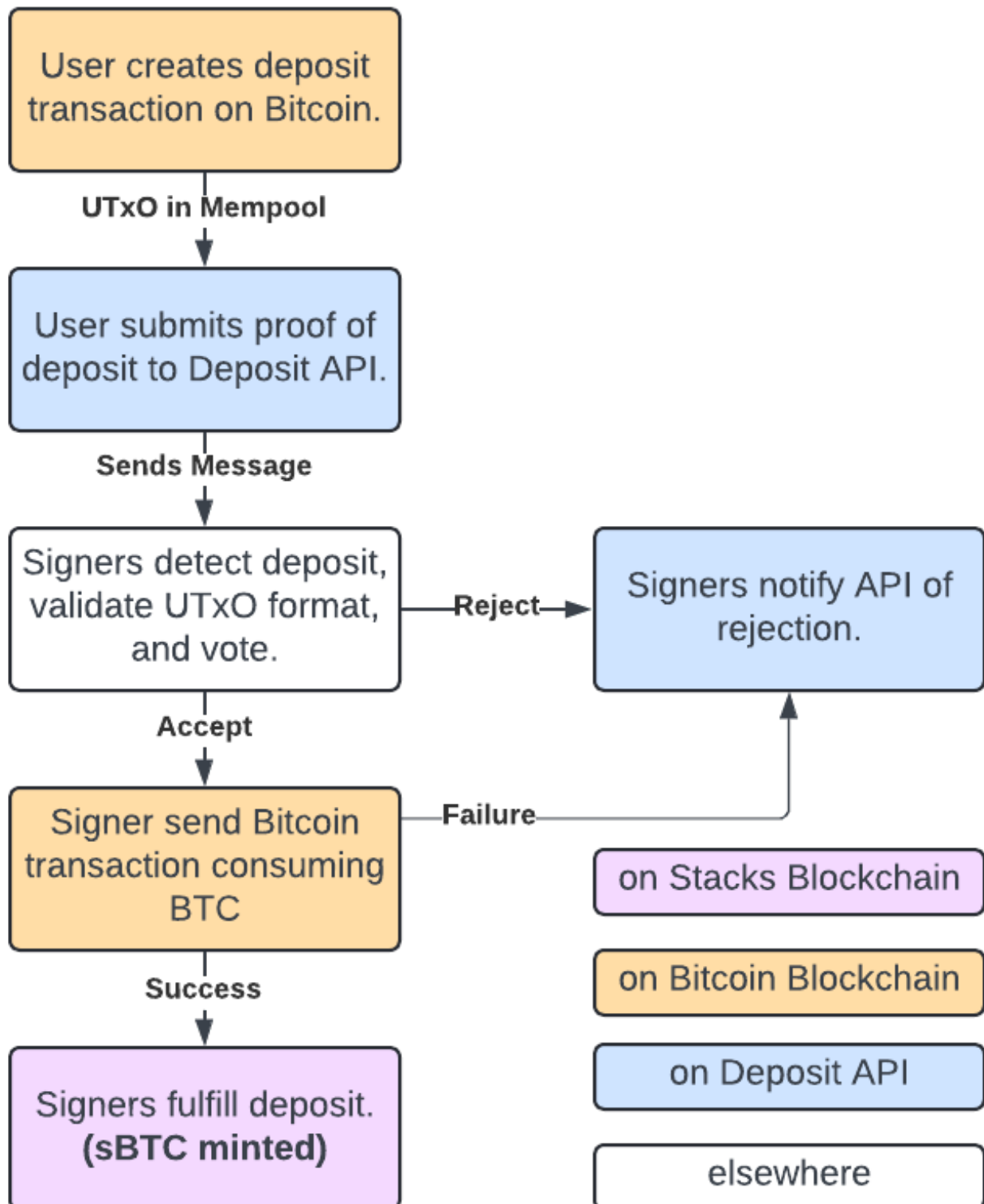
이 두 동작이 sBTC의 핵심 기능으로, 사용자가 비트코인과 스택스 생태계 사이에서 가치를 이동하도록 합니다.

## Deposit (입금)

입금 오퍼레이션은 사용자가 비트코인 체인의 **스레시홀드 월릿(threshold wallet)**에 BTC를 예치하고, 그에 연동된 sBTC를 **민트**하도록 합니다. 이 과정은 일반적으로 **단일 비트코인 블록** 내에서 완료될 수 있어 사용자 경험이 간단합니다.

## 프로세스 개요

## sBTC-v1 High Level Deposit Flow



입금은 사용자가 **두 개의 출력**을 가진 특정 비트코인 트랜잭션을 생성하면서 시작됩니다:

1. 서명자들이 지출할 수 있도록 하는 스크립트

## 2. 예치자가 자금을 회수할 수 있도록 하는 타임락

그 다음, 예치자(일반적으로 예치 기능을 제공하는 앱을 통해)는 해당 비트코인 트랜잭션을 참조하는 **API 호출**을 시작합니다. 이 호출은 **Emily API**를 트리거하여 입금 정보를 sBTC 서명자들에게 전달합니다. 서명자들은 입금을 검증·처리하고, 검증이 완료되면 **동일 수량의 sBTC**가 스택스 블록체인에서 민트됩니다.

입금은 보통 **1개의 비트코인 블록** 내에 완료되지만, **최대 3개 블록** 내 완료가 보장됩니다. 더 자세한 확인 시간과 이유는 [Deposit and Withdrawal Times](#) 문서를 참고하세요.

## 비트코인 입금 요건

입금이 유효하려면 다음 요건을 충족해야 합니다. 입금은 **탭루트 주소**로 이루어져야 하며, **합의 임계치에 도달한 서명자 집합**이 지출할 수 있어야 합니다. 또한 **단기 클로백**을 방지하는 특정 포맷을 따라야 하며, 이를 통해 시스템의 보안과 무결성을 보장합니다.

## 사용자 경험

사용자 관점에서 입금 과정은 단순합니다. 지정된 주소로 BTC 트랜잭션을 전송하고, 해당 트랜잭션이 비트코인 체인에서 컨펌되면 **스택스 지갑으로 동등한 sBTC**를 받습니다.

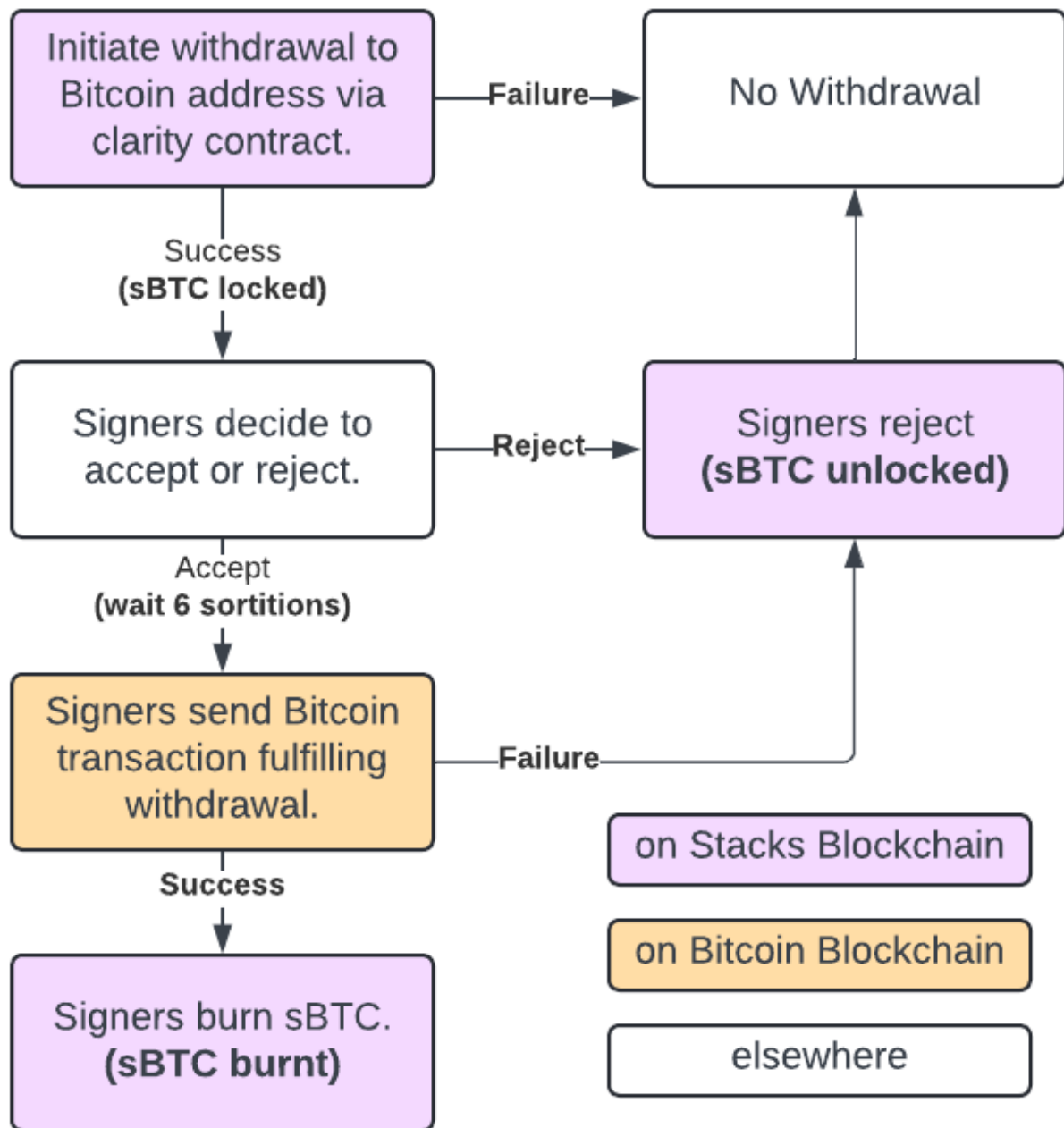
경험 향상을 위해, **sBTC 브리지 웹 앱**이 개발 중이며, 사용자가 입금 진행 상태를 직관적으로 추적할 수 있는 UI를 제공합니다.

## Withdrawal (출금)

출금 오퍼레이션은 사용자가 보유한 **sBTC**를 소각(burn) 하고, 비트코인 블록체인에서 **동일 수량의 BTC**를 수령하도록 합니다.

## 프로세스 개요

## sBTC-v1 High Level Withdrawal Flow



출금은 사용자가 **Clarity 컨트랙트 호출**을 시작하면서 진행됩니다. 일련의 이벤트를 거쳐 사용자가 지정한 비트코인 주소로 BTC가 전송됩니다. 이 과정은 **6개의 비트코인 블록 컨펌**을 필요로 하며, 컨펌이 완료되면 sBTC 서명자들이 비트코인 네트워크에서 출금 트랜잭션을 생성합니다.

## 출금 컨펌(확정)

**6블록 컨펌** 요건은 다음 이유로 중요합니다.

- 스택스 트랜잭션의 **최종성(finality)** 확보
- 잠재적 **비트코인 포크**로 인한 문제 완화
- sBTC 서명자들이 출금 요청을 **정확히 검증**할 충분한 시간 확보

입·출금 시간과 입금이 더 빠른 이유는 [Deposit and Withdrawal Times](#) 문서를 참고하세요.

## 실패 케이스

일부 출금 실패는 6블록 컨펌 전에 탐지·해결될 수 있으나, 다른 경우는 **sBTC 부트스트랩 서명자**가 비트코인 출금 트랜잭션을 생성하려는 시점 이후에야 드러날 수 있습니다. 이는 **크로스체인 오퍼레이션의 복잡성**과 단계별 철저한 검증 필요성 때문입니다.

## 보안 고려사항

여러 블록 컨펌을 요구하는 것은 **이중지불 방지**에 핵심적입니다. 컨펌 요건을 통해 요청의 유효성과 최종성을 확인한 뒤 비트코인 네트워크에서 처리합니다. 또한 sBTC 서명자들은 각 출금 요청을 **검증**하여 추가 보안 레이어를 제공합니다.

## 사용자 경험

사용자는 스택스 지갑 또는 dApp에서 출금을 시작하며, **출금할 sBTC 수량**과 **수령할 비트코인 주소**를 지정합니다. 요청 제출 후 **6블록 컨펌**을 기다리면, BTC가 지정한 주소로 전송됩니다.

사용자 경험을 개선하기 위해, **sBTC 브리지 웹 앱**은 실시간 상태 업데이트를 제공하여 각 단계의 진행 상황을 투명하게 보여줍니다.

## 입금 vs 출금 시간

### 왜 입금은 빠르고 출금은 느린가?

sBTC는 **Wrapped BTC(sBTC)**를 통해 비트코인을 스택스 L2에서 사용하도록 합니다. **BTC → sBTC**는 **최단 1 비트코인 블록** 내 완료가 가능하지만, **sBTC → BTC**는 **6 블록**이 필요합니다. 이유를 이해하려면 우선 스택스의 **PoX(Proof of Transfer)** 합의 메커니즘을 알아야 합니다.

스택스에서는 각 비트코인 블록마다 스택스 채굴자들이 일정량의 BTC를 지불(커밋)하여 **다음 수 초~수십 초 간의 스택스 블록 제작 권한(테뉴어)** 을 경매처럼 경쟁합니다. **서명자(Signers)** 는 비트코인 블록을 참고해 **현재 권한을 가진 채굴자가 제안한 블록만 승인**하며, 동시에 “가장 좋은 비트코인 포크”에 결부된 블록만 승인합니다. 즉, 스택스 블록은 서명자 합의 없이는 추가될 수 없습니다.

핵심 포인트: **비트코인 체인이 더 나은 포크로 전환되면**, 서명자들은 그 포크에 결부된 스택스 블록만 승인합니다. 즉, **스택스 체인도 비트코인 체인과 함께 포크**됩니다.

이제 왜 출금은 느려야 하는지 봅시다.

**출금**은 스택스 레이어에서 sBTC를 소각하고, 비트코인 레이어에서 BTC를 지급하는 동작입니다. **서명자들이 비트코인 출금 트랜잭션을 만들면 되돌릴 수 없고**, 해당 트랜잭션은 **모든 비트코인 포크에서 유효**합니다. 만약 그 사이에 비트코인 체인이 포크되어, 스택스 상의 출금 트랜잭션이 **재조직(reorg)**으로 사라진다면 어떻게 될까요? 비트코인에서는 BTC가 이미 전송되었지만, 스택스에선 출금이 없었던 일이 되어 버립니다. 이를 막기 위해 **6블록의 확정성**을 요구하는 것입니다.

반대로 **입금**은 왜 빠를까요? 비트코인에서 입금 트랜잭션이 발생하자마자 sBTC를 민트했다고 가정합니다. 만약 비트코인 체인이 포크되어 그 입금이 사라지면, **스택스도 함께 포크**되며 그 민트 역시 사라집니다. 즉, 입금이 영원히 오지 않으면 **민트도 영원히 일어나지 않습니다**. 따라서 **민트는 비트코인의 최종성을 기다리지 않아도 안전**합니다(어차피 둘 다 같이 되돌아가므로).

정리하면:

- **스택스 → Bitcoin(출금)**: 비가역적인 **BTC 전송**이 수반되므로, 안전을 위해 **충분한 비트코인 최종성(6블록)** 을 기다립니다.
- **Bitcoin → 스택스(입금)**: 최종성을 기다리지 않고도 **스택스가 비트코인을 따라 재조직**되므로 빠르게 민트 가능합니다. 다만 **민트를 최종적으로 안전하다고 보려면** 사용자가 비트코인 블록을 충분히 기다리면 됩니다(이는 사용자의 선택입니다).

# Emily API

Emily는 sBTC 브리지를 **지원·감독**하는 API로, sBTC 사용자와 서명자들 사이의 **프로그래매틱 중계자** 역할을 합니다.

## 개요

Emily API는 **입금/출금의 진행 상황**을 추적하고, **진행 중인 sBTC 오퍼레이션의 상태**를 제공합니다. 주요 이용자는 **sBTC 사용자와 sBTC 앱 개발자**입니다.

# 왜 Emily인가?

Emily는 단순한 입·출금 중계 이상의 역할(시스템 헬스 체크 등)을 수행하며, 향후 기능 확장 가능성을 내포합니다. 과거 “Revealer API”, “Deposit API” 등의 이름을 거쳤으나 책임 범위를 오인시킬 여지가 있어, **브리지의 감독자** 역할을 맡았던 **에밀리 로블링(Emily Warren Roebling)**에서 착안한 이름으로 정했습니다.

## 핵심 기능

- 1. **입금 추적**: BTC → sBTC 변환 과정 모니터링
- 2. **출금 추적**: sBTC → BTC 변환 과정 모니터링
- 3. **상태 제공**: 진행 중 오퍼레이션의 실시간 상태 제공
- 4. **이력 조회**: 과거 sBTC 오퍼레이션 기록 질의

## 핵심 개념

### sBTC 오퍼레이션

Emily가 추적하는 기본 단위는 다음과 같습니다.

- 1. **Deposits**: BTC → sBTC
- 2. **Withdrawals**: sBTC → BTC

### 오퍼레이션 상태

각 오퍼레이션은 다음 상태를 거칩니다.

- 1. **PENDING**: 요청이 시작됨
- 2. **ACCEPTED**: 서명자 승인
- 3. **CONFIRMED**: 블록체인 상에서 완료·확정
- 4. **FAILED**: 완료 불가

## 상호작용 플로우

### 입금 플로우

- 1. 사용자가 비트코인에서 **입금 트랜잭션** 생성
- 2. 사용자가 **입금 증명**을 Deposit API에 제출

3. Emily가 상태를 **PENDING**으로 기록
4. 서명자들이 **검증 및 투표**
5. 승인 시 Emily가 **ACCEPTED**로 업데이트
6. 서명자들이 비트코인 트랜잭션 처리
7. 서명자들이 스택스에서 **sBTC**를 **민트**
8. Emily가 상태를 **CONFIRMED**로 업데이트

## 출금 플로우

1. 사용자가 **sBTC Clarity 컨트랙트**로 출금 요청
2. Emily가 상태를 **PENDING**으로 기록
3. 서명자들이 **수락/거절** 결정
4. 수락 시 Emily가 **ACCEPTED**로 업데이트
5. 서명자들이 비트코인 출금 트랜잭션 처리
6. 서명자들이 스택스에서 **sBTC** 소각
7. Emily가 상태를 **CONFIRMED**로 업데이트

# Peg Wallet UTXO

**페그 월렛 UTXO(Peg Wallet UTXO)**는 유통 중인 모든 sBTC 토큰을 담보하는 비트코인 측 자산으로, sBTC 시스템의 핵심 요소입니다. 이 시스템은 **단일 UTXO 모델(Single UTXO Model)**을 채택하여, sBTC 페그 지갑을 비트코인 블록체인 상의 **단 하나의 미사용 트랜잭션 출력(UTXO)**로 일관되게 표현합니다. 이 설계는 페그 지갑 관리의 단순성과 효율성을 높여 줍니다.

UTXO 관리는 **서명자 집합(Signer set)**의 책임입니다. 먼저 **서명자 코디네이터**가 UTXO를 구성합니다. 이어서 서명자 집합은 모든 입금/출금 요청을 집계하여, **단일 UTXO** 내에서 처리될 수 있도록 **최적화된 배치(batch)**를 만듭니다. 새 UTXO는 이전 UTXO의 금액을 소진(spend)하면서, **확정된 입금을 더하고 확정된 출금을 빼는 방식**으로 생성됩니다.

배치 과정은 단일 UTXO **불변식(invariant)**을 유지하면서도 최적의 배치를 만들도록 신중하게 설계되어 있습니다. 여러 건의 sBTC 오퍼레이션 요청이 동시에 존재하면, 서명자 코디네이터는 이를 **승인 세트(approval set)**별로 그룹화합니다. 활성 오퍼레이션 간 승인 세트가 서로 다를 경우, 코디네이터는 **각 승인 세트별 최대 크기**로 입금 UTXO들을 묶어 배치합니다.

이 **단일 UTXO 모델**은 여러 장점이 있습니다. 추적 및 관리가 단순해지고, sBTC 오퍼레이션에 필요한 비트코인 트랜잭션 수가 줄어들며, 자금을 하나의 **잘 보호된 출력**으로 집중시킵니다. 이러한 이점은 sBTC 시스템의 전반적 효율성과 보안성에 기여합니다.

보안은 이 모델에서 최우선 과제입니다. 단일 UTXO는 **sBTC 부트스트랩 서명자 세트**가 관리하며, **임계치 (threshold)** 이상 서명자의 승인이 있어야 지출이 가능합니다. 이러한 **다중서명** 접근은 자금 보호를 한층 강화합니다. 또한, **정기 감사**와 **지속 모니터링**을 통해 UTXO가 언제나 유통 중인 sBTC 총량을 정확히 반영하는지 확인하는 것이 필수적입니다.

이처럼 단순성, 효율성, 보안성 간 균형을 취하는 **Peg Wallet UTXO** 시스템은 sBTC 생태계의 견고한 기반을 이룹니다.

## 보조 기능

다음은 sBTC의 기능성과 보안성을 강화하는 보조 기능입니다.

1. 거래 수수료 스폰서십(Transaction Fee Sponsorship): sBTC 트랜잭션의 수수료를 스폰서가 대납
2. 서명자 지갑 로테이션(Signer Wallet Rotation): sBTC 서명자의 안전한 키 교체 및 구성 변경

이러한 보조 기능은 sBTC 생태계의 **견고함**과 **사용자 친화성**을 높여 줍니다.

## 거래 수수료 스폰서십(Transaction Fee Sponsorship)

거래 수수료 스폰서십은 사용자가 **STX 대신 sBTC**로 스택스 트랜잭션 수수료를 부담할 수 있도록 해 주는 기능입니다.

### 개요

- 스택스 상의 sBTC 트랜잭션은 **스폰서에게 sBTC를 지불**하는 조건으로 **대납**될 수 있습니다.
- sBTC 보유자가 **별도의 STX 없이**도 트랜잭션을 실행할 수 있어 **사용성(UX)**이 개선됩니다.

### 구현

수수료 스폰서십은 [stacks-network/stacks-core#4235](https://github.com/stacks-network/stacks-core/pull/4235)에서 제안된 접근을 바탕으로 구현됩니다.

핵심 포인트:

1. sBTC 사용자는 기존 **STX 보유 스폰서**의 지원을 받아 수수료를 처리할 수 있습니다.
2. 스폰서가 **STX 수수료를 지불하고, 대가로 sBTC**를 받습니다.

## 사용자 경험

1. 사용자는 sBTC 트랜잭션을 시작하면서 **스폰서십 옵션**을 선택할 수 있습니다.
2. 사용자는 **소량의 sBTC**를 스폰서 수수료로 지불하는 데 동의합니다.
3. 트랜잭션은 **스폰서가 STX로 수수료를 납부**하는 형태로 처리됩니다.

## 이점

1. **UX 개선**: sBTC만 보유해도 트랜잭션 실행 가능
2. **진입장벽 완화**: 신규 사용자는 먼저 STX를 확보할 필요 없음
3. **유연성**: 수수료 처리 방식에 **추가 선택지** 제공

# 서명자 지갑 로테이션(Signer Wallet Rotation)

서명자 지갑 로테이션은 sBTC 서명자가 **개인키를 갱신**하고, **서명자 집합 구성**을 변경할 수 있게 하는 메커니즘입니다. 네트워크는 이를 통해 시간이 지나도 보안을 유지하고 참여자 변동에 적응합니다.

## 동작 방식

sBTC 시스템은 비트코인에서 **멀티시그 지갑**로 BTC 예치를 보관합니다. 지갑의 통제권(키 교체/구성 변경)을 바꿀 필요가 있을 때, **로테이션 메커니즘**을 사용합니다.

v1.1.0 기준 시스템은 다음을 지원합니다.

- 새로운 서명자 **추가**
- 기존 서명자 **제거**
- 특정 서명자 **교체**
- 현재 서명자의 **키 로테이션**

서명자들이 새로운 설정에 합의하면, 시스템은 자동으로 **DKG(분산 키 생성, Distributed Key Generation)** 프로토콜을 실행하여 업데이트된 그룹용 **새 서명 지분(shares)**을 생성합니다. 완료되면 sBTC 지갑의 통제권이 새로운 구성으로 **안전하게 이전**됩니다.

## 로테이션 프로세스

일반적인 로테이션 동안에는 다음이 일어납니다.

1. 서명자들이 **오프체인으로 새로운 서명자 집합**을 합의
2. 각 서명자 운영자는 **동일한 집합**으로 자신의 설정을 업데이트
3. 모든 서명자가 **정확히 동일한 집합**을 구성하면 **DKG 자동 실행**
4. **새 서명자 집합**이 sBTC 지갑의 통제권을 인수

이 과정 내내 비트코인 UTXO는 **연속적으로 통제**되므로, 자금이 무방비 상태가 되는 순간은 없습니다.

## 로테이션이 발생하는 때

- **서명자 변경**: 서명자가 집합을 떠나거나 새 참여자가 합류할 때
- **보안 이벤트**: 키가 **노출/위협**되었을 우려가 있을 때 **긴급 로테이션** 수행

## 워크스루

아래 워크스루는 **사용자와 서명자**가 sBTC 시스템과 상호작용하는 방식을 높은 수준에서 설명합니다. 이를 통해 sBTC 시스템 **내부 동작**에 대한 탄탄한 이해를 얻을 수 있습니다.

# 서명자 절차 워크스루(Signer Process Walkthrough)

## 소개

이 문서는 **sBTC 서명자 노드**의 운영에 초점을 맞춘 sBTC 시스템 개요입니다. sBTC 생태계에서 자동화된 처리와 소프트웨어 간 상호작용이 어떻게 이루어지는지 설명합니다.

서명자 노드의 **설치 및 운영 가이드**(단계별)는 준비 중이며, 본 문서는 서명자 관점에서 **역할과 흐름**을 이해하기 위한 개념적 안내입니다.

## 서명자 노드 셋업

sBTC 서명자로서 가장 중요한 책임은 **서명자 노드를 안정적으로 운영**하는 것입니다. 다음을 포함합니다.

1. **하드웨어 준비**: 충분한 연산 성능과 저장 공간 확보
2. **소프트웨어 설치**: sBTC 서명자 노드 소프트웨어 및 의존성 설치
3. **키 관리**: 노드 소프트웨어가 비트코인 **개인키/공개키**를 안전하게 생성·보관
4. **노드 등록**: 최초 실행 시 노드는 자신의 **공개키를 스택스의 sBTC 레지스트리 컨트랙트**에 자동 등록

## 일상 운영(Day-to-Day Operations)

셋업이 완료되면, 서명자 노드는 **자율적으로** 다음 작업을 수행합니다:

### 1. 입금 요청 모니터링

서명자 노드는 지속적으로 sBTC 민팅 요청을 모니터링합니다.

1. 비트코인 네트워크와 스택스 블록체인에 연결
2. sBTC UTXO 주소로 전송된 비트코인 트랜잭션 감시
3. 입금이 감지되면 트랜잭션 세부 정보를 검증

---

### 2. 민트 요청 처리

입금을 확인한 뒤:

1. 노드는 개인키로 민트 오퍼레이션에 대한 서명을 준비
2. 해당 서명을 스택스 블록체인의 sBTC 입금 컨트랙트에 제출
3. 컨트랙트는 서명을 검증하고 다른 서명자들의 서명과 조합
4. 충분한 유효 서명이 수집되면, 컨트랙트가 동등한 양의 sBTC를 민팅

---

### 3. 출금 요청 처리

sBTC 출금 요청의 경우:

1. 노드는 sBTC 출금 컨트랙트를 모니터링하여 새로운 요청을 탐지
  2. 사용자의 sBTC 잔액과 요청의 유효성을 검증
  3. 노드는 자동으로 출금 오퍼레이션에 서명하고 서명을 제출
  4. 충분한 서명이 모이고 sBTC가 소각되면, 노드는 비트코인 출금 트랜잭션 생성·서명에 참여
  5. 서명된 비트코인 트랜잭션을 비트코인 네트워크에 브로드캐스트
- 

# sBTC 트랜잭션 워크스루

여기서는 1 BTC가 sBTC 시스템을 거쳐 입금에서 출금까지 이동하는 과정을 따라가 봅니다.

## Part 1: 입금 (BTC → sBTC)

### Step 1: 시작

Alice는 1 BTC를 sBTC로 변환하여 스택스 DeFi에 참여하기로 결정합니다.

1. Alice는 비트코인 네트워크에 입금 트랜잭션을 생성 (일반적으로 sBTC 브리지나 DeFi 앱 UI를 통해)
2. 해당 트랜잭션은 비트코인 mempool에 들어감

### Step 2: 증명 제출

1. Alice는 입금 증명을 Deposit API에 제출 (앱 UI를 통해 자동 처리됨)
2. Deposit API는 상태를 **PENDING**으로 설정

### Step 3: 서명자 검증

sBTC 서명자 집합은:

1. 입금 트랜잭션을 탐지
2. UTXO 포맷 검증
3. 입금에 대한 투표 진행

거부 시:

1. 서명자가 API에 거부 사실을 통지

2. Deposit API는 상태를 **FAILED**로 업데이트

승인 시:

1. Deposit API는 상태를 **ACCEPTED**로 업데이트

## Step 4: 비트코인 트랜잭션

승인되면, 서명자 집합은:

1. Alice의 입금 BTC를 소비하는 새 비트코인 트랜잭션 생성
2. 이를 비트코인 네트워크에 브로드캐스트

트랜잭션 실패 시:

1. 서명자가 API에 실패 통지
2. Deposit API는 상태를 **FAILED**로 업데이트

## Step 5: sBTC 민팅

비트코인 트랜잭션이 성공하면:

1. 서명자 집합은 스택스 블록체인과 상호작용
2. Alice의 스택스 주소에 1 sBTC를 민팅

## Step 6: 확인

1. Deposit API는 상태를 **CONFIRMED**로 업데이트
2. Alice의 스택스 지갑에 1 sBTC가 도착

---

## Part 2: sBTC 사용

Alice는 이제 1 sBTC를 스택스 생태계에서 자유롭게 활용할 수 있습니다:

1. sbtc-token 컨트랙트를 통해 다른 사용자에게 전송
2. 다양한 DeFi 애플리케이션 참여
3. SIP-010 토큰을 지원하는 모든 앱에서 사용

---

## Part 3: 출금 (sBTC → BTC)

## Step 1: 시작

Alice는 1 sBTC를 BTC로 출금하기로 합니다.

1. Alice는 스택스 블록체인의 Clarity 컨트랙트를 호출해 출금 시작
2. 본인 비트코인 주소를 지정
3. 성공 시 sBTC가 잠기고 상태는 **PENDING**
4. 실패 시 출금은 발생하지 않음

## Step 2: 서명자 검증

sBTC 서명자 집합은:

1. 출금 요청 탐지
2. 승인 또는 거부 결정

거부 시:

1. sBTC 잠금 해제
2. 상태는 **FAILED**로 업데이트

승인 시:

1. 상태를 **ACCEPTED**로 업데이트
2. **6 비트코인 블록 컨펌**을 대기

## Step 3: 비트코인 트랜잭션

대기 후 승인 시:

1. 서명자 집합은 출금을 위한 새 비트코인 트랜잭션 생성
2. 이를 비트코인 네트워크에 브로드캐스트

트랜잭션 실패 시:

1. sBTC 잠금 해제
2. 상태를 **FAILED**로 업데이트

## Step 4: sBTC 소각 및 확정

비트코인 트랜잭션 성공 시:

1. 서명자 집합은 스택스에서 Alice의 1 sBTC를 소각

2. 상태를 **CONFIRMED**로 업데이트

## Step 5: 완료

1. Alice는 지정한 비트코인 주소로 1 BTC를 수령
2. 출금된 1 sBTC는 영구적으로 유통에서 제거됨

# sBTC FAQ

## sBTC 기초

### sBTC란 무엇인가요?

sBTC는 스택스 Bitcoin Layer에서 발행되는 **탈중앙화 1:1 비트코인 담보 자산**입니다. 자세한 내용은 [스택스](#) 및 [sBTC](#) 페이지를 참고하세요.

### sBTC는 어떻게 작동하나요?

1. sBTC는 스택스 블록체인의 **SIP-010 토큰**으로, BTC를 1:1로 나타냅니다. 항상 1 sBTC = 1 BTC로 담보됩니다.
2. sBTC 페그 지갑은 sBTC **서명자 집합**이 관리하며, 이를 통해 보안성을 높이고 단일 실패 지점을 줄입니다. 자세한 내용은 [Stacker Signing](#)을 참고하세요.

### Bitcoin Finality란 무엇이며 왜 중요한가요?

스택스 및 sBTC 상태는 비트코인과 자동으로 함께 포크됩니다. 따라서 모든 트랜잭션은 **100% 비트코인 파이널리티**로 정산됩니다. 이는 하드포크 공격으로부터 사용자를 보호하며, sBTC 보안을 비트코인과 정렬시키는 핵심 보안 장치입니다. 자세한 내용은 [스택스 문서](#)를 참고하세요.

### 스택스 서명자 네트워크는 어떻게 보안을 강화하나요?

서명자는 모든 sBTC 입금 및 출금 오퍼레이션을 승인하여 시스템 무결성을 보장합니다. 트랜잭션 승인에는 **70% 합의**가 필요하며, 이를 통해 네트워크의 생존성과 보안이 유지됩니다.

sBTC 출시를 위해 스택스 커뮤니티는 [SIP-028](#)을 승인했으며, 기술 전문성·신뢰성·성능·탈중앙화 등을 기준으로 초기 15명의 기관 서명자가 Phase 1에 선정되었습니다. 이는 단순성과 안정성을 위해 선택된 것이며, 프로토콜이 성숙함에 따라 점차 확대됩니다.

서명자 목록은 [여기](#)에서 확인할 수 있습니다.

### sBTC의 보안을 위해 어떤 조치가 마련되었나요?

- sBTC는 항상 1:1로 BTC에 담보되며, **임계치 암호학(threshold cryptography)**으로 검증 가능합니다.
- 제3의 수탁자나 신뢰 설정 없이, **탈중앙화 서명자 집합**이 BTC를 보관합니다.
- **보안 파트너십:**
  1. [Asymmetric Research](#): 보안 감사 및 리서치 제공
  2. **ImmuneFi**: 버그 바운티 프로그램 운영
  3. **3rd Party Audits**: 여러 보안 감사가 수행되었으며, [sBTC Audits](#)에서 확인 가능

### sBTC의 차별점은 무엇인가요?

- sBTC는 진정한 **비트코인 네이티브** 제품입니다.
- **비트코인 커뮤니티의 주요 리더들**(서명자 네트워크)에 의해 담보됩니다.
- **탈중앙화 검증자/서명자 네트워크**가 보안을 제공하며, 단일 수탁자에 의존하지 않습니다.
- **100% 비트코인 파이널리티**를 활용합니다.
- 사용자 경험(UX)과 개발자 경험(DevEx)을 최적화했습니다.
- 코드가 공개된 **완전한 투명 프로젝트**입니다.

### sBTC 서명자에 대해 더 알아보려면?

"[sBTC Signer Set 선정](#)" 포스트를 참고하세요.

---

## sBTC 리워드 프로그램

### 리워드로 지급되는 BTC는 어디서 나오나요?

sBTC 리워드 프로그램은 특정 보상 주소로 STX를 스테킹하는 **Stackers**에 의해 운영됩니다. 이들이 PoX 합의를 통해 받은 BTC 리워드를 프로그램에 기부하는 구조입니다.

역사적으로 2주 주기 동안 약 **10% APY** 수준의 BTC가 Stacker에게 분배된 사례가 있습니다([Stacking Tracker](#) 참고).

이 BTC는 sBTC 인센티브 풀로 들어가 스마트 컨트랙트를 통해 sBTC로 브릿징되며, **sBTC 보유자에게 비례 분배**됩니다.

**리워드는 어떻게 분배되나요?**

리워드는 2주마다 **비수탁 지갑의 STX 주소**로 자동 분배됩니다.

**리워드를 받으려면 어떻게 해야 하나요?**

[bitcoinismore.org](https://bitcoinismore.org)에서 프로그램에 **등록**해야 합니다.

**sBTC 보유량이 늘어나면 다시 등록해야 하나요?**

아니요. 재등록은 필요 없습니다. 리워드 스마트 컨트랙트가 자동으로 잔액 변화를 반영합니다.

**리워드 수준은 얼마나 되나요?**

리워드는 **리워드 풀의 STX 규모, PoX 수익률, 발행된 sBTC 수량**에 따라 변동됩니다.

**PoX 리워드와 sBTC 리워드 프로그램의 차이는 무엇인가요?**

- **PoX 리워드:** STX를 스테킹한 Stacker가 받는 비트코인 리워드
- **sBTC 리워드:** sBTC 보유자를 위한 추가 BTC 리워드(스테킹 불필요, 초기 유동성 인센티브)

---

## sBTC 사용하기

**sBTC는 언제부터 이용 가능하나요?**

- **2024년 12월 16일:** 1,000 BTC 캡으로 입금 시작
- **2025년 2월 25일:** 3,000 BTC 캡 오픈
- **2025년 4월 30일:** 출금 기능 활성화

서명자 집합의 완전 탈중앙화는 [후속 단계](#)에서 점차 진행됩니다.

**어떤 지갑이 지원되나요?**

- [Xverse](#)
- [Leather](#)
- [Ledger](#)
- [Asigna](#)

추가 지갑 및 기관 수탁자와의 통합도 준비 중입니다.

### 왜 최소 0.01 BTC 입금 제한이 있나요?

**0.01 BTC 최소 입금**은 소액 스팸 트랜잭션을 방지하기 위한 조치입니다.  
향후 제한 완화를 검토 중입니다.

### sBTC 브리지를 이용해 리워드를 받는 단계는?

자세한 [영상](#) 및 [워크스루](#)를 참고하세요.

1. 지원 지갑 준비 (Xverse, Leather, Ledger, Asigna)
2. [app.stacks.co](https://app.stacks.co) 접속 → 지갑 연결
3. 변환할 BTC 금액 입력
4. 스택스 수령 주소 입력 후 전송
5. 민팅 완료 후 [bitcoinismore.org](https://bitcoinismore.org)에서 보상 등록
6. 2주마다 자동 BTC 리워드 수령

### 입금 확인 시간은 얼마나 걸리나요?

- **BTC → sBTC**: 3 비트코인 블록 내 (~1시간 미만)
- **sBTC → BTC**: 6 비트코인 블록 내 (~2시간)

자세한 내용은 [스택스 문서](#)를 참고하세요.

### 왜 전체 BTC 폐깅에 상한선(cap)이 있나요?

보안 중심의 점진적 롤아웃을 위해 **BTC 상한선**이 설정되었습니다.  
또한 상한선은 UX와 DeFi 통합 개선 시간을 제공합니다.

### sBTC 민팅에 수수료가 있나요?

있습니다.

1. 사용자가 입금 트랜잭션 생성 시 설정하는 수수료
2. 예치된 UTXO를 단일 서명자 UTXO로 합치기 위한 수수료(최대 80,000 sats, 민팅된 sBTC에서 자동 차감)

### sBTC도 멀티시그 솔루션을 지원하나요?

네. [Asigna](#)가 멀티시그 솔루션을 제공합니다.

## sBTC 수탁자(Custodian) 지원이 있나요?

현재는 없습니다. 그러나 기관 수탁자들과의 지원 협업이 진행 중입니다.  
Copper, BitGo는 이미 스택스 및 스테킹을 지원하며, sBTC 통합이 우선순위에 있습니다.

---

## sBTC 문제 해결

**BTC 입금이 확인되었는데 sBTC가 지갑에 안 보여요.**

지갑 내 '**토큰 관리(Manage Tokens)**' 메뉴에서 sBTC 표시를 활성화해야 할 수 있습니다.

**Xverse 지갑 사용 시 "Errors.Invalid\_Transaction" 에러가 발생해요.**

이는 "Nested SegWit" 지갑을 사용 중일 때 발생할 수 있습니다.  
해결 방법: Xverse 지갑을 **Native SegWit** 모드로 변경하세요.

**3 블록이 지났는데도 sBTC가 지갑에 안 보여요.**

BTC → sBTC 전환은 보통 **3 블록 내 완료**됩니다. 다만 블록 속도에 따라 최대 **2시간**이 걸릴 수 있으며, Leather 또는 Xverse 지갑에서는 추가로 20분 정도 지연될 수 있습니다.

**리워드 프로그램 등록 확인 메시지를 못 받았어요.**

[bitcoinismore.org](https://bitcoinismore.org)에 접속 → 지갑 연결 → "enrolled" 표시 여부 확인

# Gaia

스택스 블록체인으로 만든 앱은 오프체인 저장소 시스템 **Gaia**를 사용해 데이터를 저장할 수 있습니다.

Gaia는 **온체인 불변성**보다 **사용자 데이터 소유권**에 초점을 맞춘 탈중앙 스토리지 접근 방식입니다. 핵심은 **사용자 통제**입니다.

IPFS·Arweave 같은 온체인(또는 영구) 저장은 불변성과 검열 저항성을 제공하지만, 한 번 배포된 데이터를 사용자가 수정/삭제할 수 없기 때문에 **사용자 통제**를 제공한다고 보긴 어렵습니다.

Gaia는 다른 문제를 푼다. 사용자가 **저장 위치를 스스로 선택**하고, 그 데이터 접근을 **온체인 스택스 ID**에 연결할 수 있게 합니다.

공개 트랜잭션 메타데이터는 스택스 블록체인에 저장하는 편이 좋지만, **사용자 앱 데이터**는 Gaia에 저장하는 것이 더 효율적이고 프라이버시에도 유리할 때가 많습니다.

블록체인 밖에 데이터를 저장하면, 중앙 신뢰 주체 없이도 **고성능·고가용성**의 읽기/쓰기 경험을 제공할 수 있습니다.

---

## 스택스 아키텍처에서 Gaia 이해하기

아래는 스택스 아키텍처와 그 안에서 Gaia의 위치를 설명합니다:

블록체인은 다수의 합의를 필요로 하기에 **느릴 수** 있고, 많은 데이터를 담도록 설계되지 않았습니다. 따라서 사용자가 기록하는 모든 데이터를 온체인에 올리면 **비용**이 큼니다. (예: 모든 트윗을 체인에 올린다고 가정)

스택스는 **레이어드 접근**으로 성능을 해결합니다. 베이스 레이어에는 스택스 블록체인과 **BNS(Blockchain Naming System)**가 있습니다. 블록체인은 스택스 네트워크의 **아이덴티티 소유권**을 관리합니다. 아이덴티티는 도메인/사용자명/앱 이름 등이 될 수 있습니다.

아이덴티티가 생성되면 그 사실이 블록체인에 기록됩니다. 이 아이덴티티 정보는 OSI 스택의 **라우팅 데이터**에 해당하며, 라우팅 데이터는 두 번째 레이어인 **Atlas 피어 네트워크**에 저장됩니다. 스택스의 모든 코어 노드는 이 라우팅 데이터를 전체 복제합니다. 이 라우팅 데이터는 아이덴티티(도메인/사용자/앱 이름)를 최종 레이어, 즉 **Gaia Storage System**의 특정 저장 위치와 연결하는 데 사용됩니다.

**Gaia Storage System**은 **허브 서비스**와 클라우드 스토리지 리소스로 구성됩니다. 스토리지는 Azure/DigitalOcean/AWS EC2 등 어떤 상용 제공자든 가능합니다. 보통 컴퓨트와 스토리지를 같은 클라우드에 두지만 필수는 아닙니다. Gaia는 기본적으로 **S3, Azure Blob, GCP, 로컬 디스크** 드라이버를 지원하며, 드라이버 모델을 통해 다른 백엔드도 확장 가능합니다.

Gaia는 **단순 키-값 저장소**로 데이터를 저장합니다. 아이덴티티가 생성되면 Gaia에 그 아이덴티티에 대응하는 데이터 스토어가 연결됩니다. 사용자가 dApp에 로그인하면, 인증 과정에서 **Gaia 허브 URL**이 앱에 전달되고, 앱은 그 허브를 통해 사용자의 데이터를 저장합니다.

스택스 블록체인은 **아이덴티티 데이터만** 저장합니다. 아이덴티티가 수행한 행동으로 생성된 데이터는 Gaia에 저장됩니다. 각 사용자에게는 프로필 데이터가 있고, dApp은 사용자를 대신하여 애플리케이션 데이터를 저장합니다. 데이터가 오프체인에 있으므로, 스택스 dApp은 타 블록체인 dApp보다 **더 높은 성능**을 제공하는 경우가 많습니다.

---

## 사용자 통제 — Gaia는 어떻게 탈중앙화되나요?

Gaia 허브는 스토리지에 쓰기 작업을 수행하는 **서비스**로 동작합니다. 스토리지는 단순 키-값 저장소이며, 허브는 **유효한 인증 토큰**이 있는 요청만 쓰기를 허용합니다. 허브(컴퓨트)와 스토리지는 별도 리소스로 운영되며, 보통 같은 클라우드 제공자 내에 있습니다.

Gaia의 탈중앙화는 **사용자 데이터와 저장 위치에 대한 선택권**에 초점을 둡니다. 사용자는 **원하는 Gaia 허브 제공자를 선택**할 수 있습니다. 이 선택권 자체가 사용자 통제형 애플리케이션에 필요한 탈중앙화의 핵심입니다. 또한 Gaia는 앱이 데이터에 접근하는 **표준 API**를 정의합니다.

데이터 통제는 **접근 방식**에서 구현됩니다. 예를 들어 앱이 사용자 `alice.id`의 `data.txt`를 가져올 때의 순서는 다음과 같습니다:

1. `alice.id`의 **zonefile**을 가져온다.
2. zonefile에서 **프로필 URL**을 읽는다.
3. Alice의 **프로필**을 가져온다.
4. 그 프로필이 `alice.id`의 키로 **서명**되었는지 **검증**한다.
5. 프로필의 `appsMeta` 섹션에서 **읽기 전용 URL**(예: `https://example-app.gaia.alice.org`)을 찾는다.
6. `https://example-app.gaia.alice.org/data.txt`에서 파일을 가져온다.

`alice.id`는 자신의 **zonefile**에 접근할 수 있으므로 **프로필 저장 위치를 변경**할 수 있습니다(예: 기존 제공사토리지가 침해된 경우). 프로필 저장 위치를 바꾸려면 **Gaia 허브 URL을 다른 허브로 변경**하면 됩니다. 충분한 리소스가 있다면 사용자는 **직접 Gaia 스토리지 시스템을 운영**하여 상용 허브를 완전히 우회할 수도 있습니다.

 **경고:** 기존 아이덴티티를 가진 사용자는 아직 **허브 간 데이터 마이그레이션**을 지원받지 못합니다.

앱이 `alice.id`를 **대상으로 직접 쓰기**를 수행할 경우에는 위의 조회 절차가 필요 없습니다. 대신, 스택스 인증 플로우가 **Alice가 선택한 Gaia 허브 URL**을 앱에 제공합니다. 이 인증 플로우 또한 **Alice의 지갑이 인증 응답을 생성**해야 하므로 Alice의 통제 하에 있습니다.

---

## 데이터 저장 방식 이해하기

Gaia 허브는 **전달받은 그대로** 데이터를 저장합니다. 데이터 형식의 유효성·서명 유효성·암호화 여부 등은 보장하지 않습니다. 이러한 사항은 **클라이언트 측의 책임**이라는 철학을 따릅니다.

클라이언트 라이브러리(예: Stacks.js)가 이러한 보장을 제공할 수 있습니다. 이는 end-to-end 원칙의 확장된 해석에 기반합니다.

앱이 Gaia 허브에 쓰기를 수행할 때는 **인증 토큰, 키, 데이터를 함께 전달**합니다. 토큰은 앱이 **사용자를 대신 해 쓰기 권한**을 가졌음을 보장합니다.

---

## Gaia vs. 기타 스토리지 시스템

아래는 다른 탈중앙 스토리지와 비교입니다(공통 기능은 생략).

| 기능                                 | <u>Gaia</u> | <u>Sia</u> | <u>Storj</u> | <u>IPFS</u> | <u>DAT</u> | <u>SSB</u> |
|------------------------------------|-------------|------------|--------------|-------------|------------|------------|
| 사용자가 <b>호스팅 위치</b> 를 선택/통제         | X           |            |              |             |            |            |
| 일반 <b>웹 브라우저</b> 로 데이터 조회 가능       | X           |            |              | X           |            |            |
| <b>읽기/쓰기</b> 가능                    | X           |            |              |             | X          | X          |
| 데이터 <b>삭제</b> 가능                   | X           |            |              |             | X          | X          |
| 데이터 <b>리스트업</b> 가능                 | X           | X          | X            |             | X          | X          |
| 삭제된 공간 <b>회수</b>                   | X           | X          | X            | X           |            |            |
| <b>예측 가능한 성능</b> 의 조회              | X           |            | X            |             |            |            |
| <b>쓰기 권한 위임</b> 가능                 | X           |            |              |             |            |            |
| <b>리스트 권한 위임</b> 가능                | X           |            |              |             |            |            |
| <b>멀티 백엔드</b> 네이티브 지원              | X           |            | X            |             |            |            |
| <b>전역 주소 지정</b> 지원                 | X           | X          | X            | X           | X          |            |
| <b>암호화폐</b> 필요                     |             | X          | X            |             |            |            |
| <b>콘텐츠 주소화</b> (content-addressed) |             | X          | X            | X           | X          | X          |