비스무트 암호화폐 백서

저자: 비스무트 코어 개발팀

역자: heavens

버전: 1.0

출간일: 2019년 4월 3일

목차

초록 ...

개요 ...

비스무트의 주요 특징 ...

활용 분야 ...

코인 공급 및 보상 모델 ...

암호학 ...

채굴 알고리즘 ...

롱테일 블록타임의 방지 ...

오퍼레이션과 데이터 필드 ...

프라이빗 컨트랙트 ...

하이퍼노드와 사이드체인 ...

하이퍼블록 압축 ...

페널티 시스템 ...

테스트넷 ...

레그넷 ...

교육과 연구 ...

향후 전망 ...

요약 ...

Disclaimer …

초록

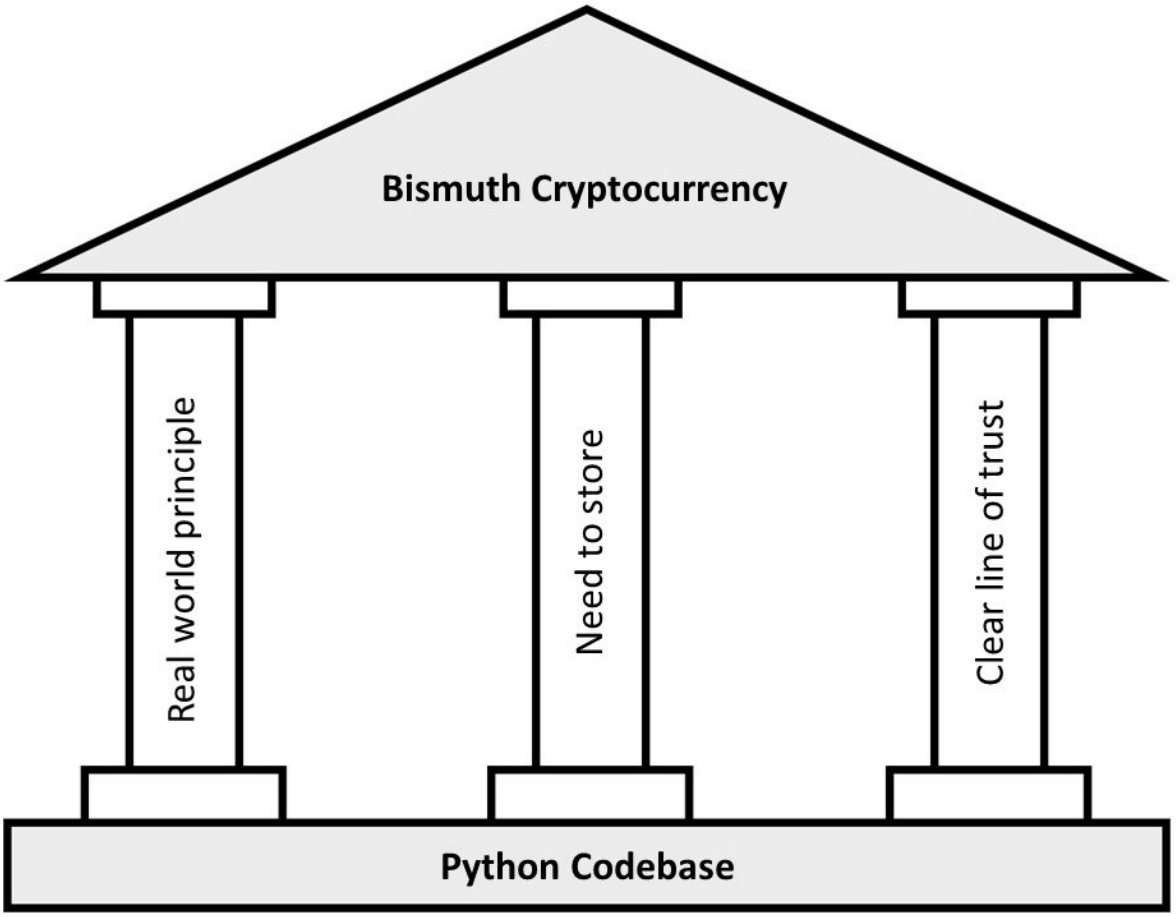
비스무트 암호화폐의 백서입니다. 이 백서는 비스무트의 기본 철학, 3대 원칙 그리고 핵심 특징들을 소개합니다. 주요 내용으로는 플러그인(Plugins), 응용 분야, 코인 공급과 보상 모델, 암호학, 채굴 알고리즘, 롱테일 블록타임 방지, 오퍼레이션(Operation)과 데이터(Data) 필드, 프라이빗 컨트랙트(Private Contracts), 하이퍼노드(Hypernodes)와 사이드체인, 테스트넷, 레그넷, 교육과 개발 등을 담고 있습니다.

개요

비스무트의 목표는 가급적 단순한 블록체인을 파이썬 프로그래밍 언어를 이용해 완전히 새롭게 구축하는 것이었습니다. 프로그래머 한 명이 블록체인 기술을 익힐 목적으로 시작한 1인 프로젝트였습니다. 그러나 이내 매우 풍부한 기능을 가진 암호화폐, 블록체인 플랫폼으로 성장했습니다. 다수의 개발자, 기술 인력, 채굴 풀 운영자, 거래소와 소셜미디어 관계자들이 이 프로젝트에 동참했습니다.

비스무트의 핵심 원칙

비스무트는 다른 암호화폐와 구별되는 3대 개발 원칙을 가지고 있습니다.



1. 현실주의 원칙(Real world principle)

일부 블록체인들은 이상주의적 세상에 존재합니다. 그 세상은 모든 것이 완벽하게 만들어져 있습니다. 프로그래밍 코드는 모든 문제를 해결할 수 있고, 이용자는 모두 복잡한 알고리즘을 이해하는 전문가들입니다. 이 이용자는 알고리즘을 완벽하게 구현할 수 있으며, 소스코드를 직접 컴파일 할 수 있고, 각종 소프트웨어 툴을 에러 없이 사용합니다.
이런 가정 하에 설계된 블록체인은 실패할 수밖에 없습니다. 현실, 현실 속 유저, 현실적인 이용 사례를 고려하지 않았기 때문입니다. 디지털 소프트웨어와 현실 세계의 연결(유저, 오라클, 네트워크 등)은 결코 완벽하지 않습니다. 비스무트는 이 문제를 해결해야 합니다.
비가역성(immutability)을 위해 엄청나게 복잡한 알고리즘이 반드시 필요한 것은 아닙니다. 에러가 없는 완전무결한 데이터 보안 시스템을 원하는 것이 아니라면 말입니다. 코드의 복잡도는 현실의 사용 목적에 따라 달라져야 합니다. 승용차를 달리게 하기 위해 특수한 원자로를 설계할 필요는 없습니다. 일반 내연기관이면 충분합니다. 이것이 일상적 사용 목적에 더 부합합니다.

비스무트는 (다른 블록체인과 달리) 트랜잭션 해시를 사용하지 않습니다. 블록 해시만으로도 그 안에 담긴 모든 트랜잭션의 검증이 가능하기 때문입니다. 다음 코드는 이를 잘 보여줍니다.

```
block_hash = hashlib.sha224((str(transaction_list_converted) +
db_block_hash_prev).encode("utf-8")).hexdigest()
```

이것이 비스무트의 철학입니다. 단순하고, 이해하기 쉬우면서도, 효과적이고, 충분히 훌륭한 시스템을 만들자는 것입니다.
암호화폐가 대중적으로 널리 쓰이기 위해서는 개발자나 암호학 전문가만을 위한 무엇에서 벗어나 더 나아가야 합니다.
이를 위해 비스무트는 블록체인 시스템을 가급적 단순하고 이해하기 쉽게 설계하고자 합니다. 개발을 위한 개발은 하지 않습니다. 아직 할 일이 많지만, 방향은 분명합니다.
완벽한 추상적 시스템을 먼저 만든 뒤 현실을 이 시스템에 맞추려 하지 않습니다. 애초부터 구체적인 현실의 문제를 풀기 위한 가장 단순하면서도 충분한 해법을 찾는 방식으로 개발을 진행합니다.

2. 필수적인 데이터만 블록체인에 저장(Need to store)

현실주의 원칙을 충족하려면 필수적인 데이터가 블록체인에 저장돼야 합니다. 비스무트 이용자는 이를 위해, BTC나 ETH 이용자와 달리, 꼼수를 쓸 필요가 없습니다. 비스무트 블록체인은 두 개의 추상적인 데이터 필드를 기본으로 제공합니다.
‘오퍼레이션(operation)’과 ‘데이터(data)’ 필드가 그것입니다. 유저는 이 두 개의 데이터 필드를 이용해 어떤 프로토콜, 아무리 복잡한 프로토콜이라도, 비스무트 블록체인 위에 구축할 수 있습니다. 이 방식은 복잡한 프로토콜의 작동 성능을 원만하게 보장한다는 장점도 있습니다.
모든 데이터를 블록체인에 저장하고 싶은 유혹을 경계해야 합니다. 망치를 손에 들면, 눈에 보이는 못이란 못은 모두 박아 넣고 싶어집니다. 많은 암호화폐들이 데이터 처리용량 확대(scalability)에 골몰하는 것도 이 때문입니다. 모든 데이터를 블록체인에 저장하려는 것은 너무 과한 욕심입니다.
비스무트의 철학은 필수적인 데이터만 블록체인에 저장하자는 것입니다. 데이터 처리용량 문제는 시스템 설계의 문제입니다. 애초부터 필수적인 데이터만 저장하도록 설계하면, 처리용량 문제의 상당 부분을 미리 막을 수 있습니다. 왜 문서 전체를 블록체인에 저장해야 합니까? 해당 문서의 해시값만 저장해도 되는데 말입니다. 데이터 전체 대신 해당 데이터의 증거(proofs)만 저장합시다. 체크포인트(checkpoints), 전자서명(signatures), 핑거프린트(fingerprints)를 적극적으로 활용합시다.

비스무트는 기본 블록체인 이외에도 사이드체인과 같은 실용적인 프레임워크를 활용해 더 많은 데이터를 안전하게 저장할 수 있는 방법을 지원합니다. 비스무트 하이퍼노드가 그 실증 사례입니다.

3. 신뢰 대상과 범위의 명확화(Clear line of trust)

많은 암호화폐는 “신뢰가 필요없는(trustless)” 시스템이라고 주장합니다. 그러나 이는 현실주의 원칙에 부합하지 않습니다.
블록체인에 저장됐다고 해서 그 데이터가 반드시 “참(true)”은 아닙니다. 블록체인이 보장하는 것은 저장된 데이터가 위변조되지 않는다는 것 뿐입니다. 이 역시 이면의 신뢰(trust)를 전제로 합니다. 소스코드를 믿어야 하고, 초기화 과정의 데이터를 믿어야 하고, 기초 알고리즘과 다른 피어(peers), 서비스 제공자를 믿어야 합니다.
스마트컨트랙트는 마법이 아니며 완벽하지도 않습니다. 수많은 버그가 있고, 가상 머신(Virtual Machine)의 코드 변경이 수시로 필요하고, 오라클 등 외부 데이터 제공자를 믿어야 합니다. 여전히 신뢰가 필요하지만, 무엇을 누구를 믿어야 할지 알 수 없습니다.
비스무트는 이런 사실을 숨기지 않습니다. 우리는 블록체인 역시 신뢰(trust)가 필요할 때가 있다는 사실을 분명히 합니다. Zircodice나 Dragginator 같은 미스무트 컨트랙트들은 이 서비스 운영자를 믿어야 합니다. 다만, 이 서비스의 트랜잭션 이력을 확인할 수 있고 약속한 대로 작동했는지 검증할 수 있습니다.

비스무트의 가치

비스무트의 가치 프로포지션은 다음으로 요약할 수 있습니다.
- 가볍다: 비스무트의 노드는 가볍습니다. 고성능 CPU나 대용량 메모리가 필요 없습니다.
- 블록체인 연구, 개발, 학술적 분석에 적합합니다.
- 소스코드는 다루기 쉽고 신속히 개발할 수 있습니다.
- 각용 응용 사례의 프로토타입을 빠르게 실험할 수 있습니다.
- 다층적인 API와 라이브러리를 여러 프로그래밍 언어로 이용할 수 있습니다.

파이썬 활용과 개발 편리성

비스무트는 늘 아래 목표를 추구하면서 개발됐습니다.
- 단순화: 여타 블록체인의 복잡한 속성을 그대로 카피하지 않았습니다. 핵심 기능부터 복잡성을 줄이고 단순화했기 때문에 이해하기 쉽습니다.

- 혁신성: (부가 기능을 추가할 수 있는) 비스무트 프로토콜을 코어에 포함시켰습니다. 새로운 기능을 유연하고 신속하게 추가하거나 테스트할 수 있습니다.
- 확장성이 매우 높고 다양한 변형이 가능합니다.

비스무트는 개발자를 위해 고안됐습니다.
- 인기있는 프로그래밍 언어인 파이썬으로 개발돼 점점 더 많은 개발자가 개발에 참여할 수 있습니다.
- 파이썬은 대학, 과학계, 인공지능 및 데이터 사이언스 연구자, 학생들이 선호하는 언어입니다.
- 파이썬은 소스코드를 컴파일 할 필요가 없습니다. 변형하거나 테스트하기 쉽습니다.
- 비스무트 시스템은 다층적으로 접근할 수 있습니다. SQL 언어를 이용해 데이터베이스를 직접 액세스할 수도, 각종 프로그래밍 언어로 작성된 API를 이용할 수도 있습니다.
- 비스무트 노드는 소프트웨어 개발에 필요한 각종 연결 고리와 필터를 다양하게 제공합니다. 플러그인을 개발하기 쉽습니다. 응용 소프트웨어와 플러그인 개발은 모두 파이썬으로 가능합니다. 다른 프로그래밍 언어를 배우지 않아도 됩니다.
- 토네이도 지갑은 '크리스탈'이라는 이름의 플러그인 기능을 내장하고 있습니다. 지갑과 연동되는 dApps 개발이 매우 쉽습니다.
- 비스무트의 추상적(Abstract) 트랜잭션과 프로토콜 개념은 상상할 수 있는 모든 종류의 응용 소프트웨어를 블록체인과 연동할 수 있게 해줍니다.

한 블록체인 개발자는 비스무트 개발팀에게 이렇게 말했습니다.
"(비스무트로 개발을 해보니) 파이썬 언어의 단순성 덕분에, 불과 몇 시간만에 모델 소프트웨어(proof of concept)를 완성하는 것이 가능했다. 대부분의 개발 시간을 (코딩이 아니라) 블록체인 연구 자체에 집중할 수 있었다. 반면, 이더리움의 솔리더티(solidity) 개발은 단순한 모델 개발도 지나치게 복잡해지는 경우가 많았다."
개발에 동참할 분들은 개발자를 위한 가이드인 'Hack with Bis'를 먼저 읽어보시기 바랍니다. 깃허브 링크.

비스무트 코드 실행 모델

비스무트의 코드 실행 모델은 이더리움과 매우 다릅니다. 이더리움 솔리더티로 개발한 것을 그대로 옮길 수 있다고 생각하시면 안됩니다. 비스무트는 현재 '스마트' 컨트랙트가 필요 없습니다.
비스무트에는 모든 노드가 똑같은 코드를 실행하는 가상머신(VM)이 존재하지 않습니다. 약점으로 비칠 수도 있지만, 실제로는 강점입니다. ETH 스마트 컨트랙트들이 받았던 각종 해킹 공격들은 비스무트 같은 설계 구조에는 통하지 않습니다.

‘스마트’ 컨트랙트 vs ‘스마트’ 프로토콜

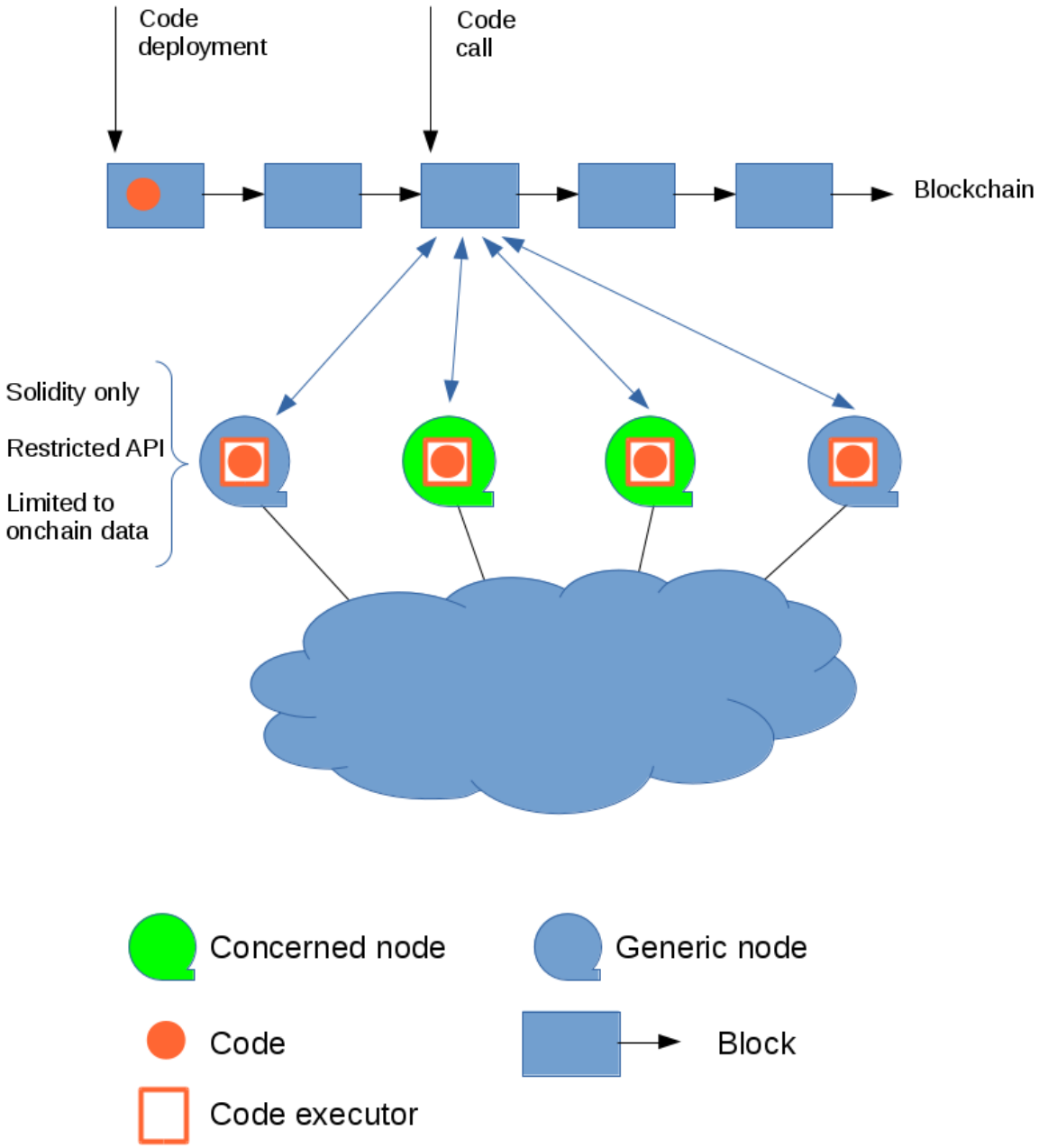
ETH 방식의 스마트 컨트랙트는 특정한 언어(솔리더티)로 작성되고, 블록체인 내부에 저장되며, 모든 노드에서 실행됩니다. 비스무트 방식의 스마트 프로토콜은, 비스무트 블록체인 내부가 아니라, 별도의 층위(layer)에서 구현되며, (모든 노드가 아니라) 해당되는 dApps에서만 작동됩니다.

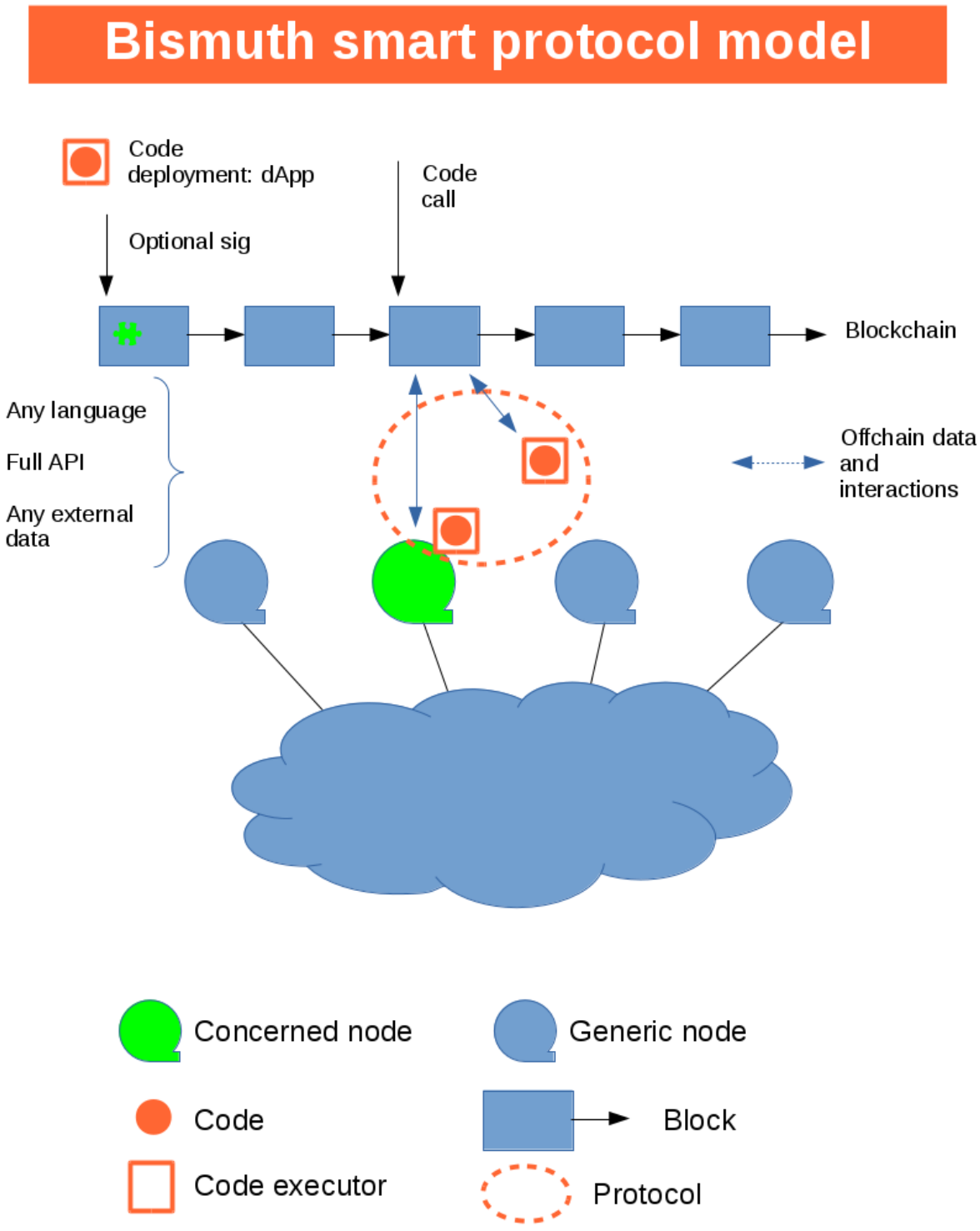
이더리움

- 새로운 프로그래밍 언어인 솔리더티를 배워야 합니다.
- 특정한 약점들이 존재합니다. 언더플로우(underflow), 가시성(visibility), 접근 권한(access rights) 등.
- 잘못 작성된 스마트 컨트랙트는 해커가 무한정 코인을 빼낼 수 있게 길을 열어줄 수 있습니다.
- 이더리움 스마트 컨트랙트는 여러차례 해킹 위험에 노출된 전력이 있습니다.
- 스마트 컨트랙트는 코인을 ‘소유할’ 수 있습니다.
- 스마트 컨트랙트는 블록체인에 영구히 존속합니다. 개발자가 킬 스위치를 만들지 않았다면, 멈추는 것이 불가능할 수 있습니다.
- 만약 킬 스위치가 존재한다면, 이 킬 스위치의 소유자는 컨트랙트가 보유한 모든 코인을 가져갈 수 있습니다.
- 각각의 스마트 컨트랙트는 모든 이더리움 노드에서 실행되며, 가상 머신의 시스템 자원(gas)을 소비합니다.
- 스마트 컨트랙트는 블록체인 외부 데이터를 직접 액세스 할 수 없습니다.

이더리움 모델에는 비스무트에는 없는, 비스무트로 구현할 수 없는, 일부 장점과 일부 응용 사례가 있습니다. 그러나 비스무트는 그밖의 응용 사례에서 이더리움보다 더 유리합니다.

Eth smart contract model





비스무트

- 새로운 언어를 배울 필요가 없다. 파이썬이 기본 프로그래밍 언어이지만, 자바, C++ 등 어떤 언어로도 비스무트 컨트랙트와 프로토콜을 코딩할 수 있습니다.

- 소프트웨어 코딩에 내재한 일반적 문제점 이외의 특수한 약점이 없습니다.
- 컨트랙트의 과도한 낭비가 없습니다.
- 가상머신(VM), 블록체인 안에 저장되는 소스코드, 퍼블릭(모든 노드에서 실행되는) 컨트랙트가 없습니다.
- 유저가 프라이빗(관련된 노드에서만 실행되는) 컨트랙트를 구현해 작동할 수 있습니다.
- 컨트랙트의 소유자는 완전한 통제 권한을 가집니다. 컨트랙트의 코드를 수정하거나 업그레이드할 수 있습니다.
- 컨트랙트는, 소유자가 내부 작동 매커니즘을 공개할 경우, 부정 행위 여부를 검사하거나 검증할 수 있습니다.
- 프라이빗 컨트랙트는 블록체인뿐 아니라 외부 데이터에 대한 완전한 접근이 가능합니다. 오라클 등 블록체인에 외부 데이터를 공급하는 프로바이더에 의존할 필요가 없습니다.

토큰

토큰처럼 광범위하게 쓰이는 컨트랙트는 가상머신(VM)이나 유저 코드로 다루지 않습니다. 많이 쓰이는 컨트랙트는 비스무트 코어 코드에 포함될 수 있습니다. 비스무트 개발팀은 엉덩이가 무겁지 않습니다. 비트코인이나 ETH와 달리 새로운 기능을 재빨리 코어 소프트웨에 추가할 수 있습니다.
토큰이 대표적인 사례입니다.
- 비스무트는 토큰 기능을 내장하고 있습니다.
- 비스무트 토큰은 시스템 자원 면에서 최적화 돼 있으며, 데이터베이스 인덱스 기능을 제공합니다.
- 코드가 공개돼 있고 누구나 이용할 수 있습니다. 버그는 공개적인 방식으로 발견되고 바로 바로잡습니다.

ETH의 용어를 빌어 표현하자면, 비스무트 토큰은 현재 ERC20 표준을 부분적으로 준수합니다. 권한 위임은 허용하지 않습니다. 즉 다른 사람에게 토큰의 지출과 승인을 위임할 수 없습니다. 향후 더 많은 기능이 추가될 수 있습니다.

‘스마트’ 프로토콜

비스무트는 ‘스마트’ 컨트랙트 대신 ‘스마트’ 프로토콜 개념을 선호합니다. ‘스마트’ 컨트랙트와 달리 코드를 불변의 블록체인에 저장할 필요가 없습니다. 코드가 모든 노드에서 실행되거나, 파기되거나, 블록체인 속 미아가 될 필요도 없습니다.

스마트에 인용부호를 붙인 까닭은 블록체인 상의 컨트랙트나 프로토콜은 결코 '스마트' 하지 않기 때문입니다. 스마트 컨트랙트는 단순한 코드일 뿐입니다. 이 코드를 작성한 개발자가 얼마나 스마트한지 혹은 멍청한지에따라 스마트한 수준이 달라집니다.
비스무트 프로토콜은 블록체인 트랜잭션에 담긴 추상적 데이터에 의해 만들어집니다. 이 데이터가 무슨 의미인지, 그에 따라 무엇을 할 지 둘 이상의 참여자가 합의한 바가 프로토콜입니다.
- 특정 프로토콜에 관계된 클라이언트만 해당 데이터를 읽고 코드를 실행합니다. (이더리움처럼) 모든 노드가 모든 코드를 실행하지는 않습니다.
- 코드는 블록체인에 저장하지 않습니다. 코드는 블록체인과 상관없이 수정되거나 업데이트할 수 있습니다. 블록체인에 부담을 주거나, 여타 노드의 자원을 소비하지도 않습니다.
- 코드는 이를 합의한 주체들 간의 '계약(contract)'입니다. 그 로직(logic)은 일반적으로 공개하는 것이 바람직합니다.
- 누구나 블록체인에 기록된 데이터를 이용해 해당 로직을 실행하고, 각각의 참여자들이 약속(contract)을 지켰는지 검증할 수 있습니다.
- 프로토콜은 업그레이드하거나 중첩해 사용할 수 있고, 더 발전된 프로토콜의 기초로 활용할 수 있습니다.
- 프로토콜은 다른 프로토콜들을 사용할 수 있습니다. 예컨대, 하나의 프로토콜은 유효한 하위 응용 프로토콜들(valid implementations)을 블록체인 해시를 이용해 정의할 수 있습니다.
비스무트 프로토콜의 기존 응용 사례는 다음 링크를 참고하기 바랍니다. 깃허브 링크

비스무트의 주요 특징

이 섹션은 비스무트의 주요 특징과 기능에 대한 설명을 답고 있습니다.

파이썬과 플러그인

우리는 비스무트 노드 위에 dApps을 구축하려는 개발자들에게 '플러그인(plugin)' 기능을 적극적으로 활용하도록 권장합니다. 각종 플러그인은 ~/Bismuth/plugins 디렉토리에 저장됩니다. 비스무트의 플러그인 시스템은, 비록 짤막한 코드로 작성하더라도, 블록체인의

중요 이벤트에 ‘액션(action)’, ‘필터(filter)’, ‘후크(hooks)’ 등의 방법으로 영향을 줄 수 있습니다. 이를 이용하면, 매우 손쉽게 새로운 기능을 추가할 수 있습니다.

예컨대, ‘블록(block)’ 액션 후크를 구현하는 플러그인을 만든다고 가정합시다. 방법은 간단합니다. 펑션(function)을 아래처럼 정의하면 됩니다.

plugins/900_test/__init__.py:

```
def action_block(block):
    print(block)
```

더 자세한 정보는 Bismuth plugins 를 참고하시기 바랍니다. 링크.

새로운 플러그인을 활성화 하려면 , 비스무트 노드(node.py)를 재실행해야 합니다.

활용 분야

1. 블록체인 실험용

블록체인을 학습하려는 사람은 한 대의 PC만으로도 자신만의 프라이빗 블록체인을 간단히 만들 수 있습니다. PC에 가상 머신(Virtual Machine)들을 설치한 뒤, 각각의 머신에서 비스무트 노드를 작동시킵니다. 기본으로 설정된 포트 번호(5658)는 임의의 다른 번호로 변경해 줍니다. 아울러 이 노드들의 내부 IP 주소(10.0.x.x 또는 192.168.x.x 등)를 비스무트의 설정 파일(config.txt)에 화이트리스트(whitelist)로 입력합니다. 이렇게 하면, 외부 세계와 단절된, 내부 실험용 네트워크가 만들어집니다. 학습자는 이 프라이빗 네트워크를 이용해, 새로운 기능을 실험하거나 dApps을 개발할 수 있습니다. 연구 실험용 네트워크를 만들기 위해 메인넷이나 테스트넷을 하드포크 할 필요는 없습니다.

2. 차일드(Child) 체인

차일드 블록체인을 활용하면, 네트워크 처리용량 증대 및 새로운 기능 추가가 가능합니다. 우선 새로운 속성(블록타임, 프라이빗 또는 퍼블릭 속성 변경 등)을 가진 차일드 체인을 만들어야 합니다. 차일드 체인은 기존 메인넷(POW 체인)의 제약을 받지 않습니다. 비스무트 하이퍼노드와 같은 POS(Proof of Stake) 블록체인이 POW 체인 위에서 작동할 수

있습니다. 개발자는 자신의 앱(app)을 위한 특수한 셋팅과 트랜잭션 유형을 가진 차일드 체인을 구축할 수 있습니다. 이 블록체인은 화폐 기능을 가질 수도 화폐 기능이 없는 데이터 처리 전용일 수도 있습니다.

3. 이벤트 소싱

이벤트 소싱은 외부 세계의 이벤트를 저장하는 객체/데이터 모델입니다. 이벤트 소싱은 주로 블록체인 자체의 현재 스테이트(state)보다 객체의 스테이트(state)에 영향을 줍니다. 다음 링크를 클릭하면, 비스무트 이벤트 소싱의 개념 모델(proof of concept)과 샘플 dApps를 보실 수 있습니다. 링크.

이벤트 소싱은 프라이빗 차일드 체인과 궁합이 잘 맞습니다. 예컨대, 특정한 데이터베이스를 공유하는 고객, 서비스 공급자, 협력 업체의 네트워크가 있다고 가정합시다. 이 네트워크의 데이터는 상품의 위치, 배송 정보, 청구서 등입니다. 이들은 비스무트의 하이퍼노드와 유사한 차일드 체인을 미리 등록한 주체만 참여할 수 있는 프라이빗 블록체인으로 구축합니다. 이 경우, 이벤트 소싱의 개념 모델이 유용할 수 있습니다. 각각의 행위자는 분산 관리되는 데이터베이스를 공유하고, 각각의 데이터 변경은 이벤트가 됩니다. 이 이벤트는 위변조가 불가능하도록 타임스탬프(trimestamp)가 찍히고, 이벤트의 소스와 통제 권한이 분명하며, 모든 과정에 대한 감사(audit)가 가능합니다.

4. 파일 핑거프린팅

비스무트의 레거시 지갑(wallet.py)은 1개 또는 다수 파일의 핑거프린프를 생성하는 기능을 가지고 있습니다. 코드는 다음과 같습니다.

```
def fingerprint():
    root.filename = filedialog.askopenfilename(multiple=True,
        initialdir="", title="Select files for fingerprinting")
    dict = {}
    for file in root.filename:
        with open(file, 'rb') as fp:
            data = hashlib.blake2b(fp.read()).hexdigest()
            dict[os.path.split(file)[-1]] = data
```

```
openfield.insert(INSERT, dict)
```

해당 파일(또는 파일들)의 해시값은 트랜잭션의 ‘Data(추가자료)’ 필드에 삽입돼 자신 또는 타인의 지갑 주소로 전송될 수 있습니다. 해당 파일을 (별도로) 다운로드 받은 사람은 블록체인에 기록된 핑거프린트(해시값)를 이용해 원본 파일과 동일한지 위변조되지는 않았는지 검사할 수 있습니다.

아래 리스트는 이 비스무트 핑거프린팅 기능을 사용하는 게임 또는 dApps 입니다.

- anon.py, 프라이빗 컨트랙트 익명화 서비스
- Dragginator, 디지털 공룡 수집 게임
- PokaPoka, 비스무트 토큰을 사용한 포커 게임
- zircodice, 주사위 게임
- autogame, 멀티플레이어 확률 게임

코인 공급 및 보상 모델

비스무트는 2031년까지 총 6000여만개의 코인을 채굴자 보상, 개발자 보상(채굴자 보상의 10%), 하이퍼노드 운영자 보상 등으로 발행합니다. 그 이후부터 채굴자 보상(개발자 보상 포함) 지급은 중단됩니다. 그러나 하이퍼노드 운영자 보상은 계속 지급돼 해마다 약 120만개 코인을 발행하는 tail emission(POW 보상 중단 후 소량의 코인 발행을 계속 하는) 구조를 가지고 있습니다.

비스무트 발행량은 약 50년 뒤 현재의(2019년 7월 2일 현재) ETH 발행량 수준(약 1억개)이 될 것으로 예상됩니다.

A rewrite(including recent hardfork changes) needed. Will be translated later.

적용된 암호학

비트코인을 비롯한 대부분 암호화폐는 ECC(Elliptic Curve Cryptography) 암호학과 ECDSA(Elliptic Curve Digital Signature Algorithm) 전자서명 기술을 채택하고 있습니다. ECC 암호키와 전자서명 기술은 기존 암호학 기술에 비해 적은 데이터량으로 동등한 보안 수준을 구현했다는 장점이 있습니다. 그러나 상대적으로 새로운 이 암호학은 결함이 새로 발견될 가능성을 배제할 수 없습니다. 만약 결함이 발견되면, 이를 사용한 블록체인 역시 안전하지 않게 됩니다.

비스무트는, 역설적이지만, 더 오래되고 더 잘 알려진 암호학 기술인 RSA(Ribest Shamir Adelman)를 채택하는 '혁신'을 감행했습니다. RSA는 1977년 일반에 공개된 이후 수십년 간 ssh, ssl 인증서 등(한국의 공인인증서 포함) 웹 보안 기술에 널리 활용되고 있습니다.

큰 프라임 넘버의 속성을 활용해 보안성을 높이는 RSA는 암호키의 크기(bits 단위의 길이)에 비례해 보안 수준이 높아집니다.

- 1024 비트 단위의 RSA 암호키는 웹사이트 로그인 등 중간 수준의 보안 목적에 적합합니다.

- 더 높은 보안 수준이 필요한 어플리케이션이나 데이터에 대해서는, 2048 비트 암호키 사용이 권장됩니다. 2048 비트 암호키는 2030년까지 깨지지 않을 것으로 알려져 있습니다.

- 2048 비트를 초과하는 암호키는 향후 20여년 간 안전할 것으로 전문가들은 예상하고 있습니다.

- 3072 비트 암호키는 2031년 이후에도 보안이 유지돼야 하는 데이터에 추천되고 있습니다.

비스무트는 4096 비트 RSA 암호키를 사용합니다. 고로 매우 안전합니다.

개인키(private key)

개인키는 지갑 소유자만 알아야하는 암호키입니다. 비스무트 지갑은 개인키를 PEM(base64) 형식으로 관리합니다.

- - - - -BEGIN RSA PRIVATE KEY- - - - -
MIIBgjAcBgoqhkiG9w0BDAEDMA4ECKZesfWLQOiDAgID6ASCAWBu7izm8N4V
2puRO/Mdt+Y8ceywxiC0cE57nrbmvaTSvBwTg9b/xyd8YC6QK7lrhC9Njgp/
...
- - - - -END RSA PRIVATE KEY- - - - -

공개키(public key)

공개키 역시 PEM(base64) 형식으로 사용합니다.

- - - - -BEGIN PUBLIC KEY- - - - -

MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAnzgE34oTDlzlPFMsVkNo

foMg9Pm4rG6U8V1fZ/Ewzbtu8UjyvpERblDSaSGBy3C8uZuPpZm/VYTq5KHYJJ6y

...

kLYgWGdQc+MRSkwCwWGQtXECAwEAAQ==

- - - - -END PUBLIC KEY- - - - -

지갑주소(address)

비스무트의 지갑 주소는 공개키(PEM 형식)의 sha224 해시값을 16진수로 표시한 것입니다.

예:

3e08b5538a4509d9daa99e01ca5912cda3e98a7f79ca01248c2bde16

전자서명(signature)

비스무트는 PKCS1.5 전자서명 알고리즘을 사용합니다. 모든 트랜잭션은 공개키와 전자서명을 담아서 전송되며, 트랜잭션의 수신자는 그 유효성을 검증합니다.

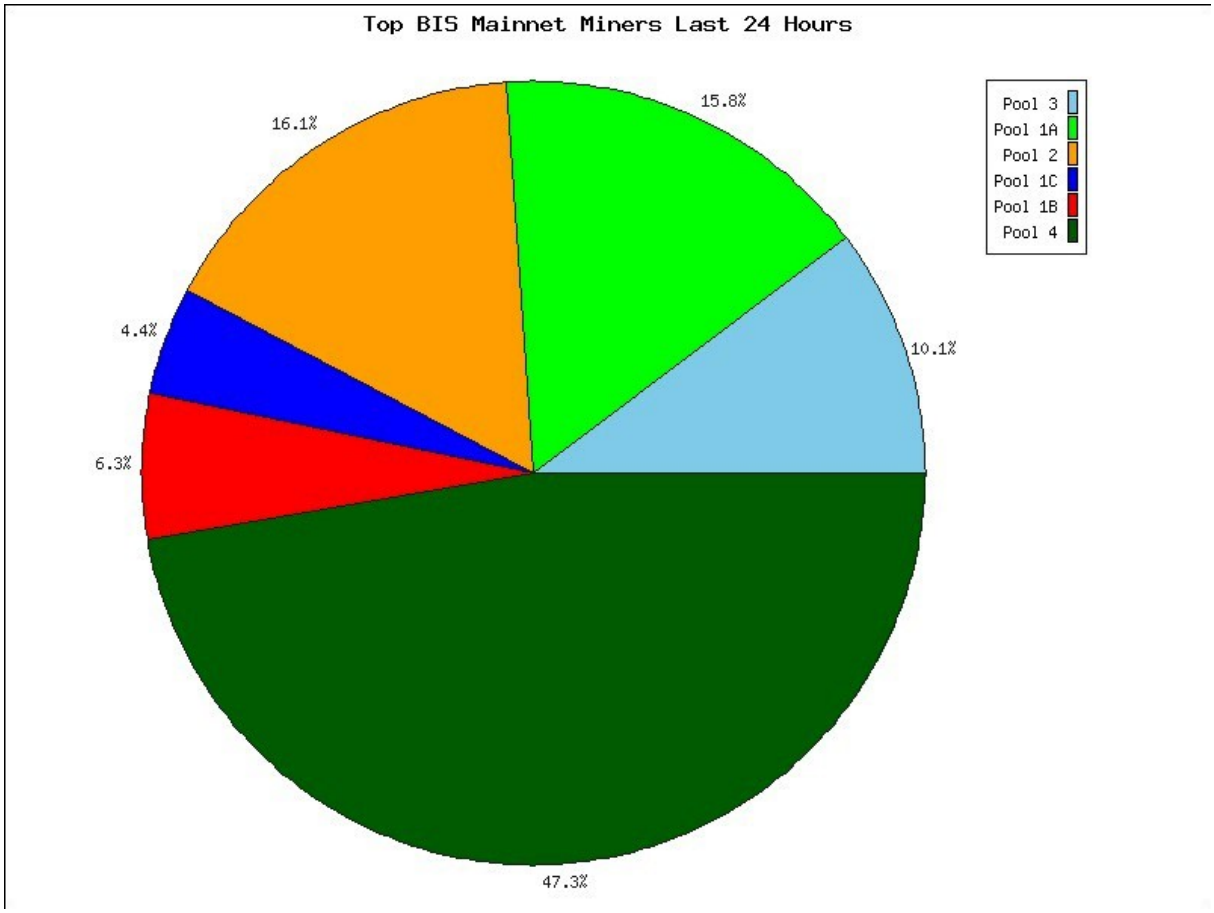
채굴 알고리즘

비스무트의 메인넷은 2017년 5월 1일 가동됐습니다. POW(Proof of Work) 채굴 알고리즘은 sha224를 사용했습니다. 이에 대한 간략한 설명은 http://dx.doi.org/10.4173/mic.2017.4.1 을 참고하시기 바랍니다.

처음에는 CPU 채굴 소프트웨어 밖에 없었습니다. 약 6개월 후 GPU 채굴 소트프웨어가 등장했고, GPU 채굴 풀도 곧 이어 생겨났습니다. 비스무트는 2017년 10월 크립토피아 거래소에 처음 상장됐습니다. 이후 신규 지갑 주소가 급증하면서, 2018년 1월에는 신규 지갑 주소가 4배로 늘었습니다.

비스무트의 초기 채굴 알고리즘은 GPU 메모리 사용량이 매우 적었기 때문에, FPGA나 ASIC 채굴기를 사용한 51% 공격 에 취약했습니다. 비스무트 개발팀은 이 문제점을 알고 있었지만, 네트워크 안정성을 개선하거나 하이퍼노드 등 새로운 기능을 추가하는 데 더 집중했습니다.

2018년 8월경 FPGA 채굴기가 비스무트 채굴에 사용되고 있다는 사실이 명확해졌습니다. FPGA 채굴 해쉬는 51%에 육박했습니다. 아래 그래프는 당시 채굴 풀에 배분된



해쉬량입니다.

FPGA 채굴자는 단독 채굴 또는 '풀 4'(위 그래프 참고)를 오가면서 채굴을 했습니다. 비스무트 개발팀은, 하이퍼노드를 성공적으로 론칭한 뒤, 채굴 알고리즘 개선을 취우선 과제로 정하고 신속히 움직였습니다. 수정된 채굴 알고리즘은 2018년 9월 개발돼 테스트넷 환경에서 실험을 거쳤습니다. 새 채굴 알고리즘은 구상에서 완성까지 약 3주의 기간이

소요됐으며, 채굴자와 풀 운영자들은 1주일 안에 새 알고리즘으로 업그레이드하도록 통보 받았습니다.

FPGA 채굴의 수익성이 지나치게 높았던 이유는 기존 채굴 알고리즘이 마이크로프로세서의 연산력만 사용할 뿐 메모리를 거의 사용하지 않았기 때문입니다. 철저한 조사를 거쳐, 개발팀 멤버인 EggdraSyl이 제안한, 새 알고리즘을 대안으로 채택했습니다. 이 알고리즘은 다음 특징을 갖고 있습니다.

- 메모리 자원을 많이 사용한다.

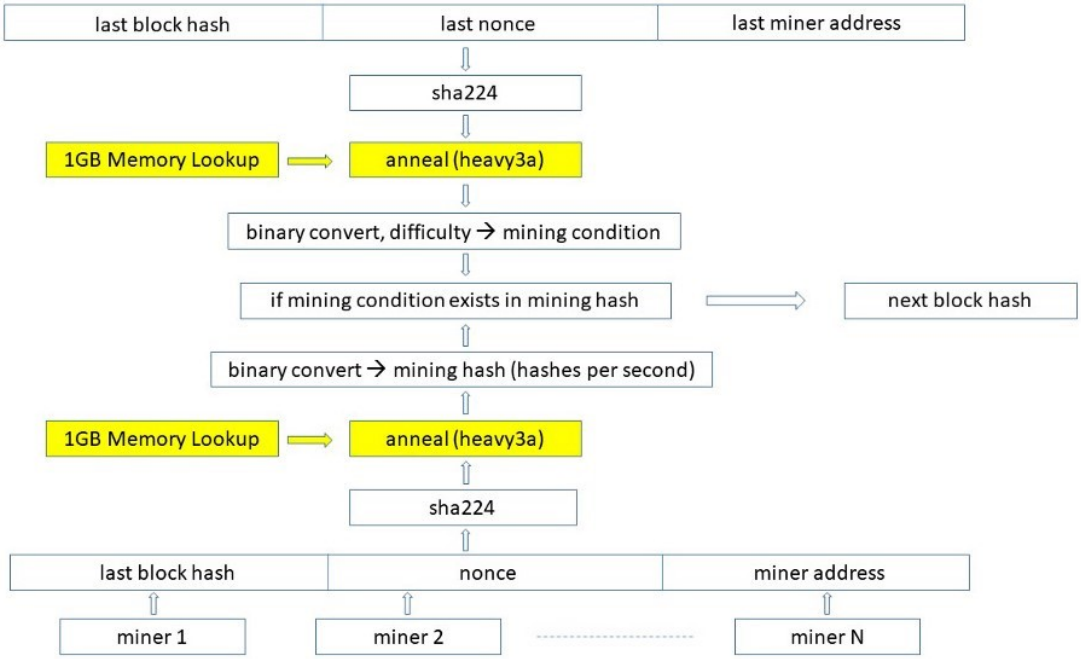
- 특정한 FPGA 채굴기에 많은 패널티를 적용해 채굴 효율성을 감소시킨다.

- 일반 노드의 트랜잭션 검증 속도는 저하시키지 않는다.

- 기존 채굴 프로그램의 코드 수정은 최소화하고, 채굴 풀 등에서 재빨리 적용할 수 있게 한다.

이렇게 해서 새 채굴 알고리즘인 ‘Bismuth Heavy 3’가 탄생했습니다. 새 채굴 알고리즘을 적용하기 위해서는 하드포크가 필요했습니다.

2018년 10월, 85만 4660 번째 블록부터, Heavy 3 채굴 알고리즘이 메인넷에 적용됐습니다. 새 알고리즘은, 아래 일러스트의 노란 박스로 표시된 것처럼, 메모리에 1GB 크기의 랜덤 바이너리 파일을 저장하도록 함으로써 FPGA, ASIC 저항성을 높였습니다.



Heavy 3

비스무트의 Heavy 3 채굴 알고리즘은 단순하고 효율적입니다. 이 알고리즘은 테스트를 거친 각각의 논스(nonce)에 대해 정해진 룩업 테이블(lookup table)에서 임의의 오프셋(offset) 값을 읽어들이도록 합니다.

이 개념은 여타 채굴 알고리즘에도 FPGA 또는 ASIC 공격을 막는 부가 기능으로 적용할 수 있습니다. 매칭 알고리즘이 해시캐시(hashcash)를 사용하는지 여부와 상관 없습니다. 비스무트 채굴 알고리즘은 해시캐시를 사용하지 않습니다. 테스트를 거친 최종 해시 스테이트(state)는 32비트 words의 벡터(a vector of 32 bits words)입니다. 해시값이기 때문에 임의의 데이터를 담은 일종의 랜덤 벡터로 볼 수 있습니다. 해시 연산의 핵심 특징은 이 연산을 되돌릴 수 없다는 것입니다. 룩업 테이블 역시 랜덤 데이터를 담고 있습니다. 각각의 논스에 대해 추가된 단계는 해시 결과를 룩업 테이블에서 주어진 랜덤 벡터로 XOR 변환을 하는 것입니다. 인덱스의 시작점은 해시값 자체에 의해 정해집니다. 랜덤하고 예측할 수 없는 위치입니다. 결과값(XOR 변환한 해시 스테이트)은 난이도 매칭 함수의 입력 벡터가 됩니다.

- 이 방법은 선량한 채굴 후보(good candidate)를 선택하는 확률에 영향을 주지 않습니다.

- 이 방법은 해싱 알고리즘 자체를 변화시키지 않습니다.

- 이 방법은 난이도 매칭 알고리즘에도 변화를 주지 않습니다.

- 이 방법은 각각의 논스의 랜덤 인덱스로부터 약 8개 words를 읽도록 요구합니다.

- 채굴 프로그램은 메모리에 항상 전체 룩업 테이블을 카피한 상태를 유지해야 합니다.

이 방법은 어떤 채굴 알고리즘에도 즉각 적용할 수 있는 범용 솔루션입니다.

중장기 고려사항

비스무트 개발팀은 FPGA 또는 전용 ASIC 하드웨어를 채굴에 사용하는 행위가 잘못됐다고 생각하지 않습니다. 오히려 환영합니다.

- FPGA, ASIC 하드웨어를 채굴에 사용함으로써 POW 코인은 네트워크의 보안을 강화시킬 수 있습니다. GPU 채굴은 nicehash 등 해시 대여 서비스로부터 빌린 해시의 공격을 받고 휘둘릴 수 있습니다.

- 전기요금 측면에서 GPU 채굴보다 더 유리합니다. 전용 하드웨어 채굴기는 효율적으로 더 높은 해시를 네트워크 보안에 적용할 수 있게 해줍니다. 악의적 공격의 비용을 증가시켜 보안을 향상시킵니다.

그러나 이같은 장점이 실현되려면, 전용 채굴 장비를 누구나 손쉽게 구할 수 있어야 합니다. 한 명의 채굴자가 수 천개의 전용 채굴 장비를 독점적으로 운영하는 상황이라면 장점이 아니라 약점이 됩니다.

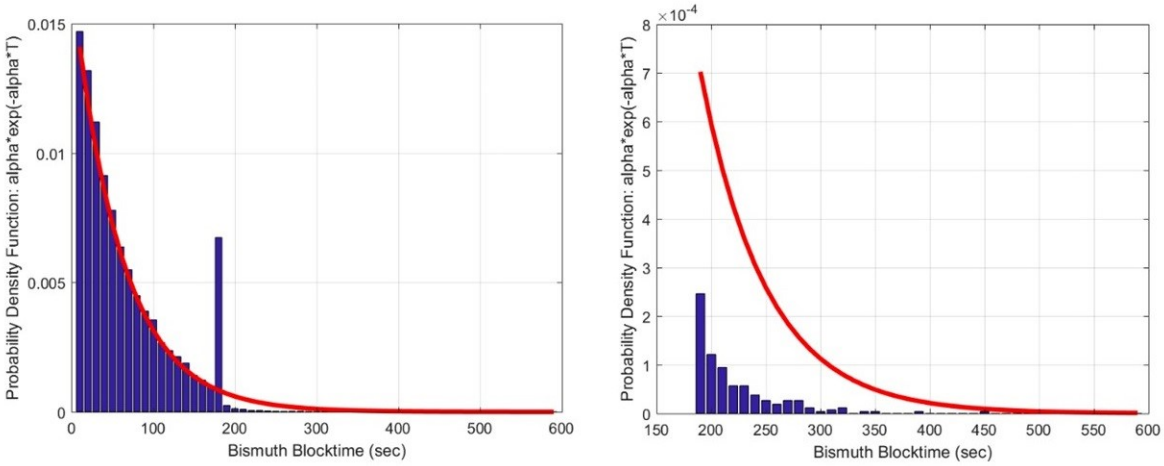
개발팀의 차기 과제는 여러가지 다양한 채굴 수단(CPU, GPU, FPGA, ASIC 등)이 공존할 수 있는 환경을 구축하는 것입니다. 누구나 채굴에 참여할 수 있고, 이익을 얻고, 네트워크 보안에 기여할 수 있도록 공정한 기회가 열려 있어야 합니다. 만약 특정 채굴자가 독점적인 채굴 수단으로 불공정한 이익을 챙기는 상황이 재연된다면, 채굴 알고리즘은 신속히 변경될 수 있습니다.

롱테일 블록타임 방지(Tail Removal Block Validation)

비트코인을 비롯한 많은 암호화폐는 블록타임을 PDF(Probability Density Function) 방식으로 배분합니다. PDF는 구조상 롱테일(long tail) 문제가 발생합니다. 네트워크 해시가 동일하더라도, 이따금, 블록타임이 평소보다 매우 길어질 가능성이 있다는 것입니다. 이례적으로 긴 블록타임은 두 가지 이유에서 문제가 됩니다.

1) 트랜잭션을 컨펌하는 시간이 길어지는 것은 바람직하지 않습니다. 평균 블록타임보다 매우 긴 컨펌 시간은 부정적인 사용자 경험을 유발합니다.

2) 블록체인의 블록타임 대응 알고리즘은 롱테일 블록타임과 네트워크 해시 저하를



구별하지 못할 수 있습니다. 그 결과 일부 재빠른 노드들이 난이도를 하향 조정하는 오류를 범할 수 있습니다.

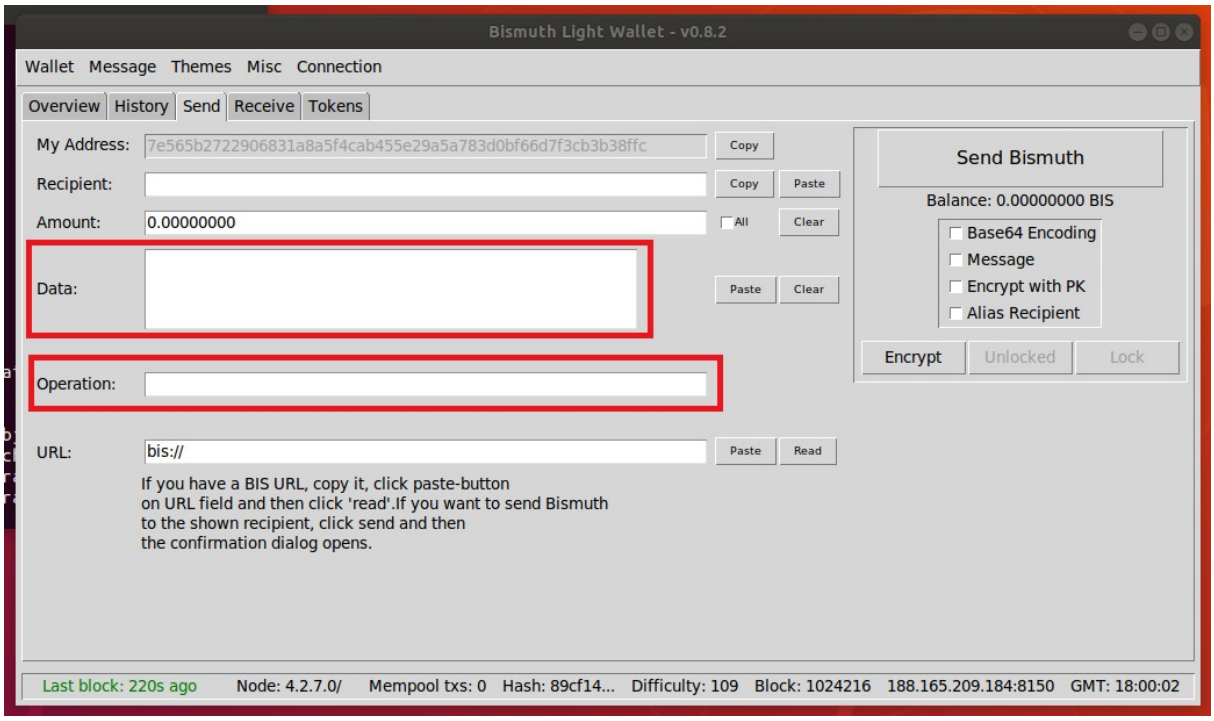
이는 의도하지 않은 진폭과 시스템 불안정을 유발할 수 있습니다.

비스무트는 이런 롱테일 블록타임을 방지하는 알고리즘을 적용했습니다. 그 결과는 상기 그래프가 잘 보여줍니다. 왼쪽 그래프는 롱테일 제거 코드를 적용하지 않은 PDF(빨간 선)와 롱테일 제거 코드를 적용한 PDF(파란색 막대 그래프)를 비교한 것입니다. 오른쪽 그래프는 같은 데이터를 180초가 넘는 블록타임만 확대해서 비교한 것입니다. 이 그래프들에서 볼 수 있듯이, 180초를 초과하는 롱테일 블록타임의 가능성은 현저히 줄어들었습니다. 덕분에 비스무트는 상대적으로 안정적인 블록타임에 트랜잭션을 처리할 수 있습니다. 더 자세한 정보는 다음 논문을 참고하시기 바랍니다.

J. Kučera and G. Hovland, “Tail Removal Block Validation: Implementation and Analysis”, http://dx.doi.org/10.4173/mic.2018.3.1

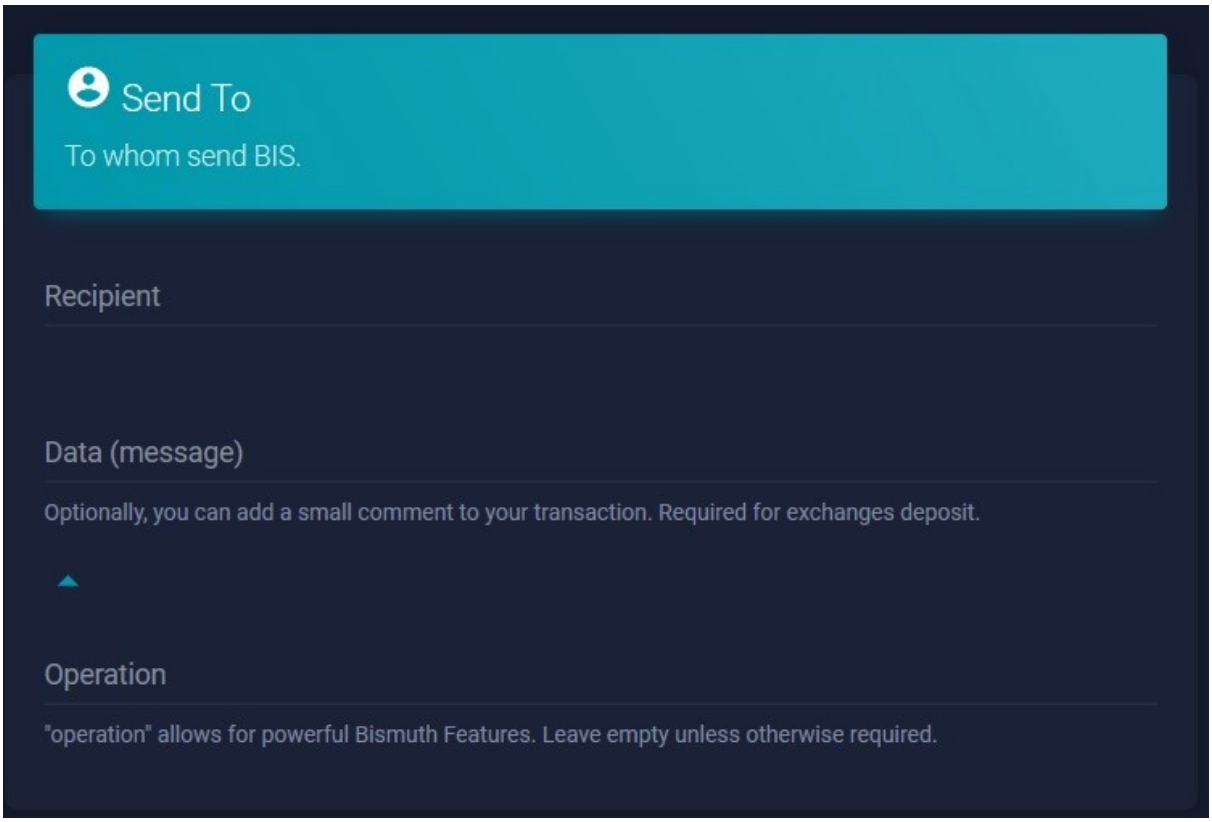
오퍼레이션과 데이터 필드

비스무트는 dApps 개발자들이 사용할 수 있는 두 개의 블록 필드(Operation, Data)를 가지고 있습니다. 아래 그림(레거시 지갑 화면)에서 빨간색 박스로 표시한 것이 바로 그



필드입니다.

토네이도 지갑에서는 아래 화면으로 표시됩니다.



오퍼레이션(Operation)과 데이터(Data) 필드는 프로그래밍에 활용할 수 있습니다. 그 사례는 다음 링크를 참고하시기 바랍니다. 링크.

이 사례의 소스코드를 살펴보면,

```
from bismuthclient.bismuthclient import BismuthClient
if __name__ == "__main__":
    client = BismuthClient(wallet_file='wallet.der')
    if not client.address:
        client.new_wallet()
    client.load_wallet()
    print(f"My address is {client.address}")
    txid = client.send(recipient=client.address, amount=0) # Tries to send
        0 to self
    print(f"Txid is {txid}")
```

이 예제는 0 BIS를 자기 주소로 보내는 코드입니다. BIS를 다른 주소로 보내려면, 코드에서, client.address 를 원하는 주소의 문자열로 바꿔주면 됩니다. 예를 들자면, recipient=”9ba0f8ca03439a8b4222b256a5f56f4f563f6d83755f525992fa5daf”

오퍼레이션과 데이터 필드에 대해서는, 아래 예제와 같은 명령을 사용할 수 있습니다.

```
Txid =
client.send(recipient="9ba0f8ca03439a8b4222b256a5f56f4f563f6d83755f525992fa5daf",
operation='dragg:transfer', data='draggon_adn')
```

오퍼레이션과 데이터 필드를 사용한다는 점이 비스무트가 다른 암호화폐와 다른 주요 특징 중 하나입니다. 현실주의 원칙 때문에, 비스무트는 데이터를 블록체인에 저장할 필요성을 인정합니다. 비스무트 유저는 데이터 저장을 어렵지 않게 할 수 있습니다.

비스무트는 이 데이터를 상위 계층의 오퍼레이션을 위한 메타 데이터로 사용합니다. 이는 비스무트의 기본 정신에도 부합합니다. '추상적 데이터', '읽고 쓰기 쉽게'. 비스무트 노드는 오퍼레이션과 데이터 필드에 담긴 정보가 무엇을 의미하는지 알 필요도 명령을 수행할 필요도 없습니다. 이 오퍼레이션에 참여하는 앱들만 이 정보를 해석하고 그에 따라 동작할 수 있습니다. 또한 오퍼레이션과 데이터 필드는 노드와 dApps가 서로 분리되도록 해줍니다. 노드는 기초적인 기능, 네트워크 전송, 인증, 위변조 방지, 추상적 데이터만 담당하고, dApps는 해당 데이터를 사용하고, 해석하며, 이에 따라 동작하는 상위 계층의 기능을 수행합니다.

프라이빗 컨트랙트

앞서 설명했듯이 비스무트는 추가적인 프로토콜로 활용할 수 있는 두 개의 필드, 오퍼레이션과 데이터 필드를 가지고 있습니다. 이 필드는 프라이빗 컨트랙트를 만들기 위해 다방면으로 활용할 수 있습니다.

1) 암호화된 메시지 등 프라이빗 데이터를 주고 받는 용도

2) 퍼블릭 컨트랙트와 대비되는 프라이빗 컨트랙트

3) 추상적 트랜잭션과 암호화된 메시지를 활용해 수신자를 추적할 수 없게 하는 익명화 용도

추상적 트랜잭션

비스무트의 또 다른 강점으로는 추상적 트랜잭션(abstract transaction)이 있습니다. 추상적 트랜잭션은 송금 금액 없이(0 BIS), 특정 프로토콜에 참여하는 dApps만 이해할 수 있는 (추상적) 데이터를 담은 트랜잭션입니다. 이 트랜잭션에서 '오퍼레이션' 필드는 일종의 명령어로 사용됩니다. 일반적인 사용 형식은 '프로토콜 분류 명칭(class):명령어(operation)'입니다. 프로그래밍에서 보편적으로 쓰이는 네임스페이스(namespace)처럼 식별하기 쉽도록 말입니다. '데이터' 필드는 오퍼레이션과 연관된 (짧은) 데이터를 담습니다. 개발자는 자신만의 오퍼레이션을 정의할 수 있습니다. 그러나 이미 사용된 오퍼레이션 네임스페이스와 겹치지 않도록 해야 합니다. 이미 사용된 오퍼레이션 프로토콜 명칭 목록은 다음 링크를 참고하시기 바랍니다. 링크.

비스무트의 트랜잭션 수수료는 고정 수수료에 데이터 필드의 데이터량에 비례한 추가 수수료를 더해서 산출합니다. Fee = 0.01 + len(openfield/100000). 블록체인에는 가급적 많은 데이터를 저장하지 않도록 해야 하며, 이를 위해 향후 데이터 수수료가 인상될 수 있습니다. 개발자는 블록체인에 저장하는 데이터를 최소화해야 하며, 실제 데이터의 저장은 사이드체인이나 dApps, dApps 채널 등을 활용해야 합니다.

하이퍼노드와 사이드체인

마스터노드 코인들이 새로 쏟아지는 요즘, 비스무트의 하이퍼노드가 왜 혁신적인 지 설명할 필요성이 있습니다.

신기술 시험대이자 비스무트 잠재력의 증거인 하이퍼노드

비스무트가 성장하고 개발팀이 더 많은 경험을 축적하면서, 기존 노드의 약점이 보이기 시작했습니다. 이미 잘 작동하고 있는 블록체인의 디자인을 변경을 하거나 신기술을 시험을 하는 것은 쉽지 않습니다. 이미 많은 개인과 조직이 이 블록체인을 사용하고 있기

때문입니다. 그러나 하이퍼노드를 신기술과 새 라이브러리, 알고리즘 테스트에 활용하는 것은 상대적으로 쉽습니다. 비스무트 하이퍼노드에는 코어 코드에 향후 적용할 여러 신기술이 미리 적용될 것입니다.

하이퍼노드는 비스무트의 추상적 트랜잭션의 효율성과 개발하기 쉽다는 장점을 잘 보여줍니다. 하이퍼노드는 비스무트 블록체인의 개방적이고 추상적인 계층 모델을 이용해 무엇을 할 수 있는 지 잘 보여주는 사례입니다.

네트워크 가치

하이퍼노드의 목적은 비스무트 네트워크에 새로운 가치를 부가하는 것입니다. 일부 초보적인 마스터노드들은 단지 일정량의 코인을 묶어두는 '스테이킹(staking)'이거나 'ping' 반응을 확인하는 루틴에 불과합니다.

비스무트의 하이퍼노드는 완전히 다른 차원에서 작동합니다. 자체적인 블록체인을 가지고 있으며, 이 블록체인은 암호화폐 대신 'KPI(Key Performance Indicators)'라는 척도(metrics)를 사용합니다. 비스무트 노드와 하이퍼노드는 느슨하게 연동되어 서로 독립적으로 작동합니다. 노드는 POW(Proof of Work) 채굴 체인인 반면, 하이퍼노드는 POS(Proof of Stake) 체인입니다. 하이퍼노드들간에는 채굴 경쟁이 없습니다. 하이퍼노드들은 서로 KPI를 관측하며 이를 POS 체인에 저장합니다. 하이퍼노드는 POW 체인에 속하지 않기 때문에 POW 체인에 대한 공격 벡터를 늘리지 않습니다.

하이퍼노드의 적격성 여부는 POW 체인에 저장된 비가역적(immutable) 정보에 의해 정해집니다. POS 및 POW 노드들의 'Quality index'와 'bad behavior'는 다음 액션을 위해 POS 체인에 기록됩니다. 이 기록 역시 비가역적입니다.

두 체인의 상호 독립성은 아래를 보장합니다.

a) 네트워크 조작이 훨씬 더 어렵다. 매우 다른 방식으로 작동하는 두 체인을 모두 조작해야 하기 때문입니다.

b) 나쁜 행위자 그리고 조작 시도들은 독립적이고 비가역적인 방법으로 기록됩니다. 들키지 않고 속이는 것은 불가능합니다.

비스무트는 아마도 KPI를 메인체인과 사이드체인에 각각 독립적이면서도 통합적으로 적용한 최초의 암호화폐일 것입니다. 이로써 행위자들의 공정한 게임 플레이가 보장됩니다.

하이브리드 POW/POS

비스무트 POW와 POS에 사용한 프로토콜은 하이브리드, 두 계층 접근법입니다. 이 방법은 POW와 POS 체인 각각의 강점을 이용하며, 여타 하이브리드 접근법들의 취약점을 피하려고 고안됐습니다. 비스무트의 POS 계층은 메인체인의 내부 컨센서스에 포함되지 않고, POW 체인에 대해 독립적인 관찰자로 작동합니다. 관찰 결과(metrics)는 나쁜 행위자를 제재하는 데 사용될 수 있습니다.

POS는 POW 행위자를 감시합니다. POW는 누가 POS 행위자가 될 수 있는 지 결정합니다. 서로 꼬리를 무는 모양새입니다. 각각의 체인은 서로에 대해 일정한 통제력을 가집니다. 이로써 보안은 크게 개선됩니다. 비스무트 네트워크를 조작하기 위해서는 두 체인을 동시에 그리고 일관되게 공격해야 합니다. 두 체인의 작동 방식은 매우 다르기 때문에 공격이 성공할 가능성은 극히 낮습니다.

사이드체인의 향후 응용 분야

하이퍼노드는 두 블록체인 계층(layer)의 느슨한 결합(loosely coupled) 방식을 사용합니다. 이 방식의 사이드체인은 여러 장점이 있고 앞으로 많은 사례에 응용될 수 있습니다.

- 노드들이 서로 더 많은 블록을 생성하려 경쟁하거나 다른 노드의 블록 생성을 방해할 필요가 없습니다.

- 앞으로 수많은 POS 체인이 비스무트 네트워크에 추가될 수 있습니다. 그렇더라도, 느슨한 결합 덕분에, POW 메인 체인에는 부담을 거의 주지 않습니다. 다양한 POS 사이드체인들을 활용하면 매우 유연한 서비스가 가능합니다.

- 비스무트 방식의 사이드체인은 메인 노드의 알고리즘을 거의 수정할 필요가 없기 때문에 다른 암호화폐에도 쉽게 적용할 수 있습니다. 거의 모든 POW 암호화폐들은 비스무트 하이퍼노드 방식의 POS 체인을 추가해 보안을 강화하고 사이드체인, 차일드체인으로 활용할 수 있습니다.

비스무트는 실용적이고 현장 검증을 거친 신기술을 계속해서 발굴하고 있습니다. 하이퍼노드를 운영하기 쉬운 사이드체인 프레임워크로 발전시키는 것이 목표입니다. 이를 활용해 만든 사이드체인은, 고유한 운영 규칙과 블록타임, 수수료(또는 무료), 데이터 저장방법을 가지면서도, 비스무트 POW 체인에 의해 보안이 유지될 수 있습니다.

신기술의 시험대

비스무트는 단순한 개념증명(proof of conecpt)에서 시작해, 필요한 기능을 모두 갖춘 블록체인 노드와 클라이언트 코드 베이스로 성장했습니다. 현재의 네트워크와 호환성을 유지하기 위해서는 신기술 적용을 보류해야 하는 경우가 있습니다. 하이퍼노드는 이런 제약으로부터 자유롭습니다. 처음부터 첨단 신기술을 적용할 수 있습니다.

비동기식(Async) 입출력(IO)

최신 파이썬 문법은 'Async / Await, co-routine' 등 강력한 기능을 제공합니다. 이를 활용하면 효율적이면서도 읽기 쉬운 비동기식 코드 작성이 가능합니다. 수 백개의 프로세스를 동시에 띄우고, 록(lock)을 관리하고, 프로세스 간 레이스(race) 컨디션을 디버그하기 위해 곤란을 겪을 필요가 없습니다. 하나의 데이터베이스를 여러 프로세스가 동시다발적으로 접근하는 상황을 관리하려 애먹을 필요도 없습니다. (파이썬 멀티쓰레드 프로그래밍에서) 악명 높은 GIL(Global Interpreter Lock)의 제약도 없습니다.

하이퍼노드는 최신 파이썬의 Async 문법을 그 한계까지 사용합니다. 하이퍼노드의 핵심 코드는 토네이도 서버와 클라이언트, 비동기식 호출을 하는 콜백 함수들(callbacks)입니다. 이 방식은 보안에도 유리하지만 노드 자체의 성능에도 큰 이점으로 작용합니다.

Protobuf

구글이 고안한 네트워크 데이터 교환 프로토콜인 '프로토버프(Protobuf)'는 처리 속도가 빠르고 효율적입니다. 거의 모든 프로그래밍 언어를 지원합니다. 하이퍼노드는 프로토버프를 기초 데이터 교환 포맷으로 사용합니다.

장점은 다음과 같습니다.

- 빠르고 잉여 데이터가 적습니다.

- 패킷 사이즈가 작습니다.

- 데이터 체크 수단이 많습니다.

- 여러 운영체제 및 프로그래밍 언어와 호환됩니다.

하이퍼노드의 암호학

해시

하이퍼노드는 최신 blake2 암호학 기반을 사용합니다. 빠르고 안전하며 결과값의 길이(length)를 조절할 수 있습니다.

암호키와 전자서명

하이퍼노드의 주소는 ECDSA 암호학 커브를 사용합니다.

POW/POS 블록체인의 결합

POW와 POW 체인 계층(layer)은 어떻게 상호작용할까? POS(하이퍼노드)는 POW(노드)를 어떻게 사용하나?

하이퍼노드는 비스무트 노드(POW)와 함께 작동시켜야 합니다. 하이퍼노드는 POW 블록체인의 데이터베이스와 피어(peers), 콘센서스(consensus), 블록하이트(block height) 등 현 상태(status)를 읽을 수 있습니다. POW 블록체인에 데이터를 쓸 수는 없고 읽을 수만 있습니다. 하이퍼노드는 단지 관찰자일 뿐입니다. 하이퍼노드는 관찰한 데이터를 이용해 다음을 할 수 있습니다.

A) 유효한 하이퍼노드들에 대한, 안전하고, 비가역적이며, 공유된 리스트를 얻을 수 있습니다. 매번 라운드가 시작될 때마다, 하이퍼노드는 유효한 자격을 갖춘 하이퍼노드들의 리스트를 공유해야 합니다. 이를 바탕으로 해당 라운드의 심판(juror)이 정해집니다. 리스트는 POW 체인에서 추출합니다. 체크포인트는 충분한 컨펌이 쌓여

안정성을 확보한 과거 블록을 사용합니다. 유효한 하이퍼노드 리스트는 소유자가 유효하게 등록한 하이퍼노드들의 목록입니다. 이 리스트는 POS 계층에서 조작할 수 없습니다.

B) POW 피어(peers)에 대한 관찰 데이터(metrics)를 수집합니다. POW 노드에 어떤 피어가 접속할 때마다 이 피어에 대한 많은 정보가 공개됩니다. 버전, 블록하이트, 아이피 주소, 핑 타임, 컨센서스 상태, 연결 상태 등. 롤백(rollbacks) 같은 특정 행위 정보도 입수됩니다. 이 정보들은 모두 하이퍼노드가 수집하고 POS 체인에 기록할 수 있는 관찰 데이터입니다. 하이퍼노드들이 수집한 관찰 데이터는 매 라운드마다 축적되며 POW 노드들의 상태를 평가하고 '선량한' 그리고 '나쁜' 피어 목록을 각각 작성하는데 사용할 수 있습니다.

POW는 POS를 어떻게 활용하나?

마찬가지로, POW 계층 역시 하이퍼노드가 제공하는 데이터를 사용할 수 있습니다.

- 나쁜, 선량한 피어들 목록을 실시간으로 작성할 수 있습니다.

- POW 네트워크 블록하이트에 대해 더 믿을 수 있는 정보를 얻을 수 있습니다.

이를 활용해 각각의 노드들은 질이 나쁜 피어 노드를 미리 차단할 수 있습니다. 버전이 너무 오래됐거나 관리가 소홀한 노드, 나쁜 블록에 갇혀 헤어 나오지 못하는 노드를 차단할 뿐 아니라 가짜 노드들의 벌떼 공격, 채굴 노드에 대한 집중 공격을 막는 데도 유용합니다.

POS 체인은, POW 코드의 내부 로직에 포함되지 않기 때문에 POW 체인의 복잡성과 취약성을 증가시키지 않습니다. POW 및 POW 행위자들에 대한 더 객관적인 평가 자료를 제공하는 역할을 합니다.

관찰 데이터(metrics, KPI)

많은 관찰 데이터가 활용될 수 있습니다. 하이퍼노드의 역할은 모든 관찰 정보를 정확하게 수집하는 것입니다. 개발팀은 이 정보들을 분석하고 무엇을 어떻게 사용할지 결정합니다. 관찰할 데이터의 종류와 어떤 데이터에 대해 어떤 대응할 지는 차차 달라질 수 있습니다. 처음에는 수동으로 이 데이터를 분석하지만, 향후 거버넌스 시스템의 기초 데이터가 될 것으로 예상합니다. 하이퍼노드 또는 노드 소유자들은 각각의 관찰 데이터를 활용해 나쁜 행위자 대해 어떤 조치를 취할 지, 대응 수위는 어떻게 할 지 투표로 결정할 수 있을

것입니다. 하이퍼노드가 수집하는 구체적 데이터들에 대해서는 추후 작성할 문서에서 상세하게 소개할 예정입니다.

하이퍼노드 보상 지급

일반적으로 마스터노드는 블록을 작성할 때마다 일정한 보상을 받습니다. 이는 각각의 노드가 더 많은 보상을 받으려고 속임수를 쓰거나 다른 노드의 블록 작성을 방해하려는 동기를 유발합니다.

비스무트 하이퍼노드는 보상을 블록 작성과 상관 없이 정기적으로 지급합니다. 랜덤한 요소는 개입되지 않습니다. 아울러 속임수를 시도할 경우 그 기록이 POS 블록체인에 영원히 남습니다.

매 라운드가 끝날 때마다 잘 작동하고 있는 모든 하이퍼노드가 보상을 받습니다. 보상은 여러 방법으로 지급할 수 있습니다. 비스무트는 일단 프라이빗 컨트랙트를 사용해 지급하고 있습니다. 이 컨트랙트는 유지 보수가 쉽고 개발팀이 안전하게 관리합니다.

하이퍼노드 보상 지급 방법은 다음과 같습니다.

- 일정 기간마다 정해진 보상 금액이 있습니다(고정 금액이거나 거버넌스 시스템에 의해 정해지는 금액)

- 이 기간에 ‘잘 작동한(active)’ 하이퍼노드는 보상을 나눠 가집니다. ‘잘 작동한(active)’ 하이퍼노드가 되려면, ‘유효한(valid) 하이퍼노드’ 목록에 포함돼 있으면서, 네트워크 관찰 데이터(metrics)를 수집 전송하고, 피어(peers)들과 상호작용해야 합니다. 이 기간 중 심판(juror)으로 선정돼 직접 블록을 생성할 수도 아닐 수도 있지만, 그 여부는 보상 금액에 영향을 주지 않습니다.

- 보상은 해당 하이퍼노드에 예치한 담보(collateral) 금액에 비례해서 정해집니다. A하이퍼노드에 예치한 담보가 B 하이퍼노드에 예치한 담보의 두 배라면, A 노드의 보상은 B 노드의 두 배가 됩니다.

거버넌스(Governance)

수집된 KPI(관찰 데이터)와 그 레벨에따라 특정 행위자들을 거부하거나 제재하는 조치가 취해질 수 있습니다. 이를 결정하는 거버넌스 시스템에 대한 질문이 자연스럽게 제기될 수 있습니다. 그러나 초창기에는, 프로젝트의 실험성을 고려할 때, 거버넌스 시스템을 적용하는 것이 불가능합니다. 개발팀은 현재 수동으로 KPI를 분석하고 의미 있는 정보를 찾아내고 있습니다. 향후 이 필터링 과정을 자동화하고, 하이퍼노드 소유자들의 투표로 중요 결정이 이뤄지는 방식으로 전환 할 수 있습니다.

하이퍼블록 압축

비스무트는 이중(dual) 데이터베이스 시스템을 사용하고 있습니다. 불필요한 데이터 중복을 막고 처리 속도를 빠르게 하기 위해서입니다. 표준의 블록체인 풀(full) 데이터베이스에 덧붙여 이 데이터를 압축한 하이퍼블록(hyperblocks)이라는 데이터베이스를 추가로 사용합니다. 하이퍼블록은 최신 1만5000개 블록만 모든 데이터를 보관합니다. 그 이전 블록들은 최종 잔액(0가 아닌 계정의) 정보만 추출해 보관합니다. 하이퍼블록의 잔액을 표준 블록체인 데이터베이스의 잔액과 비교해 서로 일치하는지 대조하는 경우도 있습니다.

하이퍼블록은 노드 실행시 메모리에 로딩해 사용합니다. 하드디스크 사용을 최소화하고 데이터베이스 검색 속도와 접근성을 높이기 위해서입니다.

페널티 시스템

비스무트 컨센서스 시스템에서, 모든 노드는 자신과 연결된 클라이언트들의 행동을 추적합니다. 싱글 블록 롤백(single block rollback)을 통해 체인 갈아타기(switch)를 강요하는 모든 노드는 패널티를 받습니다. 이 패널티는 해당 노드가 정직성을 유지하고 고의적이지 않은 방식으로 가장 긴 체인 블록을 전달할 경우 절반으로 경감됩니다. (블록하이트가) 로컬 컨센서스에서 지나치게 멀리 앞선 노드는 접속을 거부당합니다.

이런 규칙들은 공격자가, 공격을 성공시키기 위해, 네트워크 연산력의 절반 이상을 보유할 뿐 아니라 전체 노드의 과반수를 확보해야만 하도록 강제합니다. 네트워크의 모든 노드와

연결되고 이 노드들을 동시에 공격하지 않으면 성공하기 어렵습니다. 설령 한 번의 공격이 성공하더라도, 다음 공격은 블록이 누적될 수록 더 어려워집니다.

테스트넷

비스무트의 테스트넷은 메인넷보다 노드 수가 훨씬 적습니다. 테스트넷 노드를 설치하려면, 다음 설정값을 config.txt 파일에 반영해야 합니다.

```
Port=2829
version=testnet
version_allow=testnet
testnet=True
```

테스트넷 채굴 알고리즘의 난이도는 의도적으로 낮게 설정돼 있습니다. 덕분에, 개발자는 로컬 풀과 채굴 소프트웨어를 설치해 손쉽게 테스트용 BIS를 생성할 수 있습니다.
테스트넷 채굴용으로 optipoolware를 추천합니다.

레그넷(Regnet)

레그넷 노드를 설치하려면, 다음 설정값을 config.txt 파일에 반영해야 합니다.

```
version=regnet
version_allow=regnet
```

테스트넷이나 레그넷이 잘 작동하는지 확인하려면 아래 명령어를 사용하면 됩니다.

```
python3 commands.py statusget
```

그 출력 결과는 아래와 비슷할 것입니다.

```
Number of arguments: 2 arguments.
Argument List: ['commands.py', 'statusget']
Regtest mode
```

{"protocolversion": "regnet", "address":
"6a8b4990784617730af465a0dfcbb87284bca8b2189e02798d0a5a5f",
"walletversion": "4.2.9", "testnet": false, "blocks": 1, "timeoffset":
0, "connections": 0, "connections_list": {}, "difficulty": 24.0,
"threads": 3, "uptime": 131, "consensus": null, "consensus_percent":
0, "server_timestamp": "1549123577.02", "regnet": true}

레그넷은 채굴 소프트웨어를 설치할 필요가 없습니다. ‘generate’ 명령어를 사용하면 지갑 주소로 N 블록을 채굴한 보상이 즉시 입금됩니다. 예를 들어, ‘generate 10’으로 10개 블록을 생성하면, 레그넷에서 테스트를 할 충분한 BIS를 얻을 수 있습니다. 이후 ‘generate 1’ 블록 명령을 실행할 때마다 1블록씩 채굴이 이뤄지고, 멤풀(mempool)에 있는 트랜잭션이 블록에 기록됩니다.

레그넷은 블록하이트 1번부터 시작되고 블록체인 데이터베이스를 싱크할 필요가 없기 때문에 편리합니다. 새로운 dApps은 개발 초기 많은 테스트를 레그넷에서 할 수 있습니다. 분산 네트워크에서 기능을 테스트할 필요가 생기면, 그 때 레그넷에서 테스트넷으로 개발 환경을 전환하면 됩니다.

연구와 개발

비스무트는 연구 개발 용도로 이상적인 플랫폼입니다. 비스무트 메인넷과 테스트넷을 활용한 연구 사례로는 아래 학술 저널 논문들이 있습니다.

• J. Kučera and G. Hovland, ”Tail Removal Block Validation: Implementation and Analysis”, http://dx.doi.org/10.4173/mic.2018.3.1

• G. Hovland and J. Kučera, Nonlinear Feedback Control and Stability Analysis of a Proof-of-Work Blockchain”, http://dx.doi.org/10.4173/mic.2017.4.1

레그넷은 특히 교육 환경에서 유용합니다. 학생들은, 분산 네트워크에 연결될 필요 없이, 각자 자신의 컴퓨터에서 레그넷을 가동할 수 있습니다. 블록체인 기술의 많은 개념은 비스무트 레그넷을 사용해 교육할 수 있습니다.

비스무트를 사용해 새로운 기능과 서비스를 개발하는 연구자와 학생은 디스코드 앱을 이용해 비스무트 코어 개발팀(https://discord.gg/4tB3pYJ)과 컨택하기 바랍니다. 제안한 새로운 기능과 서비스는 일정한 테스트를 거쳐 비스무트 코드에 반영될 수도 있습니다.

향후 전망

비스무트 프로젝트의 목표는 가능한 슬림하고 효율적인 코어 노드를 만들고, 그 사용 설명서를 잘 갖추는 것입니다. 이 비스무트 기반 위에서 미래의 응용 프로그램과 확장 기능은 플러그인 시스템을 활용하고 개별적으로 소스코드 보관(repositories) 관리를 할 것을 권장합니다.

비스무트는 이미 많은 기능을 갖춘 암호화폐입니다. 백서는 이미 구현되고 테스트되고 작동하는 것들을 주로 기술하고 있습니다. 더 많은 기능과 서비스가 미래에 개발될 것입니다. 앞으로 추가될 기능은 이 백서의 차기 버전에게 업데이트 될 것입니다. 개발 로드맵은 다음 github 링크를 클릭해 보실 수 있습니다. 링크.

요약

이 백서는 비스무트의 철학과 원칙(pillars), 기능들에 대한 개관을 제공합니다. 비스무트의 3대 개발 원칙은 1) 현실주의, 개발팀이 새로운 기능을 도입할 때 실용주의적 접근을 한다는 뜻, 2) 필수적 데이터만 블록체인에 저장, 개발자들은 오퍼레이션(operation)과 데이터(data) 필드를 사용해 분산 어플리케이션 또는 프라이빗 스마트 컨트랙트를 개발하고 중요 데이터를 온체인 저장소에 보관할 수 있다는 뜻, 3) 신뢰 대상과 범위의 명확화, 이용자가 해당 서비스 운영자를 신뢰할 필요성이 있다는 사실을 분명히 한다는 뜻입니다.
비스무트에서 신뢰는 프라이빗 컨트랙트(어플리케이션 소스코드)를 공개하고 컨트랙트 이용자가 해당 컨트랙트의 트랜잭션들을 장기간 관찰함으로써 높아질 수 있습니다.

백서는 비스무트의 핵심 기능을 일부 자세히 설명합니다. 더 많은 정보를 원하는 독자에게는 관련 링크와 참고 자료를 소개합니다.

Disclaimer

이 백서의 내용은 순전히 정보 제공 목적이며, 특정한 투자나 금융 조언을 담고 있지 않습니다. 투자 결정을 하기에 앞서 스스로 충분한 조사를 하시기 바랍니다. 이 문서에 담긴 정보는, 암호화폐 투자와 관련해서, 이에 의존하거나, 구매 추천, 제안, 권장 또는 비추천 하는 내용을 다루고 있지 않습니다. 암호화폐 투자는 가격 변동성이 매우 심하고 위험도가 높습니다. 투자 손실을 감당할 수 있는 금액 이상으로는 투자하지 마시기 바랍니다.

문서 수정 이력

- 2019년 4월 3일: 백서 v1.0 공개