SASEUL Origin Whitepaper ver 1.1.0

Written by Jiwon Kim, Sunghwan Shin, Jungwoo Lee

나카모토 사토시가 비트코인으로 블록체인의 시대를 연 뒤로, 블록체인 기술은 발전을 거듭했다. 화폐의 가치만 부각한 비트코인과 달리 이더리움은 프로그래밍 언어를 내장한 블록체인으로 계약을 자유자재로 사용해 토큰, 파생상품 등의 금융 시스템을 구현했다. 또한 이더리움을 비롯해 스마트 콘트랙트 기능을 장착한 2세대 블록체인은 금융계를 넘어서 본인인증 시스템, 분산형 데이터 저장소 등으로 쓰임을 확장해 나갔다.

블록체인이 화폐를 넘어서 금융 시스템, 인증 시스템, 데이터 저장소 등으로 영역을 확장해 나감에 따라, 수요가 폭발적으로 증가하고 있으나 이더리움을 비롯한 지금의 기성 블록체인들은 합의 시스템이나 채굴 시스템이 가지는 근본적인 기술적 한계 때문에, 요구하는 역할을 100% 수행하지 못하고 있다.

위에서 이야기한 문제를 해결하기 위해, 비트코인과 이더리움 등 기존 주류 블록체인들이 가진 기술적 한계들을 극복하고 새로운 합의 알고리즘을 채택한 블록체인 시스템 SASEUL Origin을 제안한다.

목차

Table of Contents

1. 용어와 선행지식(Notations and Preliminary)

사슬 오리진과 여러 가지 블록체인 구조를 설명하고 장, 단점을 비교하기 위하여 이 문서에서 사용되는 용어들의 의미를 먼저 통일하고 필요한 개념들에 대해 간단히 정의하겠다.

1.1 순서 집합과 체인 구조

Definition 1.1.1 부분 순서 집합(Partially Ordered Set)

부분순서 집합은 다음 공리들을 만족하는 오브젝트와 순서 관계들의 집합을 말한다.
A partially ordered set(POset) is a collection of objects and relations with the following axioms.

- a. Reflexivity

  A는 A 자기 자신과 순서 관계를 가진다. (A >= A)

- b. Antisymmetry

  A가 B와 순서 관계를 가지고 B가 A와 순서 관계를 가진다면 A와 B는 같다. (A>=B, B>=A 이면 A=B)

- c. Transitivity

  A가 B와 순서 관계를 가지고 B가 C와 순서 관계를 가진다면 A와 C가 순서 관계를 가진다. (A>=B, B>=C이면 A>=C)

Definition 1.1.2 전 순서 집합(Totally Ordered Set)

전순서 집합은 다음 공리들을 추가로 만족하는 부분순서 집합을 말한다.

A totally ordered set(TOset) is a partially ordered set with the following additional axiom.

- a. Connex Property

  모든 A, B 사이에 A와 B가 순서 관계를 가지거나 B와 A가 순서 관계를 가진다. (모든 A, B에 대해 A>=B 또는 B<=A)

즉, 전 순서 집합은 우리가 알고 있는 일반적인 순서로, 모든 원소가 비교 가능한 순서 집합을 말한다.

부분순서 집합 중 서로 비교 가능한 원소들만 모으면 전 순서 집합이다. 이 경우 전순서 집합은 부분순서 집합의 부분집합이라 생각할 수 있으며, 이 경우의 전순서 집합은 체인이라고 부른다.

1.2 트랜잭션과 블록체인

Definition 1.2.1 트랜잭션(Transactions)

이 문서에서 네트워크 위의 트랜잭션이란, 다음의 정보들을 가지고 있는 집합으로 정의한다

In this paper, a transaction(Tx) over a network is a set which consists of the following data sets

```
Tx = {d, P, S}
```

- a. Transaction data(d)
- b. Public key(P)
- c. Signiture(S)

Definition 1.2.2 블록(Blocks)

블록(B)은 다음의 정보들을 가지고 있는 집합이자, 부분 집합 순서의 오브젝트로 생각할 수 있다.

```
B = {Tx, F, D, t, S, >=}
```

- a. Transactions(tx)
- b. Former Block data(F)
- c. Block data(D)
- d. Timestamps and Signiture(t,S)

Definition 1.2.3 블록체인(Blockchain)

블록체인은 블록들을 오브젝트로 가지고, 블록들의 선후 관계를 순서 관계(Order relation)로 가지는 부분순서 집합이다.

1.3 노드와 합의

Definition 1.3.1 노드 (Node)

노드(N)는 데이터셋을 가진 그래프의 꼭짓점으로, 블록체인 데이터를 포함할 수 있다.

A node N is vertex of graph with data sets, which can contain Blockchain.

Definition 1.3.2 네트워크

네트워크는 노드들을 그래프의 꼭짓점으로 하는 그래프로 정의한다.

네트워크 전체의 블록체인은, 각 노드가 가진 블록체인들의 모든 오브젝트와 순서 관계의 합집합으로 정의한다.

네트워크에 참여하는 노드들은 반드시 같은 최초의 블록(Genesis Block)을 공유하므로 네트워크 전체의 블록체인도 연결(Connect)된 부분순서 집합이라고 생각할 수 있다.

Definition 1.3.3 합의 (Consensus)

합의는 일반적으로 네트워크에 참여하는 모든 노드가 하나의 거래 내역을 공유하는 것을 이야기한다.

이 문서에서 합의는 네트워크의 블록체인 중 최초의 블록(Genesis Block)을 포함하는 체인, 즉 최초의 블록을 포함하는 전 순서 집합을 유일하게 선택하는 것으로 정의한다.

2. 기존 블록체인 기술들의 의의와 한계

2.1 비트코인과 작업증명시스템

비트코인은 암호화 기술에 기반한 전자 거래 시스템이다. 이 시스템은 노드(Node)라 부르는 네트워크상으로 연결된 거래 주체들로 이루어져 있으며 노드들이 가진 분산 장부인 블록체인을 통해 거래를 확인한다. 거래에 참여하는 각 노드는 거래를 모아 블록으로 만든 뒤 분산된 장부의 상태(State)를 모두 같게 만드는 합의(Consensus)하는 과정을 거쳐 분산 장부의 신뢰를 보증한다. 특별히 비트코인 시스템에서는 거래에 참여하는 노드들의 계산력을 이용해 분산 장부를 검증하는 작업증명방식(Proof-of-Work)을 통해 합의한다.
비트코인의 장부의 상태(State)란 다음의 정보들을 가지고 있는 데이터 뭉치를 의미한다.

- a. 각자의 코인 금액
- b. 암호화된 공개키 정보

비트코인에서 각 노드의 거래 정보는 트랜잭션(Transaction, Tx)의 형태로 보내진다. 비트코인의 트랜잭션은 상태(State)를 변화시키는 함수로 해석할 수 있다. (주, 이더리움 백서, 비트코인 백서) 이 트랜잭션은 다음과 같은 정보들을 포함한다.

- a. 보내는 쪽 지갑의 상태 (하나 이상의 지갑이 될 수 있다)
- b. 해당 지갑 주소에 대응되는 개인 키가 생성한 암호 서명

- c. 받을 쪽 상태에 추가될 새로운 데이터 뭉치

이미 만들어진 블록의 거래 내역은 비트코인의 암호화 알고리즘에 의해 보호받는다. 그러나 악의적인 공격자들이 자신들의 이득을 위하여 이중지불 공격을 하기 위해 변조한 블록을 생성하는 경우, 어떤 블록이 제대로 된 블록인지 구별하는 과정이 필요하다. 이것을 하기 위해 작업증명이라는 합의 알고리즘이 도입되었다.

작업증명방식은 블록을 생성하기 위해 일정량의 계산력(CPU 파워)을 요구한다. 구체적으로 요구하는 연산은 다음과 같다.

- a. 복호화 불가능한 해시 함수인 SHA-256을 이용해 블록의 내용을 암호화한다.
- b. 암호화한 해시 함수의 결괏값이 목표한 값 이하가 나오는지 확인한다.
- c. 목표한 값이 나오면 블록을 생성하고, 목표한 값이 나오지 않는 경우 블록 머리의 논스 값을 바꿔서 일련의 과정을 반복한다.

SHA-256은 예측 불가능한 유사 난수 함수로서 어떤 값이 나올지 예측을 하거나 역산을 하는 것이 불가능하다. 따라서 값을 하나씩 넣어보는 원시적인 방법(Brute Force)을 사용할 수밖에 없다. 결과적으로 이 알고리즘은 블록이 생성될 가능성을 확률적으로 노드의 계산력(CPU 파워)에 비례하게 만든다.
합의는 가장 긴 블록들의 체인을 옳은 블록이라고 인정하면서 마무리하는데, 악의적인 공격자가 자신의 송금 내역을 뒤집으려면 거래가 일어나기 전 블록부터 분기(Fork)를 내어 자신의 거래 내역을 무효화 한 블록을 등록한 뒤 현재의 블록들보다 더 많은 블록을 연결해 가장 긴 체인으로 만들어야 한다.

블록이 생성될 가능성은 노드의 계산력에 비례하므로, 공격자 노드의 모임이 과반의 계산력을 얻지 못하면 악의적인 공격자의 공격은 성공하지 못한다. 따라서 비트코인이 유지되기 위해서는 블록을 생성하려는 노드들이 지속해서 계산력을 제공해 주어야 하며 이를 위해 비트코인은 처음으로 블록을 생성하는 사람들에게 일정량의 코인을 제공해주는 알고리즘으로 계산력을 제공하는 마이너를 모집한다.

작업증명방식을 이용하면 블록체인, 즉 장부에 담긴 가치가 커질수록 이중지불 공격이 성공할 때 보상도 더 커지므로 공격 가능성도 늘어난다. 따라서 비트코인은 가치가 늘어날수록 불가피하게 블록 생성 계산의 난이도를 증가시키며 더 많은 계산력을 요구하게 된다.

비트코인은 전체 계산력을 계산해 블록이 약 10분에 한 개가 생성되도록 난이도를 조정한다. 확률적으로 약 5개 정도의 블록이 생성되면 거래를 신뢰할 수 있는데, 실제 거래를 위해 5개 정도의 블록이 생성되는 것을 기다리면 한 시간이 걸린다.

비트코인은 신뢰받는 제삼자 없이 돌아가는 전자 거래 시스템을 처음으로 구현했다는 의미를 가진다. 기존의 전자 거래 시스템, 또는 화폐 시스템이 전적으로 화폐를 보증하는 기관의 신뢰와 선의에만 의존한다는 점을 생각하면, 이는 분명한 진보라고 할 수 있다. 그러나 실제 거래의 확정에 한 시간 정도의 시간이 걸린다는 점은 현실 거래에 분명한 한계로 작용한다.

또한 많은 계산력에 많은 보상을 주는 비트코인의 특성상, 비트코인의 가치가 실제로 커짐과 더불어 엄청난 전력을 소모하는 마이닝 풀들이 여럿 등장했다. 이들이 만드는 해시 계산은 비트코인 시스템을 유지하는 것 외에 어떠한 가치 있는 계산도 하지 않으나 이들은 해시 계산을 위해 2019년 현재 전 세계 전력에 0.3 %에 육박하는 막대한 자원을 소모하고 있다.

비트코인은 블록을 처음으로 생성하는 데 성공한 마이너에게만 보상을 준다. 블록 생성 실패라는 리스크를 감수하기 위한 마이너들은 마이닝 풀이라는 연합을 구성해 보상을 분배하기 시작했고, 현재 비트코인 해시 계산의 대다수는 소수의 마이닝 풀에 의해 이루어진다. 이는 과반의 해시 파워를 장악해 장부를 바꾸는 이중지불 공격에 취약해지고 있음을 의미한다. 이러한 문제들을 해결하고, 비트코인이 보여준 블록체인 구조의 장점을 활용하고자 비트코인 이후 다양한 2세대 블록체인들이 등장하였다.

2.2 이더리움과 스마트 콘트랙트

이더리움은 스마트 콘트랙트 등으로 무장한 2세대 블록체인의 선발 주자이다. 화폐의 역할이 부각된 비트코인과 달리 이더리움은 대규모 분산 애플리케이션 제작을 위한 대체 프로토콜을 만드는 것을 목표로 하였다. 이를 위해 이더리움은 튜링-완전(Turing-Complete)한 프로그래밍 언어를 내장하고 있는 블록체인을 제공한다.
이더리움의 상태(State)는 계좌(Account)라는 오브젝트들로 이루어져 있으며 20 Byte의 주소와 다음의 필드를 가지고 있다. (주, 이더리움 백서)

- a. 논스(Nonce), 트랜잭션이 한 번만 처리되게 하는 카운터)
- b. 계좌 잔고
- c. 계약(Contract) 코드
- d. 저장 공간

이더리움의 계좌는, 개인 키에 의해 통제되는 외부 소유 계좌(Externally owned Accounts)와 계약 코드에 의해 통제되는 계약 계좌(Contract Accounts)로 나뉜다. 외부 소유 계좌의 경우 계약 코드 필드가 비어있으며, 새로운 트랜잭션을 만들고 서명함으로 메시지를 전달한다. 계약 계좌의 경우 메시지나 트랜잭션을 받을 때만 동작하는데, 메시지를 받을 때마다 자신의 코드를 활성화하고, 코드에 따라 메시지 수신, 발신 등의 역할을 하며 다른 계약(Contract)들을 차례로 생성한다.

이더리움의 트랜잭션은 다음과 같이 구성되어 있는데, 이 구조를 이해하기 위해서는 가스(Gas)라는 항목을 이해할 필요가 있다.

- a. 메시지 수신처
- b. 발신자의 서명
- c. 이더의 양
- d. 데이터 필드
- e. 시작 Gas 계약 코드를 수행하기 위한 자원
- f. 가스의 가격- 매 단계 발신처가 지불하는 수수료

가스(Gas)는 이더리움에서 계약을 수행하기 위한 자원으로, 지나치게 많은 계산을 요구하게 되는 코드나, 무한 루프가 일어나는 코드가 등장하는 경우를 방지하기 위해 만들어졌다. 이더리움 가상 머신(Ethereum Virtual Machine, EVM)에서 코드의 수행은 메시지나 트랜잭션 발신자의 가스(Gas)의 양만큼 일어나게 되며, 이 가스가 다 떨어지게 되면 자동으로 코드의 수행이 종료되게 된다.

비트코인같이 무한루프 문제를 해결하기 위해서, 반복문의 수행을 금지할 수도 있다. 이렇게 되면 튜링-완전성(Turing Completeness)가 깨지는데 (주, 튜링 완전 언어는 튜링 머신이 할 수 있는 모든 계산을 수행하는 언어를 말한다) 이 튜링 불완전성은 발전된 애플리케이션을 만드는 데 장애로 작용할 수 있으며, 단순한 계약을 수행하는 코드의 비효율성도 일으키기 때문에, 이더리움은 튜링-완전하면서 무한 실행에 빠지지 않는 언어를 만들기 위해 더 복잡한 구조인 가스를 도입한 것이다.

트랜잭션 간에는 메시지를 전달할 수 있으며 이 메시지는 따로 저장되지 않는 가상의 오브젝트로 다음과 같이 구성된다.

- a. 메시지 발신처
- b. 메시지 수신처
- c. 메시지와 함께 전달되는 이더
- d. 선택한 데이터 필드
- e. 시작 Gas

메시지는 콘트랙트에 의해 생성된다는 것을 제외하면 콘트랙트와 거의 유사하다. 이것은 외부의 실행자에 의해 생성되는 트랜잭션과 같은 방식으로 다른 콘트랙트와 관계를 맺는다.

이더리움의 합의 방식은 비트코인과 근본적으로 유사한 작업증명방식(Proof of work)이나 비트코인과 완전히 같지는 않다. 가장 중요한 차이는 이더리움 블록이 상태(State)복사본을 가지고 있다는 점이다. 이더리움은 현재 블록의 작업 증명이 유효한지, 블록에 들어가 있는 블록 넘버, 루트, 가스 제한 등이 유효한지 확인 후, 트랜잭션을 받아 블록의 상태를 변환한다. 즉, 블록을 검증하고 트랜잭션이 각 상태를 변환해주는 상태변환함수로 작동하게끔 한 후 블록을 생성한다.

이더리움은 이더와 가스라는 자원을 가지는 한, 어떤 노드든 스마트 콘트랙트를 만들어 이더리움 가상 머신을 실행할 수 있으나 근본적인 구조의 복잡함으로 인해 사용자가 실제로 이를 구현하기 위해서는 솔리디티 등의 이더리움 기반 언어를 알고, 이더리움의 구조를 이해해야 한다는 진입장벽을 느끼게 된다. 또한 이더리움은 작업증명방식을 이용하므로, 동작에 지나치게 많은 계산력을 요구하며, 애플리케이션 구현에 필요한 속도 또한 구현할 수 없다. 이상의 단점들을 극복하기 위하여 합의 방식 자체를 바꾼 블록체인들이 등장하게 되었다.

2.3 지분증명방식의 블록체인들

지분증명방식(Proof of Stake)의 블록체인은 기존의 작업증명방식(Proof of Work)의 블록체인들이 가지고 있는 근본적인 효율성의 한계를 극복하기 위해 만들어진 합의 방식을 이용한 블록체인으로 합의를 위해 막대한 계산력을 요구하지 않는다.

지분증명방식의 블록체인은, 블록을 지분에 비례해 생성할 수 있게 하는 방법으로, 이 방식의 블록체인에선 특별한 노드인 노드가 등장한다. 이 블록체인에선 블록을 만들고 보상을 받기 위해 지분(Stake) 또는 예치금만 유지하면 많은 계산력을 소모하지 않고도 보상을 받을 수 있는데 이렇게 금액을 예치하고 트랜잭션을 직접 처리해 만드는 노드를 Validator 노드라고 한다.

지분증명방식을 사용할 경우, 작업증명방식을 사용할 때 생각하지 않았던 문제들을 생각해야 한다. 지분증명방식을 사용하는 경우 태초(Genesis) 블록부터, 또는 블록체인의 어딘가부터 모두 변조해 블록을 이어 나가는 장거리 공격(The long range attack consensus problem)에 취약할 수 있다. 작업증명방식에서 지금까지의 블록을 모두 변조하는 것은 산술적으로 불가능한 양의 계산을 요구하므로 확률적으로 불가능한 것이었으나 지분증명방식에서는 블록을 변조하는데, 계산력이 필요하지 않기 때문에 51% 이상의 지분을 가진 노드들이 연합하면 장부 자체를 뒤엎는 장거리 공격을 할 수 있다.

비트코인이나 이더리움이 작업증명을 하는 채굴자들의 “선의”에 기대하지 않는 것처럼 작업증명방식의 블록체인들도 지분을 많이 가진 Validator 노드들의 호의에만 의존하지는 않는다. 지분증명방식은 지분이 많을수록 공격을 당해 신뢰가 떨어질 때 장부 자체의 가치가 떨어져 얻는 손해가 크다는 점에 기인하는데, 장부 전체의 가치를 잴 수 있는 도구가 부족하기 때문에 이 방식만으로는 한계가 있다. 그래서, 지분증명방식은 지분을 많이 가진 Validator들의 일탈을 막기 위해 위임 지분증명방식, 랜덤 지분증명방식 등의 새로운 구조를 덧입히거나 작업증명방식과 지분증명방식을 섞어서 이용하는 등의 보완책과 같이 사용하는 것이 일반적이다.

예를 들어 이오스 등에서 사용하는 위임 지분증명방식(DPoS)은 제한된 일정 수의 Validator를 뽑아 라운드(round)마다 블록을 만드는 시스템이다. 네트워크 사용자들이 네트워크상에서 보유한 토큰(지분 수)에 따라 투표를 하여 Validator들을 뽑고, Supervisor를 통해 블록을 제대로 만드는지 감시한 후, Validator들을 해고하거나 계속 고용할지 정하는 방법을 사용한다.

텐더민트의 경우 포크 책임(fork-accountability)과 제안자(Proposer)라는 개념을 도입했다. 각 라운드에는 블록을 제안하는 라운드 리더(round-leader), 즉 제안자(proposer)가 있으며 Validator들은 제안된 블록을 받아들일 것인지 또는 다음 라운드로 넘어갈 것인지에 대한 단계별 투표를 진행한다. 각 라운드의 제안자는 검증인 리스트에서 투표권에 비례해 선택된다.

사슬 오리진의 경우 지분증명의 한계를 해결하기 위한 Validator Supervisor 라운드 시스템을 도입했지만, 지분증명방식과는 다른 새로운 합의 방식을 사용함으로써
속도와 확장성, 안정성을 가진 블록체인 네트워크를 구현했다.

3 사슬 오리진(SASEUL Origin)

3.1 개괄

사슬 오리진은 기존의 블록체인들이 이용했던 합의 알고리즘인 작업증명방식, 지분증명방식보다 더 진보된 합의 알고리즘인 사슬합의방식을 적용한 블록체인이다. 구체적으로 사슬 오리진은 다음과 같이 구동된다.

사슬 오리진은 구체적으로 다음과 같은 네 종류의 노드들로 구성된다.

a. 라이트 노드(light node)

라이트 노드는 단순지불검증(Simple Payment Verification, SPV) 시스템을 이용한 노드로, 블록체인 데이터 전체를 가지고 있지 않으며 단순 트랜잭션(요청, Request)을 일으키는 역할만 하고 블록을 만들고 합의하는 과정에는 참여하지 않는다. 라이트 노드는 블록체인 데이터를 가지고 있지 않기 때문에 트랜잭션을 일으키기 위해 다른 Supervisor나

Validator에게서 자료를 요청해야 하며, 트랜잭션을 일으킨 뒤 이를 Validator 노드에 보낸다. 실제 암호화폐 거래에서 개인 지갑에 해당하는 역할을 한다.

b. Supervisor

Supervisor와 Validator는 일정 기간 이후의 블록 데이터를 가지고 있으며 트랜잭션 데이터를 전달받아 블록 생성과 합의에 직접 참여하는 노드들이다. Supervisor 노드는 블록을 실제로 생성하지는 않으나 Validator 노드들이 제대로 된 블록을 생성하는지 검증하는 역할을 한다. Supervisor 노드 중 후술할 일정 조건을 만족하면 Validator 노드가 될 자격을 얻는다.

c. Validator

Validator 노드는 Supervisor 노드 중 조건을 만족한 노드로 Validator 노드가 될 자격을 얻으며 Validator 노드들은 다른 노드에게 받은 트랜잭션을 묶어 블록을 생성하며 합의 과정을 통해 생성된 블록체인을 검증한다. 또한, Validator 노드는 계약에 사용할 코드를 업데이트할 수 있으며, 다른 Validator 노드가 만든 계약 코드를 승인할지 투표(Vote)할 수 있다.

d. 아비터 (Arbiter)

아비터는 Supervisor와 Validator 노드와 달리 블록 생성과 합의에 직접 참여하지 않는 노드이다. Validator 노드들이 만들어 놓은 블록 데이터를 저장하는 역할을 하며, 사슬 오리진의 버전 관리 역할도 겸한다.

사슬 오리진에서 발생하는 트랜잭션(Transactions, TX)은 다음과 같이 구성된다.

- a. 트랜잭션 타입 (Type)
- b. 트랜잭션 데이터 (Transaction data)
- c. 공개키 (Public key)
- d. 서명 (Signification)

사슬 오리진의 트랜잭션 데이터는 다음과 같이 구성된다.

i. 트랜잭션이 일어난 사슬 오리진의 버전(Version)

사슬 오리진은 네트워크의 버전 관리 시스템을 채택했으며, 이 버전 관리는 아비터 노드에서 일어난다.

ii. 보내는 노드의 주소와 받는 노드의 주소 (Address)

iii. 데이터 (Data)

메시지 등을 포함하는 부분으로 자율적으로 쓸 수 있다.

iv. 타임스탬프(Timestamp)

필수적으로 들어가야 하는 부분이며 트랜잭션을 생성한 서버 시간을 기록한다.

트랜잭션 내의 공개키와 서명은 트랜잭션 데이터의 유효성을 검증하는 데 사용된다. 사슬 오리진 내에서 비대칭 암호화에 사용하는 암호화 알고리즘은 Ed25519(타원곡선 암호화 알고리즘인 EdDSA의 한 종류)를 사용하며, 해시 함수는 비트코인에 사용한 것과 같은 SHA-256을 사용한다.
사슬 오리진에서는 서버의 원활한 가동을 위하여 이 트랜잭션들을 묶어 청크(Chunk)라는 단위로 처리하며, Validator들은 일정한 시간 단위(Round)마다 블록을 만들어 블록체인을 만든다.

각 노드의 자세한 역할과 기능을 뒤에서 설명한다. 각 Supervisor나 Validator를 선발하는 방법은 3.3 절에서 설명하도록 하며, Validator들이 합의하는 방식은 2.5절 합의 알고리즘에서 설명할 것이며, 의 역할과 버전 관리의 필요성은 3.4절, 3.6절의 스마트 콘트랙트 파트에서 설명할 것이다.

3.2 특성

사슬 오리진의 목적은 합의 방식을 근본적으로 바꾸어 비트코인이 주장했던 가치인 탈중앙화와 이더리움이 주장했던 확장성이라는 가치를 유지하면서 기존 블록체인들의 단점인 자원 소모와 속도 문제와 안전성 이슈 등을 해결한 새로운 블록체인 시스템을 구현하는 것이다.

사슬 오리진이 기존의 블록체인들과 차별하여 가지는 장점에는 다음과 같은 것들이 있다.

a. 속도(Speed)

블록체인의 속도는 다음 두 가지 척도로 측정한다.

i. 합의 지연속도(Commit latency)

합의 지연속도는, 트랜잭션이 일어나고 합의되기까지 걸리는 시간으로, 종결성(finality)을 보장하기까지 걸리는 시간을 의미한다. 합의지연속도 이후에는 거래 내역이 변경되지 않는다.

ii. 초당 트랜잭션 처리량 (Transactions Per Second, TPS)

초당 트랜잭션 처리량이란, 네트워크가 1초에 처리할 수 있는 최대 트랜잭션의 수를 의미하며 순간적으로 초당 트랜잭션 처리량을 넘는 트랜잭션이 발생하는 경우 지연이 일어나게 된다.

앞에 서술한 비트코인의 경우 합의 지연 속도가 평균적으로 한 시간이며 최대 초당 트랜잭션 처리량은 10 정도이다. 이는 비트코인의 합의 방식인 작업증명방식(PoW)의 근본적인 한계이며, 작업증명방식을 이용한 이더리움의 경우에도 이 문제를 피해갈 수 없다.

사슬 오리진은 새로운 합의 방식을 이용함으로 인해 0.5초의 합의 지연 속도를 가지며, 1,000 이상의 초당 트랜잭션 처리가 가능하다.

b. 확장성(Scalability)

작업증명방식의 블록체인들은 규모가 커짐에 따라 보안 유지에 필요로 하는 계산력이 같이 증가하고 TPS 등의 속도는 계산력의 향상과 무관하다. 즉 작업증명방식의 블록체인은 아무리 많은 계산력을 투자하더라도 성능의 향상을 담보할 수 없는 것이다. 사슬 오리진은 보안 유지에 많은 계산력이 필요하지 않으므로 성능을 향상하기 위해 다중 블록체인 구조를 채택할 수 있다. 사슬 오리진은 이를 위해 다중 블록체인 구조를 테스트 중이다.

c. 안전성(Safety)

블록체인 시스템에 가해질 수 있는 공격은 합의된 체인을 바꿔버리는 이중지불(Double spent)이 있다. 블록을 위조, 변조하는 방식의 공격은 암호화 알고리즘에 의해 보호받기 때문에 불가능하다.

이중지불 공격은 다른 거래 내용이 들어있는 블록을 생성해 네트워크가 현재 체인과 다른 체인을 고르도록 해 트랜잭션 내용을 무효화 하는 공격으로 정의할 수 있으며 장거리 공격은 일정 시점 이후부터의 블록을 모두 변조해 트랜잭션 내용을 무효화하는 공격으로 정의할 수 있다.
사슬 오리진 시스템의 경우, 합의를 타임스탬프라는 절대적인 기준에 의존하기 때문에, 위조하지 않는 이상 네트워크가 현재 체인과 다른 체인을 고를 수 없다. 즉 이중지불 공격에서 완전히 안전함을 의미한다. 사슬 오리진 시스템은 몇몇 악의적인 행동을 하는 Validator 노드가 존재해 노드 간 블록의 불일치가 일어난 경우 자동으로 네트워크를 분리(Fork)한다. 이 경우, 옳은 네트워크를 고르는 방법은 정책(Policy)파트에서 서술하도록 한다.

3.3 Supervisor와 Validator

Supervisor와 Validator는 일정 시점 이후의 블록 데이터를 가지고 있으며 트랜잭션 데이터를 받는 노드들로 블록체인 시스템에서 Supervisor와 Validator 노드를 구동하게 하려면, 보상이 필요할 수 있다. 사슬 오리진에서는 Supervisor와 Validator 노드의 역할만 정의할 뿐, Supervisor와 Validator 노드를 선발하는 방법이나 Validator 노드에 대한 보상을 정해 놓지 않고, 사슬 오리진이 이용될 네트워크의 특성에 따라 Validator 선발 정책(Policy)과 보상 정책을 유연하게 선택할 수 있게 했다. 여기서는 사슬 오리진 시스템이 쓰이는 목적에 따라, 어떤 선발 정책과 보상 정책을 사용할 수 있는지 소개하고자 한다.

첫 번째로는 지분증명방식(Proof of Stake)의 아이디어를 가져오는 것이다. Supervisor 노드 중에서 일정 금액 이상의 금액을 예치(Stake)한 노드만 Validator로 인증하며, 처리한 트랜잭션에 해당하는 수수료를 예치한 금액에 따라 Validator들에게 나누어 주는 방식이다.

이 방식을 이용하면 Validator 노드가 트랜잭션을 왜곡하거나 변조해 포크가 일어나는 경우, Validator의 예치금을 통해 책임을 물을 수 있다. 구체적으로, 각 Validator가 처리할 수 있는 트랜잭션의 총량을 예치금 이하로 제한해 손해가 일어난 금액을 Validator의 예치금 지갑(라이트 노드)에서 제하는 방법이 있다.

사슬 오리진 팀에서 테스트하는 모델의 경우 기본 거래 수수료는 0.1%로 설정되어 있다. 거래 수수료는 Validator 노드의 수에 따라 유동적으로 변경되며, 초기 거래 수수료와 최솟값 또한 사슬 오리진의 정책 책정에 따라 변동할 수 있다. 어떤 경제정책 모델을 짜야 할지 합리적인 모델은 5.1 경제학적 이슈에서 제시한다.

두 번째로는 권위증명방식(Proof of Authority)의 아이디어를 가져오는 방법이 있다. Validator 노드로 선발하기 위해서 신원이나 보증을 요구하는 정책으로 구체적으로 Validator 노드 간의 만장일치가 있다면 신원을 인증한 Validator를 새로 Validator 노드로 추가하는 정책을 사용할 수도 있고, 2/3 또는 과반의 Validator 노드가 동의할 경우 새로운 Validator 노드를 선발하는 정책을 사용할 수도 있다. 또는, 필요에 따라 관리자 노드가 Validator 노드들을 선발하는 정책을 사용해도 된다. 이 방식은 프라이빗 블록체인(Private Blockchain)과 같이, 공개되지 않은 네트워크에서 이용되는 정책으로, 퍼블릭 블록체인(Public Blockchain)같이 공개된 네트워크에서는 Validator들의 신원을 확인하기가 어렵고, 개인 정보들이 모두 공개되기 때문에 다른 정책을 사용하는 것이 좋다.

이러한 정책(Policy)은 사슬 오리진의 계약 코드를 이용해 정책을 투표할 수 있는 시스템을 업로드 함으로써 채택할 수 있다. 정책 채택에 관한 자세한 사항은 5.2절 거버넌스와 정책 파트에서 이어서 설명한다

3.4 아비터 시스템과 버전 관리

아비터 시스템(Arbiter System)은 Supervisor와 Validator의 거래 내역을 주기적으로 받고 사슬 오리진 네트워크의 버전 관리를 도와주는 아비터를 도입하는 시스템으로 사슬 오리진에서 처음 제시하는 시스템이다.

또한 아비터 시스템은 사슬 오리진의 버전 관리 시스템의 역할을 한다. 사슬에는 이더리움의 가스와 같이 연산의 양 또는 무결성을 강제로 제한하는 코드가 시스템이 없다. 따라서 잘못된 코드가 승인될 가능성을 항상 가지며 이 위험을 아비터 시스템을 통해 관리하여야 한다.

아비터 시스템은 Validator 노드들이 만든 소스 코드를 버전에 따라 모두 저장하며 네트워크의 요구가 있을 시 지난 버전의 소스 코드를 제공한다. Validator들이 무한루프나 지나치게 연산 시간이 긴 소스 코드를 승인하여 네트워크가 다운되거나 충돌이 나는 소스 코드를 승인해 분기(Fork)가 일어나는 경우, 저장한 코드를 가지고 이전 버전으로 돌아갈 수 있도록 긴급 복구하는 역할을 한다.

사슬 오리진에서 아비터 시스템을 구현하는 방법은 아비터 노드 도입과 분산형 데이터베이스 도입의 두 가지가 있다.

아비터 노드를 도입해 시스템을 구현하는 방법은 구체적으로 다음과 같다. 신뢰할만한 아비터 노드들을 도입해 버전 관리 권한을 주고, 지금까지의 거래 데이터들을 모두 저장하는 역할을 맡기는 방법이다. 이 경우, 적절한 보상을 통해 시스템에 노드를 유지할만한 유인을 주는 방안이 필요하다.

두 번째는, 4.2에서 서술할 데이터 분산 저장 시스템을 이용하는 방법이 있다. 일정 라운드가 지나면 데이터를 조각내 각 Supervisor와 Validator 노드들에 분산해 저장하고 필요한 경우 다운받아서 확인하는 방법이다.

3.5 합의 알고리즘(Hypothesis Algorithm Protocol-2, HAP-2)

Validator 노드 그룹(validator group)은 합의 알고리즘(Hypothesis Algorithm Protocol-2, HAP-2)을 기반으로 트랜잭션의 승인 및 거절에 대한 합의를 생성한다. Validator 노드들은 각각의 트랜잭션에 대한 승인 및 거절 여부를 자체적으로 판단한 뒤, 일정 개수의 트랜잭션을 모아 다른 Validator와 판단 결과를 비교한다. 즉, 모든 개별 트랜잭션 요청에 대하여 판단 결과를 공유하고 동기화(Synchronize)하는 대신 네트워크에서 정한 단위의 트랜잭션들을 한 번에 비교한다. 이러한 트랜잭션의 묶음을 청크(chunk)라고 부르며, 청크의 비교는 해시값 비교를 통해 이루어지게 된다.

1. 트랜잭션 요청 확인(transaction request configuration)
   라이트 노드에서 생성된 transaction에는 네 개의 요소가 필수적으로 포함되며, 서비스 등의 특성에 따라 추가적인 정보가 포함된다.

    - 트랜잭션 데이터(transaction data)

      트랜잭션 데이터 안에는 트랜잭션이 수행해야 할 동작에 대한 명세가 포함된다. 송금 트랜잭션의 경우, 보낼 계좌, 받을 계좌, 금액 등이 포함된다.

    - 타임스탬프(timestamp)

      해당 트랜잭션 요청이 생성된 시간이다.

    - 서명(signature)

      트랜잭션 데이터와 타임스탬프, 개인 키(private key)를 이용해 생성한 서명이다. SASEUL Origin에서는 트랜잭션의 해시값에 서명하는 방식을 이용한다.

    - 공개키(public key)

      개인 키(private key)로 만든 서명의 위변조를 확인할 수 있는 공개키를 함께 전달한다. 모든 노드는 공개키(public key)를 기반으로 주소를 생성하므로, 트랜잭션 요청을 생성하고 전송한 노드의 주소를 기반으로 공개키를 신뢰할 수 있게 된다.

2. 트랜잭션 검증(transaction validating)

   각각의 Validator는 트랜잭션 요청(request)을 승인할지 거절할지 개인적으로 결정하여 합의를 위한 요청 처리 결과를 생성한다. 요청의 승인 여부는 아래 두 가지의 기준으로 결정된다.

    - 본인이 생성한 트랜잭션인가 ?

      트랜잭션 요청 안에는 개인 키로 생성한 서명이 포함되어 있다. 공개키를 이용하면 서명의 위변조 여부를 확인할 수 있으며, 공개키의 신뢰 가능 여부는 해당 트랜잭션 요청을 생성하고 요청한 노드의 주소로부터 확인할 수 있다.

    - 유효한 트랜잭션인가?

      이전 단계에서 본인이 만든 트랜잭션이 맞는 상황에 해당 트랜잭션이 유효한지 검증하게 된다. 예를 들어, 가장 범용적인 암호화폐 네트워크에서의 트랜잭션의 경우 보내는 사람의 잔액이 트랜잭션을 수행하기에 충분한지, 받는 사람의 지갑 주소가 유효한지를 확인한다.

3. 라운드 확인(round validating)

    1. 라운드 동기화(round synchronizing)

       각 Validator 노드들은 본인들이 연산하고 있는 블록체인의 현재 체인 길이를 저장하고 있으며, 이를 라운드라고 부른다. 즉, 해당 체인 life cycle 시작으로부터 몇 번째 블록에 대한 연산이 수행 중인가 하는 상태를 라운드로 저장하게 된다. 각각의 Validator들은 다른 Validator와의 합의를 위해, 서로가 몇 번째 라운드를 연산 중인지 확인한다. 확인은 현재 자신의 라운드를 브로드캐스트(broadcast)하는 방식이 아니라, 자신을 제외한 다른 Validator 노드들의 라운드를 요청한다.

라운드 체크 결과 본인보다 앞선 라운드를 진행하고 있는 Validator 노드가 존재하는 경우, 해당 Validator 노드에 현재 라운드 이전까지의 모든 데이터를 동기화 받아 검증한다. 복수의 Validator 노드가 본인보다 앞선 경우, 가장 앞선 하나의 Validator 노드로부터만 데이터를 동기화 받아 검증한다. 잘못된 데이터가 포함된 경우 해당 데이터를 전달한 Validator 노드를 트래커(tracker)에서 삭제한다.

본인보다 앞선 라운드를 진행하고 있는 Validator 노드가 없는 경우, 자신과 같은 최신 라운드를 수행하는 다른 Validator들과 함께 새로운 라운드의 블록을 생성하고 커밋한다.

> Q. 왜 라운드 정보를 브로드캐스트하지 않고 요청하나요?
>
> 네트워크를 통해 해시나 요청 전달되는 시간 동안 각 Validator 노드는 계속해서 트랜잭션 요청을 수신하게 된다. 네트워크상의 정보적/시간적 거리는 무시할 수 없으므로, 결과적으로는 모든 Validator 노드의 완벽한 real-time 동기화는 불가능해진다. 이러한 네트워크 상황 속에서 Validator들이 각자 라운드 종료 시마다 브로드캐스트 한다면 Validator 노드가 서로의 라운드 완료를 인지하는 시점이 모두 달라질 수밖에 없으며, 결과적으로는 특정한 라운드에 대해 합의를 할 시간과 라운드를 특정할 수 없게 된다.
>
> Validator 노드들이 라운드의 합의에 있어 real-time 동기화를 이루지 못한다는 것은 곧 모든 Validator 노드가 합의를 생성할 시간이 특정될 수 없음을 뜻하며, Validator 노드들이 투표(vote)하는 등 상호 작용을 통해 다 같이 합의를 생성할 환경이 구성될 수 없음을 뜻한다.
>
> 이를 해결하기 위하여, 사슬 오리진에서는 모든 Validator 노드들이 서로의 데이터를 공유하고 합의를 진행하는 대신, 각자 다른 Validator 노드들의 상황을 확인하고 자신이 인정하는 Validator 노드들만 follow-up 하는 형식을 취한다. 자신의 라운드 정보를 브로드캐스트하는 대신 다른 Validator 노드들의 라운드 정보를 요청함으로써, 자신보다 앞서서 블록을 생성한 Validator 노드를 발견하는 경우에는 선행된 데이터의 타당성을 검증하여 같은 네트워크상에 존속할지의 여부를 결정하며, 같은 단계를 합의하는 경우 의견이 같은지를 확인하여 같은 네트워크상에 존속할지의 여부를 결정한다.
>
> 이 과정에서 각 Validator 노드들은 서로의 의견을 하나로 모으기 위해 다른 Validator 노드에 압력이나 처벌을 가하는 것이 아니라, 자신과 다른 Validator 노드를 자신의 네트워크로부터 차단하게 된다. 이러한 방식을 사용하는 경우 선한 Validator 노드가 소수인 경우에도 해당 Validator 노드들이 살아남을 수 있게 된다.
>
> 이전까지의 합의 방식(i.e. PoS, PoW 등)은 합의에 동참한 Validator 노드의 수(혹은 지분)에 의존하여 옳고 그름을 판단하였기 때문에 비잔틴 장군 문제(Byzantine Generals' Problem)를 겪거나 51% 공격에 마땅한 대책을 찾을 수 없었으나, HAP-2의 경우 하나의 합의를 만드는 것이 아니라 다른 Validator 노드들을 각자 검증하는 형태를 취함으로써 하나의 순간에 데이터가 동기화되어야 하는 다른 합의 방식과는 다른 논리적 구조를 갖게 되었다. 그렇기 때문에 사슬 오리진(SASEUL Origin) 기반의 network에서는 올바른 Validator 노드가 51% 이하의 소수 그룹이 되더라도 각자 분기(fork)되어 살아남을 수 있으며, 라이트 노드들 또한 해시값 비교 등을 통해 소수의 선한 Validator 노드들을 follow-up 할 수 있게 된다.
>
> Q. 언제 라운드 정보를 요청하나요?

네트워크가 처음 생성될 때 정해진 규칙에 따라서 라운드 정보를 요청할 시간이 정해진다. 정해진 시간(각 네트워크의 규칙으로써 결정됩니다)이 되면, 모든 Validator 노드들이 해당 시간보다 *일정 시간(T_request)* 이전에 생성된 라운드 정보를 요청하게 된다.

일정 시간(T_request) 이전의 라운드 정보를 요청하는 것은 (이전에 설명한 바와 같이) Validator 노드 사이의 real-time 동기화가 불가능하기 때문입니다. 이 *일정 시간*은 Validator 노드들의 네트워크상 거리와 네트워크 상태에 의해 결정되며, 네트워크상의 거리를 고려하여 모두가 같은 시점에 대한 round 정보를 공유하기에 충분하도록 계산된다.

여기서 Validator 노드 간의 네트워크상 거리란 정보 요청을 전송한 후 라운드 정보를 수신할 때까지의 걸리는 시간적 거리를 뜻하며, 이 거리 중 가장 긴 시간적 거리(*T_max*)가 *일정 시간(T_request)*의 결정에 관여하게 됩니다. *일정 시간(T_request)*의 목적은 네트워크상에서 요청이 전송되는 시차를 고려하고도 같은 시점의 라운드 정보를 공유할 수 있도록 하는 것이므로, *T_request*는 *T_max*의 두 배(왕복에 걸리는 시간) 이상이어야 한다.

요약하자면, 라운드 정보 요청은 네트워크의 생성 단계에서 합의된 모종의 시간에 각 Validator 노드에 의해 생성되며, 해당 요청은 네트워크상에서 가장 긴 시간적 거리의 두 배 만큼 과거 시점의 라운드 정보를 요청하게 된다.

2. 블록 생성

   자신보다 앞선 라운드를 연산하고 있는 Validator 노드가 없다면 해당 라운드의 블록을 생성한다. 블록 생성은 각각의 Validator 노드에서 이루어지며, 생성된 블록에 대한 합의와 검증은 다음번 라운드 동기화에서 이루어지게 된다. 즉, 잘못된 내용을 포함하여 블록을 생성한 Validator 노드가 있다면, 다음번 라운드 동기화 때 올바른 네트워크 그룹의 tracker에서 삭제되어 네트워크로부터 분기된다.

3. 블록 커밋

   블록 생성 완료 후, 같은 라운드를 진행한 Validator 노드들끼리 블록 해시를 통한 투표를 진행하여 커밋 여부를 결정한다. 해시값이 다른 경우, 다른 해시값을 전달한 Validator 노드와 블록을 동기화하여 다른 부분을 점검하고 해당 부분에 대한 검증을 진행한다. 검증 결과에 따라서 블록을 업데이트하여 다시 커밋하거나 네트워크를 분기한다. 단, 블록 업데이트 이후에 다시 투표하지는 않는다. 즉 투표는 단 한 번만 일어난다.
   '

3.6 스마트 콘트랙트

블록체인 기술의 다양한 응용을 위하여, 블록체인에 들어가는 튜링-완전성(Turing-Complete)을 보장하는 소스 코드를 동작하게 하는 것은 필수적인 일이다. 사슬 오리진에서는 각 Validator들이 소스 코드를 업데이트할 수 있으며, 모든 Validator가 소스 코드의 업데이트를 승인하면 사슬 오리진이 새로운 버전으로 변하며, 아비터에게 지금까지의 소스 코드가 들어간다.

사슬 오리진의 경우 소스 코드의 완결성은 Validator들의 합의에 의존한다. 이더리움은 위에서 설명한 가스(Gas)라는 자원이 있는 이상 모든 계약 계좌(Contract Accounts)들이 메시지를 받음으로 소스 코드를 실행시켜 계약을 생성할 수 있으나 사슬 오리진의 경우 모든 Validator 노드의 합의가 없으면 새로운 계약을 생성할 수 없다.

이론적으로 이더리움은 계좌가 있는 한 누구든지 소스 코드를 짜서 계약을 생성할 수 있고 가스(Gas)라는 자원이 있는 이상 누구든지 계약을 실행할 수 있다. 하지만, 현실적으로 이더리움 가상머신(Ethereum Virtual Machine, EVM)을 이용해 계약을 생성하는 것, 거기에 가스 소모 감소를 최적화하는 계약을 생성하는 것은 아무나 할 수 있는 일이 아니다. 이러한 이

유로, 사슬 오리진의 경우 계약을 생성하는 것을 Validator 노드만 할 수 있도록 제한하고, 가스 소모같이 소스 코드의 비효율 및 복잡도를 증가시키는 요소를 제거할 수 있었다.

사슬 오리진의 경우, 튜링-완전 소스 코드를 지원하기 때문에 이것을 이용해 위에서 이야기한 정책(Policy)들을 자유자재로 넣을 수 있으며, 후에 서술할 금융, 준 금융, 비금융 애플리케이션 등도 계약 소스 코드를 이용해 구현할 수 있다.

4. 애플리케이션

4.1 암호화폐

이더리움 백서에서는 블록체인의 응용 분야를 세 가지로 분류했다. 첫 번째는 화폐 그 자체와 화폐와 연동된 금융 애플리케이션, 두 번째는 준 금융 애플리케이션, 세 번째는 비금융 애플리케이션이다. 이에 먼저, 사슬 오리진의 여러 애플리케이션들에 대해 논의하기 위해 먼저 금융 애플리케이션 중에서 가장 기본적인 화폐 기능을 이야기하고자 한다.
1세대 블록체인인 비트코인과 2세대 블록체인인 이더리움은 속도의 한계로 인해 기존의 화폐를 완전히 대체하는 것은 불가능하다. 암호화폐가 통용 화폐와 같이 작동하려면, 통화가 가진 다음의 기능들을 수행할 수 있어야 한다.

- a. 지불 기능
- b. 가격의 척도
- c. 저축 기능
- d. 교환 수단

기존의 비트코인이나 이더리움 같은 1, 2세대 블록체인의 경우 기존 화폐의 기능 중 가격의 척도로서의 기능, 저축 기능, 지불 기능은 완벽하게 수행하나 기존 화폐가 수행하는 교환 수단의 기능, 즉 거래는 제대로 수행하지 못한다.

화폐가 교환 수단으로서 기능하기 위해서는, 가치를 즉시 측정할 수 있어야 한다. 가치를 인증받는 데 시간이 많이 소요된다면, 거래 기능을 갖추지 못했다고 볼 수 있을 것이다.

비트코인과 이더리움이 화폐로서 기능하지 못하는, 또는 기존의 화폐를 완벽하게 대체하지 못하는 가장 큰 이유는 속도 이슈, 즉 거래 확정까지 걸리는 시간(Commit latency)과 초당 처리할 수 있는 거래량의 한계(Transaction per second, TPS) 때문이다. 비트코인의 경우 거래 확정까지 약 5블록이 생성되어야 한다고 치면, 평균적으로 거래 확정까지 50분 정도 걸린다. 물론, 비트코인을 금이나 부동산 같은 재화로서 생각한다면, 또는 인터넷 상거래를 위한 사이버머니로 생각한다면, 거래 확정에 50분 정도 걸리는 것은 큰일이 아닐 것이다. 그러나 쇼핑이나 외식 등 소액거래에서 50분 정도의 시간이 걸린다면 사람들은 블록체인 대신 통용 화폐를 사용할 것이다.

초당 트랜잭션 처리량(TPS) 또한 문제이다. 현재 비트코인의 초당 트랜잭션 처리량은 10개 정도인데, 이는 비트코인의 대중화에 큰 걸림돌이 될 수 있다. 하루에 거래가 백만 건 이하로 일어나는 작은 국가의 화폐로서는 적절할 수 있으나, 초당 거래가 수천 건 이상 일어나는 거대한 국가의 화폐를 대체하거나 전 세계에서 통용되는 기축통화를 대체하기에는 처리해야 할 거래가 너무 많다.

사슬 오리진은 라운드 단위로 거래가 확정되는데, 한 라운드가 1초 이내로 이루어지기 때문에, 실시간 거래에 이용하기 적합하다. 또한 최소 초당 1000개 이상의 트랜잭션을 처리할 수 있기 때문에 식당이나 커피숍, 택시 같은 데서 이루어지는 소액결제 또한 모두 커버할 수 있다.

사슬 오리진의 거래 수수료는 거래를 일으키는 주체들이 임의로 정할 수 있고 Validator들은 거래를 받아 줄 최소의 수수료를 임의로 정할 수 있다. 그러나 사슬 오리진의 경우 기존의 카드 거래 수수료 1~~ 2%보다 더 저렴한 0.1 ~~ 0.2% 정도의 수수료를 제안한다. 이에 대한 자세한 설명은 5.1 경제학적 이슈에서 언급한다.

4.2 분산형 데이터베이스

현재 개인이나 소규모 기업이 이용하는 서버나 데이터베이스들은 전적으로 몇몇 거대 기업의 서버에 의존하고 있다. 이는 거대 기업들이 개인 정보 또는 중요한 기밀 정보들에 무분별하게 접근할 수 있음을 의미하며 고객들이 데이터의 보안 및 안정성을 기업의 서버에 절대적으로 의존함을 의미한다.

이 문제를 해결하기 위해 분산형 데이터베이스를 제안한다. 이더리움 또는 그 전의 P2P 파일 전송 시스템이 보여준 것처럼 데이터를 비대칭 키 알고리즘으로 암호화해 나눠서 전송한다면 공격자가 일부 노드를 해킹해 데이터를 탈취한다고 하더라도 중요한 정보에 전혀 접근할 수 없다.

사슬 오리진은 분산형 데이터의 구현에만 쓰이는 새로운 메인 네트워크(Main network)를 제공하며 이 분산형 데이터베이스의 구현에만 쓰이는 경제 정책 모델을 제안할 수 있다. 구체적으로 네트워크를 구성하는 개개인은 미리 내장된 계약 소스 코드에 따라 데이터를 얼마나 저장할지 결정할 수 있으며 데이터 보관료 또는 네트워크를 유지하는 보상으로 사슬 오리진의 화폐를 이용할 수 있다. 이는 비-금융 애플리케이션에 사슬 오리진이 적용될 수 있는 예시로 이 외에도 많은 금융-비-금융 애플리케이션에 사슬 오리진을 이용할 수 있다.

또한, 3.4절에 서술한 사슬 오리진 아비터 시스템의 구현을 분산형 데이터베이스를 통해 해결할 수 있다. 처음 블록체인 네트워크를 만들 때, 모든 Validator 노드가 아비터 역할을 부여받는 계약 소스 코드를 내장하는 방법으로 시스템을 구현한다.

사슬 오리진은 서로 분리된 네트워크, 즉 새로운 메인 네트워크 생성이 비교적 자유롭기 때문에 블록체인의 용도에 따라 하나의 목적성만 가진 메인 네트워크를 만드는 것을 추천한다. 사슬 오리진은 다른 네트워크들과 달리 거대하지 않은 네트워크들도 충분히 보안을 유지할 수 있기 때문에 메인 네트워크를 하나로 통합할 이유가 없다. 다른 기능을 가진 애플리케이션들이 하나의 메인 네트워크에 의존한다면, 경제 정책 등을 채택하는 데서 충돌이 일어나기도 쉽고 소스 코드가 길어져 연산 속도가 늘어난다는 등의 비효율이 발생하기도 쉽다. 이는 블록체인 애플리케이션 개발에 있어서 사슬 오리진이 다른 합의 시스템을 이용하는 블록체인들과 비교해서 가지는 장점이라 할 수 있다.

5. 사슬 오리진의 한계와 해결책

5.1 경제학적 이슈

인플레이션은 역사적으로 모든 화폐에서 일어났던 일이었다. 전통적인 화폐에서 일어났던 악화가 양화를 구축한다(Bad money drives out good)는 현상은, 인플레이션이 먼 옛날부터 해결해야 할 문제로 인식되었음을 알 수 있다.

인플레이션 문제는 블록체인을 이용한 암호화폐에서도 피해갈 수 없는 문제로 여겨지고 있다. 작업증명방식을 이용한 암호화폐들, 비트코인이나 이더리움 같은 경우 화폐의 인플레이션을 막기 위해 코인의 발행량을 일정 이하로 제한한다.

작업증명방식의 블록체인을 보자. 비트코인의 경우 2019년 현재 비트코인 생성에 쓰이고 있는 평균 계산력을 측정해, 10분마다 블록 하나가 생성되어 12.5BTC(비트코인의 화폐 단위, 2019년 1월 현재 미화 50000$)만큼의 코인을 생성하게끔 채굴 난이도를 조절한다. 블록 하나를 만들 때 풀리는 비트코인의 양은 초기 50BTC에서 일정 기간이 지남에 따라 계속 반감된다. 이더리움의 경우 매년 일정량 이하의 코인이 생성되고 분배되도록 정한다. 비트코인과 이더리움 모두 인플레이션 문제를 잡는 데는 성공했으나, 비트코인의 경우 비트코인 보상이 반감하는 시점에서 가치의 혼란을 일으킨다는 비판을 받았으며 두 암호화폐 모두 코인 채굴을 선점한 사람들에게 지나치게 유리하다는 비판에서 벗어날 수 없었다.

즉, 지금까지의 암호화폐는 화폐를 먼저 보유한 사람들이 이득을 보는 경제정책, 즉 화폐 가치가 떨어지는 인플레이션을 강하게 억제하고 화폐 가치가 지속해서 증가하는 디플레이션을 다분히 의도적으로 일으켰다고 할 수 있다. 재화가 증가하는데도 화폐의 개수가 고정 또는 제한되어 있다면 화폐의 품귀현상, 즉 디플레이션이 일어나는 것은 너무나 당연하다. 이 디플레이션은 화폐를 선점한 사람들에게 이득을 줌으로 암호화폐의 흥행을 위해 다분히 의도된 것이라고 할 수 있다. 그러나 역사가 증명하듯 디플레이션은 장기적으로 화폐를 이용하는 경제 생태계에 부나 화폐의 집중 등 치명적인 부작용을 낳는다.

지분증명방식을 이용하는 경우에는 이런 선점 효과와 부의 집중을 더욱 경계해야 한다. 지분에 따라 블록을 생성하고, 네트워크의 정책(Policy) 의결권까지 가지는 지분증명방식의 블록체인의 경우 지속해서 화폐를 재분배하는 것은 네트워크의 존속을 위해서 선택의 요소가 아닌 필수 요소이다. 이를 위해 사슬 오리진에서는 신규 화폐 발행과 수수료를 탄력적으로 변화시킴으로 지나친 인플레이션으로 인한 화폐 가치 하락과 디플레이션으로 인한 부의 집중과 거래 위축의 문제를 해결할 수 있다.

사슬 오리진은 지금의 거래량과 수수료의 총량을 계산할 수 있으며, 이를 통해서 Validator 노드들에 적절한 보상을 제공하는 최소의 수수료를 제시할 수 있다. 또한 Validator 노드들의 유치를 위하여 Validator 노드로 활동하는 노드들에 처리한 트랜잭션의 양에 따라 화폐를 발행해 제공하는 방식을 채택해 노드와 지분을 유지할 유인을 제공할 수 있다.

5.2 거버넌스와 정책적 이슈

거버넌스(governance)는 일반적으로 '과거의 일방적인 정부 주도적 경향에서 벗어나 정부, 기업, 비정부기구 등 다양한 행위자가 공동의 관심사에 대한 네트워크를 구축하여 문제를 해결하는 새로운 국정운영의 방식'을 말하며 사슬 오리진에서는 계약에 투표 또는 합의 기능을 가진 소스 코드를 엔진에 추가함으로 네트워크의 문제를 해결하는 것을 거버넌스라고 부른다

사슬 오리진은 위에 서술한 경제 정책 또는 Validator 선발 및 부의 분배 정책 등을 탄력적으로 적용할 수 있는 플랫폼을 적용한다. 기술적으로 사슬 오리진에서는 모든 Validator의 동의가 있으면 소스 코드를 추가하는 기능만 제공하지만, 블록체인이 쓰이는 용도에 따라 1인 1표 또는 지분에 따른 투표 등 통해 화폐 발행 등의 정책을 결정할 수 있는 시스템을 제공할 수 있다. 이 시스템을 이용해 블록체인을 현실의 여러 문제를 해결하는 정책 및 의사결정을 다 같이 할 수 있게 만들 수 있다.

구체적으로, Validator 노드가 블록을 잘못 만들어 공격한 경우 시스템이 분리된다. 이 경우, 분리된 네트워크 중 어떤 네트워크를 신뢰할 수 있는지 사용자들이 결정해야 하는데, 이 결정을 정책으로 도와줄 수 있다. 예를 들어, 지분증명 방식의 아이디어를 빌려 지분에 해당하는 만큼 투표권을 줄 수도 있고, Validator+Supervisor 노드에 1표씩을 부여해서 과반수가 선택하는 네트워크를 옳은 네트워크로 인정할 수 있다. 이 경우, 채택되지 않은 Validator들이 속한 네트워크에 대한 처벌 또한 정책으로 정할 수 있다. 물론 권위증명방식의 아이디어를 이용해 특정 아비터 노드들에 옳은 노드 선택권을 주는 정책 또한 채택할 수 있다.

사슬 오리진은 구체적인 코인 생태계 모델, 처벌 모델, 수수료 분배 및 코인 발행 모델을 제시하지만, 이것은 절대적인 것이 아니다. 블록체인의 쓰임은 일정 시기의 일정 부문의 통화를 대체하는 것을 넘어서 데이터베이스 인증 시스템 또는 탈중앙화된 자율 기구를 수행하는 것까지 포함한다. 이는 블록체인의 용도에 따라 다 다른 정책이 필요하다는 것을 의미한다.

블록체인 용도의 확장을 처음으로 제안했던 이더리움의 경우 작업증명방식이라는 한계가 새로운 메인 네트워크를 만드는 데 걸림돌로 작용한다. 용도에 따라 다른 정책을 채택한 메인 네트워크를 만들게 되면 네트워크의 보안성을 보증하는 계산력이 나뉘어 보안성이 떨어지게 된다. 이에 이더리움의 화폐인 "이더" 분배 정책은 메인 네트워크의 개수인 하나일 수밖에 없다.

사슬 오리진의 경우 블록체인의 안정성이 계산력이 아닌 암호화 알고리즘과 합의 알고리즘에 기반하기 때문에, 새로운 메인 네트워크의 등장이 보안성의 하락을 의미하지 않는다. 즉 사슬 오리진의 경우 용도에 따라 다른 경제 정책, 이더리움 같은 작업증명방식의 블록체인에선 결코 할 수 없었던 화폐 분배 정책 또는 수수료 정책까지 다른 메인 네트워크 만들기를 자유롭게 만들 수 있다.

6. 결론(Conclusion)

사슬 오리진은 기존의 블록체인들이 이용했던 합의 방식들과 근본적으로 다른 사슬합의 방식(SASEUL Consensus Algorithm)을 이용한 블록체인으로 기존의 블록체인들보다 적은 자원으로 빠르게 돌아가며 이중지불 공격 등의 악의적인 공격에서 안전한 블록체인이다.

또한 사슬 오리진은 다른 언어나 패키지에 의존하지 않고 튜링-완전 언어를 사용한 계약을 사용할 수 있게 제공함으로 기존의 블록체인들보다 더 정교하게 애플리케이션을 제작할 수 있게 만들며 여러 가지 정책(Policy)을 탄력적으로 채택할 수 있는 모델을 제시해 블록체인의 사용 범위를 제한된 화폐의 대체재로부터 금융 시스템 전반과 비금융 시스템, 탈중앙화된 자율 기구로까지 넓힌 모델을 제시하는 데 성공했다.

Reference

- [1] Satoshi Nakamoto, (2009) Bitcoin: A Peer-to-Peer Electronic Cash System https://bitcoin.org,
- [2] Vitalik Buterin. (2018) Ethereum Whitepaper https://www.ethereum.org
- [3] A. Back, (2002) "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf
- [4] Brainerd, W.S.; Landweber, (1974) L.H Theory of Computation. Wiley,
- [5] 정운찬 김홍범 저 (2013), 화폐와 금융시장 4판
- [6] Thomas Piketty (2013). Capital in the twenty-first century
- [7] Applebaugh, John (2015). “Governance Working Group”《info.publicintelligence.net》. ISAF
- [8] Georgios Konstantopoulos (2018). Understanding Blockchain Fundamentals, Part 3: Delegated Proof of Stake
- [9] Jae Kwon, Ethan Buchman (2018). Tendermint cosmos white paper
- [10] Federal Reserve Bank of St. Louis (2013) "Money Velocity"