RaiBlocks: 수수료 없는 분산 암호화폐 네트워크

Colin LeMahieu
clemahieu@gmail.com

Abstract—최근 높은 수요와 제한된 확장성으로 인해 인기 있는 가상화폐들의 트랜잭션 시간과 수수료가 늘어났고, 이는 불만족스러운 경험을 낳았습니다. 여기에서 우리는 독창적인 블록-래티스 구조의 암호화폐로, 각 계정이 고유한 블록체인을 가지고, 거의 즉각적으로 처리되는 트랜잭션 속도와 제약 없는 확장성을 가진 RaiBlocks 를 소개합니다. 각 사용자는 자신들의 고유한 블록체인을 가지는데, 이는 네트워크의 다른 참여자와 비동기적으로 갱신할 수 있게 하며, 최소한의 오버헤드로 빠르게 트랜잭션을 처리할 수 있게 합니다. 트랜잭션들은 트랜잭션의 금액이 아닌 계정의 잔고를 추적하며, 이는 보안을 약화시키지 않고도 적극적으로 데이터베이스를 축약할 수 있게 합니다. 최근까지, RaiBlocks 네트워크는 4 백 2 십만 트랜잭션들을 처리해 왔는데 전체 원장의 크기는 1.7GB 에 불과합니다. RaiBlocks 의 수수료가 없으며, 순식간에 처리되는 트랜잭션들은 RaiBlocks 가 소비자 트랜잭션을 위한 최고의 암호화폐가 되게 합니다.

Index Terms—cryptocurrency, blockchain, raiblocks, distributed ledger, digital, transactions

I. 소 개

2009 년 비트코인의 구현 이후, 전통적인 정부가 보증하는 통화와 금융 시스템으로부터 암호화폐에 기반한 현대적인 지불 시스템으로의 변화가 커졌으며, 암호화폐는 신뢰 없이도 안전하게 자금을 보관하고 전송할 수 있게 되었습니다 [1]. 통화가 효율적으로 기능하기 위해서는 쉽게 전달할 수 있어야 하고, 비가역적이어야 하며, 수수료는 제한되거나 없어야 합니다. 늘어나는 트랜잭션 시간, 높은 수수료 그리고 의문이 생기는 네트워크의 확장성은 일상적인 통화로서 비트코인의 실용성에 대한 의문을 증폭시켰습니다. 이 논문에서 우리는 제약 없는 확장성을 제공하며 수수료가 없는 혁신적인 블록-래티스 자료 구조에 기반한 저지연 (low-latency) 암호화폐인 RaiBlocks 를 소개합니다. RaiBlocks 는 설계적으로 고성능 암호화폐가 되는 것을 유일한 목표로 가진 단순한 프로토콜입니다. RaiBlocks 프로토콜은 낮은 성능의 하드웨어에서 동작할 수 있으며 이는 일상적인 사용을 위한 실용적이고 탈중앙화된 암호화폐가 되게 합니다.

이 논문에 기술된 암호화폐에 관한 통계들은 출판 시점에는 정확합니다.

II. 배 경

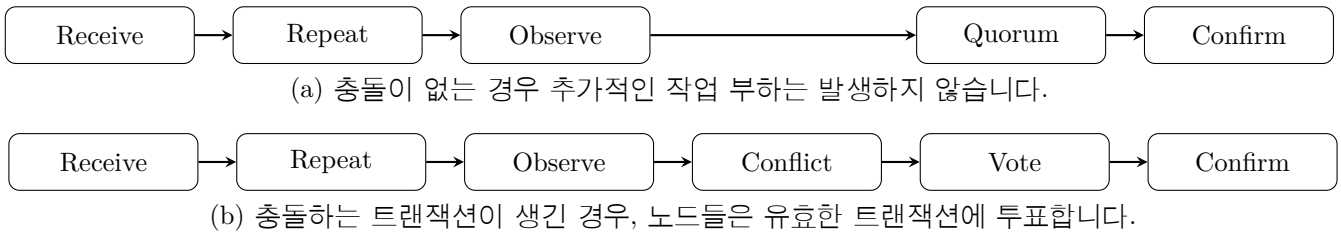
2008 년 사토시 나카모토라는 필명으로 익명의 개인이 세계 최초의 탈중앙화 암호화폐인 비트코인의 개요를 서술한 백서를 출간했습니다 [1]. 비트코인이 불러온 주요 혁신은 공개적이고 불변하며 탈중앙화된 자료 구조인 블록체인으로, 이는 통화의 거래에 대한 원장으로 사용됩니다. 불행히도 비트코인이 성장함에 따라, 프로토콜의 여러 문제점들이 많은 응용에서 비트코인을 사용할 수 없게 만들었습니다:

1) 불충분한 확장성: 블록체인의 각 블록은 제한된 양의 데이터를 저장할 수 있는데, 이는 시스템이 초당 그만큼의 트랜잭션만 처리할 수 있음을 의미하며, 블록의 자리를 상품으로 만들어 버렸습니다. 현재 평균 트랜잭션 수수료는 $10.38 [2] 입니다.
2) 높은 지연성: 평균 확인 시간은 164 분입니다 [3].
3) 전력 비효율성: 비트코인 네트워크는 년간 27.28TWh 의 예상 전력을 사용하며 각 트랜잭션당 평균적으로 260KWh 를 소비합니다. [4].

비트코인, 그리고 다른 암호화폐들은 악의적인 행위자들에 대항하면서 합법적인 트랜잭션들을 검증하기 위해 전역 원장에 대한 합의를 달성하며 동작합니다. 비트코인은 작업증명 (PoW) 이라고 불리는 경제적인 수단을 통해 합의를 이룹니다. PoW 시스템에서 시스템 참여자들은 논스라고 불리는 수를 계산하기 위해 경쟁하는데 전체 블록의 해시가 대상 범위에 포함됩니다. 이 유효 범위는 유효한 논스를 찾는데 소요되는 평균 시간을 일관되게 유지하기 위해 비트코인 네트워크의 누적 연산 능력에 반비례합니다. 유효한 논스를 발견한 사람은 블럭을 블록체인에 추가할 수 있게 됩니다. 즉 논스를 계산하기 위해 더 많은 연산 자원을 소비한 사람들이 블록체인의 상태에 있어 더 큰 역할을 하게 됩니다. 각 주체가 탈중앙화된 시스템에서 추가적인 힘을 얻기 위해 여러 엔터티들로서 동작하기 때문에 PoW 는 시빌 (Sybil) 공격에 저항성을 가지며, 글로벌한 자료 구조에 접근할 때 본질적으로 존재하는 경쟁 상황 (race condition) 을 크게 줄입니다.

다른 합의 프로토콜인 지분증명 (PoS) 은 2012 년 Peercoin [5] 에서 처음 소개되었습니다. PoS 시스템에서 참여자들은 해당 암호화폐를 보유한 양에 따라 가중치를 가지고 투표에 참여합니다. 이러한 구조에서 더 많은 재정적 투자를 한 사람들은 더 많은 힘을 가지게 되어 본질적으로 정직성을 유지하도록 동기가 부여되고, 그렇지 않을 경우 그들의 투자를 잃을 위험이 생깁니다. PoS 는 낭비적인 연산 능력 경쟁을 멀리하며, 낮은 성능의 하드웨어에서 동작하는 경량 소프트웨어만을 요구합니다.

최초의 RaiBlocks 논문과 첫 번째 베타 구현은 2014 년 12 월에 발간되었고, 최초의 방향성 비순환 그래프 (DAG) 기반의 암호화폐 중 하나가 되었습니다 [6]. 곧 다른 DAG 암호화폐들이 개발되기 시작하였는데 가장 주목할만한 것은 DagCoin/Byteball 과 IOTA 입니다 [7], [8]. 이 DAG 기반 암호화폐들은 블록체인의 틀을 깨고 퍼포먼스와 보안성을 향상시켰습니다. Byteball 은 정직하고, 평판이 좋으며 사용자가 신뢰하는 “증인” 들로 구성된 “메인체인” 에 의존하여 합의를 취득합니다. 반면에 IOTA 는 쌓여있는 트랜잭션들의 누적된 PoW 를 통해 합의를 이룹니다. RaiBlocks 는 충돌하는 트랜잭션들에 대한 잔고 가중치 투표를 통해 합의를 얻습니다. 이 합의 시스템은 강력한 탈중앙화된 시스템을 유지하면서도, 더 빠르고, 더 결정적인 트랜잭션들을 제공합니다. RaiBlocks 는 지속적으로 개발을 진행하고 있으며, 가장 고성능의 암호화폐중 하나로 자리매김하고 있습니다.



(a) 충돌이 없는 경우 추가적인 작업 부하는 발생하지 않습니다.

(b) 충돌하는 트랜잭션이 생긴 경우, 노드들은 유효한 트랜잭션에 투표합니다.

Fig. 1. RaiBlocks 는 일반적인 트랜잭션에는 추가적인 작업 부하가 생기지 않습니다. 충돌하는 트랜잭션이 생긴 경우, 노드들은 유지할 트랜잭션에 반드시 투표해야 합니다.

III. RaiBlocks 구성요소

전반적인 RaiBlocks 아키텍처를 서술하기 전에, 시스템을 구성하는 개별 컴포넌트를 정의합니다.

A. 계정

계정은 디지털 서명 키쌍에서 공개키 부분에 해당합니다. 어드레스라고도 불리는 공개키는 다른 네트워크 참여자와 공유되지만 개인키는 비밀로 유지됩니다. 디지털 서명된 데이터의 패킷은 그 내용이 개인키 소유자에 의해 승인되었다는 것을 보증합니다. 한 사용자는 여러 계정을 가질수 있지만, 한 계정에는 단 하나의 공개 어드레스가 존재합니다.

B. 블록/트랜잭션

"블럭" 과 "트랜잭션" 이라는 용어는 종종 대체하여 사용되는데, 블럭은 하나의 트랜잭션을 포함합니다. 트랜잭션은 구체적으로 어떤 행위를 말하며, 반면에 블럭은 트랜잭션의 디지털 인코딩을 나타냅니다. 트랜잭션들은 트랜잭션이 실행되는 계정에 속한 비밀키로 서명됩니다.

C. 원장

원장은 계정의 글로벌한 집합으로, 원장에서 각 계정은 고유의 트랜잭션 체인을 가집니다 (Figure 2). 이는 런타임 계약을 디자인 타임 계약으로 대체하는 범주에 속하는 핵심 설계 구성 요소입니다. 모든 참여자는 서명 확인을 통해 오직 계정의 소유자만 자신의 체인을 수정할 수 있다는 데 동의합니다. 이렇게 하면 공유된 자료구조처럼 보이는 분산 원장이 공유되지 않는 자료구조의 집합으로 변환합니다.

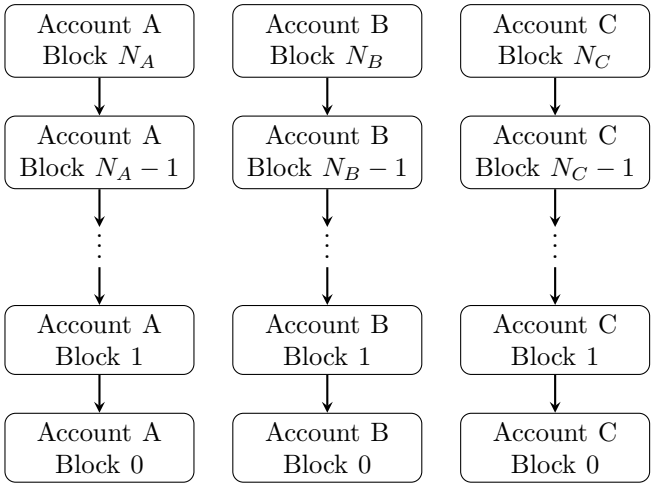


Fig. 2. 각 계정은 계정의 잔고 이력을 가지고 있는 고유의 블록체인을 가집니다. 블록 0 는 반드시 open 트랜잭션이어야 합니다 (Section IV-B).

D. 노드

노드는 컴퓨터에서 동작하며 RaiBlocks 프로토콜을 따르고 RaiBlocks 네트워크에 참여하는 소프트웨어입니다. 이 소프트웨어는 원장과 노드가 관리하는 모든 계정을 관리합니다. 노드는 전체 원장 또는 각 계정의 블록체인마다 마지막 몇 블럭들만을 보관한 축약된 이력을 저장할 수 있습니다. 새 노드를 설치할 때에는 전체 이력을 검증한 후 로컬에서 과거 이력을 제거하는 것이 권장됩니다.

IV. 시스템 개요

많은 다른 암호화폐들에서 사용되는 블록체인들과 달리, RaiBlocks 는 블록-래티스 구조를 사용합니다. 각 계정은 계정의 트랜잭션/잔고 이력에 해당하는 고유의 블록체인 (계정-체인) 을 가집니다 (Figure 2). 각 계정-체인은 오직 계정의 소유자에 의해서만 변경될 수 있습니다. 이는 각 계정-체인이 즉시 변경되고, 블록-래티스의 다른 부분에서 비동기적으로 반영되는 것을 허용하며, 이로 인해 빠른 트랜잭션이 가능해집니다. RaiBlocks 의 프로토콜은 극단적으로 가볍습니다. 각 트랜잭션은 인터넷에 발신되기 위한 최소한의 UDP 패킷 크기에 들어갑니다. 노드를 실행하기 위한 하드웨어 요구사항 역시 최소한이 되는데, 노드들은 대부분의 트랜잭션에 대해 블럭들을 기록하고 재전송하는 일만 하면 되기 때문입니다 (Figure 1).

이 시스템은 제네시스 잔고를 가지고 있는 제네시스 계정에서 시작됩니다. 제네시스 잔고는 고정된 양으로 절대 늘어나지 않습니다. 이 제네시스 잔고는 분할되어 제네시스 계정-체인에 등록되는 send 트랜잭션들을 통해 다른 계정으로 전송됩니다. 모든 계정 잔고의 총합은 절대 전체 시스템의 총량 상한을 규정하는 초기 제네시스 잔고를 넘을 수 없으며 이를 늘릴수는 없습니다.

이 섹션은 어떻게 다른 유형의 트랜잭션들이 생성되어 네트워크를 통해 전파되는지 단계별로 설명합니다.

A. 트랜잭션

한 계정에서 다른 계정으로 자금을 보내려면 두 개의 트랜잭션이 필요합니다: 발신자의 잔고에서 감액하는 send 트랜잭션과 수신자의 잔고를 증액하는 receive 트랜잭션입니다. (Figure 3).

자금을 발신자와 수신자 계정의 개별 트랜잭션으로 전달하는 것은 몇 가지 중요한 목적을 가지고 있습니다.

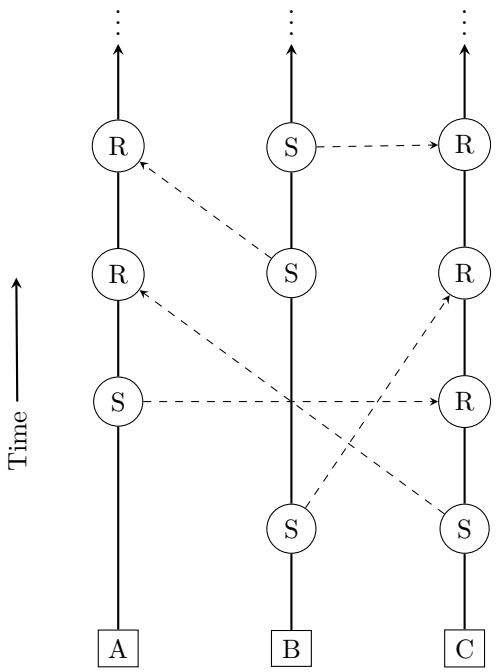


Fig. 3. 블록-래티스의 시각화. 모든 자금 이동에는 하나의 send 블록 (S) 과 하나의 receive 블록 (R) 이 필요하며, 각각은 계정 체인의 소유주 (A,B,C) 에 의해 서명되어야 합니다.

1) 본질적으로 비동기적으로 들어오는 트랜잭션들의 순서를 매김.
2) 트랜잭션들을 UDP 패킷들에 맞게 작게 유지.
3) 자료 용량을 최소화하여 원장의 축약을 가능하게 함.
4) 체결된 트랜잭션들을 미체결된 트랜잭션들과 격리.

같은 목적지 계정으로 전송하는 하나 이상의 계정들은 비동기적으로 동작합니다. 네트워크 지연과 함께 발송 계정이 상호간 통신할 필요가 없다는 것은 어느 트랜잭션이 먼저 발생했는지 알 수 있는 일반적으로 동의할 만한 방법이 없다는 것을 의미합니다. 더하기는 결합적 (associative) 이기 때문에, 입력의 순서를 매기는 차례는 중요하지 않습니다. 그러므로 우리는 그저 전체적인 합의만 필요합니다. 이것은 런타임 합의를 디자인타임 합의로 변환하는 주요 설계 요소입니다. 수신하는 계정은 어느 전송이 먼저 도착했는지, 어느 전송이 입력 블록들 중 서명된 순서로 나타나는지를 결정할 수 있습니다.

만약 계정이 많은 수의 소액전송을 받아 큰 금액을 전송하려 할 경우, 우리는 이것을 하나의 UDP 패킷에 맞도록 표현하고 싶습니다. 수신하는 계정이 입력되는 전송들에 순서를 매길 때, 자기 계정의 잔고 누계를 갱신하기 때문에 계정은 어느 시점에건 원하는 금액을 고정된 크기의 트랜잭션으로 전송할 수 있습니다. 이것은 비트코인과 다른 암호화폐들이 사용하는 입력/출력 트랜잭션 모델과 다릅니다.

어떤 노드들은 계정들의 모든 트랜잭션 이력을 저장하는데에 리소스를 쏟고 싶어하지 않습니다. 그들은 오직 각 계정의 현재 잔고에만 관심이 있습니다. 한 계정이 트랜잭션을 만들 때, 그 계정은 자신의 누적 잔고를 인코딩하며, 이러한 노드들은 오직 최신 블록들만을 유지하면 됩니다. 이는 노드들이 정확성을 유지하면서도 이전 이력 데이터를 버릴 수 있게 해 줍니다.

디자인 타임 합의에 집중하더라도, 네트워크상의 나쁜 행위자들을 식별하고 처리하는 일들로 인해, 트랜잭션의 검증시 지연 시간이 있습니다. RaiBlocks 에서 합의는 밀리세컨드에서 초 단위로 빠르게 이루어지기 때문에 우리는 사용자에게 전달받은 트랜잭션들에 대해' 체결' 과' 미체결' 이라는 두가지 친숙한 범주를 제시할 수 있습니다. 체결된 트랜잭션들은 계정이 수신 블록들을 만든 트랜잭션들입니다. 미체결된 트랜잭션들은 수신자의 누적 잔고에 아직 반영되지 않은 트랜잭션들입니다. 이는 다른 암호화폐의 더 복잡하고 낯선 승인 (Confirmation) 단위를 대신합니다.

B. 계정의 생성

계정을 만들기 위해서는 open 트랜잭션을 만들어야 합니다 (Figure 4). 오픈 트랜잭션은 항상 각 계정-체인의 첫 번째 트랜잭션이며 최초로 자금을 받을 때 생성될 수 있습니다. account 필드는 서명에 사용되는 개인키에서 도출된 공개키 (어드레스) 를 저장합니다. source 필드는 자금을 보낸 트랜잭션의 해시를 보관합니다. 계정 생성시 당신을 대신할 대리인이 반드시 선택되어야 하는데, 대리인은 추후 변경할 수 있습니다 (Section IV-F). 계정은 자기 자신을 자신의 대리인으로 선언할 수 있습니다.

```
open {
    account: DC04354B1...AE8FA2661B2,
    source: DC1E2B3F7C...182A0E26B4A,
    representative: xrb_1anr...posrs,
    work: 0000000000000000,
    type: open,
    signature: 83B0...006433265C7B204
}
```

Fig. 4. open 트랜잭션의 구조

C. 계정 잔고

계정의 잔고는 원장 자체에 기록됩니다. 트랜잭션의 금액을 기록하는 대신에 검증 (Section IV-I) 은 send 블록과 그 이전 블록의 잔고간의 차이를 확인할 것을 요구합니다. 수신 계정은 이전 잔고를 새로운 수신 블록에 주어진 마지막 잔액으로 증가시킬 수 있습니다. 이는 대량의 블록을 다운로드 할 때 처리 속도를 향상시키기 위해 수행됩니다. 계정의 이력을 요구할 때, 잔액이 즉시 제공됩니다.

D. 계정으로부터의 송금

계정에서 송금할 때, 그 어드레스는 이미 open 블럭과 잔액을 가지고 있어야 합니다. previous 필드는 계정-체인의 이전 블럭 해시를 가지고 있습니다. destination 필드는 자금이 전달될 계정을 가집니다. send 블록은 일단 승인되면 변경되지 않습니다. 네트워크에 전파되고 나면, 자금은 즉시 송금자의 계정 잔고에서 감액되고, 수신자가 이 자금을 받아들이도록 서명할 때까지 pending 상태로 보류됩니다. 보류중인 자금은 송금자의 계정에서 사용된 것과 같으며, 송금자가 트랜잭션을 취소할 수 없기 때문에 승인을 기다리는 것으로 간주해서는 안 됩니다.

```
send {
    previous: 1967EA355...F2F3E5BF801,
    balance: 010a8044a0...1d49289d88c,
    destination: xrb_3w...m37goeuufdp,
    work: 0000000000000000,
    type: send,
    signature: 83B0...006433265C7B204
}
```

Fig. 5. send 트랜잭션의 구조

E. 트랜잭션의 수신

트랜잭션을 완료하기 위해서 송금액의 수신자는 반드시 receive 블록을 자신의 계정-체인에 생성해야 합니다 (Figure 6). source 필드는 연결된 send 트랜잭션의 해시를 나타냅니다. 이 블럭이 생성되고 전파되면, 계정의 잔고가 갱신되고 그 자금은 공식적으로 수신자 계정으로 이동하게 됩니다.

```
receive {
    previous: DC04354B1...AE8FA2661B2,
    source: DC1E2B3F7C6...182A0E26B4A,
    work: 0000000000000000,
    type: receive,
    signature: 83B0...006433265C7B204
}
```

Fig. 6. receive 트랜잭션의 구조

F. 대리인 지명

계정 보유자가 자신들을 대신할 대리인을 선택할 수 있는 것은 작업 증명 또는 지분 증명 프로토콜과 유사성이 없는 강력한 탈중앙화 도구입니다. 통상적인 PoS 시스템에서 계정 소유자의 노드는 투표에 참여하기 위해 반드시 실행중이어야 합니다. 항상 노드를 실행시키는 것은 많은 사용자에게 실용적이지 않습니다. 대리인에게 계정을 대신하여 투표할 권한을 주는 것은 이런 요구사항을 완화시킵니다. 계정 소유자는 언제든 다른 계정을 대리인으로 지정할 수 있습니다. change 트랜잭션은 이전 대리인에게서 투표 가중치를 감하고, 새로운 대리인에게 가중치를 늘림으로써 계정의 대리인을 변경합니다 (Figure 7). 이 트랜잭션에서 자금은 이동하지 않으며, 대리인은 계정의 자금을 사용할 권한을 가지지 않습니다.

```
change {
    previous: DC04354B1...AE8FA2661B2,
    representative: xrb_1anrz...posrs,
    work: 0000000000000000,
    type: change,
    signature: 83B0...006433265C7B204
}
```

Fig. 7. change 트랜잭션의 구조

G. 포크와 투표

포크는 j 가 서명한 블록 $b_1, b_2, \ldots, b_j$ 가 같은 블록을 이전 블록으로 주장할 때 발생합니다 (Figure 8). 이 블록들은 계정의 상태에 대해 충돌을 일으키며 반드시 해결되어야만 합니다. 오직 계정의 소유주만이 자신의 블록을 서명하고 계정-체인에 넣을 수 있습니다. 그렇기 때문에 포크는 잘못된 프로그래밍이나 계정 소유주의 악의적인 의도 (이중 지불) 의 결과일 수 밖에 없습니다.

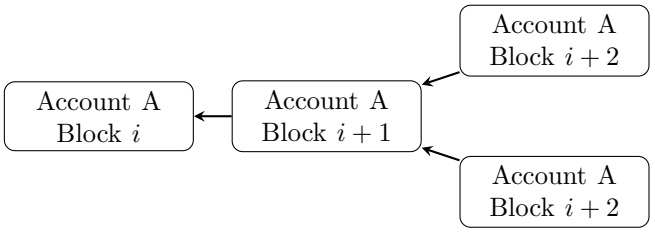


Fig. 8. 둘 또는 그 이상의 서명된 블록이 같은 previous 블록을 참고할 때 포크가 발생합니다. 이전 블럭이 왼쪽, 신규 블록들이 오른쪽에 있습니다.

포크가 발견되면, 대리인은 블록 $\hat{b}_i$ 를 가리키는 투표를 자신의 원장에 생성하고 네트워크에 전파합니다. 노드의 투표인 w_i 의 가중치는 해당 노드를 대리인으로 지정한 모든 계정들 잔고의 합입니다. 대리인 노드는 다른 M 개의 온라인 대리인들로부터 들어오는 투표들을 1 분간 4 회의 투표 기간동안 누계를 내어 관찰하며, 이긴 블록을 승인합니다 (Equation 1).

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

가장 인기있는 블록 b^* 가 투표의 대다수를 차지하고 노드의 원장에 유지됩니다 (Equation 2). 투표에서 진 블록은 버려집니다. 투표의 결과로 대리인이 자신의 원장에 있는 블록을 교체하면, 대리인 노드는 새로운 블록을 높은 시퀀스 넘버로 생성하고 이에 대해 새로운 투표를 네트워크에 전파하게 됩니다. 이것은 대리인들이 투표하는 유일한 시나리오입니다.

어떤 경우에는 짧은 네트워크 접속 문제가 전파된 블록이 모든 피어로부터 받아들여지지 않게 할 수 있습니다. 이 계정으로부터 나오는 이 이후의 모든 블록들은 초기 전파를 보지 못한 피어들로부터 무효로 무시될 것입니다. 이 블록을 재전파하면 남아있는 피어들은 수용할 것이며, 이후 블록들은 자동적으로 회수될 것입니다. 포크나 유실된 블록이 발생하더라도, 오직 트랜잭션에서 참조된 계정들만이 영향을 받습니다. 네트워크의 다른 부분들은 다른 모든 계정들의 트랜잭션들을 처리하면서 진행합니다.

H. 작업 증명

모든 네가지 트랜잭션 유형은 work 필드를 가지고 있으며 이 필드는 정확하게 채워져야 합니다. 이 work 필드는 트랜잭션 생성자가 이 work 필드는 트랜잭션 생성자가 receive/send/change 트랜잭션의 previous 필드 또는 open 트랜잭션의 계정 필드에 연결되어있는 논스의 해

시가 특정 임계 값 미만이되도록 논스를 계산할 수있게 합니다. 비트코인과는 달리 RaiBlocks 의 PoW 는 단지 Hashcash 처럼 스팸 방지 도구로만 사용되며, 몇 초 정도면 계산됩니다 [9]. 트랜잭션이 발송되면 previous 블록 필드를 알 수 있기 때문에, 다음 블록을 위한 PoW 가 미리 계산될 수 있습니다. 이는 트랜잭션간의 간격이 PoW 를 계산하는 시간보다 큰 동안에는 트랜잭션이 최종사용자에게 즉각적으로 처리되는 것으로 보이게 합니다.

I. 트랜잭션 검증

블록이 유효하게 간주되기 위해서는 다음의 특성들을 가져야 합니다:

1) 블록은 아직 원장에 있지 않아야 합니다 (중복된 트랜잭션).
2) 계정 소유자에 의해 서명되어야 합니다.
3) 이전 블록이 계정-체인의 헤드 블록입니다. 만약 이전 블록이 존재하지만 헤드 블록이 아니라면, 이것은 포크입니다.
4) 계정은 반드시 open 블록을 가지고 있어야 합니다.
5) 계산된 해시는 PoW 임계값 요구사항을 충족해야 합니다.

만약 블록이 receive 블록이라면 source 블록 해시가 보류중인지, 즉 아직 취득되지 않은 블록인지 확인합니다. 만약 send 블록이라면, 잔액이 이전 잔고보다 적어야 합니다.

V. 공격 방식

다른 모든 탈중앙화된 암호화폐들과 마찬가지로, RaiBlocks 도 금전적 이득 또는 시스템 붕괴를 목적으로 하는 악의적인 집단의 공격을 받을 수 있습니다. 이 섹션에서 우리는 몇 가지 가능한 공격 시나리오와 그러한 공격의 결과, 어떻게 RaiBlocks 의 프로토콜이 예방 조치를 취하는지 설명합니다.

A. 블록 갭 동기화

IV-G 섹션에서 우리는 블록이 적절하게 전파되지 않을 수 있고 그 결과로 네트워크가 그 이후의 블록들을 무시하게 되는 경우를 논의했습니다. 만약 노드가 어떤 블럭이 카리키고 있는 이전 블럭이 없는 것을 발견했을 때 두 가지 선택이 존재합니다.

1) 악의적인 쓰레기 블럭일 수 있으므로 해당 블럭을 무시한다.
2) 다른 노드에게 재동기화를 요청한다.

재동기화의 경우에 재동기화가 필요로하는 트래픽 증가를 용이하게 하기 위해, 부트스트래핑 노드와의 TCP 연결이 반드시 형성되어야 합니다. 하지만 만약 그 블럭이 실제로 나쁜 블럭이었다면, 그 동기화는 필요 없는 것이며 불필요하게 네트워크의 트래픽을 증가시키게 됩니다. 이것은 네트워크 증폭 공격으로 서비스 거부의 결과를 가져 옵니다.

불필요한 재동기화를 피하기 위해, 노드들은 부트스트랩 노드와 연결을 맺고 동기화하기 전에, 잠재적으로 악의적인 블록에 대해 특정 수준의 투표들을 확인할 때까지 기다립니다. 만약 블록이 충분한 투표를 받지 못 하면 그 블록은 쓰레기 데이터로 가정할 수 있습니다.

B. 트랜잭션 홍수

악의적인 참여자는 네트워크를 포화시키려는 목적으로 자신이 관리하는 계정간에 불필요하지만 유효한 트랜잭션을 많이 보낼 수 있습니다. 트랜잭션 수수료가 없으므로 그들은 이런 공격을 끝없이 계속할 수 있습니다. 하지만 각 트랜잭션에 필요한 PoW 때문에 연산 자원에 상당한 투자를 하지 않고서는 악의적인 공격자가 생성할 수 있는 트랜잭션의 속도는 제한됩니다. 원장을 부풀리기 위한 이러한 공격하에 있다손 치더라도 풀 노드가 아닌 노드들은 예전 트랜잭션을 자신들의 체인에서 잘라낼 수 있습니다. 이는 거의 모든 사용자들의 스토리지 사용량을 이런 공격으로부터 고정시킵니다.

C. 시빌 (Sybil) 공격

공격자는 하나의 컴퓨터에 수백개의 Raiblocks 노드를 만들 수도 있습니다. 하지만 투표시스템은 계정의 잔고에 기반하여 가중치를 가지기 때문에, 네트워크에 노드를 추가하더라도 공격자가 추가적인 투표를 얻을 수는 없습니다. 그러므로 시빌 공격으로는 이점을 얻을 수 없습니다.

D. 소액 지불 고객

페니 지불 공격은 공격자가 저장 공간을 낭비시키기 위해 극소량의 금액을 많은 수의 계정에 보내는 공격입니다. 블록 발행은 PoW 에 의해 비율이 제한됩니다. 그렇기 때문에 이는 계정과 트랜잭션의 생성을 어느 정도 제한하게 됩니다. 모든 이력을 보관하지 않는 노드들은 통계적 지표 이하의 유효하지 않을 것 같은 계정들을 잘라낼 수 있습니다. 마지막으로 RaiBlocks 는 최소한의 영구 저장 공간을 사용하도록 최적화되어 있고, 하나의 계정을 추가적으로 저장하기 위해 필요한 공간은 open 블록 + 인덱싱 = 96B + 32B = 128B 에 비례합니다. 이는 1GB 로 8 백만 페니 지불 공격을 저장할 수 있다는 계산이 됩니다. 만약 노드가 좀 더 공격적으로 축약하려 한다면, 액세스 빈도의 분포를 계산하여 자주 사용되지 않는 계정을 느린 스토리지에 맡길 수 있습니다.

E. 사전계산된 PoW 공격

계정의 소유주가 계정-체인에 블록을 추가하는 유일한 엔터티가 되기 때문에 네트워크에 블록들을 전파하기 전에 순차적인 블록들이 PoW 와 함께 계산될 수 있습니다. 여기서 공격자는 오랜 시간동안 최소한의 값을 가진 무수히 많은 순차적인 블록들을 생성할 수 있습니다. 특정 시점에 공격자는 다른 노드들이 가능한 빨리 처리하고 응답할 많은 수의 유효한 블록들을 네트워크에 흘려보냄으로써 서비스 공격을 할 수 있습니다. 이것은 V-B 섹션에 기술된 트랜잭션 홍수의 진보된 버전입니다. 이런 공격은 잠시동안만 동작하겠지만 효과를 높이기 위해 >50% 과 같은 다른 공격과 함께 진행될 수 있습니다. 공격을 완화하기 위해 트랜잭션 속도 제한과 다른 기술들을 현재 조사중입니다.

F. >50% 공격

RaiBlocks 에서 합의의 척도는 잔고 가중치 투표 시스템입니다. 만약 공격자가 50% 이상의 투표 파워를 취득할 수 있다면, 네트워크가 합의를 일으켜 시스템이 파괴될 수 있습니다. 공격자는 네트워크 서비스 거부 공격을

일으켜 선량한 노드들이 투표하는 것을 막음으로써 확보해야 할 잔고의 양을 낮출 수 있습니다. RaiBlocks 은 이런 공격을 방지하기 위해 아래의 방법들을 사용합니다.

1) 이런 유형의 공격에 대한 기본적인 방어는 투표 가중치를 시스템에 대한 투자와 연결시키는 것입니다. 계정 보유자는 근본적으로 자신의 투자를 보호하기 위해 정직성을 유지할 동기가 생깁니다. 원장을 뒤집으려 시도하는 것은 시스템 전반적으로 파괴적일 수 있으며, 자신들의 투자를 파괴할 수 있습니다.
2) 이 공격의 비용은 RaiBlocks 의 시가총액에 비례합니다. PoW 시스템에서 공격이 성공할 경우 자금 투자와 비교하여 부적절한 통제력을 부여하는 기술이 만들어질 수 있습니다. 그리고 이 기술이 공격이 성공한 후 다른 목적을 가지게 될 수 있습니다. RaiBlocks 에서 공격을 하기 위한 비용은 시스템 그 자체와 함께 커지며, 공격이 성공적이 된다면 공격에 들어간 투자는 복구될 수 없습니다.
3) 투표자의 최대 정족수를 유지하기 위한 다음 방어선은 대리인 투표입니다. 연결 이유로 투표에 안정적으로 참여할 수 없는 계정 보유자는 자신들의 잔고의 가중치로 투표할 수 있는 대리인을 지명할 수 있습니다. 대리인의 수와 다양성을 최대화하는 것은 네트워크의 회복력을 증가시킵니다.
4) RaiBlocks 에서 포크는 우연히 일어나지 않습니다. 그러므로 노드들은 포크된 블록을 어떻게 대응할지 정책을 결정할 수 있습니다. 공격자가 아닌 계정이 블록 포크로 인해 취약해지는 유일한 때는 공격자 계정으로부터 잔액을 받을 때입니다. 블록 포크로부터 안전하고자 하는 계정들은 포크를 만드는 계정들로부터 블록을 받는 것을 다소 기다리거나, 또는 전혀 받지 않는 것을 선택할 수 있습니다. 수신자는 또한 다른 계정들을 격리하기 위해, 의심스러운 계정으로부터 자금을 받는 별도의 계정들을 만들 수도 있습니다.
5) 아직 구현되지 않은 최종 방어선은 블록 고정입니다. RaiBlocks 는 투표를 통해 블록 포크를 확정하는데에 많은 시간을 할애합니다. 노드들은 블록들이 일정 시간이 지나면 롤백되지 않도록 고정하도록 설정될 수 있습니다. 네트워크는 모호한 포크들을 방지하기 위해 빠르게 체결하는 데에 집중함으로써 충분히 안전해집니다.

더 복잡한 버전의 $> 50\%$ 공격은 그림 9에 상세히 기술되어 있습니다. “Offline” 은 지명되었지만 투표를 위한 온라인 상태가 아닌 대리노드들의 비율입니다. “Stake” 는 공격자가 투표하는 투자의 양입니다. “Active” 는 온라인 상태의 대리노드들이며 프로토콜에 따라 투표합니다. 공격자는 네트워크 서비스 거부 공격으로 다른 투표자들을 공격함으로써 박탈해야 할 지분의 양을 상쇄할 수 있습니다. 만약 이 공격이 지속될 수 있다면, 공격받은 대리노드들은 비동기화될 것이고, “Unsync” 로 표시됩니다. 마지막으로 공격자는 이전 대리노드들이 자신들의 원장을 재동기화하는 도중에 새로운 대리노드들에 서비스 거부 공격을 함으로써 상대적인 투표 권한을 높일 수 있습니다. 이 것은 “Attack” 으로 표현됩니다.

만약 공격자가 이런 환경을 조합하여 Stake >Active 를 이룰 수 있다면, 공격자는 자신들의 지분을 이용하여 원장에 대한 투표를 성공적으로 뒤집을 수 있습니다. 이런 유형의 공격이 어느 정도의 비용이 들 것인지는 다른 시스템의 시장 규모를 추산하여 예상할 수 있습니다. 만약 33% 의 대리노드들이 오프라인이거나 DoS 공격을 받는다고 추정하면, 공격자는 투표를 이용하여 시스템을 공격하기 위해서 전체 시장 규모의 33% 를 구매해야 합니다.

Offline	Unsync	Attack	Active	Stake

Fig. 9. 51% 공격 요구 사항을 낮출 수있는 잠재적 투표 구성.

G. 부트스트랩 오염

공격자가 오래된 잔고가 있는 개인키를 오래 가지고 있을수록, 그 시점에 존재했던 대리인이 존재하지 않을 확률이 높아지는데, 왜냐하면 잔액이나 대리인이 새로운 계정으로 옮겨졌을 것이기 때문입니다. 이것은 한 노드가 공격자들이 그 시점의 다른 대리인들에 비해 투표 지분의 정족수를 가지고 있던 오래된 네트워크 상태에서 부트스트랩되면 공격자들은 그 새로운 노드에 대한 결정을 흔들 수 있게 된다는 것을 의미합니다. 만약 이 새로운 사용자가 공격 노드가 아닌 다른 노드와 상호작용하려 한다면, 그들은 다른 헤드 블록을 가지고 있기 때문에 그 트랜잭션들은 모두 거부될 것입니다. 결과적으로 악의적인 노드들이 네트워크의 새로운 노드들에게 잘못된 정보를 제공함으로써 시간을 낭비하게 할 수 있습니다. 이를 막기 위해서 노드들은 계정과 알려진 올바른 블록 헤드에 대한 초기 데이터베이스와 짝지워질 수 있습니다. 이는 제네시스 블록까지 데이터베이스를 다운받고 교체하는 것입니다. 다운로드가 최신에 가까워질수록 이런 공격에 정확하게 대항할 확률이 높아집니다. 결국 이 공격은 노드들이 초기 시작할 때 쓰레기 데이터를 공급하는 이상의 악영향은 없는데, 이런 노드들은 최신 데이터베이스를 가진 어떤 노드와도 거래할 수 없기 때문입니다.

VI. 구 현

현재 리퍼런스 구현체는 C++ 로 구현되어 있으며 2014 년부터 Github 에 공개되고 있습니다 [10].

A. 설계 특징

RaiBlocks 구현은 이 논문에 개요가 서술된 아키텍처 표준에 부합합니다. 추가적인 규격은 여기에 기술합니다.

1) 서명 알고리즘: RaiBlocks 는 수정된 ED25519 엘립틱 커브 알고리즘과 Blake2b 해싱을 모든 디지털 서명에 사용하고 있습니다 [11]. ED25519 는 빠른 서명과 검증, 그리고 높은 보안성때문에 선택하였습니다.

2) 해싱 알고리즘: 해싱 알고리즘은 오직 네트워크 스팸을 막기 위해서만 사용되기 때문에 마이닝 기반의 암호화폐에 비해 알고리즘의 선택은 중요성이 덜합니다. 우리의 구현체는 Blake2b 를 블록 콘텐츠에 대한 다이제스트 알고리즘으로 사용합니다 [12].

3) 키 유도 함수: 리퍼런스 월렛에 키들은 패스워드로 암호화되며 패스워드는 ASIC 크래킹 시도로부터 보호하기 위해 키 파생기능을 통해 제공됩니다. 현재 Argon2 [13] 는 탄력성 있는 키 유도 함수를 만드는 것을 목표로 하는 유일한 공개적인 경쟁에서 승리한 알고리즘입니다.

4) 블록 간격: 각 계정은 고유의 블록체인을 가지기 때문에 블록 갱신은 네트워크의 상태에 비동기적으로 수행될 수 있습니다. 그렇기 때문에 블록 생성 간격은 없으며, 트랜잭션들은 즉시 발행될 수 있습니다.

5) UDP 메시지 프로토콜: 우리의 시스템은 가능한 최소한의 컴퓨팅 자원을 사용하여 무기한으로 운영되도록 설계되었습니다. 시스템의 모든 메시지는 상태를 가지지 않으며 하나의 UDP 패킷에 들어가도록 설계되었습니다. 또한 간헐적인 연결을 사용하는 경량 피어가 단기 TCP 연결을 재수립하지 않고도 네트워크에 쉽게 참여할 수 있습니다. TCP 는 오직 새로운 피어들이 블록체인을 대량으로 부트스트랩하는 경우에만 사용됩니다.

노드들은 자신들의 트랜잭션이 네트워크에 수신되었는지 다른 노드들로부터의 트랜잭션 전파 트래픽을 관찰함으로써 알 수 있는데, 자신에게 여러 복사본이 돌아오는 것을 볼 수 있을 것이기 때문입니다.

B. IPv6 와 멀티캐스트

연결이 없는 UDP 상에 구축하는 것은 추후의 구현체가 전통적인 트랜잭션 흐름과 투표 전파를 대신하여 IPv6 멀티캐스트를 사용하는 것을 허용합니다. 이는 네트워크의 대역 소모를 줄이고 노드들이 발전해 나가는데 있어 더 많은 정책 유연성을 제공합니다.

C. 성능

이 글을 작성하는 시점에, 4 백 2 십만 개의 트랜잭션들이 RaiBlocks 네트워크에서 처리되었고, 1.7GB 의 블록체인 크기를 만들었습니다. 트랜잭션 시간은 수초 단위로 측정됩니다. 일반적인 SSD 에서 구동되는 현재의 리퍼런스 구현체는 초당 1 만 트랜잭션 이상을 처리할 수 있으며 주로 IO 에 제약을 받습니다.

VII. 자원 사용

여기서는 RaiBlocks 노드가 사용하는 자원을 개략적으로 설명합니다. 추가적으로 특별한 사용 사례에서의 자원 사용을 줄이는 방법에 대해서 검토합니다. 축소된 노드는 일반적으로 경량, 축약 또는 단순 지불 검증 (SPV) 노드라고 불립니다.

A. 네트워크

네트워크 활동량은 네트워크의 건강함에 네트워크가 얼마나 기여하는지에 달려 있습니다. The amount of network activity depends on how much the network contributes towards the health of a network.

1) 대리인 노드: 대리인 노드는 다른 대리인들로부터의 투표 트래픽을 감시하고 자신의 투표를 발행하기 때문에 최대한의 네트워크 자원을 필요로 합니다.

2) 무신뢰 노드: 무신뢰 노드는 대리인 노드와 비슷하지만 오직 감시만 합니다. 이 노드는 대리인 계정 개인키를 가지고 있지 않으며 자신의 투표를 발행하지도 않습니다.

3) 신뢰 노드: 신뢰하는 노드는 올바르게 합의를 수행하기 위해 신뢰하는 하나의 대리인 노드로부터의 투표 트래픽을 감시합니다. 이는 이 노드로 가는 대리인들로부터의 투표 트래픽의 양을 줄입니다.

4) 경량 노드: 경량 노드 역시 신뢰하는 노드로 최소한의 네트워크 사용만을 허용하며 관심 있는 계정의 트래픽만을 감시합니다.

5) 부트스트랩 노드: 부트 스트랩 노드는 온라인 상태로 전환하는 노드들에 대해 원장의 일부 또는 전부를 제공합니다. 이는 고급 흐름 제어가 필요한 많은 양의 데이터가 관련되어 있기 때문에 UDP 가 아닌 TCP 연결을 통해 수행됩니다.

B. 디스크 용량

사용자 요구에 따라, 다른 노드 구성은 다른 저장 장치 요구 사항을 필요로합니다.

1) 전체 이력: 모든 트랜잭션에 대한 전체 이력 기록을 보관하려는 노드는 최대한의 저장 공간을 필요로합니다.

2) 현재: 블록으로 누적된 잔액을 유지하는 설계로 인해, 노드는 합의에 참여하기 위해 각 계정에 대한 최신 또는 헤드 블록 만 유지하면 됩니다. 노드가 전체 이력을 보관하는 데 관심이 없다면 헤드 블록 만 보관하도록 선택할 수 있습니다.

3) 경량: 라이트 노드는 로컬 원장 데이터를 보관하지 않고 네트워크에 참여하여, 관심있는 계정의 활동을 관찰하거나 또는 선택적으로, 보유한 개인 키로 새 트랜잭션을 만듭니다.

C. CPU

1) 트랜잭션 생성: 새로운 트랜잭션 생성에 관심이 있는 노드는 RaiBlocks 의 속도 제한 메커니즘을 통과하기 위해 작업 증명 논스를 생성해야 합니다. 다양한 하드웨어에서의 계산은 부록 A에서 벤치마크되어 있습니다.

2) 대리인: 대리인은 블록, 투표에 대한 서명을 검증하고 합의에 참여할 자체 서명을 생성해야합니다. 대리인 노드의 CPU 리소스 양은 트랜잭션 생성보다 현저히 적으며 최신 컴퓨터의 단일 CPU 에서 작동해야합니다.

3) 감시자: 감시자 노드는 자신의 투표를 생성하지 않습니다. 서명 생성에 대한 부하가 적기 때문에 CPU 요구사항은 대리인 노드 실행과 거의 동일합니다.

VIII. 결 론

이 백서에서 우리는 참신한 블록 래티스 구조와 위임된 지분 증명 투표를 활용하는 무신뢰, 무수수료, 저지연 암호화폐를 위한 프레임워크를 제시하였습니다. 네트워크는 최소한의 자원을 필요로 하며 고성능의 채굴 하드웨어를 필요로 하지 않고 많은 수의 트랜잭션을 처리할 수 있습니다. 이 모든 것들은 각 계정이 개별적인 블록체인을 가지고 접근 문제와 글로벌한 자료 구조의 비효율성을 제거함으로서 달성됩니다. 우리는 시스템에 대해 가능한 공격 방법을 식별하고, RaiBlocks 가 어떻게 이러한 형태의 공격들에 내성을 가지는지에 대한 논증을 제시하였습니다.

Appendix A
PoW 하드웨어 벤치마크

앞서 언급하였듯이, RaiBlocks 에서 PoW 는 네트워크 스팸을 줄이기 위해 사용됩니다. 우리의 노드 구현은 OpenCL 호환 GPU 의 강점을 활용할 수 있는 가속을 제공합니다. 표 I는 실제 다양한 하드웨어의 실제 벤치마크 비교를 제공합니다. 현재 PoW 임계값은 고정되어 있으나 평균적인 연산 능력이 진보함에 따라 적응형 임계값이 구현될 수도 있습니다.

TABLE I
하드웨어 PoW 성능

디바이스	초당 트랜잭션
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

감 사

이 백서의 편집과 서식설정을 해 준 Brian Pugh 에게 감사를 표합니다.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] “Bitcoin median transaction fee historical chart.” [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html
[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.
[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.
[8] S. Popov, “The tangle,” 2016.
[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf