

LTO.network

분산형 워크플로우를 네트워크 블록체인

www.lto.network

초록

비즈니스 프로세스의 디지털화와 자동화는 생산성과 비용 절감 측면에서 큰 이점을 제공한다. 그러나 조직 간 프로세스에서는 부분적으로 신뢰가 부족하기 때문에 이러한 이점을 활용하려고 애쓴다. 비트코인은 블록체인이 어떻게 분배와 암호화를 사용하여 신뢰에 의존하지 않는 시스템을 제공하는지를 입증했다.

LTO는 탈중앙화된 워크플로우 엔진으로 구축하며, ad-hoc 협업을 기반으로 한다. 정보는 개인 블록체인(프로세스당 새로운 체인)을 사용하는 당사자 간에 공유되고 공공 블록체인에 해쉬화된다. 이러한 하이브리드 접근 방식을 통해 조직은 모든 데이터 보호 규정을 준수하고 블록체인 프로젝트와 관련된 확장성 문제를 방지할 수 있다.

도입

디지털 혁명은 우리 삶을 보다 효율적으로 만드는 많은 변화를 가져왔다[1]. 이러한 진보의 물결은 주로 사용자 및 내부 비즈니스 프로세스에서 일어났다. 조직과 조직 사이에서의 변화에 관해서, 우리는 이러한 변화들이 과감하지 않다는 것을 인정해야 한다. 편지와 팩스는 대부분 이메일로 대체되었고, 타자기는 워드 프로세서로 대체되었지만, 이러한 피상적인 변화를 넘어, 기본 과정의 실행은 거의 변하지 않았다.

자동화가 없는 주된 이유는 정보 배포가 서로의 관계[3]에 중요한 역할을 하기 때문에, 기업들이 상대방 시스템[2], 예의 의존하기를 거리기 때문이다[3]. 힘의 불균형이 존재하는 경우, 한 당사자가 주도권을 쥐어 다른 당사자가 중앙집권적 관리 시스템을 사용하도록 할 수 있다. 우리는 정부와 더 나아가 기업들이 위와 같은 현상을 다룰 때 지켜본다. 어느 한 당사자가 통제권을 주장할 수 없는 상황에서는 자동화가 발생하지 않는다[4].

조직 간 프로세스의 자동화와 관련된 문제를 해결하기 위해 사람들은 20년 이상 분산형 워크플로우로 실험을 해 왔다[5]. 이러한 연구와 실험에서, 높은 수준의 신뢰와 페어플레이가 가정되며, 주로 기술적인 도전들을 해결하는 데 초점을 맞춘다. 사실, 이것은 잘못된 가정이다, 왜냐하면 신뢰의 부족은 성공적인 조종사들이 생산되는 것을 방해하기 때문이다.

자동화가 없는 또 다른 이유는 효율성과 부패의 상관성이다[6]. 전통적으로, 대기업과 정부 기관들은 많은 사람들이 절차를 시행하도록 요구한다. 그러한 과정을 조정하기 위해서는 상당한 양의 관료주의가 요구된다. 이는 뇌물수수 비용을 증가시켜 자동화 동기를 감소시킨다. 그러나 효율성 증가는 이 효과를 부정한다.

본 논문은 모든 당사자들이 동등한 입장에서 설 수 있는 해결책을 제시하면서 블록체인을 사용하여 두 문제를 어떻게 해결할 수 있는지를 보여줄 것이다.

CONTENTS

파트 I. 라이브 계약	3
1 라이브 VS 스마트 계약	3
1.1 Ricardian Contracts	3
1.2 시행	3
1.3 사용자 인터페이스	3
1.4 지침 및 통합	3
2 Finite state machine	3
2.1 Deterministic Finite State Machine	4
2.2 Extended Finite State Machine	4
2.3 Communicating finite state machines	4
2.4 자동 계약	4
3 가지 대체 모델링 방법론	4
3.1 Petri Nets	4
3.2 BPMN	5
3.3 DEMO	5
4 시나리오	5
4.1 상태	5
4.2 행위	5
4.3 행위자	5
4.4 에셋	5
5 개의 데이터 객체	5
5.1 변경 불가능	5
5.2 구성	5
5.3 문서	5
5.4 사용자 정의 유형	6
6 신원	6
6.1 신원 초대	6
6.2 신원 업데이트	6
7 프로세스	6
7.1 행위	6
7.2 수동 행위	6
7.3 시스템 행위	6
7.4 하위 프로세스	6
7.5 투영	7
7.6 데이터 윤영자	7
7.7 수동 시험	7
8 적응형 워크플로우	7
8.1 코멘트	7
8.2 일탈	7
8.3 시나리오 업데이트	7
9 이벤트 체인	7
9.1 암호화 서명	7
9.2 해시 체인	7
10 분산	7
10.1 개인 체인	8
10.2 제네시스	8
11 컨센서스 메커니즘	8
11.1 충돌 가능성	8
11.2 분기 유효성 검사	9
11.3 우선순위	9
11.4 동기화 되지 않은 사건	9
11.5 분기 병합	9
11.6 포크	9

12	프라이버시	9
12.1	링크된 데이터	9
12.2	GDPR	9
12.3	영 지식 증명	10
13	공통 패턴	10
13.1	체인 상호작용	10
13.2	명시적 동기화	10
파트 II. 글로벌 블록체인		11
14	(중앙 집중형 VS 분산형) 동기화	11
15	합의 알고리즘	11
15.1	리스	11
15.2	라플 계수	11
15.3	Forge probability	11
15.4	공정한 PoS	12
15.5	생성기 서명	12
15.6	NG 프로토콜	12
16	개의 트랜잭션 유형	12
16.1	동기화	12
16.2	인증 및 승인	12
16.3	인증서	12
16.4	신뢰 사슬	13
16.5	스마트 계정	13
17	요약 블록	13
17.1	키 블록 크기	13
17.2	애그리게이션 없는 성장	13
17.3	분리 중인	13
17.4	애그리게이션	13
17.5	축소의 차이	14
17.6	요약 블록 크기	14
17.7	총 크기	14
17.8	기록 노드	14
18	네트워크 취약성	14
18.1	중요도 인플레이션	14
18.2	Nothing at stake	15
18.3	LPoS 중앙 집중화	15
18.4	서비스 거부 공격	15
18.5	SHA-2 취약성	15
부 III. 플랫폼		16
19	설계	16
19.1	マイ크로 설계	16
19.2	애플리케이션 계층 및 서비스	16
20	UI 계층	16
21	애플리케이션 계층	16
21.1	웹 서버	16
21.2	워크플로 엔진	16
22	개인 체인 계층	16
22.1	이벤트 체인 서비스	16
22.2	이벤트 인큐 서비스	16
22.3	이벤트 발송 서비스	16
23	공공 체인 계층	16
23.1	동기화 서비스	16
24	컨테이너 조정	17

Part I. 라이브 계약

비즈니스 프로세스 모델링은 모든 중/대 규모 조직에서 공통적인 전략이다[7]. 워크플로우 프로세스를 시각적으로 표현함으로써 분석, 개선 및 자동화할 수 있다 (그림 1). 자연언어 또는 프로그래밍 언어로 작성된 절차와는 달리, 이 모델들은 인간과 컴퓨터 모두에 의해 이해될 수 있다.

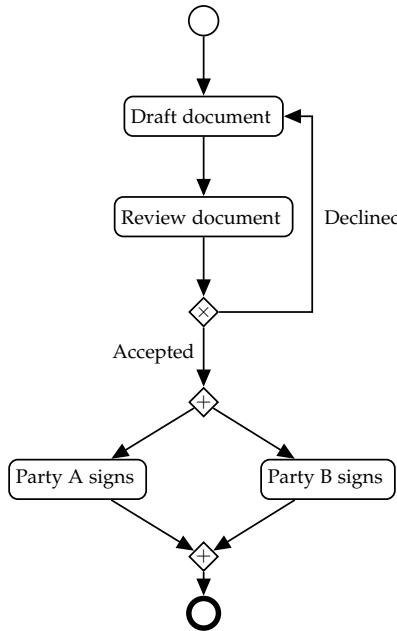


Fig. 1: BPMN 다이어그램을 사용하여 워크플로우를 시각화할 수 있다.

조직간 협력을 위해서, 모델화는 단지 의사소통을 향상시키기 위해서만 이루어지는 것이 아니다. 관련 당사자는 LTO 플랫폼에서 라이브 계약(구속력 있는)[8]; 사용 절차를 명시해야 한다 **Live Contract**.

LTO 플랫폼은 각 라이브 계약에 대해 ad-hoc 개인 블록체인을 생성한다. 그러한 블록체인은 변경할 수 없는 원장 역할을 하려는 것이 아니라 모든 당사자 사이 부서된(이미 다른 사람이 서명한 문서를 인정하는 서명을 하는 것) 사건 및 공유된 상태를 갖도록 하기 위한 것이다.

1 라이브 VS 스마트 계약

라이브 계약에서는 이더리움[9]과 비슷한 목표를 가지고 있다. 신뢰가 필요 없으며, 검증 가능한 방식으로 적용할 수 있는 논리를 정의하고 공고히 한다.

그러나 이 두 유형의 디지털 계약의 이면에 있는 철학은 상당히다르다. 이더리움은 스마트 협약을 값이 포함된 암호화 "상자"라고 설명하며, 특정 조건이 충족되어야만 잡금이 해제된다[10].

라이브 계약은 직접 가치를 보유하지 않고 둘 이상의 당사자가 상호 작용하는 방식을 기술한다. 그들의 의도는 전통적인 계약서에 훨씬 가깝다.

1.1 Ricardian Contracts

라이브 계약은 Ricardian contract¹[11]. 가장 주목할 만한 것은, 사람과 프로그램이 쉽게 읽을 수 있다는 것이다. 이것은 라이브 계약의 정의로부터 얻은 창발성이다. 법적 목적을 위한 자연스러운 언어 버전도 없고 프로그래밍된 실행을 위한 코드화된 버전도 없다.

1. Ricardian Contract은 아래와 같이 단일 문서로 정의되는데 a) 발행자가 보유자에게 제공하는 계약, b) 보유자가 보유하며 발행자가 관리하는 가치 있는 권리, c) 개인이 쉽게 읽을 수 있는 사람(예: 서류상 계약), d) 읽을 수 있는 프로그램(데이터베이스처럼 가능한), e) 전자 서명, f) 키와 서버 정보를 전달 g) 고유하고 안전한 식별자와 제휴.

1.2 시행

온 체인의 집행은 많은 실제 사건에 잘 맞지 않는다. 스마트 계약은 선제적 시행에 의존하는데, 이는 계약을 위반하는 것이 불가능하거나 어느 한쪽이 탈퇴할 수 있어야 한다는 것을 의미한다[12].

비공개 계약을 예로 들어보자. 블록체인은 일방 당사자가 정보를 공개하는 것을 막을 수 없으며, 위반사항 해결에 적극적으로 참여하도록 강요할 수도 없다. 그러한 계약이 자율시행 합의[13]로 작용하려면, 그것은 완전한 벌금을 보증금으로 유지해야 한다. 이것은 모든 당사자들이 결의안에 참여하는 것이 그들의 최선이라는 것을 보장한다.

많은 양의 자금을 임의계약에 대한 벌금의 보증금으로 묶어야 하는 것은 대부분의 기업[14]에서는 비현실적이다[14]. 또한, 위약금 이자 및 이와 유사한 조치의 효과성은 스마트 계약이 보유하는 가치로 제한된다.

대부분의 업무 과정은 권위 있는 주체를 통한 오프체인 분쟁 해결을 요구한다. 라이브 계약서는 분쟁 해결을 용이하게 할 수 있다. 여기에는 분쟁 협상, 중재, 심지어 중재자 또는 재판관이 포함될 수 있다.

LTO 플랫폼에서 프로세스를 실행하면 검증 가능한 이벤트 기록을 형성하여 비대칭적 정보의 양을 줄인다. 정보의 배포는 분쟁의 경우 협상에 영향을 미치고[3] 권위 있는 제3자가 수행한 평가에 영향을 미친다.

1.3 사용자 인터페이스

이더리움은 수학적으로 정의할 수 있는 스마트 계약 또는 거래 유형을 구축하기 위해 프로그래머가 사용할 수 있는 튜링 완전(Turing-Complete) 스크립팅 언어를 내부적으로 제공한다[10]. 이것은 그것을 매우 추상적으로 만든다, 왜냐하면 실행계약에 포함된 상태는 본질적인 의미가 없기 때문이다.

이러한 계약과 상호작용하기 위해서는 특정 스마트 계약에 대해 사용자 인터페이스를 구축하거나 그러한 계약의 인터페이스를 보다 정확하게 만들어야 한다[15]. 이러한 인터페이스는 ERC-20[16] 및 ERC-721[16]과 같이 표준화되어 UI를 계약 논리에서 분리할 수 있다. 단점은 그것이 또한 계약을 설계할 때 가능성을 제한한다는 것이다.

라이브 계약에서는 정보가 본질적인 의미를 갖는다. 이는 사용 사례를 제한하지만, 계약과 그 프로세스에서 제공하는 데이터에 전적으로 기초하는 인터페이스를 생성할 수 있다. 따라서 각 워크플로우에 대해 특정 UI를 작성할 필요 없이 모든 워크플로우를 디지털화하고 LTO에서 실행할 수 있다.

1.4 지침 및 통합

라이브 계약에는 특정 개인이나 노드를 위한 지침이 포함되어 있다. 노드는 지시에 따라 행동하거나 사용자에게 행위를 수행할 것으로 예상됨을 통지할 수 있다. 그러한 행위는 HTTP API 호출을 통해 인터넷에서 정보를 얻는 것을 포함할 수 있다. 이더리움과 하이퍼렛저에서 구현된 스마트 계약의 논리는 블록체인 상태를 바꿀 수 있을 뿐이다. A smart contract requires an oracle to submit data from an external source to the blockchain[17]. 스마트 계약에는 외부 출처의 데이터를 블록체인[17]에 제출하는 오라클이 필요하다. 이 오라클은 스마트 계약 상태에 따라 작용할 수 있지만, 오라클의 논리는 계약의 일부가 아니다. 그러한 논리는 라이브 계약의 일부이므로 관련된 모든 당사자들에 의해 검증되고 논의될 수 있다.

2 FINITE STATE MACHINE

라이브 계약에서는 워크플로우를 Finite State Machine (FSM)[17]으로 정의한다. 이를 통해 플로우차트로 시각화할 수 있다 (그림 2). 그렇게 하면 인간과 컴퓨터 모두에서 작업흐름을 이해할 수 있다.

2.1 Deterministic Finite State Machine

모든 블록체인 논리는 결정론적이어야 한다[18]. 컴퓨터 프로그램이 이를 준수하기 위해 추가적인 노력이 필요할 수 있는 경우, Deterministic Finite State Machine (DFSM)은 정의에 의해 결정론적이다.

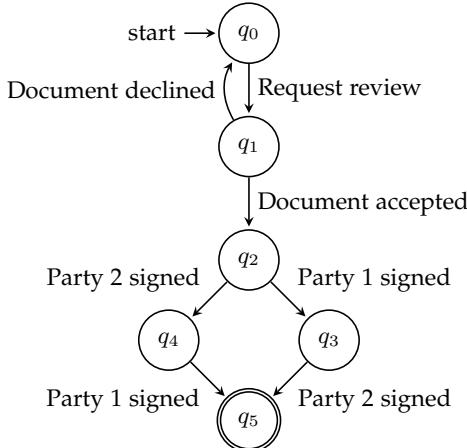


Fig. 2: 흐름도로 시각화된 Finite State Machine의 예

2.2 Extended Finite State Machine

그림 2는 한 상태에 도달하기 위해 여러 조치가 필요하지만, 그 발생 순서가 자의적일 때 문제가 발생하는 방법을 보여준다. 이는 그림 2에서와 같이 가능한 모든 주문에 대한 전환 경로로 모델링할 수 있다. 그러나, 이 접근방식으로 보면, 상태의 수 및 상태 전환은 행위 수에 따라 기하급수적으로 증가한다. 이렇게 하면 워크플로우의 시각화가 멀 명확해질 뿐만 아니라 워크플로우를 정의하기가 더 어렵고 오류가 발생하기 쉽다.

그렇기 때문에 일반 FSM을 사용하는 대신에, 라이브 계약은 조건적인 상태 전환을 허용하는 Extended Finite State Machine[19] (EFSM)를 사용한다.

그림 3은 EFSM을 사용하는 그림 2와 동일한 워크플로우를 정의한다.

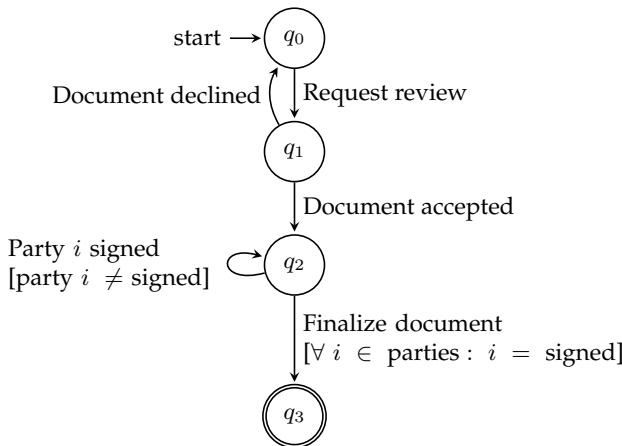


Fig. 3: Example of an Extended Finite State Machine: conditions in brackets have to be true for the transition to be valid

2.3 Communicating finite state machines

Finite State Machines은 순차적 동작으로 제한되며; 동시 프로세스를 지원하지 않는다. 동시에 워크플로우를 표현하기 위해 별도로 명령어 시퀀스는 개별 FSM으로 나타낼 수 있다. Communicating finite state machines (CEFSMs)하면 이벤트의 개별 조건들을 결합하여 보다 복잡한 프로세스를 모델링할 수 있다.

이벤트 체인 (섹션 9 참조)은 두 개의 FSM 사이의 통신 채널 역할을 할 수 있다. 두 프로세스가 서로 다른 이벤트 체인을 사용하여 분리되는 경우, 통신 채널은 비결정적인 것으로, 다시 전체 시스템을 비결정적으로 만든다[20]. 13.1 절과 같이 이벤트를 승인하면 이를 극복할 수 있다.

2.4 자동 계약

A Finite State Machine은 당사자들이 상대방에게 부과하는 의무, 허가 및 금지를 공식화함으로써 참가자들 간의 합의로 적용될 수 있다 [21]. 금융 계약[22]과 서비스 계약[23]과 같은 계약은 완전히 FSM으로 디지털화할 수 있다.

그러나 이 논문[23, 24]에 제시된 표현으로는 프로세스 내의 조정, 통신 및 연출을 정의하지 않기 때문에 워크플로로 사용할 수 없다. 이러한 요소들이 통합될 수 있지만, 그것은 FSM을 기하급수적으로 더 복잡하게 만든다[24].

현실적으로, FSM은 기껏해야 불완전한 계약을 나타낼 것이다. 차이는 디폴트 규칙으로 채워질 수 있기 때문에 이것이 반드시 문제 가 될 필요는 없다[25]. 이 시스템은 8절과 같이 특정 상황을 해결하기 위해 또는 일반적으로 라이브 계약의 재협상을 허용한다 8.

또 다른 주목할 것은 공정의 모든 행위가 구속력 있는 인자로 구성되는 것은 아니라는 점이다. 예를 들어, 그림 1에서 문서 본문의 수락은 구속력 있는 계약을 구성하지 않는다; 이는 문서가 서명될 때만 발생한다. 이러한 구별을 용이하게 하기 위해, 행위는 정보 제공 또는 수행 능력으로 분류될 수 있다[26].

3 가지 대체 모델링 방법론

Communicating EFSMs는 일반적으로 통신 시스템과 기타 실시간 시스템[28]을 설명하는 데 사용되지만 비즈니스 프로세스는 사용하지 않는다[27].

분산형 조직 간 프로세스를 모델링할 때 워크플로우에 대한 보다 일반적인 표기법은 추가적인 과제를 제공한다.

3.1 Petri Nets

Petri nets[28]는 복수의 독립적인 활동이 동시에 진행되는 시스템을 그래픽으로 나타낸 것이다. 복수의 동시 활동을 모델링하는 능력은 FSM과 Petri Nets을 구별한다. FSM에는 다음에 어떤 행위가 발생할 수 있는지를 결정하는 하나의 “현재” 상태가 항상 존재한다. Petri Nets에는 여러 개의 상태가 있을 수 있으며, 그 중 한 개는 Petri Nets 상태를 변화시킴으로써 진화할 수 있다. 이러한 상태 중 일부, 또는 모든 것이 동시에 진화하여 Petri Nets에 대한 몇 가지 독립적 변경이 즉시 발생할 수 있다[29].

비즈니스 프로세스를 설명하는 데 사용될 수 있는 워크플로우 네트(WF-net)는 Petri Nets의 서브셋이다.[30]. WF-net은 하나의 시퀀스만이 아니라 전체 과정을 설명할 수 있다.

EFSM은 정보를 보유하기 위해 글로벌 상태를 사용한다. 이 정보는 각 시퀀스에 따라 격리되어야 한다. 데이터를 불변화할 수 없는 경우 제5.1절에 설명한 것처럼 이용할 수 있다[5.1]. Petri Nets와 함께 글로벌 정보를 사용하는 것은 가능하지 않다. 대신에 정보는 워크플로우를 통해 흐를 필요가 있다.

CEFSMs의 접근으로 시퀀스는 개별 프로세스로 정의된다. 이것은 전체 과정의 일부에 접근 제어를 적용하는 것을 사소한 과제로 만든다. 워크플로우를 전체적으로 나타낼 때, 이것은 동일한 방식으로 설명할 수 없다.

Petri Nets에는 재미있는 기법이 있다. 여러 연구 결과[31]에 따르면 이러한 연구들이 비즈니스 워크플로우를 나타내는 데 사용될 수 있다고 한다. CEFSMs를 Petri Nets로 모델링할 수 있는 경우, WF-네트를 사용하는 것이 개별 FSM을 사용하는 것보다 바람직할 수 있다.

Petri Nets와 FSM의 유사성 때문에, WF 네트로의 전환은 이 문서에서 설명한 것처럼 우리의 솔루션을 근본적으로 바꾸지 않는다.

3.2 BPMN

BPMN(Business Process Model and Notation)은 비즈니스 모델 프로세싱의 업계 표준이며 LTO의 모델링 표기법 유력하다. 그러나 조직 간 시스템에는 특히 문제가 되는 많은 제약이 있다[31].

BPEL(Business Process Execution Language, BPEL)은 전형적으로 BPMN과 연계되어 있으며, 웹 서비스를 위한 완전한 언어를 제공하는 비결정적인 Turing이다[32]. 이것은 블록체인 자동화에 적합하지 않다.

이에 대한 대안은 BPMN 모델을 Petri Nets로 변환하는 것이다, 즉 차례로 스마트 계약으로 변환하는 것이다[33].

(완전한) 스마트 계약으로의 변환은 불필요하지만, 현재의 산업 표준을 지원하기 위해 BPMN에서 Petri Nets(또는 CEFMS)로의 변환은 흥미로울 수 있다.

3.3 DEMO

"DEMO"는 "설계 및 엔지니어링 방법 - 조직을 위한 로고"의 약자다. 조직과 그 사업 프로세스를 기술하기 위한 이 방법론은 "상호적 행위"에 기초한다. 그것은 건설 모델(CM), 공정 모델(PM), 행위 모델(AM), 사설 모델(FM)[35] 등 네 가지 모델을 사용하여 전체적인 보기를 생성한다[34].

DEMO는 프로세스의 각 단계를 개별적으로 고려하는 대신 모든 수행 거래별로 일반화된 워크플로우를 구축한다. 그러한 거래에는 개시자와 실행자의 두 가지 역할이 있다. 표준 순서는 다음과 같이 진행된다: 개시자는 요청을 하고 실행자는 약속을 한다. 실행자는 행위를 수행하고 결과에 대해 진술한다; 개시자는 이를 수용하거나 거래를 완료하거나 거부한다[26].

요청을 거절하는 이전 실행자, 요청을 취소하려는 개시자, 약속을 이행할 수 없는 실행자 등과 같은 다른 대체 흐름도 모델링된다. 이러한 대체 흐름은 실제로 항상 존재하는 반면 다른 모델링 방법론을 사용할 때 종종 모델링되지 않거나 부분적으로만 모델링된다.

프로세스 모델(PM)은 이러한 거래를 결합하여 완전한 비즈니스 프로세스를 모델링한다. 이 모델과 워크플로우의 차이점은 이 모델에서 모든 사람이 가능한 한 많이 병렬로 작업하며, 필요한 경우 트랜잭션 간의 종속성을 명시한다는 것이다. 이 설계 선택은 DEMO 모델이 다른 모델링 방법에 비해 현재 프로세스에 대한 명확한 개요를 제공하지 않는다는 것을 의미한다. 더욱이, 상호간의 정보 배제(서명할 준비가 된 문서를 편집하지 않음)는 생동감이 없다. 대신에, 그것은 구체적으로 명시될 필요가 있다.

DEMO는 높은 수준의 모델을 만드는 좋은 방법일 수 있으며, 이는 세밀한 조정 이상의 워크플로우를 산출한다. 이는 보다 완전한 계약을 맺어, 편차에 대한 의존도를 감소시켜야 한다(제 8.2 절 참조).

4 시나리오

워크플로는 데이터 객체, 시나리오로 정의된다. 그것은 다음과 같은 요소로 구성된다:

- q_x as a state with q_0 as the initial state,
- Q as the set of all possible states $Q = \{q_0, \dots, q_{n-1}\}$,
- σ_x as an action,
- Σ as the set of all possible actions $\Sigma = \{\sigma_1, \dots, \sigma_n\}$,
- δ as the transition function $\delta : Q \times \Sigma \rightarrow Q$,
- F as the set of final states with $F \subset Q \mid F = \emptyset$,
- \bar{I} as all actor definitions,
- \bar{A} as all asset definitions,
- D as the set of embedded data-objects.

4.1 상태

설정 Q 의 상태는 일반적으로 다음과 같이 구성된다:

- 제목: 상태의 짧은 제목,
- 과(와) 같은 트랜잭션 관련 행위 집합 $\{(\vec{q}_x, \delta), \dots\}$,
- 설명: 상태에 대한 긴 설명,
- 특정 행위자에 대한 지도 또는 행위자.

상태는 이 상태에서 수행될 수 있는 행위를 기술하며, 상태 전환을 포함한다. 이것은 다른 상태에서 행위를 사용할 수 있도록 한다.

4.2 행위

시나리오에서는 워크플로우에서 수행할 수 있는 모든 가능한 행위의 집합인 Σ 을 정의한다. 그러한 행위의 예는 양식 작성, 문서 검토 및 HTTP 호출을 포함한다. 행위 유형 및 개체 속성은 JSON Schema를 사용하여 정의한다.

행위는 set I 행위자들 중 어느 쪽이 그것을 실행할 수 있고 선택적으로 실행을 위한 추가 제약조건들을 정의한다. 이러한 제약 조건으로 인해 시나리오는 확장 FSM이 된다.

행위가 실행되면 상태 전환이 시작된다. 행위는 인간의 간섭을 실행해야 하는 수동 행위와 시스템에 의해 자동으로 실행될 수 있는 시스템 작용으로 분류될 수 있다.

선택적으로, 행위는 응답의 데이터를 사용하여 행위자와 에셋을 업데이트하기 위한 지침을 정의할 수 있다.

4.3 행위자

Set \bar{I} 는 프로세스에서 역할을 수행할 수 있는 모든 행위자를 정의한다. 각 행위자는 JSON Schema를 사용하는 개체로 정의된다. 과정과 관련된 행위자 특성을 정의해야 한다.

시나리오에서 행위자는 정적인 정의에 불과하며, 이는 과정 중에 순간화될 수 있다.

4.4 에셋

\bar{A} 프로세스에서 이용 가능한 모든 에셋을 정의한다. 에셋은 가변 데이터 객체다. 프로세스와 관련된 속성을 정의해야 한다.

시나리오에서는 자산의 구조만 정의한다는 점에 알아야 한다. 에셋은 과정 내에서만 순간화 할 수 있다.

5 개의 데이터 객체

시나리오는 제외하고 다른 유형의 데이터 객체들이 정의될 수 있다. 시나리오를 포함한 모든 데이터 객체는 JSON 스키마를 유형 정의로 사용한다. 이들의 일반적인 예는 양식, 문서 및 템플릿이다.

데이터 객체는 프로세스에 포함되거나 독립적으로 연결 및 저장될 수 있다.

연결된 객체는 해당 JSON 표현의 SHA256 해시에 의해 식별된다. JSON 인코딩이 항상 동일한 결과를 산출하도록 하기 위해, JSON 인코딩의 결정론적 방법을 적용한다.

5.1 변경 불가능

데이터 객체는 데이터 객체가 수정될 때 새로운 데이터 객체가 발생하는 정도까지 변경할 수 없다. 데이터 객체가 프로세스 내에 에셋으로 포함되어 있으면, 기존 객체가 수정된 객체로 교체된다.

특히 데이터 객체를 여러 프로세스에서 사용할 수 있는 경우, 한 프로세스에서 객체를 변경하면 다른 프로세스로 자동으로 전파되지 않는다.

그렇게 하지 않으면 (부당하게) 이용할 수 있는 상황으로 이어질 수 있다. Figure 1 그림 1에서 우리는 문서를 협상하고 서명하는 과정을 시연했다. 서명 순서 중에는 문서가 수정되지 않아야 한다는 것이 분명하다.

5.2 구성

형태의 정의는 JSON Schema를 사용하여 양식 작성에 따른 데이터 구조를 정의한다. 옵션 동맹인 추가 UI 스키마를 사용하여 해당 필드를 렌더링하고 표시하는 방법을 지정할 수 있다.

이에 대한 몇 가지 유사한 구현이 있다[35-38]. 우리의 목표는 이 프로젝트들과 함께 협력하여 통일된 표준을 만드는 것이다.

5.3 문서

디지털 작업흐름은 종이 문서의 필요성을 크게 줄일 수 있다. 그러나 법률 준수, 이전 버전과의 호환성, 회사 정책과 같은 것들은 여전히 문서 사용을 필요로 할 수 있다. 템플릿을 라이브 계약의 일부로 정의하면 프로세스에서 수집한 데이터를 사용하여 자연 언어 문서를 생성할 수 있다.

템플릿 작성은 위한 필드 및 조건 섹션을 지원하는 Open Document Format[39]을 사용할 것을 권장한다.

5.4 사용자 정의 유형

개체를 정의하는 모든 JSON 스키마는 데이터-객체 유형으로 사용할 수 있다. 다른 당사자가 해당 유형을 지원하지 않는 노드를 통하여 참여할 수 있기 때문에, 사용자 정의 유형은 워크플로가 제대로 작동하지 않을 위험을 수반한다. 알 수 없는 유형의 데이터는 “있는 그대로” 저장되며 프로세스의 맥락 밖에서 사용할 수 없다.

6 신원

하나의 신원은 라이브 계약 내에서 사람, 팀 또는 조직을 정의한다. 하나의 신원은 항상 다음 정보를 포함한다.

- 식별자,
- 노드 URI,
- 사용자 정보,
- 서명 키,
- 암호화 키.

신원들은 행위자와 같지 않다. 행위자는 “학생”과 같은 추상적인 역할이지만, 신원은 “브루스 윌리스”나 “아킴 코퍼레이션”이 될 수 있다.

서명 키는 신원과 연결된 하나 이상의 공용 키가 있는 맵이다. “사용자” 키는 신원에 속하며 행위 서명에만 사용할 수 있다. “시스템” 키는 신원이 사용하고 자동화된 작업에 서명하는 데 사용되는 노드에 의해 소유된다. 프로세스 내에서 다른 핵심 키를 정의할 수 있다.

공용 암호화 키는 이 신원 의해서만 해독될 수 있는 데이터를 암호화하는데 사용될 수 있다.

6.1 신원 초대

프로세스에 당사자를 추가하려면 시나리오에서 다른 신원을 추가하기 위한 행위를 정의해야 한다. 프로세스 내에서 공용 키를 알고 있으면 신원을 직접 추가할 수 있다.

공개 키를 모를 경우, 신원 확인을 요청해야 한다(그림 4). 이것은 이메일을 포함하여 충분히 안전하다고 여겨지는 어떤 수단을 통해서도 이루어질 수 있다. 초대 시스템은 1회용 키를 만들어 초청된 신원으로 보낸다. 초청된 당사자는 이 키를 자체 보안 “사용자” 및 “시스템” 키로 대체해야 한다.

새로운 신원이 프로세스에 완전히 참여하기 전에 추가 인증이 필요할 수 있다. 이것은 SMS 검증에서 연방 신분 확인 및 심지어 공증 승인까지 포함할 수 있다.

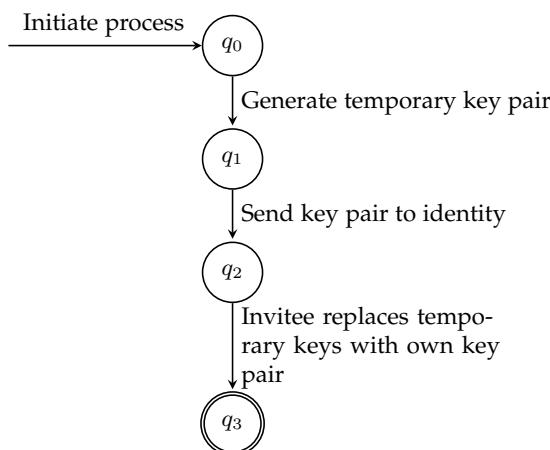


Fig. 4: Inviting an identity to join the process.

6.2 신원 업데이트

신원은 신원 확인자를 제외하고 자신의 정보를 자유롭게 수정할 수 있다. 이것은 또한 한 당사자가 다른 노드로 전환할 수 있도록 한다. 사용자가 노드를 전환하는 것이 허용되지 않아야 하는 경우, 그러한

변경을 거부하는 것은 신원자의 노드에 달려 있다. 서명 키, 암호화 키 및 노드 URI를 삭제하여 신원을 제거할 수 있다.

다른 신원 업데이트는 해당 작업이 시나리오에서 정의되고 현재 상태에서 허용되는 경우에만 가능하다.

7 프로세스

시나리오가 워크플로우에서 어떠한 상태가 아닌 경우, 프로세스는 (컴퓨터 보안 장치가) 네트워크 연결 상태를 추적할 수 있는 인스턴트를 생성, 다음과 같이 구성된다.

- θ_x as response, where $f : q_x \mapsto \theta_x$
- Θ as an ordered list of all responses $\Theta = \{\theta_0, \dots, \theta_n\}$
- q_t as the current state
- I as set of all available actors
- A as set of created assets

7.1 행위

행위를 실행하는 것은 항상 반응을 낳는다. 이 응답은 행위자가 서명하고 새로운 이벤트로 제출해야 한다. 노드는 현재 상태 및 실행된 행위에 따라 독립적으로 새 상태를 결정한다.

시나리오는 deterministic FSM으로 정의된다. 그러나 이것은 상태 전환과 투영에만 관련이 있다. 이더리움 및 Hyperledger와 같은 시스템에서는 모든 노드가 실행되므로 모든 논리가 결정론적이어야 하며, 모든 시스템에 동일한 결과를 제공해야 한다[40].

LTO를 사용하면 단일 노드 또는 단일 행위자만 작업을 실행하므로 행위가 결정적일 필요가 없다. 1.4절에서 설명한 것처럼, 외부 소스에서 데이터를 가져오는 등의 작업에는 오라클이 필요하지 않다.

7.2 수동 행위

LTO 플랫폼에 구축된 애플리케이션은 현재 상태에서 수행할 수 있는 행위에 대해 인간 행위자에게 알려야 한다. 인간 행위자는 그것을 그의 노드로 제출하기 전에 자신의 응답 이벤트에 서명할 것이며, 그것은 그것을 모든 당사자들에게 배포할 것이다.

7.3 시스템 행위

시스템 행위은 사람의 간섭을 필요로 하지 않고, 노드에서 자동으로 실행된다. 따라서, 인간 사용자 대신에 노드는 응답에 서명한다.

이러한 조치는 항상 단일 시스템에 의해 수행되며, 결정론적일 필요는 없다. 프로세스의 다른 당사자가 응답을 검증하고, 필요한 경우 거부할 수 있다. 시스템 행위에 관련된 인간 간섭이 없기 때문에, 행위는 행위자 대신에 시스템 자체에 의해 서명된다.

또한 나중에 실행되도록 시스템 행위를 예약할 수도 있다. 이것은 시나리오에 명시되어 있다. 이것은 상태에 대한 시간 제한을 허용하거나 미리 결정된 신호에서 외부 소스를 가져올 수 있다.

시스템 동작은 자동으로 실행되며 잘못 사용하면 오류를 발생하거나, 실패할 수 있다. 이러한 행위의 경우 한 개에서 두 개의 상태 전환이 정의되어야 한다. 하나는 성공적인 실행을 위한 것이고 다른 하나는 오류의 경우이다.

7.4 하위 프로세스

FSM에 기반한 프로세스는 한 번에 한 가지 상태만 될 수 있다. 하위 프로세스들은 라이브 계약이 복수의 상태를 유지하고 다른 절차들이 동시에 수행될 수 있도록 한다. 이러한 프로세스는 이벤트 체인을 공유하지만 각 프로세스의 데이터는 여전히 격리된 상태로 남아 있다.

하위 프로세스를 용이하게 하기 위해, 라이브 계약에는 주요 시나리오에서 인스턴스화할 수 있는 하위 시나리오가 포함될 수 있다.

7.5 투영

FSM 상태를 제외하고, 이 프로세스에는 에셋 및 행위자와 같은 다른 상태 저장 데이터도 포함된다. 모든 응답의 페이로드가 이 데이터를 업데이트하는데 사용될 수 있다. 페이로드가 데이터를 업데이트하는 방법에 대한 규칙이 시나리오에 정의된다. 투영을 업데이트하는 것은 결정론적이어서 시나리오에 대해 지정된 반응 집합을 적용하면 항상 동일한 투영이 발생한다.

투영은 행위에 대해 정의된 제약 조건뿐만 아니라 행위의 매개 변수를 설정하는 데 사용할 수 있다(4.2 참조).

7.6 데이터 운영자

데이터 운영자는 투영이 프로세스에 미치는 영향을 지정하는 시나리오에서 사용할 수 있다. 이 운영자들은 부작용이 없는 최소한의 기능들이다. 그것들은 산술 또는 논리 연산을 위해 사용될 수 있다. 이러한 작업의 결과는 투영에 저장될 수 있으며 상태 전환의 기반으로 사용할 수 있다.

7.7 수동 시험

시스템 동작만으로 구성된 루프를 포함하는 시나리오는 무한 루프를 초래하여 대량의 트랜잭션을 야기할 수 있다. 시나리오를 검증할 때, 우리는 그러한 구성이 있는 경우 이를 거부하고자 한다.

프로그램이 영원히 실행될 수 있는지를 결정하는 것은 중단 문제[41]로 알려져 있다. 이 문제는 Turing-완성 기계에서 해결할 수 없는 것으로 판명되었지만, FSMs에서는 해결될 수 있다[42]. FSM은 제한된 수의 전환 경로를 가지고 있기 때문에, 그들 모두는 루프를 확인할 수 있다.

EFSM 모델에 대한 수동 시험은 실현 불가능한 경로를 전제로 하여 복잡하며 개방형 연구 문제다[19, 43]. 간단한 이유로, 우리는 FSM을 통과하는 어떤 경로도 조건을 무시함으로써 실현 가능하다고 가정할 수 있다. 우리는 이것이 잘못된 긍정의 원인이 될 수 있다는 것을 인정한다.

8 적응형 워크플로우

시나리오는 프로세스의 가장 일반적인 사례를 모델링한다. 모든 상황을 미리 예측하는 것은 불가능하며 가능한 모든 경우를 모델링하는 것은 지루하다. 법칙주의를 채택하는 것은 그 제도를 경직시킬 것이다. 대신에 라이브 계약은 그러한 문제를 해결하는 세 가지 방법을 지원한다.

8.1 코멘트

코멘트는 다른 신원자들과 의사소통하기 위해 사용된다. 예를 들어, 그들은 갈등을 해결하거나 프로세스 밖에서 토론을 수행하기 위해 사용될 수 있다. 오프 체인 통신 방법 대신에 코멘트를 사용하면 대화가 블록체인에 기록되도록 보장할 수 있다. 그것은 또한 어떤 대화가 행해진 절차에서 언제 일어났는지 역추적하는 것을 허용한다.

코멘트는 문자메시지로 제한되지 않는다. 또한 이미지나 문서를 사용하여 의사소통을 돋는 것도 가능하다. 코멘트는 프로세스의 일부가 아니며, 코멘트를 추가하는 것이 상태 전환을 발생하지 않는다는 것을 의미한다. 따라서, 절차에서 미리 정의되지 않은 주제에 대한 논의를 수행하는 것이 항상 가능하다.

8.2 일탈

어느 당사자도 부분 시나리오를 정의하여 주 흐름에서 일탈을 제안할 수 있다. 이 하위 흐름은 기존 시나리오의 상태 중 하나에서 시작하여 해당 시나리오의 상태로 끝나야 한다. 일탈 흐름은 프로세스가 기존 상태로 돌아갈 때 더 이상 사용할 수 없으므로 한 번만 실행된다.

모든 당사자들은 이 일탈에 대해 합의할 필요가 있다. 일탈은 수동적으로 충돌 해결을 통해서만 해결할 수 있는 포크로 이어질 수 있다.

일탈은 분쟁을 해결하는 데 사용될 수 있다. 어떤 당사자도 이를 이전 사건의 정확성에 대해 이의를 제기하고 그것을 수정하는 방법에 대한 해결책을 제시하도록 제안할 수 있다.

일탈을 사용하는 전형적인 경우는 지급약정을 하는 것이다. 조직들은 명백히 그 선택사항을 계약 내에서 알리기를 원하지 않는다. 미리 정의된 하위 흐름은 그러한 조정을 허용하면서 기밀로 유지할 수 있다.

8.3 시나리오 업데이트

예를 들어, 협정이 업데이트되거나 새로운 법이 통과될 때 실행 과정의 시나리오를 변경해야 할 수 있다.

당사는 일탈의 흐름을 통해 주어진 과정에 대한 새로운 시나리오를 제공할 수 있다. 이러한 흐름은 상태를 구시대적인 시나리오에서 벗어나 새로운 시나리오로 이동시킨다. 시나리오 업데이트는 이벤트의 순서 중 하나이기 때문에, 해결의 결정론적 요소를 깨뜨리지 않는다.

9 이벤트 체인

FSM과 투영의 상태를 결정하려면, 주어진 순서로 반응 집합을 처리해야 한다. 이벤트 삽입 또는 제거, 사건 순서 변경 또는 페이로드 수정은 근본적으로 다른 상태를 초래할 수 있다.

중앙 집중식 솔루션에서, 통제 당사자는 데이터 무결성을 책임진다. 모든 구성원들은 통제 당사자가 하나의 진실의 근원을 대표하기 때문에 이 당사자에 의존한다. 분산형 시스템에서는 권력과 책임이 모든 당사자에 의해 공유된다.

이를 용이하게 하기 위해, 이벤트 체인은 임시 개인 블록체인처럼 작용한다. 각 행위는 하나의 동작으로 블록으로 볼 수 있는 이벤트에 싸여 있다. 이 이벤트들은 당사자들 사이에 공유되는 해시 체인을 형성한다. 합의 알고리즘은 당사자들이 사건의 순서에 동의하도록 한다.

9.1 암호화 서명

아무도 다른 사람의 사건을 조작하거나 조작할 수 없도록 각 이벤트는 비대칭 암호학을 사용하여 제출되기 전에 서명된다. 서명된 이벤트는 또한 영수증의 역할을 하며, 다른 당사자들이 서명 신원에 의해 조작이 수행되었음을 증명할 수 있도록 한다.

플랫폼은 ED25519[44] 서명을 사용한다. 이러한 elliptic curve signatures는 NIST[45] 및 ENISA[46]와 같은 기관에 의해 광범위하게 사용되고, 잘 지원되며, 승인된다. Elliptic curve 암호법은 보안을 잃지 않고 단일 서명 확인 및 서명 속도를 높일 수 있다. 그것은 또한 키와 서명 둘 다 필요한 크기를 줄인다. 구성원들이 여전히 자신들의 사건을 조작하거나 조작할 수 있기 때문에, 이 방법 자체로는 완전한 보안을 부여하지 않는다는 점에 유의해야 한다. 즉, 암호화 서명은 사건이 일어나지 않았다는 것을 증명할 수 없다.

9.2 해시 체인

각 이벤트는 SHA-2 256비트 해시를 사용해 고유하게 식별할 수 있다. 이 업계 표준 알고리즘은 충돌뿐만 아니라 사건 이미지 및 두 번째 이미지 공격에 대한 빠른 연결성과 내성을 보장한다[47]. 이것은 NIST[48]가 권장하는 암호화 해싱 알고리즘이다.

이전 이벤트의 해시를 다음 이벤트의 해시에 포함시키면 해시 체인이 생성되며, 해시 체인은 이벤트의 연대순을 기록한다. 암호화 서명과 함께 사용할 경우 해시 체인은 특정 이벤트 순서가 현재 상태를 초래했다는 것을 입증하는 적절한 척도를 제공한다[49].

10 분산

각 당사자는 중앙 서버 또는 상호로부터 정보를 얻도록 요구하기보다는, 이벤트를 관련된 다른 모든 당사자의 시스템으로 밀어 넣을 책임이 있다.

시스템이 항상 사용 가능해야 이벤트가 사라지지 않는다. 메시지 큐의 디커플링과 사용은 일시적인 가용성 문제로 문제를 감소시킨다. 일반적인 경우, 모든 당사자는 자신이 신뢰하는 노드에 접속하여

그들을 위한 이벤트를 수신하고 처리한다. 이 노드는 더 큰 시스템의 일부분이다 (섹션 19.1 참조).

조직과 정부에 초점을 맞추는 것은 이러한 조직들이 노드를 운영하는 것에 달려 있다. 사용자는 자신의 조직의 노드 또는 프로세스에 참여하기로 선택한 공개적으로 사용 가능한 노드에 연결한다.

10.1 개인 체인

이벤트 체인은 신원들에 의해 선택된 노드들 사이에서만 공유되는 개인 체인이다. 노드는 개인 체인이 노드에 속해 있지 않다는 것을 알지 못한다.

노드는 많은 이벤트 체인을 동시에 저장 및 가능하게 한다. 사이드 체인과는 달리, 사건 사슬은 완전히 격리된다. 체인은 서로 직접적으로 영향을 미치지 않는다. 이는 사건 사슬당 활동이 상당히 낮은 경우 수평선 이동 스케일링을 허용한다.

10.2 제네시스

누구든지 마음대로 새로운 이벤트 체인을 만들 수 있다. 이 체인의 제네시스 블록은 프로세스를 생성하는 사용자의 신원을 포함하며, 그 이후의 블록은 시나리오를 포함한다. 시나리오의 일부로, 다른 신원들을 이 개인 블록체인에 초대할 것이다.

11 컨센서스 메커니즘

LTO는 모든 당사자가 자신의 노드를 통해 참여할 수 있는 분산 시스템이다. 노드는 모든 이벤트를 피어에게 배포하고, 그 후 이러한 이벤트를 처리한다. 이는 노드 간의 프로세스 상태가 다른 짧은 순간이 있음을 의미한다. 이벤트 일관성[50]은 새로운 이벤트가 제출되지 않은 경우, 결국 모든 노드의 프로세스 상태가 동일함을 보장한다.

그러나 일관성을 달성하기 전에 새로운 이벤트가 제출되는 경우도 있다. 현재, 둘 이상의 노드가 이벤트를 이벤트 체인에 추가할 수 있다. Byzantine failure[51],는 모든 노드가 자신의 정보가 타당하다고 믿지만, 전체적인 시스템이 일관성이 없는 상태에 있다. 이 상태에서 노드는 더 이상 서로 새로운 이벤트를 수용하지 않는다; 노드는 멈추기보다는 합의에 도달할 수 있어야 한다.

분산형 애플리케이션은 이에 대해 다른 종류의 합의 알고리즘을 사용한다. 일반적으로, 이것은 Byzantine 내결합성의 사례다. 초기 BFT(Byzantine Fault Tolerance) 방법은 잘 확장되지 않는다 [52]. 작업 증명(PoW) [55], 스테이크 증명(PoS)[56] 및 승인 증명(PoA) [57]과 같은 더 나은 규모의 합의 도출의 발명은 많은 참여자가 참여하는 분산 네트워크를 만들 수 있도록 해주었으며, 이를 분산형 렛저 기술이라고도 한다.

이러한 합의 방법은 기존의 BFT 방식보다 훨씬 잘 확장되지만, (PoW ≥ 1000) 안전을 위해 비교적 많은 수의 참가자가 필요하다. 전통적인 BFT 방법들은 $(n-1)$ 명의 이하의 참가자들이 나쁜 행위자들이라는 사실에 의존한다[59]. 즉, 참가자가 4명 미만일 경우, 하나의 나쁜 행위자가 시스템에 영향을 미칠 수 있으며, 적어도 7명의 참가자가 두 명의 나쁜 행위자로부터 보호받아야 한다. 이벤트 체인은 비교적 적은 수의 참가자를 가진 개인 블록체인으로, 대개 7명 미만이며, 이는 그러한 알고리즘이 매우 취약하다는 것을 의미한다. 다수표를 신뢰하기보다는 노드는 달리 입증되지 않는 한 자신의 상태를 올바르게 간주한다.

11.1 충돌 가능성

이벤트 체인은 낙관적인 동시 통제에 의존한다; 많은 수의 충돌이 합의 알고리즘에 압박을 가할 것이며, 그것은 발생될 블록에서 기다려야 하기 때문에 상대적으로 느릴 수 있다.

분산 이벤트 체인은 다음과 같이 정의한다.

- Let N be the set of entities $\{n_1, n_2, n_3, \dots\}$ contributing to the event chain.
- Let C_n be the event chain, a sequence consisting of events (e_1, e_2, e_3, \dots) , belonging to entity n and let C be the set of all copies of the event chain $\{C_n | n \in N\}$.

- Let's define a conflict or branch as $\exists i, j \in N : i \neq j, C_{i_0} = C_{j_0}, C_i \not\subset C_j, C_j \not\subset C_i$.

우발적으로 충돌이 일어나려면, 두 당사자는 다른 체인으로부터 업데이트를 받기 전에 그들의 체인에 블록을 더해야 한다.

누군가 체인 $P(x)$ 에 업데이트를 전파할 기회를 잡아라. 이 기회는 주어진 시간 프레임 내에서 체인에 추가되는 블록의 수에 따라 달라진다. 블록을 네트워크의 나머지 부분에 전파하고, 이 기간 내에 체인에 기여하는 엔터티의 수. 이모든 사람이 네트워크에 동등하게 기여한다고 가정할 때, 이것은 공식 (1)에 따라 도출될 수 있다.

$$P(x) = \frac{f \cdot t}{n} \quad (1)$$

with:

f = Total amount of transactions / time frame

n = Total amount of active participants

t = Time it takes to propagate a block to the rest of the network

이 기회는 충돌이 일어날 확률을 계산하는데 사용될 수 있다. 이 확률은 1에서 충돌이 없을 가능성을 빼서 구한다. 1. 충돌이 없을 경우, 그 순간 아무도 체인에 기여하지 않았다는 것을 의미하며, 그 가능성은 공식 (2)를 사용하여 계산된다 (2),

$$(1 - P(x))^n \quad (2)$$

or only one node contributed to the chain, the chance of which is calculated in formula (3).

$$P(x) \cdot (1 - P(x))^{n-1} \cdot n. \quad (3)$$

Therefore the chance of a conflict is calculated by (4).

$$P(c) = 1 - (1 - P(x))^n - P(x) \cdot (1 - P(x))^{n-1} \cdot n. \quad (4)$$

With a network delay of 1200ms we see about chance on a conflict of < 2%:

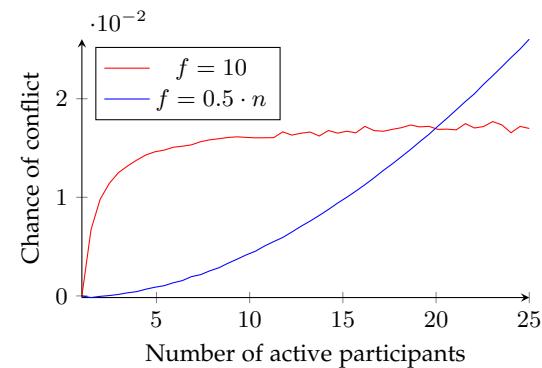


Fig. 5: This plot shows how the chance of a conflict occurring increases when the number of participants increases. However, it stabilizes if the total number of transactions stays the same.

Figure 5 shows that the chance of a conflict occurring is more or less constant when the number of participants increases but the total number of transactions stay the same. However, when the number of participants increases usually the number of transactions increase as well. For example, more participants may lead to more comments. When this is the case, the chance of a conflict increases exponentially with the number of participants.

LTO는 단일 이벤트 체인에 매우 적은 수의 활동적인 참가자들과 함께 일한다; 이것은 충돌의 가능성을 줄여준다. 5명 이상의 참가자가 있는 경우, 그 수는 더 이상 관련이 없다. 10명 이상의 참가자가 있을 경우, 충돌 가능성은 네트워크 지연과 거래 빈도수에 따라 다소 선형적이다.

만일 분리된 메시지 큐가 있는 이벤트 체인을 개별적으로 공유하지 않는다면, 거래 빈도는 매우 낮을 것이다. 이것은 충돌의 가능성을 좁혀준다.

11.2 분기 유효성 검사

한 노드는 상대방이 마지막 사건의 분기까지 체인에 대해 알고 있었다는 것을 증명할 수 있을 뿐이다. 어떤 당사자도 그들의 마지막 행동이 있은 후 그 체인을 분기할 수 있다. 만일 한 당사자가 그의 마지막 사건 전의 한 지점에서 그 체인을 분기하려고 하면, 그 지점은 모든 (다른) 당사자들에 의해 자동으로 폐기되고, 이벤트가 기록된다.

충돌하는 분기에서 새 이벤트를 수락하기 전에 그들은 수신된 이벤트 집합처럼 검증된다. 이벤트는 신원 중 하나에 의해 올바르게 서명되어야 하며 적절히 동기화되어야 한다. 이벤트의 타임스탬프가 동기화의 타임스탬프와 300초 이상 다를 경우, 이벤트를 거부될 수 있다.

11.3 우선순위

In case of a conflict, the following rules are applied in this order

- the priority of the event,
- the priority of the actor,
- the order of anchoring.

An action in the scenario may be given priority, this translates to the event priority. Additionally, some other event types like comments have a lower priority as the exact sequence is not of any importance. In case of a conflict occurs, the block that contains the action with the highest priority will be accepted.

In case the priority is the same, rules are applied based on identity, creating different levels of authority within the process. Every actor in the scenario may be given a priority level. By default, this priority is the same for all actors. To resolve the conflict, the events added by the actor with the highest priority must be accepted.

If both the events priorities and the identities authorities are the same, the third method of consensus is applied. 노드는 이벤트를 글로벌 블록체인에 동기화해야 한다. 채굴에 의해 설정된 블록 순서, 채굴 블록 내에서의 거래 순서도 고정된다. 이것은 우리가 세 계의 블록체인 동기화들의 순서를 이용하여 사건의 순서를 결정할 수 있게 해준다. 충돌이 발생할 경우 먼저 동기화된 블록을 승인해야 한다. 개인 이벤트 체인에 대한 합의는 공공 블록체인에 대한 합의를 통해 이루어진다. 공공 블록체인에서, 합의는 PoS에 대한 변형을 사용하는 다수의 참가자 사이의 익명의 협력에 의해 이루어진다.

행위 또는 행위자의 우선 순위를 매기는 것이 필요할 수 있으므로, 마지막으로 동기화되었더라도 먼저 순서를 정해야 한다. 시나리오에서 그러한 우선순위를 설정할 수 있다. 코멘트 같은 일부 이벤트 유형은 당연히 우선순위가 낮다. 우선순위를 사용하면 전방위 공격이 가능해지며, 이후에 해당 이벤트를 무효화하는 새 분기를 만들어 행위자가 이벤트에 응답할 수 있다. 우선 순위는 문제가 없는 경우에만 사용해야 한다.

11.4 동기화 되지 않은 사건

아직 동기화되어 있지 않은 블록이 수신되면 어쨌든 블록을 수락할 수 있다. 물론, 그것을 받아들임으로써 분쟁이 발생하지 않는다면 괜찮다. 블록 동기화가 단지 지연되었을 경우, 이를 수용하면 공정 자체에서 불필요한 지연을 방지할 수 있다. 반면에, 이 블록이 동기화되지 않으면, 진짜 문제가 발생하지 않는다. 모든 사람이 블록을 수용하면 정상적으로 공정을 진행할 수 있고 블록은 나중에 동기화될 수 있다.

11.5 분기 병합

포크가 발생할 때, 대부분의 블록체인 어플리케이션은 하나의 체인을 선택하여 계속 진행하며 다른 분기에서 볼 수 있는 모든 것을 무시한다. 비트코인과 같은 블록체인이 있으면, 모든 거래는 결국 각 지점의 채굴 블록에 포함될 것이다.

이벤트 체인에서 이벤트 자체는 해시 체인을 구축한다. 포크를 한 개 고르면 실행된 행위에 대한 정보가 손실될 수 있다. 대신에, 어떤 노드가 자신의 분기에 비해 우선하는 다른 분기에 대해 알게 될 때, 그것은 자신이 가지고 있는 사건들을 다른 체인의 상단에 로컬로 기초해야 한다. 이것은 git[53]을 사용할 때의 기준 조치와 유사하다.

11.6 포크

합의를 이를 수 있는 확실한 방법이 있을지라도, 참여 당사자는 다른 체인을 무시하고 포크를 제자리에 두기로 결정할 수 있다. 이더리움 [61]과 같은 대부분의 블록체인 애플리케이션의 경우, 이 값은 메인 체인에만 참여하는 것에서 기인하므로 이를 방해할 이유가 없다.

라이브 계약은 기존 프로세스를 디지털화하고 부분적으로 자동화하는 도구다. 비록 블록체인이 포크가 존재하도록 허용하지만, 그러한 과정은 대개 그렇지 않다. 포크의 경우, 당사자들은 분쟁을 수동으로 해결하기 위한 이차 절차를 시작할 수 있다.

12 프라이버시

LTO는 당사자 간의 프로세스를 운영하기 위해 구축되었다. 이러한 당사자들 외에, 아무도 그 과정이나 심지어 협업에 대해 알 필요가 없다.

공공 블록체인은 익명 계정을 허용하지만, 이러한 계정은 가명으로 기능한다. 어떤 거래라도 계좌의 신원을 밝혀 전체 거래 내역을 노출시킬 수 있다. 스마트 계약에는 모든 노드에서 사용할 수 있어야 하므로 데이터가 공개되어야 한다. 전소시엄 블록체인 경우 참가자들이 서로를 알고 있다.

Ad-hoc 개인 블록체인은 무작위적인 참여자가 승인이나 정보를 공개할 필요 없이 협업할 수 있도록 한다. 이 블록체인은 프로세스가 완료되면 완전히 지워질 수 있다.

12.1 링크된 데이터

각 당사자는 자체 노드 또는 자신이 신뢰하는 노드를 통해 연결한다. 각 노드에는 사용자가 데이터를 저장할 수 있는 개인 스토리지 서비스가 있다. 사용자는 Dropbox와 같은 서비스에 저장된 데이터와 유사하게 여기에 저장된 데이터에 대한 완전한 제어를 할 수 있다. 그들은 언제든 자신의 개인 정보를 삭제할 수 있다. 데이터는 유효한 데이터 처리 계약 및 사용자의 명시적 승인 없이는 공유되거나 간주되지 않는다.

어떤 작업으로 인해 데이터가 링크되면, 해당 데이터는 다른 사용자와 직접 공유되지 않으며; 블록체인에는 해시만 추가된다. LTO는 타임스탬프 및 일부 랜덤 데이터와 함께 봉투에 넣어 수신된 데이터를 확인하는 것 이외의 다른 작업에 해시가 사용되는 것을 방지한다. 봉투를 형성하기 위해 만들어진 해시는 블록체인에서 한 번 이상 발생하지 않을 것이다.

조직이 링크된 데이터가 필요한 행위를 수행하려고 한다고 표시할 때, 그 조직의 노드는 자동으로 요청을 할 것이다. 데이터 소유자의 노드는 지정된 동작이 유효한지, 따라서 실제로 현재 상태에서 이 행위자가 수행할 수 있는지 확인한다.

12.2 GDPR

유럽에 새로운 GDPR[54] (일반 데이터 보호 규정)이 도입됨에 따라, 많은 블록체인 애플리케이션이 이를 준수하지 못한다는 사실에 대한 논란이 제기되었다 [55]. 이에 대한 두 가지 주요 이유는 다음과 같다:

- 블록체인의 불변의 속성이 데이터를 수정 및 삭제할 수 있는 권리와 상충된다는 사실,

- 분산된 환경이기 때문에 전용 데이터 컨트롤러가 없다는 사실

링크된 데이터는 참여 당사자가 선택한 노드가 해당 사용자의 데이터 컨트롤러 역할을 한다는 것을 의미한다. 다른 모든 당사자는 항상 데이터 프로세서로 기능한다. 데이터 요청은 데이터를 처리하는 데 필요한 목적과 시간을 포함하는 적절한 데이터 처리 계약을 작성하기 위해 자동으로 포맷된다.

Ad-hoc 블록 체인은 필요한 경우 전체 체인을 삭제할 수 있다. 간단히 말해서, LegalThings의 개인 정보 보호 기능은 추가적인 노력 없이 솔루션 GDPR을 준수하도록 만든다.

12.3 영 지식 증명

시나리오는 한 당사자가 특정 가치를 알고 있음을 다른 당사자에게 입증하도록 요구할 수 있다. 영지식증명(zk-proof)은 그 가치를 알고 있다는 사실과는 별도로 정보를 전달하지 않고 그렇게 하는 방법이다.

LTO는 상호 작용적인 입증 시스템을 통해 zk-proof를 지원한다. 두 당사자, 즉, 입증자와 검증자는 그것의 정직성(완벽함)의 입증자를 납득시킬 수 있는 입증자와 부정직한 입증자[65]를 드러낼 수 있는 검증자를 목표가 지닌 메시지를 서로 교환한다.

라이브 계약을 위한 zk-proof는 항상 두 당사자 사이의 행동이다. 여전히 실험으로 간주되는 zkSNARKS와 같은 비인터랙티브 zk-proof는 필요하지 않다.

13 공통 패턴

13.1 체인 상호작용

예를 들어, 프로세스가 계속 진행하거나 충돌 해결 프로세스의 결과를 검색하기 위해 다른 프로세스의 권한이 필요한 경우, 일부 프로세스는 다른 프로세스와 상호작용해야 할 수 있다. 다른 프로세스에서 데이터 요청은 그림 6에 표시된 패턴을 따라 수행한다..

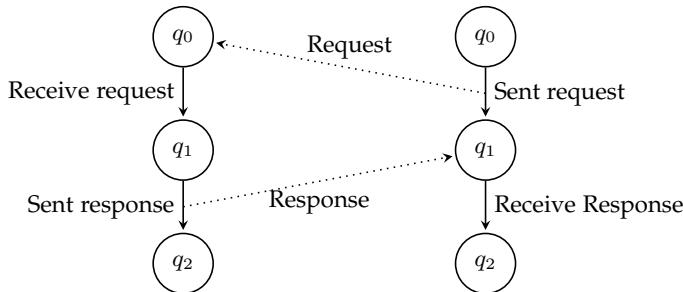


Fig. 6: Pattern followed when two processes interact.

13.2 명시적 동기화

어떤 경우에는 이벤트가 너무 늦게 전파되면 특히 골치 아플 수 있다. 이러한 경우 명확한 동기화가 시나리오에 내장될 수 있다. 이것은 모든 당사자들이 과정을 계속하기 전에 현재 상태를 인정하도록 요구한다. 이것은 FSM 내부에 있는 패턴으로, 당사자들이 이 이벤트에 앞서 일련의 사건을 처리하는 것을 막으려는 것이다. 분쟁이나 갈등이 있는 경우, 그러한 인지는 절차를 계속하기 위해 사용되어야 하는 분기를 결정하는 데 사용될 수 있다.

이러한 종류의 명시적 동기화는 모든 당사자가 현재 상태를 인정하는 경우에만 작동한다. 일부 당사자들이 계속 진행할 동기가 부족하거나 포크를 진행할 동기가 있는 경우, 다른 해결책이 필요하다. 이 경우 절차를 계속 진행하고자 하는 당사자는 다른 모든 당사들에게 자신의 의도를 발표한다. 만약 이 발표를 받은 당사자가 공표된 조치가 그들의 체인에서 유효하지 않다는 것을 알게 되면, 그들은 이를 전파할 수 있다. 이는 체인이 포크로 연결되었고 정기적인 충돌 해결이 적용되어야 한다는 것을 의미한다. 이 발표 중에 네트워크 자연의 영향을 제거하기 위해, 어느 누구도 발표된 이벤트를 거부하지 않았다고 가정하기 전에 정해진 시간을 기다린다.

Part II. 글로벌 블록체인

LTO 글로벌 블록체인은 신뢰가 필요 없는 공공 블록체인으로, 정보를 확인하기 위해 의도적으로 만들어졌다. 그것은 라이브 계약과 개인 이벤트 체인을 지원하기 위해 존재한다. 글로벌 블록체인은 이벤트 체인, 블록체인 및 애플리케이션에서 상호 운용할 수 있는 동기화 및 디지털 신원들을 특징으로 한다.

공중 거래는 거의 모든 블록체인에서 수행될 수 있다. 그러나 금융 거래나 일반 논리에 최적화된 블록체인 경우, 통상적인 유형의 거래는 비싸고 비효율적이다[56]. 또한, 제거 및 샤딩과 같은 최적화는 관련 정보가 생략되어 부정적인 영향을 미칠 수 있다[57, 58].

세계 블록체인은 Nxt 가문에 속한다[59]. 이 블록체인의 독특한 특성은 네트워크 노드의 부분에서 스크립트 처리나 트랜잭션 입력/출력 처리가 필요하지 않은 일련의 핵심 트랜잭션 유형을 기반으로 트랜잭션이 이루어졌다는 것이다. 이것은 블록체인의 크기를 줄이고 효율성을 높이며, 공중 거래에 특히 유리한 애그리게이션 방법을 허용한다.

Nxt로부터 직접 출발하기보다는 WAVES 플랫폼[70]의 포크를 기초로 사용한다. 이 플랫폼은 NG 프로토콜(제15.6 절)과 같은 많은 개선사항을 구현하였으며, 이는 우리의 네트워크가 이익을 얻을 것이다. 색 동전을 포함한 디지털 애셋에 초점을 맞춘 기존의 거래 유형은 제거되거나 비활성화되며 공중 거래 유형으로 대체된다.

14 (중앙 집중형 VS 분산형) 동기화

블록체인 동기화를 위한 다른 솔루션은 일정 기간 동안 모든 해시가 수집되는 중앙 집중식 접근방식을 사용한다. 이러한 거래에서, 맥클트리는 비트코인과 같은 제3자 블록체인에서 단일 거래로 만들어진다. 이에 따라 시스템으로부터 직접적인 피드백이 없으며, 중앙 서비스에서 최종 영수증을 수집하기까지 몇 시간이 걸릴 수 있다.

LTO 글로벌 블록체인은 모든 노드가 모든 트랜잭션을 관리하는 분산형 솔루션이다. 동기화 거래가 방송되면 즉시 볼 수 있으며, 약 3초 후 NG를 사용하여 사전 승인된다(제 15.6 절). 그 거래는 1분 안에 한 블록 안에 있다.

노드는 모든 동기화 해시 또는 그들 자신의 해시만을 추적할 수 있다. 필요한 경우 영수증을 독립적으로 작성할 수 있다(제 16.1 절). 중앙집중식 서비스는 필요 없다.

15 합의 알고리즘

글로벌 블록체인은 일반적인 공공 블록체인 역할을 한다. 노드는 트랜잭션의 유효성을 확인하고 블록을 만드는 생성기로써 선택된다. 생성기를 결정하기 위해, 우리는 Leased Proof of Importance(LPoI) 합의 알고리즘을 사용한다. 생성기는 위조된 블록에서 거래된 수수료로 보상받아야 한다.

Proof of Importance은 보유 중인 토큰 수와 스테이크 수에 따라 블록 제작을 위해 선택될 가능성이 있는 스테이크의 증명(PoS)의 변형이다. 중요도 증명서를 통해, 가능성은 노드의 네트워크 사용에 따라 증가한다[72, 73].

이 합의에서 비롯된 토큰 경제는 "LTO 토큰 경제" 논문[74]에 자세히 설명되어 있다.

15.1 리스

NXT, 웨이브 및 기타 블록체인 계열은 Leased Poof of Stake[59, 60]를 사용한다. 토큰을 임대함으로써 토큰 홀더는 선택한 노드에 블록을 생성할 수 있는 권리를 전달한다. 이러한 노드는 광업 풀처럼 작동할 수 있으며, 보상금을 리스자에 비례하여 공유할 수 있다.

NXT 스타일 네트워크는 현지에서 모든 활성 잔액의 최소 1/3을 제어할 때 공격에 취약하다[61]. LPoS 알고리즘을 구현한 프로젝트는 중앙 집중화 수준이 높은 경향이 있어 문제가 된다. 상위 두 개의 Nxt 노드는 네트워크의 50% 이상을 제어한다[62]. 웨이브에서, 상위 2개 노드는 네트워크의 1/3을, 상위 5개 노드가 50% 이상[63]을 제어한다.

18절에서는 이 네트워크에 대한 높은 분권화의 중요성에 대해 논의할 것이다. 대규모 임대 스테이크를 보유한 노드를 방지하기 위해 임대 토큰을 활용하는 데 제한이 있다. 모든 노드는 자신의 토큰 최소 10%를 스테이크해야 한다. Proof of Importance 또한 수동적인 수동적인 스테이커 노드를 불리한 위치에 놓기 때문에 이 효과를 반박한다.

15.2 라플 계수

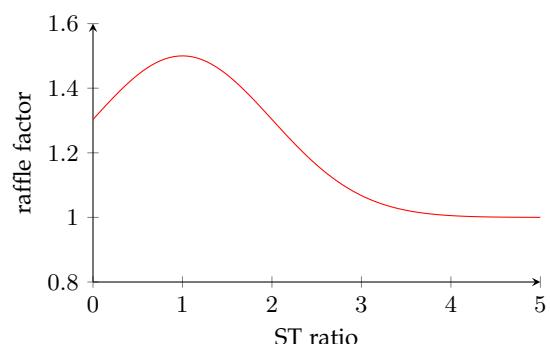
블록 생성 확률에 영향을 미치는 노드별 네트워크 사용량을 계산하면 고정 대 트랜잭션의 비율(S/T-ratio)을 사용한다.

$$\text{ST ratio} = \frac{\text{Staked tokens as \% of total}}{\text{Contributed transactions as \% of total}}$$

S/T-ratio는 "raffle factor"와 관련이 있다. 라플 인자는 새로운 블록을 생성하기 위해 노드가 선택될 가능성에 영향을 미치는 수학 공식이다. ST-비율(점수 1.0에 가까움)가 클수록 라플 계수가 더 높아진다. ST-비율이 불균형한 경우(노드는 어떤 거래도 기여하지 않는다), 라플 인자는 1.0이 된다.

$$\text{raffle factor } r = 1 + (0.5 \cdot e^{-0.5 \cdot (\text{ST ratio} - 1)^2})$$

이 공식은 종 모양의 표준 편차를 초래한다.

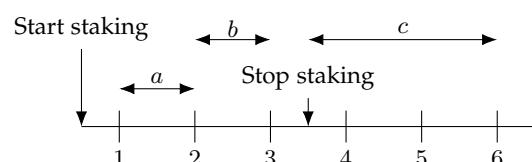


최대 라플 인자는 1.5로, 최소값과 절대값 2.0 사이의 반이다. 중요성을 부풀려서 X 토큰이 쌓이지 않도록 하려면 거래 수수료에 2 · X를 더 써야 한다. 중요도 인플레이션은 섹션 18.1에 자세히 설명되어 있다 18.1.

라플 인자의 개념과 그것이 토큰 경제에 미치는 영향을 완전히 이해하려면 "LTO 토큰 경제" 논문을 읽으세요[64].

15.3 Forge probability

생성될 가능성은 $P(\text{포지}) = S \cdot r^c$ 이다. 기여된 거래 T는 시간에 따라 계산된다. P(생성)를 계산할 때 S는 남용 가능성을 방지하기 위해 동일한 시간 동안 일정해야 한다.



$a = \text{Moment when } P(\text{forge}) \text{ is calculated}$

$b = \text{Moment when you forge a block}$

$c = \text{Moment when tokens are still locked}$

Fig. 7: 토큰 저장 및 블록 생성 시간. 시간은 요약 블록의 수로 측정된다..

15.4 공정한 PoS

새 블록을 만들 수 있는 노드를 결정하는 공식은 웨이브가 생성한 Fair Proof of Stake 알고리즘[65]을 기반으로 한다. 이것은 더 높은 스테이크의 가치를 초과하는 원래의 Nxt PoS 알고리즘의 개선이다.

기본 알고리즘에 대한 자세한 내용은 "Fair Proof of Stake" 논문[65] 년]을 참조하세요.

이 알고리즘을 PoS에서 POI로 변환하기 위해, 이 공식은 라플 계수를 고정 균형에 적용하여 $b_i \cdot r$ 으로 유효 균형을 맞춘다 $b_i \cdot r$.

- T_i as block generation time for i-th account,
- X_n is the generation signature,
- r raffle factor,
- b_i percentage of staked vs total staked,
- Λ_n is the base target
- $T_{min} = 5$ is a constant for delay between blocks,
- $C_1 = 70$, a constant defining shape of delay distribution,
- $C_2 = 5E17$ is a constant to adjust base target.

$$T_i = T_{min} + C_1 \cdot \log\left(1 - C_2 \cdot \frac{\log(X_n/X_{max})}{r \cdot b_i \cdot \Lambda_n}\right)$$

할당된 시간이 경과하기 전에 새 블록을 받은 경우, 이 블록을 체인에 추가하고 새로운 시간 지연을 계산해야 한다. 이전 T_i 는 더 이상 관련이 없다. 각 노드는 T_i 자체를 계산하며; 이 정보는 생성기에 의해 공급되지 않는다. 이것은 계산에 잘못된 스테이크를 사용할 의미가 없다는 것을 의미한다.

15.5 생성기 서명

시간 지연 T_i 를 결정할 때는 블록 해시를 고려하지 않는다 T_i . 이 해시는 블록을 위조하는 노드에 의해 결정되는 블록의 내용을 기반으로 한다. 만약 이 해시가 누가 다음 것을 생성할 수 있는지 결정하는데 어떤 역할을 했다면, T_i 를 조종하는 것은 쉬울 것이다 T_i . 노드는 몇 개의 다른 블록을 만들 수 있으며, 그 블록이 가장 짧은 블록만 방송할 수 있다.

이를 방지하기 위해 생성기 서명만 사용한다. 이 서명은 이전 세대 서명과 발전기의 공용 키만 사용하는 보조 해시 체인이다. Nxt를 사용하면 이것이 완전히 결정적이어서[76] 이용이 용이하다[61].

Fair PoS는 포크의 가능성을 줄이기 위해 100블록 전에 사용된 세대 서명을 사용한다. 노드나 리스에서 잔액의 변동은 주어진 X_n 으로 T_i 를 변경한다. 그럼에도 불구하고, 최소한 1/3의 토큰을 관리하는 그룹은 30%의 이점을 가지고 있다.

POI는 스테이크된 블록의 기간에 걸쳐 스테이크 된 잔고와 라플 인자를 일정하게 유지해야 한다. 이것은 그것을 그러한 공격에 훨씬 더 취약하게 만들 것이다.

해결책으로서, 생성은 공개적으로 계산될 수 있는 해시가 아니다. 대신, 노드는 이전 생성 서명을 해시하고 개인 키로 해시에 서명해야 한다. 이것은 생성 서명의 역할을 한다. Nxt와 Waves와는 반대로, 노드는 스스로 T_i 를 계산할 수 있기 때문에 생성기를 미리 결정할 수 없다.

15.6 NG 프로토콜

비트코인의 확장성 문제를 줄이기 위해 NG 프로토콜이 제안되었다. 반면 비트 코인에서 구현되지 않을 것으로 웨이브 NG는 웨이브 메인넷에 12월 2017년부터 활발하게 활동했다.

NG를 사용하면 마이크로 블록과 키 블록이라는 두 가지 유형의 블록이 생성된다. 이전에 선출된 노드는 트랜잭션의 유효성을 계속하여 평균 3초마다 마이크로 블록을 생성할 수 있다. 새 노드가 선출되면 아직 처리되지 않은 마이크로 블록에서 키 블록을 생성한다. 마이크로 블록에서의 거래는 동기화와 같은 저위험 트랜잭션의 경우 어느 정도 안전한 것으로 간주할 수 있다.

거래 수수료 보상은 마이크로 블록을 구축한 노드와 키 블록을 구축하는 노드 간에 40% - 60% 분할된다. 이 분할은 항상 키 블록 생성기에 유리해야 한다. 그렇지 않으면, 이미 만들어진 마이크로 블록을 무시하고 스스로 새로운 마이크로 블록을 만드는 동기가 있을 것이다. 공공 스트레스 테스트에서 웨이브 NG는 최대 6000 Tx/min

의 피크(피크 17,000 Tx/min[80])까지 처리할 수 있는 것으로 입증되었다. 잠재적으로 NG 프로토콜은 초당 최대 1000 Tx 또는 60,000 Tx/min을 처리할 수 있다.

NG는 실제 대기 시간을 줄이고 다른 최적화의 핵심 요소다.

16 개의 트랜잭션 유형

LTO 글로벌 블록체인은 미리 정의된 트랜잭션 유형을 사용한다. 이것은 더 작은 블록을 가능하게 하고 스크립팅의 필요성을 없애준다. 필요한 경우 거래 유형 목록을 향후 확장할 수 있다. 현재 가능한 거래의 유형은 다음과 같다:

- 동기화: 개인 블록체인에서의 트랜잭션을 확인하는 데 사용,
- 인증서 발급: 신원간의 관계를 선언하는 데 사용,
- 인증서 확장/재설행: 이러한 관계를 확장하거나 제거하는 데 사용,
- 토큰 전송: 토큰을 다른 신원으로 보내는 데 사용,
- 스테이크 보유 토큰: 참가자가 지분 또는 임대 토큰을 갖도록 하는 데 사용,
- 지분 보유 취소: 토큰 저장 또는 임대 중지 시 사용,
- 스크립트 트랜잭션 설정: 스마트 계정 구성 시 사용.

16.1 동기화

동기화는 문서 또는 기타 데이터의 해시를 가져와서 이를 블록체인 트랜잭션 내에 저장하는 방법이다. 여기서의 목표는 작성자를 포함한 모든 사람이 자신의 문서를 소급하거나 회송하는 것을 불가능하게 하는 것이다[66].

개인 이벤트 체인의 모든 이벤트는 글로벌 블록체인에 동기화되어 있다. 3자 애플리케이션은 증거 문서를 동기화하기 위해 글로벌 체인을 사용할 수 있다. 우리는 글로벌 체인의 모든 거래 중 99%가 거래를 동기화시키고 있을 것으로 추정한다. 대부분의 트랜잭션이 동기화에 대한 것임을 고려할 때 이를 집계하면 블록체인이 요구하는 디스크 공간이 줄어든다.

블록을 생성할 때, 노드는 목록에 표시된 순서대로 트랜잭션에 서[67]를 생성한다. 블록체인에는 Merkle 루트만 추가된다. 검증 프로세스의 일부로 각 노드가 이 Merkle 트리를 다시 생성한다.

노드는 모든 동기화 해시를 색인화할 수 있다. 그러나 디스크 크기를 줄이려면 대부분의 노드가 자체 동기화 트랜잭션의 Merkle 경로를 추출하는 것을 선택해야 한다. 이 경로는 이벤트와 같이 원본 데이터로 저장할 수 있는 영수증을 형성한다.

16.2 인증 및 승인

사용자 이름 및 암호 확인과 같은 이의/응답 인증 방법에는 중앙 집중식 시스템이 필요하다. 완전한 분권형 시스템에서, 우리는 인증을 제공하는 암호화 서명에 의존한다. 사건 이벤트에서 정보는 공유되지 않지만, 서명에 사용된 키 쌍을 고려할 때, 당사자들은 여전히 체인에 걸쳐 식별될 수 있다.

더 넓은 의미에서, 당사자들은 이런 식으로 어떤 종류의 정보도 서명할 수 있다. 이것은 PKI 인증서에 의해 현재 제시된 것과 유사한 사용 사례다. 이 인증서를 발급하고 해지하기 위해 중앙 당국에 의존하는 것은 이의/응답 인증의 대체물로 채택하는 것을 방해했다.

공공 블록 체인을 사용하면 중앙 당국의 필요 없이 공용/개인 키 쌍을 생성 및 사용할 수 있다. 키 쌍은 고유한 신원을 형성하며, 공용 키의 해시로부터 파생된 주소를 통해 참조될 수 있다.

16.3 인증서

인증서 거래는 모든 신원이 그것의 주소를 참조함으로써 다른 신원에 대한 정보를 전달할 수 있도록 한다. 토큰과는 달리, 계좌의 허가와 폐기는 전적으로 발행자의 통제하에 있다.

인증서는 인증서를 발급하는 당사자가 선택하는 특정 형식을 지정할 수 있다. 필수는 아니지만 인증서를 다른 사람에게 표시하기 전에 수취인 계정으로 승인하는 것이 좋다.

16.4 신뢰 사슬

개인 키 쌍이 있는 공용 주소는 인증 방법이지만, 인증 솔루션은 제공하지 않는다. 인증서를 사용하여 신원간의 관계를 지정할 수 있다.

이 접근법은 신뢰 웹(WoT)과 유사하다. WoT는 우리 플랫폼에 없는 PKI에 대해 많은 단점이 있다.

블록체인에서 관계를 수립하고 파기하거나 신원을 훼손된 것으로 표시하는 것은 간단하고 즉각적이며 취소할 수 없다. 블록체인 거래는 특정 시점에서 관계의 존재를 확인할 수 있는 타임스탬프된다.

우리는 단순히 신원을 확립하기 보다는 특정한 관계를 수립한다. 그 거래는 다른 상대방을 물리적으로 만날 필요를 없앰으로써, 관계의 존재를 제외한, 신원에 대한 어떤 다른 정보도 확인하거나 부인하지 않는다.

주어진 상황에서, 우리는 오직 이 관계를 기초로 한 두 정체성 사이의 신뢰 사슬을 찾는 것에만 관심이 있다. 이것은 PKI 유효성 검사는 했지만 중앙 권한은 없는 신뢰 체인을 흉내낸다. 절대적 루트 인증서라기보다는, 우리 조직이나 우리가 거래하는 조직의 블록체인 주소가 신뢰 루트로 기능한다.

16.5 스마트 계정

기본적으로, 계정에 대한 모든 거래는 해당 계정과 연결된 개인 키를 사용하여 서명되어야 한다. 웨이브는 누구나 이 논리를 사용자 정의할 수 있도록 스마트 계정의 개념을 도입했다[68].

그렇게 하기 위해, 이 논리는 non-Turing complete 언어를 사용하여 스크립트될 수 있다. 이 스크립트는 특정 계정에 대한 트랜잭션을 확인하거나 거부하는 데만 사용된다. 그것은 다른 거래를 축발시킬 수 없다. 따라서 이러한 유형의 스마트 계약은 애그리게이션을 방해하지 않는다. 이를 보장하기 위해 LTO 스마트 계정에는 추가 제한이 있다.

LTO에는 데이터 트랜잭션이 없으며 스크립트에서 다른 트랜잭션에 액세스할 수 없다. 스마트 계정은 거래 서명을 위해 사용해야 하는 대체 공용 키를 지정하여 다중 서명 계정을 만드는 데 사용할 수 있다. 이 키를 직접 지정하기 보다는 특정 인증서를 보유한 사람이 거래에 서명할 수 있음을 명시한다.

다른 한계들도 고려될 수 있다. 계좌는 잡길 수 있고, 단지 몇 개의 블록 후에만 토큰의 전송을 허용하거나, 최소 수의 토큰을 계정에 남아있도록 요구하거나, 특정 계좌로 토큰을 전송할 수 있다.

노드를 실행할 때는 다중 서명 사용을 권장한다. 노드와 관련된 계정은 일반적으로 많은 수의 토큰을 보유하여 스테이크와 동기화 시킨다. 이 계정의 개인 키는 노드에 알려져 있다. 다중 서명으로, 그 키를 얻는 것은 그 토큰에 직접 접근하지 못할 것이다. 거래 동기화는 스마트 계정의 영향을 받지 않는다. 그들은 항상 계좌의 개인 열쇠로 서명할 필요가 있다. 이 논리는 글로벌 블록체인 노드의 수평 스케일링과 같은 미래의 최적화를 가능하게 한다.

17 요약 블록

동기화는 블록체인에 추가적인 보안 층을 가져오기 위한 저충격, 비파괴적 방법이다. 우리는 라이브 계약을 사용하지 않는 다른 애플리케이션들도 동기화 기능을 사용할 것으로 예상한다.

블록체인의 확장성에 반하는 한 가지 측면은 블록체인이 계속 증가할 것이라는 사실이다[69]. 크기는 사본을 보관하기 위해 하드웨어에 특정 요구 사항을 부과한다. 그것은 또한 전체 체인을 되돌려야 하는 새로운 노드에 부담을 준다. 체인의 증가 속도를 줄이기 위해 요약 블록을 사용한다.

17.1 키 블록 크기

표 3은 블록체인 키 블록의 구조를 보여준다. 글로벌 체인은 하루에 최대 5천만 번의 트랜잭션을 확장할 수 있어야 한다. 이것은 예상 사용량의 약 5배이다. 이러한 키 블록의 크기는 블록 데이터와 트랜잭션 데이터(5)에 의해 결정된다 (5).

$$\text{Keyblock size} = d + t \quad (5)$$

with:

d = block data,
 t = transaction data.

예상 블록 크기를 계산하기 전에 다음과 같은 가정을 한다:

- 99.98% of the transactions are anchoring transactions (table 5). This is the main use of the global blockchain,
- The other 0.02% are issue certificate transactions (table 7),
- All other transactions are infrequent and can be neglected,
- All transactions are uniformly distributed over the blocks,
- On average one key block per minute is generated,
- Every day 1440 key blocks are created.

The block data is 277 bytes big (table 3). Under the assumptions made previously, the size of the transaction data can be calculated by equation (6).

$$\text{Transaction data size} = n \cdot (0.9998 \cdot a + 0.0002 \cdot c) \quad (6)$$

with:

a = Size of an anchor transaction (table 5),
 c = Size of an issue certificate transaction (table 7),
 n = Amount of transaction per block.

This makes the total size of a key block about 3.8MB.

17.2 애그리게이션 없는 성장

3.8MB 블록당 그리고 일일 1440블록, 블록체인은 지속적으로 최대 용량으로 실행된다면 하루에 5.47GB/2TB씩 증가하게 될 것이다.

예상 사용량은 약 1,000만 건의 거래로 블록체인 1.1 건을 증가시키고 있다. 이는 36.5억 건의 거래 또는 연간 약 400GB의 거래 결과를 가져온다.

총 3억4000만 건의 거래[85]를 보유한 비트코인의 경우 네트워크와 하드웨어 속도에 따라 제네시스와 동기화하는 데 약 7일이 소요된다. 수십억 개의 거래에서, 순진하게 이렇게 하는 것은 글로벌 블록체인이 동기화되기를 몇 주 또는 심지어 몇 달 동안 기다리는 것을 의미할 수 있다.

트랜잭션을 집계하는 목표 중 하나는 다시 네트워크 및 하드웨어 속도에 따라 연간 20분만 동기화하면 되는 것이다. 일년에 365 개의 요약 블록이 있으면, 노드는 3초 안에 요약 블록을 처리할 수 있어야 한다.

17.3 분리 증인

분리된 증인은 비트코인에서 처리해야 할 데이터와 거래를 감소하는 데 사용되는 데이터로 구분해 블록[86]의 데이터를 줄이기 위해 채택된 전략이다[70]. 이 두 번째 부분은 다른 것들의 서명을 포함하는 증인 데이터라고 불린다.

최종성은 충분히 깊은 블록이 절대로 블록체인에서 제거되지 않는다는 것을 보장한다. 확률론적 최종성 또는 프로토콜 최종성 여부에 관계 없이, 블록을 되돌리지 않을 경우 증인 데이터는 더 이상 유용하지 않다. 노드는 최종 단계에 도달한 블록에 대한 감시 데이터를 자유롭게 제거하여 디스크 공간을 절약한다.

우리는 분리된 증인의 논리를 바탕으로 요약 블록의 개념을 도입했다.

17.4 애그리게이션

이것들은 1440블록마다 만들어진 특별한 블록이다. 이 값은 이전 요약 블록 이후 모든 블록의 집합된 값을 포함한다. 체인을 재생할 때는 현재 상태에 접근하기 위해 요약 블록만 적용해야 한다. 그런 다음 마지막 요약 블록 이후에 생성된 블록만 재생해야 한다. 이는 재생 시간을 크게 감소시킨다.

두 번째에서 마지막까지 요약 블록과 그 이전까지의 모든 블록이 최종이다. 노드는 가장 긴 체인에 관계 없이 그 지점 이전에 포크를 고려하지 않을 것이다. 이는 이전의 1440 - 2880개 키 블록의 전송 동작만 저장해야 한다는 것을 의미한다. 요약 블록에 저장된 후 키 블록에서 트랜잭션 데이터를 제거하는 것은, 키 블록 크기가 11.3에서 MB - 277 바이트로 감소함에 따라, 요약 블록에 비해 무시할 수 있다.

17.5 축소의 차이

언뜻 보기에도 이 접근법은 제한된 일련의 거래만 유지하므로 블록체인 축소 방식과 유사하다. 축소시 위험은 위조된 상태가 도입되면 불변의 본성을 위협한다는 것이다.

그 위험은 블록체인 상태를 분배하는 데서 온다. 분리된 중인과 함께, 거래는 최종성의 개념에 의존하여 서명 검증 없이 적용된다. 그러나, 모든 노드는 현재 상태를 계산하기 위해 여전히 제네시스의 모든 거래를 적용할 필요가 있다.

실제 거래 데이터는 블록 서명을 계산하는 데 사용되지 않고 오히려 블록 옆에 부착물로 저장된다(표 2). 트랜잭션이 없는 이벤트로서, 키 블록은 블록체인의 일부분이므로 무시할 수 없다. 유효성 검사 없이 트랜잭션을 적용하는 경우, 그들을 통합하는 것은 거의 위험을 주지 않는다.

17.6 요약 블록 크기

요약 블록은 비-동기화 트랜잭션에 대한 모든 정보와 모든 트랜잭션 수수료 및 기타 토큰 전송의 집합 버전을 포함한다. 그것들은 특히 키 블록과 비교했을 때 비교적 큰 블록이다. 사용된 메모리 양을 줄이기 위해, 거래 수수료와 토큰 전송 트랜잭션은 참가자당 잔액 변경으로 감소한다(표 4)). 요약에는 또한 인증서, 저장 및 스크립트 트랜잭션과 같은 집계할 수 없는 트랜잭션도 포함되어 있다. 요약 블록의 예상 크기를 계산하기 위해 다음과 같은 가정을 한다:

- 총 200,000명의 참가자가 있다,
- 매일 하나의 요약 블록이 생성된다,
- 이전 셕션의 가정을 바탕으로 우리는 균형(상태) 변경 요약이 요약 블록의 유일한 유의한 부분이라고 가정할 수 있다.

이러한 가정을 이용하여 요약 블록의 크기를 계산할 수 있다. 계산식 7, (7)을 사용하여 크기를 계산할 때 결과는 약 10.3이다.MB.

$$\text{Summary block size} = \text{Transaction summary} = p \cdot e \quad (7)$$

with:

p = Amount of participants in the last 1500 blocks,

e = Size of a balance change summary entry (table 4).

17.7 총 크기

블록체인의 총 크기는 정적 부분과 증가되는 부분으로 구성된다. 정적 부분은 마지막 1000개의 키 블록으로 구성되어 있다. 그것들은 여전히 첨부자료(5)를 포함할 것이다.

$$\text{Static part} = n \cdot k \quad (8)$$

with:

n = 트랜잭션 데이터와 함께 저장된 키 블록 양,

k = 키 블록 크기 (5).

등식 8을 확률의 크기를 계산하는 데 사용할 경우, 총 크기가 약 11.3GB임을 나타낸다. 8. 트랜잭션을 포함하여 마지막 1000개의 블록만 완전히 저장되므로, 이 크기는 약간 다를 수 있지만 눈에 띄게 증가하지는 않는다.

증가되는 부분은 요약블록 (7)과 거래데이터를 제거한 후 핵심 블록의 남아 있는 것으로 구성된다. 우리는 그 크기를 방정식 (9)을 사용하여 연간 성장으로 정의할 것이다 (9).

$$\Gamma_s = n \cdot k + m \cdot s \quad (9)$$

with:

n = 연간 주요 블록의 양,

m = 연간 요약 블록의 양,

k = 트랜잭션 데이터가 없는 키 블록의 크기,

s = 요약 블록의 크기 (7).

이전에 이루어진 가정에 따라, 이 방정식은 블록체인이 연간 약 3.7GB씩 증가한다는 것을 보여준다.

17.8 기록 노드

오래된 거래를 삭제하기 위해 노드가 필요하지 않다. 모든 트랜잭션을 주도함으로써, 기록 노드는 필요할 때 블록체인의 정확성을 증명할 수 있다. 기록 노드의 블록이 다른 노드의 블록과 일치해야 하므로 기록 노드는 기록을 통과할 수 없다. 네트워크는 상대적으로 적은 수의 기록 노드에 의존할 수 있다.

기록 노드를 온체인의 실질적인 이득이 없다. 그것은 새로운 블록을 생성할 가능성을 증가시키지 않는다. 그러한 노드를 운영하는 것은 지역사회 이익이나 부수적인 소득에서 벗어나야 한다.

18 네트워크 취약성

18.1 중요도 인플레이션

PoI에 대한 특별한 걱정은 중요성의 인플레이션이다. 더 큰 거래를 통해서, 스팸 거래로 인한 이익/손실을 아래에 제시된 최대 라플 인자의 공식으로 계산할 수 있다.

- 라플 계수; r ,
- 저장된 토큰의 백분율; b_i ,
- 거래 비용; c ,
- 네트워크상의 총 트랜잭션; n ,
- 스팸 트랜잭션; τ ,
- 보상; p ,
- 스팸으로 인한 이익/손실; $\Delta p = p_{r_{max}} - p_{r=1}$.

$$p = (r \cdot b_i \cdot n \cdot c) - (\tau \cdot c) \quad (10)$$

$$r = 1, \tau = 0 \rightarrow p = b_i \cdot n \cdot c \quad (11)$$

$$r = r_{max}, \tau = b_i \cdot n \rightarrow (r_{max} - 1) \cdot b_i \cdot n \cdot c \quad (12)$$

This gives

$$\Delta p = ((r_{max} - 2) \cdot b_i \cdot n \cdot c) \quad (13)$$

Given

$$b_i > 0, n > 0, c > 0, \Delta p < 0 \rightarrow (r_{max} - 2) < 0 \quad (14)$$

$$r_{max} < 2 \quad (15)$$

공식 15는 최대 라플 인수가 2 미만인 스팸 거래에서 직접 얻는 것이 불가능하다는 것을 입증한다

두 개에 가까운 라플 인자는 스팸 거래를 거의 무료로 할 수 있다. 51%의 공격으로 네트워크를 손상시키려고 하는 공격자를 도울 수 있기 때문에, 적은 비용으로 네트워크의 중요성을 증가시키는 것은 바람직하지 않다. 최대 라플 인수는 1.5로 부풀리는 중요성의 높은 비용을 보장한다.

18.2 Nothing at stake

Nothing at stake 원칙은 노드가 가장 긴 체인을 선택하는 것이 아니라 포크를 계속해서 구축할 것이라는 가정이다, 왜냐하면 그렇게 하는 데는 단점이 없기 때문이다[87]. 만약 모든 노드가 그러한 잘못된 행동을 보인다면, 공격자는 네트워크가 다른 체인으로 전환되도록 하기 위해 적은 비율의 토큰, 즉 1%의 공격만 필요로 할 것이다.

그러한 상황은 Tragedy of The commons[88]이라고 일컬어진다. 모든 구성원들은 그 제도를 악용하여 개별적으로 이익을 얻으려고 한다. 하지만 모두가 그렇게 한다면 아무도 혜택을 받지 못할 것이다. 대신에, 그것은 네트워크를 손상시켜, 기본 토큰의 가치를 떨어뜨리게 할 뿐이다.

Waves Fair PoS는 나오지 않은 분기가 주요 분기를 따라잡는 것을 훨씬 더 어렵게 만들었다. 적은 비율의 토큰을 가지고 있는 나쁜 행위자들과 함께 그러한 행동을 통해 이익을 얻을 가능성은 무시해도 좋다.

나쁜 행위자는 노드의 변경 버전을 만들고 유지해야 할 것이다. 이로 인해 이득을 볼 가능성이 거의 없다는 인식과 함께, 그 비용은 행동을 약화시키기에 충분해야 한다[71].

18.3 LPoS 중앙 집중화

LPoS 알고리즘을 구현한 프로젝트는 중앙 집중화 수준이 높은 경향이 있다.

이 효과는 토큰당 보상으로 설명할 수 있다. 가동 시간이 거의 100%에 가까운 전문 설정은 정해진 수의 토큰을 보관함으로써 더 많은 보상을 얻어내므로 생성 기회를 놓치지 않는다. 이는 토큰 보유자를 끌어들여 이러한 노드로 임대하고, 오버헤드를 줄이면 리스 제공자에게 더 많은 금액을 지급하는 것을 허용하기 때문에 보강 효과를 갖는다.

노드당 토큰 수를 제한하는 것은 Sybil 공격의 기회를 만드는 결합 있는 방법이다. 허가가 적은 평판 시스템에서는 단일 노드가 다중 익명 신원으로 자신을 광고할 수 있으므로 이러한 제한을 회피할 수 있다.

솔루션의 특성 때문에, 대부분의 전송 조치는 노드와 연결된 키 쌍에 의해 서명될 것이다. 네트워크는 또한 상대적으로 많은 수의 노드로 구성될 것이다. 이것은 PoS에는 영향을 미치지 않지만, PoI에서는 플랫폼 사용자가 비사용자 토큰 보유자에 비해 유리하도록 한다.

추가 조치는 임대한 토큰의 활용에 대한 제한이다; 모든 노드는 자신이 보유한 토큰의 최소 10%를 소유할 필요가 있다.

18.4 서비스 거부 공격

확장성이 제한되기 때문에 공공 블록체인에는 너무 많은 트랜잭션을 처리하는 것이 매우 쉽다. 거래 수수료는 이러한 공격에 대한 일차적인 방어책이지만, 자금이 부족하다는 결론을 내리기 위해서는 여전히 거래가 검증되어야 한다. 더욱이, 2018년 7월에 이더리움에서 보듯이, 상당한 액수의 자금으로 인해, 네트워크에 스팸 거래를 과도하게 할 수 있다.

노드는 그러한 부착의 경우에 전송 수수료를 자동으로 인상할 수 있는 옵션을 가지고 있다. 이상적으로, 노드는 거래에 소비하는 만큼 스테이킹함으로써 많은 이득을 얻는다. 스팸 토큰이 거래 수수료의 보상을 늘림에 따라, 이것은 자동적으로 공격에 대항하기 위해 사용된다.

18.5 SHA-2 취약성

SHA-1은 지난해 구글 연구소가 서로 다른 두 문서가 같은 해시(해시)를 발생시킨 충돌사고를 발견했을 때 취약하다는 것이 입증됐다. SHA-2 256비트가 유사하게 취약한 경우, Merkle Tree에 기초한 동기화에 치명적일 수 있다.

충돌이 발견되면 충돌 문서가 공증되었다고 주장할 수 있다. 더 나쁜 것은, Merkle root와 무작위 해시를 고려할 때, 유효한 Merkle path를 생성할 수 있다는 것이다. 이것은 해커가 어떤 문서를 검증할 수 있도록 할 것이다. 그러나, 이것은 여전히 쉬운 일이 아닐 것이다,

왜냐하면 트리의 모든 분기에 대해 정확히 32바이트 길이의 해시 두 개를 결합해야 하기 때문이다.

특정 SHA-1을 잔혹하게 만드는 것은 여전히 일생 동안 달성하기에는 너무 많은 계산이 필요할 수 있지만, Birthday 역설은 충돌을 찾기 위해 요구되는 계산이 훨씬 더 적어지게 한다. Birthday 패러독스는 또한 많은 Merkle root가 있고, 각각의 뿌리는 최대 Merkle path를 가지고 있기 때문에 LTO 공공 체인에도 적용된다.

이를 극복하기 위해 검증이 될 경우 검증은 기록 노드에 다시 귀속될 수 있다. 그러나 이는 동기화 노드의 전반적인 사용을 감소시킨다.

대신에 SHA-2 해시가 SHA-3 또는 Blake2와 같은 다른 알고리즘으로 해시되는 경우 보조 Merkle 트리를 추가할 수 있다. 단지 이중 해싱은 Merkle을 변조할 수 있을 때 유용하지 않다, 왜냐하면 외부 알고리즘이 (부당하게)활용 시 약점만 필요하기 때문이다. 그러나, 두 개의 Merkle 트리가 있다면, 두 개의 알고리즘이 모두 깨질 필요가 있을 것이다. 하지만, 그 때조차도, Birthday 역설의 이점을 없애면서, 한 블록의 나무 둘 다에서 충돌이 발견되어야 한다.

Part III. 플랫폼

19 설계

19.1 마이크로 설계

LTO 노드는 마이크로서비스 구조 패턴을 사용하여 개발된다. 이는 노드의 모든 기능이 마이크로 디바이스로 분할되고, 각 서비스가 전체 노드의 작은 부분만 담당한다는 것을 의미한다. 이 패턴에는 다음과 같은 몇 가지 장점이 있다.

- 실패 격리: 서비스가 실패할 경우, 반드시 다른 서비스에 간섭할 필요는 없다
- 확장성: 노드 내의 모든 서비스가 서로 다른 시스템에서 결합되어 실행될 수 있다. 이것은 수평 스케일링에 매우 적합하다. 24절에서 설명한 대로 스케일링이 자동화된다.
- 유연성: 특정 기능은 특정한 프로그래밍 언어에서 더 잘 이용된다. 각 서비스는 다른 프로그래밍 언어로 개발될 수 있다.
- 코드 품질: 노드를 작고 잘 정의된 모듈로 분할하여 개발자가 보다 쉽게 읽고 검토할 수 있다. 이것은 코드 품질을 향상시킨다.

마이크로소스는 도커 컨테이너에 분류되어 있다. 이 모든 컨테이너는 Kubernetes 컨테이너 또는 시험 플랫폼을 사용하여 운영된다; 이는 섹션 24에 설명될 것이다. 각 마이크로서비스는 독립적으로 실행되도록 설계되었다. 즉, 공유 종속성이 없으므로, 각 컨테이너에 자체 데이터베이스 또는 이벤트 대기열이 있는 것이다. 마이크로서비스는 또한 쉽게 확장될 수 있도록 상태 없이 작동하도록 설계되었다.

19.2 애플리케이션 계층 및 서비스

19.1절에서 설명한 것처럼, 노드는 여러 서비스로 분할된다. 이 서비스들은 4개의 서로 다른 계층으로 분류된다. 노드는 다음 계층으로 구성된다:

- UI 계층: 이것은 애플리케이션 계층과 상호 작용하는 UI 애플리케이션으로 구성된다,
- 애플리케이션 계층: 이것은 이벤트 체인의 이벤트에 의해 작동된 작업을 처리하는 모든 서비스를 포함한다,
- 개인 체인 계층: 이것은 노드의 디커플링을 처리한다,
- 공용 체인 계층: 이것은 공용 체인 서비스를 관리한다. 우리의 글로벌 공공 블록체인은 해시 저장에 최적화되어 있다. 우리의 글로벌 공공 블록체인은 해시 처리를 위해 최적화되었다.

20 UI 계층

UI 계층에는 사용자가 라이브 계약을 쉽게 개발하고 디버그할 수 있도록 해주는 두 가지 프런트엔드 애플리케이션이 포함되어 있다. 첫째, 사용자가 특정 노드에 접속하여 모든 체인을 나열할 수 있는 체인 뷰이다. 사용자는 자신이 속한 사슬만 나열하고 볼 수 있다. 두 번째 애플리케이션은 사용자가 실시간 계약 시나리오를 개발할 수 있는 플레이그라운드의 응용 프로그램이다. 이것은 상황을 상태 다이어그램으로 시각화하고 다른 시각화 및 검증 도구를 포함한다.

21 애플리케이션 계층

21.1 웹 서버

웹 서버 애플리케이션은 프런트 엔드와 노드 내의 애플리케이션 사이의 프록시 역할을 한다. 웹 서버는 다음과 같은 두 가지 기능을 수행한다:

- 서비스에 대한 모든 요청 인증,
- 모든 요청을 올바른 서비스로 전달.

21.2 워크플로 엔진

라이브 계약의 실제 생성 및 실행은 워크플로우 서비스에 의해 이루어진다. 이벤트 체인 서비스에 의해 수신된 이벤트가 라이브 계약의 작업을 포함하는 경우, 그것은 워크플로우 서비스로 보내질 것이다. 그런 다음, 워크플로우 서비스는 워크플로우의 상태 전환과 워크플로우의 새로운 투영으로 이어지는 작업을 실행한다. 이 투영법은 MongoDB 데이터베이스에 저장된다.

22 개인 체인 계층

개인 체인 계층은 노드를 분리한다. 디커플링은 연결이 불량하거나 부하가 높은 경우에도 안정된 시스템을 보장한다. 메시지 큐는 개인 체인의 통신 계층이다. 메시지 큐를 제공하는 기술은 RabbitMQ로, 다른 노드뿐만 아니라 노드 내에서 메시지를 전달하기에 완벽하게 적합한 경량 메시지 브로커다. RabbitMQ는 다른 RabbitMQ 브로커와 동적으로 연결을 설정하고 메시지를 교환하는 Shovel이라는 기능을 가지고 있다. 이 메커니즘은 한 노드에서 다른 노드로 이벤트를 전송하는 데 사용된다. 세 가지 서비스가 모든 인바운드 및 아웃바운드 이벤트를 관리한다.

22.1 이벤트 체인 서비스

개인 체인을 관리하는 서비스는 이벤트 체인 서비스다. 이 서비스는 들어오는 모든 이벤트를 처리한다. 다음 단계에 따라 이벤트를 처리하십시오:

- 올바르게 서명되었는지, 체인이 파손되지 않았는지 확인하여 들어오는 이벤트를 검증한다.
- 체인이 로컬로 저장된 체인과 일치하는지 확인. 그렇지 않다면, 가능한 한 분쟁 해결을 수행한다.
- 만약 이벤트가 이 노드에 속하는 신원으로 전송되면, 수신된 이벤트를 실행한다. 그렇지 않으면, 그것은 그것들을 데이터베이스에 저장만 할 것이다.
- 다른 노드에서 새로운 신원을 추가하면 전체 체인이 이 노드로 전달됨.
- 모든 새로운 이벤트는 관련 노드로 전달된다.

보관을 위해 이벤트 체인 서비스는 MongoDB 데이터베이스를 사용한다.

22.2 이벤트 인큐 서비스

이벤트 인큐 서비스에는 이벤트를 이벤트 대기열에 배치하는 작은 작업이 할당된다. 이는 노드 내 서비스(예: 워크플로우 서비스 및 이벤트 체인 서비스)와 외부 사용자에 대해 모두 적용된다.

22.3 이벤트 발송 서비스

이벤트 대기열의 모든 메시지는 이벤트 발송 서비스에서 처리한다. 그것은 모든 메시지를 수집하여 그것을 이벤트 서비스에 배포한다. 이벤트 서비스가 메시지를 처리하면 처리한 것으로 표시하고; 그렇지 않으면, 그것은 데드-레터 대기열로 이동한다.

23 공공 체인 계층

23.1 동기화 서비스

동기화 서비스는 공공 체인의 중심에 있다. 동기화 서비스는 NG 프로토콜로 확장된 NXT 플랫폼의 포크가 될 것이다. “정상” 거래와 데이터 거래를 모두 처리할 수 있도록 동기화 서비스가 확대될 것이다. 이러한 데이터 트랜잭션은 16.1절에서 설명한 것처럼 개인 체인에서 발생한 이벤트 해시를 저장하는 데 사용된다. 이 모든 해시는 매일 수집되어 결정적으로 Merkle 트리로 통합될 것이다. 이렇게 하면 데이터 트랜잭션을 스토리지에서 제거하여 스토리지 설치 공간을 줄일 수 있지만 사람들은 여전히 해시가 존재하는지 확인할 수 있다.

특정 해시가 저장되었는지 확인할 수 있도록 모든 해시가 인덱스 처리된다. 이는 모든 데이터 트랜잭션을 검색할 필요가 없기 때문에 검증을 훨씬 더 빠르게 한다.

24 컨테이너 조정

노드는 여러 마이크로 서비스로 구성되므로, 컨테이너의 실행을 관리하기 위해 컨테이너 조정 플랫폼이 필요하다. 컨테이너 조정 플랫폼은 호스트 공급, 컨테이너 인스턴스화, 실패한 컨테이너 재시작, 컨테이너 추가 또는 제거로 클러스터 확장과 같은 몇 가지 작업을 처리한다. Docker Swarm, Mesos, Nomad 또는 Kubernetes와 같은 다양한 컨테이너 조정 도구를 이러한 목적에 사용할 수 있다. 처음에는 Kubernetes를 위한 구성 파일이 포함될 것이다. 각 서비스는 자체 부하 분산 장치가 있는 자체 포드에서 실행되도록 구성될 것이다. 이것은 개별 서비스가 서로 독립적으로 확장될 수 있도록 행해진다. 수평 Pod Autoscaler 서비스 확장을 관리하는 데 사용된다[72].

REFERENCES

- [1] Hannah Ritchie Max Roser. *Technological Progress*. <https://www.ft.com/content/cb56d86c-88d6-11e7-afd2-74b8ecd34d3b>. Accessed: 03-06-2018. 2017.
- [2] Christine Legner and Kristin Wende. "The challenges of inter-organizational business process design – a research agenda". In: (2007).
- [3] Benjamin E. Hermalin and Michael L. Katz. "Moral Hazard and Verifiability: The Effects of Renegotiation in Agency". In: (1990).
- [4] Audun Jøsang. "The right type of trust for distributed systems". In: *Proceedings of the 1996 workshop on New security paradigms*. ACM. 1996, pp. 119–131.
- [5] Israel Z Ben-Shaul and Gail E Kaiser. "A paradigm for decentralized process modeling and its realization in the oz environment". In: *Proceedings of the 16th international conference on Software engineering*. IEEE Computer Society Press. 1994, pp. 179–188.
- [6] Ray Fisman and Roberta Gatti. "Bargaining for bribes: The role of institutions". In: *International handbook on the economics of corruption* (2006), pp. 127–139.
- [7] Jörg Becker, Michael Rosemann, and Christoph Von Uthmann. "Guidelines of business process modeling". In: *Business Process Management*. Springer, 2000, pp. 30–49.
- [8] Manfred Reichert, Thomas Bauer, and Peter Dadam. "Enterprise-wide and cross-enterprise workflow management: Challenges and research issues for adaptive workflows". In: (1999).
- [9] Vitalik Buterin. "Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform". In: (2013).
- [10] Vitalik Buterin and Karthik Gollapudi. *A Next-Generation Smart Contract and Decentralized Application Platform*. <https://github.com/ethereum/wiki/wiki/White-Paper/f18902f4e7fb21dc92b37e8a0963eec4b3f4793a>. Accessed: 22-05-2018.
- [11] Ian Grigg. "The Ricardian Contract". In: (2004).
- [12] Nick Sabo. "Formalizing and Securing Relationships on Public Networks". In: (1997).
- [13] Telser. "A Theory of Self-Enforcing Agreements". In: (1980).
- [14] David Joulfaian Douglas Holtz-Eakin and Harvey S. Rosen. "Sticking it out: entrepreneurial survival and liquidity constraints". In: (1993).
- [15] Toshi wallet now supports ERC20 tokens and ERC721 collectibles. <https://blog.toshi.org/toshi-wallet-now-supports-erc20-tokens-and-erc721-collectibles-e718775895aa>. Accessed: 30-08-2018.
- [16] *ERC-20 Token Standard*. <https://eips.ethereum.org/EIPS/eip-20>. Accessed: 30-08-2018.
- [17] Daniel IA Cohen and Daniel IA Cohen. *Introduction to computer theory*. Vol. 2. Wiley New York, 1991.
- [18] Marko Vukolić. "Rethinking permissioned blockchains". In: *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM. 2017, pp. 3–7.
- [19] AbdulSalam Kalaji, Rob Mark Hierons, and Stephen Swift. "A search-based approach for automatic test generation from extended finite state machine (EFSM)". In: *Testing: Academic and Industrial Conference-Practice and Research Techniques, 2009. TAIC PART'09*. IEEE. 2009, pp. 131–132.
- [20] M.G. Gouda, E.G. Manning, and Y.T. Yu. "On the Progress of Communication between Two Finite State Machines". In: (1984).
- [21] G Pace and J Schapachnik. "Contracts for Interacting Two-Party Systems". In: (2012).
- [22] Mark D. Flood and Oliver R Goodenough. "Contract as Automaton: The Computational Representation of Financial Agreements". In: (2015).
- [23] Davide Basile, Pierpaolo Degano, and Gian-Luigi Ferrari. "Automata for Service Contracts". In: (2014).
- [24] Pierpaolo Degano Davide Basile and Gian-Luigi Ferrari. "From Orchestration to Choreography through Contract Automata". In: (2014).
- [25] Ian Ayres and Robert Gertner. "Filling Gaps in Incomplete Contracts: An Economic Theory of Default Rules". In: (1989).
- [26] Jan L.G. Dietz. "Understanding and Modeling Business Processes with DEMO". In: (1999).
- [27] YoungJoon Byun, Beverly A. Sanders, and Chang-Sup Keum. "Design Patterns of Communicating Extended Finite State Machines in SDL". In: (2001).
- [28] Petri. "Kommunikation mit Automaten". In: (1962).
- [29] Dennis Kafura. *Notes on Petri Nets*. <http://people.cs.vt.edu/kafura/ComputationalThinking/Class-Notes/Petri-Net-Notes-Expanded.pdf>. Accessed: 01-09-2018.
- [30] Wil M.P. van der Aalst. "The Application of Petri Nets to Workflow Management". In: (1998).
- [31] Jan Recker et al. "How good is bpmn really? Insights from theory and practice". In: (2006).
- [32] Wil M.P. van der Aalst et al. "Life After BPEL?" In: (2005).
- [33] Luciano García-Bañuelos et al. "Optimized Execution of Business Processes on Blockchain". In: (2017).
- [34] Jan L.G. Dietz. "DEMO: Towards a discipline of organisation engineering". In: (1999).
- [35] *JSONForms - React*. <https://jsonforms.io/>. Accessed: 30-08-2018.
- [36] *JSONForm - Bootstrap 3*. <https://github.com/jsonform/jsonform>. Accessed: 30-08-2018.
- [37] *Mozilla react-jsonschema-form*. <https://github.com/mozilla-services/react-jsonschema-form>. Accessed: 30-08-2018.
- [38] *Angular Schema Form*. <http://schemaform.io/>. Accessed: 30-08-2018.
- [39] *Open Document Format*. <http://www.opendocumentformat.org/>. Accessed: 30-08-2018.
- [40] Sindhu Sajana and Sethumadhavan. "On Blockchain Applications: Hyperledger Fabric And Ethereum". In: (2018).
- [41] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. ACM. 1971, pp. 151–158.
- [42] Daniel Brand and Pitro Zafiroplulo. "On communicating finite-state machines". In: *Journal of the ACM (JACM)* 30.2 (1983), pp. 323–342.
- [43] Robert M Hierons AbdulSalam Kalaji and Stephen Swift. "New approaches for passive testing using an Extended Finite State Machine specification". In: (2003).
- [44] O Sury and R Edmonds. *Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC*. Tech. rep. 2017.
- [45] NIST. *Transition Plans for Key Establishment Schemes using Public Key Cryptography*. <https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes>. Accessed: 13-07-2018. 2017.
- [46] Daniel J. Bernstein et al. "High-speed high-security signatures". In: *Journal of Cryptographic Engineering* 2.2 (2012), pp. 77–89. ISSN: 2190-8516. DOI: 10.1007/s13389-012-0027-1. URL: <https://doi.org/10.1007/s13389-012-0027-1>.

- [47] Henri Gilbert and Helena Handschuh. "Security analysis of SHA-256 and sisters". In: *International workshop on selected areas in cryptography*. Springer. 2003, pp. 175–193.
- [48] NIST. *NIST Policy on Hash Functions*. <https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions>. Accessed: 13-07-2018. 2015.
- [49] Bruce Schneier and John Kelsey. "Cryptographic Support for Secure Logs on Untrusted Machines." In: *USENIX Security Symposium*. Vol. 98. 1998, pp. 53–62.
- [50] Peter Bailis and Ali Ghodsi. "Eventual consistency today: Limitations, extensions, and beyond". In: *Queue* 11.3 (2013), p. 20.
- [51] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.
- [52] Miguel Castro and Barbara Liskov. *Byzantine fault tolerance*. US Patent 6,671,821. 2003.
- [53] Git Documentation. *Git Branching - Rebasing*. <https://git-scm.com/book/en/v2/Git-Branching-Rebasing>. Accessed: 09-08-2018.
- [54] European Parliament. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL: on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN>. Accessed: 12-07-2018. 2016.
- [55] Olly Jackson. "Is it possible to comply with GDPR using blockchain?" In: *International Financial Law Review* (2018).
- [56] *Blockchain costs per transaction*. <https://www.blockchain.com/charts/cost-per-transaction>. Accessed: 05-09-2018.
- [57] Emanuel Palm. *Implications and Impact of Blockchain Transaction Pruning*. 2017.
- [58] Wenting Li et al. "Towards scalable and private industrial blockchains". In: *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM. 2017, pp. 9–14.
- [59] Serguei Popov. "A probabilistic analysis of the nxt forging algorithm". In: *Ledger* 1 (2016), pp. 69–83.
- [60] Waves Platform. *Blockchain Leasing For Proof Of Stake*. <https://blog.wavesplatform.com/blockchain-leasing-for-proof-of-stake-bac5335de049>. Accessed: 13-07-2018. 2018.
- [61] mthcl. "The math of Nxt forging". In: (2014).
- [62] Waves generators. <http://dev.pywaves.org/generators/>. Accessed: 28-08-2018.
- [63] Nxt Blockchain Explorer. <https://nxtportal.org/monitor/>. Accessed: 28-08-2018.
- [64] LTO. "LTO Token Economy". In: (2018).
- [65] Kofman Begicheva. "Fair Proof of Stake". In: (2018).
- [66] S. Haber and W.S. J Stornetta. "How to time-stamp a digital document". In: (1991).
- [67] Ralph C. Merkle. "Method of providing digital signatures". U.S. pat. 4309569. Jan. 5, 1982.
- [68] A. Begicheva and I. Smagin. "RIDE: a Smart Contract Language for Waves". Pat. 2018.
- [69] Saifedean Ammous. "Blockchain Technology: What is it good for?" In: (2016).
- [70] *BIP 141: Segregated Witness (Consensus layer)*. <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>. Accessed: 05-09-2018.
- [71] *Nothing considered a look at nothing at stake vulnerability for cryptocurrencies*. <https://pivx.org/nothing-considered-a-look-at-nothing-at-stake-vulnerability-for-cryptocurrencies/>. Accessed: 30-08-2018.
- [72] Kubernetes. *Kubernetes, Horizontal Pod Autoscaling*. <https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod-autoscaler.md>. Accessed: 03-08-2018.

#	Field Name	Length
1	Version	1
2	Timestamp	8
3	Parent block signature	64
4	Consensus block length	4
5	Base target	8
6	Generation Signature	32
7	Transaction list Hash	32
8	Anchor Merkle root	32
9	Generator public key	32
10	Block's signature	64

TABLE 1: Key block structure

#	Field Name	Length
1	Amount of transactions (x)	4
2	Transaction #1 bytes	113
...
$2 + (K - 1)$	Transaction #K bytes	113

TABLE 2: Key block attachment

#	Field Name	Length
1	Version	1
2	Timestamp	8
3	Parent block signature	64
4	Consensus block length	4
5	Base target	8
6	Generation Signature	32
7	Transaction list Hash	32
8	Transaction #1 bytes	TODO
...
$8 + (K - 1)$	Transaction #K bytes	TODO
$9 + (K - 1)$	Balance change summary entry #1	40 (Table 4)
...
$9 + (K - 1) + (N - 1)$	Balance change summary entry #N	40 (Table 4)
$10 + (K - 1) + (N - 1)$	Generator public key	32
$11 + (K - 1) + (N - 1)$	Block's signature	64

TABLE 3: Summary Block structure

#	Field Name	Length
2	Wallet address	32
3	Balance change	8

TABLE 4: Balance summary entry

#	Field Name	Length
1	Transaction type	1
2	Anchor hash	32
3	Fee	8
4	Timestamp	8
5	Signature	64

TABLE 5: Anchor transactions structure

#	Field Name	Length
1	Transaction type	1
2	Sending address	32
3	Receiving address	32
4	Amount	8
5	Fee	8
6	Timestamp	8
7	Signature	64

TABLE 6: Transfer transaction structure

#	Field Name	Length
1	Transaction type	1
2	Id	32
3	Sending address	32
4	Receiving address	32
5	Expiration Date	8
6	Certificate Type	32
7	Fee	8
8	Timestamp	8
9	Signature	64

TABLE 7: Issue certificate transaction structure

#	Field Name	Length
1	Transaction type	1
2	Id	32
3	New expiration Date	8
4	Fee	8
5	Timestamp	8
6	Signature	64

TABLE 8: Update certificate transaction structure

#	Field Name	Length
1	Transaction type	1
2	Sending address	32
3	Receiving address	32
4	Amount	8
5	Fee	8
6	Timestamp	8
7	Signature	64

TABLE 9: Lease transaction structure

#	Field Name	Length
1	Transaction type	1
2	Sending address	32
3	Receiving address	32
4	Amount	8
5	Fee	8
6	Timestamp	8
7	Signature	64

TABLE 10: Cancel Lease transaction structure