

알고랜드¹ (Algorand ALGO)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다

Jing Chen

컴퓨터과학과

스토니브룩 대학교

Stony Brook, NY 11794, USA

jingchen@cs.stonybrook.edu

Silvio Micali

CSAIL

MIT

Cambridge, MA 02139, USA

silvio@csail.mit.edu

초록

공개 원장은 모든 사람이 읽고 데이터를 추가할 수 있는, 변조 불가능한 데이터의 순차적 기록이다. 공개 원장은 무수히 많은 매력적인 활용처를 가진다. 소유권, 판매, 지불과 같은 모든 거래를 발생 순서 그대로 누구나 볼 수 있는 형태로 안전하게 기록할 수 있다. 공개 원장은 부패를 억제할 뿐 아니라 암호화폐나 스마트 계약과 같은 고도화된 응용을 가능하게 한다. 이는 민주 사회의 운영 방식을 근본적으로 혁신할 잠재력을 갖는다. 그러나 현재 구현된 방식은 확장성이 떨어져 이러한 잠재력을 달성하지 못한다.

Algorand는 공개 원장을 구현하는 민주적이고 효율적인 방법이다. 작업 증명에 기반한 기존 구현과 달리, 매우 미미한 수준의 연산만 요구하며, 압도적으로 높은 확률로 “포크”가 발생하지 않는 거래 기록을 생성한다.

Algorand는 (새롭고 매우 빠른) 메시지 전달 기반 비잔틴 합의에 기초한다.

구체성을 위해 우리는 Algorand를 화폐 플랫폼으로서만 설명한다.

1 서론

화폐는 점점 더 가상화되고 있다. 미국 달러의 약 80%가 단지 원장 기록으로만 존재한다는 추정치가 있다 [5]. 다른 금융 상품들 또한 같은 방향으로 나아가고 있다.

이상적인 세계에서는, 보편적으로 신뢰받는 중앙 기관이 존재하고 그 기관이 모든 사이버 공격에 면역이라면, 화폐와 기타 금융 거래는 전자적으로만 이루어질 수 있을 것이다. 그러나 현실은 그렇지 않다. 따라서 비트코인 [29]과 같은 탈중앙화 암호화폐, 이더리움과 같은 “스마트 계약” 시스템이 제안되었다 [4]. 이러한 시스템의 핵심에는 결제부터 계약까지 다양한 거래를 신뢰성 있게 변조 불가능하게 기록하는 공유 원장이 존재한다. 이러한 변조 불가능성을 보장하기 위한 기술로 **블록체인**이 채택되었다. 블록체인은 암호화폐 [29], 금융 응용 [4], 사물인터넷 [3]과 같은 응용의 기반이 된다. 블록체인 기반 원장을 관리하기 위해 **작업 증명** [29], **지분 증명** [2], **실용적 비잔틴 내결함성** [8], 혹은 이들의 조합 등 여러 기법이 제안되었다.

그러나 현재의 원장은 관리 효율성이 떨어질 수 있다. 예를 들어, 비트코인의 **작업 증명** 방식(원래 [14]에서 제안된 개념에 기반)은 막대한 연산을 요구하고, 낭비적이며 확장성이 떨어진다 [1]. 게다가 이는 **사실상** 소수에게 권력을 집중시킨다.

따라서 우리는 신뢰할 수 있고 위배될 수 없는 중앙 기관이 운영하는 집중형 시스템의 편리함과 효율성을 제공하면서, 현존하는 탈중앙화 구현의 비효율성과 취약점을 극복하는 새로운 공개 원장 구현 방식을 제안하고자 한다. 우리는 이 방식을 *Algorand*라 부른다. 이는 지금까지 구성된 원장을 기반으로, 다음 유효 거래 블록을 구성할 책임을 가진 검증자 집합을 알고리즘적 무작위성에 따라 선택하는 방식이다. 우리는 이러한 선택이 조작으로부터 증명 가능하게 안전하며, 최종적으로는 모두가 명확히 알 수 있도록 하고, 마지막 순간까지 예측 불가능하도록 보장한다.

Algorand의 접근은 매우 민주적이다. 즉, 원칙적으로나 *사실상*으로도 비트코인에서의 “채굴자”와 “일반 사용자”처럼 사용자 계층을 나누지 않는다. Algorand에서는 “모든 권한이 모든 사용자 집합에 속한다.”

Algorand의 중요한 성질 중 하나는 거래 이력이 포크될 확률이 매우 작다는 것이다(예: 1조분의 1 혹은 10^{-18} 수준). 또한 Algorand는 법적·정치적 우려에도 대응할 수 있다.

Algorand의 접근은 블록체인뿐 아니라 블록의 변조 불가능한 순차열을 생성하는 모든 방식에 적용 가능하다. 실제로 우리는 블록체인보다 더 효율적인, 독립적으로도 흥미로울 수 있는 방법을 제시한다.

1.1 비트코인의 가정과 기술적 문제들

비트코인은 매우 기발한 시스템이며, 이후 방대한 연구를 촉발시켰다. 그러나 동시에 여러 문제를 안고 있다. 여기서는 비트코인과 같이 작업 증명에 기반한 암호화폐들이 공유하는 근본적 가정과 기술적 문제들을 요약한다.

이 요약을 위해, 비트코인에서 한 사용자가 디지털 서명 체계의 다수의 공개키를 가질 수 있고, 화폐는 공개키와 연계되며, 지불은 한 공개키에서 다른 공개키로 일정 금액을 이전하는 디지털 서명임을 상기하면 충분하다. 본질적으로 비트코인은 처리된 모든 지불을 블록들의 연쇄 $B_1, B_2, \dots$ 로 구성한다. 각 블록은 다수의 지불로 이루어져 있으며, B_1 의 모든 지불(순서와 무관), 이어서 B_2 의 모든 지불(순서와 무관), 등으로 이어지는 일련의 지불은 유효한 지불 시퀀스를 형성한다. 각 블록은 평균적으로 10분마다 생성된다.

이 블록들의 연쇄는 *체인*이라 불린다. 이는 단일 블록이라도 변경되면 이후 모든 블록에 그 영향이 전파되도록 구조화되어 있어, 지불 이력의 변조를 쉽게 탐지할 수 있도록 한다. (이는 각 블록에 이전 블록의 *암호학적 해시*를 포함시킴으로써 달성된다.) 이러한 블록 구조를 *블록체인*이라 부른다.

가정: 계산력의 정직한 다수

비트코인은 악의적 개체(또는 악의적으로 협력하는 개체 집단)가 블록 생성에 투입되는 계산력의 과반수를 통제하지 못한다고 가정한다. 만약 그런 개체가 존재한다면, 그는 블록체인을 변조하여 지불 이력을 임의로 다시 작성할 수 있다. 특히 그는 지불 p 를 수행하고 대가를 받은 후, p 의 흔적을 완전히 “지워버릴” 수 있다.

기술적 문제 1: 연산 낭비

비트코인의 작업 증명 기반 블록 생성은 엄청난 계산량을 요구한다. 현재 시스템 내 공개키가 수십만 개에 불과한 상황에서도, 세계 최상위 500대 슈퍼컴퓨터가 합쳐도 비트코인 네트워크에서 요구되는 총 계산력의 고작 12.8%밖에 제공하지 못한다. 사용자가 크게 늘어나면, 요구되는 계산량은 엄청나게 증가할 것이다.

기술적 문제 2: 권력 집중

오늘날 막대한 계산 요구로 인해, 일반 데스크톱(하물며 휴대전화로는 더더욱)으로 블록을 생성하려 시도하는 사용자는 오히려 손해를 본다. 즉, 보통의 컴퓨터로 블록을 생성할 경우, 계산에 필요한 전기 비용이 기대되는 보상보다 크다. 오직 블록 “채굴”만을 위해 제작된 특수 컴퓨터를 대규모로 모은 채굴풀에서만 블록 생성으로 이익을 낼 수 있다. 따라서 오늘날에는 *사실상* 두 계층의 사용자가 존재한다. 즉, 단순히 지불만 수행하는 일반 사용자와, 새 블록 탐색만을 수행하는 전문 채굴풀이다.

따라서 최근에는 블록 생성에 필요한 총 계산력이 단지 다섯 개 채굴풀 안에 집중된 상황이다. 이런 조건에서는 계산력의 과반이 정직하다는 가정이 설득력을 잃는다.

기술적 문제 3: 모호성

비트코인에서 블록체인은 반드시 유일하지 않다. 실제로 최신 부분은 종종 포크된다. 예를 들어, 한 사용자에게 블록체인은 $B_1, \dots, B_k, B_{k+1}^1, B_{k+2}^1, B_{k+3}^1$ 일 수 있고, 다른 사용자에게는 $B_1, \dots, B_k, B_{k+1}^2, B_{k+2}^2, B_{k+3}^2$ 일 수 있다. 체인에 여러 블록이 추가된 후에야 비로소 처음 $k + 3$ 개의 블록이 모든 사용자에게 동일하다고 reasonably 확신할 수 있다. 따라서 체인의 마지막 블록에 포함된 지불을 곧바로 신뢰할 수 없다. 더 현명한 방법은 그 블록이 블록체인 내에서 충분히 깊어지고, 따라서 충분히 안정적이 될 때까지 기다리는 것이다.

별개로, 비트코인에 대해 *법 집행*과 *통화 정책* 측면의 우려 또한 제기되어 왔다.²

1.2 Algorand 요약

환경 Algorand는 매우 까다로운 환경에서 동작한다. 간단히 정리하면,

(a) **퍼미션리스 및 퍼미션드 환경.** Algorand는 완전히 퍼미션리스 환경에서도 효율적이고 안전하게 동작한다. 즉, 아무런 심사나 허가 없이도 임의의 수많은 사용자가 언제든지 시스템에 참여할 수 있는 환경이다. 물론, *퍼미션드* 환경에서는 Algorand가 더욱 안정적으로 작동한다.

(b) **매우 적대적인 환경.** Algorand는 다음과 같은 능력을 가진 매우 강력한 적대자(Adversary)에 대해서도 안전하다.

1. 원하는 시점에 *임의의 사용자*를 즉시 *부패(corrupt)*시킬 수 있다. 단, 퍼미션리스 환경에서는 시스템 내 자금의 $2/3$ 이상이 정직한 사용자에게 속해야 한다. (퍼미션드 환경에서는 보유 자금과 관계없이 사용자 중 $2/3$ 이상이 정직하면 충분하다.)
2. 부패한 모든 사용자를 *완전히 통제하고 완벽히 조율*할 수 있다.
3. 모든 메시지의 전달을 *스케줄링*할 수 있다. 단, 정직한 사용자가 보낸 각 메시지 m 은 그 크기에만 의존하는 시간 λ_m 내에 정직한 사용자 중 95% 이상에게 도달해야 한다.

주요 성질

이처럼 강력한 적대자가 존재하더라도, Algorand에서는

- **필요한 계산량이 최소화된다.** 시스템 내 사용자 수와 무관하게, 약 1,500명의 사용자 각각이 몇 초 이내의 계산만 수행하면 충분하다.
- **새 블록은 10분 이내에 생성되며, 사실상 블록체인을 벗어나지 않는다.** 예를 들어, 첫 번째 구현에서 블록 생성의 기대 시간은 $\Lambda + 12.4\lambda$ 이하이다. 여기서 Λ 는 블록 크기와 관계없이 피어-투-피어 가십 방식으로 블록이 전파되는 데 필요한 시간이고, λ 는 1,500개의 200B 길이 메시지를 전파하는 데 필요한 시간이다. (진정한 탈중앙 시스템에서는 Λ 가 본질적 지연 시간이며, Algorand에서 블록 생성의 병목 요소는 네트워크 속도이다.) 두 번째 구현은 실제로 실험적으로 테스트되었으며(?에 의해), 블록 생성 시간이 40초 미만임이 확인되었다.
- **게다가 Algorand의 블록체인은 무시할 수 있을 정도의 확률(1조분의 1 이하)로만 포크된다.** 따라서 사용자는 새로운 블록이 나타나면 즉시 그 안의 지분을 신뢰할 수 있다.
- **모든 권한은 사용자 자신에게 있다.** Algorand는 진정한 분산 시스템이다. 특히, 비트코인에서의 “채굴자”처럼 거래 인정을 통제할 수 있는 외부적 개체는 존재하지 않는다.

Algorand의 기술

1. **새롭고 빠른 비잔틴 합의 프로토콜.** Algorand는 새로운 암호학적 *메시지 전달* 이진 비잔틴 합의(BA) 프로토콜 BA^* 를 통해 블록을 생성한다. 프로토콜 BA^* 는 추가적인 성질들을 만족할 뿐 아니라 매우 빠르다. 간단히 말해, 이 프로토콜은 3단계 루프로 이루어져 있으며, 각 플레이어 i 는 하나의 메시지 m_i 를 모든 플레이어에게 보낸다. 완전하고 동기적인 네트워크에서 플레이어의 $2/3$ 이상이 정직하다면, 각 루프가 끝날 때마다 $1/3$ 이상의 확률로 합의에 도달한다. (중요한 점은, BA^* 는 Pease, Shostak, Lamport [31]의 원래 비잔틴 합의 정의를 어떠한 약화도 없이 만족한다는 것이다.)

Algorand는 이 이진 BA 프로토콜을 확장하여, 우리의 다른 통신 모델 하에서 새로운 블록에 대한 합의를 달성한다. 합의된 블록은 이후 적절한 검증자들의 디지털 서명으로 *인증*되며 네트워크 전반으로 전파된다.

2. **암호학적 소팅(Cryptographic Sortition).** 프로토콜 BA^* 가 매우 빠르긴 하지만, 수백만 사용자가 참여할 경우 속도 개선이 추가적으로 필요하다. 따라서 Algorand는 BA^* 실행에 참여하는 플레이어들을 전체 사용자 집합의 소규모 부분집합으로 선택한다. 권력 집중 문제를 피하기 위해, 각 새로운 블록 B^r 은 별도의 *선정된 검증자 집합* SV^r 에 의해 BA^* 실행을 통해 구성·합의된다. 이러한 집합을 직접 선택하는 것은 곧 블록 B^r 자체를 선택하는 것만큼 어려울 수 있다. 우리는 이를 해결하기 위해, Maurice Herlihy의 통찰을 수용해 *암호학적 소팅*이라 부르는 방식을 도입한다. 소팅은 자격을 갖춘 대규모 집단에서 임의로 공직자를 추천하는 제도이다 [6]. (이는 고대 아테네, 피렌체, 베네치아 공화국에서 수 세기에 걸쳐 시행되었으며, 현대 사법제도에서는 배심원 선발에 무작위 추출이 사용된다. David Chaum [9]도 최근 선거에 무작위 추출 방식을 제안하였다.)

탈중앙 시스템에서는 검증자 집합 SV^r 의 무작위 선택을 위한 난수를 어떻게 정하는지가 문제가 된다. 따라서 우리는 암호학적 기법을 사용하여, 모든 사용자 집합에서 자동적이고(즉, 메시지 교환이 필요 없는) 무작위로 검증자 집합을 선택한다. 구체적으로는 이전 블록 B^{r-1} 로부터 새로운 블록 제안자인 *리더*와, 그 블록에 합의를 이끌 *검증자 집합* SV^r 를 자동적으로 결정한다. 악의적 사용자가 B^{r-1} 의 구성(예: 특정 지분 포함)을 조작할 수 있기 때문에, 우리는 특별히 추가 입력을 설계하여 r 번째 블록의 리더와 검증자 집합 SV^r 가 실제로 무작위로 선택되었음을 증명한다.

3. **수량(Seed) Q^r .** 우리는 블록체인에서 마지막 블록 B^{r-1} 을 사용해 새로운 블록 B^r 을 구성할 리더와 검증자 집합을 자동적으로 결정한다. 그러나 이 방식의 문제는, 이전 라운드에서 단 하나의 지분만 바뀌어도 강력한 적대자가 다음 라운드의 리더를 거의 완전히 통제할 수 있게 된다는 점이다. 비록 시스템 내 단 1/1000의 사용자자금만 제어하더라도, 모든 리더가 악의적이 되도록 만들 수 있다. (직관은 4.1절 참조.) 이는 모든 지분 증명 방식에서 핵심적인 도전 과제이며, 지금까지 만족스럽게 해결되지 못했다.

이를 해결하기 위해, 우리는 별도의 정교하게 정의된 수량 Q^r 을 설계·갱신한다. 이 값은 강력한 적대자조차 *예측 불가능하고 영향을 줄 수 없는 것*임이 증명된다. 우리는 이를 *n번째 시드*라고 부르며, Algorand는 Q^r 로부터 비밀 암호학적 소팅을 통해 r 번째 블록 생성 과정에서 특별한 역할을 맡을 모든 사용자를 선택한다.

4. **비밀 암호학적 소팅과 비밀 자격 증명.** 마지막 블록 B^{r-1} 을 무작위 선택 기반으로 사용하여 새로운 블록 B^r 의 리더와 검증자 집합을 정하는 것만으로는 충분하지 않다. B^{r-1} 이 공개되면 Q^{r-1} 또한 알려지고, 따라서 B^r 을 담당할 리더와 검증자도 공개된다. 이 경우, 적대자는 이들을 즉시 모두 부패시켜 블록을 완전히 장악할 수 있다.

이를 방지하기 위해, 리더(그리고 검증자들 역시)는 자신의 역할을 *비밀리에* 알게 되지만, 이를 증명할 수 있는 *자격 증명*을 계산할 수 있다. 리더는 자신이 차기 블록의 리더임을 알게 되면, 먼저 비밀리에 블록을 준비한 후, 자격 증명과 함께 이를 전파한다. 이 시점에서 적대자가 그를 부패시켜도 이미 메시지가 퍼져 있기 때문에 블록 선택에는 영향을 줄 수 없다. 이는 위키리크스에 의해 퍼진 메시지를 정부가 나중에 회수할 수 없는 것과 같다.

우리가 보게 되겠지만, 리더의 유일성이나 리더 본인조차 확신하는 것을 보장할 수는 없다. 그러나 Algorand에서는 모호성 없이 진전이 보장된다.

5. **플레이어 교체 가능성.** 리더는 블록 제안 후 “사라져도”(혹은 적대자에게 부패되어도) 무방하다. 하지만 검증자 집합 SV^r 은 단순하지 않다. 이들은 리더가 제안한 블록에 충분한 서명을 붙여 인증해야 하므로, 먼저 그 블록에 대해 비잔틴 합의를 수행해야 한다. 문제는 BA^* 가 여러 단계를 필요로 하고, 각 단계에서 참여자의 2/3 이상이 정적해야 한다는 점이다. 효율성을 위해 각 BA^* 의 플레이어 집합은 전체 사용자 집합에서 무작위로 선택된 작은 SV^r 이므로, 적대자는 전체 사용자 1/3을 부패시키지 못하더라도 특정 SV^r 전체를 부패시킬 수 있다.

다행히도, 우리는 BA^* 프로토콜이 *플레이어 교체 가능성(player-replaceable)*을 만족함을 증명한다. 이는 각 단계가 완전히 새롭게 무작위로 선택된 독립적인 집합에 의해 실행되더라도 프로토콜이 올바르게 합의에 도달한다는 뜻이다. 따라서 수백만 사용자가 있는 상황에서는, 한 단계의 작은 검증자 집합과 다음 단계의 집합이 거의 겹치지 않는다.

또한 단계별 검증자 집합의 크기도 서로 달라질 수 있고, 각 집합의 구성원은 다음 집합이 누구인지 알지 못하며 내부 상태를 전달하지 않는다.

추가 성질/기술: 느긋한 정직성(Lazy Honesty). 정직한 사용자는 온라인 상태에서 프로토콜을 실행하는 등, 규정된 지침을 따른다. Algorand에서는 계산 및 통신 요구가 크지 않으므로, 백그라운드에서 프로토콜을 실행하는 것은 큰 부담이 아니다. 물론 정직한 사용자 일부가 갑작스러운 접속 장애나 재부팅으로 부재하는 경우는 자동으로 허용된다. 그러나 Algorand는 정직한 사용자가 대부분 오프라인 상태인 새로운 모델에도 적용할 수 있다. 이를 우리는 비공식적으로 다음과 같이 소개한다.

느긋한 정직성. 대략적으로, 사용자 i 가 “느긋하지만 정직하다”는 것은 (1) 프로토콜 참여 요청을 받으면 모든 규정을 따르며, (2) 이러한 요청은 드물게, 충분한 사전 통보와 함께 이루어진다는 뜻이다.

이러한 느슨한 정직성 개념하에서, 필요한 순간 정직한 사용자가 실제로 참여할 가능성이 더욱 높아지고, Algorand는 다음을 보장한다.

어떤 시점에 참여자 대다수가 악의적이라 하더라도, 시스템은 안전하게 동작한다.

1.3 관련 연구

작업 증명 기반 접근([29], [4] 참조)은 우리의 접근과 상당히 직교한다. 메시지 전달 기반 비잔틴 합의나 실용적 비잔틴 내결함성([8] 참조) 또한 마찬가지다. 사실 이러한 프로토콜들은 전체 사용자 집합에서 실행될 수 없으며, 우리 모델에서는 적절히 작은 사용자 집합으로 제한될 수도 없다. 실제로 우리의 강력한 적대자는 특정 BA 프로토콜 실행을 담당하는 작은 사용자 집합 전체를 즉시 부패시킬 수 있다.

우리의 접근은 지분 증명 [2]과도 연관된다고 볼 수 있다. 블록 생성에서 사용자의 “권한”이 *시스템 내 보유 자금*에 비례한다는 점에서 그렇다. (이는 예치금(escrow)과 같은 방식과 대비된다.)

우리와 가장 가까운 연구는 Pass와 Shi의 *Sleepy Consensus Model* [30]이다. 이들은 작업 증명이 요구하는 막대한 계산을 피하기 위해, Algorand의 비밀 암호학적 소팅을 차용(그리고 친절히 인용)한다. 이 중요한 공통점을 바탕으로, 두 논문 사이에는 여러 중요한 차이가 존재한다. 특히,

1. **그들의 모델은 퍼미션드 환경에만 국한된다.** 반면 Algorand는 퍼미션리스 시스템이기도 하다.
2. **그들은 나카모토 스타일 프로토콜을 사용하기 때문에, 블록체인이 자주 포크된다.** 작업 증명은 제거했지만, 그들의 프로토콜에서는

비밀리에 선택된 리더가 가장 긴 유효 블록체인을 연장한다. 따라서 포크가 불가피하며, 블록이 체인 내에서 충분히 “깊어질” 때까지 기다려야 한다. 특히, 적대자가 적응적 부패를 할 수 있는 상황에서 목표를 달성하려면, 블록은 시스템 전체 사용자 수 N 에 대해 $poly(N)$ 깊이어야 한다. 예를 들어, 블록 생성이 1분에 하나 가능하고 $N = 100$ 만이라면, 블록이 N^2 깊이가 되려면 약 200만 년, N 깊이가 되려면 약 2년을 기다려야 한다. 반면 Algorand의 블록체인은 적대자가 즉시 적응적으로 사용자를 부패시키더라도 무시할 수 있는 확률로만 포크되며, 새로운 블록은 곧바로 신뢰할 수 있다.

3. 그들은 개별 비잔틴 합의를 다루지 않는다. 그들은 “성장하는 값 시퀀스에 대한 최종적 합의”만 보장한다. 즉, 상태 복제 프로토콜이 지 BA 프로토콜은 아니다. 따라서 특정 단일 값에 대한 비잔틴 합의를 달성하는 데 사용할 수 없다. 반면 Algorand는 원하는 경우 단 한 번만 실행하여도 수백만 사용자가 특정 값에 대해 신속히 합의할 수 있다.
4. 그들은 약하게 동기화된 시계를 필요로 한다. 즉, 모든 사용자의 시계가 작은 오차 δ 내에 있어야 한다. 반면 Algorand에서는 시계가 단지 동일한 “속도”만 가지면 된다.
5. 그들의 프로토콜은 느긋하지만 정직한 사용자 혹은 온라인 사용자 다수 정적 모델에서 동작한다. 이들은 Algorand가 제시한 “정직한 사용자가 대규모로 오프라인 상태가 될 수 있음”이라는 문제의식을 인정하고, 이에 대한 응답으로 느긋한 정직성 모델을 도입한 것을 언급한다. 그들의 프로토콜은 느긋한 정직성 모델뿐 아니라 적대적 수면 모델(adversarial sleepy model)에서도 동작한다. 이 모델에서는 적대자가 온라인/오프라인 사용자를 선택하되, 항상 온라인 사용자 다수는 정직해야 한다.³

6. 그들은 단순 다수 정직성을 요구한다. 반면 Algorand의 현재 버전은 2/3 정직 다수를 요구한다.

또 다른 가까운 연구는 Kiayias, Russell, David, Oliynykov의 Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol [20]이다. 이 시스템 또한 Algorand 이후에 등장하였다. Ouroboros 역시 암호학적 소팅을 사용하여 작업 증명을 제거한다. 그러나 이는 다시 나카모토 스타일 프로토콜이며, 포크가 불가피하고 자주 발생한다. (다만 sleepy consensus 모델만큼 깊은 블록을 요구하지는 않는다.) 게다가 그들의 시스템은 다음과 같은 가정에 의존한다. 저자들에 따르면, “(1) 네트워크가 고도로 동기화되어 있음, (2) 선택된 지분 보유자의 다수가 각 에포크에 필요할 때 참여 가능함, (3) 지분 보유자가 장기간 오프라인 상태로 머무르지 않음, (4) 부패의 적응성이 보안 파라미터에 선형적인 라운드 지연을 받음.”

반면 Algorand는 압도적 확률로 포크가 없으며, 위 네 가지 가정 중 어느 것도 필요로 하지 않는다. 특히, Algorand에서 적대자는 원하는 사용자를 즉시 부패시킬 수 있다.

2 예비 지식 (Preliminaries)

2.1 암호학적 기본 요소 (Cryptographic Primitives)

이상적 해시(Ideal Hashing). 우리는 효율적으로 계산 가능한 암호학적 해시 함수 H 를 가정한다. H 는 임의 길이의 문자열을 고정된 길이의 이진 문자열로 매핑한다. 오랜 전통에 따라 우리는 H 를 랜덤 오라클로 모델링한다. 즉, 가능한 각 문자열 s 를 무작위로 독립적으로 선택된(그리고 고정된) 이진 문자열 $H(s)$ 로 매핑하는 함수로 간주한다.

본 논문에서 H 의 출력은 256비트이다. 이 길이는 시스템 효율성 측면에서 충분히 짧으면서도 보안성 측면에서 충분히 길다. 예를 들어, 우리는 H 가 충돌 저항성을 가지길 원한다. 즉, 서로 다른 두 문자열 x 와 y 에 대해 $H(x) = H(y)$ 를 찾는 것이 어렵다는 것이다. H 가 256비트 출력을 갖는 랜덤 오라클일 때, 그러한 쌍을 찾는 것은 실제로 어렵다. (무작위 시도를 하고 생일 역설을 고려하면, 2^{128} 회의 시도가 필요하다.)

디지털 서명(Digital Signing). 디지털 서명은 사용자들이 비밀 키를 공유하지 않고도 정보를 상호 인증할 수 있게 한다. 디지털 서명 체계는 세 가지 빠른 알고리즘으로 구성된다: 확률적 키 생성기 G , 서명 알고리즘 S , 검증 알고리즘 V .

보안 매개변수 k 가 충분히 큰 정수일 때, 사용자 i 는 G 를 사용해 k 비트 길이의 키 쌍(즉, 문자열)을 생성한다: “공개” 키 pk_i 와 대응되는 “비밀” 서명 키 sk_i . 중요한 점은, 공개키를 알더라도 그에 대응하는 비밀키를 합리적인 시간 내에 계산할 수 없다는 것이다.

사용자 i 는 sk_i 를 사용해 메시지에 서명한다. 임의의 메시지(이진 문자열) m 에 대해, i 는 먼저 m 을 해시한 후, 입력 $(H(m), sk_i)$ 로 알고리즘 S 를 실행하여 k 비트 문자열을 생성한다.

$$sig_{pk_i}(m) \triangleq S(H(m), sk_i).$$

[^3]

이 이진 문자열 $sig_{pk_i}(m)$ 을 i 의 m 에 대한 디지털 서명이라고 하며, 맥락상 pk_i 가 분명할 경우 간단히 $sig_i(m)$ 이라고 표기한다.

누구나 pk_i 를 알고 있다면, 이를 사용해 i 가 생성한 서명의 유효성을 검증할 수 있다. 구체적으로, 입력이 (a) 플레이어 i 의 공개키 pk_i , (b) 메시지 m , (c) m 의 서명이라고 주장되는 문자열 s 일 때, 검증 알고리즘 V 는 YES 또는 NO를 출력한다.

디지털 서명 체계가 만족해야 할 성질은 다음과 같다.

1. 정상 서명은 항상 검증된다. 즉, $s = sig_i(m)$ 이면 $V(pk_i, m, s) = YES$.
2. 위조가 불가능하다. sk_i 를 알지 못한 상태에서, $V(pk_i, m, s) = YES$ 가 되도록 하는 s 를 찾는 것은 사실상 불가능하다. (Goldwasser, Micali, Rivest [17]의 강한 보안 요구에 따르면, 다른 메시지의 서명을 마음대로 얻을 수 있다고 해도 이 성질은 유지된다.)

따라서, 다른 이가 자신을 사칭해 메시지를 서명하지 못하도록, 플레이어 i 는 자신의 비밀키 sk_i 를 반드시 비밀로 유지해야 한다. 반대로, 자신이 서명한 메시지를 누구나 검증할 수 있도록, i 는 공개키 pk_i 를 널리 알려야 한다.

일반적으로, 메시지 m 은 서명 $sig_i(m)$ 으로부터 직접 복원되지 않는다. 그러나 개념적으로 편리한 “복원 가능성(retrievability)” 속성을 만족하는 것처럼 다루기 위해, 우리는 다음과 같이 정의한다.

$$SIG_{pk_i}(m) = (i, m, sig_{pk_i}(m)) \quad \text{그리고} \quad SIG_i(m) = (i, m, sig_i(m)), \quad pk_i \text{가 명확할 경우.}$$

고유 디지털 서명(Unique Digital Signing). 우리는 추가적인 성질을 만족하는 디지털 서명 체계 (G, S, V) 도 고려한다.

3. 고유성(Uniqueness). 다음 조건을 만족하는 문자열 pk', m, s, s' 를 찾는 것이 어렵다.

$$s \neq s' \quad \text{그리고} \quad V(pk', m, s) = V(pk', m, s') = 1.$$

(이 성질은 올바르게 생성되지 않은 공개키 pk' 에 대해서도 성립한다. 특히, 올바른 방식으로 G 를 사용해 생성한 키 쌍 (pk, sk) 를 알고 있다 하더라도, 동일한 메시지 m 에 대해 두 개의 서로 다른 유효 서명을 찾는 것은 사실상 불가능하다.)

비고

- **고유 서명에서 검증 가능한 난수 함수로.** 고유성 속성을 가진 디지털 서명 체계에 대해, $m \mapsto H(sig_i(m))$ 매핑은 각 문자열 m 을 고유하고 무작위로 선택된 256비트 문자열에 대응시키며, 그 유효성은 서명 $sig_i(m)$ 으로 증명 가능하다. 즉, 이상적 해시와 고유성 서명 체계를 결합하면 Micali, Rabin, Vadhan [27]이 제안한 검증 가능한 난수 함수의 기본 구현을 제공한다.
- **디지털 서명의 세 가지 용도.** Algorand에서 사용자 i 는 디지털 서명을 다음에 사용한다.
 - (1) 자신의 지분을 인증할 때. 이 경우 키는 장기 사용 가능하며, 일반적인 서명 체계에서 생성된다.
 - (2) 자신이 특정 라운드 r 의 특정 단계 s 에서 역할을 맡을 권한이 있음을 증명하는 자격 증명을 생성할 때. 이 경우 키는 장기 사용 가능하되, 고유성 속성을 가진 체계에서 생성되어야 한다.
 - (3) 각 단계에서 자신이 보내는 메시지를 인증할 때. 이 경우 키는 일회성이며 사용 후 파괴되지만, 일반적인 서명 체계에서 생성될 수 있다.
- **저비용 단순화.** 단순화를 위해, 우리는 각 사용자 i 가 하나의 장기 키만 가진다고 가정한다. 따라서 이 키는 고유성 속성을 가진 서명 체계에서 생성되어야 한다. 이러한 단순화는 약간의 연산 비용을 수반하지만, 보통 고유 디지털 서명은 일반 서명보다 약간 더 비싸다.

2.2 이상적 공개 원장 (The Idealized Public Ledger)

Algorand는 다음과 같은 이상적 공개 원장에 기반한 결제 시스템을 모방하고자 한다.

1. 초기 상태. 화폐는 개별 공개키에 연계되어 있으며, 각 키는 사용자가 개인적으로 생성·소유한다. 초기 공개키들을 $pk_1, \dots, pk_j$ 라 하고, 각각의 초기 금액을 $a_1, \dots, a_j$ 라 하자. 그러면 초기 상태는 다음과 같다.

$$S_0 = (pk_1, a_1), \dots, (pk_j, a_j),$$

이 상태는 시스템 내에서 공통 지식으로 가정된다.

2. 지불(Payments). pk 가 현재 $a \geq 0$ 의 화폐 단위를 가진 공개키이고, pk' 는 또 다른 공개키, a' 는 a 이하의 비음수가 아닌 수라고 하자. 그러면 (유효한) 지불 p 는 pk 에 대한 디지털 서명으로, pk 에서 pk' 로 a' 단위를 이전한다는 것을 명시하며, 추가적인 정보를 포함한다. 기호로 나타내면,

$$p = SIG_{pk}(pk, pk', a', I, H(\mathcal{I})),$$

여기서 I 는 민감하지 않으면서도 유용할 수 있는 추가 정보(예: 시간 정보, 지불 식별자 등)를 나타내고, $\mathcal{I}$ 는 민감한 추가 정보(예: 지불 사유, pk 와 pk' 소유자의 신원 등)를 나타낸다.

우리는 pk (혹은 그 소유자)를 지불자(payer), pk' (혹은 그 소유자)를 수취인(payee), a' 를 지불 p 의 금액이라고 부른다.

지불을 통한 자유로운 참여. 사용자는 언제든지 스스로 공개/비밀 키 쌍을 생성해 시스템에 참여할 수 있다. 따라서 위 지불 p 에 나타나는 pk' 는 이전에 어떤 자금도 소유하지 않았던 새로운 공개키일 수 있다.

3. **마법의 원장(Magic Ledger).** 이상적 시스템에서는 모든 지불이 유효하며, 누구나 볼 수 있도록 하늘에 “게시된” 변조 불가능한 지불 집합의 목록 L 에 나타난다.

$$L = PAY^1, PAY^2, \dots$$

각 블록 PAY^{r+1} 은 블록 PAY^r 가 등장한 이후 발생한 모든 지불의 집합으로 이루어진다. 이상적 시스템에서는 일정(혹은 유한) 시간마다 새로운 블록이 나타난다.

논의.

- **더 일반적인 지불과 미사용 트랜잭션 출력(UTXO).** 일반적으로 공개키 pk 가 금액 a 를 소유한다면, pk 의 유효한 지불 p 는 $a'_1, a'_2, \dots$ 금액을 각각 $pk'_1, pk'_2, \dots$ 로 이전할 수 있으며, 단 $\sum_j a'_j \leq a$ 여야 한다. 비트코인 및 유사 시스템에서는 공개키 pk 가 보유한 금액이 분리된 단위로 관리되며, pk 가 하는 지불 p 는 반드시 해당 단위 a 전체를 이전해야 한다. 만약 pk 가 a 중 일부 $a' < a$ 만 다른 키로 이전하고 싶다면, 남은 잔액은 **미사용 트랜잭션 출력(UTXO)**으로 다른 키(혹은 자기 자신)에게 반환해야 한다. Algorand도 분리된 금액을 가진 키와 함께 동작할 수 있다. 그러나 Algorand의 새로운 측면에 집중하기 위해, 하나의 금액만을 가진 단순한 형태의 지불 및 키로 설명하는 것이 개념적으로 더 단순하다.
- **현재 상태(Current Status).** 이상적 시스템은 시스템의 현재 상태(즉, 각 공개키가 보유한 금액)를 직접적으로 제공하지 않는다. 이 정보는 마법의 원장으로부터 유도될 수 있다. 이상적 시스템에서 적극적인 사용자는 항상 최신 상태 정보를 저장·갱신하거나, 그렇지 않으면 처음부터 다시 계산하거나 마지막 계산 시점 이후부터 재구성해야 한다. (이 논문의 차기 버전에서는 Algorand 사용자들이 효율적으로 현재 상태를 재구성할 수 있도록 개선할 예정이다.)
- **보안과 “프라이버시”.** 디지털 서명은 다른 사용자가 위조 지불을 생성하지 못하도록 보장한다. 지불 p 에서 공개키와 금액은 숨겨지지 않지만, 민감한 정보 $\mathcal{I}$ 는 숨겨진다. 실제로 p 에는 $H(\mathcal{I})$ 만 나타나며, H 가 이상적 해시 함수이므로 $H(\mathcal{I})$ 는 무작위 256비트 값이다. 따라서 $\mathcal{I}$ 를 추측 외의 방법으로 알아내는 것은 불가능하다. 그러나 지불자가 $\mathcal{I}$ 를 공개하면(예: 지불 사유 증명), 검증자는 $H(\mathcal{I})$ 를 계산하여 p 에 포함된 값과 비교함으로써 진위를 검증할 수 있다. 또한 H 가 충돌 저항성을 가지므로, 동일한 해시 값을 주는 다른 $\mathcal{I}'$ 를 찾는 것은 어렵다.

2.3 기본 개념과 표기 (Basic Notions and Notations)

키, 사용자, 소유자. 별도의 언급이 없는 한, 각 공개키(줄여서 “키”)는 장기 키이며 고유성 속성을 가진 디지털 서명 체계에 속한다. 공개키 i 는 이미 시스템에 있는 다른 공개키 j 로부터 i 에게 지불이 이루어질 때 시스템에 참여하게 된다.

표현을 풍부하게 하기 위해 키를 의인화한다. 즉, 키 i 를 “그”라고 부르며, i 가 정적하다거나, 메시지를 보내고 받는다고 말한다. 사용자(user)는 k 와 동의어이다. 키와 그것을 소유한 사람을 구분하고자 할 때에는 각각 “디지털 키”와 “소유자”라는 용어를 사용한다.

퍼미션리스 및 퍼미션드 시스템. 시스템은 **퍼미션리스(permissionless)**일 수 있는데, 이 경우 디지털 키는 언제든지 자유롭게 참여할 수 있으며 소유자는 여러 디지털 키를 가질 수 있다. 반대로 **퍼미션드(permissioned)** 시스템은 그렇지 않다.

유일 표현(Unique Representation). Algorand 내 모든 객체는 유일한 표현을 가진다. 특히 집합 $\{(x, y, z, \dots) : x \in X, y \in Y, z \in Z, \dots\}$ 는 사전 정의된 방식으로 순서화된다. 예: 먼저 x 를 사전식으로, 그다음 y 를 기준으로 하는 식이다.

동일 속도의 시계(Same-Speed Clocks). 전역 시계는 존재하지 않고, 각 사용자는 자신의 시계를 가진다. 사용자 시계는 서로 동기화될 필요는 없다. 다만, 모두 같은 속도를 가진다고 가정한다. 예를 들어, 사용자 i 의 시계로 12:00일 때, 다른 사용자 j 의 시계로는 2:30일 수 있다. 그러나 i 의 시계로 12:01이 될 때, j 의 시계는 2:31이 된다. 즉, “1분은 모든 사용자에게 동일하다(충분히, 본질적으로 동일하다).”

라운드(Rounds). Algorand는 $r = 0, 1, \dots$ 로 표시되는 논리적 단위인 **라운드**로 구성된다. 우리는 일관되게 라운드를 위 첨자로 표기한다. 예를 들어, 비수치적 개체 Q (문자열, 공개키, 집합, 디지털 서명 등)가 라운드 r 과 관련된 경우, 단순히 Q^r 이라 쓴다. 반대로 Q 가 실제 수일 경우(즉, 수로 해석될 수 있는 이진 문자열이 아닐 경우)에는 $Q(r)$ 라고 써서 r 이 Q 의 지수가 아님을 명확히 한다.

라운드 $r > 0$ 의 시작 시점에, 모든 공개키 집합은 PK^r 이고 시스템 상태는

$$S^r = \left\{ (i, a_i^{(r)}, \dots) : i \in PK^r \right\},$$

여기서 $a_i^{(r)}$ 는 공개키 i 가 보유한 금액이다. PK^r 는 S^r 로부터 유도될 수 있으며, S^r 은 각 공개키 i 에 대한 다른 구성 요소들도 포함할 수 있다.

라운드 0에서는 PK^0 가 초기 공개키 집합이고, S^0 는 초기 상태이다. PK^0 와 S^0 는 시스템 내에서 공통 지식으로 가정된다. 단순화를 위해, 라운드 r 시작 시점에 $PK^1, \dots, PK^r$ 와 $S^1, \dots, S^r$ 도 공통 지식으로 간주한다.

라운드 r 에서 시스템 상태는 S^r 에서 S^{r+1} 로 전환된다. 기호로는 다음과 같다.

$$\text{Round } r : S^r \longrightarrow S^{r+1}.$$

지불(Payments). Algorand에서 사용자는 계속해서 지불을 생성하고(2.7절에서 설명된 방식으로 전파된다), 그 형식과 의미는 이상적 시스템에서와 같다. 즉,

$$p = \text{SIG}_i(i, i', a, I, H(\mathcal{I})).$$

지불 p 는 라운드 r 에서 개별적으로 유효하다(라운드- r 지불이라 부름). 조건은 (1) 금액 a 가 $a_i^{(r)}$ 이하일 것, (2) p 가 어떤 $r' < r$ 의 공식 지불 집합 $PAY^{r'}$ 에 이미 포함되어 있지 않을 것이다. (두 번째 조건은 p 가 이미 효력을 발휘하지 않았음을 의미한다. 사용자 i 의 라운드- r 지불 집합은 금액 합이 $a_i^{(r)}$ 이하일 경우 집합적으로 유효하다.)

지불 집합(Paysets). 라운드- r 지불 집합 $\mathcal{P}$ 는 라운드- r 지불들의 집합으로, 각 사용자 i 의 지불들이 집합적으로 유효해야 한다. 모든 라운드- r 지불 집합의 집합을 $PAY(r)$ 라 한다. 라운드- r 지불 집합 $\mathcal{P}$ 는 그 상위 집합이 더 이상 라운드- r 지불 집합이 아닐 경우 *최대* (maximal)라고 한다.

우리는 실제로 지불 p 가 라운드 ρ 를 명시한다고 제안한다.

$$p = \text{SIG}_i(r, i, i', a, I, H(\mathcal{I})),$$

그리고 p 는 어떤 정수 $k \geq 0$ 에 대해 라운드 $[\rho, \rho + k]$ 내에서만 유효하다. ⁴

공식 지불 집합(Official Paysets). 매 라운드 r 마다 Algorand는 단일(또는 비어 있는) 지불 집합 PAY^r 을 공식적으로 선택한다. 이를 그 라운드의 공식 지불 집합이라 부른다. (본질적으로, PAY^r 는 “실제로” 발생한 라운드- r 지불을 나타낸다.)

이상적 시스템(및 비트코인)과 마찬가지로, (1) 새로운 사용자 j 가 시스템에 진입하는 유일한 방법은 어떤 라운드 r 의 공식 지불 집합 PAY^r 에 수취인으로 등장하는 것이다. (2) PAY^r 는 현재 상태 S^r 에서 다음 상태 S^{r+1} 을 결정한다. 즉,

$$PAY^r : S^r \longrightarrow S^{r+1}.$$

구체적으로,

1. 라운드 $r + 1$ 의 공개키 집합 PK^{r+1} 은 PK^r 와, PAY^r 에 처음 등장하는 모든 수취인 키들의 합집합이다.
2. 사용자 i 가 라운드 $r + 1$ 에서 가지는 금액 $a_i^{(r+1)}$ 은 이전 라운드에서 i 가 가진 금액 $a_i(r)$ (만약 $i \notin PK^r$ 이면 0)과 PAY^r 에 따라 i 에게 지불된 금액들의 합이다.

결국, 이상적 시스템에서와 같이 각 상태 S^{r+1} 은 이전 지불 이력으로부터 유도된다.

$$PAY^0, \dots, PAY^r.$$

2.4 블록과 증명된 블록 (Blocks and Proven Blocks)

Algorand에서 라운드 r 에 해당하는 블록 B^r 은 다음을 지정한다: 라운드 번호 r 자체, 라운드 r 의 지불 집합 PAY^r , 추후 설명할 수량 Q^r , 그리고 이전 블록의 해시 $H(B^{r-1})$.

따라서, 어떤 고정된 블록 B^0 에서 시작하면 전통적인 블록체인이 형성된다.

$$B^1 = (1, PAY^1, Q^0, H(B^0)), \quad B^2 = (2, PAY^2, Q^1, H(B^1)), \quad B^3 = (3, PAY^3, Q^2, H(B^2)), \dots$$

Algorand에서 블록의 진위성은 별도의 정보, 즉 “블록 인증서(block certificate)” $CERT$ 에 의해 보증된다. 이는 B^r 을 증명된 블록(proven block) $\overline{B}^r$ 로 만든다. 따라서 마법의 원장은 증명된 블록들의 연속으로 구현된다.

$$\overline{B}^1, \overline{B}^2, \dots$$

논의.

$CERT$ 는 $H(B^r)$ 에 대한 디지털 서명 집합으로 구성되며, 이는 SV^r 의 구성원 다수의 서명으로 이루어진다. 또한, 각각의 서명자가 실제로 SV^r 에 속함을 증명하는 자료도 함께 포함된다. 물론, 인증서 $CERT$ 를 블록 내부에 포함시킬 수도 있지만, 개념적으로 구분하는 편이 더 명확하다.

비트코인에서는 각 블록이 “암호 퍼즐의 해답”을 포함해야 하므로, 블록 생성이 계산 집약적이며 포크가 불가피하고 흔하다. 반면, Algorand의 블록체인은 두 가지 주요 장점을 가진다.

- 최소한의 계산으로 생성되며,
- 압도적으로 높은 확률로 포크되지 않는다.

따라서 각 블록 $\overline{B^i}$ 는 블록체인에 추가되는 즉시 안전하게 최종(final) 상태가 된다.

2.5 허용 가능한 실패 확률 (Acceptable Failure Probability)

Algorand의 보안을 분석하기 위해, 우리는 허용 가능한 실패 확률 F 를 지정한다. 이는 무언가 잘못될 가능성(예: 특정 라운드 r 의 검증자 집합 SV^r 이 정직한 다수를 가지지 못하는 경우)을 의미한다.

암호학적 해시 함수 H 의 출력 길이와 마찬가지로, F 또한 하나의 매개변수이다. 그러나 H 의 경우와 같이, F 를 구체적인 값으로 설정하는 것이 Algorand에서 충분한 보안성과 효율성을 동시에 확보할 수 있음을 직관적으로 이해하는 데 유용하다.

이를 강조하기 위해, 첫 번째 구현과 두 번째 구현에서 우리는 각각 F 를 다음과 같이 설정한다.

$$F = 10^{-12} \quad \text{및} \quad F = 10^{-18}.$$

논의.

10^{-12} 는 실제로 1조분의 1보다 작은 확률이며, 이러한 F 값 선택은 우리의 응용 환경에서 충분하다고 본다. 특히 강조해야 할 점은, 10^{-12} 가 “적대자가 정직한 사용자의 지분을 위조할 확률”을 의미하지 않는다는 것이다. 모든 지분은 디지털 서명되며, 적절한 디지털 서명이 사용되는 한 위조 확률은 10^{-12} 보다 훨씬 낮고 사실상 0에 가깝다.

우리가 F 확률로 허용하려는 “나쁜 사건”은 Algorand의 블록체인이 포크되는 것이다. 우리의 매개변수 설정에 따라, 라운드가 1분 길이 라면, Algorand 블록체인이 포크되는 사건은 약 190만 년에 한 번 꼴로 발생할 것으로 예상된다. 반대로, 비트코인에서는 포크가 꽤 자주 발생한다.

더 엄격한 보안을 원하는 사람은 F 를 더 낮게 설정할 수도 있다. 예를 들어, 두 번째 구현에서는 F 를 10^{-18} 로 설정하였다. 이는 블록이 1초 마다 생성된다고 가정할 때, 10^{18} 이 지금까지 우주가 존재한 초 수(빅뱅 이후 현재까지)로 추산된다는 점에서 직관적으로 이해할 수 있다. 따라서 $F = 10^{-18}$ 일 경우, 블록이 매초 생성되더라도, 블록체인 포크는 우주의 나이 동안 단 한 번 기대될 정도이다.

2.6 적대자 모델 (The Adversarial Model)

Algorand는 매우 적대적인 모델에서도 안전하도록 설계되었다. 이를 설명하자면 다음과 같다.

정직한 사용자와 악의적 사용자.

- 사용자가 정직(honest)하다 함은 모든 프로토콜 지침을 따르고, 메시지를 완벽히 송수신할 수 있다는 의미이다.
- 사용자가 악의적(malicious) 혹은 비잔틴(Byzantine)하다 함은, 주어진 지침에서 임의로 벗어날 수 있다는 의미이다.

적대자(Adversary).

적대자는 효율적인(기술적으로 다항 시간) 알고리즘으로, 설명을 위해 의인화된다. 그는 원하는 사용자를 원하는 시점에 즉시 악의적으로 만들 수 있다. (단, 부패시킬 수 있는 사용자 수에는 상한이 존재한다.)

적대자는 부패한 모든 사용자를 완벽히 통제하고, 그들을 완벽히 조율한다. 그는 그들의 모든 행동을 대리 수행하며, 메시지 송수신을 포함해 어떤 지침에서도 벗어나게 할 수 있다. 혹은 단순히 부패한 사용자를 고립시켜 메시지를 보내거나 받지 못하게 할 수도 있다. 단, 다른 이들은 사용자 i 가 악의적이라는 사실을 자동으로 알 수 없으며, i 의 악의성은 적대자가 취하게 하는 행동을 통해 드러날 수 있을 뿐이다.

이 강력한 적대자는 그러나 다음 제한을 가진다.

- 무한한 계산 능력을 가지지 않으며, 정직한 사용자의 디지털 서명을 위조할 수는 없다. (단, 확률적으로 무시할 수준의 예외는 존재한다.)

- 정직한 사용자 간의 메시지 교환에 어떤 방식으로든 간섭할 수는 없다.

더 나아가, 정직한 사용자를 공격할 수 있는 능력은 다음 가정 중 하나에 의해 제한된다.

정직 다수의 자금 가정 (Honesty Majority of Money, HMM).

우리는 HMM 가정의 연속체를 고려한다. 즉, 각 음이 아닌 정수 k 와 $h > 1/2$ 인 실수에 대해,

$HMM_{h,k} > h$: 모든 라운드 r 에서, $r - k$ 라운드 시점에 시스템 내 자금의 h 비율 이상이 정직한 사용자에게 속한다.

논의.

- 모든 악의적 사용자가 완벽히 조율된다고 가정하는 것은 매우 비관적인 가정이다. 현실적으로 너무 많은 개인 간의 완벽한 조율은 어렵다. 그러나 악의적 사용자들이 별도의 그룹으로만 조율될 수도 있다. 하지만 악의적 사용자들이 어느 수준의 조율을 할지 확신할 수 없으므로, 우리는 최악의 경우를 대비하는 편이 낫다.
- 적대자가 사용자를 비밀리에, 즉시, 동적으로 부패시킬 수 있다고 가정하는 것도 비관적이다. 현실적으로는 한 사용자를 완전히 장악하는 데 시간이 필요하다.
- 즉, 적대자가 어느 사용자든, 어떤 시점에든, 사전 통보 없이 장악할 수 있다고 가정하는 것은 매우 비관적이다.

그럼에도 불구하고 Algorand는 이러한 비관적 가정하에서도, 충분히 오랜 기간 동안 자금의 과반이 정직한 사용자에게 속해 있다면 안전하다. 특히, 매 라운드 r 에서 정직한 사용자가 시스템 자금의 절반 이상을 보유한다면 Algorand는 안전하다.

마지막으로, HMM 가정과 기존의 “계산력 정직 다수” 가정은 관련이 있다. 계산력은 돈으로 살 수 있으므로, 만약 악의적 사용자가 대부분의 자금을 보유한다면, 결국 대부분의 계산력을 확보할 수 있기 때문이다.

2.7 통신 모델 (The Communication Model)

우리는 메시지 전파, 즉 “피어-투-피어 가십(peer-to-peer gossip)”⁵ 만을 유일한 통신 수단으로 상정한다.

임시 가정: 네트워크 전체에서의 메시지의 적시 전달.

논문의 대부분에서는, 모든 전파된 메시지가 거의 모든 정직한 사용자에게 적시에 도달한다고 가정한다. 10장에서 우리는 자연적 혹은 적대적으로 유도된 네트워크 분할 상황을 다루며, 이 가정을 제거할 것이다. (거기서는 네트워크의 각 연결된 컴포넌트 내부에서만 메시지가 적시에 도달한다고 가정한다.)

메시지의 적시 전달을 구체적으로 표현하는 한 가지 방법은 다음과 같다.

도달 비율 $\rho > 95\%$ 와 메시지 크기 $\mu \in \mathbb{Z}_+$ 에 대해, $\lambda_{\rho,\mu}$ 가 존재하여, 만약 정직한 사용자가 시각 t 에 크기 μ 의 메시지 m 을 전파한다면, 시각 $t + \lambda_{\rho,\mu}$ 까지 정직한 사용자 중 적어도 ρ 비율이 m 을 수신한다.

그러나 위 성질만으로는 Algorand 프로토콜을 지원할 수 없다. 별도의 최신 블록체인을 획득하는 메커니즘을 가정하지 않고서는 불가능하다. 실제로 새로운 블록 B^r 을 구성하기 위해서는 적절한 검증자 집합이 라운드 r 의 메시지를 적시에 수신해야 할 뿐만 아니라, 이전 라운드 메시지들도 알고 있어야 한다. 그래야 B^{r-1} 및 모든 이전 블록을 파악하여 B^r 의 지분들이 유효한지 판별할 수 있기 때문이다.

이를 대신하여, 우리는 다음의 가정을 도입한다.

메시지 전파(MP) 가정:

모든 $\rho > 95\%$ 와 $\mu \in \mathbb{Z}_+$ 에 대해 $\lambda_{\rho,\mu}$ 가 존재하여, 모든 시각 t 와 $t - \lambda_{\rho,\mu}$ 이전에 정직한 사용자가 전파한 크기 μ 의 메시지 m 은, 시각 t 까지 정직한 사용자 중 적어도 ρ 비율이 수신한다.

프로토콜 Algorand'에서는 실제로 각 라운드 단계에서 소수의 사용자(즉, 그 단계의 검증자)들이 정해진 크기의 별도 메시지를 전파하도록 지시한다. 우리는 이 전파 완료에 필요한 시간을 상한으로 두어야 하며, 이를 위해 MP 가정을 다음과 같이 강화한다.

모든 $n, \rho > 95\%, \mu \in \mathbb{Z}_+$ 에 대해, $\lambda_{n,\rho,\mu}$ 가 존재하여, 모든 시각 t 와 $t - \lambda_{n,\rho,\mu}$ 이전에 정직한 사용자들이 각각 전파한 크기 μ 의 메시지 $m_1, \dots, m_n$ 은, 시각 t 까지 정직한 사용자 중 적어도 ρ 비율이 수신한다.

비고

- 위 가정은 의도적으로 단순화되어 있으며, 실제 논문에서 필요한 것보다 강하다.⁶
- 단순화를 위해, 우리는 $\rho = 1$ 로 두고, 이후 ρ 를 생략한다.
- 우리는 비관적으로 가정하기를, 적대자가 MP 가정을 위반하지 않는 한, 모든 메시지의 전달을 완전히 통제한다고 본다. 즉, 정직한 사용자가 눈치채지 못하는 사이에, 적대자는 어떤 메시지가 언제 누구에게 전달될지를 임의로 결정할 수 있고, 원하는 메시지는 임의

3 전통적 환경에서의 BA 프로토콜 BA^*

이미 강조했듯이, 비잔틴 합의는 Algorand의 핵심 요소이다. 실제로 이러한 BA 프로토콜의 사용 덕분에 Algorand는 포크의 영향을 받지 않는다. 그러나 우리의 강력한 적대자에 대해 안전하려면, Algorand는 *플레이어 교체 가능성(player-replaceability)* 제약을 만족하는 BA 프로토콜에 의존해야 한다. 또한 효율성을 위해, 해당 BA 프로토콜은 매우 효율적이어야 한다.

BA 프로토콜은 처음에 이상화된 통신 모델, 즉 *동기식 완전 네트워크(synchronous complete networks, SC networks)*에서 정의되었다. 이러한 모델은 BA 프로토콜의 설계와 분석을 단순하게 한다.

따라서 본 장에서는, 플레이어 교체 가능성 문제를 완전히 무시한 채 SC 네트워크에서 동작하는 새로운 BA 프로토콜 BA^* 를 소개한다. 프로토콜 BA^* 자체만으로도 가치가 있다. 실제로 지금까지 알려진 SC 네트워크를 위한 가장 효율적인 암호학적 BA 프로토콜이다. 이후 Algorand 프로토콜 내에서 BA^* 를 사용할 수 있도록 일부 수정하지만, 10장에서 Algorand' 내에서 BA^* 가 어떻게 사용되는지를 명확히 강조할 것이다.

우리는 먼저 BA^* 가 동작하는 모델과 비잔틴 합의의 개념을 상기한다.

3.1 동기식 완전 네트워크와 대응하는 적대자

SC 네트워크에서는 공통 시계가 존재하며, 정수 시각 $r = 1, 2, \dots$ 마다 틱이 발생한다.

각 짝수 시각 r 에, 각 플레이어 i 는 메시지 m_i^r (비어 있을 수도 있음)를 모든 플레이어 j 에게 동시에 보낸다. 여기에는 자기 자신도 포함된 다. 각 메시지 m_i^r , j 는 시각 $r + 1$ 에 송신자 i 의 신원과 함께 플레이어 j 에게 도착한다.

다시 말해, 통신 프로토콜에서 플레이어가 *정직(honest)*하다는 것은 모든 지침을 따른다는 의미이고, *악의적(malicious)*이라는 것은 그렇지 않음을 뜻한다. 모든 악의적 플레이어는 적대자에 의해 완벽히 통제되고 조율된다. 적대자는 특히 악의적 플레이어에게 도착하는 모든 메시지를 즉시 열람하며, 그들이 보낼 메시지를 선택한다.

적대자는 원하는 홀수 시각에 원하는 정직한 사용자를 즉시 악의적으로 만들 수 있다. 단, 이는 *이미 정직한 사용자가 보낸 메시지*에는 간섭하지 못한다. 해당 메시지는 정상적으로 전달된다.

또한 적대자는 추가적인 능력을 가진다. 각 짝수 라운드에서 정직한 플레이어가 보낸 메시지를 즉시 확인하고, 그 정보를 기반으로 같은 시각에 악의적 플레이어들이 보낼 메시지를 선택할 수 있다.

비고

- *적대자의 권한*. 위 환경은 매우 적대적이다. 실제 비잔틴 합의 문헌에는 이보다 약한 가정이 많다. 그러나 더 적대적인 모델도 존재한다. 예컨대, 적대자가 특정 정직한 플레이어 i 가 시각 r 에 보낸 메시지를 본 후, 네트워크에서 그것을 삭제하고, 즉시 i 를 부패시켜 악의적 플레이어로 만들며, 새 메시지를 선택해 정상적으로 배달하는 모델이다. 본 논문에서 상정하는 적대자의 권한은 이와 동일하다.
- *물리적 추상화*. SC 네트워크의 통신 모델은 물리적 모델을 추상화한 것이다. 여기서 각 플레이어 쌍 (i, j) 는 별도의 비공개 통신 라인 $l_{i,j}$ 로 연결된다. 즉, 다른 누구도 $l_{i,j}$ 에 메시지를 삽입하거나 간섭하거나 엿들 수 없다. 적대자가 $l_{i,j}$ 에 접근할 수 있는 유일한 방법은 i 나 j 를 부패시키는 것이다.
- *프라이버시와 인증*. SC 네트워크에서는 메시지 프라이버시와 인증이 가정으로 보장된다. 그러나 우리의 피어-투-피어 메시지 전파 모델에서는 인증이 디지털 서명으로 보장되며, 프라이버시는 존재하지 않는다. 따라서 BA^* 프로토콜을 우리의 환경에 적용하려면 각 메시지를 디지털 서명해야 한다(보낸 시점의 상태까지 식별하도록). 다행히도, 우리가 Algorand에서 사용하는 BA 프로토콜은 메시지 프라이버시를 필요로 하지 않는다.

3.2 비잔틴 합의의 개념

비잔틴 합의 개념은 Pease, Shostak, Lamport [31]에 의해 *이진 값*의 경우 처음 도입되었다. 그러나 곧 임의의 초기 값을 다루도록 확장되었다. (관련 연구로 Fischer [16]와 Chor & Dwork [10]의 설문 참조.) 여기서 BA 프로토콜이라 함은 임의의 값(arbitrary-value)에 대한 프로토콜을 뜻한다.

정의 3.1. 동기식 네트워크에서, 플레이어 집합이 공통 지식으로 주어진 n -플레이어 프로토콜 $\mathcal{P}$ 가 있다고 하자. t 는 양의 정수로, $n \geq 2t + 1$ 이다. $\mathcal{P}$ 가 임의 값(혹은 이진 값) (n, t) -비잔틴 합의 프로토콜이며, 건전성 $\sigma \in (0, 1)$ 을 가진다고 한다. 이는, 값 집합 V 가 특수 기호 $\perp$ 을 포함하지 않는 경우(이진 값의 경우 $V = \{0, 1\}$), 플레이어 중 최대 t 명이 악의적이고 각 플레이어 i 가 초기 값 $v_i \in V$ 를 가지고 시작하는 실행에서, 모든 정직한 플레이어 j 가 확률 1로 종료하며, 출력 값 $out_j \in V \cup \{\perp\}$ 를 산출하고, 확률이 적어도 σ 일 때 다음 조건 두 가지가 성립함을 의미한다.

1. **합의(Agreement):** 모든 정직한 플레이어 i 에 대해 $out_i = out$ 인 어떤 $out \in V \cup \{\perp\}$ 가 존재한다.
2. **일관성(Consistency):** 만약 모든 정직한 플레이어의 초기 값이 동일한 $v \in V$ 라면, $out = v$ 가 된다.

우리는 out 을 $\mathcal{P}$ 의 출력이라 부르고, 각 out_i 를 플레이어 i 의 출력이라 부른다.

3.3 BA 표기법 (The BA Notation)

우리의 BA 프로토콜에서 플레이어는 특정 단계에서 특정 메시지를 보낸 플레이어 수를 세어야 한다. 따라서, 플레이어 i 가 단계 s 에서 어떤 값 v 를 보낸 플레이어 수를 나타내기 위해,

$$\#_i^s(v)$$

(혹은 s 가 명확할 때는 단순히 $\#_i(v)$)를 사용한다. 이는 플레이어 i 가 단계 s 에서 값 v 를 수신한 플레이어 j 의 수를 의미한다.

플레이어 i 는 각 플레이어 j 로부터 정확히 하나의 메시지를 수신하므로, 플레이어 수가 n 일 때, 모든 i 와 s 에 대해

$$\sum_v \#_i^s(v) = n.$$

3.4 이진 BA 프로토콜 BBA^*

이 절에서는 새로운 이진 BA 프로토콜 BBA^* 를 소개한다. 이 프로토콜은 플레이어의 $2/3$ 이상이 정직하다는 가정하에 동작하며, 매우 빠르다. 즉, 악의적 플레이어들이 무엇을 하든 간에, 루프의 각 실행은 $1/3$ 의 확률로 합의에 도달한다.

각 플레이어는 고유성 속성을 만족하는 디지털 서명 체계의 공개키를 가진다. 이 프로토콜은 동기식 완전 네트워크에서 실행되므로, 플레이어 i 가 보낸 각 메시지를 서명할 필요는 없다.

다만, 디지털 서명은 3단계에서 충분히 공통된 난수 비트를 생성하는 데 사용된다. (Algorand에서는 다른 모든 메시지 인증에도 디지털 서명이 사용된다.)

이 프로토콜에는 최소한의 사전 준비가 필요하다: 즉, 플레이어의 키와 무관한 공통 난수 문자열 r . (Algorand에서는 실제로 r 대신 수량 Q^r 가 사용된다.)

프로토콜 BBA^* 는 3단계 루프 구조로 되어 있으며, 플레이어들은 반복적으로 볼리언 값을 교환한다. 서로 다른 플레이어들이 서로 다른 시점에 루프를 종료할 수 있다. 플레이어 i 가 루프를 종료할 때에는, 어떤 단계에서 특별한 값 0^* 또는 1^* 를 전파하여, 모든 플레이어에게 이후 단계에서 자신이 각각 0 또는 1을 보낸 것으로 간주하도록 지시한다. (즉, 플레이어 j 가 플레이어 i 로부터 마지막으로 받은 메시지가 비트 b 였다면, j 는 이후 i 로부터 메시지를 받지 못하더라도 i 가 b 를 보낸 것으로 간주한다.)

프로토콜은 카운터 γ 를 사용하며, 이는 루프가 몇 번 실행되었는지를 나타낸다. BBA^* 시작 시 $\gamma = 0$ 이다. (γ 를 전역 카운터로 생각할 수 있지만, 실제로는 각 플레이어가 루프를 실행할 때마다 개별적으로 증가시킨다.)

플레이어 수는 $n \geq 3t + 1$ 이고, 여기서 t 는 악의적 플레이어의 최대 수이다. 이진 문자열 x 는 그 이진 표현(앞에 0이 붙을 수 있음)이 나타내는 정수로 식별된다. $lsb(x)$ 는 x 의 최하위 비트(least significant bit)를 뜻한다.

프로토콜 BBA^*

(통신) 단계 1. [코인이 0에 고정된 단계]

각 플레이어 i 는 b_i 를 보낸다.

- 1.1. 만약 $\#_i^1(0) \geq 2t + 1$ 이면, i 는 $b_i = 0$ 으로 설정하고, 0^* 를 보내며, $out_i = 0$ 을 출력하고 종료한다.
- 1.2. 만약 $\#_i^1(1) \geq 2t + 1$ 이면, i 는 $b_i = 1$ 로 설정한다.
- 1.3. 그 외에는, i 는 $b_i = 0$ 으로 설정한다.

(통신) 단계 2. [코인이 1에 고정된 단계]

각 플레이어 i 는 b_i 를 보낸다.

- 2.1. 만약 $\#_i^2(1) \geq 2t + 1$ 이면, i 는 $b_i = 1$ 로 설정하고, 1^* 를 보내며, $out_i = 1$ 을 출력하고 종료한다.
- 2.2. 만약 $\#_i^2(0) \geq 2t + 1$ 이면, i 는 $b_i = 0$ 으로 설정한다.
- 2.3. 그 외에는, i 는 $b_i = 1$ 로 설정한다.

(통신) 단계 3. [코인이 실제로 뒤집히는 단계]

각 플레이어 i 는 b_i 와 $SIG_i(r, \gamma)$ 를 보낸다.

- 3.1. 만약 $\#_i^3(0) \geq 2t + 1$ 이면, i 는 $b_i = 0$ 으로 설정한다.
- 3.2. 만약 $\#_i^3(1) \geq 2t + 1$ 이면, i 는 $b_i = 1$ 으로 설정한다.
- 3.3. 그 외에는, $S_i = \{j \in N \mid j \text{가 이 단계 3에서 } i \text{에게 올바른 메시지를 보냈다}\}$ 라 하면,

$$b_i = c \triangleq \text{lsb}\left(\min_{j \in S_i} H(SIG_j(r, \gamma))\right)$$

로 설정하고, γ_i 를 1 증가시킨 후, 다시 단계 1로 돌아간다.

정리 3.1. $n \geq 3t + 1$ 일 때, BBA^* 는 건전성 1을 가지는 이진 (n, t) -BA 프로토콜이다.

정리 3.1의 증명은 [26]에 제시되어 있다. 우리의 환경에의 적용과 플레이어 교체 가능성 속성은 본 논문에서 새롭게 제시되는 것이다.

역사적 비교.

확률적 이진 BA 프로토콜은 Ben-Or가 비동기 환경에서 처음 제안했다 [7]. 프로토콜 BBA^* 는 Feldman과 Micali [15]의 이진 BA 프로토콜을 우리의 공개키 환경에 맞게 새롭게 적용한 것이다. Feldman과 Micali의 프로토콜은 기대 상수 시간 내에 동작하는 최초의 프로토콜이었다. 이는 플레이어들이 스스로 공통 코인(common coin)을 구현함으로써 가능했는데, 이 개념은 Rabin이 처음 제안했으며, 그는 이를 외부의 신뢰 기관을 통해 구현했다 [32].

3.5 단계적 합의(Graded Consensus)와 프로토콜 GC

임의의 값에 대해, 비잔틴 합의보다 훨씬 약한 합의 개념을 상기해 보자.

정의 3.2. 모든 플레이어 집합이 공통 지식으로 주어진 프로토콜 $\mathcal{P}$ 가 있다고 하자. 각 플레이어 i 는 임의의 초기 값 v_i' 를 개인적으로 알고 있다.

우리는 $\mathcal{P}$ 가 (n, t) -단계적 합의(graded consensus) 프로토콜이라고 부른다. 이는 n 명의 플레이어 중 최대 t 명이 악의적일 수 있는 모든 실행에서, 각 정직한 플레이어 i 가 (v_i, g_i) 라는 값-등급 쌍을 출력하며, $g_i \in \{0, 1, 2\}$ 이고, 다음 세 가지 조건을 만족하는 경우이다.

1. 모든 정직한 플레이어 i, j 에 대해 $|g_i - g_j| \leq 1$ 이다.
2. 모든 정직한 플레이어 i, j 에 대해 $g_i, g_j > 0$ 이면 $v_i = v_j$ 이다.
3. 어떤 값 v 에 대해 $v_1' = \dots = v_n' = v$ 라면, 모든 정직한 플레이어 i 는 $v_i = v, g_i = 2$ 를 출력한다.

역사적 주석.

단계적 합의 개념은 Feldman과 Micali [15]가 제안한 단계적 브로드캐스트(graded broadcast)에서 유도된 것이다. 이는 Dolev [12]가 제안하고 Turpin과 Coan [33]이 정교화한 십자군 합의(crusader agreement)의 개념을 강화한 것이다.⁸

[15]에서는 $n \geq 3t + 1$ 인 경우를 위한 3단계 (n, t) -단계적 브로드캐스트 프로토콜 gradecast를 제공했다. 이후 Katz와 Koo [19]는 $n \geq 2t + 1$ 인 경우를 위한 더 복잡한 (n, t) -단계적 브로드캐스트 프로토콜을 발견했다.

아래의 2단계 프로토콜 GC 는 gradecast의 마지막 두 단계를, 우리의 표기법으로 표현한 것이다. 이를 강조하기 위해, 그리고 4.1절의 Algorand' 프로토콜 단계와 일치시키기 위해, 이 단계를 각각 2와 3이라 명명한다.

프로토콜 GC

- 단계 2. 각 플레이어 i 는 자신의 v_i' 를 모든 플레이어에게 전송한다.
- 단계 3. 각 플레이어 i 는, $\#_i^2(x) \geq 2t + 1$ 인 문자열 x 가 존재할 때만 x 를 모든 플레이어에게 전송한다.

출력 결정. 각 플레이어 i 는 (v_i, g_i) 를 다음과 같이 계산하여 출력한다.

- 어떤 x 에 대해 $\#_i^3(x) \geq 2t + 1$ 이면, $v_i = x$ 이고 $g_i = 2$ 이다.
- 어떤 x 에 대해 $\#_i^3(x) \geq t + 1$ 이면, $v_i = x$ 이고 $g_i = 1$ 이다.
- 그렇지 않으면, $v_i = \perp$ 이고 $g_i = 0$ 이다.

정리 3.2. $n \geq 3t + 1$ 이면, GC 는 (n, t) -단계적 브로드캐스트 프로토콜이다.

이 정리의 증명은 [15]에 제시된 *gradecast* 프로토콜의 증명에서 바로 도출되므로 생략한다. ⁹

3.6 프로토콜 BA^*

이제 우리는 이진 BA 프로토콜 BBA^* 와 단계적 합의 프로토콜 GC 를 통해, 임의 값에 대한 BA 프로토콜 BA^* 를 설명한다. 아래에서 각 플레이어 i 의 초기 값은 v_i' 이다.

프로토콜 BA^*

- **단계 1 및 2.** 각 플레이어 i 는 입력 v_i' 로 GC 를 실행하여 쌍 (v_i, g_i) 를 계산한다.
- **단계 3 이후.** 각 플레이어 i 는 BBA^* 를 실행한다. 이때 초기 입력은 $g_i = 2$ 이면 0, 그렇지 않으면 1이다. 그리고 최종적으로 비트 out_i 를 계산한다.

출력 결정. 각 플레이어 i 는 $out_i = 0$ 이면 v_i 를 출력하고, 그 외에는 $\perp$ 을 출력한다.

정리 3.3. $n \geq 3t + 1$ 일 때, BA^* 는 건전성 1을 가지는 (n, t) -BA 프로토콜이다.

증명. 우리는 먼저 일관성(Consistency), 이후 합의(Agreement)를 증명한다.

- **일관성 증명.**
어떤 값 $v \in V$ 에 대해 모든 정직한 플레이어의 초기 값이 $v_i' = v$ 라고 가정하자. 단계적 합의의 성질 3에 따라, GC 실행 이후 모든 정직한 플레이어는 $(v, 2)$ 를 출력한다. 따라서 모든 정직한 플레이어는 BBA^* 의 초기 비트를 0으로 설정한다. 이로 인해, BBA^* 실행이 끝나면 모든 정직한 플레이어의 출력은 $out_i = 0$ 이 된다. 결과적으로 BA^* 에서 모든 정직한 플레이어 i 의 출력은 $v_i = v$ 이다.
- **합의 증명.**
 BBA^* 가 이진 BA 프로토콜이므로, 두 가지 경우가 존재한다.
(A) 모든 정직한 플레이어 i 가 $out_i = 1$ 을 출력한다.
(B) 모든 정직한 플레이어 i 가 $out_i = 0$ 을 출력한다.
경우 A에서는, 모든 정직한 플레이어가 BA^* 에서 $\perp$ 을 출력하므로 합의가 성립한다.
경우 B에서는, BBA^* 실행에서 적어도 하나의 정직한 플레이어의 초기 비트가 0이어야 한다. (만약 모든 정직한 플레이어의 초기 비트가 1이었다면, 이진 BA 프로토콜의 일관성 성질에 의해 모든 출력이 1이 되었을 것이다.) 따라서 GC 실행 결과, 어떤 정직한 플레이어 i 가 $(v, 2)$ 를 출력한다. 성질 1에 의해 모든 정직한 플레이어 j 는 $g_j > 0$ 을 가지며, 성질 2에 의해 모든 정직한 플레이어 j 는 동일한 값 $v_j = v$ 를 출력한다. 따라서 BA^* 종료 시 모든 정직한 플레이어는 동일한 값 v 를 출력하게 된다. 결과적으로, 경우 B에서도 합의가 성립한다.

일관성과 합의가 모두 성립하므로, BA^* 는 임의 값에 대한 BA 프로토콜이다.

역사적 주석.

Turpin과 Coan은 $n \geq 3t + 1$ 일 때, 모든 이진 (n, t) -BA 프로토콜이 임의 값 (n, t) -BA 프로토콜로 변환될 수 있음을 처음 보였다.

단계적 합의를 통한 임의 값 비잔틴 합의로의 환원은 더 모듈화되고 명확하며, Algorand 프로토콜 *Algorand*'의 분석을 단순화한다.

Algorand에서 사용하기 위한 BA^* 의 일반화.

Algorand는 모든 통신이 가십을 통해 이루어져도 동작한다. 그러나 우리는 기존 연구와 비교를 용이하게 하고 이해를 돕기 위해, 익숙한 전통적 통신 네트워크 환경에서 BA^* 를 제시했다. 실제로 BA^* 는 가십 네트워크에서도 동작한다.

또한, 우리는 BA^* 가 Algorand가 상정하는 매우 적대적인 모델에서 안전하기 위해 필수적인 *플레이어 교체 가능성(player-replaceability)* 성질을 만족함을 지적한다.

따라서 가십 통신 네트워크에서 플레이어 교체 가능한 어떤 BA 프로토콜도 Algorand 내에서 안전하게 사용될 수 있다. 특히, Micali와 Vaikunthanathan은 단순 정직 다수만으로도 매우 효율적으로 동작하는 BA^* 의 확장판을 제시했다. 이 프로토콜 또한 Algorand에서 사용할 수 있다.

4 Algorand의 두 구현 (Two Embodiments of Algorand)

앞서 논의했듯이, 높은 수준에서 Algorand의 한 라운드는 다음과 같이 이상적으로 진행된다.

첫째, 무작위로 선택된 사용자(리더)가 새로운 블록을 제안하고 이를 네트워크에 전파한다. (이 과정에는 잠재적 리더들을 먼저 선정하고, 그 중 상당 비율에서 하나의 공통 리더가 등장하도록 하는 과정이 포함된다.)

둘째, 무작위로 선택된 위원회(committee)가 리더가 제안한 블록에 대해 비잔틴 합의를 이룬다. (이 과정에서는 BA 프로토콜의 각 단계를 별도로 선정된 위원회가 실행한다.)

합의된 블록은 일정 임계치(T_H) 이상의 위원 서명으로 인증된다. 이 서명들은 전파되어, 모두가 해당 블록이 새로운 블록임을 확인할 수 있게 한다. (이 과정에는 서명자의 자격 증명 전파와, 블록 자체 대신 해시에 대한 인증이 포함되며, 이를 통해 일단 블록 해시가 명확해지면 누구나 해당 블록을 확인할 수 있게 된다.)

다음 두 절에서 우리는 두 가지 구현, $Algorand'_1$ 과 $Algorand'_2$ 를 제시한다. 두 구현은 모두 정직한 사용자 다수 가정 하에서 동작한다. 8 장에서는 이 구현들을 정직한 자금 다수 가정 하에서 동작하도록 변형하는 방법을 보인다.

- $Algorand'_1$ 은 위원회의 $2/3$ 이상이 정직하다고 가정한다. 또한 $Algorand'_1$ 에서는 BA 합의에 필요한 단계 수를 충분히 큰 상한으로 제한하여, 일정한 수의 단계 내에서 압도적으로 높은 확률로 합의가 보장된다. (그러나 $Algorand'_2$ 보다 시간이 오래 걸릴 수 있다.) 만약 마지막 단계까지 합의에 도달하지 못하면, 위원회는 항상 유효한 빈 블록(empty block)에 합의한다.
- $Algorand'_2$ 는 위원회의 정직한 구성원이 항상 임계치 t_H 이상이라고 가정한다. (이 임계치는 위원회 구성원의 $2/3$ 이상이 정직한 확률이 압도적으로 높도록 보장한다.) 또한 $Algorand'_2$ 에서는 BA 합의가 단계 수에 제한 없이 진행될 수 있다. (따라서 $Algorand'_1$ 보다 짧은 시간에 끝낼 수 있다.)

이 기본 구현들을 기반으로 다양한 변형을 쉽게 파생시킬 수 있다. 예를 들어, $Algorand'_2$ 를 기반으로 $Algorand'_1$ 의 단계 제한을 제거하는 변형을 만들 수 있다.

두 구현은 모두 공통적인 핵심, 표기법, 개념, 매개변수를 공유한다.

4.1 공통 핵심 (A Common Core)

목표(Objectives). 이상적으로, 각 라운드 r 에 대해 Algorand는 다음 속성을 만족해야 한다.

1. **완전한 정확성(Perfect Correctness).** 모든 정직한 사용자는 동일한 블록 B^r 에 합의한다.
2. **완전성 조건 1(Completeness 1).** 확률 1로, 블록 B^r 의 지불 집합 PAY^r 는 최대이다.¹⁰

물론, 완전한 정확성만 보장하는 것은 간단하다. 예를 들어, 모든 사용자가 항상 공식 지불 집합 PAY^r 를 빈 집합으로 선택하면 된다. 그러나 이 경우 완전성은 0이 된다. 안타깝게도, 악의적 사용자가 존재하는 상황에서 정확성과 완전성을 모두 보장하는 것은 쉽지 않다.

따라서 Algorand는 보다 현실적인 목표를 채택한다. 정직한 사용자 비율을 h 라 하고, $h > 2/3$ 이라고 하자. Algorand의 목표는 다음과 같다.

압도적으로 높은 확률로, 완전한 정확성과 h 에 근접한 완전성을 동시에 보장하는 것.

여기서 정확성을 완전성보다 우선시하는 것은 합리적인 선택이다. 지불이 한 라운드에서 처리되지 않더라도 다음 라운드에서 처리할 수 있다. 그러나 포크는 가능하다면 반드시 피해야 한다.

리드된 비잔틴 합의 (Led Byzantine Agreement).

완전한 정확성은 다음 방식으로 보장될 수 있다. 라운드 r 시작 시, 각 사용자 i 는 자신의 후보 블록 B_i^r 을 구성하고, 모든 사용자가 이 후보 블록들 중 하나에 대해 비잔틴 합의를 수행한다.

도입부에서 언급했듯이, 이 BA 프로토콜은 정직 사용자 비율이 $2/3$ 이상일 것을 요구하며, *플레이어 교체 가능성*을 가진다. 즉, 각 단계는 무작위로 선택된 소규모 검증자(verifiers) 집합에 의해 실행될 수 있고, 이들은 내부 변수를 공유하지 않는다.

그러나 이 접근은 완전성을 보장하지 못한다. 정직한 사용자들의 후보 블록은 서로 완전히 다를 가능성이 높기 때문이다. 결국 합의된 블록은 항상 최대 지불 집합을 포함하지 않을 수 있다. 실제로 기본값인 빈 블록 $B_\emptyset$ — 즉, 지불 집합이 비어 있는 블록 —으로 귀결될 수 있다.

Algorand'는 이 완전성 문제를 다음 방식으로 피한다. 먼저 라운드 r 의 리더 ℓ^r 을 선택한다. 그 후 ℓ^r 이 자신의 후보 블록 $B_{\ell^r}^r$ 을 전파한다. 마지막으로 모든 사용자는 ℓ^r 로부터 실제 수신한 블록에 합의한다. ℓ^r 이 정직하다면, 완전한 정확성과 완전성이 동시에 성립한다. 따라서 Algorand'는 ℓ^r 이 압도적으로 높은 확률로 정직하도록 보장한다. (리더가 악의적일 경우, 합의된 블록이 빈 블록이 되더라도 상관없다. 결국 악의적 리더는 언제든지 빈 블록 $B_\emptyset$ 을 선택해 전파할 수 있으며, 그 경우 정직한 사용자들은 빈 블록에 합의할 수밖에 없다.)

리더 선택 (Leader Selection).

Algorand에서 r 번째 블록은 $B^r = (r, PAY^r, Q^{r-1}, H(B^{r-1}))$ 형태를 가지며, PAY^r 는 최대 지불 집합이다. (이후 본 절에서 그 이유에 대한 직관을 제공한다.)

라운드 r 시작 시, 모든 사용자는 지금까지의 블록체인 $B^0, \dots, B^{r-1}$ 을 알고 있으며, 이를 통해 모든 이전 라운드의 사용자 집합 $PK^1, \dots, PK^{r-1}$ 을 유도한다. 라운드 r 의 잠재 리더(potential leader)는 다음 조건을 만족하는 사용자 i 이다.

$$H(SIG_i(r, 1, Q^{r-1})) \leq p.$$

설명하자면, Q^{r-1} 이 블록 B^{r-1} 의 일부이므로, 고유성 서명 체계의 성질에 의해 $SIG_i(r, 1, Q^{r-1})$ 는 i 와 r 에 대해 고유한 이진 문자열이다. H 는 랜덤 오라클이므로, $H(SIG_i(r, 1, Q^{r-1}))$ 는 i 와 r 에 대해 고유하게 연관된 무작위 256비트 문자열이다. “.” 표시는 이진 소수점을 뜻하며, $r_i \triangleq H(SIG_i(r, 1, Q^{r-1}))$ 는 0과 1 사이의 무작위 256비트 수의 이진 전개이다. 따라서 $r_i \leq p$ 확률은 본질적으로 p 이다. (이 잠재 리더 선택 방식은 Micali와 Rivest [28]의 마이크로 지불 체계에서 영감을 받았다.)

p 는 압도적으로 높은 확률(즉, $1 - F$ 이상)로 적어도 하나의 정직한 잠재 리더가 존재하도록 선택된다. (실제로 p 는 그러한 확률을 만족하는 최소값으로 정한다.)

사용자 i 만이 자신의 서명을 계산할 수 있으므로, 자신이 라운드 r 의 잠재 리더인지 아는 사람은 i 뿐이다. 그러나 자격 증명 $\sigma_i^r \triangleq SIG_i(r, 1, Q^{r-1})$ 을 공개함으로써, i 는 누구에게든 자신이 잠재 리더임을 증명할 수 있다.

리더 ℓ^r 는 잠재 리더들 중에서, 그 자격 증명의 해시 값이 가장 작은 사용자로 정의된다. 즉,

$$H(\sigma_{\ell^r}^r) \leq H(\sigma_j^r) \quad \forall j.$$

단, 악의적 ℓ^r 이 자신의 자격 증명을 공개하지 않을 수도 있으므로, 라운드 r 의 실제 리더가 누구인지 영원히 알려지지 않을 수도 있다. 그러나 거의 항상(동률이 일어나지 않는 한) ℓ^r 은 유일하다.

마지막으로 중요한 세부사항은, 사용자 i 가 라운드 r 의 잠재 리더(혹은 리더)가 되려면 최소 k 라운드 이상 시스템에 존재했어야 한다는 것이다. 이는 Q^r 및 모든 미래의 Q 값이 조작 불가능하도록 보장한다. 실제로 Q^r 는 잠재 리더 중 하나에 의해 결정된다.

검증자 선택 (Verifier Selection).

라운드 r 의 각 단계 $s > 1$ 은 소규모 검증자 집합 $SV^{r,s}$ 에 의해 실행된다. 마찬가지로, $SV^{r,s}$ 에 속하는 검증자 i 는 $r - k$ 시점에 이미 존재한 사용자 집합 PK^{r-k} 에서 무작위로 선택되며, 수량 Q^{r-1} 을 기반으로 한다. 구체적으로, $i \in PK^{r-k}$ 가 $SV^{r,s}$ 에 속하기 위한 조건은

$$H(SIG_i(r, s, Q^{r-1})) \leq p'.$$

다시 말해, i 자신만이 $SV^{r,s}$ 에 속하는지 여부를 알 수 있다. 그러나 그렇다면, 자신의 자격 증명 $\sigma_i^{r,s} \triangleq H(SIG_i(r, s, Q^{r-1}))$ 을 공개함으로써 이를 증명할 수 있다. 검증자 i 가 단계 s 에서 보낸 메시지 $m_i^{r,s}$ 에는 자격 증명 $\sigma_i^{r,s}$ 가 포함되어, 다음 단계 검증자들이 $m_i^{r,s}$ 가 정당한 단계- s 메시지임을 인식할 수 있다.

p' 는 $SV^{r,s}$ 에 대해, 정직 사용자 수 $\#good$ 과 악의적 사용자 수 $\#bad$ 가 압도적으로 높은 확률로 다음 조건을 만족하도록 선택된다.

- Algorand'₁의 경우:
 1. $\#good > 2 \cdot \#bad$
 2. $\#good + 4 \cdot \#bad < 2n$ (단, n 은 $SV^{r,s}$ 의 기대 크기)
- Algorand'₂의 경우:
 1. $\#good > t_H$
 2. $\#good + 2 \cdot \#bad < 2t_H$ (단, t_H 는 특정 임계치)

이 조건은 압도적으로 높은 확률로 다음을 보장한다.

- (a) BA 프로토콜의 마지막 단계에서 충분히 많은 정직 플레이어가 새로운 블록 B^r 에 서명한다.
- (b) 라운드당 단 하나의 블록만 필요한 서명 수를 충족한다.
- (c) BA 프로토콜의 각 단계에서 $2/3$ 이상의 정직 다수가 보장된다.

블록 생성 과정 설명 (Clarifying Block Generation).

라운드 r 의 리더 ℓ^r 이 정직하다면, 생성된 블록은

$$B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$$

형태를 가지며, PAY^r 는 최대 지불 집합이다. (모든 지불 집합은 집합적으로 유효하다는 점을 상기하라.)

반면 ℓ^r 이 악의적이라면, B^r 은 두 가지 형태 중 하나일 수 있다.

$$B^r = (r, PAY^r, SIG_i(Q^{r-1}), H(B^{r-1})) \quad \text{혹은} \quad B^r_{\emptyset} = (r, \emptyset, Q^{r-1}, H(B^{r-1})).$$

첫 번째 형태에서, PAY^r 는 반드시 최대가 아닐 수 있으며, 심지어 $\emptyset$ 일 수도 있다. 여기서 i 는 라운드 r 의 잠재 리더일 수 있다. (단, 반드시 리더 ℓ^r 일 필요는 없다. ℓ^r 이 자신의 자격 증명을 비공개로 유지한다면 이런 상황이 발생할 수 있다.)

두 번째 형태는, 라운드 r 의 BA 프로토콜 실행 결과 모든 정직한 사용자가 기본값 $\perp$ 을 출력했을 때 발생한다. 이 경우 출력 블록은 빈 블록 $B^r_{\emptyset}$ 이다.

따라서 두 경우 모두 지불 집합이 비어 있더라도, $B^r = (r, \emptyset, SIG_i(Q^{r-1}), H(B^{r-1}))$ 와 $B^r_{\emptyset} = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$ 는 구문적으로는 다른 블록이며, 각각 다른 상황에서 발생한다. 전자는 “BA 프로토콜 실행이 정상적으로 진행된 경우”, 후자는 “BA 프로토콜 실행이 실패하여 기본값이 출력된 경우”이다.

시드 Q^r 와 회고 파라미터 k .

이상적으로는 각 Q^r 이 무작위로 독립적으로 선택되어야 한다. 그러나 실제로는 적대자에 의해 조작될 수 없어야 한다는 조건만으로도 충분하다.

첫눈에, Q^{r-1} 을 $H(PAY^{r-1})$ 로 정의해, 블록 B^{r-1} 에 명시하지 않고 유도할 수 있을 것처럼 보인다. 그러나 간단한 분석으로, 악의적 사용자가 이 방식을 이용해 다음 라운드 리더를 조작할 수 있음을 알 수 있다.¹¹ 더 나아가 전통적인 블록 속성에 기반한 여러 대안들도 쉽게 공격당할 수 있다.

따라서 우리는 새로운 Q^r 을 다음과 같이 구체적으로 정의한다.

$$Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r) \quad (\text{단, } B^r \text{이 빈 블록이 아닐 경우})$$

$$Q^r \triangleq H(Q^{r-1}, r) \quad (\text{그 외의 경우}).$$

이러한 정의는, 리더가 정직한 경우 Q^r 이 사실상 무작위 함수의 출력처럼 동작하도록 보장한다. 리더가 악의적일 경우, Q^r 은 여러 대안을 가질 수 있으나, 우리의 분석(4.1 직관 섹션)에서는 이러한 상황에서도 정직한 리더가 선택될 확률이 충분히 유지됨을 보인다.

일시적 키 (Ephemeral Keys).

우리의 프로토콜 실행은 무시할 수 있을 정도의 확률을 제외하면 포크를 생성하지 않는다. 그러나 적대자가 r 번째 블록이 생성된 직후, 라운드 r 의 검증자들을 모두 부패시켜 새로운 블록 B^r 을 만들게 할 수 있다. 이 경우 가짜 블록은 진짜 블록 이후에 전파되므로, 주의 깊게 관찰하는 사용자는 속지 않는다.¹² 그러나 여전히 B^r 은 문법적으로 올바른 블록일 수 있으며, 우리는 이런 위조를 원천적으로 방지하고자 한다.

이를 위해 새로운 규칙을 도입한다. 라운드 r 단계 s 의 검증자 집합 $SV^{r,s}$ 의 구성원들은 일시적 공개키 $pk_i^{r,s}$ 를 사용하여 메시지를 디지털 서명한다. 이 키는 일회용이며, 대응하는 비밀키 $sk_i^{r,s}$ 는 사용 즉시 폐기된다. 따라서 검증자가 이후에 부패하더라도, 적대자는 그가 원래 서명하지 않았던 메시지에 강제로 서명시킬 수 없다.

물론, 적대자가 새로운 키 $pk_i^{r,s}$ 를 조작해 정직한 사용자가 이것이 i 의 진짜 일시적 키라고 믿도록 만드는 일이 불가능해야 한다. 따라서 키 발급검증 방식은 그러한 위조가 불가능하도록 설계된다.

4.2 공통 표기법, 개념, 매개변수 요약

표기법 (Notations)

- $r \geq 0$: 현재 라운드 번호.
- $s \geq 1$: 라운드 r 의 현재 단계 번호.
- B^r : 라운드 r 에서 생성된 블록.
- PK^r : 라운드 $r - 1$ 종료 시점 및 라운드 r 시작 시점의 공개키 집합.
- S^r : 라운드 $r - 1$ 종료 시점 및 라운드 r 시작 시점의 시스템 상태.¹³
- PAY^r : B^r 에 포함된 지불 집합.
- ℓ^r : 라운드 r 의 리더. ℓ^r 은 라운드 r 의 지불 집합 PAY^r 을 선택하고, 다음 Q^r 를 결정한다.
- Q^r : 라운드 r 의 시드(seed). 이 값은 라운드 r 종료 시 생성되는 이진 문자열이며, 라운드 $r + 1$ 의 검증자 선정에 사용된다. Q^r 는 블록 내 지불 집합과 독립적이며, ℓ^r 이 조작할 수 없다.
- $SV^{r,s}$: 라운드 r 의 단계 s 에 선택된 검증자 집합.
- SV^r : 라운드 r 의 전체 검증자 집합. $SV^r = \bigcup_{s \geq 1} SV^{r,s}$.
- $MSV^{r,s}, HSV^{r,s}$: 각각 $SV^{r,s}$ 의 악의적 검증자 집합과 정직한 검증자 집합. $MSV^{r,s} \cup HSV^{r,s} = SV^{r,s}$ 이고, $MSV^{r,s} \cap HSV^{r,s} = \emptyset$.
- $n_1 \in \mathbb{Z}^+, n \in \mathbb{Z}^+$: 각각 $SV^{r,1}$ 의 잠재 리더 기대 수, 그리고 $s > 1$ 인 각 $SV^{r,s}$ 의 검증자 기대 수. $n_1 < n$ 임에 유의하라. 이는 $SV^{r,1}$ 에서는 최소 하나의 정직한 사용자가 필요하지만, $s > 1$ 인 단계의 $SV^{r,s}$ 에서는 정직한 다수가 필요하기 때문이다.
- $h \in (0, 1)$: $2/3$ 보다 큰 상수. 이는 시스템 내 정직성 비율(honesty ratio)로, 각 PK^r 에서 정직한 사용자(혹은 정직한 자금)의 비율이 최소 h 임을 의미한다.
- H : 랜덤 오라클로 모델링되는 암호학적 해시 함수.
- $\perp$: H 의 출력과 동일한 길이를 가진 특수 문자열.
- $F \in (0, 1)$: 허용되는 오류 확률을 지정하는 매개변수. 확률이 F 이하이면 “무시할 만하다(negligible)”고 하고, $1 - F$ 이상이면 “압도적(overwhelming)”이라고 한다.
- $p_h \in (0, 1)$: 라운드 r 의 리더 ℓ^r 가 정직할 확률. 이상적으로는 $p_h = h$ 이다. 그러나 적대자가 존재하면 실제 p_h 값은 분석을 통해 결정된다.
- $k \in \mathbb{Z}^+$: 회고 파라미터(look-back parameter). 즉, 라운드 r 의 검증자는 PK^{r-k} 에서 선택된다. 형식적으로 $SV^r \subseteq PK^{r-k}$.¹⁴
- $p_1 \in (0, 1)$: 라운드 r 의 첫 단계에서, PK^{r-k} 의 사용자가 $SV^{r,1}$ 에 포함될 확률. $p_1 \triangleq \frac{n_1}{|PK^{r-k}|}$.
- $p \in (0, 1)$: 라운드 r 의 각 단계 $s > 1$ 에서, PK^{r-k} 의 사용자가 $SV^{r,s}$ 에 포함될 확률. $p \triangleq \frac{n}{|PK^{r-k}|}$.
- $CERT^r$: B^r 에 대한 인증서. 이는 라운드 r 의 적절한 검증자 t_H 명이 $H(B^r)$ 에 서명한 집합이다.
- $\overline{B^r} = (B^r, CERT^r)$: 증명된 블록(proven block).
- 사용자 i 는 증명된 블록의 두 부분을 모두 가지고 있고 유효성을 확인했을 때 B^r 을 “안다(knows)”고 한다. 서로 다른 사용자가 보는 $CERT^r$ 는 서로 다를 수 있다.
- $\alpha_i^{r,s}, \beta_i^{r,s}$: 각각 사용자 i 가 라운드 r 의 단계 s 를 시작하고 종료하는 (지역) 시각.
- Λ, λ : 각각 Algorand 프로토콜의 단계 1 실행 시간 상한, 그리고 단계 $s > 1$ 실행 시간 상한.

- Λ : 하나의 1MB 블록 전파 시간 상한. (기호로 $\Lambda = \lambda_{1,1,1MB}$.) 우리는 $\rho = 1$ 로 두므로, 블록 크기는 최대 1MB라 가정하며 $\Lambda = \lambda_{1,1,1MB}$ 이다.
- λ : 단계 $s > 1$ 에서 검증자당 작은 메시지(200B)를 전파하는 시간 상한. (비트코인과 마찬가지로 타원곡선 서명을 사용하면, 검증자 메시지는 200B이다. 따라서 $\lambda = \lambda_{n,1,200B}$ 이다.)
- 우리는 $\Lambda = O(\lambda)$ 라고 가정한다.

개념 (Notions)

• 검증자 선택 (Verifier selection).

각 라운드 r 와 단계 $s > 1$ 에 대해,

$$SV^{r,s} \triangleq \{i \in PK^{r-k} : H(SIG_i(r, s, Q^{r-1})) \leq p\}.$$

각 사용자 $i \in PK^{r-k}$ 는 자신의 장기 키로 서명을 계산해 $i \in SV^{r,s}$ 여부를 비공개로 판단한다. $i \in SV^{r,s}$ 이면, (r, s) 자격 증명 $\sigma_i^{r,s} = SIG_i(r, s, Q^{r-1})$ 를 갖는다.

첫 단계 $SV^{r,1}$ 과 $\sigma_i^{r,1}$ 도 동일하게 정의되며, 단 p 대신 p_1 이 사용된다. $SV^{r,1}$ 에 속하는 검증자는 잠재 리더(potential leaders)이다.

• 리더 선택 (Leader selection).

$i \in SV^{r,1}$ 인 사용자가, 자신의 자격 증명 해시 값이 모든 다른 잠재 리더의 해시 값보다 작을 경우, 라운드 r 의 리더 ℓ^r 이 된다. 해시 값이 동일한 경우(매우 드뭄), 타이브레이크는 잠재 리더의 공개키를 사전식으로 비교해 결정한다.

해시 값이 무작위 균등 분포이므로, $SV^{r,1} \neq \emptyset$ 인 경우 ℓ^r 은 항상 존재하며, 최소 h 의 확률로 정직하다. n_1 은 압도적으로 높은 확률로 $SV^{r,1}$ 이 공집합이 아니도록 충분히 크게 선택된다.

• 블록 구조 (Block structure).

비어 있지 않은 블록: $B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$.

빈 블록: $B_\emptyset^r = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$.

비어 있지 않은 블록이라도 $PAY^r = \emptyset$ 일 수 있다. (해당 라운드에 지불이 없거나 리더가 악의적일 경우.) 그러나 비어 있지 않은 블록이라면, ℓ^r 의 신원, 자격 증명 $\sigma_{\ell^r}^{r,1}$, 그리고 $SIG_{\ell^r}(Q^{r-1})$ 가 적시에 공개되었다는 것을 의미한다. 리더가 정직하다면 블록은 압도적으로 높은 확률로 비어 있지 않다.

• 시드 Q^r .

B^r 이 비어 있지 않으면 $Q^r = H(SIG_{\ell^r}(Q^{r-1}), r)$.

B^r 이 빈 블록이면 $Q^r = H(Q^{r-1}, r)$.

매개변수 (Parameters)

• 매개변수 간 관계.

– 라운드 r 의 리더 및 검증자는 PK^{r-k} 에서 선택된다. k 는 적대자가 라운드 $r - k - 1$ 시점에서 Q^{r-1} 을 F 이상의 확률로 예측할 수 없도록 충분히 크게 선택된다. 그렇지 않으면 적대자는 라운드 $r - k$ 에 악의적 사용자를 추가해, 라운드 r 의 리더나 $SV^{r,s}$ 를 악의적 다수로 만들 수 있다.

– 각 라운드 r 의 단계 1에 대해, n_1 은 $SV^{r,1} \neq \emptyset$ 이 압도적으로 높은 확률로 성립하도록 선택된다.

• 중요 매개변수의 예시.

– H 의 출력 길이는 256비트.

– $h = 80\%$, $n_1 = 35$.

– $\Lambda = 1$ 분, $\lambda = 10$ 초.

• 프로토콜 초기화.

프로토콜은 시간 0에 $r = 0$ 으로 시작한다. B^{-1} 이나 $CERT^{-1}$ 은 존재하지 않으므로, 구문적으로 B^{-1} 은 공개 매개변수로 간주된다. B^{-1} 의 세 번째 구성 요소는 Q^{-1} 을 지정하며, 모든 사용자는 시간 0에 B^{-1} 을 알고 있다.

5 Algorand'

본 장에서는 Algorand'의 한 버전을 구성한다. 이 버전은 다음 가정을 바탕으로 동작한다.

정직 사용자 다수 가정 (Honest Majority of Users Assumption):

각 라운드 r 의 PK^r 에서 사용자의 $2/3$ 이상이 정직하다.

8장에서 우리는 위 가정을, 더 의미 있는 정직 자금 다수 가정(Honest Majority of Money Assumption)으로 대체하는 방법을 보인다.

5.1 추가 표기법 및 매개변수 (Additional Notations and Parameters)

표기법

- $m \in \mathbb{Z}^+$: 이진 BA 프로토콜의 최대 단계 수. m 은 3의 배수이다.
- $L' \leq m/3$: 베르누이 시행에서 1을 관측하는 데 필요한 시행 횟수를 나타내는 확률 변수. 각 시행은 확률 $2/3$ 로 1이 된다. 최대 $m/3$ 회 시행할 수 있으며, 모든 시행이 실패하면 $L' \triangleq m/3$ 으로 정의한다. L' 은 블록 B^r 을 생성하는 데 필요한 시간을 상한하는데 사용된다.
- $t_H = \frac{2n}{3} + 1$: 프로토콜 종료 조건에서 필요한 서명 수.
- $CERT^r$: B^r 에 대한 인증서. 이는 라운드 r 의 적절한 검증자들이 $H(B^r)$ 에 대해 남긴 t_H 개의 서명 집합이다.

매개변수

- 매개변수 간 관계.
 - 라운드 r 의 각 단계 $s > 1$ 에서, n 은 압도적으로 높은 확률로 $|HSV^{r,s}| \geq 2|MSV^{r,s}|$ 이고 $|HSV^{r,s}| + 4|MSV^{r,s}| < 2n$ 이 되도록 선택된다.
 - h 값이 1에 가까울수록 n 은 더 작아도 된다. 우리는 체르노프 경계(Chernoff bound)의 변형을 사용해 이 조건이 압도적으로 높은 확률로 성립함을 보장한다.
 - m 은 $L' < m/3$ 이 압도적으로 높은 확률로 성립하도록 선택된다.
- 중요 매개변수의 예시.
 - $F = 10^{-12}$.
 - $n \approx 1500$, $k = 40$, $m = 180$.

5.2 Algorand'에서의 일시적 키 구현

앞서 언급했듯이, 우리는 검증자 $i \in SV^{r,s}$ 가 라운드 r 의 단계 s 에서 자신의 메시지 $m_i^{r,s}$ 를 일시적 공개키 $pk_i^{r,s}$ 에 대응하는 일시적 비밀키 $sk_i^{r,s}$ 로 서명하도록 하고, 사용 직후 이 비밀키를 파괴하도록 하고자 한다. 따라서 모든 사용자가 $pk_i^{r,s}$ 가 실제로 i 의 올바른 공개키임을 검증할 수 있는 효율적인 방법이 필요하다. 이를 위해 우리는, 우리가 알기로는 처음으로, ID 기반 서명 체계(identity-based signature scheme)를 새로운 방식으로 활용한다.

고수준에서, 이러한 체계는 중앙 권한 A 가 공통 마스터 공개키 PMK 와 대응하는 마스터 비밀키 SMK 를 생성하는 것으로 시작한다. 어떤 플레이어 U 의 신원(identity)이 주어지면, A 는 SMK 를 사용해 U 의 공개키에 대응하는 비밀 서명 키 sk_U 를 계산하고, 이를 U 에게 비밀리에 제공한다. (실제로 ID 기반 디지털 서명 체계에서, 사용자 U 의 공개키는 단순히 " U " 그 자체이다.) 만약 A 가 SMK 를 파괴하고 이미 생성한 비밀키를 보관하지 않는다면, U 만이 자신의 공개키에 대응하는 서명을 생성할 수 있게 된다. 따라서 누구든 " U 의 이름"을 알면 자동으로 U 의 공개키를 알게 되고, PMK 를 사용해 U 의 서명을 검증할 수 있다.

우리 응용에서, 권한 A 는 사용자 i 이며, 가능한 사용자 집합 U 는 라운드-단계 쌍 (r, s) 로 구성된다. 예를 들어,

$$S = \{i\} \times \{r', \dots, r' + 10^6\} \times \{1, \dots, m + 3\}$$

와 같이 정의할 수 있다. 여기서 r' 은 어떤 특정 라운드이고, $m + 3$ 은 한 라운드 내에서 발생할 수 있는 단계 수의 상한이다. i 는 이 집합 S 의 모든 쌍에 대한 비밀키를 계산한 후, SMK 를 파괴한다. 만약 자신이 $SV^{r,s}$ 에 속하지 않는다면, i 는 $sk_i^{r,s}$ 를 사용하지 않는다. 반대로 $i \in SV^{r,s}$ 라면, i 는 $sk_i^{r,s}$ 를 사용해 자신의 메시지 $m_i^{r,s}$ 를 서명한 후 즉시 $sk_i^{r,s}$ 를 파괴한다.

사용자 i 는 시스템에 처음 참여할 때 자신의 첫 번째 마스터 공개키를 공표할 수 있다. 즉, i 가 시스템에 들어오게 하는 지불 p 에, i 의 마스터 공개키 PMK 가 포함될 수 있으며, 이는 i 가 라운드 $[r', r' + 10^6]$ 동안 사용할 공개키임을 나타낸다.

또한, 한 라운드는 최대 $m + 3$ 단계이므로, 라운드가 평균 1분 정도 걸린다고 가정하면, i 가 준비한 일시적 키들은 거의 2년 동안 사용 가능하다. 동시에, 이 일시적 비밀키들을 생성하는 데 걸리는 시간은 길지 않다. 예를 들어, 32바이트 키를 사용하는 타원곡선 기반 체계에서는 각 키를 몇 마이크로초 내에 생성할 수 있다. 따라서 $m + 3 = 180$ 이라면, 1억 8천만 개의 비밀키를 한 시간 이내에 생성할 수 있다.

라운드 $r' + 10^6$ 에 가까워지면, 이후 100만 라운드를 대비하기 위해 i 는 새로운 (PMK', SMK') 쌍을 생성한다. 그런 다음 새로운 공개 마스터키 PMK' 를, 예를 들어 $(PMK', [r' + 10^6 + 1, r' + 2 \cdot 10^6 + 1])$ 형태의 데이터로 새로운 블록에 포함시킨다. 이는 별도의 “거래(transaction)”일 수도 있고, 기존 지분의 추가 정보일 수도 있다. 이를 통해 i 는 모든 이에게 이후 100만 라운드 동안 자신의 일시적 서명 검증에 PMK' 를 사용해야 함을 알린다.

(이 기본 접근법을 변형하여, ID 기반 서명을 사용하지 않고도 일시적 키를 구현할 수 있다. 예를 들어, 머클 트리(Merkle tree)를 사용하는 방식도 가능하다.¹⁵⁾)

5.3 Algorand'의 단계와 BA^* 단계 매칭

앞서 말했듯이, $Algorand'$ 의 한 라운드는 최대 $m + 3$ 단계로 이루어진다.

단계 1.

이 단계에서, 각 잠재 리더 i 는 자신의 후보 블록 B_i^r 과 자격 증명 $\sigma_i^{r,1}$ 을 계산하여 전파한다.

이 자격 증명은 i 를 명시적으로 식별한다. 이는 $\sigma_i^{r,1} \triangleq SIG_i(r, 1, Q^{r-1})$ 이기 때문이다.

잠재 검증자 i 는 메시지의 일부로 자신의 후보 블록 해시 $H(B_i^r)$ 에 대한 서명도 전파한다. 이때 서명은 일시적 공개키 $pk_i^{r,1}$ 에 대해 생성되므로, 전파된 값은 $sig_{pk_i^{r,1}}(H(B_i^r))$ 이다.

우리의 규약상, B_i^r 과 $sig_{pk_i^{r,1}}(H(B_i^r))$ 를 각각 전파하기보다는, $SIG_{pk_i^{r,1}}(H(B_i^r))$ 로 묶어 전파할 수도 있다. 그러나 분석을 위해 서명값 $sig_{pk_i^{r,1}}(H(B_i^r))$ 에 명시적으로 접근할 수 있어야 하므로, 별도로 다룬다.

단계 2.

이 단계에서, 각 검증자 i 는 자격 증명 해시가 가장 작은 잠재 리더를 ℓ_i^r 로 설정하고, ℓ_i^r 이 제안한 블록을 $B_{\ell_i}^r$ 로 둔다.

효율성을 위해, 합의는 블록 B^r 자체가 아니라 그 해시 $H(B^r)$ 에 대해 진행된다. 따라서 검증자 i 는 $B_{\ell_i}^r$ 의 해시 $v_i' = H(B_{\ell_i}^r)$ 를 BA^* 의 초기 값으로 사용하고, 이를 전파한다. 물론 이 값도 일시적 서명으로 인증된다. (i 의 $(r, 2)$ 용 일시적 공개키 $pk_i^{r,2}$ 가 사용된다.) 메시지에는 i 의 자격 증명도 포함된다.

BA^* 의 첫 단계는 단계적 합의 프로토콜 GC 의 첫 단계이므로, $Algorand'$ 의 단계 2는 GC 의 첫 단계에 해당한다.

단계 3.

이 단계에서, 각 검증자 $i \in SV^{r,2}$ 는 BA^* 의 두 번째 단계를 실행한다. 즉, GC 의 두 번째 단계에서 보내야 할 메시지를 전파한다. 역시 메시지는 일시적 서명과 함께 전송되며, i 의 자격 증명도 포함된다.

(이후부터는 “검증자가 메시지를 일시적 서명하고 자격 증명을 함께 전파한다”는 설명을 반복적으로 생략한다.)

단계 4.

이 단계에서, 각 검증자 $i \in SV^{r,4}$ 는 GC 의 출력을 계산한다. 즉, (v_i, g_i) 쌍을 산출한다.

그 후 BA^* 의 첫 번째 단계(BBA^* 의 첫 번째 단계)에 해당하는 메시지를 전파한다. 이때 입력 비트는 $g_i = 2$ 라면 0, 그렇지 않으면 1이다.

단계 $s = 5, \dots, m + 2$.

이 단계들은 각각 BA^* 의 단계 $s - 1$ 에 해당하며, 따라서 BBA^* 의 단계 $s - 3$ 에 해당한다.

우리의 전파 모델은 충분히 비동기적이므로, 단계 s 도중 검증자 $i \in SV^{r,s}$ 가 B^r 블록이 이미 선택되었다는 사실을 알게 될 수 있다. 이런 경우 i 는 라운드 r 실행을 중단하고, 라운드 $r + 1$ 실행으로 넘어간다.

따라서 검증자 $i \in SV^{r,s}$ 의 지침에는, BBA^* 실행이 이전 단계 s' 에서 종료되었는지 여부를 확인하는 과정이 포함된다. BBA^* 는 오직 “코인이 0으로 고정된 단계”나 “코인이 1로 고정된 단계”에서만 종료될 수 있으므로, s' 에 따라 다음과 같이 구분된다.

- **A (종료 조건 0):** $s' - 2 \equiv 0 \pmod{3}$.
- **B (종료 조건 1):** $s' - 2 \equiv 1 \pmod{3}$.

경우 A에서는 B^r 이 비어 있지 않은 블록이며, 따라서 검증자 i 는 B^r 과 올바른 인증서 $CERT^r$ 를 재구성해야 한다.
 경우 B에서는 B^r 이 빈 블록이므로, i 는 $B^r = B_\emptyset^r = (r, \emptyset, H(Q^{r-1}, r), H(B^{r-1}))$ 으로 두고, $CERT^r$ 를 계산한다.

만약 i 가 단계 s 를 실행하는 동안 B^r 이 이미 생성되었다는 증거를 보지 못한다면, 그는 BBA^* 의 단계 $s - 3$ 에서 보냈을 메시지를 그대로 전송한다.

단계 $m + 3$.

만약 $i \in SV^{r, m+3}$ 가 단계 $m + 3$ 실행 도중 B^r 이 이미 이전 단계 s' 에서 생성되었다는 것을 알았다면, 위에서 설명한 대로 처리한다.

그렇지 않고 B^r 이 아직 생성되지 않았다면, i 는 BBA^* 의 마지막 단계에 해당하는 메시지를 보내는 대신, 자신이 보유한 정보를 기반으로 B^r 과 인증서 $CERT^r$ 를 계산한다.

기억하자면, 라운드의 단계 수는 $m + 3$ 을 상한으로 둔다.

5.4 실제 프로토콜 (The Actual Protocol)

기억하자면, 라운드 r 의 단계 s 에서 검증자 $i \in SV^{r, s}$ 는 장기 공개/비밀키 쌍을 사용해 자격 증명 $\sigma_i^{r, s} \triangleq SIG_i(r, s, Q^{r-1})$ 을 생성한다. 또한 $s = 1$ 인 경우에는 $SIG_i^s(Q^{r-1})$ 도 생성한다.

검증자 i 는 자신의 (r, s) 메시지 $m_i^{r, s}$ 를 일시적 비밀키 $sk_i^{r, s}$ 로 서명한다.

편의를 위해, r 과 s 가 문맥상 분명할 때, $esig_i(x)$ 를 $sig_{pk_i^{r, s}}(x)$ 의 약칭으로 사용한다. 또한 $ESIG_i(x)$ 는 $SIG_{pk_i^{r, s}}(x) = (i, x, esig_i(x))$ 를 의미한다.

단계 1: 블록 제안 (Block Proposal)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되는 즉시 라운드 r 의 단계 1을 시작한다.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r, 1}$ 에 속하는지 확인한다.
- $i \notin SV^{r, 1}$ 이면, 즉시 단계를 중단한다.
- $i \in SV^{r, 1}$ 이면(즉, i 가 잠재 리더라면), 지금까지 자신에게 전파된 라운드- r 지분들을 모아 최대 지분 집합 PAY^r 을 계산한다. 그 다음 후보 블록을 $B_i^r = (r, PAY^r, SIG_i(Q^{r-1}), H(B^{r-1}))$ 로 구성한다.
- 마지막으로 메시지 $m_i^{r, 1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r, 1})$ 를 계산하고, 자신의 일시적 비밀키 $sk_i^{r, 1}$ 을 파기한 뒤, $m_i^{r, 1}$ 을 전파한다.

비고.

실제에서는 단계 1의 전역 실행을 단축하기 위해 $(r, 1)$ 메시지를 *선택적으로 전파(selectively propagated)*하는 것이 중요하다.

즉, 사용자 i 가 처음으로 수신하고 성공적으로 검증한 $(r, 1)$ 메시지는 항상 전파한다. 그러나 그 이후 수신한 $(r, 1)$ 메시지는, 자격 증명 해시 값이 지금까지 본 모든 $(r, 1)$ 메시지의 자격 증명 해시 중 최소일 때만 전파한다.

또한, Georgios Vlachos의 제안처럼, 잠재 리더 i 가 자신의 자격 증명 $\sigma_i^{r, 1}$ 을 블록과 별도로 전파하는 것도 유용하다. 작은 메시지이므로 빠르게 전파되며, 자격 증명 해시가 작은 $m_j^{r, 1}$ 의 신속한 확산을 돕는다. 반면 해시가 큰 것들은 곧 사라진다.

단계 2: 단계적 합의 프로토콜 GC의 첫 번째 단계

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되면 라운드 r 의 단계 2를 시작한다.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r, 2}$ 에 속하는지 확인한다.
- $i \notin SV^{r, 2}$ 이면, 즉시 단계를 중단한다.
- $i \in SV^{r, 2}$ 이면, 시간 $t_2 \triangleq \lambda + \Lambda$ 만큼 대기한 후 다음을 수행한다.
 1. 자신이 받은 $(r, 1)$ 메시지들 중 자격 증명 해시가 가장 작은 $\sigma_\ell^{r, 1}$ 을 찾아, 라운드 r 의 리더를 ℓ 로 정한다.¹⁶
 2. ℓ 로부터 유효한 메시지 $m_\ell^{r, 1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r, 1})$ 를 받았다면, $v_i' = H(B_\ell^r)$ 로 둔다. 그렇지 않으면 $v_i' = \perp$ 으로 둔다.¹⁷
 3. 메시지 $m_i^{r, 2} = (ESIG_i(v_i'), \sigma_i^{r, 2})$ 를 계산하고, 자신의 일시적 비밀키 $sk_i^{r, 2}$ 를 파기한 뒤, 이를 전파한다.¹⁸

단계 3: GC의 두 번째 단계

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되면 라운드 r 의 단계 3을 시작한다.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,3}$ 에 속하는지 확인한다.
- $i \notin SV^{r,3}$ 이면 즉시 단계를 중단한다.
- $i \in SV^{r,3}$ 이면, 시간 $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$ 만큼 대기한 후 다음을 수행한다.
 1. 자신이 받은 유효한 $m_j^{r,2}$ 메시지들 중에서, 동일한 값 x 를 포함하는 메시지가 $2t + 1$ 개 이상 있으면 $m_i^{r,3} = (ESIG_i(x), \sigma_i^{r,3})$ 를 생성한다.
 2. 그렇지 않으면 $m_i^{r,3} = (ESIG_i(\perp), \sigma_i^{r,3})$ 를 생성한다.
 3. 자신의 일시적 비밀키 $sk_i^{r,3}$ 를 파기하고 $m_i^{r,3}$ 를 전파한다.

단계 4: GC 출력 및 BBA*의 첫 단계

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되면 라운드 r 의 단계 4를 시작한다.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,4}$ 에 속하는지 확인한다.
- $i \notin SV^{r,4}$ 이면 즉시 단계를 중단한다.
- $i \in SV^{r,4}$ 이면, 시간 $t_4 \triangleq t_3 + 2\lambda = 5\lambda + \Lambda$ 만큼 대기한 후 다음을 수행한다.
 1. GC의 출력을 계산한다.
 - 만약 자신이 받은 유효한 $m_j^{r,3}$ 메시지들 중 동일한 값 $x \neq \perp$ 가 $2t + 1$ 개 이상 있으면, $v_i = x, g_i = 2$ 로 둔다.
 - 그렇지 않고, 동일한 값 $x \neq \perp$ 가 $t + 1$ 개 이상 있으면, $v_i = x, g_i = 1$ 로 둔다.
 - 그 외에는 $v_i = H(B_\ell^r), g_i = 0$ 으로 둔다.
 2. BBA*의 입력 비트를 $g_i = 2$ 이면 $b_i = 0$, 아니면 $b_i = 1$ 로 둔다.
 3. 메시지 $m_i^{r,4} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$ 를 계산하고, 자신의 일시적 비밀키 $sk_i^{r,4}$ 를 파기한 뒤, 이를 전파한다.

단계 s , $5 \leq s \leq m + 2$, $s - 2 \equiv 0 \pmod{3}$: BBA*의 코인-0 고정 단계 (Coin-Fixed-To-0 Step)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되면 라운드 r 의 단계 s 를 시작한다.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,s}$ 에 속하는지 확인한다.
- $i \notin SV^{r,s}$ 이면 즉시 단계를 중단한다.
- $i \in SV^{r,s}$ 이면, $t_s \triangleq t_{s-1} + 2\lambda = (2s - 3)\lambda + \Lambda$ 시간 동안 대기한다.

종료 조건 0:

대기 도중, 어떤 s' 와 문자열 $v \neq \perp$ 이 존재하여

- (a) $5 \leq s' \leq s$ 이고 $s' - 2 \equiv 0 \pmod{3}$ 이며,
- (b) $t_H = \frac{2n}{3} + 1$ 개 이상의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했으며,
- (c) 또한 $v = H(B_j^r)$ 인 유효한 메시지 $m_j^{r,1} = (B_j^r, esig_j(H(B_j^r)), \sigma_j^{r,1})$ 를 수신했다면,

i 는 즉시 라운드 r 실행을 종료한다. 이때 $B^r = B_j^r$ 로 두고, $CERT^r$ 은 (b)에서 수신한 메시지 집합으로 둔다. ¹⁹

종료 조건 1:

대기 도중, 어떤 s' 가 존재하여

- (a') $6 \leq s' \leq s$, $s' - 2 \equiv 1 \pmod{3}$ 이고,
- (b') t_H 개의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했다면,

i 는 즉시 라운드 r 실행을 종료한다. 이때 $B^r = B_\emptyset^r$ 로 두고, $CERT^r$ 은 (b')에서 수신한 메시지 집합으로 둔다. ²⁰

그 외에는 대기 종료 시점에 다음을 수행한다.

- v_i : 자신이 받은 유효한 $m_j^{r,s-1}$ 메시지의 두 번째 요소 v_j 값들 중 다수결로 결정한다.
- b_i :
 - 자신이 받은 $m_j^{r,s-1}$ 중 $2/3$ 이상이 $(ESIG_j(0), ESIG_j(v), \sigma_j^{r,s-1})$ 형태이면 $b_i = 0$.
 - 자신이 받은 $m_j^{r,s-1}$ 중 $2/3$ 이상이 $(ESIG_j(1), ESIG_j(v), \sigma_j^{r,s-1})$ 형태이면 $b_i = 1$.
 - 그 외에는 $b_i = 0$.

마지막으로 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ 를 계산하고, $sk_i^{r,s}$ 를 파기한 후 전파한다.

단계 s , $6 \leq s \leq m+2$, $s-2 \equiv 1 \pmod{3}$: BBA*의 코인-1 고정 단계 (Coin-Fixed-To-1 Step)

- 구조는 코인-0 고정 단계와 동일하다.
- 종료 조건 0과 1은 동일하게 적용된다.
- 차이점은 기본 규칙에서, 대기 종료 후 $b_i = 1$ 로 기본 설정된다는 점이다.

단계 s , $7 \leq s \leq m+2$, $s-2 \equiv 2 \pmod{3}$: BBA*의 실제 코인 뒤집기 단계 (Coin-Genuinely-Flipped Step)

- 구조는 위와 동일하다.
- 종료 조건 0과 1은 동일하게 적용된다.
- 대기 종료 후, b_i 는 다음과 같이 정해진다.
 - 수신한 $m_j^{r,s-1}$ 중 $2/3$ 이상이 비트 0을 지지하면 $b_i = 0$.
 - 수신한 $m_j^{r,s-1}$ 중 $2/3$ 이상이 비트 1을 지지하면 $b_i = 1$.
 - 그 외의 경우, $b_i = \text{lsb}\left(\min_{j \in SV^{r,s-1}} H(\sigma_j^{r,s-1})\right)$ 로 둔다. (즉, 이전 단계 검증자들의 자격 증명 해시 중 최소값의 최하위 비트를 택한다.)

이후 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ 를 전파한다.

단계 $m+3$: BBA*의 마지막 단계

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침.

- 사용자 i 는 B^{r-1} 의 세 번째 구성요소로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,m+3}$ 에 속하는지 확인한다.
- $i \notin SV^{r,m+3}$ 이면 단계를 중단한다.
- $i \in SV^{r,m+3}$ 이면, $t_{m+3} \triangleq (2m+3)\lambda + \Lambda$ 시간 동안 대기한다.

대기 종료 시, i 는 $out_i = 1$ 로 두고 $B^r = B_\emptyset^r$ 로 설정한다.

이후 $m_i^{r,m+3} = (ESIG_i(out_i), ESIG_i(H(B^r)), \sigma_i^{r,m+3})$ 를 계산하고, 자신의 일시적 비밀키 $sk_i^{r,m+3}$ 를 파기한 후 이를 전파하여 B^r 을 인증한다.

비검증자(non-verifiers)의 라운드 r 블록 재구성

모든 일반 사용자 i 를 위한 지침. 사용자 i 는 B^{r-1} 을 알게 되면 라운드 r 을 시작하고, 블록 정보를 기다린다.

- 어떤 s' 와 문자열 v 가 존재하여
 - (a) $5 \leq s' \leq m+3$, $s'-2 \equiv 0 \pmod{3}$ 이고,
 - (b) t_H 개의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했으며,
 - (c) $v = H(B_j^r)$ 인 유효한 메시지 $m_j^{r,1}$ 도 수신했다면, i 는 $B^r = B_j^r$ 로 두고, $CERT^r$ 을 해당 메시지 집합으로 둔 뒤, 라운드 r 실행을 종료한다.
- 또는, 어떤 s' 가 존재하여
 - (a') $6 \leq s' \leq m+3$, $s'-2 \equiv 1 \pmod{3}$ 이고,
 - (b') t_H 개의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했다면, i 는 $B^r = B_\emptyset^r$ 로 두고, $CERT^r$ 을 해당 메시지 집합으로 둔 뒤, 라운드 r 실행을 종료한다.
- 또는, i 가 t_H 개의 유효한 $(r, m+3)$ 메시지 $m_j^{r,m+3} = (ESIG_j(1), ESIG_j(H(B^r)), \sigma_j^{r,m+3})$ 를 수신했다면, i 는 $B^r = B_\emptyset^r$ 로 두고, $CERT^r$ 을 해당 메시지 집합으로 둔 뒤, 라운드 r 실행을 종료한다.

5.5 Algorand', 분석 (Analysis of Algorand')

분석을 위해, 각 라운드 $r \geq 0$ 에 대해 다음 표기법을 도입한다.

- T^r : 첫 번째 정직한 사용자가 B^{r-1} 을 알게 되는 시간.
- I^{r+1} : 구간 $[T^{r+1}, T^{r+1} + \lambda]$.

라운드 0에서는 초기화에 따라 $T^0 = 0$ 이다.

각 단계 $s \geq 1$ 와 검증자 $i \in SV^{r,s}$ 에 대해, $\alpha_i^{r,s}$ 와 $\beta_i^{r,s}$ 는 각각 플레이어 i 가 라운드 r 의 단계 s 를 시작하고 종료하는 시각이다.

또한 $t_s = (2s - 3)\lambda + \Lambda$ (단, $2 \leq s \leq m + 3$)로 둔다.

편의를 위해 $I^0 \triangleq \{0\}$, $t_1 \triangleq 0$ 이라 정의한다.

마지막으로, $L' \leq m/3$ 은 확률 변수로, 각 시행이 $2/3$ 의 확률로 성공하는 베르누이 시행에서 1이 나올 때까지 걸리는 시행 횟수를 나타낸다. 최대 $m/3$ 회 시행 가능하며, 모두 실패하면 $L' = m/3$ 이다.

분석에서 계산 시간은 무시한다. 실제로 계산 시간은 메시지 전파 시간에 비해 미미하다. (물론, λ 와 Λ 를 약간 크게 잡으면 계산 시간을 직접 포함시킬 수 있다.)

아래의 대부분의 진술은 “압도적으로 높은 확률로(with overwhelming probability)” 성립한다. 분석에서는 반복적으로 이를 강조하지 않겠다.

5.6 주요 정리 (Main Theorem)

정리 5.1. 모든 라운드 $r \geq 0$ 에 대해, 압도적으로 높은 확률로 다음 성질이 성립한다.

1. 모든 정직한 사용자는 동일한 블록 B^r 에 합의한다.
2. 리더 ℓ^r 이 정직하다면, B^r 은 ℓ^r 이 생성하며, B^r 은 ℓ^r 이 I^r 내 시간 $\alpha_{\ell^r}^{r,1}$ 까지 수신한 최대 지분 집합을 포함한다. 또한 $T^{r+1} \leq T^r + 8\lambda + \Lambda$ 이고, 모든 정직한 사용자는 I^{r+1} 구간 내에 B^r 을 알게 된다.
3. 리더 ℓ^r 이 악의적이라면, $T^{r+1} \leq T^r + (6L' + 10)\lambda + \Lambda$ 이고, 모든 정직한 사용자는 I^{r+1} 구간 내에 B^r 을 알게 된다.
4. $p_h = h^2(1 + h - h^2)$ 일 때, 리더 ℓ^r 은 최소 p_h 의 확률로 정직하다.

비고 (Remarks).

• 블록 생성과 실제 지연.

블록 B^r 생성 시간은 $T^{r+1} - T^r$ 로 정의한다. 즉, 첫 번째 정직한 사용자가 B^{r-1} 을 알게 된 시각과 첫 번째 정직한 사용자가 B^r 을 알게 된 시각의 차이이다.

- 리더가 정직한 경우, 정리 5.1의 성질 2에 의해 정확히 $8\lambda + \Lambda$ 이다. $h > 2/3$ 라는 값과 무관하다.
- 리더가 악의적인 경우, 성질 3에 의해 기대 블록 생성 시간은 $(\frac{12}{p_h} + 10)\lambda + \Lambda$ 이하이다. h 의 정확한 값과 무관하다. ²¹

그러나 실제 기대 블록 생성 시간은 h 에 의존한다. 성질 4에 따르면 리더가 정직할 확률은 $p_h = h^2(1 + h - h^2)$ 이므로,

$$\mathbb{E}[T^{r+1} - T^r] \leq p_h \cdot (8\lambda + \Lambda) + (1 - p_h) \cdot \left(\left(\frac{12}{p_h} + 10 \right) \lambda + \Lambda \right).$$

예를 들어, $h = 80\%$ 일 때 $\mathbb{E}[T^{r+1} - T^r] \leq 12.7\lambda + \Lambda$ 이다.

• λ 와 Λ 비교.

검증자들이 단계에서 보내는 메시지의 크기는 대부분 디지털 서명 키의 길이에 의해 지배된다. 이 키 길이는 사용자 수가 아무리 많아도 고정될 수 있다. 따라서 사용자 수가 커져도 Λ 값은 크게 변하지 않는다.

반면 거래율이 일정하다면, 사용자 수가 증가할수록 거래 수가 증가한다. 그러면 거래를 적시에 처리하려면 블록 크기도 증가해야 하므로, Λ 역시 장기적으로는 커진다. 따라서 결국 $\lambda \ll \Lambda$ 가 된다. 그러므로 λ 앞에는 큰 계수가, Λ 앞에는 1이 계수로 붙는 것이 합리적이다.

5.7 완전성 보조정리 (The Completeness Lemma)

보조정리 5.2 (완전성 보조정리, 재진술).

라운드 $r - 1$ 에서 성질 1-3이 성립한다고 가정하자. 이때 리더 ℓ^r 이 정직하다면, 압도적으로 높은 확률로 다음이 성립한다.

- 모든 정직한 사용자는 동일한 블록 B^r 에 합의한다. 이 블록은 ℓ^r 이 생성하며, ℓ^r 이 I^r 구간 내 $\alpha_{\ell^r}^{r,1}$ 시각까지 수신한 최대 지분 집합을 포함한다.
- $T^{r+1} \leq T^r + 8\lambda + \Lambda$ 이며, 모든 정직한 사용자는 I^{r+1} 구간 내에 B^r 을 알게 된다.

증명.

귀납 가정 및 보조정리 5.5에 따라, 라운드 r 의 각 단계 s 와 정직한 검증자 $i \in HSV^{r,s}$ 에 대해 $\alpha_i^{r,s} \in I^r$ 이다. 이제 프로토콜을 단계 별로 분석한다.

단계 1.

모든 정직한 검증자 $i \in HSV^{r,1}$ 는 시각 $\beta_i^{r,1} = \alpha_i^{r,1}$ 에 메시지 $m_i^{r,1}$ 을 전파한다.

여기서 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$ 이며, $B_i^r = (r, PAY^r, SIG_i(Q^{r-1}), H(B^{r-1}))$ 이고, PAY^r 은 i 가 $\alpha_i^{r,1}$ 까지 수신한 지불들 중 최대 지불 집합이다.

단계 2.

정직한 검증자 $i \in HSV^{r,2}$ 를 하나 고정하자. 보조정리 5.5에 의해, i 는 $\beta_i^{r,2} = \alpha_i^{r,2} + t_2$ 시점에 $HSV^{r,1}$ 의 모든 메시지를 수신한다.

따라서 $m_\ell^{r,1}$ 역시 수신하게 된다. 정의에 따라 ℓ^r 의 자격 증명 해시는 모든 잠재 리더의 것보다 작으므로, i 는 리더를 ℓ^r 로 인식한다.

따라서 i 는 $\beta_i^{r,2}$ 시점에 $m_i^{r,2} = (ESIG_i(H(B_\ell^r)), \sigma_i^{r,2})$ 를 전파한다.

($r = 0$ 일 때는 $\beta_i^{r,2} = t_2$ 가 된다는 점만 다르며, 다른 분석은 동일하다.)

단계 3.

정직한 검증자 $i \in HSV^{r,3}$ 를 하나 고정하자. 보조정리 5.5에 의해, i 는 $\beta_i^{r,3} = \alpha_i^{r,3} + t_3$ 시점에 $HSV^{r,2}$ 의 모든 메시지를 수신한다.

n 의 선택에 의해, 압도적으로 높은 확률로 $|HSV^{r,2}| > 2|MSV^{r,2}|$ 이다. 또한 정직한 검증자는 서로 모순되는 메시지를 보내지 않으며, 일시적 비밀키는 서명 후 파괴되므로 위조가 불가능하다.

따라서 i 가 받은 유효한 $(r, 2)$ 메시지 중 $2/3$ 이상은 정직한 검증자가 보낸 $m_j^{r,2} = (ESIG_j(H(B_\ell^r)), \sigma_j^{r,2})$ 이다.

따라서 i 는 $\beta_i^{r,3}$ 시점에 $m_i^{r,3} = (ESIG_i(H(B_\ell^r)), \sigma_i^{r,3})$ 를 전파한다.

단계 4.

정직한 검증자 $i \in HSV^{r,4}$ 를 하나 고정하자. 보조정리 5.5에 의해, i 는 $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$ 시점에 $HSV^{r,3}$ 의 모든 메시지를 수신한다.

위와 동일한 논리로, i 가 받은 유효한 $(r, 3)$ 메시지 중 $2/3$ 이상은 정직한 검증자가 보낸 $m_j^{r,3} = (ESIG_j(H(B_\ell^r)), \sigma_j^{r,3})$ 이다.

따라서 i 는 $v_i = H(B_\ell^r)$, $g_i = 2$, $b_i = 0$ 으로 두고, $\beta_i^{r,4}$ 시점에 $m_i^{r,4} = (ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,4})$ 를 전파한다.

단계 5.

정직한 검증자 $i \in HSV^{r,5}$ 를 하나 고정하자. 보조정리 5.5에 의해, i 가 $\alpha_i^{r,5} + t_5$ 까지 기다리면, $HSV^{r,4}$ 의 모든 메시지를 수신한다.

n 의 선택에 따라 $|HSV^{r,4}| \geq t_H$ 가 성립한다.²² 또한 $HSV^{r,4}$ 의 모든 메시지는 $H(B_\ell^r)$ 에 대한 것이다.

따라서 i 는 t_H 개의 유효한 $(r, 4)$ 메시지를 수신하면 멈춘다. 이 중 적어도 하나는 정직한 검증자가 보낸 것이다. 이 메시지는 $T^r + t_4$ 이후에 보내졌고, ℓ^r 의 메시지 $m_\ell^{r,1}$ 도 그 전에 수신했을 것이다.

따라서 i 는 $\beta_i^{r,5} = T$ 시점에 멈추며, $B^r = B_\ell^r$ 로 설정하고, $CERT^r$ 은 해당 메시지 집합으로 둔다.

단계 $s > 5$.

동일한 논리로, 모든 정직한 검증자 $i \in HSV^{r,s}$ 는 멈추고 $B^r = B_\ell^r$ 를 설정한다. 악의적 검증자들이 메시지를 전파하지 않을 수 있지만, t_H 개 이상의 정직한 서명이 확보되므로 결과는 동일하다.

비검증자.

일반 사용자 i 에 대해서도 동일한 논리가 적용된다. i 는 t_H 개의 유효한 $(r, 4)$ 메시지를 수신하면 $B^r = B_\ell^r$ 로 설정하고 멈춘다.

마지막으로, 모든 정직한 사용자가 I^{r+1} 구간 내에 B^r 을 알게 됨을 보인다.

$HSV^{r,5}$ 의 모든 정직한 검증자는 $\alpha_i^{r,5} + t_5 \leq T^r + 8\lambda + \Lambda$ 시점에 B^r 을 알게 된다.

T^{r+1} 은 첫 번째 정직한 사용자가 B^r 을 알게 되는 시간이므로,

$$T^{r+1} \leq T^r + 8\lambda + \Lambda$$

이다. 이후 모든 정직한 사용자는 $T^{r+1} + \lambda$ 이내에 B^r 을 수신한다. 따라서 I^{r+1} 구간 내에 모두 B^r 을 알게 된다.

$r = 0$ 일 때는 실제로 $T^1 \leq 6\lambda + \Lambda$ 가 된다.

이를 종합하면, 보조정리 5.2가 성립한다. ■

5.8 건전성 보조정리 (The Soundness Lemma)

보조정리 5.3 (건전성 보조정리, 재진술).

라운드 $r - 1$ 에서 성질 1-3이 성립한다고 가정하자. 이때 리더 ℓ^r 이 악의적이라면, 압도적으로 높은 확률로 모든 정직한 사용자는 동일한 블록 B^r 에 합의하고,

$T^{r+1} \leq T^r + (6L' + 10)\lambda + \Lambda$ 이며, 모든 정직한 사용자는 I^{r+1} 구간 내에 B^r 을 알게 된다.

증명.

우리는 프로토콜의 두 부분, 즉 GC와 BBA^* 를 나누어 분석한다.

GC 분석.

귀납 가정 및 보조정리 5.5에 따라, $s \in \{2, 3, 4\}$ 인 각 단계와 정직한 검증자 $i \in HSV^{r,s}$ 에 대해, i 는 $\beta_i^{r,s} = \alpha_i^{r,s} + t_s$ 시점에 이전 단계의 모든 정직한 검증자 메시지를 수신한다.

단계 4를 두 가지 경우로 나눈다.

- **경우 1.** 어떤 검증자도 $g_i = 2$ 로 설정하지 않은 경우.
이때 모든 $HSV^{r,4}$ 의 검증자는 $b_i = 1$ 로 설정한다. 따라서 BBA^* 는 비트 1로 시작하며, 이 경우 합의는 빈 블록 $B_\emptyset^r$ 로 귀결된다.
- **경우 2.** $g_i = 2$ 로 설정한 검증자가 존재하는 경우.
이때 모든 정직한 검증자는 $g_i \geq 1$ 이고, 모두 동일한 v' 값을 공유한다. 이는 리더 ℓ 의 후보 블록 해시 $H(B_\ell^r)$ 에 해당한다.

따라서 단계 4의 결과로, 정직한 검증자들은 모두 동일한 값 $v' = H(B_\ell^r)$ 를 공유하게 된다.

BBA^* 분석.

여기서는 두 경우를 구분한다.

- **Case 1.** 모든 $HSV^{r,4}$ 의 검증자가 $b_i = 1$ 로 설정한 경우.
이 경우 단계 5에서 t_H 개의 “비트 0” 메시지를 모을 수 없으므로, 어떤 정직한 검증자도 비어 있지 않은 블록을 확정하지 못한다. 그러나 단계 6은 코인-1 고정 단계이므로, t_H 개의 “비트 1” 메시지가 모여, 모든 정직한 검증자가 $B_\emptyset^r$ 에 합의한다.
- **Case 2.** 일부 $HSV^{r,4}$ 검증자가 $b_i = 0$ 으로 설정한 경우.
이는 GC 단계에서 $g_i = 2$ 인 검증자가 존재했음을 의미한다. 따라서 모든 정직한 검증자가 동일한 $v' = H(B_\ell^r)$ 를 공유한다. 이 경우, 어떤 단계 s^* 에서 처음으로 정직한 검증자가 합의를 종료하게 된다.
 s^* 이전까지는 BBA^* 의 코인 뒤집기 단계가 실행되었고, 정직한 리더의 확률 p_h 에 따라 일정 확률로 합의가 조기에 끝날 수 있다.

따라서 s^* 가 L' 에 의해 결정된다고 볼 수 있다. 기대값을 기준으로, 합의는 $6L' + 10$ 라운드 내에 종료되며, B^r 은 I^{r+1} 구간 내에서 모든 정직한 사용자에게 알려진다.

결론.

따라서 리더가 악의적일 경우에도, $T^{r+1} \leq T^r + (6L' + 10)\lambda + \Lambda$ 이고, 모든 정직한 사용자는 동일한 블록 B^r 을 알게 된다. 보조정리 5.3이 성립한다. ■

5.9 시드 Q^r 의 보안성과 정직한 리더 확률

마지막으로 보조정리 5.4를 증명한다. 다시 상기하면, 라운드 r 의 검증자는 PK^{r-k} 에서 선택되며, Q^{r-1} 에 따라 결정된다.

회고 파라미터 k 를 도입한 이유는, 적대자가 라운드 $r - k$ 시점에서 Q^{r-1} 을 예측하지 못하게 하기 위함이다. H 가 랜덤 오라클이고, Q^{r-1} 은 검증자 선택 과정의 입력 중 하나이므로, PK^{r-k} 에 악의적 사용자를 아무리 추가하더라도, 각 악의적 사용자가 검증자로 선택될 확률은 여전히 정확히 p (또는 1단계의 경우 p_1)이다.

보조정리 5.6.

$k = O(\log_{1/h} F)$ 일 때, 압도적으로 높은 확률로 적대자는 라운드 $r - k$ 시점에서 Q^{r-1} 을 랜덤 오라클에 질의하지 못한다.

증명 (개요).

귀납적으로 가정하자: 라운드 $\gamma < r$ 에 대해, 적대자는 $Q^{\gamma-1}$ 을 라운드 $\gamma - k$ 시점에 질의하지 못했다.

이제 라운드 $r - k$ 시점에서, 적대자가 Q^{r-1} 을 예측하려고 한다고 하자.

라운드 $\gamma = r - k, \dots, r - 1$ 의 각 단계 1에서, $Q^{\gamma-1}$ 이 적대자에게 알려지지 않은 상태라면, 잠재 리더 선정은 $PK^{\gamma-k}$ 위에서 무작위 순열을 생성하는 것과 동일하다.

즉, $H(SIG_i(\gamma, 1, Q^{\gamma-1}))$ 값들에 의해 $PK^{\gamma-k}$ 의 무작위 순열이 유도된다.

리더 ℓ^γ 는 이 순열에서 첫 번째 정직한 사용자가 된다. 따라서 ℓ^γ 가 정직할 확률은 h 이다.

만약 ℓ^γ 가 정직하다면, $Q^\gamma = H(SIG_{\ell^\gamma}(\gamma, 1), \gamma)$ 는 적대자가 위조할 수 없는 무작위 값이다. 따라서 이후 $Q^{\gamma+1}, \dots, Q^{r-1}$ 도 랜덤 오라클의 출력이므로, Q^{r-1} 은 예측 불가능하다.

따라서 적대자가 Q^{r-1} 을 예측할 수 있는 경우는, 라운드 $r - k, \dots, r - 1$ 동안 리더가 모두 악의적인 경우뿐이다. 그러나 이는 h 의 값에 따라 확률이 급격히 줄어든다.

마르코프 체인 유사한 분석을 통해, 충분히 큰 k 를 선택하면, 압도적으로 높은 확률로 적대자가 Q^{r-1} 을 예측하지 못한다.

보조정리 5.4 (재진술).

각 라운드 r 이전의 모든 라운드에서 성질 1-3이 성립한다고 가정하자. 이때 $p_h = h^2(1 + h - h^2)$ 이며, 라운드 r 의 리더 ℓ^r 은 최소 p_h 의 확률로 정직하다.

증명.

보조정리 5.6에 따라, 적대자는 라운드 $r - k$ 시점에서 Q^{r-1} 을 예측할 수 없다. 이는 곧 리더가 항상 h 의 확률로 정직하다는 것을 의미하지는 않는다. 왜냐하면, 리더 선정은 잠재 리더 집합의 무작위 순열에 의해 결정되는데, 이 집합의 앞부분에 악의적 사용자가 몰려 있을 수 있기 때문이다.

그러나 확률적으로, 첫 번째 정직한 사용자가 순열에서 x 번째 위치에 있을 확률은 $(1 - h)^{x-1}h$ 이다.

$x = 1$ 이면 리더는 확률 h 로 정직하다. $x = 2$ 이면, 적대자는 두 가지 선택지를 갖게 된다. 그러나 이 경우에도 정직한 리더가 등장할 확률은 최소 h^2 이다.

이를 일반화하면, 리더가 정직할 확률은

$$p_h \geq h^2(1 + h - h^2)$$

임을 얻는다.

따라서 보조정리 5.4가 성립한다. ■

6 Algorand'₂

본 장에서는 Algorand'의 또 다른 버전인 Algorand'₂를 구성한다. 이 버전은 다음 가정을 바탕으로 한다.

정직 사용자 다수 가정 (Honest Majority of Users Assumption):

각 라운드 r 의 PK^r 에서 사용자의 $2/3$ 이상이 정직하다.

8장에서 우리는 이 가정을 정직 자금 다수 가정(Honest Majority of Money Assumption)으로 대체하는 방법을 보일 것이다.

6.1 Algorand'₂를 위한 추가 표기법 및 매개변수

표기법

- $\mu \in \mathbb{Z}^+$: 압도적으로 높은 확률로 한 라운드에서 실제로 실행될 단계 수의 상한. (매개변수 μ 는 사용자가 라운드별로 미리 준비해야 하는 일시적 키 수를 결정한다.)
- L' : 각 시행이 확률 $p/2$ 로 성공하는 베르누이 시행에서 1이 나올 때까지 걸리는 시행 횟수를 나타내는 확률 변수. 이는 블록 B^r 을 생성하는 데 필요한 시간을 상한하는 데 사용된다.
- t_H : 라운드 r 의 단계 $s > 1$ 에서 정직한 검증자 수의 하한. n 과 p 가 주어졌을 때, 압도적으로 높은 확률로 $SV^{r,s}$ 내 정직한 검증자 수는 t_H 를 초과한다.

매개변수

- 매개변수 간 관계.

- 라운드 r 의 각 단계 $s > 1$ 에서, n 은 압도적으로 높은 확률로

$$|HSV^{r,s}| > t_H \quad \text{이고} \quad |HSV^{r,s}| + 2|MSV^{r,s}| < 2t_H$$

가 성립하도록 선택된다. 이는 $SV^{r,s}$ 에서 $2/3$ 정칙 다수가 보장됨을 의미한다.

- h 가 1에 가까울수록, 필요한 n 은 더 작아도 된다.

- 우리는 체르노프 경계(Chernoff bounds)의 변형을 사용해 위 조건이 압도적으로 높은 확률로 성립함을 보장한다.

- 중요 매개변수의 예시.

- $F = 10^{-18}$.

- $n \approx 4000, t_H \approx 0.69n, k = 70$.

6.2 Algorand'2에서의 일시적 키 구현

앞서 설명했듯이, 검증자 $i \in SV^{r,s}$ 는 라운드 r 의 단계 s 에서 자신의 메시지 $m_i^{r,s}$ 를 일시적 공개키 $pk_i^{r,s}$ 와 일시적 비밀키 $sk_i^{r,s}$ 로 서명하며, 사용 직후 비밀키를 파기한다.

라운드에서 실행될 수 있는 단계 수가 μ 로 제한된다면, 미리 μ 개의 일시적 키를 준비하면 된다.

예를 들어, Algorand'1에서는 $\mu = m + 3$ 이므로, i 는 10^6 라운드 동안 사용할 (PMK, SMK) 를 준비해 필요한 모든 일시적 키를 생성해둘 수 있었다.

그러나 Algorand'2에서는, 특정 라운드 r 이 μ 단계를 초과해 더 많은 단계를 필요로 할 수도 있다. 만약 이런 상황이 발생하면, i 는 라운드 r 에서 단계 $s > \mu$ 에 해당하는 메시지를 서명할 수 없게 된다. 왜냐하면 i 는 오직 μ 개의 키만 준비했기 때문이다.

이를 해결하기 위해, i 는 마지막 일시적 키 $pk_i^{r,\mu}$ 를 사용해 새로운 키 저장소를 생성할 수 있다. 예를 들어:

1. 새로운 마스터 키 쌍 (PMK, SMK) 를 생성한다.
2. 이를 사용해 추가적인 10^6 개의 일시적 키 $sk_i^{r,\mu+1}, \dots, sk_i^{r,\mu+10^6}$ 를 생성한다.
3. $pk_i^{r,\mu}$ 에 대응하는 비밀키 $sk_i^{r,\mu}$ 로 PMK 를 서명한다.
4. SMK 와 $sk_i^{r,\mu}$ 를 파기한다.

이 과정을 통해, 라운드 r 에서 $\mu + 1, \dots, \mu + 10^6$ 단계가 실행되더라도, i 는 적절한 일시적 키로 메시지를 서명할 수 있다.

또한 다른 사용자들은 PMK 가 $pk_i^{r,\mu}$ 로부터 서명되었음을 확인함으로써, 새로운 키 저장소가 실제 i 의 것임을 검증할 수 있다.

이 접근법은 필요하다면 여러 번 반복할 수 있다. 즉, 마지막 일시적 키는 언제든지 새로운 마스터키를 인증하는 용도로 사용될 수 있으며, 이를 통해 무제한으로 새로운 일시적 키 저장소를 확장할 수 있다.

6.3 실제 프로토콜 Algorand'2

기억하자면, 라운드 r 의 단계 s 에서 검증자 $i \in SV^{r,s}$ 는 장기 공개/비밀키 쌍으로 자격 증명 $\sigma_i^{r,s} \triangleq SIG_i(r, s, Q^{r-1})$ 을 생성한다.

또한 $s = 1$ 인 경우에는 $SIG_i(Q^{r-1})$ 도 생성한다.

검증자 i 는 자신의 일시적 키 쌍 $(pk_i^{r,s}, sk_i^{r,s})$ 로 필요한 다른 메시지들을 서명한다.

편의를 위해, $esig_i(m)$ 은 $sig_{pk_i^{r,s}}(m)$ 을, $ESIG_i(m)$ 은 $SIG_{pk_i^{r,s}}(m) = (i, m, esig_i(m))$ 을 의미한다.

단계 1: 블록 제안 (Block Proposal)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 $CERT^{r-1}$ 을 알게 되면 라운드 r 의 단계 1을 시작한다.

이는 i 가 $H(B^{r-1})$ 와 Q^{r-1} 을 명확히 계산할 수 있게 해준다.

- 사용자 i 는 Q^{r-1} 을 사용해 자신이 $SV^{r,1}$ 에 속하는지 확인한다.
 - $i \notin SV^{r,1}$ 이면 아무것도 하지 않는다.
 - $i \in SV^{r,1}$ 이면 (즉, 잠재 리더라면) 다음을 수행한다.

1. 만약 i 가 블록체인 $B^0, \dots, B^{r-1}$ 전체를 직접 보았다면, 라운드 $-r$ 까지 전파된 지불들을 모아 최대 지불 집합 PAY^r 을 계산한다.
2. 그렇지 않다면, $PAY^r = \emptyset$ 로 둔다.
3. 후보 블록 $B_i^r = (r, PAY^r, SIG_i(Q^{r-1}), H(B^{r-1}))$ 을 구성한다.
4. 메시지 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$ 를 계산하고, 자신의 일시적 비밀키 $sk_i^{r,1}$ 을 파기한다.
5. 마지막으로 두 메시지를 동시에 전파한다: $m_i^{r,1}$ 과 $(SIG_i(Q^{r-1}), \sigma_i^{r,1})$.⁴

선택적 전파 (Selective Propagation).

단계 1을 빠르게 마치려면 $(r, 1)$ 메시지를 선택적으로 전파하는 것이 중요하다.

- 각 사용자 j 는 자신이 처음으로 수신하고 검증에 성공한 $(r, 1)$ 메시지는 항상 전파한다. (블록이 포함된 경우는 단순 자격 증명 메시지든 상관없다.)
- 이후 수신하는 $(r, 1)$ 메시지는, 지금까지 본 자격 증명 해시 중 최소일 때만 전파한다.
- 단, 동일한 사용자 i 로부터 두 개의 다른 $m_i^{r,1}$ 메시지를 받으면, j 는 두 번째 메시지를 폐기한다. (이는 i 가 악의적임을 의미한다.)

또한, 잠재 리더 i 가 자신의 자격 증명 $\sigma_i^{r,1}$ 을 블록과 별도로 전파하는 것이 유용하다. 작은 메시지는 빠르게 퍼지고, 자격 증명 해시가 작은 후보의 메시지가 네트워크에 빨리 도달하도록 돕는다.

단계 2: GC의 첫 단계 (The First Step of Graded Consensus)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. i 는 $CERT^{r-1}$ 을 알게 되면 라운드 r 의 단계 2를 시작한다.

- i 는 최대 $t_2 \triangleq \lambda + \Lambda$ 까지 대기한다. 대기 도중 다음을 수행한다.
 1. 2λ 가 지나면, i 는 자신이 받은 $(r, 1)$ 메시지들 중 자격 증명 해시가 최소인 리더 ℓ 을 찾는다.¹⁶
 2. 만약 $CERT^{r-1}$ 이 B^{r-1} 과 일치하며, ℓ 로부터 유효한 메시지 $m_\ell^{r,1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r,1})$ 를 받았다면, $v'_i = (H(B^r), \ell)$ 로 둔다.¹⁷
 3. 그렇지 않다면, t_2 가 끝날 때 $v'_i = \perp$ 로 둔다.
 4. Q^{r-1} 을 계산한 후, 자신이 $SV^{r,2}$ 에 속하는지 확인한다.
 5. $i \in SV^{r,2}$ 이면, $m_i^{r,2} = (ESIG_i(v'_i), \sigma_i^{r,2})$ 를 계산하고 $sk_i^{r,2}$ 를 파기한 뒤 전파한다.
 6. $i \notin SV^{r,2}$ 이면 아무것도 하지 않는다.

단계 3: GC의 두 번째 단계 (The Second Step of GC)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. i 는 $CERT^{r-1}$ 을 알게 되면 라운드 r 의 단계 3을 시작한다.

- i 는 최대 $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$ 까지 대기한다.
- 대기 도중, 만약 어떤 값 v 에 대해, 서로 모순되지 않는 t_H 개의 유효한 $m_j^{r,2} = (ESIG_j(v), \sigma_j^{r,2})$ 메시지를 수신했다면, $v' = v$ 로 두고 즉시 대기를 종료한다.
- 그렇지 않다면, t_3 종료 시 $v' = \perp$ 로 둔다.
- Q^{r-1} 을 계산한 후, 자신이 $SV^{r,3}$ 에 속하는지 확인한다.
- $i \in SV^{r,3}$ 이면, $m_i^{r,3} = (ESIG_i(v'), \sigma_i^{r,3})$ 를 계산하고 $sk_i^{r,3}$ 를 파기한 뒤 전파한다.
- $i \notin SV^{r,3}$ 이면 아무것도 하지 않는다.

단계 4: GC 출력 및 BBA*의 첫 단계

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. i 는 자신의 단계 3이 끝나면 단계 4를 시작한다.

- i 는 최대 2Λ 까지 대기한다. 대기 도중 다음을 수행한다.
 1. GC의 출력을 계산한다.
 - 만약 t_H 개의 유효한 $m_j^{r,3} = (ESIG_j(v'), \sigma_j^{r,3})$ 메시지를 수신했다면, $v_i = v', g_i = 2$ 로 둔다.
 - 만약 t_H 개의 유효한 $m_j^{r,3} = (ESIG_j(\perp), \sigma_j^{r,3})$ 메시지를 수신했다면, $v_i = \perp, g_i = 0$ 으로 둔다.
 - 그렇지 않고, $t_H/2$ 개 이상의 동일한 v' 메시지를 수신했다면, $v_i = v', g_i = 1$ 로 둔다.

- 그 외에는 $v_i = \perp, g_i = 0$ 로 둔다.
- 2. BBA^* 의 입력 비트는 $g_i = 2$ 이면 $b_i = 0$, 그렇지 않으면 $b_i = 1$ 로 둔다.
- 3. Q^{r-1} 을 계산한 후, 자신이 $SV^{r,4}$ 에 속하는지 확인한다.
- 4. $i \in SV^{r,4}$ 이면, $m_i^{r,4} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$ 를 계산하고 $sk_i^{r,4}$ 를 파기한 뒤 전파한다.
 $i \notin SV^{r,4}$ 이면 아무것도 하지 않는다.

단계 s , $5 \leq s \leq m+2$, $s-2 \equiv 0 \pmod{3}$: BBA^* 의 코인-0 고정 단계 (Coin-Fixed-To-0 Step)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. 사용자 i 는 자신의 단계 $s-1$ 을 마치면 라운드 r 의 단계 s 를 시작한다.

- 사용자 i 는 $CERT^{r-1}$ 로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,s}$ 에 속하는지 확인한다.
- $i \notin SV^{r,s}$ 이면 즉시 단계를 중단한다.
- $i \in SV^{r,s}$ 이면, $t_s \triangleq (2s-3)\lambda + \Lambda$ 만큼 대기한 뒤 다음을 수행한다.

종료 조건 0:

대기 도중, 어떤 s' 와 문자열 $v \neq \perp$ 가 존재하여

- (a) $5 \leq s' \leq s$ 이고 $s'-2 \equiv 0 \pmod{3}$ 이며,
- (b) t_H 개의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했고,
- (c) 또한 v 가 $m_j^{r,1}$ 에 포함된 블록 해시 $H(B_j^r)$ 와 일치한다면,

i 는 즉시 단계 s (및 라운드 r) 실행을 종료한다. 이때 $B^r = B_j^r$ 로 설정하고, $CERT^r$ 은 조건 (b)의 메시지 집합과 $(SIG_j(Q^{r-1}), \sigma_j^{r,1})$ 로 설정한다.

종료 조건 1:

대기 도중, 어떤 s' 가 존재하여

- (a') $6 \leq s' \leq s$ 이고 $s'-2 \equiv 1 \pmod{3}$ 이며,
- (b') t_H 개의 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v), \sigma_j^{r,s'-1})$ 를 수신했다면,

i 는 즉시 단계 s (및 라운드 r) 실행을 종료한다. 이때 $B^r = B_\emptyset^r$ 로 설정하고, $CERT^r$ 은 조건 (b')의 메시지 집합으로 설정한다.

그 외의 경우:

대기 종료 후, i 는 다음을 수행한다.

- v_i : 자신이 수신한 유효한 $m_j^{r,s-1}$ 의 두 번째 요소들의 다수결로 결정.
- b_i :
 - 수신한 메시지 중 $2/3$ 이상이 $(ESIG_j(0), ESIG_j(v), \sigma_j^{r,s-1})$ 형태이면 $b_i = 0$,
 - $2/3$ 이상이 $(ESIG_j(1), ESIG_j(v), \sigma_j^{r,s-1})$ 형태이면 $b_i = 1$,
 - 그렇지 않으면 $b_i = 0$.

그런 다음 메시지 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ 를 계산하고, $sk_i^{r,s}$ 를 파기한 뒤 전파한다.

단계 s , $6 \leq s \leq m+2$, $s-2 \equiv 1 \pmod{3}$: BBA^* 의 코인-1 고정 단계 (Coin-Fixed-To-1 Step)

- 구조는 위 코인-0 고정 단계와 동일하다.
- 종료 조건 0과 1도 동일하게 적용된다.
- 다만, “그 외의 경우”에서 기본값이 $b_i = 1$ 로 설정된다.

단계 s , $7 \leq s \leq m+2$, $s-2 \equiv 2 \pmod{3}$: BBA^* 의 실제 코인 뒤집기 단계 (Coin-Genuinely-Flipped Step)

- 구조는 동일하다.
- 종료 조건 0과 1은 위와 동일하다.
- 다만, “그 외의 경우”에서 b_i 는 다음과 같이 정해진다.
 - 수신한 메시지 중 $2/3$ 이상이 비트 0을 지지하면 $b_i = 0$.

- 수신한 메시지 중 $2/3$ 이상이 비트 1을 지지하면 $b_i = 1$.
- 그 외의 경우,

$$b_i = \text{lsb} \left(\min_{j \in SV^{r,s-1}} H(\sigma_j^{r,s-1}) \right).$$

그런 다음 메시지 $m_i^{r,s} = (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$ 를 계산하고, $sk_i^{r,s}$ 를 파기한 뒤 전파한다.

단계 $m + 3$: BBA*의 마지막 단계 (The Last Step of BBA*)

모든 사용자 $i \in PK^{r-k}$ 를 위한 지침. i 는 자신의 단계 $m + 2$ 를 마치면 라운드 r 의 단계 $m + 3$ 을 시작한다.

- i 는 $CERT^{r-1}$ 로부터 Q^{r-1} 을 계산하고, 자신이 $SV^{r,m+3}$ 에 속하는지 확인한다.
- $i \notin SV^{r,m+3}$ 이면 즉시 중단한다.
- $i \in SV^{r,m+3}$ 이면, $t_{m+3} = (2m + 3)\lambda + \Lambda$ 만큼 대기한 뒤 다음을 수행한다.

종료 조건 0/1:

위 코인-0, 코인-1 고정 단계와 동일하다.

그 외의 경우:

대기 종료 시, i 는 $out_i = 1$, $B^r = B_\emptyset^r$ 로 설정한다.

이후 메시지 $m_i^{r,m+3} = (ESIG_i(out_i), ESIG_i(H(B^r)), \sigma_i^{r,m+3})$ 를 계산하고, $sk_i^{r,m+3}$ 를 파기한 뒤 전파하여 B^r 을 인증한다.

비검증자의 라운드 r 블록 재구성 (Reconstruction of the Round- r Block by Non-Verifiers)

모든 사용자 i 를 위한 지침. i 는 $CERT^{r-1}$ 을 알게 되면 라운드 r 을 시작한다.

- i 는 각 단계에서 메시지 전파를 돕지만, 자신이 검증자가 아닌 단계에서는 메시지를 새로 생성하지 않는다.
- i 는 종료 조건 0 또는 1을 만족하는 순간 라운드 r 을 종료하고, 해당 $CERT^r$ 을 확보한다.
- 이후 라운드 $r + 1$ 을 시작한다.
- 만약 $CERT^r$ 이 $B^r = B_\emptyset^r$ 임을 나타낸다면, i 는 $CERT^r$ 을 확보하는 즉시 B^r 을 알게 된다.

6.4 Algorand'2의 분석 (Analysis of Algorand'2)

Algorand'2의 분석은 Algorand'1의 분석으로부터 쉽게 도출된다.

핵심은 동일하다. 즉, 압도적으로 높은 확률로 다음이 성립한다.

1. 모든 정직한 사용자는 동일한 블록 B^r 에 합의한다.
2. 새로운 블록의 리더는 최소 확률

$$p_h = h^2(1 + h - h^2)$$

로 정직하다.

따라서, Algorand'2에서는 라운드마다 리더가 정직할 확률이 충분히 높으며, 블록체인은 압도적으로 높은 확률로 포크가 발생하지 않는다.

또한, Algorand'2는 라운드 단계 수에 상한을 두지 않으므로, 경우에 따라 Algorand'1보다 빠르게 합의에 도달할 수 있다. 반대로, 이론적으로는 무한히 많은 단계를 거칠 수도 있으나, 실제로는 기대값에 따라 매우 적은 수의 단계에서 합의가 이루어진다.

요약.

- Algorand'1은 최대 단계 수 m 을 제한하고, 합의가 실패할 경우 빈 블록에 합의한다.
- Algorand'2는 단계 수에 제한을 두지 않고, 기대 시간 면에서 더 효율적이다.
- 두 구현 모두 동일한 보안 핵심을 공유하며, 차이는 성능-편의성의 트레이드오프에 불과하다.

7 오프라인 정직 사용자 처리 (Handling Offline Honest Users)

앞서 설명했듯이, 정직한 사용자는 자신에게 주어진 모든 지침을 따른다. 여기에는 온라인 상태를 유지하며 프로토콜을 실행하는 것도 포함된다.

Algorand에서는 계산 및 통신 요구량이 크지 않기 때문에, 정직한 사용자가 백그라운드에서 프로토콜을 실행하는 것은 큰 부담이 아니다.

그러나 Algorand는 많은 수의 정직한 사용자가 오프라인이 되더라도 동작할 수 있도록 쉽게 수정될 수 있다.

이를 위해 두 가지 모델을 고려할 수 있다.

먼저, 정직한 플레이어 비율이 95%라 가정해 보자. 이때 모든 매개변수를 $h = 80\%$ 로 설정한다면, 실제로는 정직한 사용자의 절반이 오프라인 상태여도 시스템은 정상적으로 동작한다.

즉, 온라인 사용자 중 최소 80%가 정직하기 때문에, 프로토콜이 여전히 안전하다.

지속적 참여(Continual Participation)에서 게으른 정직(Lazy Honesty)으로.

$Algorand'_1$ 과 $Algorand'_2$ 에서는 회고 파라미터 k 가 선택된다.

k 를 충분히 크게 선택하면, 모든 사용자가 지속적으로 참여해야 한다는 요구사항을 제거할 수 있다.

이는 BA 프로토콜 BBA^* 가 정직한 다수를 필요로 한다는 핵심 속성과 관련된다.

이를 다른 방식으로 충족하기 위해, **게으른 정직(lazy honesty)** 모델을 도입할 수 있다.

- 사용자 i 는 **게으르지만 정직(lazy-but-honest)**하다 함은:
 - (1) 프로토콜에 참여하라는 요청을 받으면 모든 지침을 따른다.
 - (2) 그러나 요청을 받는 일은 드물고, 미리 충분한 사전 통보를 받는다.
 - (3) 참여 시 보상을 받을 수도 있다.

예를 들어, i 가 다가오는 주간에 자신이 검증자로 선택될지를 미리 확인할 수 있다고 하자.

이를 위해, 프로토콜은 라운드 r 의 검증자를 $r - k - 2000$ 시점의 사용자 집합에서 Q^{r-2001} 을 기반으로 선택한다.

따라서, i 는 자신이 다음 2000라운드(약 1주일)에 참여해야 할지를 사전에 확인할 수 있다.

만약 검증자로 선택되지 않았다면, i 는 안심하고 오프라인 상태가 될 수 있다. 어차피 온라인 상태였더라도 실제로는 참여하지 않았을 것이기 때문이다.

만약 i 가 검증자로 선택되었다면, 필요한 데이터를 준비하고 해당 라운드에서 정직하게 참여하면 된다.

결론.

게으른 정직 모델에서는, 정직한 사용자가 드물게만 참여하더라도 안전성이 보장된다.

즉, 특정 시점에 온라인 사용자 다수가 악의적일 수 있더라도, 필요한 순간에 정직한 검증자가 참여한다면 프로토콜은 안전하다.

8 정직 자금 다수 가정 하의 Algorand'

(Protocol $Algorand'$ with Honest Majority of Money)

이제 우리는 정직 사용자 다수 가정을, 더 의미 있는 **정직 자금 다수 가정(Honest Majority of Money, HMM)**으로 대체하는 방법을 보인다.

기본 아이디어는 (지분증명 방식의 변형처럼) 사용자 $i \in PK^{r-k}$ 가 $SV^{r,s}$ 에 포함될 확률을, 해당 라운드에서 i 가 소유한 금액에 비례하도록 정하는 것이다.²³

우리의 HMM 가정에 따르면, 이 금액은 라운드 $r - k$ 시점이든, 라운드 r 시작 시점이든 선택할 수 있다.

우리는 지속적 참여를 전제로 하지 않는다면, 전자의 선택이 더 적절하다. (즉, 라운드 $r - k - 2000$ 시점의 금액을 사용하는 것이다.)

그러나 여기서는 지속적 참여를 전제한다고 가정하고, 라운드 r 시점의 금액을 사용한다.

가장 단순한 구현 (The Simplest Implementation)

가장 단순한 구현은, 각 공개키가 최대 1 단위의 화폐만 보유하도록 강제하는 것이다.

그런 다음 무작위로 n 명의 사용자를 PK^{r-k} 에서 선택하되, $a_i^{(r)} = 1$ 인 사용자만 포함시킨다.

다음으로 단순한 구현 (The Next Simplest Implementation)

다음으로 단순한 구현은 각 공개키가 최대 M 단위의 화폐만 보유하도록 요구하는 것이다.

여기서 M 은 고정된 값이며, 시스템 전체 자금에 비해 충분히 작아서, 한 키가 여러 단계에서 반복적으로 선택될 확률은 무시할 수 있다.

그렇다면, 라운드 r 에서 금액 $a_i^{(r)}$ 를 가진 사용자 $i \in PK^{r-k}$ 는, 단계 s 의 검증자로 선택될 조건이 다음과 같이 된다.

$$H(SIG_i(r, s, Q^{r-1})) \leq p \cdot \frac{a_i^{(r)}}{M}.$$

이후 과정은 이전과 동일하다.

더 복잡한 구현 (A More Complex Implementation)

위 단순한 구현은 사실상 “부유한 참가자가 여러 키를 소유하도록 강제하는” 결과를 낳는다.

이를 일반화하기 위해, 각 사용자 i 를 (i, v) 라는 복제본 $K + 1$ 개로 나눈다고 생각할 수 있다.

각 복제본은 독립적으로 검증자로 선택될 수 있으며, 고유한 일시적 키 $(pk_{i,v}^{r,s}, sk_{i,v}^{r,s})$ 를 가진다.

- 복제본 수 (Number of Copies).

라운드 r 시점에서 사용자 i 의 보유 금액을 $a_i^{(r)}$, 시스템 전체 자금을 A^r 라고 하자.

즉,

$$A^r = \sum_{i \in PK^{r-k}} a_i^{(r)}.$$

목표 검증자 집합 크기를 n 이라 할 때, 사용자 i 의 복제본 수 $K + 1$ 은

$$K = \left\lfloor \frac{n \cdot a_i^{(r)}}{A^r} \right\rfloor$$

로 정의된다.

예시. $n = 1000$, $A^r = 10^9$, $a_i^{(r)} = 3.7 \times 10^6$ 이라면,

$$K = \left\lfloor \frac{1000 \cdot (3.7 \times 10^6)}{10^9} \right\rfloor = 3$$

이다. 따라서 i 는 $(i, 1), (i, 2), (i, 3), (i, 4)$ 네 개의 복제본을 가진다.

- 검증자와 자격 증명.

각 복제본 (i, v) 는 검증자로 선택될 수 있으며, 자격 증명은

$$\sigma_{i,v}^{r,s} \triangleq SIG_i((i, v), r, s, Q^{r-1})$$

으로 정의된다.

$v \leq K$ 인 복제본은 자동으로 $SV^{r,s}$ 에 포함된다.

$v = K + 1$ 인 마지막 복제본은, 확률적으로 선택된다. 즉,

$$H(SIG_i((i, K + 1), r, s, Q^{r-1})) \leq \frac{a_i^{(r)} n}{A^r} - K$$

이면 포함된다.

따라서, 이 경우에도 검증자 집합은 사용자의 자금 보유량에 비례하여 결정된다.

요약.

정직 사용자 다수 가정을 정직 자금 다수 가정으로 대체하기 위해서는, 검증자 선택 확률을 사용자의 보유 금액에 비례하도록 조정하면 된다. 이렇게 하면 부유한 사용자가 여러 키를 만들어야 하는 비효율을 피할 수 있고, 각 사용자는 자신이 보유한 자금을 비례하여 검증자 선정에 기여하게 된다.

9 포크 처리 (Handling Forks)

우리는 이미 Algorand에서 포크 발생 확률을 10^{-12} 또는 10^{-18} 수준까지 낮추었기 때문에, 실제로 포크가 발생할 가능성은 거의 없다.

따라서 이러한 극히 드문 상황을 별도로 처리할 필요는 사실상 없다. 그럼에도 불구하고, Algorand는 필요하다면 다양한 포크 해결 절차를 사용할 수 있다. 이는 작업 증명(proof-of-work)의 도움을 받거나, 그렇지 않고도 가능하다.

사용자들에게 줄 수 있는 한 가지 지침은 다음과 같다:

- 여러 체인을 동시에 보게 되면, 항상 가장 긴 체인을 따른다.
- 만약 가장 긴 체인이 둘 이상이라면, 그 끝이 **비어 있지 않은 블록**인 체인을 따른다.
 - 끝이 모두 빈 블록인 경우라면, 두 번째로 마지막 블록을 비교한다.
- 만약 끝이 비어 있지 않은 가장 긴 체인이 둘 이상이라면, 예를 들어 그 길이가 r 이라고 하자,
 - 그 경우 블록 r 의 리더 자격 증명이 가장 작은 체인을 따른다.
 - 동률일 경우, 블록 r 자체의 해시 값이 가장 작은 체인을 따른다.
 - 그래도 동률이라면, 블록 r 을 사전식(lexicographic)으로 가장 먼저 오는 체인을 따른다.

요약.

- 포크 발생은 매우 드물지만, 일단 발생하면 위 지침에 따라 해결할 수 있다.
- 본 지침은 비트코인의 "Longest Chain Rule"과 유사하나, 빈 블록 여부와 리더 자격 증명을 함께 고려한다는 점에서 차별화된다.

10 네트워크 분할 처리 (Handling Network Partitions)

앞서 우리는 모든 사용자 간 메시지 전파 시간이 λ 와 Λ 로 상한된다고 가정했다. 이는 강한 가정이 아니다. 오늘날 인터넷은 빠르고 안정적이며, 실제 λ 와 Λ 값도 합리적이다. 여기서는 인터넷이 일시적으로 분할되더라도 Algorand'가 동작함을 보인다. (세 부분 이상으로 분할되는 경우도 유사하게 처리된다.)

10.1 물리적 분할 (Physical Partitions)

먼저, 분할이 물리적 이유로 발생할 수 있다. 예를 들어, 대규모 지진으로 인해 유럽과 미국 간 연결이 완전히 끊어지는 경우이다.

이 경우 악의적 사용자도 함께 분할되므로, 각 부분(partition)에는 독립된 적대자가 존재한다. 각 적대자는 자신이 속한 부분에서만 프로토콜을 공격할 수 있다.

라운드 r 도중 네트워크가 두 부분으로 나뉘었다고 하자. 그러면 여전히 검증자 집합은 PK^{r-k} 에서 선택된다. 이때 파트 1의 검증자 집합을 $HSV_1^{r,s}, MSV_1^{r,s}$ 라 하고, 파트 2의 검증자 집합을 $HSV_2^{r,s}, MSV_2^{r,s}$ 라 하면,

$$|HSV_1^{r,s}| + |MSV_1^{r,s}| + |HSV_2^{r,s}| + |MSV_2^{r,s}| = |HSV^{r,s}| + |MSV^{r,s}|$$

이 성립한다.

만약 어떤 파트 i 에서 $|HSV_i^{r,s}| + |MSV_i^{r,s}| \geq t_H$ 가 될 확률이 무시할 수 없을 만큼 크다면, 전체 확률은 매우 작아진다. 예를 들어 $F = 10^{-18}$ 이면, 그 확률은 10^{-16} 수준이다. 따라서 작은 파트는 사실상 오프라인 상태로 취급할 수 있다.

큰 파트(예: 파트 1)를 살펴보자.

여기서 $|HSV_1^{r,s}| < t_H$ 일 가능성이 완전히 사라지지는 않는다.

이 경우, 적대자는 라운드 r 에서 두 개의 블록을 동시에 만들 수 있다.

예컨대, 코인-0 고정 단계에서 모든 정직 검증자가 블록 B^r 해시에 서명했다고 하자.

그러나 악의적 검증자들이 서명을 보류한 뒤, 다음 코인-1 고정 단계에서 빈 블록 B_e^r 에 서명하면,

결국 두 개의 블록(B^r 과 B_e^r)이 모두 t_H 서명을 갖게 되어 포크가 발생한다.

즉, 라운드 r 에서는 B^r 과 B_e^r 라는 두 블록이 동시에 존재할 수 있다.

그러나 포크는 다음 라운드로 이어지지 않는다. 라운드 $r + 1$ 에서는 두 체인 중 하나만 성장한다.

Algorand'₂의 추가 지침:

사용자가 B^r (비어 있지 않은 블록)과 B_e^r (빈 블록)를 동시에 보게 된다면, 항상 비어 있지 않은 블록을 따른다.

이 규칙에 따라, 큰 파트에서 정직 사용자가 분할을 감지하면, 빈 블록 체인은 곧 성장을 멈춘다.

10.2 적대적 분할 (Adversarial Partition)

네트워크 분할은 적대자에 의해 의도적으로 발생할 수도 있다.

이 경우, 한 파트의 정직 사용자가 전송한 메시지가 다른 파트의 정직 사용자에게 직접 도달하지 않는다.

그러나 일단 한 파트의 메시지가 다른 파트에 도달하면, 그 안에서는 정상적으로 전파된다.

만약 적대자가 충분한 자원을 들여 인터넷을 공격한다면, 이런 분할을 일정 시간 유지할 수 있다.

그러나 분석은 위 물리적 분할과 동일하다.

즉, 일시적으로 두 체인(B^r 과 B_e^r)이 생성될 수 있으나, 동시에 성장하지는 못한다.

따라서 결국 한 체인만 살아남는다.

10.3 네트워크 분할 요약 (Network Partitions in Sum)

네트워크 분할은 발생할 수 있으며, 특정 라운드에서 포크를 일으킬 수 있다.

그러나 포크는 길게 지속되지 않는다. 최악의 경우에도 포크는 한 라운드까지만 유지된다.

즉, 모든 파트 중 하나를 제외한 나머지는 새 블록을 생성할 수 없음을 알게 되고,

사라질 블록에 의존하지 않게 된다.

따라서 Algorand는 네트워크 분할 상황에서도 안전하게 동작한다.

감사의 글 (Acknowledgements)

우선, 인용된 Democoin 시스템의 공동 저자인 Sergey Gorbunov에게 감사를 표한다.

Maurice Herlihy에게는 수많은 통찰력 있는 토론과, 파이프라이닝이 Algorand의 처리량을 향상시킬 수 있음을 지적해준 점, 그리고 본 논문의 초기 버전의 표현을 크게 개선해준 점에 대해 깊이 감사드린다.

Sergio Rajsbaum에게는 초기 버전에 대한 코멘트에 감사드린다.

Vinod Vaikuntanathan에게는 여러 심도 있는 토론과 통찰에 감사드린다.

Yossi Gilad, Rotem Hamo, Georgios Vlachos, Nickolai Zeldovich에게는 이 아이디어들을 실제로 실험해보고, 많은 유용한 코멘트와 토론을 제공해주신 점에 대해 감사드린다.

Silvio Micali는 Ron Rivest에게 개인적인 감사를 전한다.

30년 이상에 걸친 수많은 암호학 연구 토론과 조언, 그리고 인용된 마이크로페이먼트 시스템의 공동 저술을 통해 Algorand의 검증자 선택 메커니즘 중 하나에 영감을 준 점에 대해 감사드린다.

우리는 이 기술을 한 단계 더 발전시키기를 희망한다.

한편, 연구 여정 자체가 큰 즐거움이었으며, 함께한 동료들께 진심으로 감사한다.

참고 문헌 (References)

- [1] Bitcoin Computation Waste, <http://gizmodo.com/the-worlds-most-powerful-computer-network-is-being-was-50> 2013.
- [2] Bitcoinwiki. *Proof of Stake*. <http://www.blockchaintechnologies.com/blockchain-applications> (2016년 6월 기준).
- [3] Coindesk.com. *Bitcoin: A Peer-to-Peer Electronic Cash System*. <http://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet-things/> (2016년 6월 5일 기준).
- [4] Ethereum. *Ethereum*. <https://github.com/ethereum/> (2016년 6월 12일 기준).
- [5] HowStuffWorks.com. *How much actual money is there in the world*. <https://money.howstuffworks.com/how-much-money-is-in-the-world.htm> (2016년 6월 5일 기준).
- [6] 위키백과. *Sortition*. <https://en.wikipedia.org/wiki/Sortition>.
- [7] M. Ben-Or. *Another advantage of free choice: Completely asynchronous agreement protocols*. Proc. 2nd Annual Symposium on Principles of Distributed Computing, ACM, New York, 1983, pp. 27–30.
- [8] M. Castro and B. Liskov. *Practical Byzantine Fault Tolerance*. Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, Louisiana, USA, 1999, pp. 173–186.
- [9] D. L. Chaum. *Random Sample Elections*. <https://www.scribd.com/mobile/document/236881043/Random-Sample-Elections>.
- [10] B. Chor and C. Dwork. *Randomization in Byzantine agreement*, in *Randomness and Computation*. S. Micali, ed., JAI Press, Greenwich, CT, 1989, pp. 432–498.
- [11] C. Decker and R. Wattenhofer. *Information Propagation in the Bitcoin Network*. 13th IEEE Conference on Peer-to-Peer Computing, 2013.
- [12] D. Dolev. *The Byzantine Generals Strike Again*. J. Algorithms, 3, (1982), pp. 14–30.
- [13] D. Dolev and H. R. Strong. *Authenticated algorithms for Byzantine agreement*. SIAM Journal on Computing, 12 (4), pp. 656–666.
- [14] C. Dwork and M. Naor. *Pricing via Processing, Or, Combatting Junk Mail*. Advances in Cryptology, CRYPTO'92: Lecture Notes in Computer Science No. 740. Springer: pp. 139–147.
- [15] P. Feldman and S. Micali. *An Optimal Probabilistic Algorithm for Synchronous Byzantine Agreement*. (Preliminary version in STOC 88.) SIAM J. on Computing, 1997.
- [16] M. Fischer. *The consensus problem in unreliable distributed systems (a brief survey)*. Proc. International Conference on Foundations of Computation, 1983.

- [17] S. Goldwasser, S. Micali, and R. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack*. SIAM Journal of Computing, 17(2), April 1988, pp. 281–308.
- [18] S. Gorbunov and S. Micali. *Democoin: A Publicly Verifiable and Jointly Serviced Cryptocurrency*. <https://eprint.iacr.org/2015/521>, May 30, 2015.
- [19] J. Katz and C-Y Koo. *On Expected Constant-Round Protocols for Byzantine Agreement*. <https://www.cs.umd.edu/~jkatz/papers/BA.pdf>.
- [20] A. Kiayias, A. Russel, B. David, and R. Oliynycov. *Ouroboros: A provably secure proof-of-stake protocol*. Cryptology ePrint Archive, Report 2016/889, 2016. <http://eprint.iacr.org/2016/889>.
- [21] S. King and S. Nadal. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*, 2012.
- [22] D. Lazar and Y. Gilad. 개인 통신 (Personal Communication).
- [23] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
- [24] S. Micali. *Algorand: The Efficient Public Ledger*. <https://arxiv.org/abs/1607.01341>.
- [25] S. Micali. *Fast And Furious Byzantine Agreement*. Innovation in Theoretical Computer Science 2017. Berkeley, CA, January 2017. 단일 페이지 초록.
- [26] S. Micali. *Byzantine Agreement, Made Trivial*. <https://people.csail.mit.edu/silvio/SelectedScientificPapers/BA.pdf>.
- [27] S. Micali, M. Rabin, and S. Vadhan. *Verifiable Random Functions*. 40th Foundations of Computer Science (FOCS), New York, Oct 1999.
- [28] S. Micali and R. L. Rivest. *Micropayments Revisited*. Lecture Notes in Computer Science, Vol. 2271, pp. 149–163, Springer Verlag, 2002.
- [29] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. <http://www.bitcoin.org/bitcoin.pdf>, May 2009.
- [30] R. Pass and E. Shi. *The Sleepy Model of Consensus*. Cryptology ePrint Archive, Feb 2017, Report 2017/918.
- [31] M. Pease, R. Shostak, and L. Lamport. *Reaching agreement in the presence of faults*. J. Assoc. Comput. Mach., 27 (1980), pp. 228–234.
- [32] M. Rabin. *Randomized Byzantine generals*. 24th Foundations of Computer Science (FOCS), IEEE Computer Society Press, Los Alamitos, CA, 1983, pp. 403–409.
- [33] R. Turpin and B. Coan. *Extending binary Byzantine agreement to multivalued Byzantine agreement*. Inform. Process. Lett., 18 (1984), pp. 73–76.

1. 본 문서는 두 번째 저자가 발표한 ArXiv 논문 [24]의 보다 공식적이고(비동기적인) 버전이며, 해당 논문은 Gorbunov와 Micali [18]의 차트를 기반으로 작성되었다. Algorand의 기술은 다음의 특허 출원 대상이다: US62/117,138 US62/120,916 US62/142,318 US62/218,817 US62/314,601 PCT/US2016/018300 US62/296,865 62/331,654 US62/333,340 US62/343,369 US62/344,667 US62/346,775 US62/351,011 US62/653,482 US62/352,195 US62/363,970 US62/369,447 US62/378,753 US62/383,299 US62/394,091 US62/400,361 US62/403,403 US62/410,721 US62/416,995 US62/422,853 US62/455,444 US62/458,746 US62/459,652 US62/460,928 US62/465,931. ↩

2. 비트코인이 제공하는 (가짜) 익명성은 자금세탁이나 범죄-테러 단체의 자금 조달에 악용될 수 있다. 원칙적으로 완벽한 익명성을 제공하는 전통적 지폐나 금괴도 동일한 문제를 야기하지만, 물리적 형태 덕분에 자금 이동 속도가 현저히 느려지고, 이는 어느 정도의 모니터링을 가능하게 한다. 화폐

발행 능력은 국가의 가장 기본적인 권력 중 하나다. 따라서 독립적으로 부유하는 통화가 대규모로 채택된다면, 이 권력을 약화시킬 수 있다. 그러나 현재 비트코인은 정부의 통화 정책을 위협할 단계와는 거리가 멀며, 확장성 문제로 인해 앞으로도 그럴 가능성이 낮다. $\hookrightarrow$

3. 그들의 논문의 초기 버전은 실제로 적대적 수면 모델에서의 보안만 고려했다. Algorand의 초기 버전(그들의 것보다 먼저 나온)은 온라인 사용자 다수가 항상 정직하다는 가정을 상정했으나, 이를 배제하고 느긋한 정직성 모델을 채택했다. (예를 들어, 어느 순간 정직한 사용자의 절반이 오프라인을 선택하면, 온라인 사용자의 다수가 악의적일 수 있다. 따라서 이를 방지하려면 적대자도 자신의 부패한 사용자 다수를 오프라인으로 보내야 하는데, 이는 그의 이해에 반한다.) 다수의 느긋하지만 정직한 사용자만으로도 프로토콜은 잘 동작한다. 이는 온라인 사용자 다수가 항상 악의적이라도, 필요한 순간에 자신이 중요하다는 것을 아는 정직한 사용자들이 오프라인으로 가지 않기 때문이다. 또한 적대자는 누가 중요한 정직 사용자일지 모르기 때문에 그들을 오프라인으로 강제할 수 없다. $\hookrightarrow$

4. 이는 p 가 “효력을 발휘했는지” 확인하는 과정을 단순화한다. 즉, 어떤 지불 집합 PAY^r 이 p 를 포함하는지를 확인하는 절차가 단순해진다. $k = 0$ 일 경우, $p = SIG_i(r, i, i', a, I, H(\mathcal{I}))$ 이고 $p \notin PAY^r$ 라면, i 는 p 를 다시 제출해야 한다. $\hookrightarrow \hookrightarrow$

5. 기본적으로 비트코인과 마찬가지로, 사용자가 메시지 m 을 전파하면, 메시지 m 을 처음 수신하는 각 활성 사용자 i 는 무작위로 독립적으로 소수의 활성 사용자(이웃)를 선택해 m 을 전달한다. 경우에 따라 i 는 그들로부터 확인 응답을 받을 때까지 반복한다. 메시지 전파는 더 이상 새로운 사용자가 m 을 처음으로 수신하지 않을 때 종료된다. $\hookrightarrow$

6. 정직한 비율 h 와 허용 실패 확률 F 가 주어지면, Algorand는 단계별 검증자 최대 수 N 의 상한을 계산한다. 따라서 MP 가정은 $n \leq N$ 인 경우에만 성립하면 충분하다. 또한 MP 가정은 다른 메시지들이 동시에 전파되는 경우에도 성립한다고 서술되어 있다. 하지만 Algorand에서는 메시지가 본질적으로 겹치지 않는 시간 구간에 전파되므로, 해당 시간에는 하나의 블록 또는 최대 N 명의 검증자가 전파하는 소규모 메시지들만 존재한다. 따라서 MP 가정은 더 약화시킬 수 있지만, 그렇게 하면 복잡성이 증가한다. $\hookrightarrow$

7. 예를 들어, 적대자는 정직한 사용자가 보낸 메시지를 즉시 알 수 있다. 따라서 악의적 사용자 i' 가 정직한 사용자 i 와 동시에 메시지를 전파하도록 요청받으면, i' 는 i 가 실제로 전파한 메시지 m 을 본 뒤 자신의 메시지 m' 을 선택해 전송할 수 있다. 이는 분산 계산 문헌에서 “러싱(rushing)”이라 불리는 능력과 관련된다. $\hookrightarrow$

8. 본질적으로, 단계적 브로드캐스트 프로토콜에서 (a) 각 플레이어의 입력은 구분된 송신자의 신원이며, 송신자는 추가적인 개인 입력 값 v 를 가진다. (b) 출력은 단계적 합의와 동일한 조건 1, 2를 만족해야 하고, 또한 다음 조건 3': 송신자가 정직하다면 모든 정직한 플레이어 i 는 $v_i = v$, $g_i = 2$ 를 출력해야 한다. $\hookrightarrow$

9. 실제로 그들의 프로토콜에서, 1단계에서 송신자가 자신의 값 v 를 모든 플레이어에게 전송하고, 각 플레이어 i 는 자신이 송신자로부터 받은 값을 v'_i 로 설정한다. $\hookrightarrow$

10. 지불 집합은 유효한 지불들로 정의되고, 정직한 사용자는 항상 유효한 지불만 생성하므로, 최대 지불 집합 PAY^r 은 모든 정직한 사용자의 “현재 대기 중인” 지불들을 포함한다. $\hookrightarrow$

11. 예를 들어, 적대자가 PAY^{r-1} 을 조작해 특정 잠재 리더의 $SIG_i(r, 1, H(PAY^{r-1}))$ 가 매우 작은 해시 값을 가지도록 유도할 수 있다. 이는 그 리더가 라운드 r 의 리더가 될 확률을 크게 높인다. $\hookrightarrow$

12. 예를 들어, 주요 방송사의 앵커를 부패시켜 “클린턴이 지난 대선에서 승리했다”는 가짜 뉴스 영상을 지금 당장 송출한다고 해도, 대부분의 사람은 조작임을 알아차린다. 그러나 오랜 혼수상태에서 깨어난 사람은 순간 속을 수 있다. $\hookrightarrow$

13. 동기식 시스템이 아닐 경우, “라운드 $r - 1$ 종료”와 “라운드 r 시작”의 개념은 조심스럽게 정의되어야 한다. 수학적으로는 PK^r 과 S^r 가 초기 상태 S^0 과 블록 $B^1, \dots, B^{r-1}$ 로부터 계산된다. $\hookrightarrow$

14. 엄밀히 말해 “ $r - k$ ”는 “ $\max(0, r - k)$ ”여야 한다. $\hookrightarrow$

15. 이 방법에서는, i 가 $\{r', \dots, r' + 10^6\} \times \{1, \dots, m + 3\}$ 의 각 라운드-단계 쌍에 대해 공개/비밀키 쌍 $(pk_i^{r,s}, sk_i^{r,s})$ 를 생성한다. 그런 다음 이 공개키들을 정해진 순서로 배열해 머클 트리의 리프에 저장하고, 루트 값 R_i 를 계산하여 공표한다. 이후 i 가 키 $pk_i^{r,s}$ 에 대응하는 메시지를 서명할 때는, 서명과 함께 $pk_i^{r,s}$ 가 R_i 의 j 번째 리프에 저장되어 있음을 증명하는 인증 경로도 제공한다. 이를 통해 해당 키가 진짜임을 검증할 수 있다. $\hookrightarrow$

16. 본질적으로, 사용자 i 는 개인적으로 라운드 r 의 리더를 ℓ 로 선택한다. $\hookrightarrow \hookrightarrow$

17. $m_\ell^{r,1}$ 의 서명과 해시는 검증되었고, PAY_ℓ^r 은 유효한 지불 집합이지만, i 는 그것이 최대인지 여부를 확인하지 않는다. $\hookrightarrow \hookrightarrow$

18. $m_i^{r,2}$ 는 플레이어 i 가 다음 블록의 해시로 v'_i 를 고려한다는 신호이다. $v'_i = \perp$ 이면 빈 블록을 의미한다. $\hookrightarrow$

19. 동일 플레이어 j 가 비트 0과 1을 동시에 서명한 메시지를 보냈더라도, 조건 (b)에 해당하는 메시지는 인정된다. 이렇게 하는 이유는 모든 정직한 사용자가 λ 시간 내에 동일 블록 B^r 을 알도록 보장하기 위함이다. $\hookrightarrow$

20. 사용자 i 는 B^r 을 알게 되었으므로 라운드 r 을 종료한다. 이후 일반 사용자로서 메시지 전파는 돕지만, (r, s) 검증자로서 새 메시지를 만들지 않는다. $\hookrightarrow$

21. 실제로, $\mathbb{E}[T^{r+1} - T^r] \leq (6E[L'] + 10)\lambda + \Lambda = (6 \cdot \frac{2}{p_h} + 10)\lambda + \Lambda = (\frac{12}{p_h} + 10)\lambda + \Lambda$ 이다. $\hookrightarrow$

22. $|HSV^{r,4}| \geq t_H$ 라는 성질은 파라미터 선택 시 체르노프 경계를 통해 보장된다. $\hookrightarrow$

23. 지속적 참여를 제거하려면, $PK^{r-k-2000}$ 을 사용해야 한다. ↩