(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0088507
(43) 공개일자 2022년06월27일

(51) 국제특허분류(Int. Cl.)
G06Q 20/38 (2012.01) *G06Q 20/06* (2012.01)
G06Q 20/40 (2012.01) *G06Q 40/04* (2012.01)
H04L 9/00 (2022.01) *H04L 9/06* (2006.01)
(52) CPC특허분류
G06Q 20/3827 (2013.01)
G06Q 20/065 (2013.01)
(21) 출원번호 10-2022-7019913(분할)
(22) 출원일자(국제) 2017년05월04일
심사청구일자 2022년06월13일
(62) 원출원 특허 10-2018-7035067
원출원일자(국제) 2017년05월04일
심사청구일자 2020년04월28일
(85) 번역문제출일자 2022년06월13일
(86) 국제출원번호 PCT/US2017/031037
(87) 국제공개번호 WO 2017/192837
국제공개일자 2017년11월09일
(30) 우선권주장
62/331,654 2016년05월04일 미국(US)
(뒷면에 계속)

(71) 출원인
미칼리 실비오
미국 매사추세츠 02445 브룩클린 체스넛 힐 애비뉴 459
(72) 발명자
미칼리 실비오
미국 매사추세츠 02445 브룩클린 체스넛 힐 애비뉴 459
첸 징
미국 매사추세츠 02215 보스턴 보일스톤 스트리트 1350아파트먼트 1405
(74) 대리인
특허법인와이에스장

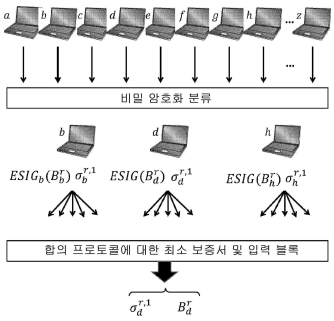
전체 청구항 수 : 총 12 항

(54) 발명의 명칭 **분산 거래 전파 및 검증 시스템**

(57) 요 약

거래가 블록으로 구성되는 거래 시스템에서, 엔티티로 하여금 이전 블록들로부터 수량 Q를 결정하게 하고, 상기 엔티티가 수량 Q와 엔티티에 고유하게 연관된 스트링 S를 계산하기 위해 비밀 키를 사용하기 하고, 상기 엔티티가 S 자체, S의 함수 및/또는 S의 해시 값인 T를 수량 Q로부터 계산하도록 하고, 상기 엔티티가 T가 주어진 속성을 소유하고 있는지 여부를 판정하게 하고, T가 상기 주어진 속성을 소유하는 경우 상기 엔티티가 상기 새로운 블록에 디지털 서명을 하여 S와 상기 새로운 블록의 디지털 서명된 버전을 사용가능하게 함으로써, 상기 엔티티가 이전 블록의 시퀀스에 대하여 새로운 블록들의 유효 거래를 구성하도록 한다. 상기 비밀 키는 상기 엔티티의 공개 키에 대응하는 비밀 서명 키일 수 있다. S는 상기 엔티티에 의한 Q의 디지털 서명 일 수 있다.

대 표 도



(52) CPC특허분류
G06Q 20/3825 (2013.01)
G06Q 20/3829 (2013.01)
G06Q 20/4016 (2013.01)
G06Q 40/04 (2013.01)
H04L 9/0643 (2013.01)
H04L 9/50 (2022.05)
H04L 2209/56 (2013.01)

(30) 우선권주장
62/333,340 2016년05월09일 미국(US)
62/343,369 2016년05월31일 미국(US)
62/344,667 2016년06월02일 미국(US)
62/346,775 2016년06월07일 미국(US)
62/351,011 2016년06월16일 미국(US)
62/353,482 2016년06월22일 미국(US)
62/354,195 2016년06월24일 미국(US)
62/363,970 2016년07월19일 미국(US)
62/369,447 2016년08월01일 미국(US)
62/378,753 2016년08월24일 미국(US)
62/383,299 2016년09월02일 미국(US)
62/394,091 2016년09월13일 미국(US)
62/400,361 2016년09월27일 미국(US)
62/403,403 2016년10월03일 미국(US)
62/410,721 2016년10월20일 미국(US)
62/416,959 2016년11월03일 미국(US)
62/422,883 2016년11월16일 미국(US)
62/455,444 2017년02월06일 미국(US)
62/458,746 2017년02월14일 미국(US)
62/459,652 2017년02월16일 미국(US)

명 세 서

청구범위

청구항 1

인증 기관과 연관된 하나 이상의 컴퓨터 디바이스에 의해 수행되는 방법으로서, 상기 방법은:

제1 엔티티가 암호화폐 거래를 수행하도록 하는 요청을 상기 제1 엔티티로부터 수신하는 단계 - 상기 요청은 상기 제1 엔티티를 식별하는 정보와 상기 제1 엔티티의 공개키를 포함함-;

상기 제1 엔티티의 공개키의 디지털 서명을 생성하는 단계 - 상기 디지털 서명은 상기 인증 기관의 비밀키를 이용하여 생성됨-;

상기 제1 엔티티와 연관된 인증서를 저장하는 단계 - 상기 인증서는 상기 제1 엔티티의 공개키 및 상기 디지털 서명을 포함함-;

이후, 상기 제1 엔티티에 의해 서명된 거래를 수신하는 단계 - 상기 거래는 암호화폐의 양과 제2 엔티티를 식별함-;

상기 제1 엔티티가 적어도 상기 암호화폐의 양과 연관되어 있는 지를 결정하는 단계;

상기 제1 엔티티 및 제2 엔티티가 각각의 대응하는 인증서와 연관되어 있는 지를 결정하는 단계; 및

상기 결정에 따라, 상기 거래가 상기 암호화폐와 연관된 블록체인에 포스트되도록 허가하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 2

제1항에 있어서,

상기 방법은:

제2 엔티티가 암호화폐 거래를 수행하도록 하는 요청을 상기 제2 엔티티로부터 수신하는 단계 - 상기 요청은 상기 제2 엔티티를 식별하는 정보와 상기 제2 엔티티의 공개키를 포함함-;

상기 제2 엔티티의 공개키의 디지털 서명을 생성하는 단계;

상기 제2 엔티티와 연관된 인증서를 저장하는 단계 - 상기 인증서는 상기 제2 엔티티의 공개키 및 상기 디지털 서명을 포함함-;

를 더 포함하는 것을 특징으로 하는 방법.

청구항 3

제1항에 있어서,

상기 인증서는 상기 제1 엔티티에 대한 정보를 식별하는 것을 포함하는 것을 특징으로 하는 방법.

청구항 4

제1항에 있어서,

상기 인증서는 상기 제1 엔티티에 대한 정보를 식별하는 것에 기초하여 생성된 해시 값을 포함하는 것을 특징으로 하는 방법.

청구항 5

제1항에 있어서,

상기 인증서는 상기 인증서의 발행일을 포함하는 것을 특징으로 하는 방법.

청구항 6

제1항에 있어서,

상기 인증서는 상기 인증서의 만료일을 포함하는 것을 특징으로 하는 방법.

청구항 7

제1항에 있어서,

상기 인증서는 상기 인증서가 만료되지 않았다는 지시를 포함하는 것을 특징으로 하는 방법.

청구항 8

제1항에 있어서,

상기 제1 엔티티와 연관된 인증서가 유효하다는 결정과 상기 제2 엔티티와 연관된 인증서가 유효하다는 결정에 응답하여, 상기 거래가 상기 암호화폐와 연관된 블록체인에 포스트되도록 허가되는 것을 특징으로 하는 방법.

청구항 9

인증 기관과 연관된 하나 이상의 컴퓨터 디바이스에 의해 수행되는 방법으로서, 상기 방법은:

제1 엔티티가 암호화폐 거래를 수행하도록 하는 요청을 상기 제1 엔티티로부터 수신하는 단계 - 상기 요청은 상기 제1 엔티티를 식별하는 정보와 상기 제1 엔티티의 공개키를 포함함-;

상기 제1 엔티티의 공개키의 디지털 서명을 생성하는 단계 - 상기 디지털 서명은 상기 인증 기관의 비밀키를 이용하여 생성됨-;

상기 제1 엔티티와 연관된 인증서를 저장하는 단계 - 상기 인증서는 상기 제1 엔티티의 공개키 및 상기 디지털 서명을 포함함-;

이후, 상기 제1 엔티티에 의해 서명된 거래를 수신하는 단계 - 상기 거래는 암호화폐의 양과 제2 엔티티를 식별함-;

상기 제1 엔티티가 적어도 상기 암호화폐의 양과 연관되어 있는 지를 결정하는 단계;

상기 제1 엔티티 및 제2 엔티티가 각각의 대응하는 인증서와 연관되어 있는 지를 결정하는 단계; 및

상기 결정에 응답하여, 상기 거래를 거부하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 10

인증 기관과 연관된 하나 이상의 컴퓨터 디바이스에 의해 수행되는 방법으로서, 상기 방법은:

제1 엔티티가 암호화폐 거래를 수행하도록 하는 요청을 상기 제1 엔티티로부터 수신하는 단계 - 상기 요청은 상기 제1 엔티티를 식별하는 정보와 상기 제1 엔티티의 공개키를 포함함-;

상기 제1 엔티티를 식별하는 정보로부터 암호문을 생성하는 단계 -상기 암호문은 외부 기관의 암호키를 이용하고 상기 인증 기관에 의해 보유됨-;

상기 제1 엔티티의 공개키 및 상기 암호문을 조합의 디지털 서명을 생성하는 단계 - 상기 디지털 서명은 상기 인증 기관의 비밀키를 이용하여 생성됨-;

상기 제1 엔티티와 연관된 인증서를 저장하는 단계 - 상기 인증서는 상기 제1 엔티티의 공개키 및 상기 디지털 서명을 포함함-;

이후, 상기 제1 엔티티에 의해 서명된 거래를 수신하는 단계 - 상기 거래는 암호화폐의 양과 제2 엔티티를 식별함-;

상기 제1 엔티티가 적어도 상기 암호화폐의 양과 연관되어 있는 지를 결정하는 단계;

상기 제1 엔티티 및 제2 엔티티가 각각의 대응하는 인증서와 연관되어 있는 지를 결정하는 단계; 및

상기 결정에 따라, 상기 거래가 상기 암호화폐와 연관된 블록체인에 포스트되도록 허가하는 단계를 포함하는 것

을 특징으로 하는 방법.

청구항 11

제1항 내지 제10항 중 어느 한 항의 방법을 수행하도록 구성된 하나 이상의 컴퓨터 프로세서를 포함하는 시스템.

청구항 12

하나 이상의 컴퓨터 프로세서에 의해 실행될 때, 상기 하나 이상의 컴퓨터 프로세서로 하여금 제1항 내지 제10항 중 어느 한 항의 방법을 수행하도록 하는 명령어를 저장하는 컴퓨터 판독 저장 매체.

발명의 설명

기 술 분 야

[0001] (관련 출원에 대한 상호 참조)

[0002] 본 출원은 2016년 5월 4일자로 출원된 "Algorand: 매우 효율적인 화폐 플랫폼"이라는 제하의 미국 가특허 출원번호 제62/331,654; 2016년 5월 9일자로 출원된 "Algorand: 매우 효율적인 화폐 플랫폼"이라는 제하의 미국 가특허 출원번호 제62/333,340; 2016년 5월 31일자로 출원된 "Algorand: 효율적인 블록체인"이라는 제하의 미국 가특허 출원번호 제62/343,369; 2016년 6월 2일자로 출원된 "Algorand: 효율적인 블록체인"이라는 제하의 미국 가특허 출원번호 제62/344,667; 2016년 6월 7일자로 출원된 "Algorand 효율적인 블록체인"이라는 제하의 미국 가특허 출원번호 제62/346,775; 2016년 6월 16일자로 출원된 "Algorand 효율적이고 민주적인 원장"이라는 제하의 미국 가특허 출원번호 제62/351,011; 2016년 6월 22일자로 출원된 "Algorand 효율적이고 민주적인 원장"이라는 제하의 미국 가특허 출원번호 제62/353,482; 2016년 6월 24일자로 출원된 "Algorand 효율적이고 민주적인 원장"이라는 제하의 미국 가특허 출원번호 제62/354,195; 2016년 7월 19일자로 출원된 "Algorand 효율적이고 민주적인 원장"이라는 제하의 미국 가특허 출원번호 제62/363,970; 2016년 8월 1일자로 출원된 "Algorand 효율적이고 민주적인 원장"이라는 제하의 미국 가특허 출원번호 제62/369,447; 2016년 8월 24일자로 출원된 "Algorand 효율적인 공개 원장"이라는 제하의 미국 가특허 출원번호 제62/378,753; 2016년 9월 2일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/383,299; 2016년 9월 13일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/394,091; 및 2016년 9월 27일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/400,361; 2016년 10월 3일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/403,403; 2016년 10월 20일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/410,721; 2016년 11월 3일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/416,959; 2016년 11월 16일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/422,883; 2017년 2월 6일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/455,444; 2017년 2월 14일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/458,746; 및 2017년 2월 16일자로 출원된 "Algorand"라는 제하의 미국 가특허 출원번호 제62/459,652의 우선권을 주장하며, 이는 모두 참조에 의해 본원에 통합된다

[0003] (기술 분야)

[0004] 본 출원은 전자 거래 분야에 관한 것이고, 특히, 거래 블록의 콘텐츠의 시퀀스 및 전자 결제의 검증을 보호하는 분산 공개 원장(public ledgers) 분야에 관한 것이다

배 경 기 술

[0005] 공개 원장은 모든 사람이 읽고 증강(augment)할 수 있는 데이터의 조작방지(tamperproof) 시퀀스이다. 공유된 공개 원장은 현대 사회가 운영하는 방식에 혁명을 일으킬 수 있다. 결제, 자산 이전 및 타이틀링(titling)과 같은 모든 종류의 전통적인 거래를 그것들이 발생한 정확한 순서대로 보호할 수 있고; 암호화폐(cryptocurrencies) 및 스마트 계약과 같은 완전히 새로운 거래를 가능하게 한다. 그것들은 부패를 억제하고 중개자(intermediary)를 제거하며 신뢰를 위한 새로운 패러다임을 안내할 수 있다. 결제 시스템과 암호화폐를 구축하기 위한 공개 원장의 사용은 특히 중요하고 매력적이다.

[0006] 비트코인 및 기타 이전의 분산 시스템에서, 사용자는 디지털 서명 체계의 공개키를 가지고 있다(해당 비밀 키를 알고 있다). 각 키를 인간화하고 그것을 사용자로 생각하는 것이 유용하다. 실제로, 시스템에서 공개키는 알려져 있지만 키 뒤에 있는 사용자는 직접 알 수 없으므로 사용자에게 일정 수준의 개인 정보가 제공된다. 돈은 각

공개키와 직접 연관된다. 각 시점에서, 지금까지 블록 순서에 대해 추적할 수 있는 바와 같이, 각 공개키는 주어진 금액을 소유한다. 하나의 공개키("지불자(payer)")에서 다른 공개키("수취인(payee)")로의 결제는 지불자가 디지털 방식으로 서명한다. 마찬가지로 모든 거래는 디지털 방식으로 서명된다. 또한, 유효하기 위해서는, 결제는 그 시간에 지불자가 소유한 금액을 초과하지 않는 금액을 이전해야 한다. 결제(더 일반적으로 거래)는 블록으로 구성된다. 모든 자신의 결제(및 거래)가 일괄하여 유효한 경우 블록이 유효하다. 즉, 주어진 블록에서의 임의의 지불자에 의해 결제된 총 금액은 지불자가 사용할 수 있는 금액을 초과하지 않는다. 비트코인은 비허가형(permissionless) 시스템이다. 즉, 새로운 사용자가 시스템에 자유롭게 참여할 수 있다. 새로운 사용자가 (유효한) 블록에서 결제의 수취인으로 나타나면 시스템에 참여한다. 따라서 비트코인에서, 사용자는 자신이 다른 유형의 결제에 사용할 수 있는 여러 개의 키를 소유함으로써 자신이 즐기는 프이버시를 향상시킬 수 있다. 비허가형 시스템에서 그는 자신이 소유한 키에서 새 키로 일부 돈을 이전함으로써 자신의 키의 수를 쉽게 늘릴 수 있다. 비허가는 중요한 속성이다. 비허가형 시스템은 권한이 부여된 시스템으로도 작동할 수 있지만, 그 반대는 참(true)일 필요가 없다. 권한이 부여된 시스템에서, 새 사용자는 자동으로 가입할 수 없지만 승인을 받아야 한다. (권한이 부여된 시스템의 특수한 경우는 사용자 세트가 고정되어있는 경우이다.) 비허가형 시스템은 더 적용 가능하고 현실적이지만 더욱 어려워진다. 비트코인 및 유사한 분산 시스템에서, 사용자는 메시지를 전파(즉, "가쉽핑(gossiping)")함으로써 통신한다. 메시지는 무작위로 선택된 몇 개의 "이웃"에게 전송되며, 각 이웃은 그런 다음 몇 개의 랜덤한 이웃으로 그것들을 전송한다. 루프를 피하기 위해 동일한 메시지를 두 번 보내지 않는다. 분산(또는 공유) 원장은 지금까지 생성된(유효한) 거래의 블록 순서로 구성된다. 일반적으로 제1 블록인, 제네시스 블록(genesis block)은 시스템 사양의 일부에 의해 공용 지식으로 가정된다. 공유 원장은 새로운 블록이 "생성"되는 방식에서 차이가 있다.

[0007] 그러나 현재 구현된 바와 같이, 공개 원장은 엄청난 잠재력을 달성하지 못한다. 중앙집중화된 공개 원장은 주어진 키에 의해 이루어지는 결제를 공개화하는 것을 임의로 거부할 수 있고 사이버 공격에 매우 취약한 단일 엔티티에 대해 모든 신뢰를 둔다. 실제로 단일 중앙 권한이 손상되면 전체 시스템도 손상된다. 비트코인과 같은 분산화된(decentralized) 시스템은 운영 비용이 비싸고 계산 및 기타 가치있는 자원을 낭비하고, 새로운 엔티티(채굴자(miner))의 손에 힘을 집중시키며, 상당한 모호함(포크)으로 고통을 겪고, 긴 대기 시간과 작은 쓰루풋을 갖는다. 다른 분산화된 구현은 허가를 받거나 부정을 저지르는(misbehaving) 사용자를 처벌할 수 있는 기능을 가정하거나, 그 모두에 해당하고, 및/또는 사용자의 일부 서브셋이 적절한 긴 시간동안 사이버 공격으로부터 면역이 되어있다고 신뢰한다.

발명의 내용

해결하려는 과제

[0008] 따라서 중앙 권위자를 신뢰할 필요가 없고 공지된 분산화 접근법의 비 효율성 및 불안정성을 겪지 않는 공개 원장 및 전자 화폐 시스템을 제공하는 것이 바람직하다.

과제의 해결 수단

[0009] 본 발명의 일 실시예에 따른 방법에 따르면, 블록체인 시스템의 제1 엔티티로 하여금 상기 블록체인 시스템과 연관된 다른 엔티티에 대해 상기 제1 엔티티가 주어진 태스크를 수행하기 위해 선택되었다는 것을 입증할 수 있게 하는 방법으로서, 상기 블록체인 시스템의 상기 제1 엔티티에 의해 수행되는, 제1 암호화 절차(cryptographic procedure)를 이용하여, 블록체인의 블록들의 미리 존재하는 시퀀스에 기초하여 제1 스트링을 연산하는 단계; 비밀 암호화 키(secret cryptographic key)를 이용하여, 상기 제1 스트링에 고유하게 연관되는 증명(proof) 스트링을 연산하는 단계; 제2 암호화 절차를 이용하여, 상기 증명 스트링에 기초하여 숫자인 양(numerical quantity)을 연산하는 단계; 상기 숫자인 양이 주어진 수치 임계치를 충족하는지를 판정하는 단계; 및 상기 판정에 응답하여, 상기 블록체인에 대해 상기 증명 스트링을 전파하는 단계;를 포함하는 것을 특징으로 한다.

[0010] 본 명세서에 설명된 시스템에 따르면, 거래들이 블록들로 구성되는 거래 시스템에서, 엔티티가 이전 블록들로부터 수량 Q를 판정하게 하고, 엔티티가 Q와 상기 엔티티에 고유하게 연관된 스트링 S를 계산하기 위해 비밀 키를 사용하게 하고, 상기 엔티티가 S로부터 S 자체, S의 함수, 및 S의 해시 값 중 적어도 하나인 수량 T를 연산하게 하고, 상기 엔티티가 T가 주어진 속성을 소유하고 있는지를 판정하도록 하고, T가 상기 주어진 속성을 소유하고 있으면 상기 엔티티가 B^{r}에 디지털 서명을 하여 S와 B^{r}의 디지털 서명 버전을 가용하게 하도록 함으로써, 상기

엔티티는 이전 블록들, B^0, ..., B^{r-1}의 시퀀스에 대해 유효 거래들의 새로운 블록 B^r을 구축한다. 비밀 키는 엔티티의 공개키에 대응하는 비밀 서명 키일 수 있고, S는 엔티티에 의한 Q의 디지털 서명일 수 있다. T는 숫자의 이진 확장이 될 수 있으며 T가 주어진 수 p보다 작으면 속성을 만족시킬 수 있다. S는 B^r로부터 S를 추론할 수 있게(deducible) 만들어서 사용할 수 있다. 각 사용자는 거래 시스템에서 잔고(balance)를 가질 수 있으며 p는 각 사용자의 잔고에 따라 각 사용자마다 다를 수 있다.

[0011] 본 명세서에 설명된 시스템에 따르면, 거래들이 블록들로 구성되고 블록들이 디지털 서명들의 세트에 의해 승인되는 거래 시스템에서, 엔티티는 이전 블록들, B^0, ..., B^{r-1}의 시퀀스가 주어지면, 상기 엔티티가 상기 이전 블록들로부터 수량 Q를 판정하게 하고, 상기 엔티티가 Q의 디지털 서명 S를 계산하도록 하고, 상기 엔티티가 S로부터 S 자체, S의 함수, 및 S의 해시 값 중 적어도 하나인 수량 T를 계산하게 하고, 상기 엔티티가 T가 주어진 속성을 소유하고 있는지를 판정하도록 하고, T가 상기 주어진 속성을 소유하고 있으면 상기 엔티티가 또한 다른 사람들에게 대해 S를 가용하게 하도록 함으로써, 새로운 블록 B^r의 거래를 승인한다. T는 숫자의 이진 확장이 될 수 있으며 T가 미리 정의된 임계값 p보다 작으면 주어진 속성을 만족시킬 수 있고, 상기 엔티티는 또한 S를 가용하게 할 수 있다. 상기 엔티티는 상기 거래 시스템에서 잔고를 가질 수 있으며 p는 엔티티의 잔고에 따라 변할 수 있다. 상기 엔티티는 적어도 다른 엔티티에 대한 권한 있는 대리인의 역할을 할 수 있다. p의 값은 상기 엔티티의 잔고 및/또는 상기 엔티티의 잔고와 다른 엔티티의 잔고의 조합에 따를 수 있다. 다른 사용자는 디지털 서명으로 사용자를 인증할 수 있다. B^r이 주어진 엔티티 세트에 의해 실행되는 비잔틴 합의 프로토콜의 출력인 경우에만 상기 엔티티는 B^r에 디지털 서명할 수 있다. 엔티티들 중 특정 엔티티의 디지털 서명이 주어진 속성을 만족하는 이전 블록들에 의해 결정된 수량을 갖는다면 상기 엔티티들 중 특정 엔티티는 주어진 엔티티 세트에 속할 수 있다.

[0012] 본원에 설명된 시스템에 따르면, 생성되고 디지털 서명된 블록, B^0, ..., B^{r-1}의 시퀀스로 거래가 구성되는 거래 시스템에서(여기서, 각 블록 B^r은 보호되어야 하고 보안(securing) 정보 S^r를 포함하는 일부 정보 $INFO^r$을 포함), 새로운 블록 B^i이 생성될 때마다 블록의 콘텐츠가 검출 불가능하게 변경되는 것을 방지하고, B^i의 정보 $INFO^i$를 이진 트리의 리프 i로 삽입하고, 머클트리, T^i를 획득하기 위해 이진 트리를 머클리파잉하고, T_i의 루트의 콘텐츠 R_i 및 T_i의 리프 i의 콘텐츠의 인증 경로를 포함하도록 블록 B^i의 보안 정보 S^i을 판정한다. 선행 블록 B^{i1}의 보안 정보 S^{i1}가 저장될 수 있고, 보안 정보 S^i는 미리 정해진 시퀀스로, S^{i1}의 값, $INFO^i$의 해시, 및 주어진 값 중 적어도 하나를 포함하는 세트로부터의 값을 해싱함으로써 얻어질 수 있다. 제1 엔티티는 제2 엔티티가 머클 트리 T^z에서 $INFO^i$의 인증 경로를 수신하도록 함으로써 블록 B^z에 선행하는 블록 B^r의 정보 $INFO^r$이 인증된다는 것을 블록 B^z의 보안 정보 S^z를 갖는 제2 엔티티에 대해 증명할 수 있다.

[0013] 본 명세서에 설명된 시스템에 따라, 제1 세트의 사용자 결제가 제1 디지털 서명된 블록 B^1로 수집되고, 제2 세트의 사용자 결제는 B^1 이후에 가용하게 되는 제2 디지털 서명된 블록 B^2로 수집되는 등의, 사용자가 잔고를 가지고 디지털 서명된 결제를 통해 서로 돈을 이전하고 최초 세트의 사용자의 잔고가 공지되는 결재 시스템에서, 엔티티 E는, 블록 시퀀스 B^0, ..., B^{r-1}에서 지정된 정보로부터 추론할 수 있는 정보로부터 모든 사용자 x에 대해 금액 a_x를 계산하고, 가용하게 되는 r번째 블록 B^r의 시간에서 시스템 내의 사용자의 수 n을 연산하고, 각각의 사용자 x에 대해 주어진 순서로 사용자 x를 순서화하고, x가 주어진 순서에서 i번째 사용자인 경우 a_x를 적어도 n개의 리프로 이진 트리 T의 리프 i에 저장하고, T의 루트에 저장된 값 R을 연산하기 위해 트리 T에 대한 머클 값을 판정하고, R을 인증하는 디지털 서명 S를 산출하고, 및 리프 l과 T의 루트 사이의 경로에서 노드의 형제(sibling)인 모든 노드의 콘텐츠를 제공하여 S를 T의 임의의 리프 i의 콘텐츠의 증거로서 가용하게 함으로써, r번째 블록 B^r의 시간에 사용자 i가 수행하고 수신한 모든 결제 후에 사용자 i에게 가용한 잔고 ari에 관한 검증된 정보를 제공한다.

[0014] 본 명세서에 더 설명된 시스템에 따라, 제1 세트의 사용자 결제가 제1 디지털 서명된 블록 B^1으로 수집되고, 제

2 세트의 사용자 결제는 B^1 이후에 가용하게 되는 제2 디지털 서명된 블록 B^2으로 수집되는 등의, 사용자가 잔고를 가지고 디지털 서명된 결제를 통해 서로 돈을 이전하고 사용자의 최초의 세트의 잔고가 공지되는 결제 시스템에서, 제1 r번째 블록의 결제 후에 각각의 사용자 i의 상기 잔고를 판정하고, 각각의 사용자의 잔고가 머클-밸런싱-검색-트리 T^r의 적어도 하나의 노드의 보안 되는 값이 되는 T^r을 생성하고, 상기 엔티티의 세트의 각각의 멤버로 하여금 T^r의 루트의 보안 값 hv_ε을 포함하는 정보의 디지털 서명을 생성하도록 하고, 및 상기 제1 r의 결제 후에 상기 사용자의 적어도 하나의 잔고를 증명하는 hv_ε의 디지털 서명을 제공함으로써, 엔티티 E의 세트는 r번째 블록, B^r의 시간에 사용자 i가 수행하고 수신한 모든 결제 후에 사용자 i가 가용한 잔고 a_i를 증명할 수 있는 정보를 제공한다. 상기 엔티티의 세트는 하나의 엔티티로 구성될 수 있다. 상기 엔티티의 세트는 그의 디지털 서명 값에 기초하여 선택될 수 있다.

[0015] 본 명세서에 더 설명된 시스템에 따라, 제1 세트의 사용자 결제가 제1 디지털 서명된 블록 B^1으로 수집되고, 제2 세트의 사용자 결제는 B^1 이후에 가용하게 되는 제2 디지털 서명된 블록 B^2으로 수집되는 등의, 사용자가 잔고를 가지고 디지털 서명된 결제를 통해 서로 돈을 이전하고 사용자의 최초의 세트의 잔고가 공지되는 결제 시스템에서, 각각의 사용자의 잔고가 머클-밸런싱-검색-트리 T^r의 적어도 하나의 노드의 정보 값이 되는 T^r의 루트의 보안 정보 hv_ε의 엔티티 세트의 멤버의 디지털 서명을 획득하고, 인증 경로 및 사용자 i에 대해 T^r에서 검색하도록 주어진 검색 알고리즘이 처리하는 모든 노드의 콘텐츠를 계산하고, 및 또다른 엔티티로 하여금 i의 잔고를 검증할 수 있도록 하기 위해 인증 경로 및 콘텐츠, 및 디지털 서명을 제공함으로써, 엔티티 E는 r번째 블록, B^r의 시간에 사용자 i가 수행하고 수신한 모든 결제 후에 사용자 i가 가용한 잔고 a_i를 증명한다.

[0016] 본 명세서에 더 설명된 시스템에 따라, 일시적이지 않은 컴퓨터 판독 가능 매체로 제공된 컴퓨터 소프트웨어는 여기에 설명된 방법 중 임의의 것을 구현하는 실행 가능 코드를 포함한다.

[0017] 공유된 공개 원장은 콘텐츠가 안전하고 쉽게 증명될 수 있는 거래 블록의 공통 시퀀스를 생성해야 한다. 따라서, 본 발명은 또한 자신들의 콘텐츠를 용이하게 입증할 수 있는 블록의 시퀀스의 콘텐츠, 블록트리를 보호하기 위한 새로운 방법으로 블록의 공통 시퀀스, Algorand를 생성하는 새로운 방법을 제공한다.

[0018] **Algorand**

[0019] 높은 수준에서, B^1, ..., B^{r-1}을 제1 r-1 블록에 두면, Algorand에서 새로운 블록 B^r이 하기의 2개의(메가) 단계에 의해 생성된다.

[0020] · 사용자, 블록 리더 ℓ^r가 랜덤하게 선택되고 태스크와 함께 신뢰되어 유효한 거래의 새 블록 B^r을 어셈블리하고, 이를 인증하고, 그것을 네트워크를 통해 전파한다. 정직한 리더 ℓ^r가 제안한 경우, 블록 B^r은 ℓ^r에 의해 보여지는 새로운 유효한 거래를 모두 포함한다.

[0021] · 집합적으로 "위원회"로 간주되는 사용자 C^r의 서브셋은 랜덤하게 선택되고 ℓ^r가 제안한 블록의 합의에 도달하고, 그것을 인증하고, 네트워크에 자신들의 인증을 전파하도록 노력한다. 블록은 그것이 적절하게 인증된 경우, 즉 적어도 주어진 수의 C^r의 입증된 멤버에 의해 디지털 서명된 경우에만 공개 원장에 기입한다.

[0022] 우리는 공개 키가 어떻게 선택되는지, 어떻게 C^r이 합의에 도달하는지, 그리고 블록 인증에 필요한 주어진 수의 디지털 서명이 어떻게 선택되는지를 자세히 설명한다.

[0023] 바람직하게는, 공개키는 비밀 암호 분류(cryptographic sortition)를 통해 블록 B^r에 대한 위원회의 리더 또는 멤버로서 선택된다. 본질적으로 사용자는 자신의 추첨을 실행해 자기 자신을 선택하여, (a) 자신이(자신이 원하더라도) 속임수를 쓸 수 없고 자신이 계획한 것보다 더 높은 확률로 자신을 선택할 수 없고; (b) 모든 사람이 검사할 수 있는, 선택되는 증명 P_i를 얻고; (c) 그는 자신이 선택되었다는 것을 알 수 있는 유일한 사람(다른 사람에게 자신의 증명 P_i를 보여줌으로써 이를 밝히도록 결정할 때까지)이 되도록 한다.

[0024] 자신이 소유한 공개키가 선택될 확률을 높이기 위해, 사용자는 자신이 소유한 공개키의 수를 늘리려는(소위 Sybil 공격) 유혹을 받을 수 있고, 이는 이미 논의된 것처럼 비허가형 시스템에서 매우 쉽게 수행할 수 있다. 그러한 공격을 방지하기 위해, 선택된 비밀 암호 분류는 주어진 공개키가 선택될 확률이 자신이 소유한 돈에 비례한다는 것을 보장한다. 이 방법을 사용하면 사용자는 자신의 모든 돈을 하나의 키와 관련되도록 유지할지 또는 그것을 여러 키에 분산할지를 자신의 공개키 중 하나가 선택될 동일한 확률을 갖는다. (특히, 단일 키에서 $1M 소유권을 가진 사용자 또는 각각 $1을 가지는 1M 키를 소유한 사용자는 동일한 선택 기회를 갖는다.)

[0025] 보다 정확하게는, 자신이 소유한 공개키 pk가 리더 또는 위원회 멤버로서 선택되는지를 결정하기 위해, 사용자는 모든 이전 블록으로부터 도출 가능한 일부 정보를 디지털 방식으로(대응하는 비밀 키 sk를 통해) 서명한다. 바람직한 실시 예에서, 그는 최종 블록 B^{r-1}로부터 도출 가능한 양 Q^r을 실제로 디지털 서명하고, 그것을 해싱하고, 및 해시가 주어진 (확률적으로) 임계 값 p(이는 사용자의 공개키에 의해 소유된 금액에 따른다)보다 적은지를 체크한다. 이 경우에, 사용자의 이러한 디지털 서명은 공개키 pk가 선택되었다는 증거로서 사용될 수 있다. 실제로 모든 사람이 사용자 서명을 확인하고, 그것을 해시한 다음, 주어진 임계 값 p와 비교할 수 있다.

[0026] 선택 임계 값 p를 변경함으로써 시스템은 각 역할에 대해 선택된 사용자 수를 제어한다. 먼저 위원회 C^r의 공개키를 선택하는 것을 고려해 보자. 예를 들어, 시스템에서 1M 키가 있고 그 각각은 동일한 금액을 가지고 있으며, C^R에 대략 1K 멤버가 있다고 가정한다. 그런 다음 p = 1/1,000을 공통 선택 임계 값으로 선택할 수 있다. 실제로, 서명의 해시는 0과 1 사이의 난수의 이진 확장으로 간주될 수 있고, 따라서 p = 1/1,000과 같은 확률로 1/1,000보다 작거나 같다. 따라서, 선택된 사용자의 총 예측되는 수는 1M/1K = 1/1K가 된다. 우리가 볼 수 있듯이, 상이한 공개키들이 상이한 금액의 돈을 소유하고 있을 때 주어진 수의 위원회 멤버를 선택하는 것을 달성할 수 있다.

[0027] r번째 블록에 대해 단일 리더 ℓ^r를 선택하는 것을 고려해보자. 그런 다음, 시스템은 먼저 비밀 암호 분류를 사용하여 100개의 공개키를 선택하고, 리더가(해시된) 증거가 더 작은 공개키가 되도록 할 수 있다. 더 정확히 말하자면, 간단하게하기 위해, 선택된 키는 pk_1, ..., pk_{100}이고, 그것들의 대응하는 선택의 증명은 P_1, ..., P_{100}라고 가정한다. 그런 다음, 선택된 각 키의 소유자, i는 새로운 유효한 거래의 자신의 블록, B_i^r을 어셈블리하고 P_i와 적절하게 인증된 블록 B_i^r를 전파한다. 그런 다음 리더 ℓ^r는 자신의 증명이 사전학적으로 (lexicographically) 더 작은 크기의 키가 될 것이고, 블록 B^r은 위원회 C^r이 ℓ^r에 의해 제안된 블록으로 합의에 도달하는 블록이 될 것이다.

[0028] 블록 리더를 선택하기 위해 비밀 암호 분류를 사용하는 것의 큰 이점은 시스템의 활동을 모니터하고 원하는 임의의 사용자를 성공적으로 제어할 수 있는 적(adversary)들이 ℓ^r가 제안한 블록을 선택하기 위해 리더 ℓ^r을 쉽게 손상시킬 수 없다는 것이다. 사실, 100명의 잠재적인 리더가 비밀리에 스스로를 선택할 때 자신의 복권을 운영하는 것에 의한다. 따라서 적들은 (예측되는) 100개의 공개키가 누구가 될지를 알지 못한다. 그러나, 일단 각각의 i가 자신의 보증서와 제안된 블록을 전파하면, 적은 리더 ℓ^r이 누구인지를 빨리 알 수 있다. 그러나 그 시점에서 그를 손상시키는 데는 거의 이점이 없다. 그의 블록과 증거는 네트워크를 통해 바이러스같이 전파되어 중단될 수 없다. (심지어 국가의 주조차도 WikiLeaks에 의해 보내진 메시지를 되담을 수는 없었다.) 한 번의 마지막 이동은 적에게 여전히 가능할 수 있다. 즉, 그는 ℓ^r을 손상시켜 그를 제어하고 그가 다른 블록을 어셈블리하고, 인증하고 전파하도록 강요할 수 있다. 그러나 이 기회조차도 적에게는 닫혀있을 것이다. 실제로 사용자는 자신의 결제 및 거래를 인증하고 자신의 선택 증거를 생성하는 데 사용하는 자신의 비밀 서명 키를 보관한다(선택된 때 및 선택한 경우). 그러나 그는 리더로 선정되면 그가 제안한 블록을 인증하기 위해 상이하고 임시의 (ephemeral) 키 세트를 사용한다. 보다 정확하게, 그는 미리 1M 개의 공개 비밀 키 쌍 (pk_i^r, sk_i^r)의 세트를 선택하고, 그가 여전히 사용할 수 있는 각각의 비밀 키 sk_i^r를 안전하게 유지한다. 발명 시스템은 pk_i^r이 i가 제안된 r번째 블록에 서명할 수 있는 유일한 가능한 공개키인지 누구나 인지할 수 있고, 누구나 pk_i^r에 대한 i의 디지털 서명을 검증할 수 있지만 i만이 대응하는 비밀 키 sk_i^r를 알고 있는 사람이기 때문에 i만이 그러한 서명을 생성할 수 있다는 것을 보장한다. i가 정직하고 블록 B_i^r를 제안하도록 선택되면, i는 자신의 공개키 쌍

(pk_i^r, sk_i^r)에 대해 그것에 디지털 서명하고 그 후 즉시 sk_i^r를 파괴한다. 이 방법으로, 적들이 i를 손상시키는 데 성공하더라도, 그는 현재 손상된 i로 하여금 제2 및 상이한 블록을 인증하도록 강요하는데 필요한 비밀 키 sk_i^r를 찾지 못할 것이다. 위원회 멤버들에게도 동일한 이점이 있다. 이제 리더 ℓ^r에 의해 전파된 블록에 대해 위원회 C^r의 멤버들이 어떻게 합의에 도달했는지 살펴 보자.

[0029] C^r이 새로운 제안된 블록에 대한 합의에 이르기 위해 선호하는 방법은 비잔틴 합의(BA)를 통하는 것이다. BA 프로토콜은 각각 자신의 값으로 시작하면서 고정된 세트의 플레이어가 공통의 값에 대한 합의에 도달할 수 있게 하는 통신 프로토콜이다. 플레이어는 자신의 모든 지시 사항을 따르는 경우 정직하고 그렇지 않으면 악의적이 된다. 악의적인 플레이어는 단일 엔티티, 적에 의해 통제되고 완벽하게 조정될 수 있다. 소수(예를 들어, 1/3)의 플레이어만이 악의적인 한에는, BA 프로토콜은 2개의 기본 속성을 만족한다: (1) 모든 정직한 플레이어가 동일한 값을 출력하고, (2) 모든 플레이어가 동일한 입력 값으로 시작하면, 모든 정직한 플레이어가 그 값을 출력한다.

[0030] 비허가 분산 원장 시스템에서 BA를 사용하는 데는 몇 가지 문제가 있다. 주요한 하나는 BA 프로토콜이 매우 느리다는 것이다. (일반적으로 BA 프로토콜에는 최대 12명의 사용자가 참여한다.) Algorand는 최악의 경우 예측시 9단계만을 취하는 새로운 프로토콜을 통해 이러한 문제를 해결한다. 또한, 이들 각 단계에서, 참여하는 플레이어는 단일한 단문 메시지만 전달하면 된다!

[0031] 그러나 다른 도전 과제가 남아 있다. 새로운 블록에 대한 비잔틴 합의를 맺기 위해서는 새로운 블록은(그들은 비밀 암호 분류를 통해 자기 자신들을 선택하기 때문에) 그들이 자신들의 제1 메시지를 보내기 전에 적이 위원회 멤버들을 공격할 수는 없더라도 하나 이상의 단계들을 취하기 때문에, 그들이 BA 프로토콜로 자신들의 제1 메시지를 전파한 후에 그는 위원회 C^r의 모든 멤버를 손상시킬 수 있다. 사실, 자신들의 제1 메시지를 통해, 그들은 위원회에 소속된 자신들의 증거를 전파한다. 다행히도, 본 발명의 합의 프로토콜은 플레이어가 교체 가능하다는 추가적이고 새로운 이점을 누리고 있다. 즉, 프로토콜의 예상되는 몇 단계의 각각은 이전 단계를 수행하기 위해 선택된 플레이어와 임의의 비밀 정보 또는 임의의 숨겨진 내부 상태를 공유할 필요가 없는 다른 플레이어 세트에 의해 실행될 수 있다. 사실, Algorand의 합의 프로토콜은 무국적(state-less) 비잔틴 합의(BA) 프로토콜이다. 따라서 새로운 BA 프로토콜의 각 단계를 실행하기 위해 비밀 암호 분류를 통해 상이한 사용자 세트가 선택될 수 있다.

[0032] 마지막 도전 과제가 남아 있다. 우리가 알고 있는 바와 같이, 공개키는 이미 공개적으로 알려진 이전 블록 B^{r-1}의 일부인 양 Q^r에 디지털 서명하고, 자신의 서명이 특수 속성, 즉, 자신의 해시는 주어진 선택 임계 값보다 작아야 하는 것을 만족하는지를 검증함으로써 블록 B^r의 생성에 역할을 가지도록 자기 자신을 선택한다. 문제는 Algorand가 비허가형이기 때문에, 적은 자신의 Q^r의 서명이 원하는 속성을 가진 사람을 찾을 때까지 적은 많은 공개/비밀 키 쌍을 생성하는 것을 유지할 수 있다. 그가 그러한 키를 발견할 때, 그는 자신의 키 중 하나가 블록 B^r 선택을 담당하는 리더임을 보장하기 위해 그것을 시스템에 가져올 수 있고, 또는 위원회 C^r의 대부분은 자신이 통제하는 키를 가지고 있어서 그가 BA 속성을 파괴할 수 있다. 이러한 가능성을 방지하기 위해, Algorand는 두 가지 전략에 의존한다. 우선, 새로 도입된 키가 블록 리더 또는 위원회 멤버로 즉시 선정될 자격을 가지는 것을 허용하지 않는 것이다. 오히려, 블록 B^r의 생성에 중요한 역할을 하기 위해, 잠시 동안 키 pk가 주변에 있어야 한다. 좀 더 정확하게 말하자면, 블록 B^{r-k}에서 처음으로 나타나거나, 또는 이전 블록에서 나타나야 하며, 여기서 k는 충분히 큰 정수이다. (예를 들어, 돈의 적어도 80%가 정직한 손에 있다면, 우리의 상세한 실시 예 각각에서 k = 70으로 충분하다.) 둘째로, Q^r을 유도적으로, 즉 이전의 수량인 Q^{r-1}에서 정의한다. 보다 정확하게, 바람직한 실시 예에서, Q^r은 쌍(Q^{r-1}, r)의 블록 B^r의 리더의 해시 디지털 서명이다. 이 디지털 서명은 실제로 블록 B^{r-1}의 명시적인 부분으로 만들어진다. 그 이유는 다음과 같다. 일부 블록 B^{r-k}에서 적이 새로운 키 pk를 시스템으로 가져 오기로 결정할 때, 공개키로 결제하면 그는 pk에 대해 제어하고 pk가 역할을 할 수 있는 가장 빠른 블록이 B^r이 된다. 그러나 우리는 B^r 이전에 k 블록이 있음을 증명할 것이며, 아무도 Q^{r-1}이 랜덤 추측에 의한 것보다 현저하게 좋을지 예측할 수 없다. 따라서 적은 전략적으로 pk를 선택할 수 없다!

[0033] **블록 트리(및 상태 트리)**

[0034] 블록체인은 자신의 블록의 변조 방지 기능을 보장하지만 자신의 블록에 대해 작동하는 것이(예를 들어, 주어진 결제가 주어진 블록의 일부인지 여부 증명)이 매우 번거롭다. 공유 공개 원장은 블록체인의 동의어가 아니며, 사실 블록을 구성하는 더 나은 방법으로부터 이익을 얻는다. Algorand는 전통적인 블록체인과 함께 작동하지만 정보의 블록, 블록트리를 구축하는 새로운 방법으로도 작동한다. 이 발명 방법은 독립적인 관심사일 수 있다.

[0035] 블록트리의 주요 이점은 전체 블록체인을 전시할 필요없이 개별 과거 블록의 콘텐츠를 효율적으로 증명할 수 있게하는 것이다.

[0036] 머클트리의 새로운 응용 프로그램을 통해 블록트리를 구축한다.

[0037] 또한, 결제가 공개키에서 다른 공개키로 돈을 이전하고 결제가 블록의 시퀀스로 구성되는 분산 결제 시스템의 경우, 우리는 결제의 제1 r번째 블록 후에 주어진 키에 대해 가용한 잔고가 무엇인지(계좌의 돈) 증명할 수 있는 효율적인 방법을 제공한다. 우리는 이러한 방법을 상태트리(statustrees)라고 부른다.

도면의 간단한 설명

[0038] 여기에 설명된 시스템의 실시 예가 도면에 따라 더 상세히 설명되며, 이는 간단히 다음과 같이 설명된다.

도 1은 여기에 설명된 시스템의 일 실시 예에 따른 네트워크 및 컴퓨팅 스테이션의 개략도이다.

도 2는 새로운 거래 블록이 제안된 Algorand 시스템의 제1 단계의 개략적인 개념 요약이다.

도 3은 Algorand 시스템의 새로운 블록에 대한 동의 및 인증에 대한 개략적인 개념 요약이다.

도 4는 자신의 하나의 노드에 포함된 값에 대한 머클 트리 및 인증 경로를 도시한 개략도이다.

도 5는 블록 트리에 구축된 제1 블록에 해당하는 머클 트리를 도시한 개략도이다.

발명을 실시하기 위한 구체적인 내용

[0039] 본 명세서에 설명된 시스템은 거래 검증 및 전파를 분산하여 거래 정보를 검증 및/또는 전파하기 위한 계산을 수행하는 단독 엔티티가 존재하지 않도록 하기 위한 메커니즘을 제공한다. 대신, 참여하는 각 엔티티는 거래를 검증 가능하고 신뢰할 수 있는 방식으로 전파하기 위해 수행되는 계산을 공유한다.

[0040] 도 1을 참조하면, 다이어그램은 인터넷과 같은 데이터 네트워크(24)에 연결된 복수의 컴퓨팅 워크스테이션(22a-22c)을 도시한다. 워크스테이션(22a-22c)은 네트워크(24)를 통해 서로 통신하여 본 명세서의 다른 곳에서 더 상세히 기술된 바와 같이 분산 거래 전파 및 검증을 제공한다. 시스템은 워크스테이션(22a-22c)이 서로 통신할 수 있는 한, 본 명세서에서 설명된 기능을 제공할 수 있는 임의의 수의 워크스테이션을 수용할 수 있다. 각각의 워크스테이션(22a-22c)은 본 명세서의 다른 곳에서 더 상세히 기술된 바와 같이, 시스템 내의 모든 다른 워크스테이션에 거래를 전파하고 거래를 검증하기 위한 프로세싱을 독립적으로 수행할 수 있다.

[0041] 도 2는 Algorand 시스템에서 라운드 r의 제1 단계를 개략적으로 그리고 개념적으로 요약한 것으로, 여기서 소수의 선택된 사용자 각각은 r 번째 블록에 대해 자신의 후보를 제안한다. 특히, 이 단계는 시스템의 사용자 a, ..., z로 시작하고, 개별적으로 비밀 암호 분류 프로세스를 수행하고, 이는 어느 사용자가 블록을 제안하도록 선택되었는지, 그리고 각각의 선택된 사용자가 자신이 블록을 생성할 권한이 있음을 증명하는 보증서를 비밀리에 연산하는지를 결정한다. 도 2의 예에서, 사용자 b, d 및 h만이 블록을 제안하도록 선택되고, 그들의 각각 계산된 보증서는 $\sigma_b^{r,1}$, $\sigma_d^{r,a1}$, $\sigma_h^{r,1}$ 이다. 각각의 선택된 사용자 i는 자신이 제안한 블록 B_i^r을 어셈블리하고, 그것에 임시로(ephemerally) 서명하고(즉, 나중에 그것에 대해 설명되는 바와 같이, 그것을 임시 키로 디지털 서명), 자신의 보증서와 함께 네트워크에 전파한다. 라운드의 리더는 자신의 보증서가 가장 작은 해시를 가진 선택된 사용자이다. 이 도면은 리더가 사용자 d라는 것을 나타낸다. 따라서 그의 제안된 블록 B_d^r는 이진 합의(binary agreement) 프로토콜의 입력으로 주어지는 것이다.

[0042] 도 3은 제안된 블록을 공식 r번째 블록인 B^r로서 합의에 도달하고 인증하기 위한 Algorand의 프로세스를 도식적으로 개념적으로 요약한 것이다. Algorand의 제1 단계는 새 블록을 제안하는 것으로 구성되고, 이 프로세스는 제2 단계로 시작된다. 이 단계는 실제로 Algorand가 선호하는 비잔틴 합의 프로토콜 $BA^\star$의 제1 단계와

일치한다. 이 프로토콜의 각 단계는 비밀 암호 분류(이 도면에 표시되지 않음)에 의해 무작위로 선택된 플레이어의 다른 "위원회"에 의해 실행된다. 따라서 각 단계를 수행하도록 선택한 사용자는 완전히 다를 수 있다. $BA^{\star}$의 단계 수는 변할 수 있다. 도 3은 Algorand의 단계 2에서부터 Algorand의 단계 8까지의 7개의 단계를 수반하는 $BA^{\star}$의 실행을 나타낸다. 도 3의 예시에서, 단계 2를 수행하도록 선택된 사용자는 a, e 및 q이다. 각 사용자 $i \in \{a, e, q\}$는, i가 실제로 Algorand의 라운드 r의 2단계에서 실제로 메시지를 전송할 자격이 있고 임시 서명된 자신의 메시지 $m_i^{r,s}$가 이 단계에서 적절하다는 것을 증명하는 자신의 보증서 $\sigma_i^{r,2}$를 네트워크에 전파한다. 단계 3-7은 도시되지 않는다. 마지막 8단계에서, 도면은 라운드 r의 공식 블록으로서 B^r에 대해 합의에 도달한 대응하는 선택된 사용자 b, f, 및 x가 블록 B^r의 자신들의 임시 서명(함께, 이 서명은 B^r을 인증함), 및 그들이 8단계에서 행동할 자격이 있음을 증명하는 자신들의 보증서를 전파한다는 것을 도시한다.

[0043] 도 4는 머클 트리 및 그의 인증 경로 중 하나를 개략적으로 도시한다. 구체적으로, 도 4.A는 깊이 3의 풀 머클 트리를 도시한다. x가 길이 ≤3인 이진 스트링으로 표시되는 각 노드 x는 값 v_x를 저장한다. x의 길이가 ≤2이면 $v_x = H(v_{x0}, v_{x1})$이다. 도 4.A의 머클 트리에 대해, 도 4.B는 값 v_{010}의 인증 경로를 도시한다.

[0044] 도 5는 깊이 3의 풀 이진 트리(full binary tree) 내에 구축된, 블록 트리(blocktree)에서 구성된 제1 8개의 블록에 대응하는 머클 트리를 개략적으로 도시한다. 도 5.i에서, 정수로 표시된 노드는 머클 트리 T_i에 속한다. i로 표시된 노드(각각 i로)의 콘텐츠는 임시적이다(각각 영구적).

[0045] 단지 구체성을 위해, 우리는 우리의 설명에서 모든 거래가 결제이고 Algorand를 화폐 플랫폼으로서만 기술하는 데 초점을 둔다고 가정한다. 당업자는 Algorand가 모든 종류의 거래를 처리할 수 있음을 또한 알 것이다.

[0046] Algorand는 매우 유연한 설계를 가지고 있고 다양한 방식으로 그러나 그와 관련된 방식으로 구현될 수 있다. 우리는 전체적인 설계의 두 가지 가능한 실시 예를 자세히 설명함으로써 그의 유연성을 예시한다. 그것들로부터, 당업자는 모든 다른 종류의 구현을 도출하는 방법을 또한 이해할 수 있다.

[0047] 발명에 대한 이해를 돕고 그의 다양한 부분에 대한 내부 상호 참조를 허용하기 위해, 우리는 번호가 매겨진 제목 부분에 프리젠테이션을 구성한다. 제1 섹션은 상세한 실시 예 모두에 대해 공통이다.

[0048] **1. 소개**

[0049] 돈은 점점 가상화되고 있다. 오늘날 미국 달러의 약 80%가 원장 엔트리(ledger entry)로서만 존재하는 것으로 추정된다. 다른 금융 상품은 다음과 같다.

[0050] 가능한 모든 사이버 공격으로부터 안전하고 보편적으로 신뢰할 수 있는 중앙 엔티티를 기대할 수 있는 이상적인 세계에서, 돈 및 기타 금융 거래는 단지 전자적인 것일 수 있다. 불행히도, 우리는 그런 세상에 살지 않는다. 따라서 비트코인과 같은 분산된 암호화폐와 이더리움(Ethereum)과 같은 "스마트 계약" 시스템이 제안되었다. 이러한 시스템의 중심에는 결제 및 계약에 따라 변하는 거래의 시퀀스를 조작방지(tamperproof) 방식으로 안전하게 기록하는 공유 원장이 있다. 이러한 조작방지를 보장하는 기술이 블록체인(blockchain)이다. 블록체인은 암호화폐, 금융 응용 프로그램 및 사물 인터넷과 같은 응용 프로그램 뒤에 있다. 블록체인 기반 원장을 관리하는 몇 가지 기술이 제안되었다: 작업 증명(proof of work), 지분 증명(proof of stake), 플랙티컬 비잔틴 장애허용(PBFT: practical Byzantine fault-tolerance) 또는 일부 조합이다.

[0051] 그러나 현재는 원장 관리가 비효율적일 수 있다. 예를 들어, 비트코인의 작업 증명 접근법은 방대한 양의 계산을 필요로하며 낭비적이고 조악하게 스케일링한다. 또한, 사실상 매우 소수에 권력을 집중시킨다.

[0052] 따라서 우리는 현재의 분산된 구현의 비효율성과 취약점이 없는 신뢰할 수 있고 불가침적인 기관에 의해 운영되는 중앙 시스템의 편리함과 효율성을 제공하는 공개 원장을 구현하는 새로운 방법을 제시하고자 한다. 우리는 우리의 접근 방식을 Algorand라고 부르는데, 왜냐하면 지금까지 구축된 원장에 기초하여 유효한 거래의 다음 블록을 구축하는 것을 담당하는 검증자의 세트를 선택하기 위해 알고리즘적 무작위성(randomness)을 사용하기 때문이다. 당연히, 우리는 이러한 선택이 조작으로부터 면역가능하고 마지막 순간까지 예측할 수 없지만 그것들은 궁극적으로 보편적으로 명확하다는 것을 보장한다.

[0053] Algorand의 접근법은 이론적으로나 실제로나 상이한 클래스의 사용자(비트코인에서는 "채굴자" 및 "일반 사용자"와 같이)를 생성하지 않는다는 점에서 매우 민주적이다. Algorand에서 "모든 권한은 모든 사용자 세트에

있다".

[0054] Algorand의 주목할만한 속성 중 하나는 자신의 거래 이력이 매우 작은 확률(예를 들어, 1조분의 1, 즉 10^{-18})로 포크(fork)를 실행할 수 있다는 것이다. Algorand는 또한 일부 법적 및 정치적 문제를 해결할 수 있다.

[0055] Algorand 접근법은 블록체인에 적용되며 더 일반적으로 조작방지 블록 시퀀스를 생성하는 모든 방법에 적용된다. 우리는 실제로 독립적인 관심사가 될 수 있는 블록체인보다 더 효율적이고 그에 대한 대안인 새로운 방법을 제시했다.

[0056] **1.1 비트코인의 가정과 기술적 문제**

[0057] 비트코인은 매우 독창적인 시스템이며 많은 양의 후속 연구에 영감을 주었다. 그러나 문제도 있다. 비트코인처럼 실제로 작업 증명에 기초하는 모든 암호화폐에 의해 필수적으로 공유되는 그에 내재하는 가정과 기술적 문제를 요약해 보자.

[0058] 이 요약에서는, 비트코인에서 사용자가 디지털 서명 체계의 여러 공개키를 소유하고 있으며 화폐가 공개키와 연관되고, 결제가 하나의 공개키로부터 다른 공개키로 일부 금액을 이전하는 디지털 서명임을 상기해두면 충분하다. 본질적으로, 비트코인은 임의의 순서로 취해진 B_1, 임의의 순서로 B_2 등에 의해 후속되는 모든 결제가 유효한 결제의 시퀀스를 구성하도록 각각 다수의 결제를 구성하는 블록체인, B_1, B_2, ...으로 모든 처리되는 결제를 구성한다. 각 블록은 평균 10분마다 생성된다.

[0059] 단일 블록에서도 임의의 변경이 모든 후속 블록으로 영향을 미치도록 하여 그것이 결제 내역의 변경 사항을 쉽게 파악하는 것을 보장하도록 구조화되어 있기 때문에, 이 블록 시퀀스는 체인이다. (우리가 볼 수 있듯이, 이것은 각 블록에 이전 블록의 암호 해시를 포함시킴으로써 달성된다.) 이러한 블록 구조를 블록체인이라고 한다.

[0060] **가정: 정직한 대다수의 계산 능력**

[0061] 비트코인은 블록 생성에 쏟은 계산 능력의 대부분을 악의적인 엔티티(또는 조정된 악성 엔티티의 연합체)가 제어하지 않는다고 가정한다. 실제로 그러한 엔티티는 블록체인을 수정할 수 있으므로, 기쁘게도 결제 내역을 다시 작성할 수 있다. 특히, 결제 ℘를 할 수 있고 결제된 이익을 얻은 다음 ℘의 흔적을 "소거"할 수 있다.

[0062] **기술적 문제1 : 연산 낭비**

[0063] 블록 생성에 대한 비트코인의 작업 증명 접근 방식은 엄청난 양의 계산이 필요하다. 현재 시스템에 있는 수십만 개의 공개키로 최상위 500대의 가장 강력한 슈퍼 컴퓨터가 비트코인 플레이어가 요구하는 총 계산 능력의 12.8%만을 소집할 수 있다. 현저하게 더 많은 사용자가 시스템에 합류하면 이 계산량은 크게 늘어난다.

[0064] **기술 문제2 : 권력 집중**

[0065] 오늘날 과도한 계산이 필요하기 때문에 일반 데스크톱(휴대 전화는 물론)을 사용하여 새 블록을 생성하고자 하는 사용자는 돈을 잃을 것으로 예상된다. 실제로 일반 컴퓨터로 새 블록을 계산할 때, 계산에 파워를 공급하기 위해 필요한 전기의 예상 비용이 예상되는 보상을 초과한다. ("새로운 블록을 채굴하는 것" 이외에는 관계없는) 특별히 제작된 컴퓨터의 풀을 사용해서만이 새로운 블록을 생성하여 수익을 기대할 수 있다. 따라서 오늘날에는 실제로는 두 개의 따로 분리된 사용자 클래스가 있다. 즉, 결제만하는 일반 사용자와 새로운 블록을 검색만 하는 특수한 채굴 풀이 있다.

[0066] 따라서 최근에 블록 생성을 위한 총 컴퓨팅 파워가 단지 5개의 풀에 불과하다는 것은 놀라운 일이 아니다. 그러한 조건에서 계산 능력의 대부분이 정직하다는 가정은 신뢰성이 떨어진다.

[0067] **기술적인 문제3 : 모호성**

[0068] 비트코인에서, 블록체인은 필수적으로 고유한 것은 아니다. 실제로 최신 부분은 종종 포크(fork)하고: 블록체인은 하나의 사용자에 따라 B_1, ..., B_k, B'_{k+1}, B'_{k+2}이고, 또 다른 사용자에 따라 B_1, ..., B_k, B''_{k+1}, B''_{k+2}, B''_{k+3}일 수 있다. 체인에 몇 개의 블록이 추가된 후에만 제1 k+3 블록이 모든 사용자에게 동일할 것이라는 것을 합리적으로 확신할 수 있다. 따라서 체인의 마지막 블록에 포함된 결제에 즉시 의존할 수는 없다. 블록이 블록체인에서 충분히 깊어져서 충분히 안정적인지 여부를 기다리고 보는 것이 더 빈틈이 없다.

[0069] 이와 별도로 비트코인[1]에 대한 법 집행 및 통화 정책 문제도 제기되었다.

[0070] (1: 비트코인 결제에 의해 제공된(의사) 익명성은 돈세탁 및/또는 범죄자 또는 테러리스트 조직의 자금 조달에 오용될 수 있다. 이론적으로 완벽한 익명성을 제공하는 전통적인 지폐 또는 금괴는 동일한 도전 과제를 야기하지만, 이러한 통화의 특질은 법 집행 기관의 모니터링을 어느 정도 허용하기 위해 이체를 상당히 지연시킨다.

[0071] "돈을 찍어내는" 권한은 국민 국가의 아주 기본적인 권한 중 하나이다. 따라서 이론적으로 대규모로 독립적으로 유동하는 통화를 채택하면 이러한 권한이 축소될 수 있다. 그러나 현재 비트코인은 정부의 통화 정책에 대한 위협이 아니며 확장성 문제로 인해 결코 그렇게 될 수 없다.)

[0072] **1.2 너트셸에서의 Algorand**

[0073] **설정**

[0074] Algorand는 매우 어려운 설정에서 작동한다. 간단히,

[0075] (a) *비허가 및 권한이 부여된(permissioned) 환경*. Algorand는 임의의 다수의 사용자가 임의의 종류의 심사 및 허가 없이 언제든지 시스템에 합류할 수 있는 완전한 비허가형 환경에서 효율적이고 안전하게 작동한다. 물론 Algorand는 허가된 환경에서 더 잘 작동한다.

[0076] (b) *매우 적대적인 환경*. Algorand는 매우 강력한 적을 견디어 내며, 그 적은 다음을 할 수 있다:

[0077] (1) 비허가형 환경에서 시스템의 돈의 2/3가 정직한 사용자에게 속한다는 조건하에 원하는 임의의 사용자를, 그가 원하는 언제든지 즉시 손상시킬 수 있다. (권한이 부여된 환경에서 돈과 관계없이 2/3의 사용자가 정직한 것으로 충분하다.)

[0078] (2) 모든 손상된 사용자를 완전히 통제하고 완벽하게 조정한다.

[0079] (3) 정직한 사용자에 의해 전송된 각 메시지 m은 오직 m의 크기에 의존하는 시간 λ_m 내에 정직한 사용자 모두에게(또는 충분하게 많은) 도달하면, 모든 메시지의 전달을 스케줄링한다.

[0080] **주요 속성**

[0081] Algorand에서 우리의 강력한 적의 존재에도 불구하고

[0082] • *필요한 계산량이 최소화된다*. 본질적으로, 시스템에 아무리 많은 사용자가 있더라도, 1500명의 사용자 각각은 최대 몇 초의 계산을 수행해야 한다.

[0083] • 새로운 블록이 신속하게 생성되며 사실상 블록체인을 떠나지 않는다. 즉, Algorand의 블록체인은 무시할 수 있는 확률(즉, 1조분의 1 또는 10^{-18} 미만)로 포크할 수 있다. 따라서 사용자는 블록이 나타나자마자 새로운 블록에 포함된 결제를 중계할 수 있다.

[0084] • 모든 권한은 사용자 자체와 함께 있다. Algorand는 진정한 분산 시스템이다. 특히 비트코인의 "채굴자"와 같이 거래가 인식되는 것을 통제할 수 있는 외부 엔티티가 존재하지 않는다.

[0085] **Algorand의 기술.**

[0086] 1. 새롭고 빠른 비잔틴 합의 프로토콜.

[0087] Algorand는 독창적인 암호화, 메시지 전달, 바이너리 비잔틴 합의(BA) 프로토콜인 $BA^\star$를 통해 새로운 블록을 생성한다. 프로토콜 $BA^\star$는(우리가 곧 논의할) 몇 가지 추가 속성을 만족시킬 뿐만 아니라 매우 빠르다. 대충 말하자면, 바이너리 입력 버전은 3단계 루프로 구성되어 있고, 여기서 플레이어 i가 하나의 메시지 m_i를 다른 모든 플레이어들에게 전송한다. 완전하고 동기적인 네트워크에서 실행되며 프로토콜이 합의시 종료하는 각 루프 후에 플레이어의 2/3 이상이 확률> 1/3로 정직하다. (우리는 프로토콜 $BA^\star$가 약화없이 비잔틴 합의의 원래 정의를 만족시킨다는 것을 강조한다.)

[0088] Algorand는 바이너리 BA 프로토콜을 레버리징하여 서로 다른 통신 모델에서 각각의 새로운 블록에 대한 합의에 도달한다. 합의된 블록은 적절한 검증자의 규정된 수의 디지털 서명을 통해 인증되고 네트워크를 통해 전파된다.

[0089] 2. 비밀 암호 분류.

[0090] 매우 빠르지만 $BA^{\star}$ 프로토콜은 수백만 명의 사용자가 플레이할 때 더 빠른 속도로 이익을 얻는다. 따라서 Algorand는 $BA^{\star}$의 플레이어를 모든 사용자 세트 중 훨씬 작은 하위 세트가 되도록 선택한다. 상이한 종류의 권력의 집중 문제를 방지하기 위해, 새로운 블록 B^{r} 각각은 별도의 세트의 선택된 검증자 SV^{r}에 의해 $BA^{\star}$의 새로운 실행을 통해 구성되고 동의될 것이다. 원칙적으로 이러한 세트를 선택하는 것은 B^{r}을 직접 선택하는 것만큼 어려울 수 있다. 우리는 이 잠재적인 문제를 우리가 비밀 암호 분류라고 부르는 새로운 접근 방식으로 주의 깊게 고찰한다. 분류란 자격을 갖춘(eligible) 다수의 세트의 개인들로부터 무작위로 담당자를 선발하는 관행이다. (예를 들어, 아테네, 플로렌스, 베니스의 공화국에 의해 수 세기 동안 분류가 실시되었다. 현대 사법 제도에서는 배심원을 선발하기 위해 무작위 선택이 종종 사용된다. 무작위 표본 추출도 선거를 위해 옹호되었다.) 분산된 시스템에서, 물론, 각 검증자 세트 SV^{r}의 멤버를 무작위로 선택하는데 필요한 랜덤 코인을 선택하는 것은 문제가 된다. 따라서, 우리는 모든 사용자의 모집단으로부터 자동적(즉, 메시지 교환을 필요로하지 않음) 및 무작위적으로 보장되는 방식으로 각 검증자 세트를 선택하기 위해 암호화에 의존한다. 유사한 방식으로 우리는 사용자, 새로운 블록 B^{r}을 제안하는 책임자인 리더, 및 리더가 제안한 블록에 대해 합의에 도달하는 것을 담당하는 검증자 세트 SV^{r}을 선택한다. 본 발명의 시스템은 이전 블록의 콘텐츠에서 추론할 수 있고 매우 강한 적의 존재 하에서도 조작할 수 없는 일부 정보, Q^{r-1}을 레버리징한다.

[0091] 3. 양(씨드) Q^{r}.

[0092] 우리는 새로운 블록 B^{r}을 구성할 다음 검증자 세트와 리더를 자동으로 결정하기 위해 블록체인에서 마지막 블록 B^{r-1}을 사용한다. 이 접근법의 문제점은 이전 라운드에서 약간 다른 결제를 선택하는 것만으로도 강력한 적이 다음 리더를 엄청나게 통제할 수 있다는 것이다. 시스템에서 플레이어/돈 중 1/1000만 통제했더라도, 그는 모든 리더가 악의적이 되는 것을 보장할 수 있다.(직관 4.1 섹션 참고.) 이 도전은 모든 지분 증명 접근법의 핵심이며, 우리가 아는 한 지금까지는 만족스럽게 해결되지 않았다.

[0093] 이 과제를 해결하기 위해 우리는 예측할 수 없을 뿐만 아니라 우리의 강력한 적에 의해 영향을 받지 않는 별도의 주의 깊게 정의된 수량인 Q^{r}을 의도적으로 구축하고 지속적으로 업데이트한다. Algorand가 비밀 암호 분류를 통해 r번째 블록 생성에 특별한 역할을 하는 모든 사용자를 선택하는 것이 Q^{r}로부터이므로 Q^{r}을 r번째 시드라고 할 수 있다. 시드 Q^{r}은 블록 B^{r-1}에서 추론될 것이다.

[0094] 4. 보안 보증서.

[0095] 검증자 세트와 새로운 블록 B^{r}을 구축하는 것을 담당하는 리더를 선택하기 위해 현재 마지막 블록인 B^{r-1}을 무작위로 그리고 모호하지 않게 사용하는 것은 충분하지 않다. B^{r}을 생성하기 전에 B^{r-1}이 알려져야 하기 때문에, 마지막으로 영향을 받지않는 양 Q^{r-1}(B^{r-1}에서 유추할 수 있는)도 공지되어야 한다. 따라서 검증자와 책임있는 리더가 블록 B^{r}을 계산할 수 있다. 따라서 우리의 강력한 적은 그들이 B^{r}에 관한 토론에 참여하기 전에 즉시 그들 모두를 손상시켜서 그들이 인증한 블록을 완전히 제어할 수 있다.

[0096] 이 문제를 방지하기 위해 리더(실제로 검증자도 역시)는 자신의 역할을 은밀하게 배우지만 실제로 그 역할을 가진다는 것을 모든 사람에게 증명할 수 있는 적절한 보증서를 계산할 수 있다. 사용자가 개인적으로 자신이 다음 블록의 리더임을 알게되면, 먼저 자신이 제안한 새 블록을 보안하에 어셈블리한 다음 자신의 보증서와 함께 그것을 전파한다(인증될 수 있도록). 이 방법으로, 적은 즉시 다음 블록의 리더가 누구인지 알게 되지만, 그가 즉시 그를 손상시킬 수 있더라도, 적이 새로운 블록의 선택에 영향을 미치기에는 너무 늦게 될 것이다. 사실, 그는 강력한 정부가 위키리크스에 의해 바이러스처럼 확산된 메시지를 다시 주워담지못하는 것처럼 리더의 메시지를 "콜백" 할 수 없다.

[0097] 우리가 볼 수 있듯이, 우리는 리더의 유일성을 보장할 수 없으며, 리더 자신을 포함하여 누가 리더인지 보장하지 못한다! 그러나 Algorand에서는, 모호하지 않은 진보가 보장된다.

[0098] 5. 플레이어 교체.

[0099] 그가 새로운 블록을 제안한 후, 리더도 마찬가지로 그의 일이 끝났기 때문에 "죽는다(die)"(또는 적에게 손상된다). 그러나 SV^r에서의 검증자에게는 상황이 덜 단순하다. 사실, 충분히 많은 서명으로 새로운 블록 B^r을 인증하는 책임을 맡기 위해서는, 먼저 리더가 제안한 블록에 대한 비잔틴 합의를 실행해야 한다. 문제는 얼마나 효율적이든 간에 $BA^\star$는 여러 단계와 플레이어의 2/3 이상의 정직함을 요구한다는 것이다. 효율성 면에서 $BA^\star$의 플레이어 세트가 모든 사용자 세트 중에서 무작위로 선택된 작은 세트 SV^r로 구성되기 때문에 이것은 문제이다. 따라서, 우리의 강력한 적들은 모든 사용자의 1/3을 손상시킬 수는 없지만 SV^r의 모든 멤버들을 확실하게 손상시킬 수 있다.

[0100] 다행히도 우리는 피어-투-피어 방식으로 메시지를 전달함으로써 실행되는 프로토콜 $BA^\star$가 플레이어 교체 가능하다는 것을 증명할 것이다. 이 새로운 요구 사항은 각 단계가 완전히 새로운 무작위로 독립적으로 선택된 플레이어의 세트에 의해 실행되더라도 프로토콜이 정확하고 효율적으로 합의에 도달한다는 것을 의미한다. 따라서 수백만 명의 사용자를 가진, $BA^\star$의 단계와 관련된 각각의 작은 세트의 플레이어는 다음 세트와 빈 교차점을 가질 가능성이 크다.

[0101] 또한, $BA^\star$의 다른 단계의 플레이어 세트는 완전히 다른 카디널리티(cardinality)를 가질 것이다. 또한 각 세트의 멤버는 다음 플레이어 세트가 누가 될지를 알지 못하며 비밀리에 내부 상태를 전달하지 않는다.

[0102] 교체 가능한 플레이어 속성은 실제로 우리가 생각하는 역동적이고 강력한 적을 물리치기 위해 중요하다. 우리는 교체 가능한 플레이어 프로토콜이 많은 문맥과 애플리케이션에서 중요하다고 믿는다. 특히, 그것들은 더 큰 규모의 플레이어에 내장된 작은 하위 프로토콜을 안전하게 실행하는 데 중요할 것이며, 전체 플레이어 중 매우 일부만을 손상시킬 수 있는 동적인 적은 더 작은 하위 프로토콜에서 모든 플레이어들을 손상시키는 데에 어려움이 없다.

[0103] **추가 속성/기법: 게으른 정직**

[0104] 정직한 사용자는 온라인 상태를 포함하고 프로토콜을 실행하는 자신의 규정된 명령을 따른다. Algorand는 작은 계산 및 통신 요구 사항을 갖기 때문에 온라인 상태이고 "백그라운드에서" 프로토콜을 실행하는 것은 큰 희생이 아니다. 물론, 정직한 플레이어 중 갑자기 연결이 끊어지거나 재부팅이 필요한 사람들과 같은 소수의 "불참"은 자동으로 용인된다(우리는 항상 이러한 소수의 플레이어를 일시적으로 악의적인 것으로 간주할 수 있기 때문에). 그러나 Algorand는 정직한 사용자가 대부분의 시간에 오프라인일 수 있는 새로운 모델에서 작업할 수 있도록 간단하게 조정할 수 있다. 우리의 새로운 모델은 다음과 같이 비공식적으로 소개될 수 있다.

[0105] 게으른 정직(Lazy Honesty).

[0106] 대략적으로 말하자면, 사용자 i는 (1) 자신이 프로토콜에 참여하도록 요청받을 때, 자신의 모든 규정된 명령을 따르고, (2) 프로토콜에 거의 참여하지 말도록 요청받고, 적절한 사전 통지를 받는 경우 게으르지만 정직하다.

[0107] 그러한 정직의 편안한 개념으로, 우리는 정직한 사람들이 필요할 때 가까이에 있을 것이라고 더 확신할 수 있고, Algorand는 이것이 사실일 때, 다음을 보장한다.

[0108] *주어진 시점에 참여하는 대다수의 플레이어가 악의적일지라도, 시스템은 안전하게 작동한다.*

[0109] **2 예비 단계**

[0110] **2.1 암호화 프리미티브(cryptographic primitive)**

[0111] **이상적인 해싱.**

[0112] 우리는 임의로 긴 스트링을 고정 길이의 이진 스트링에 매핑하는 효율적으로 계산 가능한 암호화 해시 함수 H에 의존한다. 오랜 전통에 따라, 우리는 본질적으로 각 가능한 스트링 s를 선택한 길이의 무작위로 독립적으로 선택한(그런다음 고정된) 이진 스트링 H(s)에 매핑하는 함수인 H를 랜덤 오라클로서 모델링한다.

[0113] 설명된 실시 예에서, H는 256비트의 긴 출력을 갖는다. 사실, 이러한 길이는 시스템을 효율적으로 만들기에 충분히 짧고 시스템을 안전하게 만들만큼 충분히 길다. 예를 들어, 우리는 H가 충돌-탄성이 되기를 원한다. 즉, H(x)=H(y)가 되도록 두 개의 다른 스트링 x와 y를 찾는 것이 어려워야 한다. H가 256비트 길이의 출력을 가진 랜덤 오라클일 때, 그러한 스트링 쌍을 찾는 것은 정말로 어렵다. (무작위로 시도하고 생일 패러독스에 의존한

다면 $2^{256/2} = 2^{128}$ 번의 시도가 필요하다.)

[0114] **디지털 서명.**

[0115] 디지털 서명을 통해 사용자는 비밀 키를 공유하지 않고도 정보를 서로에 대해 인증할 수 있다. 디지털 서명 기법은 확률적 키 생성기(a probabilistic key generator) G, 서명 알고리즘 S 및 검증 알고리즘 V의 세 가지 빠른 알고리즘으로 구성된다.

[0116] 충분히 높은 정수인 보안 파라미터 k가 주어지면, 사용자 i는 G를 사용하여 "공개" 키 pk_i 및 매칭하는 "비밀" 서명키 sk_i의 k 비트 키의 쌍(즉, 스트링)을 산출한다. 결정적으로, 공개키는 자신의 대응하는 비밀 키를 "배신"하지 않는다. 즉, pk_i에 대한 지식이 주어지더라도, i 이외의 다른 누구도 천문학적인 시간보다 적은 시간에 sk_i를 계산할 수 없다.

[0117] 사용자 i는 sk_i를 사용하여 메시지에 디지털 서명한다. 가능한 각 메시지(이진 스트링) m에 대해, i는 먼저 m을 해싱한 다음 k 비트 스트링을 산출하기 위해 입력 H(m) 및 sk_i에 대해 알고리즘 S를 실행한다

[0118] $$sig_{pk_i}(m) \triangleq S(H(m), sk_i)$$ [2].

[0120] *(2: H는 충돌-탄성을 가지기 때문에, m을 서명함으로써 상이한 메시지 m'에 "우연히 서명"하는 것은 실질적으로 불가능하다.)

[0121] 이진 스트링 $sig_{pk_i}(m)$은 m의 i의 디지털 서명(pk_i에 대해)이라고 하며, 공개키 pk_i가 문맥으로부터 명확할 때 $sig_i(m)$로 더 간단하게 표시될 수 있다.

[0122] pk_i를 알고 있는 모든 사람은 i가 생성한 디지털 서명을 확인하는 데 이를 사용할 수 있다. 구체적으로는, 입력 (a) 플레이어 i의 공개키 pk_i, (b) 메시지 m, 및 (c) 스트링 s, 즉, 메시지 m의 i의 추정된 디지털 서명에 대해, 검증 알고리즘 V는 YES 또는 NO를 출력한다.

[0123] 디지털 서명 체계에서 우리가 요구하는 속성은 다음과 같다:

[0124] 1. 합법적인 서명은 항상 검증된다: $s = sig_i(m)$이면, $V(pk_i, m, s) = YES$; 및

[0125] 2. 디지털 서명은 위조하기 어렵다: sk_i를 모른다면, $V(pk_i, m, s) = YES$가 되도록 스트링 s를 발견하는 시간은 i에 의해 절대로 서명되지 않은 메시지 m에 대해 천문학적으로 길게 된다.

[0126] (강력한 보안 요구 사항에 따라 다른 메시지의 서명을 얻을 수 있는 경우에도 마찬가지이다.)

[0127] 따라서 누군가 자신을 대신하여 메시지에 서명하는 것을 방지하기 위해, 플레이어 i는 자신의 서명키 sk_i를 비밀로 유지해야 하며(따라서 "비밀 키"라는 용어), 누구나 내가 서명한 메시지를 검증할 수 있게 하기 위해, i는 자신의 키 pk_i를 공개하는 것(따라서 "공개키"라는 용어)에 관심을 둔다.

[0128] **메시지 검색 가능성(Retrievability)이 있는 서명**

[0129] 일반적으로 메시지 m은 자신의 서명 $sig_i(m)$에서 검색할 수 없다. 개념적으로 편리한 "메시지 검색 가능성" 속성(즉, 서명자 및 메시지가 서명으로부터 쉽게 계산될 수 있음을 보장하기 위해)을 만족시키는 디지털 서명을 가상으로 처리하기 위해, 우리는 다음과 같이 정의한다:

[0130] $SIG_{pk_i}(m) = (i, m, sig_{pk_i}(m))$ 및 $SIG_i(m) = (i, m, sig_i(m))$이다 ($pk_i$가 명확한 경우).

[0131] **고유한 디지털 서명.**

[0132] 우리는 또한 다음의 추가 속성을 만족하는 디지털 서명 체계(G, S, V)를 고려한다.

[0133] 3. 고유성.

[0134] 스트링 pk', m, s, s'를 찾기는 어려워 s≠s'이고 $V(pk', m, s) = V(pk', m, s') = 1$가 되도록 한다.

[0135] (고유성 속성은 합법적으로 생성된 공개키가 아닌 스트링 pk'에 대해서도 적절하게 유지함을 유의하라. 그러나 특히, 고유성 속성은 지정된 키 생성기 G를 사용하여 공개키 pk와 매칭하는 보안 키 sk를 함께 계산하는 경우, 그리고 따라서 키 sk를 아는 경우, pk에 대해 동일한 메시지의 2개의 상이한 디지털 서명을 찾는 것이 그에게는 필수적으로 불가능할 것이라는 것을 의미한다.)

[0136] **비고(remark)**

[0137] · 고유한 서명으로부터 검증 가능한 랜덤 함수로. 고유성 속성을 갖는 디지털 서명 체계에 대해, m -> $H(sig_i(m))$ 매핑은 고유한 무작위로 선택된 256비트 스트링을 가능한 각 스트링 m에 연관시키고, 매핑의 정확성은 주어진 서명 $sig_i(m)$에 의해 증명될 수 있다.

[0138] 즉, 고유성 속성을 만족하는 이상적인 해싱 및 디지털 서명 체계는 본질적으로 검증 가능한 랜덤 함수(VRF)의 기본 구현을 제공한다.

[0139] VRF는 특별한 종류의 디지털 서명이다. 우리는 메시지 m의 i의 특별한 서명을 나타내기 위해 $VRF_i(m)$를 기록할 수 있다. 고유성 속성을 만족시키는 것 외에도, 검증 가능한 랜덤 함수는 충분히 랜덤하도록 보장되는 출력을 생성한다. 즉, $VRF_i(m)$는 본질적으로 랜덤하며 그것이 생성될 때까지 예측할 수 없다. 대조적으로, $SIG_i(m)$은 충분히 랜덤일 필요는 없다. 예를 들어, 사용자 i는 자신의 공개키를 선택하여 $SIG_i(m)$이 항상 (사전학적으로) 작은(즉, 처음 몇 비트가 항상 0일 수 있는) k비트 스트링이 되도록 할 수 있다. 그러나 H는 이상적인 해시 함수이므로 $H(SIG_i(m))$는 항상 랜덤 256비트 스트링이 된다는 것에 유의하라. 본 발명의 바람직한 실시 예에서, 각 메시지 m 및 각 사용자 i에 고유한 난수를 연관시킬 수 있도록 고유성 속성을 정확하게 만족시키는 해싱 디지털 서명을 광범위하게 사용한다. Algorand를 VRF로 구현해야 한다면, $H(SIG_i(m))$를 $VRF_i(m)$로 대체할 수 있다. 특히, 사용자 i는 먼저 $SIG_i(m)$을 계산하고, 그런 다음 $H(SIG_i(m))$를(즉, H(SIGi(m))를 수 p와 비교하기 위해) 계산할 필요가 없다. 그는 $VRF_i(m)$를 직접 계산할 수도 있다. 요약하자면, $H(SIG_i(m))$는 $VRF_i(m)$로 해석되거나 플레이어 i에 의해 쉽게 계산되지만 다른 사람에게는 예측할 수 없는, 그러나 i와 m에 모호하게 연관되는 충분히 랜덤한 수로서 해석될 수 있음을 이해해야 한다.

[0140] · 디지털 서명을 위한 3가지 다른 필요 사항. Algorand에서 사용자 i는 하기에 대해 디지털 서명을 사용한다.

[0141] (1) i 자신의 결제를 인증. 이 애플리케이션에서, 키는 "장기간"(즉, 장기간에 걸쳐 많은 메시지에 서명하는 데 사용됨)일 수 있으며 일반적인 서명 체계에서 비롯된다.

[0142] (2) i가 라운드 r의 일부 단계 s에서 행동할 수 있는 권한 있음을 증명하는 보증서 생성. 여기서 키는 장기간일 수 있지만 고유성 속성을 만족시키는 체계에서 가져와야한다.

[0143] (3) i가 그가 행동하는 각 단계에서 보내는 메시지를 인증. 여기에서, 키는 임시적(즉, 처음 사용된 후에 파손)이어야 하지만, 일반적인 서명 체계로부터 비롯된다.

[0144] · 작은 비용의 단순화. 간단히 하기 위해 우리는 각 사용자 i에게 하나의 장기간 키가 있다고 가정한다. 따라서 이러한 키는 고유성 속성을 가진 서명 체계에서 가져와야 한다. 이러한 단순성은 계산 비용이 적다. 실제로 일반적으로 고유한 디지털 서명은 일반 서명보다 생성하고 검증하는 데 약간 더 비싸다.

[0145] **2.2 이상화된 공개 원장**

[0146] Algorand는 이상화된 공개 원장을 기반으로 다음 결제 시스템을 모방하려 한다.

[0147] 1. 초기 상태. 돈은 개인 공개키(개인적으로 생성되고 사용자가 소유)와 연관된다. $pk_1, \ldots, pk_j$를 최초 공개키로 놓고, $\alpha_1, \ldots, \alpha_j$를 자신들의 각각의 최초 금액 단위으로 놓으면, 초기 상태는 다음과 같다:

[0148] $$S_0 = (pk_1, a_1), \ldots, (pk_j, a_j)$$

[0149] 이는 시스템에서 공통 지식으로 가정된다.

[0150] 2. 결제. pk를 현재 $\alpha \geq 0$인 금액 단위를 가지는 공개키로, pk'를 다른 공개키로, 그리고 α'를 α보다 크지 않

은 음이 아닌 수라고 한다. 그런 다음, (유효한) 결제 $\wp$는 pk에 대해 디지털 서명이며, 일부 추가 정보와 함께 pk로부터 pk'까지 α'의 화폐 단위의 이전을 지정한다. 기호에서,

[0151] $$\wp = SIG_{pk}(pk, pk', a', I, H(\mathcal{I}))$$

[0152] 여기서, I는 유용하지만 민감하지 않은 임의의 추가 정보(예를 들면, 시간 정보 및 결제 신원)를 나타내고, $\mathcal{I}$는 민감한 것으로 여겨지는 임의의 추가 정보(예를 들면, 결제 이유, 아마도 pk와 pk' 등의 소유자의 신분 등)를 나타낸다.

[0153] 우리는 pk(또는 그 소유자)를 지불자로, 각 pk'(또는 그 소유자)를 수취자로, α'를 결제 $\wp$의 크기로 지칭한다.

[0154] **결제를 통한 무료 가입.** 사용자는 자신 소유의 공개/비밀 키 쌍을 생성하여 자신들이 원하는 언제든지 시스템에 가입할 수 있다는 것에 유의하라. 따라서, 상기 결제 $\wp$에 나타나는 공개키 pk'는 이전에 돈을 결코 "소유하지" 않은 새롭게 생성된 공개키일 수 있다.

[0155] 3. 매직 원장(Magic Ledger). 이상화된 시스템에서는 모든 결제가 유효하며 모든 사람이 볼 수 있도록 "하늘에 포스팅된" 결제 세트의 조작 방지 목록 L에 나타난다:

[0156] $$L = PAY^1, PAY^2, \ldots$$

[0157] 각 블록 PAY^{r+1}은 블록 PAY^r의 출현 이후 작성된 모든 결제 세트로 구성된다. 이상적인 시스템에서는 고정된(또는 유한한) 시간 후에 새로운 블록이 나타난다.

[0158] **토론.**

[0159] · 보다 일반적인 결제 및 미사용 거래 출력. 더 일반적으로, 공개키 pk가 금액 α를 소유하면, pk의 유효한 결제 $\wp$는 $\sum_j a'_j \leq a$인 한은 각각 키 $pk'_1, pk'_2, \ldots$로 금액 $a'_1, a'_2, \ldots$를 이전할 수 있다.

[0160] 비트코인 및 유사한 시스템에서, 공개키 pk가 소유한 돈은 별도의 금액으로 분리되며, pk가 만든 결제 $\wp$는 그러한 분리된 금액 α를 전액 이전해야 한다. pk가 또 다른 키로 α분의 $a' < a$의 일부만을 이전하고자 하는 경우에는, 잔고, 미사용 거래 출력을 다른 키(아마도 pk 자체)로 이전해야 한다.

[0161] Algorand는 또한 분리된 금액을 갖는 키로 작업한다. 그러나 Algorand의 새로운 면에 초점을 맞추기 위해, 우리의 더 단순한 결제 방식과 단일 금액을 그것들에 연관시키는 키에 고정시키는 것이 개념적으로 더 간단하다.

[0162] · 현재 상태. 이상화된 체계는 시스템의 현재 상태(즉, 각 공개키가 얼마나 많은 돈의 단위를 갖는지에 관한)에 대한 정보를 직접 제공하지 않는다. 이 정보는 매직 원장에서 추론할 수 있다.

[0163] 이상적인 시스템에서, 활성 사용자는 최신 상태 정보를 지속적으로 저장 및 업데이트하거나 그것을 스크래치부, 또는 그가 그것을 연산한 최근의 시간부터 다시 구축해야한다. (그러나 나중에 Algorand를 증강시키는 방법을 보여줌으로써 그의 사용자가 효율적인 방식으로 현재 상태를 재구축할 수 있도록한다.)

[0164] · 보안 및 "프라이버시". 디지털 서명은 아무도 다른 사용자의 결제를 위조할 수 없다는 것을 보장한다. 결제 $\wp$에서는, 공개키와 금액이 숨겨지지 않지만, 민감한 정보 $\mathcal{I}$는 숨겨진다. 실제로 H($\mathcal{I}$)만 $\wp$으로 나타나고, H가 이상적인 해시 함수이기 때문에 H($\mathcal{I}$)는 랜덤 256비트 값이 되어 단순하게 추측하는 것보다 $\mathcal{I}$가 더 낫다는 것을 알아낼 방법이 없다. 그러나, $\mathcal{I}$가 무엇이었는지 증명하기 위해(예를 들어, 결제 이유를 증명하기 위해), 지불자는 $\mathcal{I}$를 드러낼 수도 있다. 공개된 $\mathcal{I}$의 정확성은 H($\mathcal{I}$)를 계산하고 결과 값을 $\wp$의 마지막 항목과 비교하여 검증될 수 있다. 실제로, H는 충돌 탄성이므로, $H(\mathcal{I}) = H(\mathcal{I}')$가 되도록 제2 값 $\mathcal{I}'$을 찾는 것은 어렵다.

[0165] **2.3 기본 개념 및 표기법**

[0166] **키, 사용자 및 소유자**

[0167] 달리 지정하지 않는 한, 각 공개키(간단히 "키")는 장기간이며 고유성 속성이 있는 디지털 서명 체계에 대한 것이다. 공개키 i는 이미 시스템에 있는 다른 공개키 j가 i에 대한 결제를 할 때 시스템에 합류한다.

[0168] 색상에 대해, 키를 개인화한다. 우리는 키 i를 "그"라고 말하고, i는 정직하며, 메시지를 송수신하는 등을 한다고 한다. 사용자는 키에 대해 동의어이다. 키를 그것이 속한 사람으로부터 구별하기를 원할 때, 우리는 "디지털 키"와 "소유자"라는 용어를 각각 사용한다.

[0169] **비허가 및 허가형 시스템.**

[0170] 디지털 키가 언제든지 가입할 수 있고 소유자가 여러 개의 디지털 키를 소유할 수 있는 경우, 시스템은 비허가형이고; 그렇지않으면 허가형이다.

[0171] **고유 표현**

[0172] Algorand의 각 객체에는 고유한 표현이 있다. 특히, 각 세트 $\{(x,y,z,\ldots) : x \in X, y \in Y, z \in Z, \ldots\}$ 는 사전지정된 방식으로 정렬된다(예를 들어, 사전학적으로 먼저 x에서, 그런 다음 y에서 등으로).

[0173] **동일 속도 시계**

[0174] 세계 시계는 없다. 오히려 각 사용자마다 자신의 시계가 있다. 사용자 시계는 어떤 방식으로도 동기화될 필요가 없다. 그러나 우리는 그것들이 모두 같은 속도를 가지고 있다고 가정한다.

[0175] 예를 들어 사용자 i의 시계에 따라 오후 12시가 되면, 다른 사용자 j의 시계에 따라 오후 2시 30분이 될 수 있지만, i의 시계에 따라 12시 01분이 되면 j의 시계에 따라 2시31분이 될 것이다. 즉, "1분은 모든 사용자에게 동일하다(충분히, 본질적으로 동일하다)".

[0176] **라운드**

[0177] Algorand는 논리 단위로 구성되며 r = 0, 1, ..., 이고, 라운드라고 한다.

[0178] 우리는 지속적으로 표제를 사용하여 라운드를 나타낸다. 숫자가 아닌 양의 Q(예를 들어, 스트링, 공개키, 세트, 디지털 서명 등)가 라운드 r을 참조한다는 것을 나타내기 위해, 단순히 Q^r을 기록한다. Q가 진정한 숫자(genuine number)인 경우에만(숫자로 해석할 수 있는 이진 스트링과 대조적으로), 기호 r이 Q의 지수로 해석될 수 없도록 $Q^{(r)}$을 작성한다.

[0179] 라운드 r> 0(의 시작)에서 모든 공개키의 세트는 PK^r이고 시스템 상태는 다음과 같다.

[0180] $$S^r = \left\{ \left(i, a_i^{(r)}, \ldots \right) : i \in PK^r \right\}$$

[0181] 여기서 $a_i^{(r)}$ 는 공개키 i에 사용 가능한 금액이다. PK^r은 S^r로부터 추론될 수 있고, S^r은 각 공개키 i에 대한 다른 구성 요소를 지정할 수도 있음에 유의해야 한다.

[0182] 라운드 0에서, PK^0은 최초 공개키의 세트이고 S^0은 최초 상태이다. PK^0와 S^0은 모두 시스템에서 일반적인 지식으로 가정한다. 단순화를 위해 라운드 r의 시작 부분에서도, PK^1, ..., PK^r 및 S^1, ..., S^r이 된다.

[0183] 라운드 r에서, 시스템 상태는 S^r에서 S^{r+1}로 전환된다 : 상징적으로,

[0184] 라운드 r: $S^r \rightarrow S^{r+1}$.

[0185] **결제**

[0186] Algorand에서, 사용자는 지속적으로 결제한다(그리고 2.7 절에서 설명한 방법으로 이를 전파한다). 사용자 $i \in PK^r$의 결제 $\wp$은 이상적인 시스템에서와 동일한 형식 및 의미를 갖는다. 즉,

[0187] $\wp = SIG_i(i, i', a, I, H(\mathcal{I}))$

[0188] 결제 $\wp$는 (1) 금액 a가 $a_i^{(r)}$ 이하이고, (2) r'<r에 대해 공식적인 페이셋(payset) $PAY^{r'}$에 나타나지 않는 경우, 라운드 r(간단히 라운드 r 결제)에서 개별적으로 유효하다. (아래에 설명된 것처럼, 제2 조건은 $\wp$가 이미 유효하지 않다는 것을 의미한다.

[0189] 금액의 합계가 최대 $a_i^{(r)}$ 인 경우, i의 라운드 r 페이셋의 세트가 집합적으로 유효하다.

[0190] **페이셋**

[0191] 라운드 r 페이셋 $\mathcal{P}$는 각 사용자 i에 대해 $\mathcal{P}$에서의 i의 결제(아마도 없음)가 집합적으로 유효하도록 라운드 r 결제의 세트가 된다. 모든 라운드 r 페이셋의 세트는 $\mathbb{PAY}(r)$이다. 라운드 r 페이셋 $\mathcal{P}$는 라운드 r 페이셋인 $\mathcal{P}$의 수퍼셋이 없는 경우 최대이다.

[0192] 우리는 결제 $\wp$가 라운드 ρ를 $\wp = SIG_i(\rho, i, i', a, I, H(\mathcal{I}))$로 규정하고, 일부 고정된 음이 아닌 정수 k에 대해 $[\rho, \rho + k]$ 외부의 임의의 라운드에서 유효할 수 없도록 실제로 제안한다.[3]

[0193] (3: 이것은 $\wp$가 "효과적"이 되었는지 확인을 간단하게 한다(즉, 어떤 페이셋, PAY^r가 $\wp$를 포함하는지를 판정하는 것을 간단하게 한다), k=0일 때, $\wp = SIG_i(r, i, i', a, I, H(\mathcal{I}))$이고, $\wp \notin PAY^r$이면, i는 $\wp$를 다시 제출해야 한다.)

[0194] **공식 페이셋**

[0195] 모든 라운드 r에 대해, Algorand는 라운드의 공식 페이셋인, 단일한(아마 빈) 페이셋 PAY^r을 공개적으로(나중에 설명하는 방식으로) 선택한다(PAY^r은 "실제로" 발생한 라운드 r 결제를 나타낸다.)

[0196] 이상적인 시스템(및 비트코인)에서와 같이, (1) 신규 사용자 j가 시스템에 진입하는 유일한 방법은 주어진 라운드 r의 공식 페이셋 PAY^r에 속한 결제의 수취인이 되는 것이고; (2) PAY^r은 현재 라운드의 상태 S^r로부터 다음 라운드의 상태 S^{r+1}를 판정한다. 상징적으로,

[0197] $PAY^r : S^r \longrightarrow S^{r+1}$이다.

[0198] 구체적으로는,

[0199] 1. 라운드 r+1의 공개키 세트, PK^{r+1}은 PK^r과 PAY^r의 결제에 처음 나타나는 모든 수취인 키의 세트의 조합으로 구성되고;

[0200] 2. 라운드 r+1에서 사용자 i가 소유한 금액 $a_i^{(r+1)}$은 $a_i(r)$(즉, 이전 라운드에서 i가 소유한 금액($i \notin PK^r$ 인 경우 0))의 합계 및 PAY^r의 결제에 따라 i에게 지급된 금액의 합계이다.

[0201] 요약하면 이상적인 시스템에서처럼 각 상태 S^{r+1}은 이전 결제 내역:

[0202] $PAY^0, \ldots, PAY^r$에서 추론할 수 있다.

[0203] **2.4 블록 및 입증된 블록**

[0204] $Algorand_0$에서 라운드 r에 해당하는 블록 B^r은 다음을 지정한다: r 자체; 라운드 r의 결제 세트, PAY^r; 설명될

수량 $SIG_{\ell^r}(Q^{r-1})$ 및 이전 블록의 해시 $H(B^{r-1})$. 따라서 고정 블록 B^0를 시작으로, 우리는 하기의 전통적인 블록체인을 가진다:

[0205] $$B^1 = (1, PAY^1, SIG_{\ell^1}(Q^0), H(B^0)), \quad B^2 = (2, PAY^2, SIG_{\ell^2}(Q^1), H(B^1)), \quad \ldots$$

[0206] Algorand에서 블록의 신뢰성은 실제로 별도의 정보 조각인 "블록 증명서(block certificate)" $CERT^r$에 의해 보증되어 B^r을 입증된 블록 $\overline{B^r}$으로 바꾼다. 따라서 매직 원장은 입증된 블록의 시퀀스로 구현된다:

[0207] $$\overline{B^1}, \overline{B^2}, \ldots$$

[0208] **토론**

[0209] 우리가 볼 수 있듯이, $CERT^r$은 $H(B^r)$에 대한 디지털 서명 세트, SV^r의 대다수 멤버의 디지털 서명 세트 및 각 멤버가 실제로 SV^r에 속한다는 증명과 함께 구성된다. 물론 블록 자체에 $CERT^r$ 증명서를 포함할 수 있지만, 별도로 유지하려면 개념적으로 더 명료하다는 것을 발견해야 한다.

[0210] 비트코인에서, 각 블록은 특수한 속성을 만족해야 한다. 즉, "비밀 퍼즐의 해답을 포함해야" 하고, 이는 블록 생성을 계산 집약적으로 만들고 포크가 필연적이거나 드물지 않게 만든다. 대조적으로, Algorand의 블록체인은 두 가지 주요 이점을 가지는데: 최소 계산으로 생성되며, 압도적으로 높은 확률로 포크하지 않는다는 것이다. 각 블록 B^i는 블록체인에 들어가자마자 안전하게 최종적이 된다.

[0211] **2.5 허용 가능한 실패 확률**

[0212] Algorand의 보안을 분석하기 위해, 우리는 무언가가 잘못되었다고 받아들일 확률 F를 지정한다(예를 들어, 검증자 세트 SV^r에 정직한 다수가 없음). 암호 해시 함수 H의 출력 길이의 경우에서와 마찬가지로, F도 파라미터이다. 그러나 이 경우에서와 마찬가지로, 그것은 F를 구체적인 값으로 설정하는 데에 유용하여 Algorand에서 실제로 충분한 보안과 충분한 효율성을 동시에 누릴 수 있다는 사실을 보다 직관적으로 파악할 수 있다. F가 원하는 대로 설정될 수 있는 파라미터라는 것을 강조하기 위해, 제1 및 제2 실시 예에서 다음과 같이 각각 설정한다:

[0213] $F = 10^{-12}$ 및 $F = 10^{-18}$.

[0214] **토론**

[0215] 10^{-12}는 실제로 1조분의 1보다 작으며, 우리는 F의 그러한 선택이 우리의 애플리케이션에서 적절하다고 믿는다. 우리가 10^{-12}가 적들이 정직한 사용자의 결제를 위조할 수 있는 확률이 아니라는 것을 강조하겠다. 모든 결제는 디지털 서명되어 있으므로, 적절한 디지털 서명을 사용하면 결제를 위조할 확률은 10^{-12}보다 훨씬 낮고, 실제로는 본질적으로 0이다. 우리가 기꺼이 확률 F로 용인할 수 있는 나쁜 사건은 Algorand의 블록체인이 포크하는 것이다. 우리의 F 설정과 1분간의 장거리 라운드에서는 Algorand의 블록체인에서 포크가 드물게 (약) 190만 년에 한 번 발생한다고 예상된다. 반대로 비트코인에서는 포크가 자주 발생한다.

[0216] 더 까다로운 사람은 F를 더 낮은 값으로 설정할 수 있다. 이를 위해, 우리의 제2 실시 예에서는, F를 10^{-18}로 설정하는 것을 고려한다. 블록이 1초마다 생성된다고 가정할 때, 10^{18}은 지금까지 빅뱅에서 현재까지 우주가 취한 예상 초수이다. 따라서 $F = 10^{-18}$일 때, 블록이 초 단위로 생성되면, 우주 나이 동안 포크를 볼 것으로 기대해야 한다.

[0217] **2.6 적 모델**

[0218] Algorand는 매우 적대적인 모델에서 안전하게 설계되었다. 설명한다.

[0219] **정직하고 악의적인 사용자**

[0220] 자신의 모든 프로토콜 명령을 따르고 메시지를 완벽하게 송수신할 수 있다면 사용자는 정직하다. 사용자가 자신의 규정된 명령에서 임의로 벗어날 수 있는 경우 사용자는 악의적(즉, 분산 컴퓨팅의 관점에서 비잔틴)이다.

[0221] **적들**

[0222] 적은 색상에 대해 개인화된 효율적인(기술적으로 다항-시간) 알고리즘이고, 이는 자신이 원하는 때에 언제든지 원하는 모든 사용자를 즉시 악의적으로 만들 수 있다(손상될 수 있는 사용자의 수를 상한으로만).

[0223] 적은 모든 악의적인 사용자를 완전히 통제하고 완벽하게 조정한다. 그는 모든 메시지를 수신하고 보내는 것을 포함하여 그들을 대신하여 모든 행동을 취하며, 임의의 방식으로 그들이 규정된 명령을 벗어나게 할 수 있다. 또는 그는 메시지를 송수신하는 손상된 사용자를 격리할 수 있다. 적이 그에게 취하는 동작에 의해 i의 악의가 발생할 수는 있지만 사용자 i가 악의적인 사용자임을 자동으로 알지 못한다는 것을 분명히 하자.

[0224] 그러나 이 강력한 적들은,

[0225] · 무한한 계산 능력을 갖지 못하며 무시할 확률을 제외하고는 정직한 사용자의 디지털 서명을 성공적으로 위조할 수 없고; 및

[0226] · 정직한 사용자 간의 메시지 교환을 어떤 식으로건 간섭할 수 없다.

[0227] 또한 정직한 사용자를 공격할 수 있는 그의 능력은 다음 가정 중 하나에 의해 제한된다.

[0228] **정직한 대다수의 돈**

[0229] 우리는 연속한 정직한 대다수의 돈(HMM) 즉, 각각의 음이 아닌 정수 k와 실수 h>1/2에 대한 가정을 고려한다.

[0230] HHM_k> h: 매 라운드 r마다 정직한 사용자가 라운드 r-k에서 시스템의 모든 돈을 h보다 큰 비율로 소유했다.

[0231] **토론.**

[0232] 모든 악의적인 사용자가 자신의 행동을 완벽하게 조정한다고 가정하면(단일한 엔티티, 적에 의해 통제되는 것처럼), 이는 비관적인 가설이다. 너무 많은 개인 간의 완벽한 조정은 성취하기가 어렵다. 아마도 조정은 악의적인 플레이어의 개별 그룹 내에서만 발생한다. 하지만 악의적인 사용자가 즐길 수 있는 조정 수준에 대해 확신할 수 없기 때문에, 우리는 미안하기 보다는 더 안전하다.

[0233] 적들이 비밀리에, 동적으로, 그리고 즉시 사용자를 손상시킬 수 있다고 가정하면 또한 비관적이다. 결국 현실적으로 사용자의 작업을 완전히 제어하려면 시간이 필요하다.

[0234] 예를 들어, 가정 $HMM_k > h$는 (평균적으로) 1분 안에 라운드가 실행된다면 주어진 라운드에서의 돈의 대부분은 k=120이면 적어도 2시간 동안 정직한 상태로 유지될 것이고, k=10,000인 경우 적어도 일주일 동안 정직한 상태로 유지될 것임을 암시한다.

[0235] HMM 가정과 이전의 정직한 대다수의 컴퓨팅 파워 가정은 컴퓨팅 파워를 돈으로 구입할 수 있기 때문에 악의적인 사용자가 대부분의 돈을 소유하면 컴퓨팅 능력의 대부분을 얻을 수 있다는 점에서 관련이 있다.

[0236] **2.7 통신 모델**

[0237] 우리는 메시지 전파, 즉 "피어 투 피어 가십"[4]이 유일한 통신 수단이며, 모든 전파된 메시지가 거의 모든 정직한 사용자에게 적시에 도달한다고 가정한다. 우리는 본질적으로 정직한 사용자에 의해 전파된 각 메시지 m이 모든 정직한 사용자에게 m의 길이에 의존하는 주어진 시간 내에 도달한다고 가정한다. (실제로 m은 정직한 사용자 중 충분히 높은 비율에 도달하는 것으로 충분하다.)

[0238] (4: 기본적으로 비트코인에서와 같이, 사용자가 메시지 m을 전파할 때, 메시지 m을 제1 시간동안 수신하는 모든 활성 사용자 i는 랜덤하게 그리고 독립적으로 적절하게 작은 수의 활성 사용자인 자신의 "이웃들"을 선택하고, 자신이 이웃들로부터 수신확인을 받을 때까지 자신의 이웃들에게 m을 포워딩한다. m의 전파는 제1 시간동안 m을 수신하는 사용자가 없을 때 종료된다.)

[0239] **3 전통적인 설정의 BA 프로토콜 BA***

[0240] 이미 강조한 바와 같이, 비잔틴 합의는 Algorand의 주요 구성 요소이다. 실제로, Algorand는 포크에 영향을 받지 않는 BA 프로토콜을 사용한다. 그러나 Algorand는 강력한 적에 대해 보안을 유지하기 위해 새로운 플레이어-교체가능 제약 조건을 충족하는 BA 프로토콜을 사용해야 한다. 또한 Algorand가 효율적이기 위해서는 이러한 BA 프로토콜이 매우 효율적이어야 한다.

[0241] BA 프로토콜은 이상적인 통신 모델, 동기식 완전 네트워크(SC 네트워크)에 대해 처음 정의되었다. 이러한 모델은 BA 프로토콜의 더 단순한 설계 및 분석을 가능하게한다. 따라서 이 절에서는 SC 네트워크를 위한 새로운 BA 프로토콜 $BA^{\star}$를 소개하고 플레이어 교체 가능성 문제를 모두 무시한다. $BA^{\star}$ 프로토콜은 별도의 값을 제공한다. 사실, 이것은 지금까지 알려진 SC 네트워크를 위한 가장 효율적인 암호화 BA 프로토콜이다.

[0242] 우리 Algorand 프로토콜 내에서 그것을 사용하기 위해, 우리는 우리의 다른 통신 모델 및 컨텍스트를 설명하기 위해 $BA^{\star}$를 조금 수정했다.

[0243] 우리는 $BA^{\star}$가 운영되는 모델과 비잔틴 합의의 개념을 상기함으로써 시작한다.

[0244] **3.1 동기식 완전한 네트워크와 매칭하는 적**

[0245] SC 네트워크에서, 각각의 적분 시간 r = 1, 2, ...에 똑딱 거리는 공통 클록이 있다.

[0246] 각각의 짝수 시간에 r을 클릭하면, 각 플레이어 i는 자신을 포함하여 각 플레이어 j에게 즉각적으로 그리고 동시에 하나의 메시지 $m_{i,j}^{r}$ (아마도 빈 메시지)를 보낸다. 각 $m_{i,j}^{r}$는 송신자 i의 신원과 함께 플레이어 j에 의해 시간 r+1에서 정확하게 수신된다.

[0247] 다시 말하면, 통신 프로토콜에서, 플레이어는 자신이 규정한 모든 명령을 따르는 경우 정직하고, 그렇지 않으면 악의적이다. 모든 악의적인 플레이어는 악의적인 플레이어를 대상으로 하는 모든 메시지를 즉시 수신하고 그들이 전송하는 메시지를 선택하는 적에 의해 완전히 통제되고 완벽하게 조정된다.

[0248] 적은 악의적인 플레이어의 수에 상응하는 상한선 t까지만 원하는 홀수 시간에 원하는 모든 정직한 사용자를 즉시 악의적으로 만들 수 있다. 즉, 적은 "정직한 사용자 i가 이미 보낸 메시지를 간섭할 수 없고", 이는 평소와 같이 전달된다.

[0249] 적은 또한 현재 정직한 플레이어가 보내는 메시지를 각각의 짝수 라운드에서도 즉각적으로 볼 수 있는 추가 기능을 갖추고 있으며, 이 정보를 즉시 사용하여 악의적인 플레이어가 동시에 보낸 메시지를 선택한다.

[0250] **3.2 비잔틴 합의의 개념**

[0251] 비잔틴 합의의 개념은 이진 경우, 즉, 모든 초기 값이 비트로 구성되어있을 때에 처음 도입되었을 수 있다. 그러나 이것은 임의의 초기 값으로 빠르게 확장되었다. BA 프로토콜에 의해, 우리는 임의의 값 하나를 의미한다.

[0252] **정의 3.1.**

[0253] 동기식 네트워크에서, $\mathcal{P}$는 그의 플레이어 세트가 플레이어들 사이의 공통 지식인 n 플레이어 프로토콜이고, t는 n≥ 2t+1이 되도록 양의 정수가 된다. $\mathcal{P}$는, 적어도 확률 σ를 가지고 하기의 2가지 조건을 충족하기 위해, 특수 기호 $\perp$를 포함하지 않는 값 V의 모든 세트에 대해(각각 V={0, 1}에 대해), 실행시 플레이어의 대부분인 t가 악의적이고, 여기서 모든 플레이어 i는 초기 값 $v_i \in V$로 시작하고, 모든 정직한 플레이어 j는 확률 1로 정지하고, 값 $out_i \in V \cup \{\perp\}$을 출력하는 경우, 건전성 $\sigma \in (0,1)$을 가지는 임의 값(각각 이진) (n, t)-비잔틴 합의 프로토콜이라고 한다:

[0254] 1. 합의: 모든 정직한 플레이어 i에 대해 out_i=out이 되도록 $out \in V \cup \{\perp\}$이 존재한다

[0255] 2. 일관성: 어떤 값 $v_i \in V$에 대해, 모든 플레이어 i에 대해 v_i=v인 경우, out=v이다.

[0256] 우리는 out을 $\mathcal{P}$'의 출력이라고 하고, 그리고 각 out_i에 대해 플레이어 i의 출력이라고 한다.

[0257] **3.3 BA 표기법 #**

[0258] 우리의 BA 프로토콜에서, 플레이어는 얼마나 많은 플레이어가 주어진 단계에서 주어진 메시지를 그에게 보냈는지 계산할 필요가 있다. 따라서, 전송될 수 있는 각각의 가능한 값 v에 대해,

[0259] $\#_i^s(v)$

[0260] (또는 s가 명확할 때 단지 $\#_i(v)$)는 단계 s에서 i가 v를 수신한 플레이어 j의 수이다.

[0261] 플레이어 i가 각 플레이어 j로부터 정확히 하나의 메시지를 받는다는 것을 상기하고, 플레이어의 수가 n이라면 모든 i와 s에 대해 $\sum_v \#_i^s(v) = n$을 구한다.

[0262] **3.4 새로운 이진 BA 프로토콜 BBA⋆**

[0263] 이 섹션에서는 플레이어의 3분의 2 이상의 정직을 기반으로 하고 매우 빠른 새로운 이진 BA 프로토콜 BBA⋆를 소개한다. 악의적인 플레이어가 어떤 작업을 수행하든, 그의 메인 루프의 각 실행은 사소하게 실행될 뿐만아니라, 플레이어를 확률 1/3로 동의하게 만든다.

[0264] BBA⋆에서, 각 플레이어는 고유 서명 속성을 만족하는 디지털 서명 체계의 자신의 공개키를 가지고 있다. 이 프로토콜은 동기식 완전 네트워크에서 실행되기 때문에 자신의 각 메시지에 플레이어 i가 서명할 필요가 없다.

[0265] 디지털 서명은 3단계에서 충분히 공통 랜덤 비트를 생성하는 데 사용된다. (Algorand에서 디지털 서명은 다른 모든 메시지도 인증하는 데 사용된다.)

[0266] 프로토콜은 최소 셋업: 플레이어의 키와 독립적인 공통 랜덤 스트링 r을 필요로 한다.(Algorand에서 r은 실제로 양 Q^r로 대체된다.)

[0267] 프로토콜 BBA⋆는 플레이어가 부울 값을 반복적으로 교환하고 상이한 플레이어가 상이한 시간에 이 루프를 종료할 수 있는 3단계 루프이다. 플레이어 i는 일부 단계에서 특수 값 0* 또는 특수 값 1*을 전파함으로써 이 루프를 빠져나와 모든 플레이어가 모든 미래의 단계에서 i로부터 0과 1을 각각 수신하는 "척"을 하도록 지시한다. (다른 말로 표현하자면, 다른 플레이어 i로부터 플레이어 j가 받은 마지막 메시지가 비트 b라고 가정한다. 그런 다음, 그가 i로부터 어떤 메시지도 받지 않은 모든 단계에서, j는 그에게 비트 b를 보낸 것처럼 행동한다.)

[0268] 프로토콜은 자신의 3단계 루프가 실행된 횟수를 나타내는 카운터 γ를 사용한다. BBA⋆ 시작시, γ=0이다. (γ은 글로벌 카운터라고 생각할 수 있지만, 루프가 실행될 때마다 각각의 개별 플레이어에 의해 실제로 증가한다.)

[0269] $n \geq 3t+1$이 있고, 여기서 t는 가능한 최대의 악의적인 플레이어의 수이다. 이진수 스트링 x는 그의 2진 표현(가능한 0을 유도하는)이 x이고; lsb(x)가 x의 최하위 비트를 나타내는 정수로 식별된다.

[0270] 프로토콜 BBA⋆

[0271] **(통신) 1단계.** [0으로 고정된 코인(Coin-Fixed-To-0) 단계] 각 플레이어 i는 b_i를 전송한다.

[0272] 1.1 $\#_i^1(0) \geq 2t+1$이면, i는 b_i=0로 설정하고, 0*을 전송하고, out_i=0을 출력하고, HALT한다.

[0273] 1.2 $\#_i^1(1) \geq 2t+1$이면, i는 b_i=1을 설정한다.

[0274] 1.3 그렇지 않으면, i는 b_i = 0으로 설정한다.

[0275] **(통신) 2단계.** [1로 고정된 코인(Coin-Fixed-To-1) 단계] 각 플레이어 i는 b_i를 전송한다.

[0276] 2.1 $\#_i^2(1) \geq 2t+1$이면, i는 b_i=1로 설정하고, 1*을 전송하고, out_i=1을 출력하고, HALT한다.

[0277] 2.2 $\#_i^2(0) \geq 2t+1$이면 b_i=0으로 설정한다.

[0278] 2.3 그렇지 않으면, i는 b_i = 1로 설정한다.

[0279] **(통신) 3단계.** [순수하게 플리핑된 코인(Coin-Genuinely-Flipped) 단계] 각 플레이어 i는 b_i와 $SIG_i(r,\gamma)$를 전송한다.

[0280] 3.1 $\#_i^3(0) \geq 2t+1$이면, i는 b_i=0로 설정한다.

[0281] 3.2 $\#_i^3(1) \geq 2t+1$이면 i는 b_i=1을 설정한다.

[0282] 3.3 그렇지 않으면 S_i = {j∈N(이 단계 3에서 적절한 메시지를 i에게 전송한)}으로 놓고, i는 $b_i = c \triangleq \mathtt{lsb}(\min_{j\in S_i} H(SIG_i(r,\gamma)))$로 설정하고; γ_i를 1씩 증가시키고; 및 다시 단계 1로 복귀한다.

[0283] **정리 3.1.**

[0284] n≥3t+1일 때마다, BBA*는 건전성(soundness) 1을 가진 이진(n, t)-BA 프로토콜이다.

[0285] 정리 3.1의 증명은 https://people.csail.mit.edu/silvio/Selected-ScientificPapers/DistributedComputation/BYZANTINEAGREEMENTMADETRIVIAL.15 pdf에서 볼 수 있다.

[0286] **3.5 단계적(Graded) 합의와 프로토콜 GC**

[0287] 임의의 값에 대해, 우리는 비잔틴 합의보다 훨씬 약한 합의의 개념을 생각해보자.

[0288] **정의 3.2.** $\mathcal{P}$는 모든 플레이어들의 세트가 공통적인 지식인 프로토콜이고, 각 플레이어 i는 개인적으로 임의의 초기 값 v_i'를 알고 있다고 하자.

[0289] 우리는 $\mathcal{P}$를 그의 대부분인 t가 악의적인 n 명의 플레이어로 실행될 때마다, 모든 정직한 플레이어 i가 가치 등급 쌍(v_i, g_i)을 출력하는 것을 중단하는 경우, 하기의 3가지 조건을 만족시키기 위해 $g_i \in \{0,1,2\}$인 (n, t)-단계적 합의 프로토콜이라고 한다:

[0290] 1. 모든 정직한 플레이어 i와 j에 대해, $|g_i - g_j| \leq 1$이다.

[0291] 2. 모든 정직한 플레이어 i와 j에 대해, $g_i, g_j > 0 \Rightarrow v_i = v_j$이다.

[0292] 3. 일부 값 v에 대해 $v_1' = \cdots = v_n' = v$인 경우, 모든 정직한 플레이어 i에 대해 $v_i = v$이고 g_i=2이다.

[0293] 다음 두 단계 프로토콜 GC는 문헌에서 단계적 합의 프로토콜이다. 4.1 절의 프로토콜 $Algorand_1'$의 단계를 매칭하기 위해, 각각 GC의 단계 2와 3을 명명한다. (실제로 $Algorand_1'$의 제1 단계는 다른 것, 즉, 다른 블록을 제안하는 것과 관련된 것이다.)

[0294] 프로토콜 GC

[0295] 2단계. 각 플레이어 i는 v_i'를 모든 플레이어에게 전송한다.

[0296] 3단계. 각 플레이어 i는 $\#_i^2(x) \geq 2t+1$인 경우 그리고 이 경우에만 모든 플레이어들에게 스트링 x를 전송한다.

[0297] **출력 정의.**

[0298] 각 플레이어 i는 다음과 같이 계산된 쌍(v_i, g_i)을 출력한다.

[0299] · 일부 x에 대해, $\#_i^3(x) \geq 2t+1$인 경우, v_i=x이고, g_i=2이다.

[0300] · 일부 x에 대해, $\#_i^3(x) \geq t+1$인 경우, v_i=x이고, g_i=1이다.

[0301] · 그렇지 않으면, $v_i = \perp$ 및 g_i=0이다.

[0302] 프로토콜 GC는 문헌에서 프로토콜이므로, 다음의 정리가 성립하는 것으로 알려져 있다.

[0303] **정리 3.2.** $n \geq 3t+1$이면 GC는 $a\ (n,t)$-단계적 방송 프로토콜이다.

[0304] **3.6 프로토콜 BA***

[0305] 우리는 이제 바이너리 BA 프로토콜 BBA*와 단계적 합의 프로토콜 GC를 통해 임의의 값 BA 프로토콜 BA*를 설명한다. 아래에서 각 플레이어 i의 초기 값은 v_i'이다.

[0306] 프로토콜 BA*

[0307] 단계 1 및 2. 각 플레이어 i는 입력 v_i'에 대해 GC를 실행하여 쌍(v_i, g_i)을 계산한다.

[0308] 단계 3, ... 각 플레이어 i는 비트 out_i를 연산하기 위해 g_i=2인 경우는 최초 입력 값 0을 가지고, 그렇지 않은 경우에는 1을 가지고 BBA*를 실행한다.

[0309] **출력 결정.** 각 플레이어 i는 out_i=0인 경우 v_i를 출력하고 그렇지 않은 경우는 $\perp$를 출력한다.

[0310] **정리 3.3.** n≥3t+1이면 언제나, BA*는 건전성 1을 가진 $a\ (n,t)$-BA 프로토콜이다.

[0311] 증명. 우리는 먼저 일관성(Consistency)을 증명하고 그런다음 합의(Agreement)를 증명한다.

[0312] **일관성의 증명**. 일부 값 $v \in V$에 대해, $v_i' = v$라고 가정한다. 단계적 합의의 속성 3에 의해, GC 실행 후 모든 정직한 플레이어가 (v, 2)를 출력한다. 따라서 0은 BBA* 실행이 끝난 모든 정직한 플레이어의 최초 비트이다. 따라서 이진 비잔틴 합의의 합의 속성에 의해, BA*의 실행이 끝날 때 모든 정직한 플레이어에 대해 out_i = 0이 된다. 이것은 BA*에서 각 정직한 플레이어 i의 출력이 $v_i = v$임을 의미한다. □

[0313] 합의 증명. BBA*는 이진 BA 프로토콜이기 때문에

[0314] (A) 모든 정직한 플레이어 i에 대해, out_i = 1이거나

[0315] (B) 모든 정직한 플레이어 i에 대해 out_i = 0이다.

[0316] A의 경우, 모든 정직한 플레이어가 BA*에서 $\perp$을 출력하므로 합의가 유지된다. 이제 B 경우를 고려해보자. 이 경우, BBA*의 실행에서 적어도 하나의 정직한 플레이어 i의 최초 비트는 0이다. (실제로 모든 정직한 플레이어의 최초 비트가 1이면, BBA*의 일관성 속성에 의해, 우리는 모든 정직한 j에 대해 out_j = 1을 가질 것이다.) 따라서, GC 실행 후, i는 일부 값 v에 대해 쌍(v, 2)를 출력한다. 따라서, 단계적 합의의 속성 1에 의해 모든 정직한 플레이어 j에 대해 g_j > 0이다. 따라서, 단계적 합의의 속성 2에 의해, 모든 정직한 플레이어 j에 대해 $v_j = v$ 이다. 이것은 BA*의 종료시 모든 정직한 플레이어 j가 v를 출력함을 의미한다. 따라서 합의는 B의 경

우에도 유지된다. □

[0317] 일관성과 합의가 모두 유지되므로 $BA^{\star}$는 임의 값 BA 프로토콜이다. ■

[0318] 프로토콜 $BA^{\star}$는 가쉽 네트워크에서도 작동하며, 사실상 Algorand가 예상되는 매우 적대적인 모델에서 안전하게 유지되는 데 중요한 플레이어 교체 가능성 속성을 만족시킨다.

[0319] **$BBA^{\star}$와 $BA^{\star}$의 플레이어 교체 가능성**

[0320] 이제는 $BA^{\star}$와 $BBA^{\star}$ 프로토콜이 피어-투-피어 가쉽을 통해 통신이 이루어지는 네트워크에서 실행되도록 조정될 수 있고 플레이어 교체 가능성을 만족시키는지에 대한 몇 가지 직관을 제공해보자. 구체적으로는 네트워크에 10M 명의 사용자가 있고 $BBA^{\star}$(또는 $BA^{\star}$)의 각 단계 x가 비밀 암호 분류를 통해 무작위로 선택되고, 따라서 단계 x에서 메시지를 전송할 권한이 있다고 증명하는 보증서를 가지는 10,000 명의 플레이어로 구성된 위원회에 의해 실행된다고 가정한다. 주어진 단계에서 전송된 각 메시지는 단계 번호를 지정하고, 그의 발신자가 디지털 서명하고, 그의 발신자가 해당 단계에서 말하도록 권한이 있음을 증명하는 보증서를 포함한다고 가정한다.

[0321] 우선, 정직한 플레이어의 비율 h가 2/3(예를 들어, 75%)보다 충분히 크다면, 압도적인 확률로 각 단계에서 선택된 위원회는 요구되는 2/3의 정직한 대다수를 가진다.

[0322] 또한, 1만명의 강력하게 무작위로 선정된 위원회가 각 단계에서 변경된다는 사실은 $BBA^{\star}$ 또는 $BA^{\star}$ 중 어느 하나의 올바른 작동을 저해하지 않는다. 실제로, 어느 프로토콜에서든, 단계 s에서의 플레이어 i는 그가 단계 s-1에서 주어진 메시지 m을 수신한 다중성에만 반응한다. 우리가 가쉽 네트워크에 있기 때문에 단계 s-1에서 보낸 모든 메시지는 단계 s에서 플레이하도록 선택된 메시지를 포함하여 모든 사용자에게 전달될 것이다(이 직관의 목적을 위해 즉시). 또한, 단계 s-1에서 전송된 모든 메시지가 단계 번호를 지정하고 보증서를 포함하기 때문에, 발신자는 실제로 단계 s-1에서 발언하도록 승인된다. 따라서, 단계 s-1에서 우연히 그가 선택되었는지 여부에 관계없이, 단계(s)에서 플레이하도록 선택된 사용자 i는 그가 정확한 단계 s-1 메시지를 수신한 다중성을 정확하게 카운트할 수 있다. 그가 모든 단계를 지금까지 플레이했는지 여부는 중요하지 않는다. 모든 사용자는 "같은 배"에 있으므로 다른 사용자가 쉽게 교체할 수 있다.

[0323] **4 Algorand의 두 가지 실시예**

[0324] 논의된 바와 같이 매우 높은 수준에서 Algorand 라운드는 다음과 같이 이상적으로 진행된다. 먼저, 무작위로 선택된 사용자인 리더는 새로운 블록을 제안하고 배포한다.(이 프로세스는 초기에는 몇 명의 잠재적인 리더를 선정하고, 그런 다음 최소한의 시간 동안 단 하나의 공통 리더가 나온다고 보장하는 것을 포함한다.) 둘째, 무작위로 선택된 사용자 위원회가 선정되어 리더에 의해 제안된 블록에 대한 비잔틴 합의에 이른다. (이 프로세스는 BA 프로토콜의 각 단계가 별도로 선택된 위원회에 의해 운영되는 것을 포함한다.) 합의된 블록은 위원회 멤버의 주어진 임계 값(T_H)에 의해 디지털 서명된다. 이러한 디지털 서명은 모든 사람이 새로운 블록인지 확신할 수 있도록 전파된다. (서명자의 보증서를 회람하고 새로운 블록의 해시를 인증하여, 해시가 명확해지면 모든 사람이 블록을 배울 수 있도록 보장한다.)

[0325] 다음 두 섹션에서는 적절한 대다수의 정직한 사용자 가정하에서 각각 작동하는 기본 Algorand 설계인 $Algorand'_1$ 및 $Algorand'_2$의 두 가지 구현을 제시한다. 8절에서는 이러한 실시 예들이 정직한 대다수의 돈의 가정하에서 작동하도록 조정하는 방법을 보여준다.

[0326] $Algorand'_1$은 위원회 멤버 중 2/3 이상이 정직하다고 생각한다. 또한, $Algorand'_1$에서는 비잔틴 합의에 도달하기 위한 단계 수를 적절히 높은 수로 설정하여 고정된 수의 단계 내에서 압도적인 확률로 합의가 이루어지도록 보장하도록 한다(단, $Algorand'_2$의 단계들보다 긴 시간이 필요함). 마지막 단계에 의해 합의가 아직 도달되지 못한 원격의 경우, 위원회는 항상 유효한 빈 블록에 동의한다.

[0327] $Algorand'_2$는 위원회에서 정직한 멤버의 수는 항상 고정된 임계 값 t_H 이상임을(이는 압도적인 확률로 적어도 위원회 멤버의 2/3이 정직하다는 것을 보장한다) 예상한다. 또한 $Algorand'_2$는 임의의 수의 단계에서 비

잔틴 합의에 도달할 수 있도록 한다(단, $Algorand'_1$ 보다 잠재적으로 더 짧은 시간에).

[0328] 당업자는 이러한 기본 실시 예의 많은 변형 예를 도출할 수 있다는 것을 알 것이다. 특히, 임의의 수의 단계로 비잔틴 합의에 도달할 수 있게 하기 위해 $Algorand'_2$가 주어지면, $Algorand'_1$을 수정하는 것은 쉽다.

[0329] 두 가지 실시 예는 다음과 같은 공통 코어, 표기법, 개념 및 파라미터를 공유한다.

[0330] **4.1 공통 코어**

[0331] **목적**

[0332] 이상적으로 각 라운드 r에 대해 Algorand는 다음 속성을 만족해야 한다:

[0333] 1. 완벽한 정확성. 모든 정직한 사용자는 동일한 블록 B^r에 동의한다.

[0334] 2. 완전성 1. 확률 1로 블록 B^r이 정직한 사용자에 의해 선택되었다.

[0335] (실제로 악의적인 사용자는 항상 자신의 페이셋이 자신의 단지 그의 "친구"의 결제를 포함하는 블록을 선택할 수 있다.)

[0336] 물론 완벽한 정확성만 보장하는 것은 쉬운 일이다. 모든 사람은 항상 공식 페이셋 PAY^r을 비워 두도록 선택한다. 그러나 이 경우, 시스템의 완전성은 0이다. 불행히도 완벽한 정확성과 완전성 1을 모두 보장하는 것은 악의적인 사용자가 있는 경우 쉽지 않다. 따라서 Algorand는 더 현실적인 목표를 채택한다. 비공식적으로, h를 정직한 사용자의 비율(h> 2/3)으로 나타내면, Algorand의 목표는 다음과 같다:

[0337] 압도적인 확률로 h에 근접한 완벽한 정확성과 완전성을 보장하는 것.

[0338] 완전성에 대한 정확성의 특권은 합리적인 선택으로 보인다: 한 라운드에서 처리되지 않은 결제는 다음 번에 처리될 수 있지만 가능한 경우 포크를 피해야 한다.

[0339] **지배받는(led) 비잔틴 합의**

[0340] 과도한 시간과 통신을 잠시 무시하면, 다음과 같이 완벽한 정확성을 보장할 수 있다. 라운드 r의 시작에서, 각 사용자 i는 자신의 후보 블록 B_i^r를 제안한다. 그런 다음 모든 사용자는 후보 블록 중 하나에 대한 비잔틴 합의에 도달한다. 우리의 소개에 따르면, 채택된 BA 프로토콜은 2/3의 정직한 다수를 요구하며 플레이어 교체가 가능하다. 각 단계는 내부 변수를 공유하지 않는 작고 무작위로 선택된 검증자의 세트에 의해 실행될 수 있다.

[0341] 불행하게도 이 방법은 잘 효과가 없다. 정직한 사용자가 제안한 후보 블록이 서로 대부분 완전히 다를 수 있기 때문에 이는 매우 그렇다. 실제로 각 정직한 사용자는 상이한 결제를 본다. 따라서 상이한 정직한 사용자가 보는 결제의 세트가 다수 겹치기는 하지만, 모든 정직한 사용자가 동일한 블록을 제안할 가능성은 거의 없다. 따라서 BA 프로토콜의 일관성 동의는 결코 구속력이 없으며, 합의만 일치하므로, 합의는 좋은 블록이 아닌 $\perp$에 항상 도달할 수 있다.

[0342] $Algorand'$는 다음과 같이 이 문제를 방지한다. 먼저 라운드 r의 리더 ℓ^r가 선택된다. 그런 다음 ℓ^r는 자신의 후보 블록 $B_{\ell^r}^r$을 전파한다. 마지막으로 사용자는 그들이 ℓ^r로부터 실제 수신한 블록에 대한 합의에 도달한다. 왜냐하면, ℓ^r이 언제나 정직하고 완벽한 정확성과 완전성 1이 모두 유지되기 때문에, $Algorand'$는 ℓ^r이 h에 가까운 확률로 정직하다는 것을 보장한다.

[0343] **리더 선택**

[0344] Algorand에서 r 번째 블록은 다음과 같은 형식이다.

[0345] $$B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$$

[0346] 도입부에서 이미 언급했듯이, 양 Q^{r-1}은 우리의 매우 강력한 적들이 본질적으로 조작할 수 없도록 조심스럽게 구축된다(이 절의 후반부에서 우리는 이것이 왜 그런지에 대한 직관을 제공할 것이다). 라운드 r의 시작에서, 모든 사용자는 모든 이전 라운드의 사용자 세트 즉 PK^{1}, ..., PK^{r-1}을 추론하는 지금까지 블록체인 B^{0}, ..., B^{r-1}을 알고 있다. 라운드 r의 잠재적 리더는 .

[0347] $.H\left(SIG_i\left(r,1,Q^{r-1}\right)\right) \leq p$ 이 되도록 사용자 i가 된다.

[0348] 설명해보자. 양 Q^{r-1}은 블록 B^{r-1}에서 추론할 수 있기 때문에 기본 디지털 서명 체계의 메시지 검색 가능성 특성 때문에, 또한 기본 서명 체계는 고유성 특성을 만족시킨다. 따라서, $SIG_i(r, 1, Q^{r-1})$은 i와 r에 고유하게 연관된 이진 스트링이다. 따라서, H는 랜덤 오라클이기 때문에, $H(SIG_i(r, 1, Q^{r-1}))$은 i와 r에 고유하게 연관된 랜덤한 256비트 길이의 스트링이다. $H(SIG_i(r, 1, Q^{r-1}))$ 앞의 기호 "."는 10진수(우리의 경우 이진수) 포인트여서, $r_i \triangleq .H\left(SIG_i\left(r,1,Q^{r-1}\right)\right)$는 i와 r에 고유하게 연관된 0과 1 사이의 랜덤한 256비트 수의 이진 확장이 되도록 한다. 따라서 r_i가 p이하일 확률은 본질적으로 p이다.

[0349] 확률 p는 압도적인(즉, 1-F) 확률로 적어도 하나의 잠재적인 검증자가 정직하도록 선택된다. (사실, p는 가장 작은 확률이 되도록 선택된다.)

[0350] i가 자신의 서명을 계산할 수 있는 유일한 사람이기 때문에, 그는 자신이 라운드 1의 잠재적인 검증자인지 여부를 단독으로 판정할 수 있다. 그러나 자신의 보증서, $\sigma_i^r \triangleq SIG_i\left(r,1,Q^{r-1}\right)$를 나타냄으로써, i는 라운드 r의 잠재적인 검증자라는 것을 임의의 사람에게 증명할 수 있다.

[0351] 리더 ℓ^r는 자신의 해시된 보증서가 다른 모든 잠재적 리더 j의 해시된 보증서 보다 작은 잠재적 리더로, 즉, $H(\sigma_{\ell^r}^{r,s}) \leq H(\sigma_j^{r,s})$로 정의된다.

[0353] *악의적인 ℓ^r이 자신의 보증서를 밝힐 수 없으므로 r 라운드의 정확한 리더를 알 수 없으며, 가능성이 없는 관계를 제외하면 ℓ^r가 실제로 라운드 r의 유일한 리더이다.

[0354] 마지막으로 최종이지만 중요한 세부 정보를 가져오도록 하겠다. 적어도 k 라운드에 대한 시스템에 그가 속해있는 경우에만 사용자 i는 라운드 r의 잠재적 리더(그리고 그에 따른 리더)가 될 수 있다. 이것은 Q^r과 모든 미래의 Q-수량의 비조작 가능성을 보장한다. 실제로, 잠재적 리더 중 한 명이 실제로 Q^r를 판정할 것이다.

[0355] **검증자 선택**

[0356] 라운드 r의 각 단계 s> 1은 작은 검증자 세트 $SV^{r,s}$에 의해 실행된다. 다시 말하면, 각각의 검증자 $i \in SV^{r,s}$는 r 이전에 시스템 k에 이미 있는 사용자들 사이에서, 그리고 다시 특수한 양 Q^{r-1}을 통해 사용자들 사이에서 무작위로 선택된다. 특히,

[0357] $.H\left(SIG_i\left(r,s,Q^{r-1}\right)\right) \leq p'$ 인 경우,

[0358] $i \in PK^{r-k}$는 $SV^{r,s}$에서의 검증자이다.

[0359] 다시 한 번, i만 그가 $SV^{r,s}$에 속해 있는지를 알지만, 이 경우 그는 자신의 보증서 $\sigma_i^{r,s} \triangleq H(SIG_i\left(r,s,Q^{r-1}\right))$을 표시함으로써 그것을 증명할 수 있다. 검증자 $i \in SV^{r,s}$는 라운드 r의 단계 s에서 메시지 $m_i^{r,s}$를 송신하고, 이 메시지는 자신의 보증서 $\sigma_i^{r,s}$을 포함하여, 검증자 f가 다음 단계에서

$m_i^{r,s}$ 이 정당한 단계 s의 메시지라는 것을 인지할 수 있도록 한다.

[0360] 확률 p' 는 $SV^{r,s}$에서 #good가 정직한 사용자의 수이고 #bad가 악의적인 사용자의 수로 놓는 것을 보장하기 위해 선택되며, 압도적인 확률로 다음 두 조건이 유지된다:

[0361] 실시 예 $Algorand'_1$에 대해:

[0362] (1) $\#good > 2 \cdot \#bad$ 그리고

[0363] (2) $\#good + 4 \cdot \#bad < 2n$, 여기서 n은 $SV^{r,s}$의 예상 카디널리티이다.

[0364] 실시 예 $Algorand'_2$에 대해:

[0365] (1) $\#good > t_H$ 그리고

[0366] (2) $\#good + 2\#bad < 2t_H$, 여기서 t_H는 지정된 임계 값이다.

[0367] 이러한 조건은 충분히 높은 확률로, (a) BA 프로토콜의 마지막 단계에서, 새로운 블록 B^r에 디지털 서명하는 적어도 주어진 수의 정직한 플레이어가 있을 것이고, (b) 라운드당 하나의 블록만이 필요한 수의 서명을 가질 수 있고, 및 (c) 사용된 BA 프로토콜은(각 단계에서) 필요로하는 2/3의 정직한 대다수를 가진다는 것을 의미한다.

[0368] **블록 생성 명료화**

[0369] 라운드 r 리더 ℓ^r가 정직한 경우, 해당 블록은 다음과 같은 형식이다.

[0370] $$B^r = \left(r, PAY^r, SIG_{\ell^r}\left(Q^{r-1}\right), H\left(B^{r-1}\right)\right)$$

[0371] 여기서 페이셋 PAY^r은 최대 값이다. (모든 페이셋은 정의에 의해 집합적으로 유효함을 상기하라).

[0372] 그렇지 않은 경우(즉, ℓ^r가 악의적인 경우), B^r은 다음 두 가지 가능한 형태 중 하나를 가진다:

[0373] $B^r = \left(r, PAY^r, SIG_i\left(Q^{r-1}\right), H\left(B^{r-1}\right)\right)$ 및 $B^r = B_\varepsilon^r \triangleq \left(r, \emptyset, Q^{r-1}, H\left(B^{r-1}\right)\right)$.

[0374] 제1 형태에서, PAY^r은 (반드시 최대가 아닌) 페이셋이며 그것은 $PAY^r = \emptyset$일 수 있고, i는 라운드 r의 잠재적 리더이다. (그러나 i는 리더 ℓ^r가 아닐 수도 있다. 이것은 ℓ^r가 자신의 보증서를 비밀로 유지하고 자신을 밝히지 않는 경우 실제로 발생할 수 있다.)

[0375] 두 번째 형태는 BA 프로토콜의 라운드-r 실행에서 모든 정직한 플레이어가 디폴트값을 출력할 때 발생하고, 이것은 우리의 애플리케이션의 빈 블록 B_ε^r이다. (정의에 의해, BA 프로토콜의 가능한 출력은 일반적으로 $\perp$로 표시되는 디폴트값을 포함한다. 3.2 절 참조)

[0376] 페이셋이 두 경우 모두 비어 있더라도, $B^r = (r, \emptyset, SIG_i(Q^{r-1}), H(B^{r-1}))$ 및 B_ε^r은 문법적으로 상이한 블록이며 두 가지 상이한 상황: 각각 "BA 프로토콜의 실행에서 모두 부드럽게 충분하고", "BA 프로토콜에서 일부 문제가 발생하여 디폴트값이 출력되는" 상황에서 발생한다는 것에 유의하라.

[0377] 이제 우리는 $Algorand'$의 라운드 r에서 블록 B^r의 생성이 어떻게 진행되는지 직관적으로 설명하겠다. 제1 단계에서 자격을 갖춘 각 플레이어, $i \in PK^{r-k}$인 각각의 플레이어는 그가 잠재 리더인지 여부를 확인한

다. 이 경우, 지금까지 본 모든 결제와 현재 블록체인 B^0, ..., B^{r-1}을 사용하여 최대 결제 세트인 PAY_i^r을 은밀하게 준비하고 자신의 후보 블록 B^r =(r, PAY_i^r, $SIG_i(Q^{r-1})$, $H(B^{r-1})$)을 은밀하게 조립하도록 i는 요청받는다. 즉, 그는 자신의 두 번째 구성 요소로서 방금 준비된 페이셋뿐만 아니라, 마지막 블록인 B^{r-1}의 자신의 제3 구성 요소로서 Q^{r-1}의 자신의 서명을 B_i^r에 포함한다. 마지막으로, 그는 (a) 그의 후보 블록 B_i^r, (b) 그의 후보 블록에 대한 그의 적절한 서명(즉, B_i^r의 해시의 그의 서명), 및 (c) 자신이 라운드 r의 잠재적 검증자임을 증명하는 자신의 보증서 $\sigma_i^{r,1}$을 포함하는 자신의 라운드 r 단계 1 메시지 $m_i^{r,1}$을 전파한다.

[0378] (정직한 i가 자신의 메시지 $m_i^{r,1}$을 산출할 때까지, 적은 i가 잠재적인 검증자라는 단서가 없다는 것에 유의하라. 정직한 잠재 리더를 손상시키려 한다면, 적들은 또한 랜덤한 정직한 플레이어를 손상시킬 수 있다. 그러나. 그가 $m_i^{r,1}$을 보면 그것은 i의 보증서를 가지고 있기 때문에, 적들은 i를 알고 있고 손상시킬 수 있지만, 바이러스처럼 전파된 $m_i^{r,1}$이 시스템의 모든 사용자에게 도달하는 것을 막을 수는 없다.)

[0379] 제2 단계에서, 각각의 선택된 검증자 $j \in SV^{r,2}$는 라운드의 리더를 식별하려고 시도한다. 구체적으로 j는 자신이 수신한 적절한 제1 단계 메시지 $m_i^{r,1}$에 포함된 1단계 보증서 $\sigma_{i_1}^{r,1}, \ldots, \sigma_{i_n}^{r,1}$을 취하고; 그것들 모두를 해시한다. 즉, $H\left(\sigma_{i_1}^{r,1}\right), \ldots, H\left(\sigma_{i_n}^{r,1}\right)$)를 계산하고; 그의 해시가 사전학적으로 최소인 보증서 $\sigma_{\ell_j}^{r,1}$을 찾고; ℓ_j^r를 라운드 r의 리더로 간주한다.

[0380] 고려된 각 보증서는 Q^{r-1}의 디지털 서명이며 $SIG_i(r, 1, Q^{r-1})$)이 i 및 Q^{r-1}에 의해 고유하게 판정되고, H가 랜덤 오라클이고, 따라서 각각의 $H(SIG_i(r, 1, Q^{r-1}))$은 라운드 r의 각각의 잠재적 리더 i에 대해 고유한 랜덤한 256비트 길이의 스트링인 Q^{r-1}의 디지털 서명이라는 것을 상기하라.

[0381] 이것으로부터 우리는 256 비트 스트링 Q^{r-1} 자체가 무작위로 독립적으로 선택되었다면, 라운드 r의 모든 잠재적인 리더의 해시된 보증서가 될 것이라는 결론을 내릴 수 있다. 사실상 모든 잠재적 리더는 잘 정의되어 있고, 또한, 그것들의 보증서도 그렇다(실제로 연산되거나 또는 연산되지 않을지라도). 또한, 라운드 r의 잠재적 리더의 세트는 라운드 r-k의 사용자 중 랜덤 하위세트이며, 정직한 잠재적 리더 i는 i의 보증서를 포함하는 자신의 메시지 $m_i^{r,1}$을 항상 적절하게 구성하고 전달한다. 따라서, 정직한 사용자의 비율이 h이므로, 악의적인 잠재적 리더가 어떤 일을 하든지(예를 들어, 자신의 보증서를 밝히거나 숨김) 최소한의 해싱된 잠재적 리더의 보증서는 모든 사람에 의해 라운드 r의 리더 ℓ^r로 필수적으로 식별되는 정직한 사용자에게 속한다. 따라서 256비트 스트링 Q^{r-1} 자체가 무작위로 독립적으로 선택되었을 때 정확하게 확률 h로 (a) 리더 ℓ^r는 정직하고 (b) 모든 정직한 단계 2 검증자 j에 대해 $\ell_j = \ell^r$이다.

[0382] 실제로, 해싱된 보증서는 무작위로 선택되고 예스이지만, 랜덤하고 독립적으로 선택되지 않는 Q^{r-1}에 의존한다. 그러나 조심스럽게 분석하면 라운드의 리더가 h에 충분히 근접한 확률 h', 즉 $h' > h^2(1 + h-h^2)$을 가지고 정직하다는 것을 보장하기에 Q^{r-1}은 충분히 조작할 수 없다는 보장이 있다. 예를 들어, h = 80%이면 h'> .7424이다.

[0383] 라운드 리더(리더 ℓ^r가 정직할 때 정확하게 수행함)를 식별한 후에, 2단계 검증자의 임무는 리더의 블록으로 생

각하는 초기 값으로 $BA^{\star}$를 실행하며 시작하는 것이다. 실제로, 필요한 통신량을 최소화하기 위해, 검증자 $j \in SV^{r,2}$는 비잔틴 프로토콜에 대한 자신의 입력 값 v'_j로서 ℓ_j(사용자 j는 리더라고 믿어진다)로부터 실제로 수신한 블록 B_j를 사용하지 않지만, 그 리더, 그 블록의 해시 즉 $v'_j = H(B_i)$은 사용한다. 따라서, BA 프로토콜의 종료시, 최종 단계의 검증자는 원하는 라운드 r 블록 B^r을 계산하지 않지만, $H(B^r)$을 계산(인증 및 전파)한다. 따라서 $H(B^r)$은 BA 프로토콜의 최종 단계의 충분히 많은 검증자에 의해 디지털 서명되기 때문에 시스템의 사용자는 $H(B^r)$가 새로운 블록의 해시임을 인식할 것이다. 그러나, 그들은 또한 적들이 무엇을 할지에 관계없이 프로토콜이 실제로 사용 가능함을 보장하는 블록 B^r 자체를 검색(또는 실행이 매우 비동기적이기 때문에 기다려야 함)해야 한다.

[0384] **비동기성(Asynchrony) and 타이밍(Timing)**

[0385] $Algorand'_1$와 $Algorand'_2$는 상당한 정도의 비동기성을 가지고 있다. 이것은 적들이 전파되는 메시지의 전달 스케줄을 잡을 때 큰 허용범위(latitude)를 가지고 있기 때문에 그렇다. 또한 라운드의 총 단계 수가 제한되어 있는지 여부에 관계없이, 실제로 취해진 단계 수만큼 변동이 있다.

[0386] 그가 B^0, ..., B^{r-1}의 인증서를 습득하자마자, 사용자 i는 Q^{r-1}을 연산하고 라운드 r에서 작업을 시작하여 그가 잠재적 리더인지 또는 라운드 r의 일부 단계 s에서 검증자인지를 체크한다.

[0387] 논의된 비동기성에 비추어 볼 때 i가 단계 s에서 행동해야 한다고 가정하면, i는 그가 행동하기 전에 충분한 정보를 확보할 수 있도록 보장하는 다양한 전략에 의존한다.

[0388] 예를 들어, 그는 이전 단계($Algorand'_1$)의 검증자로부터 적어도 주어진 수의 메시지를 수신하기 위해 대기하거나, 또는 이전 단계($Algorand'_2$)의 충분히 많은 수의 검증자의 메시지를 수신하는 것을 보장하기 위해 충분한 시간 동안 기다릴 수 있다.

[0389] **시드 Q^r와 룩백(Look-Back) 파라미터 k**

[0390] 그것들이 적에 의해 충분히 조작할 수 없더라도 수량 Q^r은 랜덤하고 독립적이어야 한다는 것을 상기하라.

[0391] 일견, Q^{r-1}을 선택하여 $H(PAY^{r-1})$와 일치시킬 수 있다. 그러나 기본 분석에 따르면 악의적인 사용자가 이 선택 메커니즘[5]을 이용할 수 있음을 보여준다. 일부 추가 노력으로 전통적인 블록 수량을 기반으로 한 수많은 다른 대안들은 악의적인 리더가 매우 빈번하게 되는 것을 보장하도록 적에 의해 용이하게 악용된다는 것을 보여준다. 우리는 대신에 적에 의해 조작될 수 없다는 것을 입증할 수 있도록 우리의 브랜드의 새로운 양 Q^r을 구체적이고 유도적으로 정의한다. 즉,

[0392] B^r이 빈 블록이 아니면, $Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r)$이고, 그렇지 않으면, $Q^r \triangleq H(Q^{r-1}, r)$이다.

[0393] (5: 우리는 라운드 r-1의 시작에 있다. 따라서 Q^{r-2} = PAY^{r-2}는 공개적으로 알려져 있고, 적들은 자신이 통제하는 잠재적인 리더가 누구인지 개인적으로 안다. 적이 10%의 사용자를 제어하고 매우 높은 확률로 악의적인 사용자 w가 라운드 r-1의 잠재적인 리더라고 가정한다. 즉, $H\left(SIG_w\left(r-2, 1, Q^{r-2}\right)\right)$은 매우 작아서 정직한 잠재적 리더가 실제로 라운드 r-1의 리더가 될 가능성은 매우 낮다. (우리가 비밀 암호 분류 메커니즘을 통해 잠재적 리더를 선택했으므로, 적들은 누가 정직한 잠재적 리더인지 알지 못한다는 것을 상기하라.) 그렇기 때문에 적은 그가 원하는 페이셋 PAY'를 선택하는 부러운 입장에 있고, 라운드 r-1의 공식적인 페이셋이 된다. 그러나 그는 더 많은 것을 할 수 있다. 그는 또한 자신의 악의적인 사용자 중 하나가 라운드 r의 리더가 될 확

률(*)이 높아서 그는 PAY^r가 무엇이 될지 자유롭게 선택할 수 있도록 할 수 있게 보장한다. (기타 등등. 적어도 장시간 동안, 즉 이러한 높은 확률의 이벤트가 실제로 발생하는 한). (*)를 보장하기 위해, 적들은 다음과 같이 행동한다. PAY'을 라운드 r-1에 대해 적들이 선호하는 페이셋이라고 한다. 그런 다음 그는 H(PAY')를 계산하고 이미 악의적인 플레이어 z에 대해, $SIG_z(r, 1, H(PAY'))$가 특히 작은 지, 즉, 매우 높은 확률로 z가 라운드 r의 리더가 되기에 충분히 작은지 여부를 확인한다. 그럴 경우, 그는 w에게 명령을 내려 자신의 후보 블록을 $B_i^{r-1} = (r-1, PAY', H(B^{r-2}))$이 되도록 선택한다. 그렇지 않으면 일부 악의적인 사용자 z에 대해(또는 일부 고정된 사용자 z에 대해서도) $H(SIG_z(PAY' \cup \{\wp\}))$도 특히 작을 때까지 두 명의 다른 악의적인 사용자 x와 y가 서로 새로운 결제 $\wp'$를 생성하는 것을 유지하도록 한다. 이러한 실험은 매우 신속하게 멈출 것이다. 그것이 그렇게 할 때, 적은 w에게 후보 블록 $B_i^{r-1} = (r-1, PAY' \cup \{\wp\}, H(B^{r-2}))$을 제안하도록 요청한다.)

[0394] 이 Q^r의 구조가 왜 작동하는지에 대한 직관은 다음과 같다. Q^{r-1}이 정말로 랜덤하고 그리고 독립적으로 선택된다고 가정하자. 그렇다면 Q^r은 어떻게 될까? ℓ^r가 정직할 때, 대답은 (대체로) yes이다. 이는 하기와 같이

[0395] $$H(SIG_{\ell^r}(\cdot), r) : \{0,1\}^{256} \longrightarrow \{0,1\}^{256}$$

[0396] 이 랜덤 함수이기 때문이다. 그러나 ℓ^r이 악의적인 경우, Q^r은 Q^{r-1}과 ℓ^r로부터 더 이상 일의적으로(univocally) 정의되지 않는다. Q^r에 대해 적어도 2개의 개별적인 값들이 있다. 하나는 계속 $Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r)$이 되고, 다른 하나는 계속해서 $H(Q^{r-1}, r)$이 된다. 먼저 제2 선택은 다소 임의적이지만 제2 선택은 절대적으로 필수적이라고 주장한다. 그 이유는 악의적인 ℓ^r이 항상 완전히 상이한 후보 블록들이 제2 단계[6]의 정직한 검증자에 의해 수신되도록 하기 때문이다. 이 경우에는, 라운드 r의 BA 프로토콜을 통해 궁극적으로 합의된 블록이 디폴트 알고리즘이 되고, 따라서 Q^{r-1}의 임의의 디지털 서명이 포함되지 않도록 보장하기는 쉽다. 그러나 시스템은 계속 진행해야 하며, 이를 위해서는 그것은 라운드 r의 리더가 필요하다. 이 리더가 자동으로 공개적으로 선택되면, 적은 사소하게 그를 손상시킨다. 이전 Q^{r-1}가 동일한 프로세스를 통해 선택되면 ℓ^r가 다시 라운드 r+1에서 리더가 될 것이다. 우리는 특별히 동일한 비밀 암호 분류 메커니즘을 사용하지만, 새로운 Q 수량 즉, $H(Q^{r-1}, r)$에 적용되는 것을 제안한다. 이러한 수량이 H의 출력이 되도록 보장함으로써 출력이 랜덤이 되도록 보장하고, H의 제2 입력으로서 r을 포함함으로써, H의 모든 다른 사용이 단일 입력 또는 적어도 3개의 입력을 가지면서, 이러한 Q^r이 독립적으로 선택되도록 "보장한다". 다시 말하면, 대안 Q^r에 대한 구체적인 선택은 중요하지 않고, 중요한 것은 ℓ^r이 Q^r에 대해 2개의 선택을 가지고, 따라서 그는 다른 악의적인 사용자를 다음 리더로 두게 될 가능성을 배가할 수 있다.

[0397] (6: 예를 들어, "제2 번째 단계의 시간이 거의 만료될 때"를 간단하게(그러나, 극단적으로) 유지하기 위해, ℓ^r가 각 사용자 i에게 다른 후보 블록 B_i를 직접 전자 메일로 보낼 수 있다. 이렇게 하면 제2 단계의 검증자가 누구든 간에 완전히 상이한 블록을 받게 된다.)

[0398] Q^r에 대한 옵션은 악의적인 ℓ^r을 통제하는 적에게 더 많을 수도 있다. 예를 들어, x, y, z가 라운드 r의 3명의 악의적인 잠재력있는 리더라고 하여,

[0399] $H(\sigma_x^{r,1}) < H(\sigma_y^{r,1}) < H(\sigma_z^{r,1})$이 되고,

[0400] $H(\sigma_z^{r,1})$이 특히 작게 되도록 하자. 즉, $H(\sigma_z^{r,1})$가 모든 정직한 잠재 리더의 해싱된 보증서보다 작을 좋은 기회가 있도록 작게 된다. 그런 다음, x에게 자신의 보증서를 숨기도록 요청함으로써, 적은 y가 라운드 r-

1의 리더가 될 수 있는 좋은 기회를 갖게 된다. 이것은 그가 Q^r에 대한 또 다른 옵션, 즉 $H(SIG_y(Q^{r-1}), r)$을 갖는다는 것을 의미한다. 유사하게, 적은 x와 y 모두에게 자신의 보증서를 숨겨 z가 라운드 r-1의 리더가 되어 Q^r에 대한 다른 옵션, 즉 $H(SIG_Z(Q^{r-1}), r)$를 획득하도록 요청할 수 있다.

[0401] 그러나 물론 이러한 옵션과 다른 옵션은 모두 실패할 0이 아닌 기회가 있다. 왜냐하면 적들이 정직한 잠재 사용자의 디지털 서명 해시를 예측할 수 없기 때문이다.

[0402] 신중한 마르코프 체인과 같은 분석은 적이 라운드 r-1에서 어떤 옵션을 선택하든지 시스템에 새로운 사용자를 투입할 수 없는 한, 그가 정직한 사용자가 라운드 r+40의 리더가 되는 확률을 h 아래로 많이 낮출 수는 없다는 것을 보여준다. 이것이 우리가 라운드 r의 잠재적 리더가 이미 라운드 r-k에 있는 사용자가 될 것을 요구하는 이유이다. 그것은 라운드 r-k에서 적들이 정직한 사용자가 라운드 r의 리더가 될 확률을 많이 바꿀 수 없도록 보장하는 방법이다. 실제로 사용자가 라운드 r을 통해 라운드 r-k에서 시스템에 추가할 수 있는 사용자가 무엇이든 간에 그들은 라운드 r의 잠재적인 리더(그리고 더 한층의 리더)가 될 자격이 없다. 따라서 룩백 파라미터 k는 궁극적으로 보안 파라미터이다. (7절에서 보게 되겠지만 그것은 또한 일종의 "편의 파라미터(convenience parameter)"일 수도 있다.)

[0403] **임시 키**

[0405] *우리의 프로토콜의 실행은 무시할 수 있는 확률을 제외하고 포크를 생성할 수 없지만, 적은 r번째 블록에서 합법적 블록 r이 생성된 후에 포크를 생성할 수 있다.

[0406] 대략 B^r가 생성되면, 적들은 라운드 r의 각 단계의 검증자가 누구인지 배웠다. 따라서, 그는 따라서 그들 모두를 손상시키고 새로운 블록 $\widetilde{B^r}$을 증명하도록 강요할 수 있다. 이 가짜 블록은 합법적인 블록 이후에만 전파될 수 있으므로 주의를 기울인 사용자는 속지는 않을 것이다.[7] 그럼에도 불구하고 $\widetilde{B^r}$은 구문상 정확하며, 우리는 제조되는 것을 방지하기를 원한다.

[0407] (7: 주요 TV 네트워크의 뉴스 앵커를 손상시키고, 클린턴 장관이 지난 대통령 선거에서 승리한 것을 보여주는 오늘 뉴스 릴을 제작 및 방송하는 것을 고려해보라. 우리 중 대부분은 그것을 사기라고 인식할 것이다. 그러나 혼수상태에서 벗어난 누군가는 속을 수 있다.)

[0408] 우리는 새로운 규칙을 통해 그렇게 한다. 본질적으로, 라운드 r의 단계 s의 검증자 세트 $SV^{r,s}$의 멤버는 자신의 메시지에 디지털 서명하기 위해 임시 공개키 $pk_i^{r,s}$를 사용한다. 이 키들은 일회용으로 사용되며 해당 비밀 키 $sk_i^{r,s}$는 한번 사용되면 폐기된다. 이렇게 하면 검증자가 나중에 손상된 경우, 적이 원래 서명하지 않은 것에 다른 사람의 서명을 강요할 수 없다.

[0409] 당연히, 우리는 적이 새로운 키 $\widetilde{p_i^{r,s}}$를 계산하는 것이 불가능하다는 것을 보증해야 하고 정직한 사용자에게 그것이 단계 s에서 사용하는 검증자 $i \in SV^{r,s}$의 올바른 임시 키임을 확신시켜야 한다.

[0410] **4.2 표기법, 개념 및 파라미터에 대한 일반적인 요약**

[0411] **표기법**

[0413] *· r> 0: 현재의 라운드 수이다.

[0414] · s≥ 1 : 라운드 r의 현재 단계 번호.

[0415] · B^r: 라운드 r에서 생성된 블록.

[0416] · PK^r: 라운드 r-1이 끝날 때까지 및 라운드 r의 시작 부분에서의 공개키 세트.

[0417] · S^r: 라운드 r-1 끝과 라운드 r 시작시 시스템 상태.[8]

[0418] (8: 동기화되지 않은 시스템에서는 "라운드 r-1의 끝" 및 "라운드 r의 시작"의 개념을 신중히 정의해야 한다. 수학적으로, PK^r과 S^r는 초기 상태 S^0와 블록 B1, ..., B^{r-1}로부터 연산된다.)

[0419] · PAY^r : B^r에 포함된 페이셋.

[0420] · ℓ^r: 라운드 r 리더. ℓ^r은 라운드 r의 페이셋 PAY^r을 선택한다(그리고, 다음 Q^r을 판정한다).

[0421] · Q^r: 라운드 r의 시드, 라운드 r의 끝에서 생성되고 라운드 r+1에 대한 검증자를 선택하는 데 사용되는 수량(즉, 이진 스트링). Q^r은 블록의 페이셋에 독립적이고 ℓ^r에 의해 조작될 수 없다.

[0422] · $SV^{r,s}$: 라운드 r의 단계 s에서 선택된 검증자의 세트.

[0423] · SV^r: 라운드 r에 대해 선택된 검증자 세트, $SV^r = \cup_{s\geq 1} SV^{r,s}$

[0424] · $MSV^{r,s}$ 및 $HSV^{r,s}$: 각각 $SV^{r,s}$에서의 악의적인 검증자 세트 및 정직한 검증자 세트. $MSV^{r,s} \cup HSV^{r,s} = SV^{r,s}$ 및 $MSV^{r,s} \cap HSV^{r,s} = \emptyset$.

[0425] · $n_1 \in \mathbb{Z}^+$ 및 $n \in \mathbb{Z}^+$ 각각: s>1에 대해, 각각의 $SV^{r,1}$에 있는 잠재적 리더의 예상 수와 각각의 $SV^{r,s}$의 예상 검증자 수.

[0426] $SV^{r,1}$에서 적어도 하나의 정직한 멤버를 필요로하지만, s>1에 대해 각각의 $SV^{r,s}$에서 적어도 과반수의 정직한 멤버를 필요로 하기 때문에 n_1 <<n임을 유의하라.

[0427] · h∈(0, 1): 2/3보다 큰 상수. h는 시스템의 정직 비율이다. 즉, 각 PK^r에서 사용된 가정에 따라 정직한 사용자 또는 정직한 돈의 비율은 적어도 h이다.

[0428] · H: 랜덤 오라클로서 모델링된 암호화 해시 함수이다.

[0429] · ⊥: H의 출력과 동일한 길이의 특수 스트링이다.

[0430] · F∈(0, 1): 허용된 오류 확률을 지정하는 파라미터. 확률≤F는 "무시할 수 있는" 것으로 간주되며, 확률≥1-F는 "압도적인" 것으로 간주된다.

[0431] · p_h∈(0, 1): 라운드 r의 리더 ℓ^r가 정직할 확률. 이상적으로 p_h=h. 적의 존재와 함께, p_h의 값은 분석에서 결정될 것이다.

[0432] · $k \in \mathbb{Z}^+$: 룩백 파라미터. 즉, 라운드 r-k는 라운드 r에 대한 검증자가 즉, $SV^r \subseteq PK^{r-k}$로부터 선택된다.[9]

[0433] (9: 엄밀히 말하면 "r-k"는 "max{0, r-k}"이어야 한다.)

[0434] · p_1∈(0, 1): 라운드 r의 제1 단계에서, 라운드 r-k의 사용자는 확률 $p_1 \triangleq \frac{n_1}{|PK^{r-k}|}$을 갖는 $SV^{r,1}$에 있도록 선택된다.

[0435] · p∈(0, 1): 라운드 r의 각 단계 s> 1에 대해, 라운드 r-k의 사용자는 확률 $p \triangleq \frac{n}{|PK^{r-k}|}$을 갖는 $SV^{r,s}$에

있도록 선택된다.

[0436] · $CERT^r$: B^r에 대한 보증서. 그것은 라운드 r에서의 적절한 검증자로부터의 $H(B^r)$의 t_H 서명 세트이다.

[0437] · $\overline{B^r} \triangleq (B^r, CERT^r)$은 입증된 블록이다.

[0438] 사용자 i는 그가 입증된 블록의 두 부분을 소유하고 성공적으로 검증한다면 B^r을 안다. 다른 사용자가 본 $CERT^r$은 다를 수 있다는 것에 유의하라.

[0439] · τ_i^r: 사용자 i가 B^r을 아는(현지) 시간. Algorand 프로토콜에서 각 사용자는 자신의 시계를 가지고 있다. 서로 다른 사용자의 시계를 동기화할 필요는 없지만 동일한 속도를 가져야 한다. 분석 목적으로만, 참조 클록을 고려하여 그에 대해 플레이어의 관련 시간을 측정한다.

[0440] · $\alpha_i^{r,s}$ 및 $\beta_i^{r,s}$: 각각 사용자 i가 라운드 r의 단계 s의 실행을 시작하고 종료하는 (현지)시간.

[0441] · Λ와 λ: 근본적으로 Algorand 프로토콜의 단계 1을 실행하는 데 필요한 시간 및 다른 단계에 필요한 시간에 대한 상한.

[0442] 파라미터 Λ는 단일 1MB 블록을 전파하는 시간의 상한이다.

[0443] 파라미터 λ는 단계 s> 1에서 검증자당 하나의 작은 메시지를 전달하는 시간의 상한이다.

[0444] 우리는 Λ≤4λ라고 가정한다.

[0445] **관념**

[0446] · 검증자 선택.

[0447] 각 라운드 r과 단계 s>1에 대해, $SV^{r,s} \triangleq \{i \in PK^{r-k} : H(SIG_i(r,s,Q^{r-1})) \leq p\}$이다. 각 사용자 i∈$PK^{r-k}$는 개인적으로 자신의 장기 키를 이용하여 자신의 서명을 연산하고, i∈$SV^{r,s}$인지 아닌지를 결정한다. 예를 들어 i∈$SV^{r,s}$라면 $SIG_i(r, s, Q^{r-1})$는 i의 (r, s)-보증서이고, 축약해서 $\sigma_i^{r,s}$로 표시한다.

[0448] 라운드 r의 제1 단계에 있어서, $SV^{r,1}$ 및 $\sigma_i^{r,1}$은 p가 p_1로 대체되며 유사하게 정의된다. $SV^{r,1}$에서의 검증자는 잠재적 리더이다.

[0449] · 리더 선택.

[0450] 사용자 i∈$SV^{r,1}$이 라운드 r의 리더라면, 모든 잠재적 리더 j∈$SV^{r,s}$에 대해 $H(\sigma_i^{r,1}) \leq H(\sigma_j^{r,1})$인 경우 ℓ^r로 표기된다. 두 플레이어의 보증서의 해시가 비교될 때마다, 연계가 없는 경우 프로토콜은 항상 잠재적 리더(의 장기 공개키)에 따라 사전학적으로 파기한다.

[0451] 정의에 따르면, 플레이어 ℓ^r의 보증서의 해시 값은 PK^{r-k}의 모든 사용자 중에서 가장 작다. 잠재적인 리더는 다른 잠재적인 리더의 보증서를 보지 않고 자신이 리더인지 여부를 개인적으로 결정할 수 없다.

[0452] 해시 값은 랜덤하게 균일하므로 $SV^{r,1}$이 비어 있지 않으면 ℓ^r은 항상 존재하며 적어도 h일 확률로 정직하다. 파라미터 n_1은 각 $SV^{r,1}$이 압도적인 확률로 비어 있지 않음을 보장하기에 충분히 크다.

[0453] · 블록 구조.

[0454] 비어 있지 않은 블록은 $B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$의 형태이고, 빈 블록은 $B_\epsilon^r = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$의 형태이다.

[0455] 이 라운드에서 결제가 발생하지 않거나 리더가 악의적인 경우 비어 있지 않은 블록에 빈 페이셋 PAY^r이 계속 포함될 수 있다는 것에 유의하라. 그러나 비어 있지 않은 블록은 ℓ^r의 신원, 그의 보증서 $\sigma_{\ell^r}^{r,1}$ 및 $SIG_{\ell^r}(Q^{r-1})$이 모두 시의 적절하게 밝혀졌다는 것을 의미한다. 프로토콜은 리더가 정직하다면 블록이 압도적인 확률로 비어 있지 않음을 보장한다.

[0456] · 시드 Q^r.

[0457] B^r이 비어 있지 않으면 $Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r)$이고, 그렇지 않으면 $Q^r \triangleq H(Q^{r-1}, r)$이다.

[0458] **파라미터**

[0459] · 다양한 파라미터 간의 관계.

[0460] - 라운드 r의 검증자 및 잠재적 리더는 PK^{r-k}의 사용자로부터 선택되고, 여기서 k는 F 보다 나은 확률로 적이 라운드 r-k-1에서 Q^{r-1}을 다시 예측할 수 없도록 선택된다. 그렇지 않으면, 라운드 r-k를 위해 악의적인 사용자를 소개할 수 있을 것이고, 이 모두는 라운드 r에서 잠재적 리더/검증자가 되고, 그에 의해 원해질 때 일부 단계 s에 대해 $SV^{r,s}$에서 악의적인 리더 또는 악의적인 대다수를 가지는 데에 성공했다.

[0461] - 각 라운드 r의 제1 단계에서 압도적인 확률로 $SV^{r,1} \neq \emptyset$이 되도록 n_1을 선택한다.

[0462] · 중요한 파라미터의 예시적 선택.

[0463] - H의 출력은 256비트 길이이다.

[0464] - h=80 %, n_1=35.

[0465] - Λ=1 분, λ=15초.

[0466] · 프로토콜의 초기화.

[0467] 프로토콜은 시간 0에서 r = 0으로 시작한다. "B^{-1}" 또는 "$CERT^{-1}$"이 없으므로, 구문적으로 B^{-1}은 Q^{-1}을 지정하는 자신의 제3 구성 요소가 있는 공개 파라미터이고, 모든 사용자는 시간 0에서 B^{-1}을 알고 있다.

[0468] ## 5. $Algorand'_1$

[0469] 이 섹션에서는 다음과 같은 가정하에 작동하는 $Algorand'$ 버전을 구성한다.

[0470] 대다수 정직한 사용자 가정: 각 PK^r의 사용자 중 2/3 이상이 정직하다.

[0471] 8절에서는 위의 가정을 원하는 대다수 정직한 사용자 가정으로 대체하는 방법을 보여준다.

[0472] ### 5.1 추가 표기법과 파라미터

[0473] **표기법**

[0474] . $m \in \mathbb{Z}^+$: 이진 BA 프로토콜의 최대 단계 수, 3의 배수.

[0475] . $L^r \leq m/3$: 각 시도가 확률 $\frac{p_h}{2}$로 1이고 최대 m/3의 시도가 있을 때 1을 보기 위해 필요한 베르누이(Bernoulli) 시도 횟수를 나타내는 확률 변수. 모든 시도가 실패하면 $L^r \triangleq m/3$이다. L^r은 블록 B^r을 생성하는 데 필요한 시간을 상한으로하는 데 사용된다.

[0476] · $t_H = \frac{2n}{3}+1$: 프로토콜의 종료 조건에서 필요한 서명 수.

[0477] · $CERT^r$: B^r에 대한 공증. 그것은 라운드 r에서 적절한 검증자로부터의 $H(B^r)$의 t_H 서명 세트이다.

[0478] **파라미터**

[0479] · 다양한 파라미터 간의 관계.

[0480] - 라운드 r의 각 단계 s> 1에 대해, n이 선택되어 압도적인 확률로,

[0481] $|HSV^{r,s}| > 2|MSV^{r,s}|$ 및 $|HSV^{r,s}| + 4|MSV^{r,s}| < 2n$이다.

[0482] h의 값이 1에 가까울수록 n은 더 작을 필요가 있다. 특히, 우리는 압도적인 확률로 원하는 조건을 유지하는 것을 보장하기 위해 체르노프(Chernoff) 범위(그의 변형)를 사용한다.

[0483] - m은 압도적인 확률로 $L^r < m/3$이 되도록 선택된다.

[0484] · 중요한 파라미터의 예제선택.

[0485] - $F = 10^{-12}$.

[0486] - n≈1500, k=40 및 m=180이다.

[0487] ### 5.2 $Algorand'_1$에서 임시 키 구현하기

[0488] 이미 언급했듯이, 검증자 i∈$SV^{r,s}$는 그가 사용 후 즉시 파기하는 임시 비밀 키 $sk_i^{r,s}$를 이용하여 임시 공개키 $pk_i^{r,s}$에 대해 라운드 r의 단계 s의 자신의 메시지 $m_i^{r,s}$을 디지털 서명한다. 따라서 우리는 $pk_i^{r,s}$가 실제로 사용자 i의 서명 $m_i^{r,s}$을 증명하기 위해 사용하는 키인지를 모든 사용자가 증명할 수 있다고 보장하는 효율적인 방법이 필요하다. 우리는 신원 확인에 기반한 서명 체계의 새로운 사용(우리가 아는 한)에 의해 그렇게 한다.

[0489] 이러한 방식에서, 상위 레벨에서, 중앙 권한 A는 공개 마스터 키인 PMK 및 대응하는 비밀 마스터 키인 SMK를 생성한다. 플레이어 U의 신원 U가 주어지면, A는 공개키 U에 대한 비밀 서명키 sk_U를 SMK를 통해 계산하고, 개인적으로 sk_U를 U에게 준다(사실, 신원 기반 디지털 서명 체계에서, 사용자 U의 공개키는 U 자체이다!) 이렇게 하면 A가 디지털 서명을 생성할 수 있게 하려는 사용자의 비밀 키를 계산한 후에 SMK를 파기하고 계산된 비밀 키를 보관하지 않으면, U는 공개키 U에 대해 메시지에 디지털 서명할 수 있는 유일한 사람이 된다. 따라서 "U의 이름"을 아는 임의의 사람은 자동적으로 U의 공개키를 알고, 따라서 U의 서명을 확인할 수 있다(공개 마스터 키 PMK를 또한 사용할 수 있음).

[0490] 우리의 응용에서, 권위자 A는 사용자 i이고 가능한 모든 사용자 U의 세트는 즉 S={i}x{r', ..., r'+10^6}x{1, ..., m+3}에서의 라운드-단계 쌍(r, s)과 일치한다 (여기서, r'는 주어진 라운드이고, m+3은 라운드 내에서 발생할 수 있는 단계 수에 대한 상한이다). 이 방법은 $pk_i^{r,s} \triangleq (i,r,s)$이 되어, i의 서명 $SIG^{r,s}_{pk_i^{r,s}}(m_i^{r,s})$을 보는 모든 사람이 압도적인 확률로 r'에 후속하는 제1 백만 라운드 r에 대해 그것을 즉시 증명할 수 있도록 한다.

[0491] 즉, i는 먼저 PMK와 SMK를 생성한다. 그런 다음 그는 PMK가 모든 라운드 r∈[r', r'+10^6]에 대한 i의 마스터 키이고, 각각의 트리플 (i, r, s)∈S에 대해 비밀 키 $sk_i^{r,s}$를 개인적으로 생성하여 저장하도록 SMK를 이용한다고 공개한다. 이것이 종료하면, 그는 SMK를 파괴한다. 그가 SV^{rs}의 일부가 아니라고 결정하면, i는 $sk_i^{r,s}$를 혼자 남겨 둘 수도 있다(프로토콜은 라운드 r의 단계 s에서 어떤 메시지도 인증하지 않기 때문에). 그렇지 않으면 i

는 먼저 $sk_i^{r,s}$ 를 사용하여 메시지 $m_i^{r,s}$ 에 디지털 서명한 다음 $sk_i^{r,s}$ 를 삭제한다.

[0492] 처음 시스템에 들어갈 때, i는 자신의 제1 마스터 공개키를 공개할 수 있다는 것에 유의하라. 즉, 시스템에 i를 가져 오는 동일한 결제 $\wp$(라운드 r', 또는 r'에 근접한 라운드에서)는 i의 요청에 따라 예를 들면 형태 (PMK, [r', r'+10^6]의 쌍을 포함함으로써 임의의 라운드 r∈[r', r'+10^6]에 대해 i의 공개 마스터 키가 PMK라고 또한 지정할 수 있다.

[0493] 또한 m+3은 라운드의 최대 단계 수이기 때문에 라운드가 1분이 소요된다고 가정하면 생성된 임시 키의 은닉이 거의 2년 동안 i에 대해 지속된다. 동시에, 이러한 임시 비밀 키는 i가 제작하기에는 너무 길어서 취하지 않는다. 32B 키가 있는 타원 곡선 기반 시스템을 사용하면 각 비밀 키가 수 마이크로초 안에 계산된다. 따라서, m+3=180이라면, 모든 180M 비밀 키는 1시간 이내에 계산될 수 있다.

[0494] 현재 라운드가 다음 백만의 라운드를 처리하기 위해 r'+10^6에 가까워지면, i는 새로운(PMK', SMK') 쌍을 생성하고, 예를 들어 SIG_i(PMK', [r'+10^6+1, r'+2 · 10^6+1])을 가짐으로써 임시 키의 그의 다음 은닉이 무엇인지가 별도의 "거래" 또는 결제의 일부인 일부 추가 정보로서 새로운 블록에 들어간다고 알린다. 그렇게 함으로써, i는 다음 모든 사람들에게 다음의 백만 라운드에서 i의 임시 서명을 증명하기 위해 그들이 PMK'를 사용해야 한다고 알린다. 등등.

[0495] (이 기본 접근법을 따르면, 신원 기반 서명을 사용하지 않고 임시 키를 구현하는 다른 방법이 확실히 가능하다는 것에 유의하라. 예를 들어, 머클 트리[10]를 통해)

[0496] (10: 이 방법에서, i는 각 라운드-단계 쌍(r, s), 즉 {r', ..., r'+10^6} x {1, ..., m+3}에 대해 공개-비밀 키 쌍 $(pk_i^{r,s}, sk_i^{r,s})$을 생성한다. 그런 다음 그는 이러한 공개키를 정식 방식으로 지시하고, 머클 트리의 j번째 리프에 j번째 공개키를 저장하고, 루트 값 R을 연산하고, 이를 공개한다. 그가 키 $pk_i^{r,s}$에 대한 메시지에 서명하기를 원할 때, i는 실제 서명뿐만 아니라 Ri에 대해 $pk_i^{r,s}$ 를 위한 인증 경로도 제공한다. 이 인증 경로는 또한 $pk_i^{r,s}$ 가 j번째 리프에 저장되었음을 증명한다는 것에 유의하라. 이 아이디어를 형성하면 나머지 상세를 쉽게 채울 수 있다.)

[0497] 임시 키를 구현하기 위한 다른 방법은 예를 들어 머클 트리를 통해 확실히 가능하다.

[0498] **5.3 알고리즘 단계와 BA⋆의 단계 매칭**

[0499] 우리가 말했듯이, $Algorand'_1$의 라운드에는 최대 m+3 단계가 있다.

[0500] 단계 1.

[0501] 이 단계에서 각 잠재적 리더 i는 후보 블록 B_i^r 를 자신의 보증서 $\sigma_i^{r,1}$ 과 함께 계산하여 전파한다.

[0502] 이 보증서가 i를 명시적으로 식별함을 상기하자. 이것은 $\sigma_i^{r,1} \triangleq SIG_i(r, 1, Q^{r-1})$이기 때문에 그렇게 된다.

[0503] 잠재적 검증자 i는 또한 자신의 메시지의 일부로서 $H(B_i^r)$의 자신의 적절한 디지털 서명을 전달한다. 결제나 보증서를 다루지 않으면서, i의 이 서명은 자신의 임시 공개키 $pk_i^{r,1}$과 관련이 있다. 즉, 그는 $sig_{pk_i^{r,1}}(H(B_i^r))$ 를 전파한다.

[0504] B_i^r와 $sig_{pk_i^{r,1}}(H(B_i^r))$를 전파하기보다는 우리의 협약을 감안할 때, 그는 $SIG_{pk_i^{r,1}}(H(B_i^r))$를 전파할 수 있었다. 그러나 우리의 분석에서 우리는 $sig_{pk_i^{r,1}}(H(B_i^r))$에 명시적으로 접근할 필요가 있다.

[0505] 단계 2.

[0506] 이 단계에서 각 검증자 i는 ℓ_i^r가 자신의 해싱된 보증서가 가장 작은 잠재적 리더가 되고 B_i^r가 ℓ_i^r에 의해 제안된 블록이 되도록 설정한다. 효율성을 위해서, 우리는 B^r에 직접적이 아닌 $H(B^r)$에 동의하기를 원하기 때문에, i는 최초값 $v_i' = H(B_i^r)$를 가지고 $BA^\star$의 제1 단계에서 그가 전파했을 메시지를 전파한다. 즉, 그는 물론 거기에 임시로 서명한 후에 v_i'를 전파한다. (즉, 우측 임시 공개키(이 경우 $pk_i^{r,2}$인)에 대해 그것에 서명한 후). 물론 i는 자신의 보증서도 전송한다.

[0507] $BA^\star$의 제1 단계는 단계적 합의 프로토콜 GC의 제1 단계로 구성되기 때문에 $Algorand'$의 제2 단계는 GC의 제1 단계에 해당한다.

[0508] 단계 3.

[0509] 이 단계에서 각 검증자 $i \in SV^{r,s}$는 $BA^\star$의 제2 단계를 실행한다. 즉, 그는 자신이 GC의 제2 단계에서 보낸 동일한 메시지를 보낸다. 다시 말하지만, i의 메시지는 임시 서명이 되어 있고 i의 보증서가 동반된다. (이제부터 우리는 검증자가 자신의 메시지에 임시 서명하고 또한 자신의 보증서를 전달한다는 말을 생략할 것이다.

[0510] 단계 4.

[0511] 이 단계에서 모든 검증자 $i \in SV^{r,s}$는 GC의 출력 (v_i, g_i)을 계산하고, $BA^\star$의 제3 단계, 즉, $BBA^\star$의 제1 단계에서 보낸 동일한 메시지를 g_i=2인 경우 0이고 그렇지 않은경우 1인 최초 비트를 가지고 임시 서명하고 전송한다.

[0512] 단계 s=5, ..., m+2이다. 그러한 단계가 도달하면, 단계는 $BA^\star$의 단계 s-1에 대응하고, 따라서 $BBA^\star$의 단계 s-3에 대응한다.

[0513] 우리의 전파 모델은 충분히 비동기적이므로, 이러한 단계 s의 중간에, 블록 B^r이 이미 선택되었다는 것을 그에게 증명하는 정보에 의해 검증자 $i \in SV^{r,s}$에 도달하는 가능성을 고려해야 한다. 이 경우에, i는 $Algorand'$의 라운드 r의 자신의 실행을 멈추고 자신의 라운드(r+1) 명령을 실행하기 시작한다.

[0514] 따라서, $BBA^\star$의 단계 s-3에 대응하는 명령에 부가하여, 검증자 $i \in SV^{r,s}$의 명령은 이전 단계 s'에서 $BBA^\star$의 실행이 중지되었는지 여부를 검사하는 것을 포함한다. $BBA^\star$만 멈출 수 있는 것은 0에 고정된 코인(Coin-Fixed-to-0) 단계 또는 1에 고정된 코인 단계에서이고, 명령은 다음 중 어느 것인지를 구별한다

[0515] A(종료 조건): $s' - 2 \equiv 0 \ mod \ 3$

[0516] B(종료 조건): $s' - 2 \equiv 1 \ mod \ 3$

[0517] 사실, A의 경우, 블록 B^r이 비어 있지 않으므로 적절한 인증서 $CERT^r$와 함께 i가 B^r을 적절하게 재구성하는 것을 보장할 추가 지침이 필요하다. B의 경우, 블록 B^r이 비어 있으므로, i가 $B^r = B_\varepsilon^r = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$으로 설정하고, $CERT^r$를 계산하도록 지시받는다.

[0518] 단계 s를 실행하는 동안 B^r 블록이 이미 생성되었다는 증거가 i에게 보이지 않으면, 그는 $BBA^\star$의 단계 s-3에서 보낸 동일한 메시지를 보낸다.

[0519] 단계 m+3.

[0520] 단계 m+3 동안. $i \in SV^{r,m+3}$이 블록 B^r이 이미 이전 단계 s'에서 생성되었다고 본다면, 그는 앞서 설명된 바와 같이 진행한다.

[0521] 그렇지 않으면 $BBA^{\star}$의 m 단계에서 그가 전송한 동일한 메시지를 보내지 않고, i는 자신의 소유 정보를 기반으로 B^r 및 자신의 대응하는 인증서 $CERT^r$을 계산하도록 지시받는다.

[0522] 사실, 우리는 라운드의 총 단계 수를 m+3만큼 상한으로 한 것을 상기하라.

[0523] **5.4 실제 프로토콜**

[0524] 라운드 r의 각 단계 s에서, 검증자 $i \in SV^{r,s}$는 s=1의 경우에 자신의 장기 공유 비밀 키 쌍을 사용하여 자신의 보증서 $\sigma_i^{r,s} \triangleq SIG_i(r,s,Q^{r-1})$ 뿐만 아니라 $SIG_i(Q^{r-1})$를 산출하도록 한다. 검증자 i는 자신의 임시 비밀 키 $sk_i^{r,s}$를 사용하여 자신의(r, s)-메시지 $m_i^{r,s}$에 서명한다. 간단히 하기 위해, r과 s가 명확하면, 우리는 라운드 r의 단계 s에서 값 x의 i의 적절한 임시 서명을 나타내기 위해 $sig_{pk_i^{r,s}}(x)$가 아닌 $esig_i(x)$를 기록하고 (i, x, $esig_i(x)$)를 나타내기 위해 $SIG_{pk_i^{r,s}}(x)$ 대신 $ESIG_i(x)$를 기록한다.

[0525] 1 단계: 블록 제안

[0526] 모든 사용자 $i \in PK^{r-k}$에 대한 지침은: 사용자 i는 B^{r-1}을 알게 되는 즉시 라운드 r의 자신의 단계 1을 시작한다.

[0527] · 사용자 i는 B^{r-1}의 제3 구성 요소로부터 Q^{r-1}을 계산하고 $i \in SV^{r,1}$이 있는지 여부를 확인한다.

[0528] · 만약 $i \notin SV^{r,1}$이라면, i는 즉시 자신의 제1 단계의 실행을 중단한다.

[0529] · 만약 $i \in SV^{r,1}$이라면, 즉, i가 잠재적 리더라면, 그는 지금까지 그에게 전파된 라운드 r 결제를 수집하고 그것들로부터 최대 페이셋 PAY_i^r을 계산한다. 다음으로 그는 "후보 블록" $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$을 연산한다. 마지막으로, 그는 메세지 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$을 연산하고, 자신의 임시 비밀키 $sk_i^{r,1}$를 파괴하고, 그런 다음 $m_i^{r,1}$을 전파한다.

[0530] **주의.**

[0531] 실제로, 1 단계의 전체 실행을 단축하려면, (r, 1)-메시지가 선택적으로 전파되는 것이 중요하다. 즉, 시스템의 모든 사용자 i에 대해, 그가 수신하고 성공적으로 검증한[11] 제1 (r, 1)-메시지에 대해, 플레이어 i가 평소와 같이 그것을 전파한다. 플레이어 i가 수신하여 성공적으로 검증한 다른 모든(r, 1) 메시지에 대해, 자신이 포함하는 보증서의 해시 값이 자신이 지금까지 수신하고 성공적으로 검증한 모든(r, 1)-메시지에 포함된 보증서의 해시 값 중 가장 작은 경우에만 그는 그것을 전파한다. 또한 게오르기오스 블라코스(Georgios Vlachos)가 제안한 것처럼, 잠재적 리더 i가 각각 자신의 보증서 $\sigma_i^{r,1}$을 별도로 전파하는 것이 유용하다. 이들 작은 메시지는 블록보다 빠르게 이동하고, 포함된 보증서가 작은 해시 값을 가지는 $m_i^{r,1}$의 시의 적절한 전파를 보장하는 반면, 더 큰 해시 값을 가진 이것들을 빠르게 사라지도록 한다.

[0532] (11: 즉, i가 포함된 페이셋이 자신의 제안자에게 최대인지 여부는 확인하지 않지만, 모든 서명이 정확하고 블록과 해당 해시가 모두 유효하다.)

[0533] 2단계 : 단계적 합의 프로토콜 GC의 제1 단계

[0534] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 2를 시작한다.

[0535] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 $i \in SV^{r,2}$가 있는지 여부를 확인한다.

[0536] · 만약 $i \notin SV^{r,2}$라면, i는 바로 자신의 2단계의 실행을 중단한다.

[0537] · $i \in SV^{r,2}$인 경우 시간 $t_2 \triangleq \lambda + \Lambda$의 만큼 대기한 후에, i는 다음과 같이 동작한다.

[0538] 1. 그는 그가 지금까지 수신한 성공적으로 검증한 (r, 1)-메시지의 일부인 모든 보증서 $\sigma_j^{r,1}$에 대해 $H(\sigma_\ell^{r,1}) \leq H(\sigma_j^{r,1})$이 되도록 사용자 ℓ을 찾는다.[12]

[0539] (12: 기본적으로, 사용자 i는 라운드 r의 리더가 사용자 ℓ임을 개인적으로 결정한다.)

[0540] 2. 만약 그가 유효한 메시지 $m_\ell^{r,1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r,1})$를 ℓ로부터 받았다면,[13] i는 $v_i' \triangleq H(B_\ell^r)$를 설정하고, 그렇지 않으면 i는 $v_i' \triangleq \perp$를 설정한다.

[0541] (13: 다시, 플레이어 ℓ의 서명과 해시는 모두 성공적으로 검증되고, i가 ℓ에 대해 PAY_ℓ^r이 최대인지를 확인하지 않더라도, B_ℓ^r에서의 PAY_ℓ^r이 라운드 r에 대해 유효한 페이셋이다.).

[0542] 3. i는 메시지 $m_i^{r,2} \triangleq (ESIG_i(v_i'), \sigma_i^{r,2})$를 연산하고,[14] 그의 임시 비밀 키 $sk_i^{r,2}$를 파괴하고, 그런 다음 $m_i^{r,2}$를 전파한다.

[0543] (14: 메시지 $m_i^{r,2}$는 플레이어 i가 v_i'를 다음 블록의 해시라고 간주하거나, 다음 블록을 비어 있다고 간주한다고 시그널링한다.)

[0544] 3단계: GC의 제2 단계

[0545] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 3을 시작한다.

[0546] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 $i \in SV^{r,3}$이 있는지 여부를 확인한다.

[0547] · 만약 $i \notin SV^{r,3}$이라면, i는 곧바로 자신의 3단계의 실행을 중단한다.

[0548] · $i \in SV^{r,3}$인 경우 시간 $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$의 만큼 대기한 후에, i는 다음과 같이 동작한다.

[0549] 1. 그가 수신한 모든 유효한 메시지 $m_j^{r,2}$ 중에서 2/3 이상이 어떠한 모순도 없이[15] $(ESIG_j(v'), \sigma_j^{r,2})$의 형태를 가지도록 값 $v' \neq \perp$이 존재하면, 그는 메시지 $m_i^{r,3} \triangleq (ESIG_i(v'), \sigma_i^{r,3})$을 계산한다. 그렇지 않으면, 그는 $m_i^{r,3} \triangleq (ESIG_i(\perp), \sigma_i^{r,3})$을 계산한다.

[0550] (15: 즉, 그는 플레이어 j로부터 $ESIG_j(v')$와 상이한 $ESIG_j(v'')$를 각각 포함하는 두 개의 유효한 메시지를 수신하지 못했다. 여기 저기에서, 추후에 정의되는 종료 조건을 제외하고는, 정직한 플레이어가 주어진 형태의 메시지를 원할 때마다, 서로 상충되는 메시지는 카운팅되지 않거나 유효한 것으로 간주되지 않는다.)

[0551] 2. i는 자신의 임시 비밀 키 $sk_i^{r,3}$을 파기하고 $m_i^{r,3}$을 전파한다.

[0552] 4단계 : GC 출력 및 BBA*의 제1 단계

[0553] 모든 사용자 i∈PK^{r-k}에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 4를 시작한다.

[0554] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 i∈SV^{r,4}가 있는지 여부를 확인한다.

[0555] · 만약 $i \notin SV^{r,4}$라면, i는 바로 자신의 4단계의 실행을 중단한다.

[0556] · i∈SV^{r,4}인 경우 시간 $t_4 \triangleq t_3 + 2\lambda = 5\lambda + \Lambda$의 만큼 대기한 후에, i는 다음과 같이 동작한다.

[0557] 1. 그는 GC의 출력인 v^i 및 g_i를 다음과 같이 계산한다.

[0558] (a) 그가 수신한 모든 유효한 메시지 $m_j^{r,3}$ 중에서 2/3 이상이 형태 $(ESIG_j(v'), \sigma_j^{r,3})$를 가지도록 값 $v' \neq \perp$이 존재하면, 그는 $v_i \triangleq v'$ 및 $g_i \triangleq 2$를 설정한다.

[0559] (b) 그렇지 않으면, 그가 수신한 모든 유효한 메시지 $m_j^{r,3}$ 중에서 1/3 이상이 형태 $(ESIG_j(v'), \sigma_j^{r,3})$를 가지도록 값 $v' \neq \perp$이 존재하면, 그는 $v_i \triangleq v'$ 및 $g_i \triangleq 1$을 설정한다.[16]

[0560] (16: (b)의 경우에 v'가 존재한다면, 유일해야 한다는 것이 증명될 수 있다.)

[0561] (c) 그렇지 않으면 그는 $v_i \triangleq H(B_\epsilon^r)$ 및 $g_i \triangleq 0$으로 설정한다.

[0562] 2. 그는 다음과 같이 BBA*의 입력인 b_i를 계산한다.

[0563] g_i = 2이면 $b_i \triangleq 0$, 그렇지 않으면 $b_i \triangleq 1$.

[0564] 3. 그는 메시지 $m_i^{r,4} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$을 계산하고 자신의 임시 비밀 키 $sk_i^{r,4}$를 파기하고, 그런 다음 $m_i^{r,4}$를 전파한다.

[0565] 단계 s, 5≤s≤m+2, s-2≡0 mod 3: BBA*의 0에 대해 고정된 코인 단계

[0566] 모든 사용자 i∈PK^{r-k}에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 s를 시작한다.

[0567] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 i∈SV^{r,s}가 있는지 여부를 확인한다.

[0568] · 만약 $i \notin SV^{r,s}$라면, i는 바로 자신의 s단계의 실행을 중단한다.

[0569] · $i \in SV^{r,s}$인 경우 그는 다음과 같이 동작한다.

[0570] - 그는 시간 $t_s \triangleq t_{s-1} + 2\lambda = (2s-3)\lambda + \Lambda$ 이 지나갈 때까지 대기한다.

[0571] - 종료 조건 0: 이러한 대기 동안 그리고 임의의 시점에,

[0572] (a) $5 \leq s' < s$, $s'-2 \equiv 0 \bmod 3$, 즉, 단계 s'는 0에 고정된 코인 단계이고,

[0573] (b) i는 적어도 $t_H = \frac{2n}{3} + 1$ 유효 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$[17]을 수신했고,

[0574] (17: 플레이어 j로부터의 그러한 메시지는 플레이어 i가 j로부터 1에 대해 서명한 메시지를 받았을지라도 카운트된다. 종료 조건 1과 유사한 사항. 분석에서 볼 수 있듯이, 이는 모든 정직한 사용자가 서로로부터 시간 λ내에 B^r을 아는 것을 보장하기 위해 수행된다.)

[0575] (c) i는 $v = H(B_j^r)$을 가진 유효 메시지 $m_j^{r,1} = (B_j^r, esig_j(H(B_j^r)), \sigma_j^{r,1})$을 수신하도록,

[0576] 스트링 $v' \neq \perp$ 및 단계 s'가 존재하는 경우, i는 아무것도 전파하지 않고 바로 단계 s(그리고 실제로 라운드 r)의 자신의 실행을 중단하고; $B^r = B_j^r$을 설정하고; 자신의 $CERT^r$을 하위 단계(b)의 메시지 $m_j^{r,s'-1}$의 세트로 설정한다.[18]

[0577] (18: 사용자 i는 이제 B^r과 그의 라운드 r이 끝나는 것을 안다. 그는 여전히 메시지를 일반 사용자로서 전파하는 데 도움을 주지만, (r, s)-검증자로서 전파를 시작하지는 않는다. 특히 그는 자신의 $CERT^r$에서 모든 메시지를 전파하는 데 도움을 주었으며, 이는 우리의 프로토콜에 충분하다. 이진 BA 프로토콜에 대해 $b_i \triangleq 0$로 설정해야하지만, 이 경우 어쨌든 b_i는 필요하지 않다는 것에 유의하라. 앞으로의 모든 지시에 대해 유사한 사항이다.)

[0578] - 종료 조건 1: 이러한 대기 동안 그리고 임의의 시점에,

[0579] (a') $6 \leq s' < s$, $s'-2 \equiv 1 \bmod 3$, 즉, 단계 s'는 1에 고정된 코인 단계이고,

[0580] (b') i는 적어도 t_H 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$를 수신하도록,[19]

[0581] (19: 이 경우에는 v_j가 무엇인지는 중요하지 않다.)

[0582] 단계 s'가 존재하는 경우, i는 아무것도 전파하지 않고 바로 단계 s(그리고 실제로 라운드 r)의 자신의 실행을 중단하고, $B^r = B_\epsilon^r$을 설정하고; 자신의 $CERT^r$을 하위 단계(b')의 메시지 $m_j^{r,s'-1}$로 설정한다.

[0583] - 그렇지 않으면 대기 상태의 종료시 사용자 i는 다음을 수행한다.

[0584] 그는 v_i를 그가 수신한 모든 유효한 $m_j^{r,s-1}$의 두 번째 구성 요소에서의 v_j의 과반수 투표로 설정한다.

[0585] 그는 b_i를 다음과 같이 계산한다.

[0586] 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는

$b_i \triangleq 0$으로 설정한다.

[0587] 그렇지 않고 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는 $b_i \triangleq 1$로 설정한다.

[0588] 그렇지 않으면 그는 $b_i \triangleq 0$로 설정한다.

[0589] 그는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$을 계산하여, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기한 다음 $m_i^{r,s}$을 전파한다.

[0590] 단계 s, $6 \le s \le m+2$, $s-2 \equiv 1 \bmod 3$: $BBA^{\star}$의 코인 고정 1 단계

[0591] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 s를 시작한다.

[0592] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 $i \in SV^{r,s}$가 있는지 여부를 확인한다.

[0593] · 만약 $i \notin SV^{r,s}$라면, i는 바로 자신의 s단계의 실행을 중단한다.

[0594] · $i \in SV^{r,s}$인 경우 그는 다음과 같이 동작한다.

[0595] - 그는 시간 $t_s \triangleq (2s-3)\lambda + \Lambda$이 지나갈 때까지 대기한다.

[0596] - 종료 조건 0: 0에 고정된 코인 단계와 동일한 지침.

[0598] *- 종료 조건 1: 0에 고정된 코인 단계와 동일한 지침.

[0599] - 그렇지 않으면 대기 상태의 종료시에 사용자 i는 다음을 수행한다.

[0600] 그는 v_i를 그가 수신한 모든 유효한 $m_j^{r,s-1}$의 두 번째 구성 요소에서의 v_j의 과반수 투표로 설정한다.

[0601] 그는 b_i를 다음과 같이 계산한다.

[0602] 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는 $b_i \triangleq 0$으로 설정한다.

[0603] 그렇지 않고 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는 $b_i \triangleq 1$로 설정한다.

[0604] 그렇지 않으면, 그는 $b_i \triangleq 1$로 설정한다.

[0605] 그는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$을 계산하여, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기한 다음 $m_i^{r,s}$을 전파한다.

[0606] 단계 s, 7≤s≤m+2, s-2≡2 mod 3: $BBA^{\star}$의 코인이 전체적으로 플립핑된(Coin-Genuinely-Flipped) 단계

[0607] 모든 사용자 i∈PK^{r-k}에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 s를 시작한다.

[0608] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 i∈$SV^{r,s}$가 있는지 여부를 확인한다.

[0609] · 만약 $i \notin SV^{r,s}$라면, i는 바로 자신의 s단계의 실행을 중단한다.

[0610] · i∈$SV^{r,s}$인 경우 그는 다음과 같이 동작한다.

[0611] - 그는 시간 $t_s \triangleq (2s-3)\lambda + \Lambda$이 지나갈 때까지 대기한다.

[0612] - 종료 조건 0: 0에 고정된 코인 단계와 동일한 지침.

[0613] - 종료 조건 1: 0에 고정된 코인 단계와 동일한 지침.

[0614] - 그렇지 않으면 대기 상태의 종료시에 사용자 i는 다음을 수행한다.

[0615] 그는 v_i를 그가 수신한 모든 유효한 $m_j^{r,s-1}$의 두 번째 구성 요소에서의 v_j의 과반수 투표로 설정한다.

[0616] 그는 b_i를 다음과 같이 계산한다.

[0617] 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는 $b_i \triangleq 0$으로 설정한다.

[0618] 그렇지 않고 그가 받은 모든 유효한 $m_j^{r,s-1}$의 2/3 이상이 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$의 형태라면, 그는 $b_i \triangleq 1$로 설정한다.

[0619] 그렇지 않으면, $SV_i^{r,s-1}$를 그가 유효한 메시지 $m_j^{r,s-1}$를 수신한 (r, s-1)-검증자의 세트로 놓자. 그는 $b_i \triangleq \mathtt{lsb}(\min_{j \in SV_i^{r,s-1}} H(\sigma_j^{r,s-1}))$를 설정한다.

[0620] 그는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$을 계산하여, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기한 다음 $m_i^{r,s}$을 전파한다.

[0621] 단계 m+3: $BBA^{\star}$의 마지막 단계[20]

[0623] *(20: 압도적인 확률로 $BBA^{\star}$가 이 단계 전에 끝났으며, 이 단계를 완료하기 위해 지정한다.)

[0624] 모든 사용자 i∈PK^{r-k}에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 m+3을 시작한다.

[0625] · 사용자 i는 B^{r-1}의 세 번째 구성 요소로부터 Q^{r-1}을 계산하고 i∈SV^{m+3}가 있는지 여부를 확인한다.

[0626] · 만약 $i \notin SV^{r,m+3}$라면, i는 바로 자신의 m+3단계의 실행을 중단한다.

[0627] · $i \in SV^{m+3}$인 경우 그는 다음과 같이 동작한다.

[0628] - 그는 시간 $t_{m+3} \triangleq t_{m+2} + 2\lambda = (2m+3)\lambda + \Lambda$이 지나갈 때까지 대기한다.

[0629] - 종료 조건 0: 0에 고정된 코인 단계와 동일한 지침.

[0630] - 종료 조건 1: 0에 고정된 코인 단계와 동일한 지침.

[0631] - 그렇지 않으면 대기 상태의 종료시에 사용자 i는 다음을 수행한다.

[0632] 그는 $out_i \triangleq 1$ 및 $B^r \triangleq B^r_\epsilon$을 설정한다.

[0633] 그는 메시지 $m_i^{r,m+3} = (ESIG_i(out_i), ESIG_i(H(B^r)), \sigma_i^{r,m+3})$을 계산하여, 자신의 임시 비밀 키 $sk_i^{r,m+3}$을 파기한 다음 $m_i^{r,m+3}$을 전파하여 B^r을 인증한다.[21]

[0634] (21: m+3 단계로부터의 인증서에는 $ESIG_i(out_j)$가 포함될 필요가 없다. 균일성만(uniformity)을 유지하기 위해 그것을 포함한다. 생성된 단계에 상관없이 인증서는 균일한 포맷을 갖는다.)

[0635] 비 검증자에 의한 라운드 r 블록의 재구성

[0636] 시스템의 모든 사용자 i에 대한 지침: 사용자 i는 그가 B^{r-1}을 알게되는 즉시 라운드 r의 자신의 단계 s를 시작하고, 하기와 같이 블록 정보를 대기한다.

[0637] - 이러한 대기 동안 그리고 임의의 시점에,

[0638] (a) $5 \leq s' \leq m+3$, $s'-2 \equiv 0 \bmod 3$이고,

[0639] (b) i는 적어도 t_H 유효 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$을 수신했고,

[0640] (c) i는 $v = H(B_j^r)$을 가진 유효 메시지 $m_j^{r,1} = (B_j^r, esig_j(H(B_j^r)), \sigma_j^{r,1})$을 수신하도록,

[0641] 스트링 v 및 단계 s'가 존재하는 경우, i는 바로 라운드 r의 자신의 실행을 중단하고; $B^r = B_j^r$을 설정하고; 자신의 $CERT^r$을 하위 단계 (b)의 메시지 $m_j^{r,s'-1}$의 세트로 설정한다.

[0642] - 이러한 대기 동안 그리고 임의의 시점에,

[0643] (a') $6 \leq s' \leq m+3$, $s'-2 \equiv 0 \bmod 3$이고,

[0644] (b') i는 적어도 t_H 유효 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$을 수신하도록,

[0645] 단계 s'가 존재하는 경우, i는 바로 라운드 r의 자신의 실행을 중단하고; $B^r = B_j^r$을 설정하고; 자신의 $CERT^r$을 하위 단계 (b')의 메시지 $m_j^{r,s'-1}$의 세트로 설정한다.

[0646] - 이러한 대기 동안 그리고 임의의 시점에, i가 적어도 t_H 유효 메시지 $m_j^{r,m+3} = (ESIG_j(1), ESIG_j(H(B_\epsilon^r)), \sigma_j^{r,m+3})$을 수신한 경우, i는 바로 라운드 r의 자신의 실행을

중단하고; $B^r = B^r_\epsilon$을 설정하고; 자신의 CERT^{r}을 1 및 $H(B^r_\epsilon)$에 대해 메시지 $m_j^{r,m+3}$의 세트로 설정한다.

[0647] ## 5.5 $Algorand'_1$의 분석

[0648] 분석에 사용된 각 라운드 r≥0에 대한 다음 표기를 소개한다.

[0649] · 제1 정직한 사용자가 B^{r-1}을 알고있는 시간을 T^{r}이라고 한다.

[0650] · I^{r+1}을 간격 [T^{r+1}, T^{r+1}+λ]라고 한다.

[0651] 프로토콜 초기화에 의해 T^0=0임을 유의하라. 각각의 s≥1과 i∈SV^{r,s}에 대해 $\alpha_i^{r,s}$와 $\beta_i^{r,s}$는 각각 플레이어 i의 단계 s의 시작 시간과 종료 시간이라는 것을 상기하라. 또한 각 2≤s≤m+3에 대해 $t_s = (2s-3)\lambda + \Lambda$임을 상기하라. 또한 $I^0 \triangleq \{0\}$ 및 $t_1 \triangleq 0$이라고 하자.

[0652] 마지막으로, $L^r \leq m/3$은, 각 시도는 확률 $\frac{p_h}{2}$을 가진 1이고 최대 m/3 번의 시도가 있을 때, 1을 보는데 필요한 베르누이 시도의 수를 나타내는 랜덤 변수임을 상기하라. 모든 시도가 실패하면 $L^r \triangleq m/3$이다.

[0653] 실제로 메시지를 전파하는 데 필요한 시간에 비해 무시할 만하기 때문에, 분석시 우리는 계산 시간을 무시한다. 임의의 경우에, 약간 큰 λ와 Λ를 사용하여, 계산 시간을 분석에 직접 통합할 수 있다. 아래의 진술 대부분은 "압도적인 확률"을 유지하며 분석에서 이 사실을 반복적으로 강조하지 않을 수도 있다.

[0654] ## 5.6 메인 정리(main theorem)

[0655] **정리 5.1.**

[0656] 다음 속성은 각 라운드 r≥0에 대해 압도적인 확률로 유지된다:

[0657] 1. 모든 정직한 사용자가 같은 블록 B^{r}에 동의한다.

[0658] 2. 리더 ℓ^r가 정직할 때, 블록 B^{r}은 ℓ^r에 의해 생성되고, B^{r}은 ℓ^r에 의해 시간 $\alpha_{\ell^r}^{r,1}$, $T^{r+1} \leq T^r + 8\lambda + \Lambda$까지 수신된 최대 페이셋을 포함하고, 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^{r}을 안다.

[0659] 3. 리더 ℓ^r이 악의적인 경우, $T^{r+1} \leq T^r + (6L^r + 10)\lambda + \Lambda$이고 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^{r}을 안다.

[0660] 4. Lr에 대해 $p_h = h^2(1 + h - h^2)$이고, 리더 ℓ^r는 적어도 p_h의 확률로 정직하다.

[0661] 우리의 메인 정리를 증명하기 전에 두 가지 유의를 해보자.

[0662] **블록 생성 및 진정한 대기 시간.**

[0663] 블록 B^{r}을 생성하는 시간은 T^{r+1}-T^{r}로 정의된다. 즉, 정직한 사용자가 B^{r}을 처음 배울 때와 일부 정직한 사용자가 B^{r-1}을 처음 배울 때의 차이로 정의된다. 라운드 r 리더가 정직할 때, 속성 2의 우리의 메인 정리는 h>2/3의 정확한 값이 무엇이든 상관없이 B^{r}을 생성하는 정확한 시간이 8λ+Λ 시간임을 보장한다. 리더가 악의적일 때, 속성 3은 B^{r}을 생성할 예상 시간이 다시 h의 정확한 값에 상관없이 $(\frac{12}{p_h} + 10)\lambda + \Lambda$로 상한이 정해짐을 의미

한다.[22] 그러나 B^r 생성 예상 시간은 h의 정확한 값에 따라 달라진다. 사실, 속성 4에 의해, $p_h = h^2(1+h-h^2)$ 이고 리더는 적어도 p_h의 확률로 정직하며, 따라서,

[0664] $\mathbb{E}[T^{r+1}-T^r] \le h^2(1+h-h^2)\cdot(8\lambda+\Lambda)+(1-h^2(1+h-h^2))((\frac{12}{h^2(1+h-h^2)}+10)\lambda+\Lambda)$ 이다.

[0665] (22: 실제로, $\mathbb{E}[T^{r+1}-T^r] \le (6\mathbb{E}[L^r]+10)\lambda+\Lambda = (6\cdot\frac{2}{p_h}+10)\lambda+\Lambda = (\frac{12}{p_h}+10)\lambda+\Lambda$ 이다.)

[0667] 예를 들어, h = 80%이면, $\mathbb{E}[T^{r+1}-T^r] \le 12.7\lambda+\Lambda$ 이다.

[0668] 정리의 증명 5.1.

[0669] 우리는 유도에 의해 속성 1-3을 증명한다: 그것들이 라운드 r-1(일반성을 잃지 않으면서, 그것들은 r=0일 때 " 라운드 -1"에 대해 자동으로 유지함)에 대해 유지한다고 가정하면, 우리는 그것들을 라운드 r에 대해 증명한다.

[0670] B^{r-1}은 귀납적 가정(inductive hypothesis)에 의해 고유하게 정의되므로 세트 $SV^{r,s}$는 라운드 r의 각 단계 s에 대해 고유하게 정의된다. n_1의 선택에 의해, 압도적인 확률로 $SV^{r,1} \neq \emptyset$ 이다. 이제 섹션 5.7과 5.8에서 증명된 다음 두 가지 보조 정리(lemma)를 기술한다. 두 보조 정리의 유도 및 증명을 통해, 라운드 0에 대한 분석은 유도 단계와 거의 동일하며, 우리는 발생하는 차이점을 강조한다.

[0671] **보조 정리 5.2.** [완료성(Completeness) 보조정리] 리더 ℓ^r가 정직할 때, 압도적인 확률로 속성 1-3이 라운드 r-1에 대해 유지한다고 가정하면,

[0672] · 모든 정직한 사용자는 리더 ℓ^r에 의해 생성되고 시간 $\alpha_{\ell^r}^{r,1} \in I^r$ 까지 ℓ^r에 의해 최대 페이셋을 포함하는 동일한 블록 B^r에 동의한다;

[0673] · $T^{r+1}<T^r+8\lambda+\Lambda$이고 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^r을 안다.

[0674] **보조 정리 5.3.** [건전성 보조정리] 리더 ℓ^r가 악의적일 때, 속성 1-3이 라운드 r-1에 대해 유지하고, 압도적인 확률로 모든 정직한 사용자가 동일한 블록 B^r에 동의하고, $T^{r+1} \le T^r + (6L^r+10)\lambda+\Lambda$이고 시간 간격 I^{r+1}에서 B^r을 안다.

[0675] 보조정리 5.2와 5.3을 r= 0 및 유도 단계에 적용함으로써 속성 1-3을 유지한다. 마지막으로 5.9 절에서 증명된 다음의 보조 정리로 속성 4를 다시 나타낸다.

[0676] **보조 정리 5.4.** r에 앞서 각 라운드마다 속성 1-3이 주어지면, L^r에 대해 $p_h=h^2(1+h-h^2)$이고, 리더 ℓ^r는 적어도 p_h 이상 확률로 정직하다.

[0677] 위의 세 가지 보조 정리를 결합하면 정리 5.1이 성립한다.

[0678] 다음의 보조 정리는 귀납적 가설을 가정한 라운드 r에 관한 몇 가지 중요한 특성을 기술하고, 위의 세 가지 보조 정리의 증명에 사용될 것이다.

[0679] **보조 정리 5.5.** 속성 1-3이 라운드 r-1에 대해 유지된다고 가정한다. 라운드 r의 각 단계 s≥1 및 각각의 정직한 검증자 $i \in HSV^{r,s}$의 경우, 우리는 다음을 가진다

[0680] (a) $\alpha_i^{r,s} \in I^r$;

[0681] (b) 플레이어 i가 t_s의 시간을 기다린 경우, r>0에 대해 $\beta_i^{r,s} \in [T^r + t_s, T^r + \lambda + t_s]$ 이고, r=0에 대해 $\beta_i^{r,s} = t_s$ 이다.

[0682] (c) 만약 플레이어 i가 시간 t_s를 기다렸다면, 시간 $\beta_i^{r,s}$ 까지 그는 모든 단계 s'<s에 대해 모든 정직한 검증자 j∈$HSV^{r,s'}$에 의해 전송된 모든 메시지를 수신했다.

[0683] 또한 각 단계 s≥3에 대해, 우리는 다음을 가진다

[0684] (d) 두명의 상이한 플레이어 i, i'∈$SV^{r,s}$와 같은 길이의 두 개의 상이한 값v, v'가 존재하지 않아 플레이어 i가 수신하고 v를 위해 서명한 모든 유효 메시지 $m_j^{r,s-1}$ 의 2/3 및 플레이어 i'가 수신하고 v'를 위해 서명한 모든 유효 메시지 $m_j^{r,s-1}$ 의 2/3를 두 플레이어가 시간 t_s를 기다리도록 한다.

[0685] 증명.

[0686] 속성(a)는 플레이어 i가 시간 간격 I^r에서 B^{r-1}을 알고 즉시 자신의 단계 s를 시작할 때 귀납적인 가설에서 직접적으로 따라온다. 속성(b)는 (a)에서 직접 따르는데: 플레이어 i는 실시하기 전에 시간 t_s를 기다리기 때문에, $\beta_i^{r,s} = \alpha_i^{r,s} + t_s$ 이다.

[0687] 이제 속성(c)를 증명한다. s=2이면, 속성(b)에 의해, 모든 검증자인 j∈$HSV^{r,1}$에 대해 우리는 다음을 가진다

[0688] $$\beta_i^{r,s} = \alpha_i^{r,s} + t_s \geq T^r + t_s = T^r + \lambda + \Lambda \geq \beta_j^{r,1} + \Lambda$$.

[0689] 각 검증자 j∈$HSV^{r,1}$은 시간 $\beta_j^{r,1}$ 에서 자신의 메시지를 전송하고 메시지는 시간 $\beta_i^{r,s}$만큼 기껏해야 Λ 시간에 모든 정직한 사용자들에게 도달하기 때문에, 플레이어 i는 원하는 대로 $HSV^{r,1}$내의 모든 검증자들에 의해 전송된 메시지들을 수신한다.

[0690] s>2인 경우, $t_s=t_{s-i}+2\lambda$ 이다. 속성(b)에 의하면, 모든 단계 s'<s 및 모든 검증자 j∈$HSV^{r,s'}$에 대해,

[0691] $$\beta_i^{r,s} = \alpha_i^{r,s} + t_s \geq T^r + t_s = T^r + t_{s-1} + 2\lambda \geq T^r + t_{s'} + 2\lambda = T^r + \lambda + t_{s'} + \lambda \geq \beta_j^{r,s'} + \lambda$$ 이다.

[0692] 각 검증자 j∈$HSV^{r,s'}$는 시간 $\beta_j^{r,s'}$ 에서 자신의 메시지를 전송하고 메시지는 시간 $\beta_i^{r,s}$만큼 기껏해야 λ시간에 모든 정직한 사용자에게 도달하기 때문에, 플레이어 i는 모든 s'<s에 대해 $HSV^{r,s'}$에서 모든 정직한 검증자에 의해 전송된 모든 메시지를 수신한다. 따라서, 속성(c)는 유지된다.

[0693] 마지막으로 속성(d)를 증명한다. 검증자 j∈$SV^{r,s-1}$는 단계 s-1에서, 해시 함수의 출력과 동일한 길이의 값 v_j와 s-1≥4인 경우 비트 b_j∈{0, 1}인 그들의 임시 비밀 키를 이용하여 최대 2개의 것에서 서명한다는 것에 유의하라. 그것이, 보조정리의 설명에서, 우리가 v와 v'가 같은 길이를 가질 것을 요구하는 이유이고, 많은 검증자가 해시 값 v와 비트 b를 모두 서명하여, 모두 2/3 임계 값을 통과하게한다.

[0694] 모순을 위해, 원하는 검증자 i 및 i'과 값 v, v'가 있다고 가정한다. $MSV^{r,s-1}$의 일부 악의적인 검증자는 v와 v' 모두에 서명했을 수 있지만, $HSV^{r,s-1}$의 각각의 정직한 검증자는 그것들 중 대부분에 서명했다는 것에 유의하라. 속성(c)에 따르면, i와 i' 모두 $HSV^{r,s-1}$의 모든 정직한 검증자가 보낸 모든 메시지를 받았다.

[0695] $HSV^{r,s-1}(v)$를 v에 서명한 정직한 (r, s-1) 검증자의 세트, $MSV_i^{r,s-1}$ 를 i가 유효 메시지를 수신한 악의적인 (r, s-1) 검증자 세트, 및 $MSV_i^{r,s-1}(v)$ 를 i가 v에 서명한 유효 메시지를 수신한 $MSV_i^{r,s-1}$ 의 하위 세트로 놓자. i와 v에 대한 요구 사항에 따라, 우리는 다음을 가진다

[0696] $$ratio \triangleq \frac{|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)|}{|HSV^{r,s-1}| + |MSV_i^{r,s-1}|} > \frac{2}{3} \quad (1)$$

[0697] 우리는 먼저 다음을 보여준다.

[0698] $$|MSV_i^{r,s-1}(v)| \leq |HSV^{r,s-1}(v)| \quad (2)$$

[0699] 그렇지 않다면, 파라미터들 사이의 관계에 의해, 압도적인 확률로 $|HSV^{r,s-1}| > 2|MSV^{r,s-1}| \geq 2|MSV_i^{r,s-1}|$ 이 되고, 따라서

[0700] $$ratio < \frac{|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)|}{3|MSV_i^{r,s-1}|} < \frac{2|MSV_i^{r,s-1}(v)|}{3|MSV_i^{r,s-1}|} \leq \frac{2}{3}$$ 이 되고,

[0701] 부등식 1과 모순된다.

[0702] 다음으로, 부등식 1에 의해, 우리는 다음을 가진다

[0703] $$2|HSV^{r,s-1}| + 2|MSV_i^{r,s-1}| < 3|HSV^{r,s-1}(v)| + 3|MSV_i^{r,s-1}(v)|$$

[0704] $$\leq \quad 3|HSV^{r,s-1}(v)| + 2|MSV_i^{r,s-1}| + |MSV_i^{r,s-1}(v)|.$$

[0705] 부등식 2와 결합하여,

[0706] $2|HSV^{r,s-1}| < 3|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)| \leq 4|HSV^{r,s-1}(v)|$ 이 되고, 이는 다음을 의미한다

[0707] $$|HSV^{r,s-1}(v)| > \frac{1}{2}|HSV^{r,s-1}|.$$

[0708] 유사하게, i'와 v'에 대한 요구사항에 따라

[0709] $$|HSV^{r,s-1}(v')| > \frac{1}{2}|HSV^{r,s-1}|$$ 이 된다.

[0710] 정직한 검증자 $j \in HSV^{r,s-1}$는 자신의 메시지를 전달하기 전에 자신의 임시 비밀 키 $sk_j^{r,s-1}$ 를 파괴하기 때문에, 적은 j가 검증자임을 알게된 후에 j가 서명하지 않은 값에 대해 j의 서명을 위조할 수 없다. 따라서 위의 두 가지 부등식은 $|HSV^{r,s-1}| \geq |HSV^{r,s-1}(v)| + |HSV^{r,s-1}(v')| > |HSV^{r,s-1}|$, 모순을 내포하고 있다. 따라서 원하는 i, i', v, v'는 존재하지 않으며 속성(d)이 유지된다.

[0711] **5.7 완전성 보조 정리**

[0712] **보조 정리 5.2.** [완전성 보조정리, 재진술] 리더 ℓ^r 가 정직할 때, 압도적인 확률로 라운드 r-1에 대해 속성 1-3을 유지한다고 가정하면,

[0713] · 모든 정직한 사용자는 ℓ^r 에 의해 생성되고 시간 $\alpha_{\ell^r}^{r,1} \in I^r$ 까지 ℓ^r 에 의해 수신되는 최대 페이셋을 포함

하는 동일한 블록 B^r에 동의한다.

[0714] · $T^{r+1} \le T^r + 8\lambda + \Lambda$ 그리고 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^r을 안다.

[0715] 증명. 귀납 가설과 보조 정리 5.5에 의해 각 단계 s와 검증자 $i \in HSV^{r,s}$에 대해 $\alpha_i^{r,s} \in I^r$이다. 아래에서 우리는 프로토콜을 단계별로 분석한다.

[0716] **단계 1.** 정의에 따라 모든 정직한 검증자 $i \in HSV^{r,1}$은 시간 $\beta_i^{r,1} = \alpha_i^{r,1}$에 원하는 메시지 $m_i^{r,1}$를 전파하고, 여기서 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$이고, $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$이고, PAY_i^r은 i가 시간 $\alpha_i^{r,1}$까지 본 모든 결제 중에서 최대의 페이셋이다.

[0717] **단계 2.** 정직한 검증자 $i \in HSV^{r,2}$를 임의대로 수정(fix)하라. 보조 정리 5.5에 따르면, 플레이어 i가 시간 $\beta_i^{r,2} = \alpha_i^{r,2}$에서 대기하면서 수행될 때, 그는 $m_{\ell^r}^{r,1}$을 포함하여 $HSV^{r,1}$에서 검증자들에 의해 전송된 모든 메시지를 수신했다. ℓ^r의 정의에 따라, 그의 보증서의 해시 값이 $H(\sigma_{\ell^r}^{r,1})$보다 작은 PK^{r-k}에 다른 플레이어가 존재하지 않는다. 물론, 적들이 $H(\sigma_{\ell^r}^{r,1})$가 매우 작다는 것을 안 후에 ℓ^r를 손상시킬 수 있지만, 그 시간까지 플레이어 ℓ^r가 자신의 임시 비밀 키를 파괴하고 메시지 $m_{\ell^r}^{r,1}$는 전파되었다. 따라서 검증자 i는 자신의 리더를 플레이어 ℓ^r로 설정한다. 따라서 시간 $\beta_i^{r,2}$에, 검증자 i는 $m_i^{r,2} = (ESIG_i(v_i'), \sigma_i^{r,2})$를 전파하고, 여기서 $v_i' = H(B_{\ell^r}^r)$이다. r=0일 때 유일한 차이는 범위 내에 있는 것이 아니라 $\beta_i^{r,2} = t_2$이다. 비슷한 일들을 미래의 단계에 대해 말할 수 있으며 우리는 그것들을 다시 강조하지 않을 것이다.

[0718] **단계 3.** 정직한 검증자 $i \in HSV^{r,3}$을 임의대로 수정(fix)하라. 보조 정리 5.5에 따르면, 플레이어 i가 시간 $\beta_i^{r,3} = \alpha_i^{r,3} + t_3$에서 대기하면서 수행될 때, 그는 $HSV^{r,2}$에서 검증자들에 의해 전송된 모든 메시지를 수신했다.

[0719] 파라미터들 사이의 관계에 의해 압도적인 확률로 $|HSV^{r,s}| > 2|MSV^{r,s}|$이다. 더욱이, 정직한 검증자가 모순되는 메시지에 서명하지 않으며, 적은 정직한 검증자의 대응하는 임시 비밀 키를 파괴한 후에 정직한 검증자의 서명을 위조할 수 없다. 따라서 i가 받은 모든 유효한(r, 2) 메시지의 2/3 이상은 정직한 검증자로부터의 것이고, 모순이 없는 형태 $m_j^{r,2} = (ESIG_j(H(B_{\ell^r}^r)), \sigma_j^{r,2})$이다.

[0720] 따라서, 시간 $\beta_i^{r,3}$에서 플레이어 i는 $m_i^{r,3} = (ESIG_i(v'), \sigma_i^{r,3})$이고, 여기서 $v' = H(B_{\ell^r}^r)$이다.

[0721] **단계 4.** 정직한 검증자 $i \in HSV^{r,4}$를 임의대로 수정(fix)하라. 보조 정리 5.5에 따르면, 플레이어 i가 시간 $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$.에서 대기하면서 수행될 때, 그는 $HSV^{r,3}$에서 검증자들에 의해 전송된 모든 메시지를 수신했다. 단계 3과 유사하게, i가 받은 모든 유효한(r, 3) 메시지의 2/3 이상은 정직한 검증자로부터의 것이고, $m_j^{r,3} = (ESIG_j(H(B_{\ell^r}^r)), \sigma_j^{r,3})$의 형태이다.

[0722] 따라서, 플레이어 i는 $v_i = H(B^r_{\ell r}),\ g_i = 2$, 및 $b_i = 0$으로 설정한다. 시간 $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$에서, 그는 $m_i^{r,4} = (ESIG_i(0), ESIG_i(H(B^r_{\ell r})), \sigma_i^{r,4})$를 전파한다.

[0723] **단계 5.** 정직한 검증자 i∈$HSV^{r,5}$를 임의대로 수정(fix)하라. 보조 정리 5.5에 따르면, 그가 시간 $\alpha_i^{r,5} + t_5$까지 대기하면, 플레이어 i는 $HSV^{r,4}$에서 검증자들에 의해 전송된 모든 메시지를 수신할 것이다. $|HSV^{r,4}| \geq t_H$ 라는 것에 유의하라.[23] 또한, $HSV^{r,4}$에서의 모든 검증자는 $H(B^r_{\ell r})$에 대해 서명했다는 것에 유의하라.

[0724] (23: 엄밀히 말하자면, 이것은 매우 높은 확률로 발생하지만 반드시 압도적 인것은 아니다. 그러나, 이 확률은 프로토콜의 실행 시간에 약간 영향을 주지만 그의 정확성에는 영향을 미치지 않는다. h=80%일 때, 확률 $1-10^{-8}$으로 $|HSV^{r,4}| \geq t_H$이다. 이 이벤트가 발생하지 않으면 프로토콜은 또 다른 3단계로 계속된다. 이것이 두 단계로 발생하지 않을 가능성은 미미하므로, 단계 8에서 프로토콜이 완료될 것이다. 예상시, 필요한 단계 수는 거의 5이다.)

[0725] $|MSV^{r,4}| < t_H$인 경우, $SV^{r,4}$에서 t_H 검증자(필수적으로 악의적이 되는)가 서명할 수 있었던 $v' \neq H(B^r_{\ell r})$가 없기 때문에 플레이어 i는 그가 t_H의 유효한 메시지 $m_j^{r,4} = (ESIG_j(0), ESIG_j(H(B^r_{\ell r})), \sigma_j^{r,4})$를 수신하기 전에는 중단하지 않는다. T를 후자의 사건이 일어나는 시간이라고 하자. 이러한 메시지 중 일부는 악의적인 플레이어에서 온 것일 수 있지만, $|MSV^{r,4}| < t_H$이기 때문에, 그것들 중 적어도 하나는 $HSV^{r,4}$에서의 정직한 검증자로부터 온 것이고 T^r+t_4 시간 후에 전송된다. 따라서, $T \geq T^r + t_4 > T^r + \lambda + \Lambda \geq \beta_{\ell r}^{r,1} + \Lambda$이고, 시간 T까지 플레이어 i도 메시지 $m_{\ell r}^{r,1}$를 수신했다. 프로토콜의 구성에 의해, 플레이어 i는 아무것도 전파하지 않고 시간 $\beta_i^{r,5} = T$에서 멈추고; $B^r = B^r_{\ell r}$을 설정하고; 자신의 $CERT^r$을 자신이 받은 0 및 $H(B^r_{\ell r})$에 대한 (r, 4) 메시지 세트로 설정한다.

[0726] **단계 s> 5.** 유사하게, 임의의 단계 s> 5 및 임의의 검증자 i∈$HSV^{r,s}$에 대해, 플레이어 i는 시간 $\alpha_i^{r,s} + t_s$까지 기다린 경우 $HSV^{r,4}$에서 검증자에 의해 전송된 모든 메시지를 수신했을 것이다. 같은 분석에 의해, 플레이어 i는 $B^r = B^r_{\ell r}$로 설정(그리고 자신의 $CERT^r$을 적절하게 설정)하는 것을 어떤 것도 전파하지 않고 중단한다. 물론, 악의적인 검증자는 멈추지 않고 임의의 메시지를 전파할 수 있지만, $|MSV^{r,s}| < t_H$이기 때문에 유도에 의해 어떤 다른 v'도 임의의 단계 4≤s'<s에서 t_H 검증자에 의해 서명될 수 없고, 따라서 정직한 검증자는 그들이 0과 $H(B^r_{\ell r})$에 대해 t_H 유효(r, 4) 메시지를 수신했기 때문에만 중단한다.

[0727] **라운드 r 블록의 재구성.**

[0728] 단계 5의 분석은 거의 변경 없이 일반의 정직한 사용자에게 적용된다. 사실, 플레이어 i는 간격 I^r에서 자신의 라운드 r을 시작하고 $H(B^r_{\ell r})$에 대해 t_H 유효(r, 4) 메시지를 수신한 시간 T에서만 멈추게 된다. 다시 말하면 이들 메시지 중 적어도 하나가 정직한 검증자로부터 왔고 시간 T^r+t_4 이후에 전송되었기 때문에, 플레이어 i는

시간 T까지 $m_{\ell^r}^{r,1}$을 받았다. 따라서 그는 적절한 CERT^{r}로 $B^r = B_{\ell^r}^r$을 설정한다.

[0729] 모든 정직한 사용자가 시간 간격 I^{r+1} 이내에 라운드 r을 끝내는 것으로 나타난다. 단계 5의 분석에 의해, 모든 정직한 검증자는 i∈HSV^{r,5}가 $\alpha_i^{r,5} + t_5 \leq T^r + \lambda + t_5 = T^r + 8\lambda + \Lambda$에 또는 그전에 B^{r}을 안다. T^{r+1}은 최초의 정직한 사용자 i^{r}이 B^{r}을 알고 있는 시간이기 때문에, 우리는 원하는 경우 다음을 가진다

[0730] $$T^{r+1} \leq T^r + 8\lambda + \Lambda.$$

[0731] 또한, 플레이어 i^{r}가 B^{r}을 알고 있을 때, 그는 이미 자신의 CERT^{r}에서 메시지 전파를 도왔다. 플레이어 i^{r}이 그것들을 처음 전파한 사람이라 할지라도, 모든 메시지는 시간 λ 내에 모든 정직한 사용자가 수신한다는 것에 유의하라. 또한 위의 분석에 후속하여, 우리는 $T^{r+1} \geq T^r + t_4 \geq \beta_{\ell^r}^{r,1}$을 가지고, 따라서 모든 정직한 사용자는 시간 $T^{r+1}+\lambda$까지 $m_{\ell^r}^{r,1}$을 수신한다. 따라서 모든 정직한 사용자는 시간 간격 $I^{r+1} = [T^{r+1}, T^{r+1} + \lambda]$에서 B^{r}을 안다.

[0732] 마지막으로, r=0에 대해 우리는 실제로 $T^1 \leq t_4 + \lambda = 6\lambda + \Lambda$를 갖는다. 모든 것을 결합하여, 보조 정리 5.2가 성립한다. ■

[0733] **5.8 건전성 보조 정리**

[0734] 보조 정리 5.3. [건전성 보조 정리, 재설명] 만약 리더 ℓ^r가 악의적일 때, 속성 1-3이 라운드 r-1에 대해 압도적인 확률로 유지한다고 가정하면, 모든 정직한 사용자들은 동일한 블록 B^r, $T^{r+1} \leq T^r + (6L^r + 10)\lambda + \Lambda$에 동의하고, 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^{r}을 안다.

[0735] 증명. 프로토콜의 두 부분인 GC와 BBA*를 별도로 고려한다.

[0736] GC. 귀납적인 가정과 보조 정리 5.5에 의해, 임의의 단계 s∈{2, 3, 4}와 정직한 검증자 i∈HSV^{r,s}에 대해서, 플레이어 i가 시간 $\beta_i^{r,s} = \alpha_i^{r,s} + t_s$에서 작동할 때, 그는 단계 s'<s에서 모든 정직한 검증자가 전송한 모든 메시지를 수신한다. 우리는 단계 4에서 2개의 가능한 경우를 구별한다.

[0737] 케이스 1. 검증자 i∈HSV^{r,4}가 g_i=2로 설정하지 않는다.

[0738] 이 경우, 모든 검증자 i∈HSV^{r,4}에 대해 정의에 의해 b_i=1이다. 즉, 그것들은 이진 BA 프로토콜에서 1에 대한 합의로 시작한다. 그들은 그들의 vi의 동의에 동의하지 않을지도 모르지만, 우리가 이진 BA에서 볼 수 있듯이 이것은 중요하지 않다.

[0739] 케이스 2. g_i=2가 되도록 검증자 i∈HSV^{r,4}가 존재한다.

[0740] 이 경우, 우리는

[0741] (1) 모든 i∈HSV^{r,4}에 대해 $g_i \geq 1$이고,

[0742] (2) 모든 i∈HSV^{r,4}에 대해 vi=v'이 되도록 값 v'가 존재하고,

[0743] (3) $v' = H(B_\ell^r)$가 되도록 일부 검증자 $\ell \in SV^{r,1}$로부터 유효한 메시지 $m_\ell^{r,1}$이 존재한다.

[0744] 사실, 플레이어 $\hat{i}$는 정직하고 g_i= 2로 설정하기 때문에, 그가 받은 모든 유효한 메시지 $m_j^{r,3}$ 중 2/3 이상은 동일한 값 $v' \neq \perp$에 대한 것이고, 그는 v= v'로 설정했다. 임의의 다른 정직한 (r, 4)-검증자 i에 대한 보조 정리 5.5의 속성(d)에 따르면, i가 받은 모든 유효한 메시지 $m_j^{r,3}$ 중 2/3 이상이 동일한 값 $v'' \neq v'$에 대한 것일 수는 없다. 따라서, 만약 i가 g_i= 2를 설정한다면, 그것은 v'에 대해 i가 2/3의 대다수보다 커야만 하고, 또한 원할 경우 v=v'로 설정해야만 한다.

[0745] 이제 g_i<2인 임의의 검증자 i∈$HSV^{r,4}$를 고려하라. 보조 정리 5.5에서 속성(d)의 분석과 유사하게, 플레이어 $\hat{i}$는 v'에 대해 2/3의 대다수보다 큰 것으로 보았기 때문에, $\frac{1}{2}|HSV^{r,3}|$ 이상의 정직한 (r, 3)-검증자가 v'에 서명했다. i는 시간 $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$까지 정직한 (r, 3)-검증자에 의한 모든 메시지를 수신했기 때문에, 그는 특히 v'에 대해 그것들로부터 $\frac{1}{2}|HSV^{r,3}|$ 이상의 메시지를 수신했다. $|HSV^{r,3}| > 2|MSV^{r,3}|$이기 때문에, i는 v'에 대해 1/3 대다수를 넘는 것으로 보인다. 따라서 플레이어 i는 g_i=1을 설정하고 속성(1)을 유지한다.

[0746] 플레이어 i는 반드시 v_i=v'로 설정 하는가? 플레이어 i가 v"에 대해 1/3 대다수를 넘는 것으로 보이도록 상이한 값 $v'' \neq \perp$이 존재한다고 가정하자. 이러한 메시지 중 일부는 악의적인 검증자로부터 온 것일 수 있지만, 그것들 중 적어도 하나는 정직한 검증자 i∈$HSV^{r,3}$으로부터 온 것이다. 실제로 $|HSV^{r,3}| > 2|MSV^{r,3}|$이고 i가 $HSV^{r,3}$으로부터 모든 메시지를 수신했기 때문에, i가 유효한 (r, 3)-메시지를 수신했던 악의적인 검증자의 세트는 그가 수신했던 모든 유효 메시지의 1/3에 대해 카운트한다.

[0747] 정의에 의해, 플레이어 j는 그가 받은 모든 유효한 (r, 2) 메시지 중에서 v"에 대해 2/3을 넘는 것으로 보여야 한다. 그러나 우리는 이미 일부 다른 정직한(r, 3)- 검증자가 v'에 대해 2/3의 대다수를 넘는 것으로 보이는 (v'에 서명했기 때문에) 것을 가진다. 보조 정리 5.5의 속성(d)에 의해, 이것은 일어나지 않을 수 있고 그러한 값 v"은 존재하지 않는다. 따라서 플레이어 i는 원하는 경우 v_i=v'를 설정해야 하며 속성(2)이 유지된다.

[0748] 마지막으로 일부 정직한(r, 3)-검증자가 v'에 대해 2/3 이상의 다수를 넘는 것으로 보이는 경우, v'에 대해 일부(실제로는 절반 이상)의 정직한 (r, 2)-검증자가 서명하고 그것들의 메시지를 전파한다. 프로토콜의 구성에 의해, 이들 정직한 (r, 2)-검증자는 일부 플레이어 $\ell \in SV^{r,1}$로부터 $v' = H(B_\ell^r)$로 유효한 메시지 $m_\ell^{r,1}$를 수신했었어야만 하고, 따라서, 속성(3)이 유지된다.

[0749] **BBA***. 우리는 다시 두 가지 경우를 구별한다.

[0750] 케이스 1. 모든 검증자 i∈$HSV^{r,4}$는 b_i=1이다.

[0751] 이것은 GC의 케이스 1에 따라 발생한다. $|MSV^{r,4}|<t_H$인 때, 이 경우, $SV^{r,5}$의 검증자는 비트 0에 대해 t_H 유효(r, 4)-메시지를 수집하거나 생성할 수 없다. 따라서 그는 비어 있지 않은 블록 B^r을 알고 있기 때문에 $HSV^{r,5}$에서의 정직한 검증자가 멈출 수 없다.

[0752] 더욱이, 비트 1에 대해 적어도 t_H 유효(r, 4)-메시지가 존재하지만, s'= 5는 s'-2≡1 mod 3을 만족하지 않으므로, 그가 $B^r = B_\epsilon^r$을 알기 때문에 중단할 $HSV^{r,5}$에서의 정직한 검증자는 없을 것이다.

[0753] 대신에, 모든 검증자 i∈$HSV^{r,5}$는 그가 $HSV^{r,4}$에 의해 전송된 모든 메시지를 수신할 때까지 보조 정리 5.5에 따라 시간 $\beta_i^{r,5} = \alpha_i^{r,5} + t_5$에서 동작한다. 따라서 플레이어 i는 1에 대해 2/3 대다수를 넘는 것으로 보이고, b_i=1로 설정한다.

[0754] 1에 대해 고정된 코인 단계인 단계 6에서, s'= 5가 s'-2≡0 mod 3을 만족시키더라도, 비트 0에 대해 t_H 유효(r, 4)-메시지가 존재하지 않고, 따라서 그는 비어있지 않은 블록 B^r을 알고 있기 때문에 중단할 $HSV^{r,6}$에서의 검증자는 없을 것이다. 그러나 s'=6, s'-2≡1 mod 3이고 $HSV^{r,5}$에서 비트 1에 대해 $|HSV^{r,5}| \geq t_H$ 유효 (r, 5) 메시지가 존재한다. 모든 검증자 i∈$HSV^{r,6}$에 대해, 보조 정리 5.5를 따르면서, $\alpha_i^{r,6} + t_6$ 시간에 또는 그 이전에, 플레이어 i는 $HSV^{r,5}$로부터 모든 메시지를 수신했고, 따라서 i는 아무것도 전파하지 않으면서 중단하고 $B^r = B_\epsilon^r$를 설정한다. 그의 $CERT^r$은 그가 중단했을 때 그에 의해 수신된 t_H 유효한(r, 5) 메시지 $m_j^{r,5} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,5})$의 세트이다.

[0755] 다음으로, 플레이어 i를 단계 s>6에서 정직한 검증자 또는 일반적인 정직한 사용자(즉, 비검증자)로 놓는다. 보조 정리 5.2의 증명과 유사하게, 플레이어 i는 $B^r = B_\epsilon^r$를 설정하고 자신의 $CERT^r$을 그가 수신한 t_H 유효 (r, 5) 메시지 $m_j^{r,5} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,5})$의 세트로 설정한다.

[0756] 마지막으로, 보조 정리 5.2와 유사하게,

[0757] $$T^{r+1} \leq \min_{i \in HSV^{r,6}} \alpha_i^{r,6} + t_6 \leq T^r + \lambda + t_6 = T^r + 10\lambda + \Lambda$$ 이고,

[0758] B^r를 알고 있는 제1 정직한 사용자 i가 자신의 $CERT^r$에서 (r, 5) 메시지를 전파하는 데 도움이 되었기 때문에 모든 정직한 사용자는 시간 간격 I^{r+1}에서 B^r을 안다.

[0759] 케이스 2. b_i=0인 검증자 $\hat{i} \in HSV^{r,4}$가 존재한다.

[0760] 이것은 GC의 케이스 2를 따라 발생하며 더 복잡한 경우이다. GC의 분석에 의해, 이 경우, 모든 i∈$HSV^{r,4}$에 대해 $v_i = H(B_\ell^r)$가 되도록 유효 메시지 $m_\ell^{r,1}$가 존재한다. $HSV^{r,4}$에 있는 검증자는 그들의 b_i에 대해 동의하지 않을 수 있다는 것에 유의하라.

[0761] 임의의 단계 s∈{5, ..., m+3} 및 검증자 i∈$HSV^{r,s}$에 대해, 보조 정리 5.5에 의해, 플레이어 i는 그가 시간 t_s를 대기하는 경우 $HSV^{r,4} \cup \cdots \cup HSV^{r,s-1}$에서의 모든 정직한 검증자가 보낸 모든 메시지를 수신했을 것이다.

[0762] 우리는 이제 다음 이벤트 E를 고려한다: 이진 BA에서 제1 시간동안 일부 플레이어 $i^* \in SV^{r,s^*}$ (악의적이든 정직하건)는 어떤 것도 전파하지 않고 중단해야 하도록 하는 단계 s*≥5가 있다. 우리는 플레이어 i^*가 악의적인 경우 프로토콜에 따라 그가 중단하지 않고 적의 선택의 메시지를 전달해야 한다고 그가 가장할 수 있다는 사실을 강조하기 위해 "중단해야 한다(should stop)"를 사용한다.

[0763] 또한, 프로토콜의 구성에 의해,

[0764] (E.a)i^*는 5≤s'≤s^* 및 s'-2≡0 mod 3을 가지고, 동일한 v와 s'에 대해 적어도 t_H 유효 메시지

$m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$를 수집 또는 생성할 수 있거나; 또는

[0765] (E.b)i^*는 $6 \le s' \le s^*$ 및 $s'-2 \equiv 1 \mod 3$을 가지고, 동일한 s'에 대해 적어도 t_H 유효 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$를 수집 또는 생성할 수 있다.

[0766] 정직한(r, s'-1)-메시지는 그것들이 단계 s'에서 대기하면서 수행되기 전에 모든 정직한(r, s')에 의해 수신되고, 적들은 정직한 사용자보다 늦지않게 모든 것을 수신하기 때문에, 일반성을 상실하지 않으면서 우리는 $s'=s^*$를 가지고 플레이어 i^*는 악의적이다. 우리는 E.a의 값 v가 유효 블록의 해시인 것을 요구하지 않았음에 유의하라: 분석에서 명확해지듯이, 이 하위 이벤트에서 $v = H(B_\ell^r)$이다.

[0767] 아래에서는 이벤트 E를 따르는 케이스 2를 먼저 분석한 다음, s^*의 값이 본질적으로 L^r에 따라 분산되어 있음을 보여준다(따라서 이벤트 E는 단계 m+3 이전에 파라미터에 대해 주어진 관계로 압도적인 확률로 발생한다). 우선, 모든 단계 $5 \le s < s^*$에 대해, 모든 정직한 검증자 $i \in HSV^{r,s}$는 시간 t_s를 기다렸고 v_i를 그가 수신한 유효한(r, s-1)-메시지의 다수 표가 되도록 설정한다. 플레이어 i는 보조 정리 5.5에 따라 모든 정직한(r, s-1) 메시지를 수신하고, $HSV^{r,4}$에서의 모든 정직한 검증자가 GC의 케이스 2에 따라 $H(B_\ell^r)$에 서명했으며, 각각의 s에 대해 $|HSV^{r,s-1}| > 2|MSV^{r,s-1}|$이므로, 유도에 의해 우리는 플레이어 i가 다음과 같이 설정하도록 한다

[0768] $v_i = H(B_\ell^r)$.

[0769] 아무것도 전파하지 않으면서 중단하지 않는 모든 정직 검증자 $i \in HSV^{r,s^*}$에게 똑같은 것이 적용된다. 이제 단계 s^*를 고려하고 4개의 하위 케이스를 구별한다.

[0770] 케이스 2.1.a. 이벤트 E.a가 발생하고, 아무것도 전파하지 않으면서 또한 중단해야 하는 정직한 검증자 $i' \in HSV^{r,s^*}$가 존재한다.

[0771] 이 경우, 우리는 $s^*-2 \equiv 0 \mod 3$을 가지며, 단계 s^*는 0에 고정된 코인 단계이다.

[0772] 정의에 따라, 플레이어 i'는 적어도 형태 $(ESIG_j(0), ESIG_j(v), \sigma_j^{r,s^*-1})$의 tH 유효 (r, s^*-1)를 받았다. HSV^{r,s^*-1}에서의 모든 검증자는 $H(B_\ell^r)$와 $|MSV^{r,s^*-1}| < t_H$에 서명했기 때문에, 우리는 $v = H(B_\ell^r)$이다.

[0773] 0과 v에 대해 i'에 의해 수신된(r, s^*-1)-메시지의 적어도 $t_H - |MSV^{r,s^*-1}| \ge 1$은 시간 $T^r + t_{s^*-1} \ge T^r + t_4 \ge T^r + \lambda + \Lambda \ge \beta_\ell^{r,1} + \Lambda$ 후에 HSV^{r,s^*-1}에서 검증자에 의해 전송되기 때문에, 플레이어 i'는 이들(r, s^*-1) 메시지를 수신할 때까지 $m_\ell^{r,1}$을 수신했다. 따라서 플레이어 i'는 아무것도 전파하지 않으면서 중단하고; $B^r = B_\ell^r$을 설정하고; 자신의 $CERT^r$을 자신이 수신한 0 및 v에 대한 유효한(r, s^*-1) 메시지 세트로 설정한다.

[0774] 다음으로, 우리는 다른 검증자 $i \in HSV^{r,s^*}$가 $B^r = B_\ell^r$로 중단되거나, 또는 b_i=0으로 설정하고

$(ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,s})$를 전파했다는 것을 보여준다. 실제로, 단계 s^*는 먼저 일부 검증자가 어떤 것도 전파하지 않고 중단해야 하기 때문에, $t_H(r, s'-1)$-검증자가 1을 서명하도록 $s'-2 \equiv 1 \bmod 3$인 단계 $s'<s^*$가 존재하지 않는다. 따라서, HSV^{r,s^*}에서 검증자가 $B^r = B_\ell^r$를 가지고 중단하지 않는다.

[0775] 또한 단계 {4, 5, ..., s^*-1}에 있는 모든 정직한 검증자가 $H(B_\ell^r)$에 서명했으므로, $s'-2 \equiv 0 \bmod 3$인 단계 $s' \leq s^*$가 존재하지 않아서, $t_H(r, s'-1)$-검증자가일부 $v'' \neq H(B_\ell^r)$, 실제로 $|MSV^{r,s'-1}| < t_H$에 서명하도록 한다. 따라서, HSV^{r,s^*}에서의 검증자는 $B^r \neq B_\epsilon^r$ 및 $B^r \neq B_\ell^r$로 중단하지 않는다. 즉, 만약 플레이어 $i \in HSV^{r,s^*}$가 어떤 것도 전파하지 않고 중단한다면, 그는 $B^r = B_\epsilon^r$로 설정해야만 한다. 플레이어 $i \in HSV^{r,s^*}$가 시간 t_{s^*}을 대기하고 시간 $\beta_i^{r,s^*} = \alpha_i^{r,s^*} + t_{s^*}$에 메시지를 전파한다면, 그는 0 및 v에 대해 그것들의 적어도 $t_H - |MSV^{r,s^*-1}|$를 포함하는 HSV^{r,s^*-1}로부터의 모든 메시지를 수신한다. i가 1에 대해 2/3 대다수 보다 큰 것으로 보이는 경우, 그는 정직한 (r, s^*-1)-검증자로부터 그것들 중 $2t_H - 3|MSV^{r,s^*-1}|$ 이상으로 1에 대해 $2(t_H - |MSV^{r,s^*-1}|)$ 이상의 유효한 (r, s^*-1) 메시지를 본다. 그러나, 이것은 다음과 같은

[0776] $$|HSV^{r,s^*-1}| + 4|MSV^{r,s^*-1}| < 2n$$

[0777] 라는 사실에 모순되면서, $|HSV^{r,s^*-1}| \geq t_H - |MSV^{r,s^*-1}| + 2t_H - 3|MSV^{r,s^*-1}| > 2n - 4|MSV^{r,s^*-1}|$를 의미하고, 이는 파라미터에 대한 관계에서 비롯된다. 따라서, i는 1에 대해> 2/3 다수를 보지 못하며, 단계 s^*가 0에 고정된 코인 단계이기 때문에 그는 $b_i=0$으로 설정한다. 우리가 보았듯이, $v_i = H(B_\ell^r)$이다. 따라서 i는 우리가 보여주고자 했던 것처럼 $(ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,s})$를 전파한다.

[0778] 단계 s^*+1의 경우, 플레이어 i'가 시간 $\alpha_{i'}^{r,s^*} + t_{s^*}$에 또는 그 이전에 자신의 CERT^{r}에서 메시지를 전파하는 데 도움이 되었기 때문에, HSV^{r,s^*+1}의 모든 정직한 검증자는 그들이 대기하면서 수행된 때 또는 그 이전에 비트 0 및 값 $H(B_\ell^r)$에 대해 적어도 t_H 유효한(r, s^*-1) 메시지를 수신했다. 또한 단계 s^*의 정의에 의해 $s'-2 \equiv 1 \bmod 3$ 및 $6 \leq s' \leq s^*+1$을 가지고 비트 1에 대해 임의의 다른 t_H 유효한$(r, s'-1)$ 메시지가 존재하지 않기 때문에, HSV^{r,s^*+1}의 검증자는 이들 (r, s^*-1) 메시지를 수신하기 전에 중단하지 않을 것이다. 특히, 단계 s^*+1 자체는 0에 고정된 코인 단계이지만, HSV^{r,s^*}에서의 정직한 검증자는 1 및 $|MSV^{r,s^*}| < t_H$에 대해 메시지를 전달하지 않았다. 따라서 HSV^{r,s^*+1}에서의 모든 정직한 검증자는 아무것도 전파하지 않으면서 중단하고 $B^r = B_\ell^r$을 이전처럼 설정하며; 그들은 원하는(r, s^*-1) 메시지를 수신하기 전에 $m_\ell^{r,1}$을 수신한다.[24] 미래의 단계의 모든 정직한 검증자와 전체적으로 모든 정직한 사용자에 대해 동일할 수 있다. 특히, 그것들은 모두 시간 간격 I^{r+1} 내의 $B^r = B_\ell^r$를 알고,

[0779] $T^{r+1} \le \alpha_{i'}^{r,s^*} + t_{s^*} \le T^r + \lambda + t_{s^*}$이다.

[0780] (24: ℓ이 악의적인 경우, 그는 정직한 사용자/검증자가 그것에 대해 원하는 인증서를 받을 때 일부 정직한 사용자/검증자가 $m_\ell^{r,1}$을 받지 못하길 희망하면서 $m_\ell^{r,1}$을 늦게 보낼 수 있다. 그러나, 검증자 $\hat{i} \in HSV^{r,4}$는 bi=0 및 $v_{\hat{i}} = H(B_\ell^r)$로 설정했기 때문에, 이전과 같이 우리가 정직한 검증자 $i \in HSV^{r,3}$의 절반 이상이 $v_i = H(B_\ell^r)$로 설정했다. 이는 정직한 검증자 $i \in HSV^{r,2}$의 절반 이상이 $v_i = H(B_\ell^r)$로 설정하고, 그것들 (r, 2)-검증자는 모두 $m_\ell^{r,1}$를 받았다는 것을 추가로 의미한다. 특히, 모든(r, 2)-검증자는 단계 2가 끝날 때 $m_\ell^{r,1}$을 전파하는 데 도움을 주었다. 그로부터, $m_\ell^{r,1}$가 나머지 정직한 사용자에게 도달하는 데 최대 Λ 시간이 걸린다. 모든 정직한 사용자가 서로 시간 λ 내에 자신의 2단계를 중단하고 마지막 정직한(r, 2)-검증자가 자신의 단계 2를 중단할 때부터 i'가 자신의 단계 s^*를 중단할 때까지 검증자 i'가 단계 s^*≥에서 중단하면서, 적어도 $t_4-t_2-\lambda=3\lambda$의 시간이 지났다. 검증자 v'이 중단한 후 λ 시간 내에 $\Lambda \le 4\lambda$일 때, 모든 정직한 사용자는 그것이 자신의 단계 2를 중단하기 위한 마지막 검증자인 (r, 2)-검증자에 의해 처음 전파된 경우에도 $m_\ell^{r,1}$을 수신했다. 실제로, 정직한(r, 2) 검증자의 절반 이상이 $m_\ell^{r,1}$을 전파하는 데 도움이 되었기 때문에 모든 정직한 사용자에게 도달하는 실제시간은 Λ보다 짧다.)

[0781] 케이스 2.1.b. 이벤트 E.b가 발생하고 아무것도 전파하지 않으면서 중단해야 하는 정직한 검증자 $i' \in HSV^{r,s^*}$가 존재한다.

[0782] 이 경우 우리는 $s^*-2 \equiv 1 \bmod 3$을 가지며 단계 s^*는 1에 고정된 코인 단계이다. 분석은 케이스 2.1.a와 유사하고, 많은 세부 사항은 생략되었다. 앞에서와 같이, 플레이어 i'는 적어도 형태 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s^*-1})$의 t_H 유효한(r, s^*-1) 메시지를 수신했어야 한다. 다시 s^*의 정의에 의해, $s'-2 \equiv 0 \bmod 3$을 갖는 $5 \le s' < s^*$의 단계가 존재하지 않고, 여기서 적어도 t_H(r, s'-1) 검증자는 0 및 동일한 v에 서명했다. 따라서, 플레이어 i'는 아무것도 전파하지 않으면서 중단하고; $B^r = B_\epsilon^r$를 설정하고; 자신의 $CERT^r$을 자신이 수신한 비트 1에 대해 유효한(r, s^*-1) 메시지 세트로 설정한다.

[0783] 더욱이, 다른 일부 검증자 $i \in HSV^{r,s^*}$는 $B^r = B_\epsilon^r$로 중단되었거나 b_i=1로 설정하고 $(ESIG_i(1), ESIG_i(v_i), \sigma_i^{r,s^*})$를 전파했다. 플레이어 i'가 시간 $\alpha_{i'}^{r,s^*} + t_{s^*}$만큼 자신의 $CERT^r$에서 (r, s^*-1) 메시지를 전파하는 것을 도왔기 때문에, 다시 HSV^{r,s^*+1}에서의 모든 정직한 검증자는 아무것도 전파하지 않으면서 중단하고 $B^r = B_\epsilon^r$로 설정한다. 유사하게, 모든 정직한 사용자들은 시간 간격 I^{r+1} 내에서 $B^r = B_\epsilon^r$를 알고,

[0785] $^*T^{r+1} \le \alpha_{i'}^{r,s^*} + t_{s^*} \le T^r + \lambda + t_{s^*}$이다.

[0786] 케이스 2.2.a. 이벤트 E.a는 발생하고 아무것도 전파하지 않으면서 중단해야 하는 정직한 검증자 $i' \in HSV^{r,s^*}$ 는 존재하지 않는다.

[0787] 이 경우 플레이어 i^*는 적들이 수집하거나 생성할 수 있는 t_H 원하는(r, s^*-1) 메시지로 구성된 유효한 $CERT^r_{i^*}$를 가질 수 있다. 그러나 악의적인 검증자가 이러한 메시지를 전달하는 것을 돕지 않을 수 있으므로, 정직한 사용자가 시간 λ에 메시지를 수신할 것이라고 결론을 내릴 수 없다. 사실, 그 메시지의 $|MSV^{r,s^*-1}|$은 메시지를 전혀 전파하지 않고 단계 s^*에서 악의적인 검증자에게만 그것들을 보낸 악의적인 (r, s^*-1)-검증자로부터 왔을 수 있다.

[0788] 케이스 2.1.a와 유사하게, 여기서 우리는 $s^*-2 \equiv 0 \bmod 3$을 가지고 단계 s^*는 0에 고정된 코인 단계이고, $CERT^r_{i^*}$의 (r, s^*-1)-메시지는 비트 0 및 $v = H(B^r_\ell)$에 대한 것이다. 실제로 모든 정직한(r, s^*-1)-검증자는 v에 서명하고, 따라서 적들은 다른 v'에 대해 t_H 유효(r, s^*-1) 메시지를 생성할 수 없다.

[0789] 더구나 모든 정직한(r, s^*)-검증자는 시간 t_{s^*}를 기다렸고 비트 1에 대해 2/3 이상의 다수를 보지 못했다. 왜냐하면 $|HSV^{r,s^*-1}| + 4|MSV^{r,s^*-1}| < 2n$이기 때문이다. 따라서 모든 정직한 검증자 $i \in HSV^{r,s^*}$는 다수결에 의해 b_i=0, $v_i = H(B^r_\ell)$로 설정하고 시간 $\alpha^{r,s^*}_i + t_{s^*}$에 $m^{r,s^*}_i = (ESIG_i(0), ESIG_i(H(B^r_\ell)), \sigma^{r,s^*}_i)$를 전파한다.

[0790] 이제 단계 s^*+1(1에 고정된 코인 단계)에서 정직한 검증자를 생각해보자. 적들이 실제로 $CERT^r_{i^*}$에서의 메시지를 그들 중 일부에게로 보내고 그들로 하여금 중단하게하는 경우, 케이스 2.1.a와 유사하게, 모든 정직한 사용자는 시간 간격 I^{r+1} 내의 $B^r = B^r_\ell$를 알고

[0791] $T^{r+1} \leq T^r + \lambda + t_{s^*+1}$이 된다.

[0792] 그렇지 않으면, 단계 s^*+1의 모든 정직한 검증자는 대기 시간 t_{s^*+1} 후에 HSV^{r,s^*}로부터 0과 $H(B^r_\ell)$에 대한 모든(r, s^*) 메시지를 수신하고, 이는 2/3 대다수 보다 크도록 한다. 왜냐하면 $|HSV^{r,s^*}| > 2|MSV^{r,s^*}|$이기 때문이다. 따라서 HSV^{r,s^*+1}에 있는 모든 검증자는 그에 따라 0과 $H(B^r_\ell)$에 대한 그들의 메시지를 전파한다. 단계 s^*+1은 0에 고정된 코인 단계가 아니므로 HSV^{r,s^*+1}에 있는 검증자는 $B^r = B^r_\ell$로 중단하지 않는다는 것에 유의하라.

[0793] 이제 단계 s^*+2(완전히 플링핑된 코인 단계)에 있는 정직한 검증자를 고려하자. 적들이 $CERT^r_{i^*}$에서의 메시지를 그들 중 일부에 전송하고 그들로 하여금 메시지를 중단시키는 경우, 다시 정직한 모든 사용자는 시간 간격 I^{r+1} 내에서 $B^r = B^r_\ell$를 알게되고

[0794] $T^{r+1} \leq T^r + \lambda + t_{s^*+2}$이다.

[0795] 그렇지 않으면, 단계 s^*+2의 모든 정직한 검증자는 2/3 대다수를 넘도록 하는 대기 시간 t_{s^*+2} 후에 0과

$H(B_\ell^r)$에 대해 HSV^{r,s^*+1}로부터의 모든(r, s^*+1) 메시지를 수신한다. 따라서 그것들 모두는 그에 따라 0과 $H(B_\ell^r)$에 대해 자신의 메시지를 전파한다. 즉, 이 경우에는 그것들은 "코인을 플립핑하지" 않는다. 다시, 단계 s^*+2는 0에 고정된 코인 단계가 아니기 때문에 그들은 전파하지 않고 중단하지 않는다는 것에 유의하라.

[0796] 마지막으로, 단계 s^*+3(0에 고정된 또 다른 코인 단계)에서 정직한 검증자들에 대해, 그들이 정말로 시간 t_{s^*+3}을 대기하는 경우, 그들 모두는 HSV^{s^*+2}로부터 0 및 $H(B_\ell^r)$에 대해 적어도 t_H 유효 메시지를 수신할 것이다. 따라서 적들이 그들 중 누구에게 $CERT_{i^*}^r$에서의 메시지를 전송했는지 여부와 관계없이 HSV^{r,s^*+3}에서의 모든 검증자는 어떤 것도 전파하지 않으면서 $B^r = B_\ell^r$로 중단한다. 적의 동작 방식에 따라, 그들 중 일부는 $CERT_{i^*}^r$에서의 그들의 (r, s^*-1)-메시지로 구성된 그들 자신의 $CERT^r$을 가질 수 있고, 다른 사람들은 그들의 (r, s^*+2)- 메시지로 구성된 그들 자신의 $CERT^r$을 가질 수 있다. 어떤 경우든, 모든 정직한 사용자는 시간 간격 I^{r+1} 내에서 $B^r = B_\ell^r$를 알고,

[0797] $T^{r+1} \le T^r + \lambda + t_{s^*+3}$이다.

[0798] 케이스 2.2.b. 이벤트 E.b가 발생하고 어떤 것도 전파하지 않으면서 또한 중단해야 하는 정직한 검증자 $i' \in HSV^{r,s^*}$가 존재하지 않는다.

[0799] 이 경우의 분석은 케이스 2.1.b 및 케이스 2.2.a의 분석과 유사하여, 많은 세부 사항이 생략되었다. 특히, $CERT_{i^*}^r$는 적들이 s*-2 ≡1 mod 3을 수집하거나 생성할 수 있는 비트 1에 대한 t_H 원하는(r, s^*-1) 메시지로 구성되며, 단계 s^*는 1에 고정된 코인 단계이고, 정직한 (r, s^*) 검증자들이 0에 대해 2/3의 대다수를 넘는 것으로 보이지 않을 수 있다.

[0800] 따라서, 모든 검증자 i∈HSV^{r,s^*}는 b_i=1로 설정하고, 시간 $\alpha_i^{r,s^*} + t_{s^*}$에서 m_i^{r,s^*}=($ESIG_i(1)$, $ESIG_i(v_i)$, σ_i^{r,s^*})를 전파한다. 케이스 2.2.a와 유사하게, 기껏해야 3 이상의 단계에서(즉, 프로토콜이 다른 1에 고정된 코인 단계인 단계 s_*+3에 도달함), 모든 정직한 사용자는 시간 간격 I^{r+1} 내의 $B^r = B_\ell^r$를 알게된다. 또한, T^{r+1}은 정직한 검증자가 전파하지 않고 중단할 수 있는 최초의 시간이 언제인가에 따라 $\le T^r + \lambda + t_{s^*+1}$, 또는 $\le T^r + \lambda + t_{s^*+2}$, 또는 $\le T^r + \lambda + t_{s^*+3}$일 수 있다.

[0801] 4개의 하위 케이스를 결합하여, 다음을 가지고 정직한 모든 사용자는 시간 간격 I^{r+1} 내에 B^r을 알고 있다.

[0802] 케이스 2.1.a 및 2.1.b에서 $T^{r+1} \le T^r + \lambda + t_{s^*}$, 및

[0803] 케이스 2.2.a 및 2.2.b에서 $T^{r+1} \le T^r + \lambda + t_{s^*+3}$.

[0804] 케이스 2의 경우 상한선 s^*에 남아 T^{r+1}을 유지하며, 프로토콜에서 전체적으로 플립핑되는 코인 단계가 실제로 실행된 횟수를 고려하여 이를 수행한다. 즉, 일부 정직한 검증자가 실제로 코인을 플립핑했다.

[0805] 특히 전체적으로 플립핑된 코인 단계 s'(즉, 7≤s'≤m+2 및 s'-2≡2 mod 3)를 임의로 수정(fix)하고

$\ell' \triangleq \arg\min_{j \in SV^{r,s'-1}} H(\sigma_j^{r,s'-1})$로 놓아라. 그렇지 않으면 정직한 검증자가 이전 논의에 따라 실제로 단계 s'에서 코인을 플립핑하지 못하기 때문에 우선 s'<s^*로 가정하라.

[0806] $SV^{r,s'-1}$의 정의에 의해, ℓ'의 보증서의 해시 값은 또한 PK^{r-k}의 모든 사용자 중에서 가장 작다. 해시 함수가 랜덤 오라클이기 때문에, 이상적으로 플레이어 ℓ'는 적어도 h 이상의 확률로 정직하다. 우리가 나중에 보여주겠지만, 비록 적들이 랜덤 오라클의 출력을 예측하고 확률을 기울이기 위해 최선을 다하더라도, 플레이어 ℓ'는 적어도 $p_h = h^2(1+h-h^2)$의 확률로 여전히 정직하다. 아래에서 우리는 그것이 실제로 일어 났을 때의 경우: 즉, $\ell' \in HSV^{r,s'-1}$를 고려한다.

[0807] 모든 정직한 검증자 $i \in HSV^{r,s'}$는 시간 $\alpha_i^{r,s'} + t_{s'}$까지 $HSV^{r,s'-1}$로부터 모든 메시지를 수신했다. 플레이어 i가 코인을 플립핑할 필요가 있다면(즉, 동일한 비트 b∈{0, 1}에 대해 그는 2/3 대다수를 넘지 않는다고 간주됨), 그는 $b_i = \mathtt{lsb}(H(\sigma_{\ell'}^{r,s'-1}))$로 설정한다. b∈{0, 1}에 대해 2/3 대다수를 넘는다고 간주되는 또다른 정직한 검증자 $i' \in HSV^{r,s'}$가 있다면, 보조 정리 5.5의 속성(d)에 의해, $HSV^{r,s'}$에서 정직한 검증자가 비트 b'≠b에 대해 2/3 대다수를 넘는다고 간주되지 않는다. 1/2의 확률로 $\mathtt{lsb}(H(\sigma_{\ell'}^{r,s'-1})) = b$이기 때문에, $HSV^{r,s'}$에서의 모든 정직한 검증자는 b에 대해 1/2의 확률로 합의에 도달한다. 물론, 만약 그러한 검증자 i'가 존재하지 않는다면, $HSV^{r,s'}$에서의 모든 정직한 검증자는 확률 1로 비트 $\mathtt{lsb}(H(\sigma_{\ell'}^{r,s'-1}))$에 합의한다.

[0808] $\ell' \in HSV^{r,s'-1}$에 대한 확률을 조합하면, $HSV^{r,s'}$에서의 정직한 검증자들은 적어도 $\frac{p_h}{2} = \frac{h^2(1+h-h^2)}{2}$의 확률로 b∈{0, 1}에 대한 합의에 도달한다. 또한, 이전과 같이 다수결 표결에 대한 유도에 의해, $HSV^{r,s'}$에서의 모든 정직한 검증자는 자신들의 v_i의 세트를 $H(B_\ell^r)$로 설정한다. 따라서, 단계 s'에서 b에 대한 합의가 이루어지면, T^{r+1}은 b=0 또는 b=1에 따라서, 케이스 2.1.a 및 케이스 2.1.b.의 분석에 따라

[0809] $\leq T^r + \lambda + t_{s'+1}$ 또는 $\leq T^r + \lambda + t_{s'+2}$이다.

[0810] 특히, 추가적인 전체적으로 플립핑되는 코인 단계는 실행되지 않을 것이다. 즉, 이러한 단계들의 검증자는 그들이 검증자이며 따라서 대기하지만, 그들이 아무것도 전파하지 않으면서 모든 것을 중단할 것인지를 여전히 체크한다. 따라서, 단계 s^* 이전에, 전체적으로 플립핑된 코인 단계가 실행되는 횟수는 랜덤 변수 L^r에 따라 분산된다. 단계 s'를 다음과 같은 프로토콜의 구성에 의해 L^r에 따른 마지막 전체적으로 플립핑된 코인 단계로 놓자,

[0811] $s'=4+3L^r$.

[0812] 적들이 T^{r+1}을 가능한 지연시키려면 적들은 단계 s^*를 언제 만들어야 하나? 우리는 적들도 L^r의 구현을 미리 알고 있다는 것을 가정할 수도 있다. s^*>s'이면 정직한 검증자가 이미 단계 s'에서 합의에 도달했기 때문에 쓸모가 없다. 확실히, 이 경우 s^*는 다시 b=0 또는 b=1인지에 따라 s'+1 또는 s'+2가 될 것이다. 그러나 이것은 실제로는 케이스 2.1.a와 2.1.b가 되며 결과적인 T^{r+1}은 그 경우와 완전히 동일하다. 더 정확하게,

[0813] $T^{r+1} \leq T^r + \lambda + t_{s^*} \leq T^r + \lambda + t_{s'+2}$ 이다.

[0814] $s^*<s'-3$, 즉, s*가 끝에서 두 번째의 전체적으로 플립핑된 코인 단계인 경우, 케이스 2.2.a 및 2.2.b 분석에 의해,

[0815] $T^{r+1} \leq T^r + \lambda + t_{s^*+3} < T^r + \lambda + t_{s'}$ 가 된다.

[0816] 즉, 적들은 실제로 B^r에 대해 합의하는 것이 더 빠르게 발생한다.

[0817] $s^*=s'-2$ 또는 $s'-1$인 경우, 즉, 단계 s' 직전인 0에 고정된 코인 단계 또는 1에 고정된 코인 단계는, 그런 다음 4개의 하위 케이스 분석에 의해, 단계 s'의 정직한 검증자가 더 이상 코인을 플립핑하지 않는데, 그들이 전파하지 않고 중단했거나 또는 동일한 비트에 대해 2/3 대다수를 넘는다고 간주되기 때문이다. 따라서 우리는

[0818] $T^{r+1} \leq T^r + \lambda + t_{s^*+3} \leq T^r + \lambda + t_{s'+2}$ 를 가진다.

[0819] 요약하면, s^*가 무엇이든 관계없이, 우리가 보여주고자 했던 것처럼,

$$
\begin{aligned}
& T^{r+1} \leq T^r + \lambda + t_{s'+2} = T^r + \lambda + t_{3L^r+6} \\
= \; & T^r + \lambda + (2(3L^r+6) - 3)\lambda + \Lambda \\
= \; & T^r + (6L^r + 10)\lambda + \Lambda,
\end{aligned}
$$

[0820] 이다.

[0821] 최악의 경우는 $s^*=s'-1$이고 케이스 2.2.b가 발생할 때이다. 바이너리 BA 프로토콜의 케이스 1과 2를 결합하면, 보조 정리 5.3이 성립한다. ■

[0822] **5.9 정직한 리더의 시드 Q^r 및 확률의 보안**

[0823] 보조 정리 5.4를 증명해야 한다. 라운드 r의 검증자는 PK^{r-k}에서 취해지고 Q^{r-1}의 양에 따라 선택된다는 것을 상기하라. 룩백 파라미터 k를 도입한 이유는 다시 라운드 r-k에서, 적들이 새로운 악의적인 사용자를 PK^{r-k}에 추가할 수 있을 때, 그는 무시할 수 있는 확률을 제외하고는 Q^{r-1}의 양을 예측할 수 없는 것을 보장하기 때문이다. 해시 함수는 랜덤 오라클이고 라운드 r에 대한 검증자를 선택할 때 Q^{r-1}은 그 입력 중 하나이다. 따라서, 악의적인 사용자가 PK^{r-k}에 어떻게 추가되더라도, 적들의 관점에서, 그들 중 각각은 라운드 r의 단계에서 필요한 확률 p(또는 단계 1에 대해서는 p')를 가지고 검증자로 여전히 선택된다. 보다 정확하게, 우리는 다음의 보조 정리를 갖는다.

[0824] **보조 정리 5.6.** $k = O(\log_{1/2} F)$를 가지고, 각 라운드 r에 대해, 압도적인 확률로, 적들은 라운드 r-k에서 랜덤 오라클에 대해 다시 Q^{r-1}을 쿼리하지 않았다.

[0825] 증명. 유도로 진행한다. 각 라운드 $\gamma < r$에 대해, 적들은 라운드 $\gamma - k$에서 랜덤 오라클에 $Q^{\gamma-1}$을 쿼리하지 않는다고 가정한다.[25] Q^{r-1}을 예측하려고 하는 라운드 r-k에서 적들에 의해 플레이되는 다음의 멘탈 게임을 고려하라.

[0826] (25: k는 작은 정수이기 때문에, 일반성 손실 없이 프로토콜의 제1 k 라운드가 안전한 환경에서 실행되고 유도 가설이 해당 라운드에 대해 수행된다고 가정할 수 있다.)

[0827] 각 라운드, $\gamma = r-k, \ldots, r-1$의 단계 1에서, 랜덤 오라클에 대해 쿼리되지 않은 특정 $Q^{\gamma-1}$이 주어지면, 해시 값 $H(SIG_i(\gamma, 1, Q^{\gamma-1}))$에 따라 플레이어들 $i \in PK^{\gamma-k}$를 증분하여 순서화함으

로써, 우리는 $PK^{\gamma-k}$에 대한 랜덤 순열(permutation)을 획득한다. 정의에 의해 리더 ℓ^{γ}는 순열에서의 제1 사용자이고 확률 h로 정직하다. 또한, $PK^{\gamma-k}$가 충분히 클 때, 임의의 정수 x≥1에 대해서, 순열의 제1 x 사용자가 모두 악의적이지만 (x+1)번째는 정직하다는 확률은 $(1-h)^{x}h$이다.

[0828] ℓ^{γ}이 정직하다면, $Q^{\gamma} = H(SIG_{\ell^{\gamma}}(Q^{\gamma-1}), \gamma)$이다. 적들이 ℓ^{γ}의 서명을 위조할 수 없기 때문에, Q^{γ}은 적들의 관점에서 랜덤하게 균등하게 분산되고, 기하급수적으로 작은 확률[26]을 제외하고는, 라운드 r-k에서 H에게 쿼리되지 않았다. 각각의 $Q^{\gamma+1}, Q^{\gamma+2}, \ldots, Q^{r-1}$은 각각 입력의 하나로서 $Q^{\gamma}, Q^{\gamma+1}, \ldots, Q^{r-2}$를 가진 H의 출력이기 때문에, 그것들은 모두 적에게 랜덤하게 보이며 적들은 라운드 r-k에서 H로의 쿼리된 Q^{r-1}을 가질 수 없다.

[0829] (26: 즉, H의 출력 길이에서 지수적이다. 이 확률은 F보다 작다는 것에 유의하라.)

[0830] 따라서 적들이 라운드 r-k에서 좋은 확률로 Q^{r-1}를 예측할 수 있는 유일한 경우는 모든 리더 $\ell^{r-k}, \ldots, \ell^{r-1}$이 악의적인 경우이다. 다시 한번 라운드 $\gamma \in \{r-k \ldots, r-1\}$ 및 해당 해시 값에 의해 유도된 $PK^{\gamma-k}$에 대한 랜덤 순열을 고려하라. 일부 x≥2인 경우, 순열의 제1 x-1 사용자가 모두 악의적이고 x번째가 정직한 경우, 적들은 x를, 플레이어 i를 라운드 γ의 실제 리더로 만듦으로써 형태 $H(SIG_i(Q^{\gamma-1}, \gamma))$(여기서 i는 제1 x-1 악의적 사용자); 또는 $B^{\gamma} = B^{\gamma}_{1}$를 강제함으로써 $H(Q^{\gamma-1}, \gamma)$ 중 어느 하나의 Q^{γ}에 대해 가능한 선택으로 만든다. 그렇지 않으면 라운드 γ의 리더가 순열에서 제1의 정직한 사용자가 되고 Q^{r-1}은 적에게 예측할 수 없게된다.

[0831] 위의 Q^{γ}의 x 옵션들 중 어느 것이 적들이 추구하는 것인가? 적들이 이 질문에 답하는 것을 돕기 위해, 멘탈 게임에서 하기와 같이 우리는 실제로 그를 실제보다 더 강력하게 만든다. 우선, 실제로, 적들은 정직한 사용자의 서명의 해시를 계산할 수 없으므로, 각 Q^{γ}에 대해 Q^{γ}에 의해 유도된 $\gamma+1$에서의 랜덤 순열의 시작 부분에 있는 악의적인 사용자의 수 $x(Q^{\gamma})$를 결정할 수 없다. 멘탈 게임에서, 우리는 그에게 수 $x(Q^{\gamma})$를 무료로 준다. 둘째로, 실제로, 그들의 서명의 해시 값도 p_1보다 작아야 하기 때문에, 순열에서 제1 x 사용자를 모두 악의적인 것으로 한다고 해서 반드시 그들이 모두 리더로 될 수 있다는 것을 의미하지는 않는다. 우리는 멘탈 게임에서 이 제약을 무시하여 적들이 더욱 많은 이점을 갖도록 했다.

[0832] 멘탈 게임에서 Q^{γ}로 표시된 적의 최적 옵션은 라운드 $\gamma+1$에서 랜덤 순열의 시작 부분에서 가장 긴 악의적 사용자 시퀀스를 생성하는 것으로 보는 것은 쉬운 일이다. 실제로, 특정 Q^{γ}이 주어지면 프로토콜은 $Q^{\gamma-1}$에 더 이상 의존하지 않으며 적들은 처음에 악의적인 사용자 수에 대해 동일한 분포를 갖는 라운드 $\gamma+1$에서 새로운 순열에 전적으로 집중할 수 있다. 따라서 각 라운드 γ에서, 위에서 언급한 $\hat{Q}^{\gamma}$은 그에게 $Q^{\gamma+1}$에 대해 가장 큰 수의 옵션을 제공하고 따라서 연속 리더가 모두 악의적인 확률을 최대화한다.

[0833] 따라서, 멘탈 게임에서 적들은 마크로프 체인을 라운드 r-k에서 라운드 r-1로 진행하며, 상태 공간은 $\{0\} \cup \{x : x \geq 2\}$이다. 상태 0은 현재 라운드 γ에서의 랜덤 순열에서 제1 사용자가 정직하다는 사실을

나타내므로, 적들은 Q^{r-1}을 예측하는 데에 실패하고; 각 상태 x≥2는 순열의 제1 x-1 사용자가 악의적이고 x번째가 정직하다는 사실을 나타내어, 적들은 Q^{γ}에 대해 x 옵션을 가진다. 트랜지션 확률 P(x, y)는 다음과 같다.

[0834] · 임의의 y≥2에 대해 P(0, 0)=1 및 P(0, y)=0이다. 즉, 적은 순열의 제1 사용자가 정직하면 게임에서 실패한다.

[0835] · 임의의 x≥2에 대해 $P(x, 0)=h^x$이다. 즉, 확률 h^x로, 모든 x 개의 랜덤 순열이 자신들의 제1 사용자를 정직하게 처리하므로, 다음 라운드에서 적들은 게임에서 실패한다.

[0836] · 임의의 x≥2 및 y≥2에 대해 P(x, y)는 Q^{γ}의 x 옵션에 의해 유도된 x 개의 랜덤 순열 중에, 그것들 중 일부의 시작에서 악의적인 사용자들의 가장 긴 시퀀스가 y-1이고, 따라서 적들은 다음 라운드에서 $Q^{\gamma+1}$에 대한 y옵션을 가진다. 즉, 다음과 같다

[0837] $$P(x,y)=\left(\sum_{i=0}^{y-1}(1-h)^i h\right)^x-\left(\sum_{i=0}^{y-2}(1-h)^i h\right)^x=(1-(1-h)^y)^x-(1-(1-h)^{y-1})^x$$.

[0838] 상태 0은 트랜지션 행렬 P의 고유한 흡수 상태이고, 모든 다른 상태 x는 0으로 가는 양의 확률을 가짐을 유의하라. 우리는 압도적인 확률로 마르코프 체인이 0으로 수렴하는 데 필요한 라운드의 수 k를 상한에 두는데에 관심이 있다. 즉, 압도적인 확률로 체인이 어디에서 시작하든지 상관없이, 적들은 게임에 지고 라운드 r-k에서 Q^{r-1}을 예측하는 데에 실패한다.

[0839] 두 라운드 이후의 트랜지션 행렬 $P^{(2)} \triangleq P \cdot P$를 고려하라. $P^{(2)}(0, 0)=1$이고 임의의 x≥2에 대해 $P^{(2)}(0, x0)=0$이다. 임의의 x≥2 및 y≥2에 대해, P(0, y)=0이고,

[0840] $$P^{(2)}(x,y)=P(x,0)P(0,y)+\sum_{z\geq 2}P(x,z)P(z,y)=\sum_{z\geq 2}P(x,z)P(z,y)$$ 이다.

[0841] $\bar{h} \triangleq 1-h$로 하면,

[0842] $$P(x,y)=(1-\bar{h}^y)^x-(1-\bar{h}^{y-1})^x$$ 및

[0843] $$P^{(2)}(x,y)=\sum_{z\geq 2}[(1-\bar{h}^z)^x-(1-\bar{h}^{z-1})^x][(1-\bar{h}^y)^z-(1-\bar{h}^{y-1})^z]$$ 이다.

[0844] 아래에서 우리는 h가 1로 갈 때, 즉, $\bar{h}$가 0으로 접근할 때 $\frac{P^{(2)}(x,y)}{P(x,y)}$의 극한(limit)을 계산한다. P(x, y)에서 $\bar{h}$의 가장 높은 차수는 계수 x와 함께 $\bar{h}^{y-1}$이다. 따라서,

[0845] $$\begin{aligned}&\lim_{\bar{h}\to 1}\frac{P^{(2)}(x,y)}{P(x,y)}=\lim_{\bar{h}\to 0}\frac{P^{(2)}(x,y)}{P(x,y)}=\lim_{\bar{h}\to 0}\frac{P^{(2)}(x,y)}{x\bar{h}^{y-1}+O(\bar{h}^y)}\\ =&\lim_{\bar{h}\to 0}\frac{\sum_{z\geq 2}[x\bar{h}^{z-1}+O(\bar{h}^z)][z\bar{h}^{y-1}+O(\bar{h}^y)]}{x\bar{h}^{y-1}+O(\bar{h}^y)}=\lim_{\bar{h}\to 0}\frac{2x\bar{h}^y+O(\bar{h}^{y+1})}{x\bar{h}^{y-1}+O(\bar{h}^y)}\\ =&\lim_{\bar{h}\to 0}\frac{2x\bar{h}^y}{x\bar{h}^{y-1}}=\lim_{\bar{h}\to 0}2\bar{h}=0.\end{aligned}$$

[0846] h가 1에 충분히 접근할 때,[27] 임의의 x≥2 및 y≥2에 대해,

[0847] (27: 예를 들어, 파라미터의 특정 선택 항목에서 제안될 때 h=80%이다.)

[0848] $$\frac{P^{(2)}(x,y)}{P(x,y)} \leq \frac{1}{2}$$ 이다.

[0849] 유도에 의해, 임의의 k> 2에 대해, $P^{(k)} \triangleq P^k$ 하여 하기와 같이

[0850] · 임의의 x≥2에 대해 $P^{(k)}(0,0) = 1,\ P^{(k)}(0,x) = 0$

[0851] · 임의의 x≥2 및 y≥2에 대해,

[0852] $$P^{(k)}(x,y) = P^{(k-1)}(x,0)P(0,y) + \sum_{z\geq 2} P^{(k-1)}(x,z)P(z,y) = \sum_{z\geq 2} P^{(k-1)}(x,z)P(z,y)$$

[0853] $$\leq \sum_{z\geq 2} \frac{P(x,z)}{2^{k-2}} \cdot P(z,y) = \frac{P^{(2)}(x,y)}{2^{k-2}} \leq \frac{P(x,y)}{2^{k-1}}$$

[0854] 가 되도록 한다.

[0855] P(x, y)≤1이면, 1-$\log_2$F 라운드 후에, 임의의 상태 x≥2로 시작하는, 임의의 상태 y≥2로의 트랜지션 확률은 무시할 수 있다. 이러한 상태 y가 많지만 다음과 같이 쉽게 이해할 수 있다

[0856] $$\lim_{y\to+\infty} \frac{P(x,y)}{P(x,y+1)} = \lim_{y\to+\infty} \frac{(1-\bar{h}^y)^x - (1-\bar{h}^{y-1})^x}{(1-\bar{h}^{y+1})^x - (1-\bar{h}^y)^x} = \lim_{y\to+\infty} \frac{\bar{h}^{y-1} - \bar{h}^y}{\bar{h}^y - \bar{h}^{y+1}} = \frac{1}{\bar{h}} = \frac{1}{1-h}.$$

[0857] 따라서, 트랜지션 행렬 P의 각 행(row) x는 y가 충분히 클 때 비율 $\frac{1}{1-h} > 2$ 인 기하학적 시퀀스로서 감소하고, $P^{(k)}$에 대해 동일하게 유지된다. 따라서, k가 충분히 크지만 여전히 임의의 x≥2에 대해 $\log_{1/2} F,\ \sum_{y\geq 2} P^{(k)}(x,y) < F$ 의 오더로 있다.

[0858] 즉, 압도적인 확률로 적들은 게임에 지고 라운드 r-k에서 Q^{r-1}을 예측하지 못한다. h∈(2/3, 1)에 대해, 더 복잡한 분석은 1/2보다 약간 큰 상수 C가 존재하여 $k = O(\log_C F)$를 충분히 취할 수 있다는 것을 보여준다. 따라서 보조 정리 5.6이 성립한다. ■

[0859] **보조 정리 5.4.**(재진술) r 전의 각 라운드마다 속성 1-3이 주어지면, L^r에 대해 $p_h = h^2(1+h-h^2)$이고, 리더 ℓ는 적어도 p_h의 확률로 정직하다.

[0860] 증명. 보조 정리 5.6에 따르면, 적들은 무시할 수 있는 확률을 제외하고는 라운드 r-k에서 Q^{r-1}을 다시 예측할 수 없다. 이것은 정직한 리더의 확률이 각 라운드마다 h일 확률을 의미하지는 않는다는 것에 유의하라. 실제로, Q^{r-1}이 주어진다면, 얼마나 많은 악의적인 사용자기 PK^{r-k}의 랜덤 순열의 시작에 있는지에 따라, 적들은 Q^r에 대해 하나 이상의 옵션을 가질 수 있고, 따라서 라운드 r+1에서 악의적인 리더의 확률을 높일 수 있다. 다시 말해서 분석을 간략화하기 위해 보조 정리 5.6에서와같이 그에게 일부 비현실적인 이점을 주고있다.

[0861] 그러나 라운드 r-k에서 적에 의해 H에 대해 쿼리되지 않은 각 Q^{r-1}에 대해, 임의의 x≥1에 대해 확률 $(1-h)^{x-1}h$로, 제1 정직한 사용자가 PK^{r-k}의 결과적인 랜덤 순열에서의 위치 x에서 발생한다. x=1일 때, 라운드 r+1에서 정직한 리더의 확률은 실제로 h이고; x = 2일 때, 적들은 Q^r에 대한 두 가지 옵션을 가지며 결과적인 확률은 h^2이다. 이들 2개의 경우만 고려하면, 라운드 r+1에서 정직한 리더의 확률은 원하는 바와 같이 적어도 $h\cdot h + (1-h)h\cdot h^2 = h^2(1+h-h^2)$ 이다.

[0862] 위의 확률은 라운드 r-k에서 라운드 r까지의 프로토콜의 무작위성만을 고려한다는 것에 유의하라. 라운드 0에서 라운드 r까지의 모든 무작위성이 고려될 때, Q^{r-1}은 적에게 보다 덜 예측 가능하며 라운드 r+1에서의 정직한 리더의 확률은 적어도 $h^2(1+h-h^2)$이다. r+1을 r로 대체하고 모든 것을 1라운드 뒤로 이동시키면, 리더 ℓ^r는 원하는 대로 적어도 $h^2(1+h-h^2)$ 확률로 정직하다.

[0863] 유사하게, 각각의 전체적으로 플립핑된 코인 단계 s에서, 그의 보증서가 가장 작은 해시 값을 갖는 $SV^{r,s}$의 검증자인 해당 단계의 "리더"는 적어도 $h^2(1+h-h^2)$ 확률로 정직하다. 따라서 L^r에 대해 $p_H=h^2(1+h-h^2)$이고 보조 정리 5.4는 유지된다.

[0864] ## 6 $Algorand'_2$

[0865] 이 섹션에서는 다음과 같은 가정하에 작동하는 $Algorand'$ 버전을 구축한다.

[0866] 대다수의 정직한 사용자 가정: 각 PK^r의 사용자 중 2/3 이상이 정직하다.

[0867] 8절에서는 원하는 대다수의 정직한 사용자 가정으로 위의 가정을 대체하는 방법을 보여준다.

[0868] ### 6.1 $Algorand'_2$에 대한 추가 표기 및 파라미터

[0869] 표기

[0870] · $\mu \in \mathbb{Z}^+$: 압도적인 확률로 한 라운드에서 실제로 취할 수 있는 단계 수에 대한 실용적인 상한. (우리가 볼 수 있듯이 파라미터 μ는 사용자가 각 라운드마다 미리 준비하는 임시 키의 수를 제어한다.)

[0871] · L^r: 각각의 시도가 확률 $\frac{p_h}{2}$로 1일 때 1을 보기 위해 필요한 베르누이 시도의 수를 나타내는 랜덤 변수. L^r은 블록 B^r을 생성하는 데 필요한 시간의 상한을 구하는 데 사용될 것이다.

[0872] · t_H: 라운드 r의 단계 s>1에서 정직한 검증자의 수에 대한 하한값으로, (n과 p로 주어진) 압도적인 확률로 $SV^{r,s}$에서의 t_H보다 큰 정직한 검증자가 존재한다.

[0873] **파라미터**

[0874] · 다양한 파라미터 간의 관계.

[0875] - 라운드 r의 각 단계 s>1에 대해 n이 선택되어 압도적인 확률로,

[0876] $|HSV^{r,s}| > t_H$ 및 $|HSV^{r,s}| + 2|MSV^{r,s}| < 2t_H$가 되도록 한다.

[0877] 위의 두 가지 부등식은 함께 $|HSV^{r,s}| > 2|MSV^{r,s}|$, 즉, 선택된 검증자 중에서 2/3의 정직한 다수가 있다는 것을 의미한다는 것에 유의하라.

[0878] h의 값이 1에 근접할수록 n은 더 작을 필요가 있다. 특히, 우리는 압도적인 확률로 원하는 조건을 유지하는 것을 보장하기 위해 체르노프 범위(의 변형)를 사용한다.

[0879] · 중요한 파라미터의 특정 선택.

[0880] - $F = 10^{-18}$.

[0881] - $n \approx 4000,\ t_H \approx 0.69n,\ k = 70$.

[0882] ### 6.2 $Algorand'_2$에서 임시 키 구현하기

[0883] 검증자 $i \in SV^{r,s}$가 그가 사용 후 즉시 파괴하는 임시 비밀 키 $sk_i^{r,s}$를 이용하여 임시 공개 키 $pk_i^{r,s}$에 대해 라운드 r에서 단계 s의 자신의 메시지 $m_i^{r,s}$을 디지털 서명한다는 것을 상기하라. 라운드가 취할 수 있는 가능한 단계의 수가 주어진 정수 μ에 의해 제한될 때, 우리는 이미 임시 키를 실제로 처리하는 방법을 이해했다. 예를 들어, $Algorand'_1$(여기서 μ=m+3)에서 설명했듯이, 라운드 r'에서 라운드 r'+10^6까지 자신의 가능한 모든 임시 키를 처리하기 위해, i는 쌍(PMK, SMK)을 생성하고, 여기서 PMK는 신원 기반 서명 체계의 공개 마스터 키이고 및 SMK는 그의 대응하는 비밀 마스터 키이다. 사용자 i는 PMK를 공개하고 SMK를 사용하여 각각의 가능한 임시 공개키의 비밀 키를 생성한다(그렇게한 후에 SMK를 파괴한다). 관련 라운드에 대한 i의 임시 공개키 세트는 S={i}x{r', ..., r'+10^6}x{1, ..., μ}이다. (논의된 바와 같이, 라운드 r'+10^6이 접근함에 따라, i는 자신의 쌍(PMK, SMK)을 "새로 고침" 한다.)

[0884] 실제로, μ가 충분히 크다면, $Algorand'_2$의 라운드는 μ이상의 단계를 취하지 않을 것이다. 그러나 이론적으로 일부 라운드 r에 대해서 실제로 취해진 단계의 수는 μ를 초과할 가능성이 있다. 이런 일이 발생하면, 라운드 r에 대해서 μ의 비밀 키만 미리 준비했으므로, i는 임의의 단계 s>μ에 대해 자신의 메시지 $m_i^{r,s}$에 사인할 수 없을 것이다. 더욱이, 그는 이전에 논의된 것처럼 새로운 은닉한 임시 키를 준비하고 공개할 수 없었다. 사실, 그렇게 하려면 그는 새로운 공개 마스터키인 PMK'를 새 블록에 삽입해야 한다. 그러나 라운드 r이 점점 더 많은 단계를 밟아야한다면 새로운 블록이 생성되지 않는다.

[0885] 그러나 해결책이 존재한다. 예를 들어, i는 다음과 같이 라운드 r의 마지막 임시 키, $pk_i^{r,\mu}$를 사용할 수 있다. 그는 (1) 다른 마스터 키 쌍 $(\overline{PMK}, \overline{SMK})$을 생성하고; (2) 이 쌍을 사용하여 예를 들어, 라운드 r의 단계들 $\mu+1, ..., \mu+10^6$에 또 다른 대응하는 10^6개의 임시 키 $\overline{sk}_i^{r,\mu+1}, \ldots, \overline{sk}_i^{r,\mu+10^6}$를 생성하고; (3) $pk_i^{r,\mu}$에 대해 $\overline{PMK}$(및, $i \in SV^{r,\mu}$인 경우 임의의 (r, μ) 메시지)에 디지털 서명하도록 $sk_i^{r,\mu}$를 사용하고; 및 (4) $\overline{SMK}$ 및 $sk_i^{r,\mu}$를 소거;함으로써 라운드 r에 대한 또 다른 은닉한 키의 쌍을 생성한다. i가 s∈{1,..., 10^6}를 가진 단계 μ+s에서의 검증자가 되고, i가 자신의 새로운 키 $\overline{pk}_i^{r,\mu+s} = (i, r, \mu+s)$에 대해 자신의 (r, μ+s) 메시지 $m_i^{r,\mu+s}$에 디지털 서명한다. 물론, i의 이 서명을 검증하기 위해, 다른 사람들은 이 공개키가 i의 새로운 공개 마스터키 $\overline{PMK}$에 대응한다는 것을 확신해야 한다. 따라서 이 서명 외에도, i는 $pk_i^{r,\mu}$에 대해 $\overline{PMK}$의 자신의 디지털 서명을 전송한다.

[0886] 물론, 이 접근법은 필요한 만큼 다수 반복될 수 있으므로 라운드 r에 대해 점점 더 많은 단계를 반복해야 한다! 마지막 임시 비밀 키는 새로운 마스터 공개키, 및 그에 따른 라운드 r에 대한 또 다른 은닉한 임시 키를 인증하는 데 사용된다. 기타 등등.

[0887] ### 6.3 실제프로토콜 $Algorand'_2$

[0888] 라운드 r의 각 단계에서, 검증자 $i \in SV^{r,s}$는 케이스 s=1에서 $SIG_i(r, s, Q^{r-1})$ 뿐만 아니라 자신의 보증서 $\sigma_i^{r,s} \triangleq SIG_i(r, s, Q^{r-1})$을 산출하기 위해 자신의 장기 공개 비밀 키 쌍을 사용한다는 것을 다시 상기하자. 검증자 i는 자신의 임시 키 쌍 $(pk_i^{r,s}, sk_i^{r,s})$을 사용하여 요구될 수 있는 다른 메시지 m에 서명한다. 간단히 하기 위해, 이 단계에서 i의 적절한 임시 서명 m을 나타내기 위해 $sig_{pk_i^{r,s}}(m)$ 대신에 $esig_i(m)$를 기

록하고, $SIG_{pk_i^{r,s}}(m) \triangleq (i, m, esig_i(m))$ 대신에 $ESIG_i(m)$)를 기록한다.

[0889] 단계 1: 블록 제안

[0890] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 그가 $CERT^{r-1}$을 가지는 즉시 사용자 i가 라운드 r의 자신의 제1 단계를 시작하여 i로 하여금 명확하게 $H(B^{r-1})$ 및 Q^{r-1}을 계산하도록 할 수 있다.

[0891] · 사용자 i는 $i \in SV^{r,1}$인지 여부를 체크하기 위해 Q^{r-1}을 사용한다. $i \notin SV^{r,1}$이면 그는 1 단계에서는 아무것도 하지 않는다.

[0892] · 만약 $i \in SV^{r,1}$라면, 즉 i가 잠재적인 리더라면, 그는 다음을 수행한다.

[0893] (a) i가 그 자신에게 B^0, ..., B^{r-1}이 보여진다면(임의의 $B^j = B^j_\epsilon$가 $CERT^j$에서의 자신의 해시 값으로부터 용이하게 도출될 수 있고 따라서 '보인다'고 가정되면) 그는 지금까지 그에게 전파된 라운드 r 결제를 수집하고 그것들로부터 최대 페이셋 PAY_i^r을 연산한다.

[0894] (b) 만약 i가 모든 B^0, ..., B^{r-1}이 아직 보이지 않으면, 그는 $PAY_i^r = \emptyset$를 설정한다.

[0895] (c) 다음으로, i는 자신의 "후보 블록" $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$을 계산한다.

[0896] (c) 마지막으로, i는 메시지 $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$을 계산하고, 그의 임시 비밀 키 $sk_i^{r,1}$을 파괴하고, 그런 다음 두 메시지 $m_i^{r,1}$와 $(SIG_i(Q^{r-1}), \sigma_i^{r,1})$을 개별적으로 그러나 동시에 전파한다.[28]

[0897] (28: i가 리더일 때, $SIG_i(Q^{r-1})$은 다른 사람들이 $Q^r = H(SIG_i(Q^{r-1}), r)$을 계산하게 한다.)

[0898] 선택적 전파

[0899] 제1 단계의 전역 실행과 전체 라운드를 단축하려면, (r, 1)-메시지가 선택적으로 전파되는 것이 중요하다. 즉, 시스템의 모든 사용자 j에 대해, 다음과 같다

[0900] · 블록이 포함되어 있는지 또는 보증서와 Q^{r-1}의 서명인지 여부와 같이, 그가 수신하고 성공적으로 검증한[29] 제1 (r, 1) 메시지에 대해, 플레이어 j는 그것을 평소와 같이 전파한다.

[0901] (29: 즉, 모든 서명은 올바르고, 그것이 $m_i^{r,1}$의 형태인 경우, 포함된 페이셋이 i에 대해 최대인지 여부를 j가 확인하지 않더라도 블록 및 그의 해시는 모두 유효하다.)

[0902] · 플레이어 j가 수신하고 성공적으로 검증한 다른 모든(r, 1) 메시지에 대해, 그는 그것이 포함하는 보증서의 해시 값이 그가 지금까지 수신하고 성공적으로 검증한 모든(r, 1) 메시지에 포함된 보증서의 해시 값 중 가장 작은 경우에만 그것을 전파한다.

[0903] · 그러나 j가 동일한 플레이어 i로부터 $m_i^{r,1}$ 형식의 두 개의 상이한 메시지를 받는 경우,[30] 그는 i의 보증서가 어떤 해시 값인지 상관없이 두 번째 메시지를 버린다.

[0904] (30: i가 악의적이라는 뜻.)

[0905] 선택적 전파하에서 각각의 잠재 리더 i가 $m_i^{r,1}$로부터 개별적으로 자신의 보증서 $\sigma_i^{r,1}$을 전파하고;[31] 이들 작은

메시지들은 블록보다 빠르게 이동하고, 더 큰 해시 값을 가진 그것들을 빠르게 사라지게 하면서, 포함된 보증서가 작은 해시값을 갖는 $m_i^{r,1}$ 의 시기가 적절한 전파를 보장한다는 것에 유의하라.

[0906] (31: 이것을 제안한 Georgios Vlachos에게 감사한다.)

[0907] 단계 2: 단계적 합의 프로토콜 GC의 제1 단계

[0908] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 그가 $CERT^{r-1}$을 가지는 즉시 라운드 r의 자신의 제2 단계를 시작한다.

[0909] · 사용자 i는 최대 시간 $t_2 \triangleq \lambda + \Lambda$를 대기한다. 대기하는 동안 i는 다음과 같이 동작한다.

[0910] 1. 시간 2λ를 기다린 후, 그는 자신이 지금까지 수신한 성공적으로 검증된 (r, 1) 메시지의 일부인 모든 보증서 $\sigma_j^{r,1}$ 에 대해 $H(\sigma_\ell^{r,1}) \leq H(\sigma_j^{r,1})$ 가 되도록 ℓ 을 찾는다.[32]

[0911] (32: 본질적으로, 사용자 i는 사적으로 라운드 r의 리더가 사용자 ℓ 임을 결정한다.)

[0912] 2. 만약 그가 $CERT^{r-1}$에 포함된 해시 값 $H(B^{r-1})$에 매칭하는[33] 블록 B^{r-1}을 수신하면, 그리고 그가 ℓ로부터 유효한 메시지 $m_\ell^{r,1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r,1})$을 수신한다면[34], i는 대기를 중단하고 $v_i' \triangleq (H(B_\ell^r), \ell)$를 설정한다.

[0913] (33: 물론 $CERT^{r-1}$이 $B^{r-1} = B_\epsilon^{r-1}$ 라는 것을 나타내면 i는 그가 $CERT^{r-1}$을 가지는 순간 이미 B^{r-1}을 "수신했다".

[0914] 34: 다시, 플레이어 ℓ의 서명과 해시들이 모두 성공적으로 검증되고, PAY_ℓ^r이 ℓ에 대해 최대인지를 i가 확인하지 않을지라도, B_ℓ^r에서의 PAY_ℓ^r 이 라운드 r에 대한 유효한 페이셋이다. B_ℓ^r이 빈 페이셋을 포함하는 경우, B_ℓ^r이 유효한지 여부를 검증하기 전에 i가 B^{r-1}을 볼 필요가 실제로는 없다.)

[0915] 3. 그렇지 않으면, 시간 t_2가 만료될 때, i는 $v_i' \triangleq \perp$로 설정한다.

[0916] 4. v_i' 의 값이 설정되면 i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고 $i \in SV^{r,2}$가 있는지 여부를 확인한다.

[0917] 5. $i \in SV^{r,2}$인 경우, i는 메시지 $m_i^{r,2} \triangleq (ESIG_i(v_i'), \sigma_i^{r,2})$를 계산하고[35], 자신의 임시 비밀 키 $sk_i^{r,2}$를 파기한 다음 $m_i^{r,2}$ 를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0918] (35: 메시지 $m_i^{r,2}$ 는 플레이어 i가 v_i' 의 제1 구성 요소를 다음 블록의 해시로 간주하거나, 또는 다음 블록이 비어 있다고 간주한다고 신호를 보낸다.)

[0919] 단계 3: GC의 제2 번째 단계

[0920] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 그가 $CERT^{r-1}$을 가지는 즉시 사용자 i는 라운드 r의 자신의 제3 단계를 시작한다.

[0921] · 사용자 i는 최대 시간 $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$동안 대기한다. 대기하는 동안 i는 다음과 같이 동작한

다.

[0922] 1. 임의의 모순 없이[36], 그가 형태 $(ESIG_j(v), \sigma_j^{r,2})$의 적어도 tH의 유효 메시지 $m_j^{r,2}$를 수신하도록 하는 값 v가 존재한다면, 그는 대기를 중단하고 v'=v로 설정한다.

[0923] (36: 즉, 그는 플레이어 j로부터 각각 $ESIG_j(v)$와 상이한 $ESIG_j(\hat{v})$를 포함하는 두 개의 유효한 메시지를 수신하지 못했다. 여기로부터 그리고 이후에, 나중에 정의된 종료 조건을 제외하고, 정직한 플레이어가 주어진 형식의 메시지를 원할 때마다 서로 모순되는 메시지는 카운팅되지 않거나 유효한 것으로 간주되지 않는다.)

[0924] 2. 그렇지 않으면, 시간 t_3이 만료될 때, 그는 $v' = \perp$로 설정한다.

[0925] 3. v'값이 설정되면 i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고, $i \in SV^{r,3}$인지 여부를 확인한다.

[0926] 4. $i \in SV^{r,3}$인 경우, i는 메시지 $m_i^{r,3} \triangleq (ESIG_i(v'), \sigma_i^{r,3})$를 계산하고, 자신의 임시 비밀 키 $sk_i^{r,3}$를 파기한 다음 $m_i^{r,3}$를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0927] 단계 4: GC 출력 및 BBA*의 제1 단계

[0928] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 그가 자신의 제3 단계를 종료하는 즉시 사용자 i는 라운드 r의 자신의 제4 단계를 시작한다.

[0929] · 사용자 i는 최대 시간 2λ동안 대기한다[37]. 대기하는 동안 i는 다음과 같이 동작한다.

[0930] (37: i가 라운드 r의 자신의 단계 1을 시작한 이후의 최대 총 시간은 $t_4 \triangleq t_3 + 2\lambda = 5\lambda + \Lambda$일 수 있다.)

[0931] 1. 그는 다음과 같이 GC의 출력인 v_i 및 g_i를 계산한다.

[0932] (a) 그가 적어도 t_H 개의 유효 메시지 $m_j^{r,3} = (ESIG_j(v'), \sigma_j^{r,3})$를 수신하도록 하는 $v' \neq \perp$이 존재하면, 그는 대기를 중단하고 $v_i \triangleq v'$ 및 $g_i \triangleq 2$를 설정한다.

[0933] (b) 그가 적어도 t_H 개의 유효한 메시지 $m_j^{r,3} = (ESIG_j(\perp), \sigma_j^{r,3})$를 수신했다면, 그는 대기를 중단하고 $v_i \triangleq \perp$ 및 $g_i \triangleq 0$로 설정한다.[38]

[0934] (38: 단계(b)가 프로토콜에 있는지 여부가 그 정확성에 영향을 미치지 않는다. 그러나 충분히 많은 단계 3 검증자가 "서명된(signed) ⊥"을 갖는다면, 단계(b)의 존재는 단계 4가 2λ 시간 이내에 끝나는 것을 허용한다.)

[0935] (c) 그렇지 않으면, 시간 2λ가 만료되면, 그가 적어도 $\lceil \frac{t_H}{2} \rceil$의 유효한 메시지 $m_j^{r,j} = (ESIG_j(v'), \sigma_j^{r,3})$를 수신하도록 하는 $v' \neq \perp$이 존재하면, 그는 $v_i \triangleq v'$ 및 $g_i \triangleq 1$를 설정한다.[39]

[0936] (39: 이 경우에 존재한다면 v'는 고유해야 한다는 것이 증명될 수 있다.)

[0937] (d) 그렇지 않으면, 2λ의 시간이 만료되면, 그는 $v_i \triangleq \perp$ 및 $g_i \triangleq 0$를 설정한다.

[0938] 2. 값 v_i와 g_i가 설정되면 i는 $BBA^{\star}$의 입력인 b_i를 다음과 같이 계산한다.

[0939] g_i=2이면 $b_i \triangleq 0$, 그렇지 않으면 $b_i \triangleq 1$.

[0940] 3. i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고 $i \in SV^{r,4}$인지 여부를 확인한다.

[0941] 4. $i \in SV^{r,4}$인 경우, 그는 메시지 $m_i^{r,4} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$를 계산하고, 자신의 임시 비밀 키 $sk_i^{r,4}$를 파기한 다음 $m_i^{r,4}$를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0942] 단계 s, $5 \le s \le m+2$, $s-2 \equiv 0 \bmod 3$: $BBA^{\star}$의 0에 고정된 코인 단계

[0943] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 그가 자신의 s-1 단계를 종료하는 즉시 사용자 i는 라운드 r의 자신의 s 단계를 시작한다.

[0944] · 사용자 i는 최대 시간 2λ동안 대기한다[40]. 대기하는 동안 i는 다음과 같이 동작한다.

[0945] (40: 따라서 i가 라운드 r의 자신의 단계 1을 시작한 이후의 최대 총 시간은 $t_s \triangleq t_{s-1} + 2\lambda = (2s-3)\lambda + \Lambda$일 수 있다.)

[0946] -종료 조건 0: 임의의 시점에 스트링 $v \neq \perp$ 및 단계 s'가 존재하여,

[0947] (a) $5 \le s' \le s$, $s'-2 \equiv 0 \bmod 3$, 즉, 단계 s'는 0에 고정된 코인 단계이고,

[0948] (b) i는 적어도 t_H 유효 메시지 $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$[41]을 수신했고,

[0949] (41: 플레이어 j로부터의 그러한 메시지는 플레이어 i가 j로부터 1에 대해 서명한 메시지를 받았을지라도 카운트된다. 종료 조건 1과 유사한 사항. 분석에서 볼 수 있듯이, 이는 모든 정직한 사용자가 서로로부터 시간 λ내에 $CERT^r$을 아는 것을 보장하기 위해 수행된다.)

[0950] (c) i는 유효 메시지 $(SIG_j(Q^{r-1}), \sigma_j^{r,1})$를 수신했고, j는 v의 제2 구성요소,

[0951] 가 되도록 하여, i는 대기를 중단하고 즉시 (r, s)-검증자로서 아무것도 전파하지 않으면서 단계 s(실제로는 라운드 r의)의 자신의 실행을 종료하고; $H(B^r)$을 v의 제1 구성 요소로 설정하고; 자신의 $CERT^r$을 $(SIG_j(Q^{r-1}), \sigma_j^{r,1})$[42]와 함께 단계 (b)의 메시지 $m_j^{r,s'-1}$의 세트로 설정한다.

[0952] (42: 사용자 i는 이제 $H(B^r)$과 그의 라운드 r이 끝나는 것을 안다. 그는 실제로 블록 B^r이 그에게 전파될 때까지 대기할 필요가 있고, 이는 일부 추가 시간을 소요할 수 있다. 그는 여전히 일반 사용자로서 메시지를 전파하는 데 도움을 주지만, (r, s)-검증자로서 전파를 시작하지는 않는다. 특히 그는 자신의 $CERT^r$에서 모든 메시지를 전파하는 데 도움을 주었으며, 이는 우리의 프로토콜에 충분하다. 이진 BA 프로토콜에 대해 $b_i \triangleq 0$로 설정해야 하지만, 이 경우 어쨌든 b_i는 필요하지 않다는 것에 유의하라. 앞으로의 모든 지시에 대해 유사한 사항이다.)

[0953] - 종료 조건 1: 임의의 시점에서 단계 s'가 단계가 존재하여,

[0954] (a') $6 \le s' \le s$, $s'-2 \equiv 1 \bmod 3$, 즉, 단계 s'는 1에 고정된 코인 단계이고,

[0955] (b') i는 적어도 tH 유효한 메시지 $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j) \ \sigma_j^{r,s'-1})$를 수신[43],

[0956] 하도록 하여, i는 대기를 중단하고 즉시 (r, s)-검증자로서 아무것도 전파하지 않으면서 단계 s(실제로는 라운드 r의)의 자신의 실행을 종료하고; $B^r = B^r_\epsilon$로 설정하고; 자신의 $CERT^r$을 하위 단계(b')의 메시지 $m_j^{r,s'-1}$의 세트가 되도록 설정한다.

[0957] (43: 이 경우에는 v_j가 무엇인지는 중요하지 않다.)

[0958] - 임의의 시점에 그가 적어도 t_H의 유효 메시지 $m_j^{r,s-1}$의 형태 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$를 수신하는 경우, 그는 대기를 중단하고 $b_i \triangleq 1$로 설정한다.

[0959] - 임의의 시점에 그가 적어도 t_H의 유효 메시지 $m_j^{r,s-1}$의 형태 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$를 수신하지만, 동일한 v에는 동의하지 않는 경우, 그는 대기를 중단하고 $b_i \triangleq 0$로 설정한다.

[0960] - 그렇지 않으면, 시간 2λ가 만료되면, i는 $b_i \triangleq 0$로 설정한다.

[0961] - 값 b_i가 설정되면, i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고 $i \in SV^{r,s}$인지 여부를 확인한다.

[0962] -$i \in SV^{r,s}$라면, i는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$(v_i는 그가 단계 4에서 계산한 값)를 계산하고, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기하고, 그런 다음 $m_i^{r,s}$를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0963] 단계 s, $6 \le s \le m+2$, $s-2 \equiv 1 \bmod 3$: $BBA^\star$의 1에 고정된 코인 단계

[0964] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 자신의 단계 s-1을 종료하자마자 라운드 r의 자신의 단계 s를 시작한다.

[0965] · 사용자 i는 2λ의 최대 시간을 대기한다. 대기하는 동안 i는 다음과 같이 동작한다.

[0966] - 종료 조건 0: 0에 고정된 코인 단계와 동일한 지침.

[0967] - 종료 조건 1: 0에 고정된 코인 단계와 동일한 지침.

[0968] - 임의의 시점에 그가 적어도 t_H의 유효한 메시지 $m_j^{r,s-1}$의 형태 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1}$을 수신하면, 그는 대기를 중단하고 $b_i \triangleq 0$로 설정한다.[44]

[0969] (44: 1에 대해 t_H 유효 (r, s-1)-메시지를 수신하는 것은 종료 조건 1을 의미한다는 것에 유의하라.)

[0970] - 그렇지 않으면, 시간 2λ가 만료되면, i는 $b_i \triangleq 1$로 설정한다.

[0971] - 값 b_i가 설정되면, i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고 $i \in SV^{r,s}$인지 여부를 확인한다.

[0972] -$i \in SV^{r,s}$라면, i는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$($v_i$는 그가 단계 4에서 계산한 값)를 계산하고, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기하고, 그런 다음 $m_i^{r,s}$를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0973] 단계 s, 7≤s≤m+2, s-2≡2 mod 3: $BBA^{\star}$의 전체적으로 플립핑된 코인 단계

[0974] 모든 사용자 $i \in PK^{r-k}$에 대한 지침: 사용자 i는 그가 자신의 단계 s-1을 종료하자마자 라운드 r의 자신의 단계 s를 시작한다.

[0976] * · 사용자 i는 최대 시간 2λ을 대기한다. 대기하는 동안 i는 다음과 같이 동작한다.

[0977] - 종료 조건 0: 0에 고정된 코인 단계와 동일한 지침.

[0978] - 종료 조건 1: 0에 고정된 코인 단계와 동일한 지침.

[0979] - 임의의 시점에 그가 적어도 t_H의 유효한 메시지 $m_j^{r,s-1}$의 형태 $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1}$을 수신하면, 그는 대기를 중단하고 $b_i \triangleq 0$로 설정한다.

[0980] - 임의의 시점에 그가 적어도 t_H의 유효한 메시지 $m_j^{r,s-1}$의 형태 $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1}$을 수신하면, 그는 대기를 중단하고 $b_i \triangleq 1$로 설정한다.

[0981] - 그렇지 않으면, 시간 2λ가 만료되면, $SV_i^{r,s-1}$가 그가 유효한 메시지 $m_j^{r,s-1}$를 수신한 (r, s-1)-검증자의 세트로 놓고, i는 $b_i \triangleq \mathbf{lsb}(\min_{j \in SV_i^{r,s-1}} H(\sigma_j^{r,s-1}))$로 설정한다.

[0982] - 값 b_i가 설정되면, i는 $CERT^{r-1}$로부터 Q^{r-1}을 계산하고 $i \in SV^{r,s}$인지 여부를 확인한다.

[0983] - $i \in SV^{r,s}$라면, i는 메시지 $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$(v_i는 그가 단계 4에서 계산한 값)를 계산하고, 자신의 임시 비밀 키 $sk_i^{r,s}$를 파기하고, 그런 다음 $m_i^{r,s}$를 전파한다. 그렇지 않으면, i는 아무것도 전파하지 않고 중단한다.

[0984] **비고.**

[0985] 이론적으로, 하위 섹션 6.2에서 고려한 바와 같이, 프로토콜은 임의의 라운드에서 임의로 많은 단계를 취할 수 있다. 이 문제가 발생하면, 논의된 바와 같이, s>μ인 사용자 $i \in SV^{r,s}$는 자신의 은닉한 미리 생성된 임시 키를 소비했다가 임시 키의 '캐스케이드(cascade)'로 자신의(r, s)-메시지 $m_i^{r,s}$를 인증해야 한다. 따라서 i의 메시지가 조금 더 길어지고 이들 더 긴 메시지를 전송하는데 시간이 조금 더 걸릴 것이다. 따라서, 주어진 라운드의 많은 단계들 후에, 파라미터 λ의 값은 자동적으로 약간 증가할 것이다. (하지만 새로운 블록이 생성되고 새로운 라운드가 시작되면 원래의 λ로 되돌아간다.)

[0986] 비 검증자에 의한 라운드 r 블록의 재구성

[0987] 시스템의 모든 사용자 i에 대한 지침: 사용자 i는 $CERT^{r-1}$을 얻자마자 자신의 라운드 r을 시작한다.

[0988] · i는 프로토콜의 각 단계의 지침을 따르고, 모든 메시지의 전파에 참여하지만, 그가 그것에서의 검증자가 아닌 경우 단계에서 전파를 시작하지 않는다.

[0989] · i는 일부 단계에서 종료 조건 0 또는 종료 조건 1을 해당 $CERT^r$과 함께 입력하여 자신의 라운드 r을 종료한다.

[0990] · 그곳에서부터, 실제 블록 B^r을 수신하기 위해 대기하는 동안, 그는 자신의 라운드 r+1을 시작하고(그가 그것

을 이미 수신하지 않았다면), 그의 해시 H(B^r)는 CERT^{r}에 의해 고정되어 있다(pinned). 다시, CERT^{r}가 $B^r = B^r_\epsilon$을 나타내면, i는 그가 CERT^{r}를 가지고 있는 순간을 B^{r}이라고 알고 있다.

[0991] 6.4 $Algorand'_1$의 분석

[0992] $Algorand'_2$의 분석은 $Algorand'_1$의 분석에서 쉽게 도출된다. 본질적으로, $Algorand'_2$에서 압도적인 확률로, (a) 모든 정직한 사용자가 동일한 블록 B^{r}에 동의하고; 새로운 블록의 리더는 적어도 $p_h=h^2(1+h-h^2)$의 확률로 정직하다.

[0993] 7. 오프라인의 정직한 사용자 핸들링

[0994] 우리가 말했듯이, 정직한 사용자는 온라인 상태이고 프로토콜을 실행하는 것을 포함한 모든 규정된 지침을 따른다. 정직한 사용자로부터 요구되는 계산 및 대역폭이 매우 적기 때문에 이것은 Algorand에서 큰 부담이 아니다. 그러나 Algorand는 정직한 사용자가 많은 수의 오프라인일 수 있는 두 가지 모델에서 작동하도록 쉽게 수정할 수 있다고 언급하자.

[0995] 이 두 모델을 논의하기 전에, 정직한 플레이어의 비율이 95%라면 Algorand는 대신에 h=80%라고 하는 모든 파라미터를 가정하여 실행할 수 있다고 설명한다. 따라서 Algorand는 정직한 플레이어의 최대 절반이 오프라인(사실, "장기 부재(absenteeism)"의 대부분의 경우)으로 전환하기를 선택한 경우에도 계속해서 제대로 작동한다. 실제로, 어떤 시점에서든 온라인 플레이어의 최소 80%는 정직할 것이다.

[0996] 게으른 정직에 대한 지속적인 참여

[0997] 살펴본 바와 같이, $Algorand'_1$과 $Algorand'_2$는 룩 백(look-back) 파라미터 k를 선택했다. k를 적절히 크게 선택하면 지속적인 참여 요구 사항을 제거할 수 있음을 보여주자. 이 요구 사항은 중요한 속성을 보장하는데: 즉, 기본 BA 프로토콜 BBA*가 적절한 정직한 다수를 가진다는 것이다. 게으른 정직이 어떻게 이 속성을 만족시키는 대안적이고 매력적인 방법을 제공하는지 설명하자.

[0998] 사용자 i는, (1) 그가 프로토콜에 참여하도록 요청받았을 때, 그가 자신의 모든 규정된 지침을 따르고, (2) 적절한 사전 예고를 통해 프로토콜에 예를 들어, 일주일에 한 번과 같이, 매우 드문 경우에만 참여하도록 요청받고, 그가 참여할 때 잠재적으로 현저한 보상을 받는 경우, 게으르지만 정직하다는 것을 상기하라.

[0999] Algorand가 그러한 플레이어와 함께 작업할 수 있게 하려면, "훨씬 초기의 라운드에서 시스템에 이미 있는 사용자들 중에서 현재 라운드의 검증자를 선택하는 것만으로" 충분하다. 실제로, 라운드 r의 검증자는 라운드 r-k의 사용자로부터 선택되며, 선택은 양 Q^{r-1}을 기반으로 이루어진다. 1주일은 약 1만 분으로 구성되며, 라운드는 대략(예를 들어, 평균적으로) 5분이 걸리므로, 1주일에 약 2,000 라운드가 소요된다고 가정한다는 것에 유의하라. 어떤 시점에서 사용자 i는 자신의 시간을 계획하고 다음주에 자신이 검증자가 될 것인지를 알고 싶어한다고 가정한다. 프로토콜은 라운드 r-k-2,000의 사용자로부터 라운드 r에 대한 검증자를 선택하며, 선택은 Q$^{r-2,001}$에 기초한다. 라운드 r에서, 플레이어 i는 실제로는 그것들은 블록체인의 일부이기 때문에 이미 값 Q$^{r-2,000}$, ..., Q^{r-1}을 알고 있다. 그런 다음, 1과 2,000 사이의 각 M에 대해, i는 하기의 경우에, 또는 하기의 경우에만 라운드 r+M의 단계 s에서 검증자이다

[1000] $$.H\left(SIG_i\left(r+M,s,Q^{r+M-2,001}\right)\right) \le p.$$

[1001] 따라서 그가 다음 2000 라운드에서 검증자의 역할을 하도록 호출될 것인지를 확인하려면, i는 각 단계 s에서 M=1 내지 2000에 대해 $\sigma_i^{M,s} = SIG_i\left(r+M,s,Q^{r+M-2,001}\right)$를 계산해야 하고, 그것들 중 일부에 대해 $.H(\sigma_i^{M,s}) \le p$인지를 확인해야 한다. 디지털 서명을 계산하는 데 1밀리 초가 걸리는 경우, 이 전체 작업에 약 1분의 계산 시간이 소요된다. 이들 라운드에서 그가 검증자로 선정되지 않으면, 그는 "정직한 양심"을 가지고 오프라인으로 갈 수 있다. 그가 계속해서 참여했다면, 그는 다음 2000 라운드에서 어쨌건 반드시 0단

계를 거쳤을 것이다! 대신에 그는 이들 라운드 중 하나에서 검증자로 선정되는 경우, 그는 적절한 라운드에서 정직한 검증자로서 역할을 하도록 자신을 준비한다(예를 들어, 필요한 모든 정보를 얻음으로써).

[1002] 그렇게 행동함으로써, 게으르지만 정직한 잠재적인 검증자인 i는 단지 메시지 전파에 참여하는 것만을 놓친다. 그러나 메시지 전파는 일반적으로 강건하다. 또한, 최근에 전파된 결제의 지불자와 수취인은 자신들의 결제에 무슨 일이 일어나는지 감시하기 위해 온라인 상태가 될 것으로 예상되고, 따라서 그들이 정직한 경우 메시지 전파에 참여하게 된다.

[1003] **8. 정직한 대다수의 돈을 가진 프로토콜 Algorand'**

[1004] 우리는 마침내, 정직한 대다수의 사용자 가정을 보다 의미있는 정직한 대다수의 돈의 가정으로 대체하는 방법을 보여준다. 기본 아이디어는 (지분 증명의 특색에서) "i에 의해 소유된 금액에 비례하는 가중치(즉, 결정력)를 가진 $SV^{r,s}$에 속하는 사용자 $i \in PK^{r-k}$를 선택하는 것"[45]이다.

[1005] (45: 우리는 지속적인 참여를 대체하기 위해 $PK^{r-k-2,000}$을 선언해야 한다. 단순함을 위해, 사람들은 어쨌든 지속적인 참여 요구하기를 원하기 때문에, 우리는 이전처럼 PK^{r-k}를 사용하여 하나의 더 적은 파라미터를 전달한다.)

[1006] 우리의 HMM 가정에 따라, 우리는 그 금액이 라운드 r-k에서 소유되어야 하는지 또는 라운드 r(의 시작시)에서 소유되어야 하는지를 선택할 수 있다. 우리가 지속적인 참여에 신경 쓰지 않는다고 가정할 때, 우리는 후자의 선택을 채택한다. (지속적인 참여를 없애기 위해 우리는 전자를 선택했을 것이다. 라운드 r-k-2,000에서 소유한 금액에 대해 더 좋았다.)

[1007] 이 아이디어를 구현하는 데는 여러 가지 방법이 있다. 가장 간단한 방법은 각 키를 최대 1단위의 화폐로 유지한 다음, $a_i^{(r)} = 1$가 되도록 PK^{r-k}로부터 랜덤한 n 명의 사용자 i를 선택하는 것이다

[1008] **다음의 가장 단순한 구현**

[1009] 다음으로 가장 단순한 구현은 고정된 M에 대해 각 공개키가 최대 금액 M을 소유하도록 요구하는 것일 수 있다. M 값은 시스템에서의 총 금액과 비교하여 충분히 작아서, 키가 예를 들어 라운드 k에서 하나 이상의 단계의 검증자 세트에 속할 가능성이 무시할 정도로 작게 되도록 한다. 그런 다음, 라운드 r에서 금액 $a_i^{(r)}$을 소유하는 키 $i \in PK^{r-k}$가 하기와 같은 경우 $SV^{r,s}$에 속하도록 선택된다:

[1010] $$.H\left(SIG_i\left(r,s,Q^{r-1}\right)\right) \leq p \cdot \frac{a_i^{(r)}}{M}$$

[1011] 그리고 모든 것은 이전처럼 진행된다.

[1012] **보다 복잡한 구현**

[1013] 마지막 구현은 "시스템 내의 풍부한 참여자가 많은 키를 소유하도록 강제되는" 것이다.

[1014] 아래에 설명된 대안의 구현은 상태의 표기를 일반화하고 각각의 사용자 i가 K+1 사본(i, v)으로 구성되는 것으로 간주하며, 이들의 각각은 검증자로 독립적으로 선택되며 라운드 r의 단계 s에서 자신의 임시 키 $(pk_{i,v}^{r,s}, sk_{i,v}^{r,s})$를 소유하게 할 것이다. 값 K는 라운드 r에서 i가 소유한 금액 $a_i^{(r)}$에 달려있다.

[1015] 이러한 시스템이 어떻게 작동하는지 더 상세히 살펴보자.

[1016] **사본의 수**

[1017] n을 각 검증자 세트의 목표로 예측된 카디널리티(target cardinality)라고 하고, 라운드 r에서 사용자 i가 소유한 금액을 $a_i^{(r)}$라고 하자. A^r을 라운드 r에서 PK^{r-k}에서의 사용자가 소유한 총 금액, 즉,

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

[1018] 이라고 하자.

[1019] i가 PK^{r-k}에 있는 사용자인 경우, i의 사본은 (i, 1), ..., (i, K+1)이고, 여기서,

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

[1020] 이다.

[1021] **예시.** n = 1,000, $A^r=10^9$, 그리고 $a_i^{(r)}$=3.7백만이라고 하자. 그러면,

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

[1022] 이 된다.

[1023] **검증자 및 보증서**

[1024] i를 K+1의 사본을 가진 PK^{r-k}에서의 사용자로 보자.

[1025] 각 v=1, ..., K에 대해, 사본 (i, v)는 $SV^{r,s}$에 자동으로 속한다. 즉, i의 보증서가 $\sigma_{i,v}^{r,s} \triangleq SIG_i((i,v), r, s, Q^{r-1})$이 되지만, 대응하는 조건은 항상 참인 $.H(\sigma_{i.v}^{r,s}) \leq 1$이 된다.

[1026] 사본(i, K+1)에 대해, 라운드 r의 각 단계에 대해서, i는

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

[1027] 인지 여부를 확인한다.

[1028] 그런 경우, 사본(i, K+1)는 $SV^{r,s}$에 속한다. 그것을 증명하기 위해, i는 보증서

$$\sigma_{i,K+1}^{r,1} = SIG_i\left((i, K+1), r, s, Q^{r-1}\right)$$

[1029] 을 전파한다.

[1030] **예.** 앞의 예시에서와 같이, n=IK, $a_i^{(r)}$=3.7M, A^r=1B라고 하고, i는 4개의 사본 (i, 1), ..., (i, 4)을 갖는다. 그런 다음 처음 3부는 $SV^{r,s}$에 자동으로 속한다. 네번째의 사본의 경우, 개념적으로, Algorand'는 그의 헤드(Head)의 확률이 0.7인 바이어싱된 동전을 굴린다(roll). 코인 토스가 헤드인 경우 또는 그런 경우에만 사본(i, 4)이 선택된다.

[1031] (물론 이 바이어싱된 코인 플립핑은 i로 하여금 그의 결과를 증명할 수 있게 하기 위해 해싱, 서명 및 비교(이 애플리케이션에서 모두 다했듯이)에 의해 구현된다.)

[1032] **일상적인 비즈니스**

[1033] 검증자가 선택되는 방법과 라운드 r의 각 단계에서 보증서가 계산되는 방식에 대해 설명했으므로, 라운드의 실행은 이미 설명한 것과 유사하다.

[1034] **9. 포크**

[1035] 포크의 확률을 10^{-12} 또는 10^{-18}로 줄이면, 그것이 발생한 멀리 떨어진 곳에서 포크를 처리할 필요가 거의 없다. 그러나 Algorand는 또한 작업 증명이 있건 없건 다양한 포크 해결 절차를 사용할 수도 있다.

[1036] **10. 블록 및 상태 정보를 위한 새로운 구조**

[1037] 이 섹션에서는 블록을 구성하고, 모든 블록의 조작방지를 보장하지만, 또한 개별 블록의 콘텐츠를 증명할 수 있는 보다 효율적인 방법, 보다 일반적으로 블록을 사용하여 전체 블록체인을 검사하지 않고도 관심을 가지는 특

정 수량을 연산하는 보다 효율적인 방법을 가능하게 하는 더 좋은 방법을 제안한다.

[1038] **블록체인의 조작방지**

[1039] 블록체인을 형성하기 위해 블록 B_r은 다음과 같은 고 레벨 구조를 가지고 있음을 상기하라:

[1040] $B_r=(r,\ INFO_r,\ H(B_{r-1}))$.[46]

[1041] (46: 우리가 Algorand에서 라운드를 나타내기 위해 아래 첨자(subscript)를 사용한다는 것을 상기하라. 그러나 이 섹션은 일반적으로 블록체인에만 사용되므로 r번째 블록은 Algorand의 의미에서 r번째 라운드에 대응하지 않을 수 있다. 즉, 위의 위의 "r"은 단지 블록 번호이며, 명확성을 위해 블록 B_r에 포함된다.

[1042] 또한 블록의 위의 일반적인 구조는 개념적이다. 예를 들어, 비트코인에서 B_r은 해당 계산상의 수수께끼를 해결한 블록 생성자(constructor)의 디지털 서명을 포함할 수 있다. Algorand에서 B_r의 인증, 즉,일치하는 인증서 $CERT_x$가 별도로 제공될 수 있다. 그러나 그것은 또한 B_r의 필수적인 부분으로서 제공될 수 있다. 후자의 경우, 다수의 유효한 인증서가 있을 수 있으므로, 라운드 r의 리더 ℓ_r는 모든 라운드 r 검증자에게 보낸 메시지에 이전 라운드의 출력에 대한 유효한 인증서를 포함할 수 있으므로 동의는 또한 각 라운드 r의 인증서가 무엇인지에 따라 합의하도록 한다.)

[1043] 위의 $INFO_r$는 r번째 블록 내에서 보안하고자 하는 정보로: Algorand의 경우, $INFO_r$에는 수량 Q^{r-1} 등 블록 리더의 서명인 PAY^r이 포함된다.

[1044] 블록체인의 잘 알려진 기본 속성은 각 블록의 내용을 조작 방지하는 것이다. 즉, 아무도 마지막 블록을 변경하지 않고 과거 블록을 변경할 수 없다. 우리는 이 속성을 빨리 아래에서 상기한다.

[1045] 최신 블록을 B_{last}로 하고, B^r을 다른("비뚤어진") 블록 $\widetilde{B_r}$로 대체한다고 가정한다. 그러면 H가 충돌 탄력성을 가지므로 $H(\widetilde{B_r})$은 압도적인 확률로 $H(B_r)$와는 상이하다. 따라서, 블록체인, 즉 $\widetilde{B_{r+1}}=(r+1,\widetilde{INFO}_{r+1},H(\widetilde{B_r}))$에서의 블록에서 정보 $\widetilde{INFO}_{r+1}$를 아무리 선택해도, 다시 H의 충돌 탄성력(collision resiliency)으로 인해 하기와 같이 된다:

[1046] $$\widetilde{B_{r+1}}\neq B_{r+1}$$

[1047] 이 부등식이 전파된다. 즉, $\widetilde{B_{r+2}}$는 B^{r+2}와 다르는 등으로 궁극적으로,

[1048] $\widetilde{B_{last}}\neq B_{last}$가 되도록 한다.

[1049] **블록체인에서의 개별 블록들의 비효율적인 검증**

[1050] 블록체인 전체를 모르지만, B_z가 그 안의 정확한 블록이라는 것을 아는 사람 X를 고려해보자. 그러면 위의 기본 속성을 통해 누군가는 r이 z보다 작은 임의의 개별 블록 B_r이 또한 올바른 것으로 이 사람에 대해 증명할 수 있다. 즉, 그가 z 블록을 재구성하고 그것이 그가 아는 블록 B_z와 일치하는지를 확인할 때까지, 그는 r 번째부터 앞으로 블록체인을 재생성하기 위해 H를 사용하는 X에게 모든 중간 블록(B_{r+1}, ..., B_{z-1})을 "증명"으로서 제공한다. 이 경우, X는 B_r의 정확성을 확신한다. 실제로, 겉으로 보기에 합법적인 증명을 찾을 수 있는 모든 사람은 찾는 것이 실제로는 불가능한 해시 함수 H에서 충돌을 발견해야만 한다.

[1051] 이러한 검증가능성의 유용성을 이해하기 위해서는 다음 예시를 고려하라. 블록체인이 비트코인 또는 Algorand와 같은 결제 시스템에 의해 생성되고, X를 피고가 2년 이전에 실제로 원고(plaintiff)에게 분쟁의 결제 P를 하였

음을 X에게 증명하기를 원하는 법원의 판사(judge)라고 가정한다. 판사가 체인에서 정확한 마지막 블록을 얻을 수 있다고, 또는 적어도 충분히 최근의 정확한 블록 B_z을 얻을 수 있다고 가정하는 것이 합리적이기 때문에, 피고가 해야하는 "모두"는 증명 B_{r+1}, ..., B_{z-1}을 판사에게 제공하는 것이고, 그는 그런 다음 그것을 설명된 대로 검증한다. 물론 문제는 그런 증명이 꽤 길 수도 있다는 것이다.

[1052] **블록 트리**

[1053] 과거 개별 블록의 정확한 내용을 효율적으로 증명할 수 있는 능력이 매우 중요하기 때문에, 우리는 새로운 블록 구조를 개발한다. 이러한 구조에서, 블록체인과 유사하게, 전체 블록 시퀀스의 무결성(integrity)이 훨씬 더 짧은 값 v에 의해 간단하게 보장된다. 이 값은 시퀀스의 마지막 블록이 아니다. 그러나 블록체인의 기본적인 속성은 유지되는데: 블록 중 하나에서 변경되면 v에서의 변화를 가져온다.

[1054] 새로운 구조의 이점은 v가 주어지면 n을 시퀀스에 현재 있는 블록의 수로 놓고, 각 개별 블록의 내용을 매우 효율적으로 증명할 수 있다는 것이다. 예를 들어, 우리의 새로운 블록 구조의 특정 구현인 블록 트리에서, 계속 생성된 블록의 총 수가 n인 경우, 각 블록은 단지 $32 \cdot \lceil \log n \rceil$의 바이트의 정보를 제공함으로써 증명될 수 있다.

[1055] 이것은 실제로 매우 컴팩트한 증명이다. 1분당 1블록을 생성하는 시스템에서, 2 천년의 작업 후, [log] <30가 된다. 따라서 각 개별 블록의 내용을 증명하는 데는 1KB-1,000 바이트 미만이면 충분하다. (2KB 미만은 20 억년이 지나면 충분하며, 4KB는 본질적으로 영원하다.) 또한 이러한 증거는 매우 효율적으로 검증된다.

[1056] **효율적인 상태**

[1057] 다른 효율성 문제는 비트코인과, 더 일반적으로 블록체인을 기반으로 한 결제 시스템에 영향을 미친다. 즉, 시스템의 상태(즉, 어떤 키가 주어진 시간에 무엇을 소유하고 있는지)를 재구성하려면, 그는 결제 내역 전체(그 때까지의)를 얻어야 하고, 이루어진 결제의 수가 매우 많을 때에는 하기가 어려울 수 있다.

[1058] 우리는 그러한 요구 사항을 달성하기 위해 블록 트리를 사용할 것이다.

[1059] 우리는 아래에서 상기된 훨씬 오래된 개념을 적절하게 수정하여 블록 트리를 구성한다.

[1060] **10.1 머클 트리**

[1061] 머클 트리는 각 값 v_i의 인증을 개별적으로 그리고 효율적으로 검증할 수 있도록, 단일 값 v에 의해 n 개의 이미 알려진 값, v_0, ..., v_{n-1}을 인증하는 방법이다.

[1062] 간단히 하기 위해, n은 2의 거듭 제곱($n=2^k$)으로 가정하여 각 값은 개별 k 비트 스트링 s에 의해 고유하게 식별되도록 한다. 그런 다음 머클 트리 T는 길이가 ≤k인 이진 스트링을 사용하여 그의 노드가 고유하게 명명된 깊이 k의 풀 이진 트리에 특정 값을 저장함으로써 개념적으로 구성된다.

[1063] 루트는 ε이라는 빈 스트링이다. 내부 노드의 이름이 s인 경우, s의 좌측의 자식(child)은 s0(즉 s와 0을 사슬로 연결하여 얻은 스트링)이고 우측 자식은 s1로 명명된다. 그런 다음, 머클 트리 T를 구성하기 위해 가능한 선행 0을 갖는 이진 k 비트 확장을 가지고 각각의 정수 $i \in \{0, \ldots, n-1\}$을 식별하면서, 리프 i에 각 값 v_i를 저장한다. 그 후에, 그는 트리를 머클파이(merklefy)하는데, 즉, 그는 T의 모든 다른 노드에 하기와 같이 상향식으로 저장한다(즉, 먼저 깊이 k-1의 모든 노드의 내용을 선택한 다음 깊이 k-2, 등의 모든 노드들을 선택함으로써). v_{s0} 및 v_{s1}이 노드 s의 좌우측 자식에 각각 저장되는 경우, 노드 s에 256비트 값 $v_s \triangleq H(v_{s0}, v_{s1})$를 저장한다. 이 프로세스가 끝나면, 루트에는 256 비트 값 v_ε이 포함된다.

[1064] 깊이 3과 8 리프의 머클 트리가 도 1. A에 나와있다.

[1065] 이제 v_ε가 공지되거나 디지털로 서명되었다고 가정하고, 각각의 원래 값 v_i가 v_ε에 대해 어떻게 인증될 수 있는지 보여 주자.

[1066] 노드 x에서 시작하여 루트에 도달하는(가장 짧은) 경로 P를 생각해보라. 그러면, x의 내용 v_x의 인증 경로는 P에서의 노드들의 형제들의 콘텐츠의 시퀀스이고, 여기서 노드 s0의 형제가 노드 s1이고 그 반대로도 마찬가지이

다. 따라서 깊이 k의 트리에서 리프 값의 인증 경로는 k-1 값으로 구성된다. 예를 들어, 도 1.A의 머클 트리에서 리프 010에서부터 루트까지의 경로는 P=010,01,0, ε 이며 루트는 형제가 없으므로 v_{010}의 인증 경로는 v_{011}, v_{00}, v_1이다. 도 1.A의 머클 트리에서 v_{010}의 경로 P와 인증 경로는 도 1.B와 같다.

[1067] 2^k 리프를 가진 풀 이진 트리를 사용하여 i 번째 리브에 각각의 v_i를 저장함으로써 n<2k 값, v_0, ..., v_{n-1}을 인증하는 머클 트리, 및 모든 나머지 리프들에서의 특수 스트링 e(비어 있음)를 가져, 일반적인 방식으로 노드의 나머지를 채울 수 있다. 결과적인 머클 트리는 사실상 n 개의 리프들 가지고 있다(다른 모든 리프들은 비어 있기 때문에).[47]

[1068] (47: 또한 그의 콘텐츠가 e인 임의의 노드 아래의 머클 트리에 있는 모든 노드를 "트림"할 수 있도록 해시 함수 H가 $H(e,e) \triangleq e$ 하도록 한다고 가정할 수 있다.)

[1069] 루트 값 v_ε에 대해 자신의 인증 경로가 주어지면 v_i(의 신빙성)를 검증하기 위해, 리프 i에서 루트까지 경로의 노드의 콘텐츠를 재구성하고, 그런 다음 마지막으로 재구성된 값이 실제로 v_ε인지 여부를 확인한다. 즉, 인증 경로가 x_1, ..., x_{k-1}인 경우, 먼저 우측의 순서에서 v_i 및 x_i와 함께 H해싱되고, 즉, $y_2 = H(v_i, x_1)$을 연산하고, 그렇지 않고 i의 마지막 비트가 0이면, $y_2 = H(x_1, v_i)$를 계산한다. 그런 다음 우측의 순서에서 y_2와 x_2를 H 해싱한다. y_k 값을 계산하여 그것을 v_ε과 비교할 때까지 계속된다. 값 v_i는 $y_k=v_\varepsilon$인 경우, 및 그런 경우에만 인증된다.

[1070] 이러한 검증 작업이 이루어지는 이유는, 다시 한번, H가 충돌 탄력성이라는 것이다. 실제로 리프나 노드에 원래 저장된 값의 단일 비트를 변경하는 경우에도 부모에 저장된 값이 압도적인 확률로 변경된다. 이 변화는 루트에서의 값을 공지된 값 v_ε과 다르게 만드는 원인이 된다.

[1071] **10.2 블록 트리**

[1072] 우리가 보았듯이 머클 트리는 임의의, 임의로 다수인, 공지된 값을 단일한 값인 루트에 저장된 값인 v_ε 값을 사용하여 효율적으로 인증한다. 실제로 머클 트리의 단일 루트의 콘텐츠에 의해 k 값, v_0, ..., v_{k-1}을 인증하기 위해, 그것들을 트리의 제1 리프 k에 저장하고, 다른 적절한 노드에 e를 저장하고, 그런 다음 루트 값을 포함하여 트리 내의 모든 다른 노드의 콘텐츠를 계산하기 위해, 먼저 v_0, ..., v_{k-1}을 알아야 한다.

[1073] 머클 트리는 주어진 블록의 결제를 인증하기 위해 비트코인에서 사용되었다. 실제로 주어진 블록을 구성할 때 이미 블록에 넣을 결제를 선택했다.

[1074] 그러나 머클 트리를 사용하여 증가하는 블록체인을 인증하는 것은 인증할 블록을 미리 알지 못하기 때문에 더욱 어려워진다.

[1075] 그러나, 우리는 개별 블록들의 효율적인 입증가능성(provability)을 가능하게 하는 새로운 블록 구조를 얻기 위해 새로운 방식으로 사용되는 머클 트리를 사용하는 방법을 보여준다. 우리가 선호하는 그러한 구조인 블록 트리를 설명해 보자.

[1076] **블록트리 보장(Blocktree Guarantees)**

[1077] 블록트리는 블록의 시퀀스 B_0, B_1, ...의 각각에 포함된 정보를 보호한다. 이 중요한 속성은 블록체인에서와 마찬가지로 이전 블록의 해시를 각 블록에 저장하여 얻을 수는 없다. 그러나, 각각의 블록은 또한 소정의 블록 B_r에 선행하는 블록 B_i의 콘텐츠에서 이루어진 임의의 변경이 B_r의 보안 정보도 변경하게할 것이라는 보장과 함께, 일부 짧은 보안 정보를 저장한다. 이러한 블록트리의 보장은 블록체인에서 제공하는 것과 동일하다. 주요 이점은 새로운 보안 정보를 통해 블록 B_r의 보안 정보를 아는 사람에게 B_i와 B_r 사이의 모든 블록을 처리해야할 필요 없이 임의의 블록 B_i의 정확한 내용을 증명하도록 하는 것이다. 이는 블록의 수가 매우 커질 수 있기(커지게 될 수 있기) 때문에 큰 이점이다.

[1078] **블록 트리의 블록 구조**

[1079] 블록 트리에서 블록 B_i의 형식은 다음과 같다.

[1080] $B_r=(r, INFO_r, S_r)$,

[1081] 여기서 $INFO_r$은 보안이 필요한 블록 B_r의 정보를 나타내고 S_r은 B_r의 보안 정보를 나타낸다.

[1082] 블록 트리의 본 발명의 바람직한 실시 예에서, 보안 정보 S_r 정보는 실제로 상당히 콤팩트하다. 이것은 [logr] 256비트 스트링의 시퀀스, 즉 각각 32바이트의 [log r] 스트링으로 구성된다. 대부분의 실제 애플리케이션에서, 2^{40}은 수천조보다 크기 때문에 [log r]<40라는 것을 유의하라.

[1083] 아래에서는 블록 트리에 대해서만 정보 S_r를 지정한다. 당업자는 그것이 본질적으로 동일한 보증을 갖는 다양한 다른 블록 구조로 쉽게 일반화될 수 있음을 인식할 수 있으며, 이들 모두는 본 발명의 범위 내에 있다.

[1084] **블록 트리 내에서 블록 생성**

[1085] 간결성을 위해 $INFO_r=v_r$로 설정해 보자. 개념적으로, 2^k가 가능한 값 v_r의 수를 상한으로 하도록 깊이 k의 풀 이진 트리 T로 시작한다. 값 v_0, v_1, ...이기 때문에, 블록은 순서대로 생성된다. 새로운 값 v_i이 생성되면, 개념적으로 말하면 머클 트리 T_i를 구성하기 위해, 그것은 T의 리프 i에 저장되고, 그런 다음 다양한 스트링이 계산되어 T의 노드에 저장된다. 이 스트링 중 하나는 차별화된(distinguished) 스트링 *e*이다. (T_i의 노드 x에 나타날 때, 스트링 *e*는 x의 자손(descendant)이 Ti에 속하지 않는다는 것을 의미하며, 우리는 $H(e,e) \triangleq e$라고 가정한다.)

[1086] 제1 값 v_0가 생성되어 리프 0에 저장되면, T_0는 T의 노드 0과 일치한다(그렇게 채워짐). 실제로, 이러한 T_0는 기본 머클 트리이다. 그것의 깊이는 [log(0+1)]=0이고, 그 루트는 R_0=0이며, 그것은 자신의 제1 깊이-0 리프(그리고, 실제로, 자신의 리프 및 노드에만)에 v_0를 저장한다.

[1087] i+1번째 값인, v_i가 생성되어 T의 리프 i에 저장될 때(이미 거기에 있는 스트링 *e*를 대체할 수 있음), 머클 트리 T_i는 이전 머클 트리 T_{i-1}에서 다음과 같이 구성된다. (유도 가설에 의해, T_{i-1}는 깊이가 [log i]이고; 루트 R_{i-1} 및 i 깊이-[log(i+1)] 리프이고, 각각 값 v_0, ..., v_{i-1}을 저장한다.)

[1088] 리프 i가 R_{i-1}의 자손이라면 $R_i=R_{i-1}$이고, 그렇지 않으면 R_i를 R_{i-1}의 부모로 보자. P를 리프 i로부터 노드 R_i까지의 T에서의 (최단) 경로라고하자. P의 모든 노드 j에 대해 j'가 비어 있으면 자신의 형제 j'에 특수 스트링 *e*를 저장한다. 마지막으로, 리프 i(제외됨)로부터 노드 R_i(포함됨)의 순서로 P의 각 노드 s에 대해, v_{s0}과 v_{s1}이 각각 s의 좌측과 우측에 저장된 값인 경우 값 $v_s=H(v_{s0}, v_{s1})$을 s에 저장한다. 이렇게 계산된 값을 자신의 노드에 저장하는 R_i에서 루팅된 T의 하위 트리가 머클 트리라는 것을 쉽게 알 수 있다. 이 머클 트리는 T_i이다.

[1089] 초기에 빈 풀 이진 트리 T가 깊이 3을 가질 때 처음 8개의 연속 머클 트리의 구성이 도 2에서 합성된다. 특히 각 하위 도면 2.i는 특수 스트링 *e*를 가지거나("T_i는 그 노드 아래에서 비어있다"는 것을 나타냄), 또는 수 j{0, ..., i-1}을 갖는(머클 트리 T_j를 구축할 때 노드의 콘텐츠가 마지막으로 변경되었음을 나타내는) 자신의 노드 각각을 마킹함으로써 머클 트리 T_i를 강조한다. T_j에서 마지막으로 변경된 노드의 콘텐츠가 더이상 변하지 않을 것이라는 것을 강조하기 위해, 우리가 구축할 수 있는 머클 트리가 아무리 많더라도 우리는 볼드체로 j를 작성한다.

[1090] 이를 염두에 두고, 우리는 다음과 같이 새로운 블록 B_i를 생성한다. i 번째 블록에서 보호하고자하는 정보 $INFO_i$를 선택한 후에, 우리는 T의 리프 i로 값 $v_i=INFO_i$를 저장하고; 머클 트리 T를 구축하고; 및 다음과 같이 설정하고,

[1091] $S_i=(R_i, auth_i)$,

[1092] 여기서 R_i는 T_i의 루트이고 $auth_i$는 T_i에서 v_i의 인증 경로이다. 그런 다음 새 블록은

[1093] $B_i=(i, INFO_i, S_i)$이 된다.

[1094] 실제로 S_i는 [log i] 스트링으로 구성된다. $auth_i$에서의 각 스트링과 그에 따라 S_i에서의 모든 스트링이 리프 i에 v_i를 저장하는 대신 실제로 256비트 길이가 되도록 보장하기 위해, 우리는 대신 $H(v_i)$를 저장할 수 있다는 것에 유의하라.

[1095] **블록 트리를 이용한 보안 및 입증가능성**

[1096] 누군가가 블록 B_r을 알고 있으며 이전 블록 B_i의 정보 $INFO_i$를 올바르게 배우고 싶다고 가정한다. 머클 트리의 시퀀스, T_0, T_1, ...를 구축할 때 각 트리는 이전 트리를 하위 트리로 포함한다. 사실, 각 트리 T_x의 리프는 이전 리프의 콘텐츠가 그대로 남아 있고 마지막 값이 채워진 리프의 오른쪽에 있는 제1 리프에 새 값이 삽입되므로 각 후속 트리 T_y의 제1 x+1 리프이다. 따라서, $INFO_i$는 머클 트리 T_r의 i번째 리프의 콘텐츠이고, 그의 r번째 리프는 $INFO_r$을 포함하고, 그의 루트 값 R_r은 블록 B_r의 보안 정보인 S_r의 제1 구성 요소이다.

[1097] 또한 B_r을 아는 일부 사람은 또한 S_r 및 R_r을 알고 있다는 것에 유의하라. 따라서, INFOi가 블록 B_i의 정보인 그러한 "사람"에게 증명하기 위해서는, 머클 트리 T_r에서 $INFO_i$의 인증 경로를 그에게 제공하는 것으로 충분하다. 사실, 우리는 이미 그러한 인증 정보를 쉽게 확인할 수 있지만 쉽게 위장할 수는 없다는 것을 알았다!

[1098] 이러한 인증 경로는 d=[log r] 값으로 구성되며(T_r은 깊이가 d+1이므로), 각각 32바이트로 구성된다(H는 32 바이트 출력을 생성하기 때문에)는 것에 유의하라. 따라서, B_r에 대해 B_i의 콘텐츠를 증명하는 것은 매우 쉽다. 언급한 바와 같이, 대부분의 실제 애플리케이션에서 d<40이다.

[1099] **쉽게 이용할 수 있는 정보로부터의 블록 구축가능성**

[1100] 블록 B_i의 일부인 구조적 정보 S_i를 구축하기 위해서는, 머클 트리 T_i에 대한전체로부터의 정보가 필요할 것이다. 결국, $INFO_i$와 리프 i에 저장된 값 v_i는 쉽게 사용할 수 있지만 v_i의 인증 경로, $auth_i$는 이론적으로 쉽게 사용할 수 없는 이전 트리의 노드 콘텐츠를 포함한다. S_i를 구성하기 위해 전체 T_{i-1}을 구해야 한다면, 새로운 블록 B_i를 구성하는 것은 너무 효율적이지 않을 수 있다.

[1101] 그러나 블록체인의 취지에서 볼 때, 각 B_i는 이전 블록 B_{i-1}와 선택된 정보 $INFO_i$에서 쉽게 계산할 수 있다는 것에 유의하라. 실제로 S_i의 각 스트링은 다음 중 하나이다.

[1102] (a) $H(INFO_i)$,

[1103] (b) 고정된 스트링 e,

[1104] (c) S_{i-1}의 스트링, 및

[1105] (d) 상기 유형의 스트링을 소정 방식으로 해싱함으로써 얻어진 스트링.

[1106] 도 3은 두꺼운 경계선을 통해 블록 트리에서의 처음 8개 블록을 구성하기 위해 그 콘텐츠가 $S_i=(R_i, auth_i)$를 계산하기에 충분한 노드를 강조한다. 특히, 각 하위 도면 3.i는 그 내용이 S_i 생성에 충분한 노드를 강조한다. 강조 표시된 각 노드는 유형이 (a), (b) 또는 (c)임을 나타내기 위해 a, b 또는 c로 더 표시된다. 루트 R_i를 포함하여 유형(d)의 노드는 표시되지 않은 채로 있다.

[1107] 요약하면, 블록트리 기반 시스템에서 블록 생성은 매우 효율적이다.

[1108] **10.3 상태 정보의 효율적인 관리**

[1109] 블록트리는 Algorand와 같은 결제 시스템을 포함하는 모든 종류의 애플리케이션에서 블록의 보안 처리를 향상시킨다. 그러나 그러한 시스템에서는 개선으로 인해 큰 이익을 얻을 수 있는 또 다른 측면인 상태 정보가 있다.

[1110] **효율적인 상태 정보의 필요성**

[1111] 라운드 r에서의 공식 상태, S^r는 각각의 현재 공개키 x에 대해, x가 소유한 금액 및 추가 정보 $S^r = \ldots, \left(x, a_x^{(r)}, \ldots\right), \ldots$를 지정하는 튜플(tuple) 목록으로 구성된다는 것을 상기하라. 시스템에서 키가 소유한 금액은 동적으로 변경되어, 우리는 그것을 가능한한 효율적으로 추적해야 한다.

[1112] 지금까지 비트코인에서와 마찬가지로(그의 상태 정보가 상당히 다르지만), 현재 상태는 인증되지 않았지만, 인증된 결제 내역에서 추론할 수 있다. 블록 트리는 라운드 r에서 상태 S^r를 증명하는 기능을 보장하지 않는다. 실제로, 블록 트리를 통해, B^r에 선행하는 블록 B^i의 페이셋 PAY^i는 B^r에 비해 효율적으로 증명될 수 있다. 그러나, 상태 S^r을 증명하기 위해서는 일반적으로 B^r에 선행하는 모든 블록의 페이셋을 명확하게 얻을 수 있어야 한다. 그러므로 증명자(prover) P가 S^r을 알지 못하거나 S^r을 알지만 현재의 가치에 대한 명확한 증거를 받고 싶어하는 다른 사용자나 엔티티들에 대해 S^r을 증명하는 효율적인 방법이 필요하다.

[1113] 특히, 그러한 방법은 새로운 사용자 또는 잠시 동안 오프라인 상태였던 사용자가 현재 시스템 상태를 따라 잡는 것을 쉽고 안전하게 만든다.

[1114] 이제 그런 방법: 상태 트리(statustrees)를 제공하자.

[1115] **상태 트리: 제1 방법**

[1116] 상태 트리 ST^r은 P가 라운드 r-1의 끝에서 상태 S^r의 값을 효율적으로 증명할 수 있게 하는 특수 정보 구조이다.

[1117] 사실, ST^r은 P가 모든 사용자의 상태를 증명할 필요없이(전체 블록 시퀀스 B^0, ..., B^{r-1}를 단독으로 제공하도록 하라) 라운드 r의 시작시 i가 소유하고 있는 정확한 금액 a_i^r을 임의의 가능한 사용자 $i \in PK^r$에 대해 효율적으로 증명할 수 있게 한다. 사실 일부 사용자는 예를 들어, 결제 수신을 고려할 수 있는 사람과 같은 소수의 사용자의 상태를 정확하게 파악하여 그들이 사전에 돈을 갖고 있는지 확인하여 그들과 협상을 시작하도록 하는 것이 중요할 수 있다. 사용자 i가 자신이 소유한 가치 a_i^r에 대한 증거를(예를 들어, 대출을 받거나, 협상에서 진지하게 취해지거나, 구매 주문을 하기 위해) 받는 것이 유용할 수도 있다.

[1118] 먼저 P는 전체 블록 시퀀스 B^0, ..., B^{r-1}를 알고 있고 적어도 사용자와 엔티티 그룹에 의해 널리 신뢰된다고 가정한다. 이 경우, P는

[1119] · 페이셋 시퀀스 PAY^0, ..., PAY^{r-1}을 얻는다(예를 들어, 검색한다).

[1120] · 사용자 세트 PK^r을 계산한다.

[1121] · 제1 n개의 리프가 n 명의 모든 사용자의 상태 정보를 저장하고, 각 리프는 단일 사용자의 정보를 저장하는 머클 트리 T^r(적어도 $n_r=|PK^r|$ 리프로)을 구성한다(다른 리프가 있는 경우, 다른 리프는 리프가 "비어 있음"을 신호처리하는 식별 스트링 e를 저장할 수 있음); 및

[1122] · 바람직하게는 디지털 서명된 T^r의 루트 값 R_r을 적어도 다른 엔티티에서 제공할 수 있다.

[1123] 그러면 어떤 사람이 P에게 일부 사용자 $x \in PK^r$의 상태를 물을 때마다, ℓ_x가 i의 상태 정보를 저장하면 P는 리프 ℓ_x에 저장된 값의 T^r에서의 인증 경로를 그에게 제공한다.

[1124] 중요한 점은, R_r을 계산하고 디지털 서명한 후에, 일부 사용자의 상태에 대한 쿼리에 응답하기 위해 더 이상 P를 사용할 필요가 없다는 것이다. 사실, T^r을 알고 있는 다른 엔티티 P'는 x에 대한 엔티티 V의 쿼리에 응답할 수 있다. 사실, P'는 V에 동일한 디지털 서명된 루트 값 R_r, x에 대한 상태 정보 및 P가 V에 제공한 후자의 정보에 대한 동일한 인증 경로를 제공할 수 있다. 따라서 V가 P를 신뢰하면, 그가 P'를 신뢰하지 않더라도 그는 P'가

제공한 답을 검증할 수 있다. 실제로 보고된 인증 경로는 x의 상태가 머클 트리 T^r의 루트 R^r에 대해 정확하다는 것을 보장하고, R_r이 P에 의해 디지털 서명되었기 때문에, V는 x에 대한 상태 정보의 신뢰성 검증을 검증한다. 그가 P를 신뢰하는 한, V는 P'를 신뢰할 필요가 없다.

[1125] 따라서, P는 R_r의 디지털 서명을 공개(또는 다른 엔티티 P'에 대해 이용 가능하게)하고 다른 사람들(P')이 증명 가능한 방식으로 상태 질문에 대답하게 할 수 있다.

[1126] 이제, 이러한 제1 방법의 아래의 특성들을 강조하고자 한다.

[1127] 1.("굿 뉴스") P는 매 라운드 r마다 T^r을 처음부터 계산할 필요가 없다.

[1128] 실제로, 이전의 머클 트리, T^{r-1}, P를 계산하고 저장했던, 사용자, $x \in PK^r$의 상태가 저장되는 위치는 중요하지 않으므로, 새로운 블록 B^r의 페이셋, PAY^r을 학습한 후에, 그는 T^{r-1}로부터 T^r을 계산할 수 있다. 본질적으로, PAY^r이 k개의 결제를 가진다면, P는 최대 2k 명의 사용자의 상태 정보(즉, PAY^r에서의 각 결제의 지불자 및 수취인의 상태 정보)를 갱신할 필요가 있고, 최초에는 최대 k 명의 신규 사용자의 상태 정보를 삽입할 필요가 있다(간략함을 위해 PAY^r의 각 결제가 최대 한 명 이상의 사용자를 시스템으로 가져올 수 있다고 가정한다).

[1129] 블록 트리의 경우와 마찬가지로, 각 새 사용자의 상태는 다음 빈 리프에 삽입될 수 있다. 또한 P가 기존 사용자 i의 상태를 업데이트할 때마다, P는 i의 상태를 저장하는 리프에서 루트까지, 경로에 대한 내용만 변경할 필요가 있다. 이 경로는 $n = |PK^r|$이면 대부분 $[\log n]$ 길이이고 가능한 한 많은 해시가 필요하다. 이는 기본적으로 새로 추가된 각 사용자에게 동일하게 적용된다. 사용자 수, n이 매우 클 수도 있지만, 주어진 라운드에서 거래하는 사용자의 수, 즉, k는 작을 수 있다. 또한, T^{r-1}로부터 T^r을 갱신하기 위해서는, 오직 $0(k \cdot [\log n])$개의 해시만 필요하다. T^r을 처음부터 계산하는 것이 실제로 더 유리할 수 있고, 이는 일부 n개 해시가 필요하다.

[1130] 2. ("배드 뉴스") 사용자 $x \notin PK^r$에 대해 묻는다면, P/P'는 $x \notin PK^r$을 증명할 수 없다. 실제로, 몇몇 V가 라운드 r에서 사용자 x의 상태에 관해 묻는 경우, P 나 P'는 x가 해당 라운드에서 시스템 내에 없기 때문에 x에 대한 상태 정보가 존재하지 않음을 V에게 쉽게 증명할 수 없다.

[1131] P 또는 P'는 T^r 내의 각 리프, ℓ의 내용을 V에게 지정할 수 있다. 이 글로벌 정보로부터, V는 (a) x의 상태를 포함하는 리프가 없는지 체크할 수 있고; (b) 이들 리프 내용에 대응하는 머클 트리의 루트 값을 재구성 할 수 있고; (c) 이렇게 계산된 값이 P에 의해 디지털 서명된 루트 값 R_r과 일치하는지 확인할 수 있다. 그러나, 이러한 글로벌 정보는 너무 크고 전송이 용이하지 않을 수 있다.

[1132] (적어도 아직은) 존재하지 않는 사용자에 대한 쿼리에 대한 신뢰할 수 있는 답변을 쉽게 제공할 수 없다는 것은 걱정할 필요가 없다. (예를 들어, 전체 사용자 수가 너무 많지 않기 때문이다.) 이 경우, 우리의 첫번째 방법은 훌륭하다. 그러나, 그것이 중요하다면, 다른 방법을 제시한다.

[1133] **상태 트리: 제2 방법.**

[1134] 다시 P가 전체 블록 시퀀스 B^0,, ..., B^{r-1}를 알고 있다고 가정한다. 그러면, 제2 방법에서, P는

[1135] · 페이셋 시퀀스 PAY^0 ,... , PAY^{r-1}를 검색하고;

[1136] · 사용자의 세트, PK^r를 계산하고;

[1137] · 모든 사용자 $i \in PK^r$의 상태 정보 (i, a_i^x)를 계산하고;

[1138] · 제1 엔트리에 따라(예컨대, 그들의 사용자의 사전학적 순서에 따라) n개의 상태-정보 쌍을 정렬시키고;

[1139] · 제1 리프가 PK^r의 제1 사용자의 상태 정보를 포함하고 있고, 제2 리프가 제2 사용자의 상태 등을 포함하는

식의 머클 트리 T^r을 구성하고(추가 리프는, 존재한다면, 스트링 e을 포함);

[1140] · T_r의 루트 값 R_r을 디지털 서명하고 그 서명을 사용 가능하게 만든다.

[1141] 이 제2 방법은 여전히 다른 엔티티 P'(반드시 신뢰할 수 있는 것일 필요는 없음)가 개별 사용자의 상태에 대한 질문에 신뢰성 있게 응답할 수 있게 하지만, 속성 1과 2에 관련된 뉴스를 "반전시킨다". 즉,

[1142] 1'. ("배드 뉴스") T^{r-1}을 사용하여 T^r을 쉽게 생성하는 것이 이전처럼 쉽지 않다.

[1143] 이것은 T^r의 리프가 이제 사용자에 따라 정렬된 사용자의 상태 정보를 포함하고 있고, 이는 새로운 사용자를 삽입하는 것이 문제가 되게 만들기 때문이다. 한 명의 새로운 사용자가, 예컨대, 리프들 중간에 삽입되어야 할 경우 리프의 절반의 내용이 변경되어야 한다.

[1144] 2' ("굿 뉴스") P 또는 P'는 실제로 사용자 $x \notin PK^r$ 가 PK^r에 속하지 않는다는 것을 증명할 수 있다.

[1145] V가 라운드 r에서 시스템에 없는 사용자 x의 상태에 관해 묻는 경우, 신뢰할 수 없는 P'조차도 2개의 연속적인 리프 내용의, P에 의해 인증된 루트 값 R_r에 대한, 인증 경로를 제공할 수 있으며, 리프 중 하나는 사용자 i'에 관한 상태 정보를 저장하고 다른 하나는 i'<i <i"가 되도록 하는 다른 사용자 i"에 대한 상태 정보를 저장한다.

[1146] V는 실제로 두 개의 인증 경로가 두 개의 연속적인 리프의 내용임을 판정할 수 있음을 유의해야 한다. 이는 동일한 값으로 해싱하는 두 개의 상이한 스트링을 찾는 것은 실제로 불가능하기 때문이며, 두 개의 스트링 a와 b가 주어진 때 우리는 $H(a,b) \neq H(b,a)$ 를 가진다. 따라서, h = H (a, b)가 주어지면, 다른 방식보다는 h는 "a 다음에 b"의 해시인 것으로 판정할 수 있다. 이것은 머클 트리 T^r에서 값 v_x의 인증 경로가 값 v_x가 무엇인지를 증명할 뿐만 아니라 그것이 T^r의 리프 x에 저장된다는 것을 나타낸다.

[1147] 예를 들어, 도 1.B.를 살펴보면, 이 머클 트리에서, 값 v_{010}의 인증 경로는 (v_{011}, v_{00}, v_1)이다. 이 인증 경로를 사용하여, v_{010}을 검증하기 위해, 다음 해시를 순서대로 계산한다.

$$\text{(a)}\ \ h_1 = H(v_{010}, v_{011}),$$

$$\text{(b)}\ \ h_2 = H(v_{00}, h_1),$$

$$\text{(c)}\ \ h_3 = H(h_1, v_1),$$

[1148]

[1149] 그런 다음, h_3가 실제로 루트 값 v_ε과 일치하는지 검사한다.

[1150] $H(x,y) \neq H(y,x)$ 이므로,

[1151] 해싱 (a)는 v_{010}이 h_1을 저장하는 어떤 노드의 0-자식에 저장되어 있음을 증명한다.

[1152] 해싱 (b)는 h_1이 h_2을 저장하는 어떤 노드의 1-자식에 저장되어 있음을 증명한다.

[1153] 해싱 (c)는 h_2가 h_3을 저장하는 어떤 노드의 0-자식에 저장되어 있음을 증명한다.

[1154] 그러므로, 우리는 h_3가 루트에 저장되어 있음을 확인했으므로, V_{010}은 리프 010에 저장되어야 하고, 이는 실제 케이스이다.

[1155] 따라서, V가 각각의 사용자 $i \in PK^r$ 에 대해 T^r의 i 번째 리프에 i의 포메이션 상태를 저장했다고 P를 신뢰한다면, 하나의 리프가 사용자 i'<i의 상태 정보를 포함한다는 증거, 및 다음 리프가 사용자 i"> i의 상태 정보를 포함한다는 증거를 볼 때, V는 $i \notin PK^r$ 임을 안전하게 결론 내릴 수 있다.

[1156] (이러한 인증 경로는 많은 값을 공유할 수 있으므로, 그들의 두 인증 경로 전체를 모두 그에게 전송할 필요는 없음을 이해해야 한다.)

[1157] 속성 1'은 중요한 것이 아닐 수도 있다. (예컨대: P는 처음부터 T^r을 완벽하게 구성할 수 있기 때문이다.) 이 경우, 제2 방법은 훌륭하다.

[1158] 그렇지 않으면, P가 T^{r-1}로부터 T^r을 쉽게 구성함 및 사용자 $x \notin PK^r$ 이 실제로 PK^r에 속하지 않음을 쉽게 증명할 수 있는 방법이 필요하다.

[1159] 이제, 그러한 방법을 제공한다.

[1160] **상태 트리: 제3 방법.**

[1161] 검색 트리가, 개념적으로 이진(단지 간략함을 위함) 트리의 노드 내에 정렬된 세트로부터의 값을 동적으로 저장하는 데이터 구조임을 상기하자. 일반적으로 이러한 트리에서 노드는 단일 값을 포함한다(또는 비어 있다). 이러한 데이터 구조에 의해 동적으로 지원되는 작업에는 주어진 값 v에 대한 삽입, 삭제 및 검색을 포함한다. (값 v가 존재하지 않으면, 예를 들어, 반환된 답이 '⊥'이기 때문에 그 값이 존재하지 않는다고 판정할 수 있다.)

[1162] 처음에는 검색 트리가 빈 루트로만 구성된다. 삭제된 것보다 더 많은 값이 삽입되면 트리가 성장한다. n 개의 노드가 있는 2 진 트리의 깊이는 트리가 꽉 찬 경우에 최상으로서 log n이고, 트리가 하나의 경로인 경우 최대 n-1이다. 그러나, 검색 트리의 깊이가 작음은 보장되지 않는다. 최악의 경우, n 개의 특정 노드를 특정 순서로 삽입하면, 검색 트리가 깊이 n으로 구성될 수도 있다.

[1163] "균형" 검색 트리는 현재 저장된 값의 수가 n인 경우 트리의 깊이가 짧은 것, 즉 로그 n임을 보장한다. n개의 노드가 있는 균형 검색 트리에서 위에서 언급한 세 가지 작업 각각은 두 값을 비교하는 것, 두 노드의 내용을 교환하는 것, 노드의 부모/우측 아들/좌측 아들의 내용을 검색하는 것과 같이 O(log n)개의 기본 단계로서 수행될 수 있다.

[1164] 균형 검색 트리의 몇 가지 예가 지금까지 알려져 있다. 특히 잘 알려진 예는 AVL 트리 및 B-트리이다. 일부 예에서 값은 트리의 리프에만 저장될 수 있다(다른 모든 노드는 실제로 트리에 저장되는 경우 주어진 값을 찾을 수 있도록 하는 "방향 정보"를 포함한다). 다른 예에서, 값은 임의의 노드에 저장될 수 있다. 아래에서는 보다 일반적인 경우를 가정하고 간략함을 위해 균형 검색 트리 작업은 잘 알려진 결정론적 알고리즘이라고 가정한다.

[1165] PK^r내의 모든 사용자의 상태 정보를 얻은 입증자 P가 라운드 r의 균형 상태 트리 T^r을 처음부터 구성하고자 하는 순간을 가정한다. 그러면, 그는 다음과 같이 행동한다.

[1166] · P는 $n = |PK^r|$ 노드가 있는 균형 검색 트리 T^r을 구성하여 사용자를 PK^r에 저장한다.

[1167] 이 작업은 최대 O(log n)개의 기본 단계를 수행하는데, 왜냐하면 최악의 경우 각각의 삽입은 O(log n)개의 기본 단계를 취하기 때문이다. (실제로는, P는 T^r에 사용자를 삽입하기 전에 PK^r 내의 사용자를 정렬시킴으로써 약간의 추가적 효율을 얻고자할 수도 있다.)

[1168] · P는 T^r에서 각각의 저장된 사용자 i를 i의 상태 정보 (i, a_i^x) 로 대체한다.

[1169] 즉, 그는 PK^r 내의 사용자의 상태 정보를 T^r에 저장하여, 사용자 i의 상태 정보에 대한 모든 삽입/삭제/검색이 사용자 i에 대한 삽입/삭제/검색을 통해 수행될 수 있도록 한다.

[1170] (즉, T^r은 노드에 저장된 제1 엔트리를 통해 검색 가능한 균형 트리이다.)

[1171] · P는 모든 논-리프 노드가 정확히 2명의 자식을 갖도록 T^r을 "완성"한다.

[1172] 2진 트리에서 각 노드는 최대 2명의 자식을 갖는다. 따라서, 리프가 아닌 노드 x는 하나의 자식만 가질 수 있다. 보편성을 잃지 않으면서, 그것이 x의 왼쪽 자식, x0라고 가정하자. 이 경우, P는 개념적으로 x에게 그 안

에 그가 e라는 스트링을 저장한 우측 자식 x1을 제공한다.

[1173] · P는 T^r의 각 노드 x에 리프로부터 위쪽으로 그렇게 계산된 해시 값 hv_x를 연관짓는다.

[1174] x가 리프이면, hv_x는 x에 저장된 값 v_x의 해시이다. 즉, $hv_x = H(v_x)$이다.

[1175] x가 값 v_x를 저장하는 깊이 d의 노드이면, $hv_x = H(hv_{x0}, hv_{x1}, H(v_x))$이다.

[1176] 마지막으로 T^r의 루트 ε에 연관된 해시 값은 $hv_\varepsilon \triangleq R_r$이다.

[1177] · P는 R_r에 디지털 서명한다.

[1178] 우리는 이렇게 얻어진 트리 T^r를 머클-균형-검색-트리라 지칭한다. 이러한 T^r은 균형 검색 트리이므로, 검색/삽입/삭제 알고리즘은 T^r 상에서 행해진다. 동시에 T^r은 머클 트리이지만 일반화된 머클 트리이다. 일반 머클 트리는 정보 값, 즉 보호될 필요가 있는 값을 리프에서만 저장하고, 내부 노드에는 보안 값, 즉, 트리를 "보호"하는 데 사용되는 해시 값만 저장한다. 머클 트리 T^r은 각 노드 x에 v_x로 표시되는 정보 값과 hv_x로 표시되는 보안 값을 모두 저장한다.

[1179] 루트 보안 값 R_r과 관련하여, 정보 값 v_x가 실제로 무엇인지의 증명은 hv_{x0}, hv_{x1}, $H(v_x)$, 및 x에서 T^r의 루트까지의 경로에 있는 노드의 형제인 모든 노드 y에 대한 값 hv_y으로 (상향순으로) 구성된 인증 경로를 포함한다. .

[1180] 이제, P가 T^r을 처음부터 계산할 필요가 없다는 것을 보이고자 한다. 실제로 P가 사용 가능한 이전 라운드의 머클-균형-검색-트리 T^{r-1}을 가진다고 가정한다. 그러면, 새로운 블록 B^r을 얻은 후, P는 B^r의 페이셋, PAY^r 내의 각각의 결제 $\wp$에 대해 다음과 같이 동작한다.

[1181] $\wp$가 기존 사용자가 소유한 금액을 수정하면, P는 그 사용자의 상태 정보가 저장되어 있는 노드 x에서 해당 사용자의 상태 정보를 업데이트한 다음, 트리를 재합병한다. 이것은 단순히 x에서 루트까지의 경로를 따라 hv_y 값을 다시 계산하는 것을 수반하며, 따라서 n이 사용자 수라면 최대 [log n]번 해시한다.

[1182] $\wp$가 초기 금액 a_i을 가진 새 사용자 i를 가져 오면, P는 상태 정보 (i, a_i^r)를 T^r에 삽입한다. T^r이 균형 탐색 트리이기 때문에, 오직 O(log n)개의 기본 단계만 수반하고, 최대 대수적으로 많은 노드에 영향을 미친다. 그 후, P는 트리를 재 합병한다.

[1183] T^r을 구성한 후에, P는 R_r의 그의 디지털 서명을 공개하거나, 그것을 P'를 신뢰하거나 신뢰하지 않을 수 있는 다양한 검증자 V로부터의 쿼리를 처리하는 P'에게 제공한다.

[1184] 사용자 i에 관한 V의 쿼리에 응답하기 위해, P'는 다음과 같이 행동한다.

[1185] · P'는 사용자 i에 대해 T^r을 검색하는 것과 동일한 알고리즘을 실행한다.

[1186] 이 검색은 T^r의 최대 O(log n)개 노드의 내용을 검색하는 것을 포함하며, 여기서 각각의 검색된 내용이 그것의 내용이 다음에 검색될 필요가 있는 노드를 결정한다.

[1187] · 알고리즘이 노드 x의 내용 v_x를 검색할 때마다 P'는 R_r에 대한 v_x의 증거를 제공한다.

[1188] 따라서, V는 R_r의 P의 디지털 서명 및 수신된 증거를 사용하여, P'에 의해 제공된 모든 노드 x의 내용이 정확한지 검사할 수 있다. 따라서, V는 실제로 머클 트리 T^r에서 동일한 검색 알고리즘을 실행하여 $i \in PK^r$일 때 사용자 i의 상태 정보를 정확하게 검색한다. 그렇지 않고 $i \notin PK^r$이면(즉, 검색 알고리즘이 심볼 ⊥/

스트링 e를 반환하면), V는 i가 PK^r 내의 사용자가 아닌 것으로 확신한다.

[1189] **Algorand에서 신뢰할 수 있는 P의 실현.**

[1190] 많은 사람이 신뢰할 수 있는 개인 입증자 P가 존재하지 않는 경우, 입증자를 "구성"하는 것이 중요해진다.

[1191] Algorand에서, 라운드 r의 모든 단계 s에서, $SV^{r,s}$내의 검증자의 (가중치 적용된) 대다수가 정직하다는 것이 보장될 수 있음에 유의해야 한다. 따라서, 이제, 우리는 검증자들이 P의 업무를 수행할 수 있음을 보장한다!

[1192] 더 정확하게,

[1193] · 블록의 다른 필드와 더불어, 라운드 r의 잠재적인 검증자, i는 또한 (예를 들어 위에서 논의된 세 가지 방법 중 하나를 사용하여) 상태트리 T^r을 생성한다. 그는 이것을 쉽게 할 수 있는데, 일반적으로 그는 지금까지의 블록체인을 알고 있기 때문이다: B^0, ..., B^{r-1}.

[1194] · i는 T^r의 루트 값 R_r을 또한 포함하는 블록 B_i^r를 제안한다.

[1195] 예를 들어, $B_i^r = (r, PAY^r, SIG_i(Q^{r-1}), R_r, H(B^{r-1}))$.

[1196] · 검증자(특히, 라운드 r의 제2 단계의 검증자)는 라운드의 리더가 제안한 블록이 유효한지 확인하는데, 이것은 4번째 컴포넌트, 즉, 상태트리 T^r의 루트 값 R_r이 유효한지 체크하는 것을 포함한다.

[1197] 실제로, 이들 검증자는 지금까지의 블록 시퀀스 B_0, ..., B_{r-1}을 알고 있고, 따라서 식별된 리더에 의해 제안된 블록이 채택된다고 가정하면, 라운드 r 후에 상태 정보가 무엇인지를 판정할 수 있다.

[1198] · BA 합의가 성립된 후, 라운드 r의 공식 블록은 올바른 루트 값, R_r을 포함하고, B^r은 충분한 보증서를 가진 충분한 수의 검증자에 의해 디지털 서명되므로, B^r의 인증서는 또한 R_r을 증명한다.

[1199] 새로운 공식 블록은 비어 있을 수 있음($B^r = B_\varepsilon^r$)에 유의해야 한다. 이러한 빈 블록은 다음과 같은 형태이다.

[1200] $B_\varepsilon^r = (r, PAY^r, SIG_i(Q^{r-1}), R_{r-1}, H(B^{r-1}))$

[1201] 이는 새로운 블록이 빈 블록인 경우 모든 사용자의 상태 정보는 변경되지 않는다는 사실에 대응한다.

[1202] 요약하면, R_r의 P의 디지털 서명은 B^r의 인증서로 대체된다. 어떤식으로든 R_r이 인증되기 때문에, 신뢰할 수 없는 입증자 P'조차도 이전과 마찬가지로 R_r에 대하여 라운드 r에 있는 모든 사용자 i의 상태 정보를 증명할 수 있다.

[1203] 생성된 새로운 블록을 적시에 배우는 모든 사람은, 채택된 상태트리가 "쉽게 업데이트 가능한" 유형이라면 이전 T^{r-1}에서 T^r을 보다 쉽게 구성할 수 있다.

[1204] **11 대리 검증자 및 잠재적 검증자**

[1205] **대리 검증자.**

[1206] 사용자 i가 $SV^{r,s}$의 멤버로 선정될 확률은 다른 사용자가 i에게 "투표"하는 돈에 기초할(다시 예를 들면, 비례할) 수 있다. 사용자, U는 평소와 같이 그가 결제하고 결제를 받는 모든 결제의 제어를 유지하기를 원할 수 있다. 그러나 그는 다른 사용자 i에게 리더 및/또는 검증자 역할을 할 권리와 의무를 위임하고자 할 수도 있다. 이 경우에, 그리고 이러한 목적으로만, 그러한 사용자 U는 그가 리더/검증자의 목적으로 i를 그의 대리인으로 만들기를 원한다고 나타낼 수 있다. 사용자 U는 실제로 그의 리더/검증자의 의무를 하나 이상의 사용자 i에게 위임할 수 있다. 잠시 동안 일반성을 잃지 않고, U가 리더 및 검증자의 목적을 위해 하나의 대리인을 지

명한다고 가정한다.

[1207] U가 라운드 r에서 자신의 대리인으로 i를 선출할 수 있는 몇 가지 방법(바람직하게는 디지털 서명식)이 존재한다. 예를 들어, 그는 추가적 기계를 도입하지 않도록(이것을 배제하는 것은 아님) (예컨대, P의 비-민감 필드 I를 이용하여) i에게로의 결제(P)를 통해, 또는 다른 사용자에게로의 결제(P')를 통해 그렇게 할 수 있다. 일단 이러한 결제 P 또는 P'이 블록 B^r의 페이셋 PAY^r에 삽입되면, 큰 커뮤니티는 U'의 선택을 실현하고, i가 U의 대리인으로서 역할하는 것이 유효하게 된다.

[1208] U가 라운드 r에서 자신의 대리인으로 i를 선택한 경우, 이 선택은 U에 의해 만들어진 이전의 것보다 우선하며, U가 다른 선택을 할 때까지 i는 U의 대리인으로 유지되는 것이 바람직하다.[48] 또한, 사용자 U는 그가 시스템에 들어가는 순간부터 대리인으로서 i를 선출할 수 있다. 예를 들어, U가 다른 사용자로부터의 결제 $\wp$를 통해 시스템에 입장할 계획이라면, U는 U가 그의 대리인으로서 i를 선택했음을 나타내는 U의 서명을 $\wp$내에 포함시킬 것을 그 사용자에게 요청할 수 있다.

[1209] (48: 물론, 여기 및 다른 곳에서, 모호성 및 타이(tie)는 미리 지정된 방법으로 깨진다. 예를 들어, PAY^r이 U가 자신의 모든 돈을 잠재적인 검증자 i에게 투표한다는 것을 나타내는 U의 하나의 서명과, i가 상이한 잠재적인 검증자 j에게 자신의 모든 돈을 투표한다는 것을 나타내는 다른 서명을 포함한다면, U의 대리인 선택은 무시될 수 있고, 또는 대안으로서, 사전학적 순서에 따라 i에 대응하는 서명된 진술이 j에 대응하는 것보다 앞선다면 U는 실제로 i에게 투표한다.)

[1210] U (및 아마도 다른 사용자)가 i에게 그렇게 투표한 돈과 i가 직접 소유한 돈은 동등하게 취급될 수 있다. 예를 들어, 라운드 x에서 소유한 돈에 따라, PK^x내의 사용자로부터 검증자가 선택되는 경우, U(및 대리인에게 그들의 돈을 "투표"할 것을 선택한 다른 모든 사용자)는 이 선택의 목적을 위해 0의 돈을 가진 것으로 간주되었을 것이고, 반면 i가 선택됨에 따른 돈은 $a_i^x + VM_i^x$이 될 것이고, 여기서 a_i^x는 i가 x 라운드에서 개인적으로 소유한 돈이고, VM_i^x는 x 라운드에서 (모든 사용자에 의해) i에게 "투표된" 총 금액이다. 예를 들어, U가 그 라운드에서 i에게 자신의 돈을 투표한 유일한 사용자이면, $VM_i^x = a_U^x$이다. 라운드 x에서 i에게 그들의 돈을 투표한 사용자의 세트가 S라면, $VM_i^x = \sum_{j \in S} a_j^x$이다. 물론, 그가 소유하고 라운드 x에서 그에게 투표된 금액에 따라 i를 선택하는 한 방법은 아래의 확률로 비밀 암호 분류를 통해 i를 선택하는 것을 포함한다.

[1211] $$\frac{a_i^x + VM_i^x}{\sum_{k \in PK^x} a_k^x}$$

[1212] i가 직접 소유한 돈을 i에게 투표된 돈과 다르게 취급할 수도 있다. 예를 들어, i는 금액 $a_i^x + c \cdot VM_i^x$에 따라 선택될 수 있고, 여기서 c는 주어진 계수이다. 예를 들어, c = 0.5일 때, i가 직접 소유한 돈은 그에게 투표된 돈의 두 배로 카운트한다.

[1213] 사용자 U가 자신이 소유한 돈의 일부만 i에게 투표하는 것도 가능하다. 예를 들어, U는 자신의 돈의 3/4 만 i에게 투표할 수 있으며, 자신을 $SV^{r,s}$로 선택되게 하여 밸런스를 계산할 수 있다. 이 경우, U는 VM_i^x에게 $0.75a_U^x$ 만 기여하고, U는 $0.25a_U^x$의 확률로 $SV^{r,s}$에서 선택된다.

[1214] 그가 $SV^{r,s}$에 속하도록 선택되었다는 결과로서 사용자 i에게 보상이 주어질 수 있으며, i는 그에게 그들의 돈의 일부를 투표했던 사용자들과 보상을 공유할 수 있다. 예를 들어, i는 그가 블록의 리더가 되면 보상 R을 받을 수 있다. 이 경우이 보상의 일부를 U에게 제공할 수 있다. 예를 들어, U가 i에게 투표한 돈이 m이라면, i가 U

에게 지불한 R의 비율은 다음과 같다.

[1215] $$\frac{m}{a_i^x + VM_i^x}$$

[1216] 대리인을 선택하고 변경할 수 있는 사용자의 능력은 대리인을 정직하게 유지하는 데 도움이 될 수 있다. 예를 들어, U가 자신의 돈을 i에게 투표했지만, i가 블록 B^r의 리더일 때마다, PAY^r의 페이셋이 너무 자주 비거나 또는 "빈약"해진다면, U가 그의 대리인을 변경할 수 있다.

[1217] 보통 사용자, U는 다수의 대리인를 지정하여 그들 각자에게 자신의 돈을 상이한 비율로 투표할 수 있다. 물론, U가 실제로 가진 것보다 많은 돈을 투표하는 것을 방지하기 위해 미리 지정된 절차가 사용될 수 있다.

[1218] 대리인을 선택하는 다른 방법은 사용자 U가 별도의 공개 - 비밀 디지털 서명 쌍 (pk'_U, sk'_U)을 생성하고, 리더/검증자 선택을 위해 U를 대리할 권한을 (예를 들어, 블록 체인 내의 블록에 들어가는 디지털 서명을 통해) pk'_U로 전달할 수 있다. 즉, pk'_U는 U를 대신하여 직접 지불 할 수는 없지만 (실제로 pk'_U는 절대로 직접 돈을 가질 수 없다), U를 대신하여 리더 또는 검증자의 역할을 할 수 있으며, 따라서 관련 라운드 x에서 그 U의 돈에 따라 리더/검증자로서 선택될 수 있다. 예를 들어, pk'_U는 아래의 확률로 선택될 수 있다.

[1219] $$a_U^x / \sum_{k \in PK^x} a_k^x$$

[1220] (이것은 U가 자신의 모든 리더/검증자 권리/권한을 위임한 것으로 가정하지만, 물론 위에서 설명한 바와 같이 U는 자신의 권리/권한의 일부만 위임할 수도 있다.) 다른 사용자 또는 엔티티 i를 자기 대신 리더/검증자로 위임하기 위해, U는 i에게 sk'_U를 준다. 이것은 U와 i 간에 (어떤 결정 비율이든) pk'_U가 "벌" 수 있는 보상을 나누는 방법을 매우 명확하게 만드는데, 이는 이전에 논의된 것처럼 U가 i의 선택 확률에 기여하게 하는 것이 아니라, 그것이 직접 선택되어야 하는 pk'_U 자체이기 때문이다.

[1221] U가 스스로 (pk'_U, sk'_U)를 생성할 필요가 없음에 유의해야 한다. 예를 들어, i는 (pk'_U, sk'_U)를 생성할 수 있고, U가 그의 대리인로 i를 선택하기를 원한다면 서명을 위해 pk'_U를 U에게 준다.

[1222] 특히, 사용자 U는 위에서 논의된 임의의 접근법에서 그의 대리인로서 은행 i를 선택할 수 있다. 대리인를 선택하는 한 가지 이점은 후자가 일반 사용자보다 훨씬 빠르고 안전한 통신 네트워크를 가질 수 있다는 것이다. 모든 일반 사용자가 적절한 대리인을 선택했거나(또는 선택하도록 요구받았던) 경우, 블록 생성이 훨씬 빨라졌다. 그런 다음 블록은 일반 사용자가 액세스할 수 있는 네트워크에서 전파될 수 있다. 그렇지 않고, i가 U를 대리한다면, i는 새로운 블록을 U에 직접 줄 수도 있고, 그가 관리하는 결제가 블록체인으로 들어갔다는 증거를 U에게 제공할 수도 있다.

[1223] **잠재적 검증자.**

[1224] 지금까지 Algorand에서, 각 사용자 i는 일부 라운드 r에서 검증자의 리더로 선택될 수 있다. 그러나, 당업자는 상기 사항이 제한 사항이 아니라는 것을 쉽게 이해할 것이다. 사실, Algorand는 언제든지 비허가 방식으로 가입할 수 있고, 평소와 같이 결제(더 일반적으로 거래)를 하고 받을 수 있는 잠재적인 검증자의 특수 클래스인, 사용자 세트를 가질 수 있는데, 이로부터 라운드 리더와 검증자가 선발된다. 이 두 세트는 중첩될 수 있으며 (이 경우 최소한 잠재적인 검증자도 지불을 만들고 받을 수 있다), 또는 분리될 수도 있다(이 경우 잠재적인 검증자는 선택된 경우 리더 또는 검증자의 역할을 할 수 있다). 이전 섹션에서 설명한 Algorand의 의미에서, 각 사용자는 잠재적인 검증자이다. Algorand는 두 개의 별도 금액을 가지는 사용자 또는 잠재적인 검증자 i를 가질 수 있고, 그 중 하나만이 i가 리더 또는 검증자로서 선택되는 것을 의미한다.

[1225] 잠재적인 검증자의 클래스는 허가 받을 수 있다. 이 경우, 잠재적 검증자의 주어진 세트 S 중에서 검증자/리더 i를 선택할 확률은 i가 소유하고 있는 금액에 의존할 필요가 없다. 예를 들어, i는 S로부터 암호화된 정렬을 통해 균일한 확률로 선택될 수 있다.

[1226] 대안으로서, 모든 잠재적 검증자가 항상 선택될 수 있다(그리고/또는 그 중 하나만 라운드 리더로 선택된다). 이 경우, BA^* 프로토콜을 사용하여 새 블록에 대한 합의를 도출할 수 있으며, 또는 더 일반적으로, 효율적이고 바람직한 플레이어-교체 가능한 프로토콜을 사용할 수 있다.

[1227] **12 허가식 Algorand**

[1228] 이 섹션에서는 프라이버시와 추적성의 균형을 유지하고, 새로운 인센티브를 제공하며, 은행이나 기타 외부 엔티티를 위한 새로운 비통제 역할을 제공하는 Algorand의 허가 버전에 대해 논의한다. 이 허가식 버전은 아래에 상기된 고전적 개념에 따른다.

[1229] **12.1 디지털 인증서**

[1230] 시스템에 사용자를 등록할 수 있는 권한을 가진 당사자 R의 공개키 pk가 공개적으로 알려져 있다고 가정한다. 그 다음, 사용자 i를 식별하고 공개키 pk_i가 실제로 i에 속하는지 확인한 후, R은 pk_i가 시스템의 합법적인 공개키일 뿐만 아니라 i에 대한 몇가지 적합한 추가 정보, $info_i$가 유지된다는 것을 보장하는 디지털 인증서 C_i를 발급한다. 이 경우 인증서는 본질적으로 다음과 같은 형식을 갖는다.

[1231] $$C_i = SIG_R(R, pk_i, info_i).$$

[1232] $info_i$에 명시된 정보는 매우 다양할 수 있다. 예를 들어, 그것은 시스템에서의 i의 역할 및/또는 인증서 발행일을 명시할 수 있다. 또한, 그것은 만료 날짜, 즉, 더 이상 C_i에 의존해서는 안되는 날짜를 지정할 수도 있다. 그러한 날짜가 지정되지 않으면 C_i는 만료되지 않는다. 인증서는 상이한 응용 분야에서, 상이한 방식으로 오랫동안 사용되어 왔다.

[1233] **12.2 (비 만료) 인증서를 갖는 Algorand**

[1234] Algorand에서, 각각의 사용자 $i \in PK^r$는 적절한 엔티티 R에 의해(예를 들어, 미리 지정된 세트 내의 은행에 의해) 발행된 디지털 인증서 C_i를 가질 것을 요구받을 수 있다. 다른 사용자 j에게 라운드-r의 결제 $\wp$를 할 때, i는 C_i 및/또는 C_j를 $\wp$와 함께 전달하거나 $\wp$ 자체에 그것 중 하나 또는 둘 모두를 포함시킬 수 있다. 즉, 기호로 나타내면 다음과 같다.

[1235] $$\wp = SIG_i(r, r', i, j, a, C_i, C_j, I, H(\mathcal{I})) \ .$$

[1236] 후자의 관행을 가정하면,

[1237] · 결제 $\wp$는 한 라운드에서 유효하다고 간주되므로, 그것이 그것의 지불자와 그것의 수취인 모두의 인증서를 포함하는 경우에만 PAY^r을 입력할 수 있다.

[1238] 또한 인증서의 만료 날짜가 $r = [t_1, t_2]$인 경우, t_2는 가장 빠른 만료 날짜보다 작아야 한다.

[1239] · 동시에 그러한 결제가 PAY^r에 속하면 그것의 대응 금액 송금은 무조건 실행된다.

[1240] **비 만료 인증서의 역할.**

[1241] 사용자 i의 인증서, C_i가 만료되면, 또는 i가 "좋은 지위"에 있다면 그것의 만료 이전에도, R은 더 늦은 만료 날짜를 가지는 새로운 인증서 C_i'를 발행할 수 있다.

[1242] 그러므로, 이 경우 R은 i의 돈에 대한 중요한 통제권을 가진다. 개인적인 사용을 위해 그것을 압수할 수는 없

지만(그렇게 하면 i'의 디지털 서명을 위조할 수 있어야 하기 때문에), 그는 i가 그것을 쓰지 못하게 할 수 있다. 실제로, C_i의 만료 날짜 다음의 라운드 r에서, i의 결제는 PAY^r로 입력되지 않을 수 있다.

[1243] R의 이러한 권한은 사용자의 전통적인 은행 계좌를 동결시키는, 예컨대, 정부와 같은 적절한 엔티티 E의 권한에 해당함에 유의해야 한다. 사실, 전통적인 환경에서, 이러한 E는 심지어 i의 돈을 도용(appropriate)할 수도 있다.

[1244] 암호화폐의 주된 매력은 정확하게 임의의 엔티티 E가 사용자를 그의 돈으로부터 분리할 수 없다는 것이다. 따라서, 이러한 불가능성이 모든 인증서가 비만료성인 Algorand의 인증서 기반 버전에서도 계속 유지된다는 것을 강조하고자 한다. 사실, 라운드 r에서 다른 사용자 j에게 결제를 행하고자 하는 사용자 i는 항상 만료되지 않는 인증서 C_i와 C_j를 j에 대한 라운드 r 결제 $\wp$ 내에 포함시킬 수 있으며, 라운드의 리더 ℓ^r가 정직하다면 $\wp$는 PAY^r 내에 표시될 것이다. 요약해서 말하면,

[1245] *비만료 인증서는 사용자를 그의 돈으로부터 분리하는 데 사용할 수 없지만, 실제로 다른 목표를 달성하는 데 매우 유용할 수 있다.*

[1246] **12.3 (비 만료) 인증서를 통한 불법 행위 방지**

[1247] 전통적인 수표로 이루어진 결제의 지불자 및 수취인은 그 수표를 소지한 모든 사람에 의해 쉽게 확인할 수 있다. 따라서 수표는 돈세탁 또는 기타 불법 행위에 이상적이지 않다. 적절한 등록 에이전트에 의해 발행된 디지털 인증서는 오직 몇몇 주어진 엔티티만이 주어진 공개키 pk_i의 소유자 i를 식별할 수 있고, 오직 적절한 환경 하에서만, i가 그가 원하는 결제를 하지 못하도록 방지함을 보장하기 위해 Algorand에서 사용될 수 있다. 간단한 예를 살펴 보자.

[1248] 시스템에 복수 등록 권한이 있을 수 있다. 단지 명확함을 의해, 그들을 공개키를 보편적으로 알고 있는 다른 상위 기관(일반적으로 인증서 체인을 통해 인증된 경우)에 의해 공개키를 보편적으로 알게된 승인된 은행으로 하고, G는 정부로 불리는 특수 엔티티라 한다.

[1249] 시스템을 디지털 공개 키의 소유자로 참여시키려면 i는 승인된 은행 중 하나로부터 인증서 C_i를 받아야 한다. 그것을 얻기 위해, 공개 비밀 서명 쌍(pk_i, sk_i)을 생성한 후, i는 승인된 은행 B에게 pk_i에 대한 인증서를 발행할 것을 요청한다. 그러한 인증서를 발급하기 위해서, B는 i를 식별하여 몇몇 식별 정보 ID_i를 생성해야 한다.[49] 그런 다음, B는 $H(ID_i)$를 계산하고, (바람직하게는) 그것을 인증서의 별도 필드로 만든다. 예를 들어, 추가 정보를 무시하면, B는 다음을 계산하고 i에게 제공한다.

[1250] $$C_i = SIG_B(B, pk_i, H(ID_i)).$$

[1251] (49: "오버보드(overboard)"로 가면, I_i는 i의 이름과 주소, i의 사진, (디지털인 경우) i의 동의의 디지털 서명(또는 i는 서명된 동의서와 함께 사진을 찍을 수 있음), 및 I_i에 포함되기 위해 디지털화된 사진을 포함할 수 있다. 자체 보호 및 i의 보호를 위해, 은행은 I_i가 실제로 정확함을 증언하는 i의 서명을 얻고 유지할 수 있다.)

[1252] H는 랜덤 오라클이므로, 아무도 C_i로부터 소유자의 신원을 복구할 수 없다. 오직 은행만이 pk_i의 소유자가 i임을 안다. 그러나 정부가, 말하자면, 결제 $\wp = SIG_{pk_i}(r, pk_i, pk_{i'}, a, C_i, C_{i'}, I, H(\mathcal{I}))$의 지불자를 조사하기를 원하면, 그것은 $\wp$로부터 관련 인증서, C_i 및 C_i를 발행했던 은행 B를 모두 검색한 후, 적절한 권한으로(예컨대, 법원 명령으로), B에게 요청하거나 요구하여 i의 정확한 식별 정보, ID_i를 얻어낸다.

[1253] H가 충돌-탄력적(collision-resilient)이기 때문에, 은행은 인증서에 원래 삽입된 것과 상이한 식별 정보 조각 ID_i'를 밝힐 수 없음에 유의해야 한다. 대안으로서, $H(ID_i)$ 대신, 암호법의 용어로 ID_i에 대한 임의의 "약속"을 사용하면 충분하다.

[1254] C_i에, 고유하게 디코딩 가능한, ID_i의 암호, E(ID_i, 바람직하게는 개인 또는 공개 키 암호화 체계 E를 이용)를 저장하기 위한 한가지 특별한 대안이 있다. 특히 ID_i는 은행 B만 알고 있는 키로 암호화될 수 있다. 기호로 하면, $E(ID_i) = E_B(ID_i)$. 이렇게 하면, 정부는 식별 정보 ID_i를 복구하기 위해 B의 도움이 필요하다.

[1255] 대안으로, E는 공개키 암호화 방식일 수 있고, ID_i는 대응하는 해독 키를 유일하게 알고 있는 정부의 공개 키로 암호화될 수 있다. 기호로 하면, $E(ID_i) = E_G(ID_i)$. 이 경우 정부는 ID_i를 복구하기 위해 B의 도움을 받을 필요가 없다. 실제로 지불자의 신원과 모든 결제의 수취인은 G에게 투명하다. 그러나 ID_i로부터 $E_G(ID_i)$를 계산했던 정부 및 은행 이외에는, 누구도 $E_G(ID_i)$로부터 ID_i를 알아낼 수 없다. 또한, 암호 $E_G(ID_i)$가 확률론적이라면, pk_i의 소유자 i를 정확하게 추측했던 사람도 자신의 추측을 확인할 수는 없을 것이다.

[1256] 일단 인증서 C_i가 발행되면, B나 G가 i의 디지털 서명에 대한 통제권을 가지지 않는다는 것을 강조하고자 한다. 왜냐하면, 오직 i만이 pk_i에 해당하는 비밀 키를 알고 있기 때문이다. 또한 오직 $H(\mathcal{I})$만이 $\wp$의 일부이기 때문에, B나 G는 i의 결제 $\wp$의 민감 정보, $\mathcal{I}$를 이해할 수 없다. 마지막으로, B나 G는 결제 $\wp$ 처리에 관여하지 않으며, 무작위로 선택된 검증자들만이 관여한다.

[1257] 요약하면, 비트코인과 유사한 시스템에 대한 논의된 법 집행 우려는 사용자의 개인 정보를 완전히 희생하지 않고도 안심할 수 있다. 사실, B와 G를 제외하고, i는 그가 비트코인 및 유사한 시스템에서 즐기는 것과 동일한 (의사(pseude)) 은행, 상인, 사용자 등 다른 것에 대한 익명성을 계속 즐긴다.

[1258] 마지막으로, 보다 정교한 암호화 기술을 사용하여 적절한 상황에서 추적성을 유지하면서 사용자 프라이버시를 높일 수도 있다.[50]

[1259] (50: 예를 들어, 마지막 시나리오에 초점을 맞추면, i가 은행 B로부터 인증서 $C_i = SIG_B(B, pk_i, E_G(ID_i))$를 얻으면, i가 상이한 pk_i'에 대한 다른 인증서 $C_i' = SIG_B(B, pk_i', E_G'(ID_i))$를 얻는 것도 가능하지만, B는 ID_i가 $E_G'(ID_i)$의 복호임을 더 이상 알 수 없다.)

[1260] **12.4 은행의 새로운 역할**

[1261] Algorand에서 키 인증은 은행의 새로운 역할이다. 이는 은행 B가 고객을 위해 쉽게 수행할 수 있는 역할이다. 사실, B는 이미 그것들을 잘 알고 있으며, 일반적으로 때때로 그것들과 상호 작용한다. 그들의 고객 중 한 명의 키, i를 인증함으로써, B는 간단하지만 중요한(그리고 아마도 재정적으로 보상받는) 서비스를 수행한다.

[1262] 은행 B가 Algorand에서 가질 수 있는 또 다른 역할은, 고객 i에 의해 적절하게 승인된 B가 i의 전통적인 은행 계좌에서 i가 Algorand에서 소유한 디지털 키인 디지털 pk_i로 돈을 이체할 수 있다는 것이다. (실제로, B 및 다른 모든 은행은 "동일한 환율로" 그렇게 하고 있고, Algorand는 국가 통화를 기반으로 매우 분산되고 편리하며 자체 규제되는 결제 시스템으로 사용될 수 있다.) 은행 B가 돈을 pk_i로 이체하는 한 방법은 B가 Algorand에서 소유하고 있는 디지털 키 pk_B에서 pk_i로의 결제를 하는 것이다. 실제로 은행들은 그들의 고객보다 더 쉽게 국가 화폐를 공개 환율로 Algorand로 변환할 수 있다.[51]

[1263] 마지막으로, 은행은 종종 신뢰를 받기 때문에, 그가 행하는 결제의 완전한 통제권을 갖는 사용자 i는 은행 B를 자신을 대신할 검증자로 위임하여 B와 그의 검증 인센티브를 공유하기를 원할 수 있다.[52]

[1264] (51: 한 단계 더 들어가면, 정부 또한 Algorand에서 돈을 인출할 수 있으며, 특히 그것을 은행으로 이체할 수

있고, 또는 은행이 충분히 규제를 받으면 특정 파라미터 내에서 Algorand 화폐를 발생시킬 수 있다.

[1265] 52: "대표적인" 메커니즘은 부록에 설명되어 있다. 이 섹션에서는 특별히 구성된 통신 네트워크를 갖는 Algorand의 구현을 다룬다. 그러나 여기에 설명된 위임 메커니즘은 기본 네트워크가 무엇이든 상관없이 채택될 수 있다.)

[1266] **12.5 오직 소매점으로부터의 보상**

[1267] 등록 기관이 은행이든 아니든, 그리고 법 강화 우려가 처리되었든 아니든, Algorand의 허가된 전개는 주어진 키 pk_i가 상인에 속한다는 것을(예컨대, 그것의 인증서 C_i 내에서) 식별할 수 있도록 한다.

[1268] 현재 신용 카드 결제를 수락하는 상인은 신용 카드 회사에 거래 수수료를 결제해야 한다는 것을 이미 승인했다. 따라서 그는 신용 카드 시스템에서 일반적으로 더 높은 거래 수수료를 결제하는 것보다 Algorand에서 1 % 수수료를 결제하는 것을 선호할 수 있다(또한, 수 일 보다는 수 분 내에 지불되는 것 및 논쟁의 여지가 더 적은 방법으로 결제하는 것을 선호한다).

[1269] 따라서 인증서 기반의 Algorand는:

[1270] (1) 모든 보상이 단지 상인에게 지급된 결제 중 적은 비율이며,

[1271] (2) 검증자가 다른 모든 결제를 처리하는 것에 대하여 인센티브를 받음

[1272] 을 보장할 수 있다.

[1273] 예를 들어, A'를 PAY^r 지불에서 소매상에게 결제한 총 금액으로 하면, 최대 잠재적 보상 R'을 계산할 수 있다(예: R'= 1%A'). 그러나 리더 및 검증자는 전체 금액 R'을 받을 수 없지만, PAY^r의 모든 결제의 총 수(또는 총액 또는 그 조합)와 함께 증가하는 R'의 일부만을 받을 수 있다. 예를 들어 이것을 유지하는 것은 매우 간단하며, PAY^r의 총 결제 횟수가 m이라면 분배되는 실제보상은 R'(1-1/m)이 된다. 전과 마찬가지로, 이는 각 소매업자에게 결제한 금액에서 1%(1-1/m)의 비율을 차감하고 선택한 수식에 따라 리더와 검증자간에 이 공제금액을 분할하여 자동으로 수행될 수 있다.

[1274] **13 변형**

[1275] 이제, Algorand에 가능한 변형을 설명한다.

[1276] **13.1 대체 검증자 선택 메커니즘**

[1277] 지금까지 Algorand는 이전 라운드에 따른 수량으로부터, 라운드 r의 리더 ℓ^r 및 검증자 세트 SV^r을 자동으로 선택하여, SV^r이 규정된 정직한 다수를 가짐을 확실하게 하였다. 그러나 우리는 검증자와 리더를 선택하는 다른 방법을 지적하고자 한다.

[1278] 물론 그러한 방법 중 하나는 모든 사용자가 실행하는 암호화 프로토콜을 통한 것이다. 그러나, 이 방법은 시스템의 사용자 수가 많으면 느려질 수 있다. 따라서 우리는 두 가지 종류의 대체 메커니즘(즉 체인화된, 자연 기반의 그리고 신뢰성 있는 파티)을 고려한다.(이미 논의된 메커니즘과 이러한 메커니즘을 혼합할 수도 있다.)

[1279] **체인화된 메커니즘**

[1280] 유도적으로 각 SV^r이 정직한 다수를 가졌다고 가정한다면, SV^r 자체(또는 보다 일반적으로 라운드 r까지의 검증자 중 일부)가 라운드 r의 검증자 세트 및/또는 리더를 선택하게 만들 수 있다. 예를 들어, 그들은 다중 파티 보안 계산을 통해 그렇게 할 수 있다. 초기 검증자 세트가 정직한 다수를 가지도록 선택된다고 가정할 때, 우리는 부트 스트래핑(boot-strapping)에 의존한다. 즉, 각 검증자 세트의 정직한 다수는 다음 검증자의 정직한 다수를 의미한다. 검증자 세트는 모든 사용자 세트에 비해 작기 때문에 그 구성원은 이 선택을 매우 신속하게 구현할 수 있다.

[1281] 다시 말하면, 이것은 매 라운드마다 검증자 세트 및 리더가 결정론적으로 도출되는 충분히 긴 랜덤 스트링을 선택하는 것으로 충분하다.

[1282] **자연 기반 메커니즘**

[1283] 주어진 라운드 r의 검증자 세트 SV^r 및 리더 ℓ^r는 라운드 r에 관련된 랜덤 값 v_r로부터 미리 결정된 방식으로 소정의 사용자 세트 PV^{r-k}로부터 선택될 수 있다. 특히, v_r은 자연적이고 공개된 무작위 값일 수 있다. 이것에 의해, 우리는 그것이 임의의 주어진 개인에 의해 거의 제어할 수 없는 랜덤화 프로세스의 널리 사용 가능한 결과라는 것을 의미한다. 예를 들어, v_r은 주어진 시간에(예컨대, 라운드 r 시작 시 또는 이전 라운드의 주어진 시간에) 여러 도시의 온도 또는 주어진 주식 거래소에서 주어진 시간에 거래되는 보안 주식의 수, 등등으로 구성될 수 있다.

[1284] 자연 및 공개 무작위 값이 충분히 길지 않을 수 있기 때문에, 아래와 같이 설정하는 것보다는,

[1285] $$SV^r \xleftarrow{v_r} PV^{r-k}$$

[1286] 대신, 아래와 같이 설정한다.

[1287] $$SV^r \xleftarrow{H(v_r)} PV^{r-k}$$

[1288] 이미 논의된 바와 같이, 적절한 의사 랜덤 방식으로 $H(v_r)$를 신장시키는 것이 필요하다.

[1289] **수탁자(trustee) 기반 메커니즘**

[1290] SV^r을 선택하는 또 다른 접근법은 그들 중 적어도 한명이 정직하다는 것을 보장하기 위해 선택된 하나 이상의 구별되는 엔티티인 수탁자을 포함한다. 수탁자는 페이셋 PAY^r을 구축하는 데 관여하지 않을 수 있지만, 검증자 세트 SV^r 및/또는 리더 ℓ^r를 선택할 수 있다.

[1291] 물론, 가장 단순한 수탁자 기반 메커니즘은 단일 수탁자 메커니즘이다. 단 한 명의 수탁자, T가 있을 때, 그는 반드시 정직하다. 따라서, 그는 라운드 r에서 사소한 선택, 디지털 서명, 및 이용 가능한 SV^r(또는 SV^r을 유도할 수 있는 충분한 랜덤 스트링 s_r)을 만들 수 있다.

[1292] 그러나, 이 간단한 메커니즘은 T에게 너무 많은 신뢰를 둔다. T를 더 적은 정도로 신뢰하기 위해, T는 오직 그만 생성할 수 있는, 라운드 r에 의해 고유하게 결정된 단일 스트링 s_r을 사용 가능하게 만들 수 있다(예컨대, $s_r = SIG_T(r)$). 그 다음, 모든 사용자는 랜덤 스트링, $H(SIG_T(v_r))$을 계산할 수 있으며, 이로부터 SV^r이 유도된다.

[1293] 이렇게 하면, T는 세트 SV^r을 제어할 권한을 갖지 못한다. 본질적으로 그는 자신의 처분에 따라 $SIG_T(r)$를 사용 가능하게 할 것인지 불가능하게 할 것인지 단일의 전략적 판정을 가진다. 따라서, 적절한 인센티브 또는 처벌과 함께, T가 정직하게 행동하고 있는지 여부를 확인하고 T가 그렇게 하고 있음을 보장하는 것이 더 쉬워진다.

[1294] 이 접근법의 문제점은 예측 불가능성이다. 실제로 T는 $SIG_T(r)$를 미리 계산할 수 있으며, 그것을 미래의 라운드의 검증자 세트를 알고 있는 누군가에게 은밀하게 공개할 수 있으며, 멤버를 공격하거나 손상시킬 충분한 시간을 갖게 될 수 있다.

[1295] 이 문제를 피하기 위해, 보안 하드웨어에 의존할 수 있다. 근본적으로 T는 적절한 라운드에서 적절한 디지털 서명을 출력하는 프로그램과 함께, "외부에" 포스팅된 공개 키 및 "내부에서" 잠겨 있는 매칭 비밀 키를 가지는 조작방지 장치를 가질 수 있다. 물론, 이 접근법은 보안 하드웨어 내부에 배포된 프로그램이 미래의 서명을 미리 공개하는 비밀 명령을 갖지 않는다는 것을 신뢰할 것을 요구한다.

[1296] 다른 접근법은 각 라운드 r과 관련된 자연 공개 랜덤 값 v_r을 사용하는 것이다. 예를 들어, T는 사용 가능한 $SIG_T(v_r)$를 만들도록 요청받을 수 있다. 이런 식으로, 미래의 라운드 r의 값 v_r은 누구에게도 알려지지

않기 때문에, T는 사전에 누설할 디지털 서명을 가지지 않는다.

[1297] 그러나, T가 여전히 누설할 수 있는 유일한 것은 자신이 소유한 비밀 서명 키이다. 이 잠재적 문제에 대처하기 위해 우리는 k명의 수탁자에게 의존할 수 있다. 그들중 적절한 대다수가 정직하다는 것을 보장할 수 있도록 선택될 수 있다면, 각 라운드 r에서 SV^r을 선택하기 위해 다중 파티 보안 계산을 확실히 사용할 수 있다. 더 간단하고 낮은 신뢰와 함께, 각 라운드 r에서, 우리는 각 수탁자 i가 r에 고유하게 연관된, 오직 i만이 생성할 수 있는, 단일 스트링을 사용 가능하게 만들 수 있고, 이러한 모든 스트링에 대해 SV^r을 계산할 수 있다. 예를 들어, 각 수탁자 i는 스트링 $SIG_i(r)$을 사용 가능하게 만들 수 있고, 그러므로 랜덤 스트링, $s^r = H(SIG_1(r), \ldots, SIG_k(r))$을 계산할 수 있고, 이로부터 검증자 세트, SV^r가 도출된다.

[1298] 이 접근법에서 우리는 각 디지털 서명 $SIG_i(r)$이 생성됨을 보장하기 위해 인센티브와 처벌에 의지할 수 있으며, 시퀀스 s^1, s^2, ...가 예측 불가능하게 유지됨을 보장하기 위해 단일 수탁자 i의 정직성에 의존한다.

[1299] 물론, 까다로운 부분은 필요한 스트링을 "사용 가능"하게 만드는 것이다. 전파 프로토콜에 의존한다면 악의적인 수탁자는 혼란을 야기하기 위해 의도적으로 그것들을 늦게 전파하기 시작할 수 있다. 따라서 수탁자 기반 메커니즘은 "보장된 브로드 캐스트 채널"의 존재에 의존해야 한다. 즉, 한 사용자가 메시지 m을 받는 경우 다른 모든 사람이 동일한 m을 수신하는 것이 보장되는 메시지 송신 방법에 의존한다.

[1300] 마지막으로, 각 라운드에서 보안 계산을 사용하는 대신 안전한 계산 전처리 단계를 사용할 수 있다. 이 단계는 정직한 다수를 가지도록 선정된 수탁자 세트에 의해 시스템 시작 시에 취해진 것이다. 이 단계는 아마도 여러 단계의 계산에 의해 각 수탁자 i에 대한 비밀 값, v_i 및 공개 값 p_v을 생성한다. 이것의 초기 계산에는 다소 시간이 걸릴 수 있지만, 각 라운드에서 필요한 계산은 간단할 수 있다. 예를 들어, 각 라운드 r에 대해, 그의 비밀 값 v_i를 사용하는 각각의 수탁자 i는 (바람직하게는 디지털 서명된) 단일 재구성 스트링 s_i^r를 생성하고 전파하여, 정확한 재구성 스트링의 대부분을 포함하는 임의의 스트링 세트 $\mathcal{S}^r$가 주어지면, 누구나 모호하지 않게 SV^r(또는 SV^r이 파생된 임의의 값)을 구성할 수 있게 한다. 물론, 이 접근법의 위험성은 고정된 수탁자 세트는 보다 쉽게 공격 받거나 손상될 수 있다는 것이다.

[1301] **13.2 더 정교한 암호화 도구**

[1302] 또한 Algorand는 보다 정교한 암호화 도구의 이점을 누릴 수 있다. 특히,

[1303] 1. 결합 가능한 서명. 종종 Algorand에서 일부 데이터 피스 D는 복수 파티에 의한 디지털 서명을 받아야 한다. 좀 더 컴팩트한 인증 레코드를 생성하기 위해, 결합 가능한 디지털 서명을 사용할 수 있다. 그러한 서명들에서, 다수의 공개 키들, 예를 들어, PK_1, PK_2 및 PK_3는 단일 공개키 $PK = PK_{1,2,3}$로 결합될 수 있고, 상이한 공개키들에 대해 동일한 데이터 D의 서명들이 대응하는 결합된 공개키에 대한 단일 서명으로 결합될 수 있다. 예를 들어, $SIG_1(D)$, $SIG_2(D)$ 및 $SIG_3(D)$는 단일 디지털 서명, $s = SIG_{1,2,3}(D)$으로 변환될 수 있고, 이것은 공개 키 $PK_{1,2,3}$에 대하여 누구나 검증할 수 있다. 관련 공개키의 식별자의 컴팩트 레코드(이 예에서는 세트 {1, 2, 3})는 s를 동반할 수 있으므로, 누구나 PK_1, PK_2, PK_3을 신속하게 수집하고 $PK = PK_{1,2,3}$를 계산하고, PK를 기반으로 D의 서명 s를 검증할 수 있다.

[1304] 이를 통해, 복수의 관련 전파를 단일 전파로 바꿀 수 있다. 본질적으로, 전파 프로토콜 동안, 사용자는 레코드{1, 2, 3}과 함께 $SIG_{1,2,3}(D)$ 및 레코드{4, 5}와 함께 $SIG_{4,5}(D)$를 수신했다고 가정하자. 그러면, 사용자는 $SIG_{1,2,3,4,5}(D)$와 레코드{1, 2, 3, 4, 5}를 전파할 수 있다.

[1305] 2. 트리-해시-및-서명. 서명이 복수의 데이터 피스를 인증할 때, 서명된 항목의 전체 목록을 보관 및 전송해야 하는 것 대신, 단일 데이터 피스의 서명만 추출할 수 있으면 유용할 수 있다. 예를 들어, 플레이어는 전체 인증된 PAY^r이 아니라, 주어진 결제 $P \in PAY^r$의 인증된 기록을 유지하고자 할 수 있다. 이를 위해, 먼저

각 결제 $P \in PAY^r$ 을 별도의 리프에 저장한 머클 트리를 생성한 후, 그 루트에 디지털 서명을 할 수 있다. 이 서명은 항목 P 및 그것의 인증 경로와 함께 본질적으로 P 단독의 대체 서명이다.

[1306] 3. 인증된 이메일.

[1307] 후자의 진행 방법의 일 장점은 플레이어가 인증된 이메일[53]로 자신의 결제를 ℓ에게 바람직하게는 발신자 익명의 방식으로 보내어, 그것이 PAY_ℓ^r 내의 결제 중 일부를 포함시키지 않기로 의도적으로 결정한 경우, ℓ을 처벌하는데 도움이 되는 영수증을 얻도록 한다.

[1308] (53: 예컨대, 미국 특허 제5,666,420호의 경량 인증 이메일)

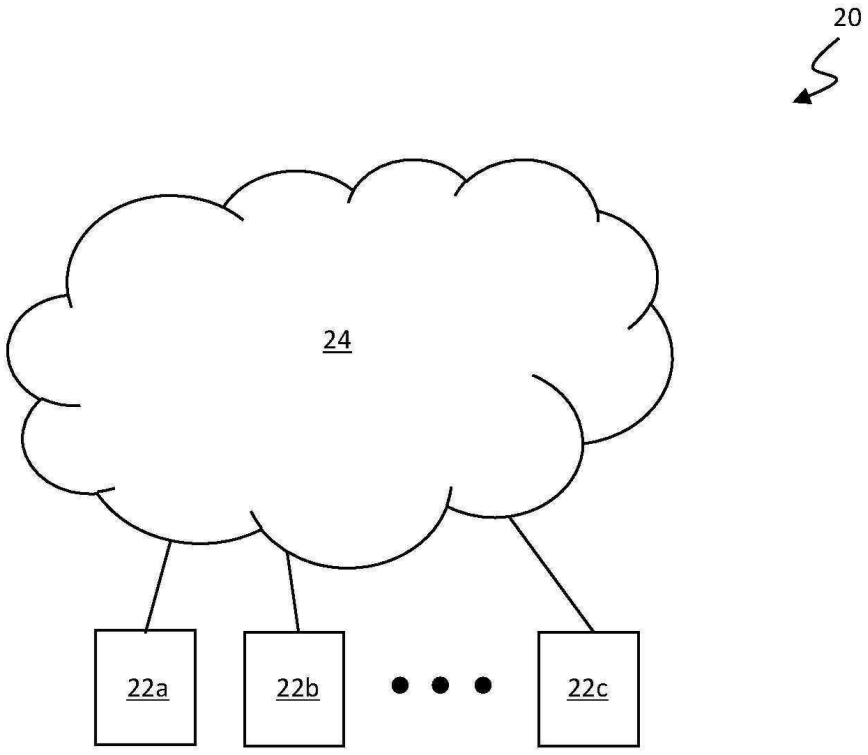
[1309] **14 범위**

[1310] 여기에 설명된 시스템의 소프트웨어 구현은 컴퓨터 판독 가능 매체에 저장되고 하나 이상의 프로세서에 의해 실행되는 실행 가능 코드를 포함할 수 있다. 컴퓨터 판독 가능 매체는 비-일시적일 수 있으며, 컴퓨터 하드 드라이브, ROM, RAM, 플래시 메모리, CD-ROM, DVD-ROM, 플래시 드라이브, SD 카드 및/또는, 예컨대, 범용 직렬 버스(USB) 인터페이스를 갖는 다른 드라이브와 같은 휴대용 컴퓨터 저장 매체 및/또는 프로세서에 의해 저장 및 실행될 수 있는 실행 가능 코드가 저장되는 임의의 다른 적절한 유형의 또는 비일시적 컴퓨터 판독 가능 매체 또는 컴퓨터 메모리를 포함한다. 여기에 설명된 시스템은 임의의 적절한 운영 체제와 함께 사용될 수 있다.

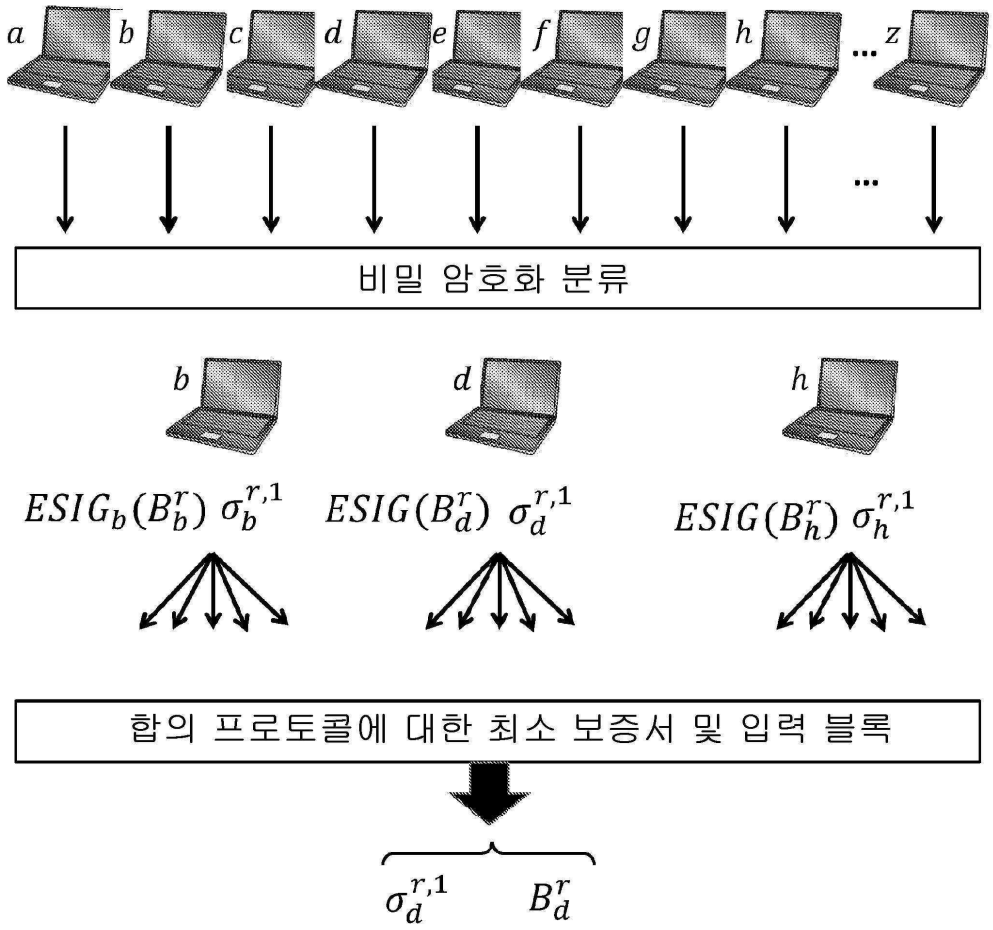
[1311] 본 발명의 다른 실시예들은 본 명세서에 개시된 본 발명의 명세서 또는 실시의 고려로부터 당업자에게 명백할 것이다. 본 명세서 및 실시예는 단지 예시적인 것으로 고려되어야 하며, 본 발명의 진정한 범위 및 사상은 다음의 청구 범위에 의해 표시된다.

도면

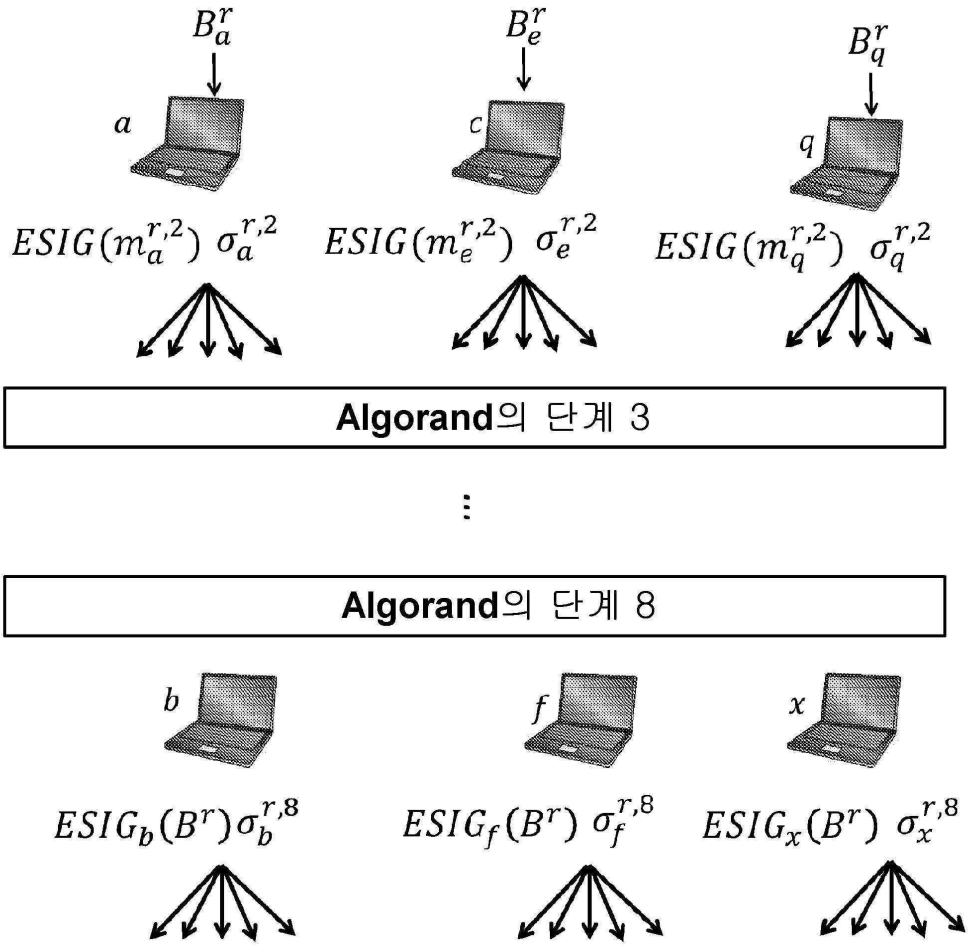
도면1



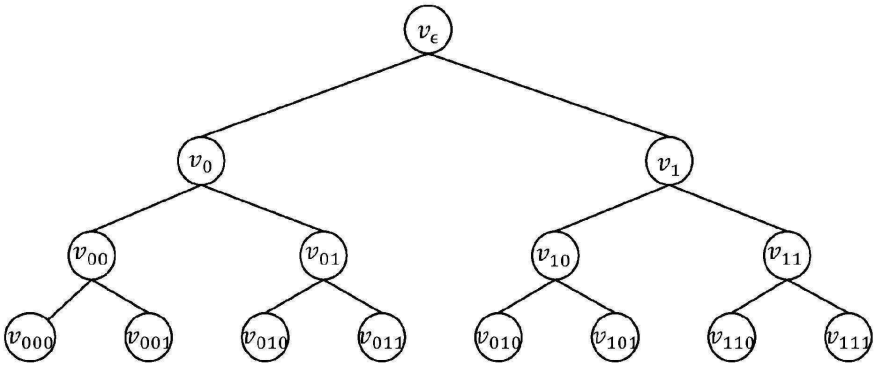
도면2


a
b
c
d
e
f
g
h
...
z
...
비밀 암호화 분류
b
d
h
$ESIG_b(B_b^r)\ \sigma_b^{r,1}$
$ESIG(B_d^r)\ \sigma_d^{r,1}$
$ESIG(B_h^r)\ \sigma_h^{r,1}$
합의 프로토콜에 대한 최소 보증서 및 입력 블록
$\sigma_d^{r,1}$
B_d^r

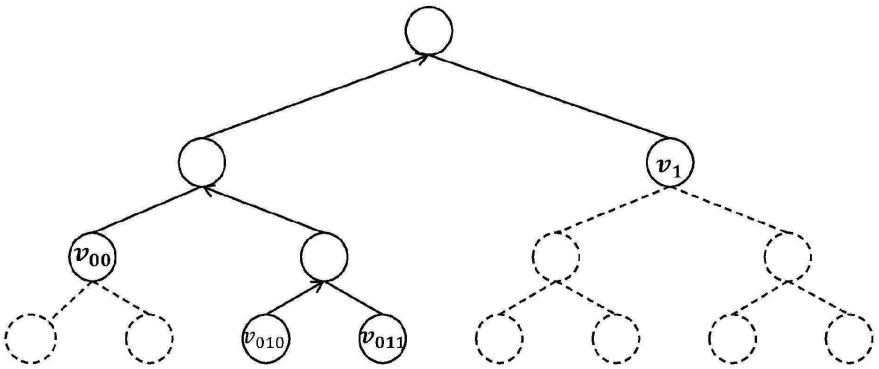
도면3


B_a^r
B_e^r
B_q^r
a
c
q
$ESIG(m_a^{r,2})\ \sigma_a^{r,2}$
$ESIG(m_e^{r,2})\ \sigma_e^{r,2}$
$ESIG(m_q^{r,2})\ \sigma_q^{r,2}$
Algorand의 단계 3
Algorand의 단계 8
b
f
x
$ESIG_b(B^r)\sigma_b^{r,8}$
$ESIG_f(B^r)\ \sigma_f^{r,8}$
$ESIG_x(B^r)\ \sigma_x^{r,8}$

도면4


v_ϵ
v_0
v_1
v_{00}
v_{01}
v_{10}
v_{11}
v_{000}
v_{001}
v_{010}
v_{011}
v_{010}
v_{101}
v_{110}
v_{111}

도 4.**A**


v_1
v_{00}
v_{010}
v_{011}

도 4.**B**

도면5