

인터넷컴퓨터(Internet Computer)

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

기술자를 위한 인터넷 컴퓨터

(v1.4)

DFINITY 팀*

2023년 12월 18일

초록

스마트 계약은 소프트웨어가 작성되는 방식, IT 시스템이 유지되는 방식, 그리고 애플리케이션과 전체 비즈니스가 구축되는 방식을 혁신적으로 변화시킬 새로운 형태의 소프트웨어이다.

스마트 계약은 탈중앙화된 블록체인 상에서 실행되는 조합 가능하고 자율적인 소프트웨어 조각으로, 이는 그것들을 변조 불가능하고 중단될 수 없게 만든다.

이 논문에서 우리는 인터넷 컴퓨터(Internet Computer, IC)를 설명하는데, 이는 속도, 저장 비용 및 계산 용량과 관련하여 전통적인 블록체인에서의 스마트 계약의 한계를 극복함으로써 스마트 계약의 잠재력을 완전히 발휘하게 하는 급진적으로 새로운 블록체인 설계이다.

이것은 스마트 계약이 처음으로 블록체인 상에서 엔드 투 엔드로 호스팅되는 완전히 탈중앙화된 애플리케이션을 구현할 수 있도록 한다.

IC는 독립적으로 운영되는 노드들을 여러 블록체인의 집합으로 연결하는 일련의 암호학적 프로토콜로 구성된다.

이 블록체인들은 IC의 스마트 계약 형태인 “캐니스터(canister)”를 호스팅하고 실행한다.

캐니스터는 데이터를 저장하고, 그 데이터에 대해 매우 일반적인 계산을 수행하며, 완전한 기술 스택을 제공하여 웹 페이지를 최종 사용자에게 직접 제공한다.

계산 및 저장 비용은 “역가스(reverse-gas) 모델”에 의해 총당되며, 여기서 캐니스터 개발자는 IC의 네이티브 토큰인 ICP에서 얻은 사이클(cycles)로 비용을 미리 지불한다.

ICP 토큰은 또한 거버넌스에 사용된다.

IC는 탈중앙화 자율 조직(DAO)에 의해 관리되며, 이 조직은 다른 일들과 더불어 네트워크의

토폴로지 변경과 프로토콜 업그레이드를 결정한다.

1 서론

1.1 스마트 계약의 잠재력

그 고유한 특성으로 인해, 스마트 계약은 웹3(Web3)의 핵심적인 실현 요소이며, 이는 애플리케이션이 사용자에게 의해 완전히 제어되고 탈중앙화된 블록체인 플랫폼에서 실행되는 웹에 대한 새로운 접근 방식이다.

이러한 탈중앙화된 애플리케이션(dapp)은 일반적으로 토큰화되어 있으며, 이는 토큰이 dapp에 참여하는 사용자에게 보상으로 분배된다는 것을 의미한다.

참여는 콘텐츠를 중재하고 제공하는 것에서부터 dapp을 관리하거나 dapp을 생성하고 유지하는 것까지 매우 다양한 형태로 이루어질 수 있다.

보통 토큰은 거래소에서 구매할 수도 있으며, 실제로 토큰을 판매하는 것은 dapp 개발 자금을 조달하는 일반적인 방법이다.

마지막으로, 토큰은 dapp이 제공하는 서비스나 콘텐츠에 대한 지불 형태로도 사용된다.

오늘날의 블록체인 플랫폼에서 실행되는 스마트 계약들은, 이더리움을 비롯한 모든 인기 있는 플랫폼을 포함하여, 높은 거래 및 저장 비용, 느린 계산 속도, 그리고 사용자에게 프론트엔드를 제공할 수 없는 등의 많은 제약을 겪고 있다.

그 결과, 많은 인기 있는 블록체인 애플리케이션들은 완전히 탈중앙화되지 않았으며, 애플리케이션의 대부분이 전통적인 클라우드 플랫폼에서 호스팅되고, 전체 기능 중 일부만을 블록체인상의 스마트 계약에 의존하는 하이브리드 형태이다.

불행하게도, 이것은 이러한 애플리케이션들을 비탈중앙화된 것으로 만들며, 클라우드 제공자에게 종속되거나 단일 실패 지점에 취약한 등 전통적인 클라우드 호스팅 애플리케이션의 많은 단점들에 노출되게 한다.

인터넷 컴퓨터(Internet Computer, IC)는 스마트 계약을 실행하기 위한 새로운 플랫폼이다.

여기서 우리는 “스마트 계약”이라는 용어를 매우 넓은 의미로 사용한다.

즉, *탈중앙화된 공개 네트워크에서 자율적으로 실행되는 범용적이고 변조 불가능한 컴퓨터 프로그램*을 의미한다.

- *범용적(general purpose)*이란 스마트 계약 프로그램의 집합이 튜링 완전(Turing complete)함을 의미하며, 즉 계산 가능한 것은 무엇이든 스마트 계약에 의해 계산될 수 있음을 의미한다.
- *변조 불가능(tamperproof)*하다는 것은 프로그램의 명령이 충실하게 수행되며, 중간 및 최종 결과가 정확하게 저장되거나 전송된다는 것을 의미한다.
- *자율적(autonomous)*이란 스마트 계약이 개별 개인의 어떤 행동도 필요 없이 네트워크에 의해 자동으로 실행된다는 것을 의미한다.
- *탈중앙화된 공개 네트워크(decentralized public network)*란 공개적으로 접근 가능하고, 지리적으로 분산되어 있으며, 소수의 개인이나 조직의 통제를 받지 않는 컴퓨터 네

트위크를 의미한다.

또한, 스마트 계약은

- 조합 가능성(*composability*) 을 가지며, 이는 서로 상호작용할 수 있음을 의미하고,
- 토큰화(*tokenization*) 를 지원하며, 이는 디지털 토큰을 사용하고 거래할 수 있음을 의미한다.

기존의 스마트 계약 플랫폼과 비교했을 때, IC는 다음과 같이 설계되었다:

- 더 높은 비용 효율성(*more cost effective*) 을 가지며, 특히 애플리케이션이 이전 플랫폼의 일부 비용으로 계산하고 데이터를 저장할 수 있게 한다.
- 더 높은 처리량과 낮은 지연 시간(*higher throughput and lower latency*) 으로 스마트 계약 거래를 처리할 수 있게 한다.
- 더 높은 확장성(*scalability*) 을 가지며, 특히 IC는 네트워크에 더 많은 노드를 추가함으로써 용량을 확장할 수 있으므로, 스마트 계약 데이터와 계산의 무한한 양을 본래적으로 처리할 수 있다.

스마트 계약이 가질 수 있는 또 다른 속성은 불변성(*immutability*) 이다.

이는 일단 배포되면, 스마트 계약의 코드가 어느 한쪽에 의해 일방적으로 변경될 수 없음을 의미한다.

이 기능은 일부 애플리케이션에서는 필수적이지만, 모든 애플리케이션에 필요한 것은 아니며, 스마트 계약에 수정이 필요한 버그가 있는 경우 문제를 일으킬 수도 있다.

IC는 완전히 불변한 것에서 일방적으로 업그레이드 가능한 것까지, 그 사이의 다른 선택지를 포함한 다양한 스마트 계약의 가변성 정책(*mutability policy*)을 허용한다.

스마트 계약 플랫폼을 제공하는 것 외에도, IC는 완전한 기술 스택(*complete technology stack*) 으로 작동하도록 설계되어, 시스템과 서비스가 완전히 IC 위에서 실행되도록 구축될 수 있다.

특히, IC의 스마트 계약은 최종 사용자가 생성한 HTTP 요청을 처리할 수 있으므로, 스마트 계약이 상호작용적인 웹 경험을 사용자에게 직접 제공할 수 있다.

이는 기업 클라우드 호스팅 서비스나 개인 서버에 의존하지 않고 시스템과 서비스를 구축할 수 있음을 의미하며, 따라서 스마트 계약의 모든 이점을 진정한 엔드 투 엔드 방식으로 제공한다.

웹3 비전의 실현

최종 사용자에게 있어 IC 기반 서비스를 이용하는 것은 대부분 투명하다.

그들의 개인 데이터는 공용 또는 사설 클라우드에서 애플리케이션에 접근할 때보다 더 안전하지만, 애플리케이션과 상호작용하는 경험은 동일하다.

그러나 이러한 IC 기반 서비스를 생성하고 관리하는 사람들에게 있어, IC는 현대적인 애플리케이션과 마이크로서비스를 개발하고 배포하는 데 수반되는 많은 비용, 위험, 그리고 복잡성을 제거한다.

예를 들어, IC 플랫폼은 인터넷을 독점하고 있는 대형 기술 회사들에 의해 주도되는 통합 현상

에 대한 대안을 제공한다.

또한, 그 보안 프로토콜은 방화벽, 백업 설비, 로드 밸런싱 서비스 또는 장애 조치 오케스트레이션에 의존하지 않고도 신뢰할 수 있는 메시지 전달, 투명한 책임성, 그리고 복원력을 보장한다.

IC를 구축하는 것은 인터넷을 개방적이고, 혁신적이며, 창의적인 근원으로 되돌리는 것이다. 즉, 웹3의 비전을 실현하는 것이다.

몇 가지 구체적인 예에 초점을 맞추면, IC는 다음과 같은 일을 수행한다:

- 상호 운용성, 공유 기능, 영구 API, 그리고 소유자가 없는 애플리케이션을 지원하며, 이는 모두 플랫폼 위험을 줄이고 혁신과 협업을 장려한다.
- 데이터를 메모리에 자동으로 지속시켜, 데이터베이스 서버와 저장 관리의 필요성을 제거하고, 계산 효율성을 개선하며, 소프트웨어 개발을 단순화한다.
- IT 조직이 통합하고 관리해야 하는 기술 스택을 단순화하여, 운영 효율성을 향상시킨다.

1.2 인터넷 컴퓨터의 개요

첫 번째 근사로 볼 때, IC는 상호작용하는 복제 상태 기계(replicated state machines)의 네트워크이다.

복제 상태 기계의 개념은 분산 시스템에서 꽤 표준적인 개념이지만 [Sch90], 우리는 여기서 상태 기계(state machine)의 개념부터 간단히 소개한다.

상태 기계(state machine)는 특정한 계산 모델이다.

이러한 기계는 상태(state)를 유지하며, 이는 일반적인 컴퓨터의 주 메모리나 다른 형태의 데이터 저장소에 해당한다.

이러한 기계는 불연속적인 라운드(round)로 실행된다.

각 라운드에서 그것은 입력(input)을 받아 상태 전이 함수(state transition function)를 입력과 현재 상태에 적용하여, 출력(output)과 새로운 상태(new state)를 얻는다.

새로운 상태는 다음 라운드에서 현재 상태가 된다.

IC의 상태 전이 함수는 보편 함수(universal function)로, 상태에 저장된 일부 입력과 데이터가 다른 입력과 데이터에 작용하는 임의의 프로그램(program)일 수 있음을 의미한다. 따라서 이러한 상태 기계는 일반적인(즉, 튜링 완전한) 계산 모델을 나타낸다.

결함 허용성(fault tolerance)을 달성하기 위해, 상태 기계는 복제(replicated)될 수 있다. 복제 상태 기계(replicated state machine)는 동일한 상태 기계를 실행하는 복제본(replica)들의 서브넷(subnet)으로 구성된다.

서브넷은 일부 복제본이 결함(faulty)이 있더라도 계속 작동해야 하며, 또한 올바르게 작동해야 한다.

서브넷의 각 복제본이 동일한 순서로 동일한 입력을 처리하는 것은 필수적이다.

이를 달성하기 위해, 서브넷의 복제본들은 합의 프로토콜(consensus protocol) [Fis83]을 실행해야 하며, 이는 서브넷의 모든 복제본이 동일한 순서로 입력을 처리하도록 보장한다.

따라서 각 복제본의 내부 상태는 시간이 지남에 따라 정확히 동일한 방식으로 진화하고, 각 복제본은 정확히 동일한 출력의 순서를 생성한다.

IC의 복제 상태 기계에 대한 입력은 외부 사용자가 생성한 입력이거나, 다른 복제 상태 기계가 생성한 출력일 수 있음을 주목하라.

마찬가지로, 복제 상태 기계의 출력은 외부 사용자에게 향하는 출력이거나, 다른 복제 상태 기계로 향하는 입력일 수 있다.

1.3 결함 모델

컴퓨터 과학의 분산 시스템 분야에서, 일반적으로 두 가지 유형의 복제본 실패를 고려한다: 충돌 결함(*crash faults*)과 비잔틴 결함(*Byzantine faults*).

충돌 결함(*crash fault*)은 복제본이 갑자기 중단되고 다시 시작되지 않는 경우에 발생한다.

비잔틴 결함(*Byzantine fault*)은 복제본이 지정된 프로토콜에서 임의의 방식으로 벗어나는 실패이다.

게다가 비잔틴 결함의 경우, 하나 또는 여러 개의 복제본이 악의적인 적대자의 직접적인 통제 하에 있을 수 있으며, 그 적대자는 이 복제본들의 행동을 조정할 수 있다.

두 가지 결함 유형 중에서, 비잔틴 결함이 잠재적으로 훨씬 더 파괴적이다.

합의 및 복제 상태 기계를 구현하기 위한 프로토콜은 일반적으로 얼마나 많은 복제본이 결함이 있을 수 있는지와 어떤 정도(*crash 또는 Byzantine*)로 결함이 있을 수 있는지에 대한 가정을 둔다.

IC에서는 주어진 서브넷이 n 개의 복제본을 가지고 있다면, 그중 $n/3$ 미만의 복제본만이 결함이 있을 수 있으며, 이 결함들은 비잔틴일 수 있다고 가정한다.

(IC 내의 서로 다른 서브넷들은 서로 다른 크기를 가질 수 있음을 주목하라.)

1.4 통신 모델

합의 및 복제 상태 기계를 구현하기 위한 프로토콜은 또한 통신 모델(*communication model*)에 대한 가정을 하는데, 이는 복제본들 간의 메시지 전달을 지연시키는 적대자의 능력을 특징짓는다.

양극단에는 다음과 같은 모델들이 있다:

- 동기식 모델(*synchronous model*)에서는, 알려진 유한한 시간 한계 δ 가 존재하여, 어떤 메시지가 전송되더라도 그것은 시간 δ 미만 내에 전달된다.
- 비동기식 모델(*asynchronous model*)에서는, 어떤 메시지가 전송되더라도, 적대자가 그것의 전달을 임의의 유한한 시간만큼 지연시킬 수 있으므로, 메시지 전달 시간에 상한이 존재하지 않는다.

IC 서브넷의 복제본들은 일반적으로 전 세계에 분산되어 있으므로, 동기식 통신 모델은 매우 비현실적일 것이다.

실제로, 공격자는 정직한 복제본이나 그들 간의 통신을 지연시킴으로써 프로토콜의 올바른 동작을 손상시킬 수 있다.

이러한 공격은 정직한 복제본을 장악하고 손상시키는 것보다 일반적으로 더 쉽게 수행될 수 있다.

전 세계적으로 분산된 서브넷의 환경에서는, 가장 현실적이고 강력한 모델은 비동기식 모델이다.

불행하게도, 이 모델에서 진정으로 실용적인 합의 프로토콜은 알려져 있지 않다.

(보다 최근의 비동기 합의 프로토콜들, 예를 들어 [MXC+16], 은 합리적인 처리량을 달성하지만, 지연 시간은 매우 좋지 않다.)

따라서 대부분의 다른 실용적인 비잔틴 결함 허용 시스템들처럼, IC는 타협점을 선택한다: 즉, *부분 동기성 통신 모델*(*partial synchrony communication model*) [DLS88].

이러한 부분 동기성 모델들은 다양한 방식으로 공식화될 수 있다.

IC가 사용하는 부분 동기성 가정은 대략적으로 말해서, 각 서브넷에서 복제본 간 통신이 짧은 시간 간격 동안 주기적으로 동기화된다는 것을 의미하며, 또한 동기성 한계 δ 는 사전에 알 필요가 없다.

이 부분 동기성 가정은 합의 프로토콜이 진행되도록(즉, *활성(liveness)* 속성) 보장하는 데만 필요하다.

이 부분 동기성 가정은 합의의 올바른 동작(즉, *안전성(safety)* 속성)을 보장하기 위해서나, IC 프로토콜 스택의 다른 곳에서는 필요하지 않다.

부분 동기성과 비잔틴 결함을 가정할 때, 결함의 수에 대한 우리의 한계 $f < n/3$ 은 최적임이 알려져 있다.

1.5 권한 모델

가장 초기의 합의 프로토콜들(예: PBFT [CL99])은 *허가형(permissioned)* 이었는데, 이는 복제 상태 기계를 구성하는 복제본들이 중앙화된 조직에 의해 관리된다는 의미이며, 이 조직이 어떤 엔티티가 복제본을 제공할지, 네트워크의 토폴로지를 어떻게 구성할지, 그리고 어떤 형태의 중앙화된 공개키 인프라를 구현할지도 결정한다는 뜻이다.

허가형 합의 프로토콜은 일반적으로 가장 효율적이며, 단일 실패 지점을 피하긴 하지만, 중앙화된 거버넌스는 특정 애플리케이션에서는 바람직하지 않으며, 성장 중인 웹3(Web3) 시대의 정신과는 상반된다.

최근에는 비트코인 [Nak08], 이더리움 [But13], 알고랜드 [GHM+17] 등과 같은 *무허가형(permissionless)* 합의 프로토콜의 등장을 보았다.

이러한 프로토콜들은 *블록체인(blockchain)* 과 *작업 증명(Proof of Work, PoW)* (예: 비트코인, 이더리움 v2.0 이전 버전) 또는 *지분 증명(Proof of Stake, PoS)* (예: 알고랜드, 이더리움 v2.0)을 기반으로 한다.

이러한 프로토콜들은 완전히 탈중앙화되어 있지만, 허가형 프로토콜에 비해 훨씬 덜 효율적이다.

또한 [PSS17]에서 지적된 바와 같이, 비트코인과 같은 PoW 기반 합의 프로토콜은 비동기식 통신 네트워크에서 올바른(즉, 안전성)을 보장할 수 없다.

IC의 권한 모델은 *하이브리드 모델(hybrid model)* 로, 허가형 프로토콜의 효율성과 탈중앙화된 PoS 프로토콜의 많은 장점을 동시에 얻는다.

이 하이브리드 모델은 *DAO 제어 네트워크(DAO-controlled network)* 라고 불리며, 대략 다음과 같이 작동한다.

각 서브넷은 허가형 합의 프로토콜을 실행하지만, *탈중앙화 자율 조직(DAO, Decentralized Autonomous Organization)* 이 어떤 엔티티가 복제본을 제공할지 결정하고, 네트워크의 토폴로지를 구성하며, 공개키 인프라를 제공하고, 복제본에 배포되는 프로토콜의 버전을 제어한다.

IC의 DAO는 *네트워크 신경계(Network Nervous System, NNS)* 라고 불리며, PoS를 기반으로 한다.

따라서 NNS에서 이루어지는 모든 결정은 NNS 내에서 IC의 네이티브 거버넌스 토큰을 스테이킹한 커뮤니티 구성원들의 투표력에 의해 결정된다. (이 토큰에 대한 자세한 내용은 1.8절 참조.)

이 PoS 기반 거버넌스 시스템을 통해 새로운 서브넷을 생성할 수 있으며, 기존 서브넷에 복제본을 추가하거나 제거할 수 있고, 소프트웨어 업데이트를 배포하거나, IC에 대한 기타 수정이 이루어질 수 있다.

NNS 자체도 복제 상태 기계로, 다른 모든 복제 상태 기계와 마찬가지로 동일한 PoS 기반 거버넌스 시스템에 의해 결정된 특정 서브넷에서 실행된다.

NNS는 *레지스트리(registry)* 라는 데이터베이스를 유지하며, 이는 IC의 토폴로지를 추적한다.

즉, 어떤 복제본이 어떤 서브넷에 속하는지, 복제본들의 공개키가 무엇인지 등을 관리한다. (NNS에 대한 추가 세부사항은 1.10절 참조.)

따라서, IC의 DAO 제어 네트워크는 허가형 네트워크의 많은 실질적인 이점(즉, 보다 효율적인 합의)을 달성하면서도, DAO에 의해 관리되는 거버넌스를 통해 탈중앙화된 네트워크의 많은 이점을 유지한다.

IC 프로토콜을 실행하는 복제본들은 지리적으로 분산되어 독립적으로 운영되는 데이터 센터의 서버에서 호스팅된다.

이것은 또한 IC의 보안성과 탈중앙화된 특성을 강화한다.

1.6 체인 키 암호학

IC의 합의 프로토콜은 실제로 블록체인을 사용하지만, 또한 공개키 암호(public-key cryptography), 구체적으로는 디지털 서명(digital signatures)을 사용한다.

NNS가 유지하는 레지스트리는 공개키를 복제본들과 서브넷 전체에 바인딩하는 데 사용된다. 이것은 우리가 *체인 키 암호학(chain-key cryptography)* 이라고 부르는 독특하고 강력한 기술 모음을 가능하게 하며, 여러 구성 요소를 포함한다.

1.6.1 임계값 서명(threshold signatures)

체인 키 암호학의 첫 번째 구성 요소는 *임계값 서명(threshold signatures)* 이다.

이는 잘 확립된 암호학적 기술로, 서브넷이 공개 서명 검증 키(public signature-verification key)를 가지게 하고, 그에 대응하는 비밀 서명 키(secret signing key)를 조각(*shares*)으로 분할하여, 이 조각들을 서브넷의 모든 복제본에 분산시킨다.

이때, 손상된 복제본들이 보유한 조각들로는 어떤 서명도 위조할 수 없으며, 정직한 복제본들이 보유한 조각들은 IC의 정책과 프로토콜에 부합하는 서명을 생성할 수 있도록 한다.

이러한 임계값 서명의 중요한 한 가지 응용은 다음과 같다.

하나의 서브넷의 개별 출력은 해당 서브넷의 공개 서명 검증 키에 대한 디지털 서명을 검증하는 것만으로, 다른 서브넷이나 외부 사용자가 검증할 수 있다.

서브넷의 공개 서명 검증 키는 NNS로부터 얻을 수 있으며, 이 공개 서명 검증 키는 서브넷의 수명이 지속되는 동안(서브넷의 구성원이 그 기간 동안 변경되더라도) 일정하게 유지된다. 이는 단일 출력을 검증하기 위해 전체 블록체인을 검증해야 하는 많은 비확장성 블록체인 기반 프로토콜과 대조된다.

보게 되겠지만, 이러한 임계값 서명은 IC 내에서 여러 다른 응용을 가진다. 그 중 하나는 서브넷의 각 복제본에게 예측 불가능한 의사난수 비트(pseudorandom bits)에 접근할 수 있도록 하는 것이다(이 비트들은 이러한 서명에서 유도된다). 이는 합의에서 사용되는 랜덤 비콘(Random Beacon) 및 실행에서 사용되는 랜덤 테이프(Random Tape)의 기반이 된다.

임계값 서명을 안전하게 배포하기 위해, IC는 혁신적인 분산 키 생성(Distributed Key Generation, DKG) 프로토콜을 사용한다.

이 프로토콜은 공개 서명 검증 키를 구성하고, 각 복제본에 대응하는 비밀 서명 키의 조각을 제공하며, IC의 결함 및 통신 모델 내에서 작동한다.

1.6.2 체인 진화 기술(Chain-evolution technology)

체인 키 암호화는 또한 시간이 지남에 따라 블록체인 기반 복제 상태 기계를 견고하고 안전하게 유지하기 위한 정교한 기술 집합을 포함하며, 이들은 함께 체인 진화 기술(chain-evolution technology)이라 불린다.

각 서브넷은 많은 라운드(보통 수백 라운드 규모)의 에포크(epoch) 단위로 작동한다.

임계값 서명 및 여러 다른 기술들을 사용하여, 체인 진화 기술은 에포크에 맞춰 주기적으로 실행되는 여러 필수 유지 관리 활동을 구현한다.

가비지 수집(Garbage collection): 각 에포크가 끝날 때, 처리된 모든 입력과, 그 입력들을 순서화하기 위해 필요한 모든 합의 수준 프로토콜 메시지는 각 복제본의 메모리에서 안전하게 제거될 수 있다.

이는 복제본의 저장 요구사항이 무한히 증가하지 않도록 유지하는 데 필수적이다.

이는 제네시스 블록(genesis block)부터 전체 블록체인을 저장해야 하는 많은 비확장성 블록체인 기반 프로토콜과는 대조된다.

빠른 진행(Fast forwarding): 서브넷의 복제본이 피어들보다 매우 뒤처졌을 경우(예를 들어 오랫동안 중단되거나 네트워크에서 분리된 경우) 또는 새로운 복제본이 서브넷에 추가될 경우, 합의 프로토콜을 실행하거나 그 시점까지의 모든 입력을 처리하지 않고도 가장 최근 에포크의 시작으로 빠르게 이동(fast forward) 될 수 있다.

이는 제네시스 블록부터 전체 블록체인을 처리해야 하는 많은 비확장성 블록체인 기반 프로토콜과는 대조된다.

서브넷 구성 변경(Subnet membership changes): 서브넷의 구성(1.5절의 NNS에 의해 결정됨)은 시간이 지남에 따라 변경될 수 있다.

이것은 오직 에포크 경계에서만 일어날 수 있으며, 일관되고 올바른 동작을 보장하기 위해 주의 깊게 수행되어야 한다.

비밀의 사전 재분할(Pro-active resharing of secrets): 1.6.1절에서 IC가 체인 키 암호학, 구체적으로 임계값 서명을 출력 검증에 사용하는 방법을 언급했다.

이것은 *비밀 분할(secret sharing)*에 기반하며, 이는 비밀(이 경우 비밀 서명 키)을 복제본들 사이에 분할하여 저장함으로써 단일 실패 지점을 피한다.

각 에포크의 시작 시점에, 이러한 조각들은 *사전 재분할(pro-actively reshared)* 된다.

이것은 두 가지 목적을 달성한다:

- 서브넷의 구성이 변경될 때, 재분할은 새로운 구성원들이 적절한 비밀 조각을 가지도록 하고, 더 이상 구성원이 아닌 복제본들은 비밀 조각을 더 이상 가지지 않도록 보장한다.
- 소수의 조각이 어떤 한 에포크나 심지어 모든 에포크에서 공격자에게 유출되더라도, 그 조각들은 공격자에게 아무런 도움이 되지 않는다.

프로토콜 업그레이드(Protocol upgrades): IC 프로토콜 자체를 버그 수정이나 새로운 기능 추가를 위해 업그레이드해야 할 때, 이것은 에포크의 시작 시점에서 특별한 프로토콜을 사용하여 자동으로 수행될 수 있다.

1.7 실행 모델

앞서 언급했듯이, IC의 복제 상태 기계는 임의의 프로그램을 실행할 수 있다.

IC에서 기본 계산 단위는 *캐니스터(canister)*라고 불리며, 이는 *프로세스(process)* 개념과 대체로 동일한데, 프로그램과 그 *상태(state)* (시간에 따라 변화함) 모두를 포함하기 때문이다.

캐니스터 프로그램은 *웹어셈블리(WebAssembly)*, 줄여서 *Wasm*으로 인코딩되며, 이는 스택 기반 가상 머신을 위한 바이너리 명령어 형식이다.

Wasm은 개방형 표준이다.¹

비록 처음에는 웹 페이지 상에서 고성능 애플리케이션을 가능하게 하기 위해 설계되었지만, 실제로는 범용 계산에 매우 적합하다.

IC는 캐니스터 내에서 Wasm 프로그램을 실행하기 위한 런타임 환경을 제공하며, 메시지 전달을 통해 다른 캐니스터 및 외부 사용자와 통신할 수 있게 한다.

원칙적으로는 Wasm으로 컴파일될 수 있는 어떤 언어로도 캐니스터 프로그램을 작성할 수 있지만, IC의 운용 의미론에 잘 부합하도록 설계된 *Motoko*라는 언어가 있다.

*Motoko*는 강한 타입을 가진 *액터 기반²(actor-based)* 프로그래밍 언어로, *직교적 지속성³(orthogonal persistence)*과 *비동기 메시지 전달(asynchronous message passing)*에 대한 내장 지원을 갖추고 있다.

직교적 지속성이란 캐니스터가 유지하는 모든 메모리가 자동으로 지속된다는 것을 의미하며, 즉 파일에 쓰지 않아도 된다.

*Motoko*는 자동 메모리 관리, 제네릭, 타입 추론, 패턴 매칭, 임의 정밀도 및 고정 정밀도 산술을 포함한 다양한 생산성과 안전성 기능을 갖춘다.

Motoko 외에도, IC는 *Candid* 라고 불리는 메시징 인터페이스 정의 언어 및 전송 형식을 제공한다.

Candid는 타입이 지정된 고수준 언어 간 상호운용성을 가능하게 하며, 서로 다른 고급 언어로 작성된 두 캐니스터 간에도 손쉽게 통신할 수 있게 한다.

특정 프로그래밍 언어에서의 캐니스터 개발을 완전히 지원하기 위해서는, 그 언어에 대한 Wasm 컴파일러뿐만 아니라 일정한 런타임 지원도 제공되어야 한다.

현재 시점에서, IC는 Motoko 외에도 Rust 프로그래밍 언어에서의 캐니스터 개발을 완전히 지원한다.

1.8 유틸리티 토큰

IC는 *유틸리티 토큰*(utility token) 인 ICP 를 사용한다.

이 토큰은 다음과 같은 기능에 사용된다.

NNS 내 스테이킹(Staking in the NNS):

1.5절에서 이미 논의했듯이, ICP 토큰은 NNS에 *스테이킹*(staked) 되어 투표권을 획득함으로써 IC 네트워크를 제어하는 DAO에 참여할 수 있다.

NNS에 ICP 토큰을 스테이킹하고 NNS 거버넌스에 참여하는 사용자들은 *새로 발행된* (newly minted) ICP 토큰을 *투표 보상*(voting reward) 으로 받는다.

보상 금액은 NNS에 의해 설정되고 시행되는 정책에 의해 결정된다.

¹ <https://webassembly.org/> 참조.

² https://en.wikipedia.org/wiki/Actor_model 참조.

³

[https://en.wikipedia.org/wiki/Persistence_\(computer_science\)#Orthogonal_or_transparent_persistence](https://en.wikipedia.org/wiki/Persistence_(computer_science)#Orthogonal_or_transparent_persistence) 참조.

사이클로의 변환(Conversion to Cycles):

ICP는 IC 사용에 대한 대가로 사용된다.

보다 구체적으로, ICP 토큰은 *사이클*(cycles) 로 변환될 수 있으며(즉, *소각됨*(burned)), 이 사이클은 캐니스터 생성(1.7절 참조) 및 캐니스터가 사용하는 자원(저장소, CPU, 대역폭)에 대한 비용을 지불하는 데 사용된다.

ICP가 사이클로 변환되는 비율은 NNS에 의해 결정된다.

노드 제공자에 대한 지급(Payment to Node Providers):

ICP 토큰은 노드 제공자에게 지급하는 데 사용된다.

노드 제공자는 IC를 구성하는 복제본을 호스팅하는 컴퓨팅 노드를 소유하고 운영하는 엔터티들이다.

정기적으로(현재는 매월) NNS는 각 노드 제공자가 받아야 할 *새로 발행된*(newly minted) 토큰의 수를 결정하고, 해당 토큰을 노드 제공자의 계정으로 전송한다.

토큰 지급은 NNS에 의해 설정되고 시행되는 특정 정책에 따라 IC에 신뢰성 있는 서비스를 제공하는 조건 하에서 이루어진다.

1.9 경계 노드

경계 노드(*boundary nodes*)는 IC의 네트워크 엣지 서비스를 제공한다. 특히, 다음과 같은 기능을 제공한다:

- IC에 대한 명확하게 정의된 진입점,
- IC에 대한 서비스 거부(Denial of Service) 방어,
- 기존 클라이언트(예: 웹 브라우저)로부터 IC에 대한 원활한 접근.

기존 클라이언트로부터 IC에 대한 원활한 접근을 가능하게 하기 위해, 경계 노드는 사용자의 표준 HTTPS 요청을 IC의 캐니스터로 향하는 인그레스(*ingress*) 메시지로 변환하는 기능을 제공하고, 그 인그레스 메시지를 해당 캐니스터가 존재하는 서브넷의 특정 복제본으로 라우팅한다.

또한, 경계 노드는 사용자 경험을 향상시키기 위한 추가 서비스를 제공한다.

이에는 캐싱, 로드 밸런싱, 속도 제한(*rate limiting*), 그리고 기존 클라이언트가 IC로부터의 응답을 인증할 수 있는 기능이 포함된다.

캐니스터는 *ico.app* 도메인의 URL로 식별된다.

처음에 기존 클라이언트는 해당 URL의 DNS 레코드를 조회하여 경계 노드의 IP 주소를 얻은 뒤, 초기 HTTPS 요청을 이 주소로 보낸다.

경계 노드는 레거시 클라이언트에서 실행될 자바스크립트 기반의 “서비스 워커(*service worker*)”를 반환한다.

그 후, 기존 클라이언트와 경계 노드 간의 모든 상호작용은 이 서비스 워커를 통해 수행된다.

서비스 워커가 수행하는 필수 작업 중 하나는 체인 키 암호화(1.6절 참조)을 사용하여 IC로부터의 응답을 인증하는 것이다.

이를 위해, NNS의 공개 검증 키는 서비스 워커에 하드코딩되어 있다.

경계 노드 자체는 지정된 캐니스터가 호스팅된 서브넷의 복제본으로 요청을 라우팅하는 책임을 가진다.

이 라우팅을 수행하는 데 필요한 정보는 경계 노드가 NNS로부터 얻는다.

경계 노드는 시의적절한 응답을 제공하는 복제본들의 목록을 유지하고, 이 목록에서 무작위로 하나의 복제본을 선택한다.

기존 클라이언트와 경계 노드 간, 그리고 경계 노드와 복제본 간의 모든 통신은 TLS를 통해 보안이 유지된다.⁴

기존 클라이언트 외에도, “IC 네이티브(IC native)” 클라이언트를 사용하여 경계 노드와 상호작용할 수도 있으며, 이러한 클라이언트는 이미 서비스 워커 로직을 포함하고 있으므로, 경계 노드로부터 서비스 워커 프로그램을 가져올 필요가 없다.

복제본과 마찬가지로, 경계 노드의 배포와 구성 역시 NNS에 의해 제어된다.

1.10 NNS의 추가 세부 사항

1.5절에서 이미 언급했듯이, 네트워크 신경계(Network Nervous System, NNS)는 IC를 제어하는 알고리즘적 거버넌스 시스템이다.

이는 특수한 시스템 서브넷(system subnet) 상의 일련의 캐니스터들로 구현된다.

이 서브넷은 다른 서브넷들과 유사하지만, 약간 다르게 구성된다(예를 들어, 시스템 서브넷의 캐니스터들은 사용한 사이클에 대해 요금이 부과되지 않는다).

가장 관련성 높은 NNS 캐니스터들은 다음과 같다:

- 레지스트리 캐니스터(registry canister) : IC의 구성(configuration)을 저장하며, 즉 어떤 복제본이 어떤 서브넷에 속하는지, 서브넷 및 개별 복제본과 연관된 공개키 등이 저장된다.
- 거버넌스 캐니스터(governance canister) : IC가 어떻게 발전해야 하는지에 대한 의사결정과 투표를 관리한다.
- 원장 캐니스터(ledger canister) : 사용자들의 ICP 유틸리티 토큰 계정과 그들 간의 거래를 추적한다.

1.10.1 NNS에서의 의사결정

누구나 이른바 뉴런(neurons)에 ICP 토큰을 스테이킹(staking)함으로써 NNS 거버넌스에 참여할 수 있다.

뉴런 보유자는 제안(proposals)을 제시하고 투표할 수 있으며, 이는 예를 들어 서브넷의 토폴로지나 프로토콜을 어떻게 변경할지 등 IC를 변경하는 방법에 대한 제안이다.

의사결정을 위한 뉴런의 투표력은 지분 증명(Proof of Stake)에 기반한다.

직관적으로, 더 많은 ICP 토큰을 스테이킹한 뉴런은 더 많은 투표력을 가진다.

그러나 투표력은 다른 뉴런의 특성에도 의존하며, 예를 들어 더 오랜 기간 동안 토큰을 스테이킹하기로 약속한 뉴런 보유자에게 더 많은 투표력이 주어진다.

각 제안에는 정해진 투표 기간이 있다.

투표 기간이 끝날 때 전체 투표력의 단순 과반수가 제안에 찬성하고, 이 찬성표가 전체 투표력의 일정 정족수(현재 3%)를 구성하면, 제안은 채택(adopted)된다.

그렇지 않으면 제안은 기각(rejected)된다.

또한 전체 투표력의 절대 과반수(절반 초과)가 찬성 또는 반대하는 경우, 제안은 언제든지 즉시 채택되거나 기각될 수 있다.

제안이 채택되면, 거버넌스 캐니스터는 해당 결정을 자동으로 실행한다.

예를 들어, 제안이 네트워크 토폴로지 변경을 제시하고 채택되었다면, 거버넌스 캐니스터는 새로운 구성으로 레지스트리를 자동으로 업데이트한다.

⁴ https://en.wikipedia.org/wiki/Transport_Layer_Security 참조.

1.11 고급 기능

IC의 아키텍처는 IC에서 실행되는 모든 dapp에 대해 토큰화된 DAO 거버넌스의 이점을 제공하고, 다른 체인과 직접 상호작용할 수 있는 기능을 가능하게 한다.

DAO 제어 캐니스터(DAO-controlled canisters).

IC의 전체 구성이 NNS에 의해 제어되는 것과 마찬가지로, 어떤 캐니스터도 *서비스 신경계* (Service Nervous System, SNS) 라 불리는 자체 DAO에 의해 제어될 수 있다.

캐니스터를 제어하는 DAO는 캐니스터 로직의 업데이트를 제어할 수 있으며, 캐니스터가 수행해야 할 특권 명령을 발행할 수도 있다.

임계값 ECDSA(Threshold ECDSA).

ECDSA 서명 [JMV01]은 비트코인과 이더리움 같은 암호화폐뿐 아니라 많은 다른 애플리케이션에서도 사용된다.

비록 임계값 서명(threshold signatures)이 이미 IC의 필수 구성 요소이지만, 이것들은 임계값 ECDSA 서명은 아니다.

이 기능은 개별 캐니스터가 ECDSA 서명 키를 제어할 수 있게 하며, 이 키는 캐니스터를 호스팅하는 서브넷의 모든 복제본에 안전하게 분산된다.

비트코인 및 이더리움 통합(Bitcoin and Ethereum integration).

새로운 임계값 ECDSA 기능을 기반으로, 이 기능은 캐니스터가 비트코인 및 이더리움 블록체인과 상호작용할 수 있게 하며, 거래를 서명하는 기능도 포함한다.

HTTPS 아웃콜(HTTPS outcalls).

이 기능은 캐니스터가 임의의 웹 페이지(IC 외부)에 HTTPS 요청을 보내고, 합의에 의해 승인된 후 응답을 처리할 수 있도록 한다.

2 아키텍처 개요

그림 1에 나타난 바와 같이, 인터넷 컴퓨터 프로토콜은 네 개의 계층으로 구성된다:

- 피어-투-피어 계층(peer-to-peer layer) (4절 참조);
- 합의 계층(consensus layer) (5절 참조);
- 라우팅 계층(routing layer) (6절 참조);
- 실행 계층(execution layer) (7절 참조).

체인 키 암호학(chain-key cryptography)은 여러 계층에 영향을 미치며, 3절(임계값 서명)과 8절(체인 진화 기술)에서 자세히 논의된다.

2.1 피어-투-피어 계층

피어-투-피어 계층의 역할은 서브넷 내 복제본들 간에 프로토콜 메시지를 전송하는 것이다. 이러한 프로토콜 메시지에는 다음이 포함된다:

- 합의를 구현하기 위해 사용되는 메시지,
- 외부 사용자에게 의해 생성된 입력 메시지.

기본적으로, 피어-투-피어 계층이 제공하는 서비스는 “최선의 노력(best effort)” 브로드캐스트 채널이다:

정직한 복제본이 메시지를 브로드캐스트하면, 그 메시지는 결국 서브넷 내의 모든 정직한 복제본에게 전달된다.

설계 목표는 다음과 같다:

- 제한된 자원(Bounded resources). 모든 알고리즘은 제한된 자원(메모리, 대역폭, CPU) 내에서 작동해야 한다.
- 우선순위(Prioritization). 메시지의 유형, 크기, 라운드 등 특정 속성에 따라 다른 우선순위로 처리될 수 있으며, 이러한 우선순위는 시간이 지남에 따라 변경될 수 있다.
- 효율성(Efficiency). 낮은 지연 시간보다 높은 처리량이 더 중요하다.
- DOS/스팸 복원력(DOS/SPAM resilience). 손상된 복제본이 정직한 복제본 간의 통신을 방해해서는 안 된다.

2.2 합의 계층

IC의 합의 계층의 역할은 입력들을 순서화하여, 서브넷 내 모든 복제본이 동일한 순서로 입력을 처리하도록 하는 것이다.

이 문제에 대한 다양한 프로토콜이 문헌에 존재한다.

IC는 새로운 합의 프로토콜을 사용하며, 여기서는 이를 고수준에서 설명한다.

모든 안전한 합의 프로토콜은 두 가지 속성을 보장해야 한다. 대략적으로 다음과 같다:

- 안전성(safety): 모든 복제본은 실제로 동일한 입력 순서에 동의해야 한다.
- 활성성(liveness): 모든 복제본은 지속적으로 진전해야 한다.

IC의 합의 프로토콜은 다음과 같이 설계되었다:

- 매우 단순하고,
- 강건하다: 일부 복제본이 악의적일 때 성능이 점진적으로 저하된다.

위에서 논의했듯이, 우리는 $f < n/3$ 개의 결함(즉, 비잔틴 복제본)을 가정한다.

또한, 활성성은 부분 동기성 가정 하에서 성립하며, 안전성은 완전히 비동기적인 네트워크에서도 보장된다.

여러 합의 프로토콜과 마찬가지로, IC 합의 프로토콜은 블록체인에 기반한다.

프로토콜이 진행됨에 따라, 트리의 루트인 특수한 제네시스 블록(*genesis block*) 부터 시작하여 블록의 트리가 성장한다.

트리의 제네시스 블록이 아닌 각 블록에는 (기타 내용 외에) 입력 시퀀스로 구성된 페이로드(*payload*) 와 부모 블록의 해시가 포함된다.

정직한 복제본들은 이 트리에 대한 일관된 관점을 가진다.

각 복제본은 이 트리에 대한 부분적이고 서로 다른 관점을 가질 수 있지만, 모든 복제본은 동일한 트리를 본다.

또한, 프로토콜이 진행됨에 따라, 이 트리에는 항상 최종화된 블록(*finalized blocks*) 의 경로가 존재한다.

정직한 복제본들은 이 경로에 대해서도 일관된 관점을 가진다.

각 복제본은 서로 다른 부분적 관점을 가질 수 있지만, 모두 동일한 경로를 본다.

이 경로를 따라 있는 블록들의 페이로드 내 입력들이 인터넷 컴퓨터의 실행 계층에서 처리될 순서화된 입력들이다.

프로토콜은 라운드(*round*) 단위로 진행된다.

라운드 h 에서, 높이 h (*height h*) 의 하나 이상의 블록이 트리에 추가된다.

즉, 라운드 h 에서 추가된 블록들은 루트로부터 정확히 거리 h 만큼 떨어진 위치에 있다.

각 라운드에서, 의사난수 프로세스가 각 복제본에 고유한 순위(*rank*) 를 할당한다.

순위는 $0, \dots, n - 1$ 범위의 정수이다.

이 의사난수 프로세스는 랜덤 비콘(*Random Beacon*) 을 사용하여 구현되며(이는 1.6.1절에서 언급된 임계값 서명을 사용하며, 3절에서 더 자세히 논의된다), 순위가 가장 낮은 복제본이 그 라운드의 리더(*leader*) 가 된다.

리더가 정직하고 네트워크가 동기적일 경우, 리더는 블록을 제안하고, 그 블록은 트리에 추가된다.

게다가 이 라운드에서 트리에 추가되는 블록은 이 블록 하나뿐이며, 이는 최종화된 경로를 확장시킨다.

리더가 정직하지 않거나 네트워크가 동기적이지 않을 경우, 더 높은 순위의 다른 복제본들도 블록을 제안할 수 있으며, 그들의 블록도 트리에 추가될 수 있다.

어떤 경우에도 프로토콜의 논리는 리더가 제안한 블록에 가장 높은 우선순위를 부여하며, 이 라운드에서 하나 이상의 블록이 트리에 추가된다.

프로토콜이 몇 라운드 동안 최종화된 경로를 확장하지 않더라도, 트리의 높이는 각 라운드마다 계속 증가하므로, 라운드 h 에서 최종화된 경로가 확장될 때 그 길이는 h 가 된다.

따라서 결함 복제본이나 예기치 않게 높은 네트워크 지연으로 인해 지연(*latency*) 이 일시적으로 증가하더라도, 프로토콜의 처리량(*throughput*) 은 본질적으로 일정하게 유지된다.

합의 프로토콜은 복제본 간에 전송되는 메시지를 인증하기 위해 디지털 서명에 의존한다.

프로토콜을 구현하기 위해, 각 복제본은 서명 체계의 공개 검증 키(*public verification key*)를 가진다.

복제본과 공개키의 연관 관계는 NNS가 유지하는 레지스트리에서 얻는다.

2.3 메시지 라우팅

1.7절에서 논의했듯이, IC의 기본 계산 단위는 *캐니스터(canister)* 라고 불린다.

IC는 캐니스터 내 프로그램을 실행하기 위한 런타임 환경을 제공하며, 메시지 전달을 통해 다른 캐니스터 및 외부 사용자와 통신할 수 있게 한다.

합의 계층은 입력들을 *페이로드(payload)* 로 묶고, 이 페이로드들은 블록에 배치된다.

블록이 최종화되면, 해당 페이로드들은 *메시지 라우팅 계층(message routing layer)* 으로 전달되고, 그 다음 *실행 환경(execution environment)* 에 의해 처리된다.

실행 환경은 복제 상태 기계의 캐니스터 상태를 업데이트하고 출력을 생성하며, 이러한 출력은 메시지 라우팅 계층에서 처리된다.

두 가지 유형의 입력을 구분하는 것이 유용하다:

- 인그레스 메시지(ingress messages): 외부 사용자로부터 온 메시지;
- 크로스 서브넷 메시지(cross-subnet messages): 다른 서브넷의 캐니스터로부터 온 메시지.

또한 두 가지 유형의 출력을 구분할 수 있다:

- 인그레스 메시지 응답(ingress message responses): 인그레스 메시지에 대한 응답(외부 사용자가 검색 가능함);
- 크로스 서브넷 메시지(cross-subnet messages): 다른 서브넷의 캐니스터로 향하는 메시지.

합의로부터 페이로드를 수신하면, 그 안의 입력들은 여러 *입력 큐(input queues)* 에 배치된다.

서브넷에서 실행 중인 각 캐니스터 C 에 대해 여러 개의 입력 큐가 있으며 — C 로 향하는 인그레스 메시지용 큐 하나와, C 가 통신하는 각 다른 캐니스터 C' 별로 큐가 존재한다.

(C' 가 C 와 동일한 서브넷에 있지 않은 경우, 이는 크로스 서브넷 메시지이다.)

각 라운드에서, 실행 계층은 이러한 큐의 일부 입력을 소비하고, 관련 캐니스터들의 복제 상태를 업데이트하며, 여러 *출력 큐(output queues)* 에 출력을 배치한다.

서브넷에서 실행 중인 각 캐니스터 C 는 여러 출력 큐를 가지며, C 가 통신하는 각 다른 캐니스터 C' 에 대해 각각의 큐가 존재한다.

(C' 가 동일한 서브넷에 있지 않은 경우, 이는 크로스 서브넷 메시지이다.)

메시지 라우팅 계층은 이러한 출력 큐의 메시지들을 가져와 *서브넷 간 스트림(subnet-to-subnet streams)* 에 배치하며, 이는 실제로 메시지를 다른 서브넷으로 전송하는 *크로스넷 전송 프로토콜(crossnet transfer protocol)* 에 의해 처리된다.

이러한 출력 큐 외에도, *인그레스 기록(ingress history)* 이라는 데이터 구조가 존재한다.

인그레스 메시지가 캐니스터에 의해 처리되면, 해당 인그레스 메시지에 대한 *응답(response)* 이 이 데이터 구조에 기록된다.

그 시점에서, 인그레스 메시지를 제공한 외부 사용자는 해당 응답을 검색할 수 있다.

(*인그레스 기록* 은 모든 인그레스 메시지의 전체 이력을 유지하지는 않는다.)

복제 상태는 캐니스터들의 상태뿐 아니라, 위에서 언급한 큐와 스트림, *인그레스 기록* 데이터 구조 등 *시스템 상태(system state)* 를 포함한다.

따라서 메시지 라우팅과 실행 계층은 서브넷의 복제 상태를 업데이트하고 유지하는 데 모두 관여한다.

이 모든 상태는 완전히 *결정적(deterministic)* 으로 업데이트되어야 하며, 따라서 모든 복제본은 *정확히 동일한 상태* 를 유지해야 한다.

또한, 합의 계층은 메시지 라우팅 및 실행 계층과 분리되어 있다.

즉, 합의 블록체인의 어떤 분기(fork)도 그 페이로드가 메시지 라우팅으로 전달되기 전에 해결되며, 실제로 합의는 메시지 라우팅과 정확히 동기화될 필요가 없으며, 다소 앞서 진행될 수도 있다.

2.3.1 라운드별 인증된 상태

각 라운드에서, 서브넷의 *일부* 상태가 인증될 것이다.

라운드별 인증된 상태(per-round certified state) 는 체인 키 암호화를 사용하여 인증된다. 기타 사항과 더불어, 주어진 라운드의 인증된 상태는 다음으로 구성된다.

- 최근에 서브넷 간 스트림에 추가된 *크로스 서브넷 메시지(cross-subnet messages)*;
- *인그레스 기록(ingress history)* 데이터 구조를 포함하는 *기타 메타데이터(other metadata)*.

라운드별 인증된 상태 는 임계값 서명(1.6.1절 참조)을 사용하여 인증된다.

라운드별 인증된 상태는 IC에서 여러 방식으로 사용된다:

- 출력 인증(Output authentication). 크로스 서브넷 메시지와 인그레스 메시지에 대한 응답은 라운드별 인증된 상태를 사용하여 인증된다.
- 비결정성의 방지 및 탐지(Preventing and detecting non-determinism). 합의는 각 복제본이 동일한 순서로 입력을 처리함을 보장한다.
각 복제본이 이러한 입력을 결정적으로 처리하므로, 각 복제본은 동일한 상태를 얻어야 한다.
그러나 IC는 (우발적인) 비결정적 계산이 발생할 경우 이를 방지하고 탐지하기 위한 추가적인 강건성 레이어로 설계되었다.
라운드별 인증된 상태는 이를 위해 사용되는 메커니즘 중 하나이다.
- 합의와의 조정(Coordination with consensus). 라운드별 인증된 상태는 두 가지 방식으로 실행 계층과 합의 계층을 조정하는 데에도 사용된다:
 - 합의가 실행보다 앞서 달리고 있는 경우(실행의 진행은 상태가 인증된 마지막 라운드에 의해 결정됨), 합의는 “스로틀(throttled)” 될 것이다.
 - 합의에 대한 입력은 특정 유효성 검사를 통과해야 하며, 이러한 유효성 검사는 모든 복제본이 동의해야 하는 인증된 상태에 의존할 수 있다.

2.3.2 쿼리 호출 대 업데이트 호출

지금까지 설명했듯이, 인그레스 메시지는 서브넷의 모든 복제본에 의해 동일한 순서로 처리되도록 합의를 거쳐야 한다.

그러나 서브넷의 복제 상태를 수정하지 않는 인그레스 메시지에 대해서는 중요한 최적화가 가능하다.

이들은 *쿼리 호출(query calls)* 이라고 불리며 — 다른 인그레스 메시지(즉, *업데이트 호출(update calls)*)와 대비된다.

쿼리 호출은 캐니스터의 상태를 읽고, 경우에 따라 업데이트하는 계산을 수행할 수 있지만, 캐니스터 상태에 대한 어떤 업데이트도 복제 상태에 커밋되지 않는다.

따라서 쿼리 호출은 합의를 거치지 않고 단일 복제본에 의해 직접 처리될 수 있으며, 이는 쿼리 호출로부터 응답을 얻는 데 필요한 지연을 크게 줄여준다.

일반적으로, 쿼리 호출에 대한 응답은 *인그레스 기록(ingress history)* 데이터 구조에 기록되지 않으며, 따라서 위에서 설명한 라운드별 인증된 상태 메커니즘을 사용하여 인증될 수 없다.

그러나 IC는 캐니스터가 특별한 *인증 변수(certified variables)* 에 데이터를 저장할 수 있도록 한다(업데이트 호출을 처리하는 동안).

이러한 변수는 이 메커니즘으로 인증될 수 있다;

따라서 그 값을 인증 변수로 반환하는 쿼리 호출은 여전히 인증될 수 있다.

2.3.3 외부 사용자 인증

인그레스 메시지와 크로스 서브넷 메시지 사이의 주요 차이점 중 하나는 이러한 메시지를 인증하는 데 사용되는 메커니즘이다.

크로스 서브넷 메시지를 인증하는 데는 체인 키 암호화가 사용되는 반면, 외부 사용자로부터의 인그레스 메시지를 인증하는 데는 다른 메커니즘이 사용된다.

외부 사용자에 대한 중앙 레지스트리는 존재하지 않는다.

대신, 외부 사용자는 *사용자 식별자(user identifier)* (일명 *프린시펄(principal)*) 를 사용하여 캐니스터에 자신을 식별하며, 이는 공개 서명 검증 키의 해시이다.

사용자는 이에 대응하는 비밀 서명 키를 보유하며, 이는 인그레스 메시지에 서명하는 데 사용된다.

이러한 서명과 해당 공개키는 인그레스 메시지와 함께 전송된다.

IC는 자동으로 서명을 인증하고, 적절한 캐니스터에 사용자 식별자를 전달한다.

그런 다음 캐니스터는 인그레스 메시지에 지정된 사용자 식별자와 작업의 다른 매개변수에 기반하여 요청된 작업을 인가할 수 있다.

처음 사용하는 사용자는 IC와의 최초 상호작용 동안 키 쌍을 생성하고 공개키로부터 자신의 사용자 식별자를 도출한다.

재방문 사용자는 사용자 에이전트에 저장된 비밀 키를 사용하여 인증된다.

사용자는 서명 위임(signature delegation)을 사용하여 단일 사용자 정체성에 여러 키 쌍을 연결할 수 있다.

이는 단일 사용자가 동일한 사용자 정체성을 사용하여 여러 장치로부터 IC에 접근할 수 있게 해주므로 유용하다.

2.4 실행 계층

실행 계층은 한 번에 하나의 입력을 처리한다.

이 입력은 입력 큐들 중 하나에서 가져오며, 하나의 캐니스터로 전달된다.

이 입력과 캐니스터의 상태에 기반하여, 실행 환경은 캐니스터의 상태를 업데이트하고, 추가로 출력 큐에 메시지를 추가하고 *인그레스 기록(ingress history)* 을 업데이트할 수 있다(이전에 온 인그레스 메시지에 대한 응답을 포함할 수 있다).

각 서브넷은 *분산 의사난수 생성기(distributed pseudorandom generator, PRG)* 에 접근할 수 있다.

의사난수 비트는 *랜덤 테이프(Random Tape)* 라고 불리는 임계값 서명 자체인 시드로부터 파생된다(1.6.1절 및 3절의 더 자세한 내용을 참조).

합의 프로토콜의 각 라운드마다 서로 다른 랜덤 테이프가 존재한다.

랜덤 테이프의 기본 속성은 다음과 같다:

1. 어떤 정직한 복제본에 의해 높이 h 의 블록이 최종화되기 *이전*에는, 높이 $h + 1$ 의 랜덤 테이프가 예측 불가능함이 보장된다.
2. 어떤 정직한 복제본에 의해 높이 $h + 1$ 의 블록이 최종화될 때쯤이면, 해당 복제본은 일반적으로 높이 $h + 1$ 의 랜덤 테이프를 구성하는 데 필요한 모든 조각(shares)을 보유하게 된다.

의사난수 비트를 얻기 위해, 서브넷은 어떤 라운드, 이를테면 h , 에서 실행 계층으로부터의 “시스템 호출(system call)”을 통해 이러한 비트에 대한 요청을 해야 한다.

그 후 시스템은 높이 $h + 1$ 의 랜덤 테이프를 사용하여 나중에 그 요청에 응답할 것이다.

위의 성질 (1)에 의해, 요청 시점에 요청된 의사난수 비트가 예측 불가능함이 보장된다.

합의는 실제로 높이 $h + 1$ 에서 랜덤 테이프와 페이로드를 동시에 메시지 라우팅으로 전달할 것이며;

위의 성질 (2)에 의해, 이는 일반적으로 추가적인 지연을 유발하지 않는다.

2.5 통합하기

IC에서 사용자 요청을 처리하는 전형적인 흐름을 추적해보자.

쿼리 호출(Query call)

1. 사용자의 캐니스터 C 에 대한 쿼리 호출 M 은 사용자의 클라이언트에 의해 경계 노드로 전송되고(1.9절 참조), 경계 노드는 M 을 캐니스터 C 를 호스팅하는 서브넷의 복제본으로 전송한다.
 M 을 수신한 후, 이 복제본은 응답을 계산하고 경계 노드를 통해 사용자에게 다시 보낸다.

업데이트 호출(Update call)

1. 사용자의 캐니스터 C 에 대한 요청 M 은 사용자의 클라이언트에 의해 경계 노드로 전송되고(1.9절 참조), 경계 노드는 M 을 캐니스터 C 를 호스팅하는 서브넷의 복제본으로 전송한다.

2. M 을 수신한 후, 이 복제본은 피어-투-피어 계층(2.1절 참조)을 사용하여 M 을 서브넷의 다른 모든 복제본에게 브로드캐스트한다.
3. M 을 수신한 다음 라운드의 리더(2.2절 참조)는 M 을 다른 입력들과 함께 묶어 리더가 제안하는 블록 B 의 페이로드를 형성한다.
4. 얼마 후, 블록 B 가 최종화되고, 해당 페이로드는 처리를 위해 메시지 라우팅 계층(2.3절 참조)으로 전송된다.
피어-투-피어 계층은 또한 이 블록을 최종화하는 데 합의에 의해 사용됨을 주목하라.
5. 메시지 라우팅 계층은 M 을 캐니스터 C 의 입력 큐에 배치할 것이다.
6. 얼마 후, 실행 계층(2.4절 참조)은 M 을 처리하여 캐니스터 C 의 내부 상태를 업데이트할 것이다.

어떤 상황에서는, 캐니스터 C 가 요청 M 에 대한 응답 R 을 즉시 계산할 수 있을 것이다.

이 경우, R 은 인그레스 기록 데이터 구조에 배치된다.

다른 상황에서는, 요청 M 을 처리하는 데 다른 캐니스터에 대한 요청을 수행해야 할 수 있다.

이 예에서, 이 특정 요청 M 을 처리하기 위해 캐니스터 C 가 다른 서브넷에 있는 캐니스터 C' 에 요청 M' 을 해야 한다고 가정하자.

이 두 번째 요청 M' 은 C 의 출력 큐에 배치되며, 그 다음 다음 단계들이 수행된다.

7. 얼마 후, 메시지 라우팅은 M' 을 적절한 크로스 서브넷 스트림으로 이동시키며, 이는 결국 캐니스터 C' 를 호스팅하는 서브넷으로 전송될 것이다.
8. 두 번째 서브넷에서, 요청 M' 은 첫 번째 서브넷으로부터 수신되고, 결국 두 번째 서브넷에서 합의와 메시지 라우팅을 거쳐 실행에 의해 처리된다.
실행 계층은 캐니스터 C' 의 내부 상태를 업데이트하고 요청 M' 에 대한 응답 R' 을 생성한다.
응답 R' 은 캐니스터 C' 의 출력 큐에 들어가며, 결국 크로스 서브넷 스트림에 배치되어 첫 번째 서브넷으로 다시 전송된다.
9. 첫 번째 서브넷으로 돌아와서, 응답 R' 은 두 번째 서브넷으로부터 수신되고, 결국 첫 번째 서브넷에서 합의와 메시지 라우팅을 거쳐 실행에 의해 처리된다.
실행 계층은 캐니스터 C 의 내부 상태를 업데이트하고 원래의 요청 메시지 M 에 대한 응답 R 을 생성한다.
이 응답은 인그레스 기록 데이터 구조에 기록될 것이다.

어떤 실행 경로가 선택되든, 요청 M 에 대한 응답 R 은 결국 캐니스터 C 를 호스팅하는 서브넷의 인그레스 기록 데이터 구조에 기록될 것이다.

이 응답을 얻기 위해, 사용자의 클라이언트는 일종의 “쿼리 호출”(2.3.2절 참조)을 수행해야 한다.

2.3.1절에서 논의했듯이, 이 응답은 체인 키 암호학(구체적으로, 임계값 서명 사용)을 통해 인증될 것이다.

인증 로직 자체(즉, 임계값 서명 검증)는 경계 노드로부터 클라이언트가 원래 획득한 서비스 워

커를 사용하여 클라이언트에 의해 수행될 것이다.

3 체인 키 암호학 I: 임계값 서명

IC의 체인 키 암호학의 핵심 구성 요소는 임계값 서명 체계(threshold signature scheme) [Des87] 이다.

IC는 여러 목적을 위해 임계값 서명을 사용한다.

서브넷의 복제본 수를 n 이라 하고 손상된 복제본 수의 상계를 f 라 하자.

- 합의 계층(Consensus Layer) 은 $(f + 1)$ -out-of- n 임계값 서명을 사용하여 *랜덤 비콘(random beacon)* 을 실현한다(5.5절 참조).
- 실행 계층(Execution Layer) 은 $(f + 1)$ -out-of- n 임계값 서명을 사용하여 *랜덤 테이프(random tape)* 를 실현하며, 이는 캐니스터에 예측 불가능한 의사난수 숫자를 제공하는 데 사용된다(7.1절 참조).
- 실행 계층(Execution Layer) 은 $(n - f)$ -out-of- n 임계값 서명을 사용하여 *복제 상태를 인증(certify the replicated state)* 한다.
이는 서브넷의 출력물을 인증하는 데(1.6.1절 참조)와 IC의 *체인 진화 기술(chain-evolution technology)* 의 *빠른 진행(fast-forwarding)* 기능을 구현하는 데(8.2절 참조) 모두 사용된다.

첫 두 가지 응용(랜덤 비콘과 랜덤 테이프)에 대해서는, 임계값 서명이 *유일(unique)* 해야만 한다, 즉 주어진 공개키와 메시지에 대해 유효한 서명은 단 하나여야 한다.

이는 우리가 서명을 의사난수 생성기의 시드로 사용하며, 그러한 임계값 서명을 계산하는 모든 복제본이 동일한 시드에 합의해야 하기 때문에 필요하다.

3.1 임계값 BLS 서명

우리는 임계값 설정에 쉽게 적용될 수 있는 BLS 서명 체계 [BLS01] 를 기반으로 임계값 서명을 구현한다.

일반(즉, 비임계값) BLS 서명 체계는 소수 차수 q 를 갖는 두 군 $\mathbb{G}$ 와 $\mathbb{G}'$ 를 사용한다.

$\mathbb{G}$ 는 $g \in \mathbb{G}$ 에 의해 생성되고 $\mathbb{G}'$ 는 $g' \in \mathbb{G}'$ 에 의해 생성된다고 가정한다.

또한 입력을 $\mathbb{G}'$ 로 사상하는 해시 함수 $H_{\mathbb{G}'}$ 가 있다고 가정하며(그리고 이를 랜덤 오라클로 모델링한다), 비밀 서명 키는 $x \in \mathbb{Z}_q$ 이고 공개 검증 키는 $V := g^x \in \mathbb{G}$ 이다.

비임계값 설정에서 메시지 m 을 서명하기 위해, 서명자는 $h' \leftarrow H_{\mathbb{G}'}(m) \in \mathbb{G}'$ 를 계산한 다음 서명 $\sigma := (h')^x \in \mathbb{G}'$ 를 계산한다.

그러한 서명이 유효한지 검증하기 위해서는 $\log_g h' = \log_V \sigma$ 인지 테스트해야 한다.

이 테스트를 효율적으로 수행할 수 있도록, BLS 체계는 군 $\mathbb{G}$ 와 $\mathbb{G}'$ 상의 *페어링(pairing)* 개념을 사용하며, 이는 $\mathbb{G}$ 와 $\mathbb{G}'$ 가 특수한 유형의 *타원곡선* 일 때 사용 가능한 대수적 연산이다.

여기서는 페어링과 타원곡선의 세부사항으로 들어갈 수 없다.

자세한 내용은 [BLS01] 을 보라.

BLS 서명은 위에서 언급했듯이 서명이 유일하다는 좋은 성질을 가진다.

t -out-of- n 임계값 설정에서, n 개의 복제본이 있으며 그중 임의의 t 개를 사용하여 메시지에 대한 서명을 생성할 수 있다.

좀 더 자세히 말하면, 각 복제본 P_j 는 비밀 서명 키 $x \in \mathbb{Z}_q$ 의 조각 $x_j \in \mathbb{Z}_q$ 를 보유하며, 이는 P_j 가 비공개로 보유하고, 군 원소 $V_j := g^{x_j}$ 는 공개적으로 이용 가능하다.

조각들 $(x_1, \dots, x_n)$ 은 x 의 t -out-of- n 비밀 분할(secret-sharing) 이다(3.4절 참조).

메시지 m 이 주어지면, 복제본 P_j 는 다음과 같은 서명 조각(signature share) 을 생성할 수 있다:

$$\sigma_j := (h')^{x_j} \in \mathbb{G}'$$

여기서 이전과 같이 $h' \leftarrow H_{\mathbb{G}'}(m)$ 이다.

그러한 서명 조각이 유효한지 검증하기 위해서는 $\log_{h'} \sigma_j = \log_g V_j$ 인지 테스트해야 한다.

이산 로그를 계산하는 것은 불가능하지만, 이는 페어링을 사용하여 확인할 수 있으며 — 사실상 이는 공개키 V_j 를 가진 일반 BLS 서명의 유효성 검사와 정확히 동일하다.

복원 속성(reconstruction property)

이 체계는 다음의 복원 속성을 만족한다:

서로 다른 복제본들이 기여한, 메시지 m 에 대한 임의의 t 개의 유효한 서명 조각 σ_j 가 주어지면, 공개 검증 키 하에서 m 에 대한 유효한 BLS 서명을 효율적으로 계산할 수 있다.

사실, σ 는 다음과 같이 계산될 수 있다:

$$\sigma = \prod_j \sigma_j^{\lambda_j}$$

여기서 λ_j 들은 기여한 t 개 복제본의 인덱스들만으로 효율적으로 계산될 수 있다.

보안 속성(security property)

$\mathbb{G}$ 에 대한 합리적인 난해성 가정 하에서, 그리고 $H_{\mathbb{G}'}$ 를 랜덤 오라클로 모델링하면, 이 체계는 다음의 보안 속성을 만족한다:

최대 f 개의 복제본이 적대자에 의해 손상될 수 있다고 가정하자.
그러면 적대자가, 적어도 $t - f$ 개의 정직한 복제본으로부터 해당 메시지에 대한 서명 조각을 얻지 않는 한, 메시지에 대한 유효한 서명을 계산하는 것은 불가능하다.

3.2 분산 키 배포

임계값 BLS 를 구현하기 위해서는 비밀 서명 키의 조각을 복제본들에게 배포하는 방법이 필요하다.

이를 수행하는 한 가지 방법은 *신뢰 당사자(trusted party)* 가 이러한 조각들을 직접 모두 계산하여 모든 복제본에게 배포하는 것이다.

불행히도, 이는 단일 실패 지점을 만든다.

대신, 우리는 *분산 키 생성(distributed key generation, DKG) 프로토콜* 을 사용하며, 이는 복제본들이 안전한 분산 프로토콜을 사용하여 사실상 그러한 신뢰 당사자의 로직을 수행할 수 있도록 한다.

현재 구현된 프로토콜의 고수준 아이디어를 개략적으로 설명한다.

자세한 내용은 [Gro21] 을 참조하라.

사용되는 DKG 프로토콜은 본질적으로 비대화형이다.

이 프로토콜은 두 가지 필수 요소를 사용한다:

- 공개 검증 가능 비밀 분할(PVSS, *publicly verifiable secret sharing*) 체계, 그리고
- 합의 프로토콜(*consensus protocol*).

어떤 합의 프로토콜도 사용할 수 있지만, 놀랍지 않게도, 우리가 사용하는 것은 5절의 것이다(8절도 참조).

3.3 가정

기본 가정들은 1절에서 개략한 것과 동일하다:

- 비동기 통신, 그리고
- $f < n/3$.

우리는 합의 프로토콜이 활성성(liveness)을 달성함을 보장하기 위해서만(5.1절과 같이) *부분 동기성 가정(partial synchrony assumption)* 을 간접적으로 사용한다.

또한 *t-out-of-n 임계값 서명 체계*에 대해 다음을 가정한다

$$f < t \leq n - f, \quad (1)$$

이는 (그 밖의 것들과 더불어) (1) 손상된 복제본들이 단독으로 서명할 수 없고, (2) 정직한 복제본들이 단독으로 서명할 수 있음을 보장한다.

또한 각 복제본이 일부 공개키들과 연관되어 있으며, 각 복제본은 해당하는 개인키를 보유하고 있다고 가정한다.

하나의 공개키는 서명 키(5.4절의 것과 동일)이다.

또 다른 공개키는 PVSS 체계를 구현하는 데 필요한 특정 공개키 암호화 체계를 위한 공개 암호화 키이다(세부사항은 뒤따른다).

3.4 PVSS 체계

위에서 도입한 생성원 $g \in \mathbb{G}$ 가 생성하는 소수 차수 q 의 군 $\mathbb{G}$ 를 두자.

$s \in \mathbb{Z}_q$ 를 비밀이라고 하자.

s 의 t -out-of- n 샤미르 비밀 분할(Shamir secret-sharing) 은 벡터 $(s_1, \dots, s_n) \in \mathbb{Z}_q^n$ 이며,

$$s_j := a(j) \quad (j = 1, \dots, n), \quad (2)$$

그리고

$$a(x) := a_0 + a_1x + \dots + a_{t-1}x^{t-1} \in \mathbb{Z}_q[x] \quad (3)$$

는 차수가 t 보다 작은 다항식으로 $a_0 := s$ 이다.

이러한 비밀 분할의 핵심 성질은 다음과 같다:

- s_j 들 중 임의의 t 개로부터(다항식 보간을 통해) 비밀 $s = a_0 = a(0)$ 을 효율적으로 계산할 수 있고,
- $a_1, \dots, a_{t-1}$ 이 $\mathbb{Z}_q$ 상에서 균등하고 독립적으로 선택되면, t 개 미만의 s_j 들의 임의의 집합은 비밀 s 에 관한 정보를 전혀 누설하지 않는다.

고수준에서, PVSS 체계는 하나의 복제본 P_i (이를 *딜러(dealer)* 라고 부른다) 가 이러한 분할을 취하고, *거래물(dealing)* 이라 불리는 객체를 계산하도록 허용하는데, 이 객체에는 다음이 포함된다

- 군 원소들의 벡터 $(A_0, \dots, A_{t-1})$, 여기서 $k = 0, \dots, t-1$ 에 대해 $A_k := g^{a_k}$,
- 암호문들의 벡터 $(c_1, \dots, c_n)$, 여기서 c_j 는 P_j 의 공개 암호화 키 하에서 s_j 의 암호화이며,
- 각 c_j 가 실제로 그러한 조각을 암호화한다는(더 정확히는 각 c_j 가 다음을 만족하는 값 s_j 로 복호화된다는) 비대화형 영지식 증명 π :

$$g^{s_j} = \prod_{k=0}^{t-1} A_k^{j^k} = g^{a(j)}. \quad (2)$$

우리의 DKG 프로토콜의 전체 보안을 확립하기 위해, PVSS 체계는 적절한 수준의 선택암호문 공격 보안(CCA 보안)을 제공해야 함을 주목한다.

구체적으로, 딜러는 자신의 정체성을 *연관 데이터(associated data)* 로 거래물에 내장해야 하며, 암호화된 조각들은, 적대자가 거래물을 생성할 때 사용한 연관 데이터와는 다른 연관 데이터 하에서 임의의 거래물을 복호화할 수 있는 선택암호문 공격 하에서도, 은닉성을 유지해야 한다.

효율성에 크게 신경 쓰지 않는다면 PVSS 체계를 구현하는 것은 쉽다.

아이디어는 ElGamal 유사 암호화 체계를 사용하여 각 s_j 를 비트 단위로 암호화하고, 그 다음 관계 (2)에 대한 표준 비대화형 영지식 증명을 사용하는 것이다.

이는 적절한 시그마 프로토콜(Sigma protocol) [CDS94] 에 대한 Fiat-Shamir 변환

[FS86]의 표준 적용에 기반할 것이다.

이 방식은 다항시간 체계를 제공하지만, 그다지 실용적이지는 않다.

그러나 이러한 유형의 체계를 최적화할 수 있는 많은 방법이 존재한다.

IC에서 사용되는 고도로 최적화된 PVSS 체계의 세부사항은 [Gro21]을 보라.

3.5 기본 DKG 프로토콜

PVSS 체계와 합의 프로토콜을 사용하면, 기본 DKG 프로토콜은 매우 간단하다.

1. 각 복제본은 무작위 비밀에 대한 *서명된 거래물(signed dealing)*을 다른 모든 복제본에게 브로드캐스트한다.
이러한 서명된 거래물은 거래물, 딜러의 정체성, 그리고 딜러의 공개 서명 키 하의 거래물에 대한 서명을 포함한다.
이러한 서명된 거래물은 올바른 구문 형식을 가지고, 서명과 비대화형 영지식 증명이 유효하면 *유효(valid)*하다고 한다.
2. 합의를 사용하여, 복제본들은 $f + 1$ 개의 유효한 서명된 거래물(서로 다른 딜러로부터)을 포함하는 집합 S 에 합의한다.
3. 집합 S 의 i 번째 거래물이 군 원소 벡터 $(A_{i,0}, \dots, A_{i,t-1})$ 와 암호문 벡터 $(c_{i,1}, \dots, c_{i,n})$ 를 포함한다고 하자.
그러면 임계값 서명 체계의 공개 검증 키는

$$V := \prod_i A_{i,0}. \quad (4)$$

비밀 서명 키는 암묵적으로 다음과 같이 정의됨을 주목하라

$$x := \log_g V. \quad (5)$$

P_j 의 비밀 서명 키 x 의 조각은

$$x_j := \sum_i s_{i,j}, \quad (6)$$

여기서 $s_{i,j}$ 는 P_j 의 비밀 복호 키 하에서 $c_{i,j}$ 를 복호화한 값이다.

복제본 P_j 의 공개 검증 키는

$$V_j := \prod_i \prod_{k=0}^{t-1} A_{i,k}^{j^k} = g^{x_j}. \quad (7)$$

조각들 x_j 가 x 의 t -out-of- n *샴ір 비밀 분할*을 구성함을 주목하라.

그에 따라, (1)에 나타난 λ_j 값들은 라그랑주 보간 계수일 뿐이다.

이는 3.1절에서 기술한 복원 속성을 확립한다.

3.1절에서 기술한 보안 속성에 대해서는, $H_{\mathbb{G}}$ 를 랜덤 오라클로 모델링하고, PVSS 체계가 안

전하며, 페어링이 가능한 군 $\mathbb{G}$ 와 $\mathbb{G}'$ 가 특정 유형의 *one-more 디피-헬만*(one-more Diffie-Hellman) 난해성 가정을 만족한다고 가정하면 성립함을 증명할 수 있다. 이는 어떤 효율적인 적대자도 다음 게임에서 무시할 수 없는 확률로 승리할 수 없다고 말할 수 있는 형태로 진술될 수 있다:

도전자(challenger)는 무작위로 $\mu_1, \dots, \mu_k \in \mathbb{Z}_q$ 및 $\nu_1, \dots, \nu_\ell \in \mathbb{Z}_q$ 를 선택하고, 적대자에게 $\{g^{\mu_i}\}_{i=1}^k$ 및 $\{(g')^{\nu_j}\}_{j=1}^\ell$ 를 제공한다.

적대자는 일련의 질의 $\{\kappa_{i,j}\}_{i,j}$ 형태의 벡터를 도전자에게 수행하고, 도전자는 다음으로 응답한다

$$\prod_{i,j} ((g')^{\mu_i \nu_j})^{\kappa_{i,j}}. \quad (8)$$

게임을 종료하기 위해, 적대자는 벡터 $\{\lambda_{i,j}\}_{i,j}$ 와 군 원소 $h' \in \mathbb{G}'$ 를 출력하며, 다음이 성립하고

$$h' = \prod_{i,j} ((g')^{\mu_i \nu_j})^{\lambda_{i,j}} \quad (9)$$

출력 벡터 $\{\lambda_{i,j}\}_{i,j}$ 가 질의 벡터들의 선형 결합이 아니면 게임에서 승리한다.

이러한 유형의 *one-more 디피-헬만* 가정은 $t > f + 1$ 인 경우에 필요하지만, $t = f + 1$ 인 경우에는 더 약한 가정으로 충분하며(소위 co-CDH 가정), 일반 BLS 체계의 보안이 이에 기반한다.

3.6 재분할 프로토콜

기본 DKG 프로토콜은, 새로운 무작위 비밀 x 의 분할을 생성하는 대신 이전에 분할된 비밀의 새로운 무작위 분할을 생성하도록 쉽게 수정될 수 있다.

- 기본 프로토콜의 1단계를 수정하여, 각 복제본이 자신의 기존 조각에 대한 서명된 거래물을 브로드캐스트하도록 한다.
- 2단계를 수정하여, t 개의 유효한 서명된 거래물 집합에 합의하도록 한다. 또한 각 거래물이 실제로 적절한 기존 조각에 대한 거래물인지 검증된다(이는 i 번째 거래물에서 $A_{i,0}$ 의 값이 이전의 V_i 값과 동일해야 함을 의미한다).
- 3단계에서는 새로운 x_j (및 V_j) 값의 계산이 i 에 대한 합(및 곱)에 라그랑주 보간 계수로 가중치를 두는 방식으로 이루어진다.

4 피어-투-피어 계층

피어-투-피어 계층의 역할은 서브넷의 복제본들 사이에서 프로토콜 메시지를 전송하는 것이다.

이러한 프로토콜 메시지는 다음으로 구성된다:

- 합의를 구현하는 데 사용되는 메시지, 예를 들어 블록 제안, 공증(notarizations) 등 (5절 참조);
- 인그레스 메시지(6절 참조).

기본적으로, 피어-투-피어가 제공하는 서비스는 “최선의 노력(best effort)” 브로드캐스트 채널이다:

정직한 복제본이 메시지를 브로드캐스트하면, 그 메시지는 결국 서브넷의 모든 정직한 복제본에게 전달된다.

설계 목표는 다음을 포함한다:

- 제한된 자원(Bounded resources). 모든 알고리즘은 제한된 자원(메모리, 대역폭, CPU) 내에서 작동해야 한다.
- 우선순위(Prioritization). 서로 다른 메시지는 특정 속성(예: 유형, 크기, 라운드)에 따라 다른 우선순위로 처리될 수 있으며, 이러한 우선순위는 시간이 지남에 따라 변경될 수 있다.
- 효율성(Efficiency). 낮은 지연 시간보다 높은 처리량이 더 중요하다.
- DOS/스팸 복원력(DOS/SPAM resilience). 손상된 복제본은 정직한 복제본들이 서로 통신하는 것을 방해해서는 안 된다.

합의 프로토콜에서, 특히 블록 제안과 같이(상당히 클 수 있는) 일부 메시지는 모든 복제본에 의해 재브로드캐스트될 것임을 관찰하라.

이는 해당 프로토콜의 올바른 동작을 보장하기 위해 필요하다.

그러나 순진하게 구현된다면, 이는 막대한 자원 낭비가 될 것이다.

모든 복제본이 동일한 메시지를 브로드캐스트하는 것을 피하기 위해, 피어-투-피어 계층은 *광고-요청-전달(advertise-request-deliver)* 메커니즘을 사용한다.

(큰) 메시지를 직접 브로드캐스트하는 대신, 메시지에 대한 (작은) *광고(advertisement)* 를 브로드캐스트할 것이다: 어떤 복제본이 메시지에 대한 그러한 광고를 수신하고, 그 메시지를 아직 수신하지 않았으며, 그 메시지가 중요하다고 판단하면, 그 메시지의 *전달(delivered)* 을 *요청(request)* 할 것이다.

이 전략은 더 높은 지연을 감수하는 대가로 대역폭 사용을 감소시킨다.

작은 메시지의 경우, 이러한 교환은 가치가 없으며, 광고 대신 메시지를 직접 보내는 것이 더 타당하다.

상대적으로 작은 서브넷의 경우, 메시지를 브로드캐스트하려는 복제본은 서브넷의 모든 복제본에게 광고를 보낼 것이며, 각 복제본은 그 후 메시지의 전달을 요청할 수 있다.

더 큰 서브넷의 경우, 이 광고-요청-전달 메커니즘은 *오버레이 네트워크(overlay network)* 상에서 동작할 수 있다.

오버레이 네트워크는 서브넷의 복제본들로 구성된 정점들을 갖는 연결된 무방향 그래프이다.

이 그래프에서 간선으로 연결된 두 복제본은 피어이며, 복제본은 자신의 피어와만 통신한다.

따라서 복제본이 메시지를 브로드캐스트하려고 할 때, 그 메시지에 대한 광고를 자신의 피어들에게 보낸다.

피어들은 메시지의 전달을 요청할 수 있으며, 메시지를 수신하면, 특정 조건이 충족되는 경우, 그 피어들은 자신의 피어들에게 그 메시지를 광고할 것이다.

이는 본질적으로 *가십(gossip) 네트워크*이다.

이 전략 또한 더 높은 지연을 감수하는 대가로 대역폭 사용을 감소시킨다.

5 합의 계층

IC의 합의 계층의 역할은 입력들을 순서화하여 서브넷의 모든 복제본이 이러한 입력을 동일한 순서로 처리하도록 하는 것이다. 이 문제에 대한 프로토콜은 문헌에 많이 존재한다. IC는 새로운 합의 프로토콜을 사용하며, 여기서는 이를 고수준에서 설명한다. 더 자세한 내용은 논문 [CDH+21] 을 보라(특히 그 논문의 Protocol ICC1).

모든 안전한 합의 프로토콜은 (대략적으로 말해) 다음의 두 속성을 보장해야 한다:

- 안전성: 모든 복제본이 실제로 동일한 입력 순서에 동의해야 한다, 그리고
- 활성성: 모든 복제본이 지속적으로 진전해야 한다.

논문 [CDH+21] 은 IC 합의 프로토콜이 이 두 속성을 모두 만족함을 증명한다.

IC 합의 프로토콜은 다음과 같이 설계되었다

- 매우 단순하고, 그리고
- 강건하다: 일부 복제본이 악의적일 때 성능이 점진적으로 저하된다.

5.1 가정

서론에서 논의했듯이, 우리는 다음을 가정한다

- n 개 복제본의 서브넷, 그리고
- 최대 $f < n/3$ 개의 복제본이 결함이 있다.

결함이 있는 복제본은 임의의, 악의적인(즉, 비잔틴) 행위를 보일 수 있다.

우리는 통신이 *비동기적*이며, 복제본 사이에 전송된 메시지의 지연에 대한 *사전* 상한이 없다고 가정한다. 사실, 메시지 전달의 스케줄링은 완전히 적대자의 통제 하에 있을 수 있다. IC 합의 프로토콜은 이러한 매우 약한 통신 가정 하에서 안전성을 보장한다. 그러나 활성성을 보장하기 위해서는, (대략적으로 말해) 네트워크가 짧은 시간 구간 동안 주기적으로 동기적일 것이라고 말하는 형태의 *부분 동기성*을 가정해야 한다. 이러한 동기 구간 동안에는, 모든 미전달 메시지가 어떤 고정된 상한 δ 보다 짧은 시간 내에 전달될 것이다. 상한 δ 는 사전에 알려질 필요가 없다 (프로토콜은 합리적인 상한으로 초기화되지만, 그 상한이 너무 작으면 동적으로 적응하여 증가시킬 것이다). 우리가 비동기 네트워크를 가정하든 부분적으로 동기적인 네트워크를 가정하든 관계없이, 한 정직한 복제본에서 다른 정직한 복제본으로 전송된 모든 메시지는 *결국* 전달된다고 가정한다.

5.2 프로토콜 개요

여러 합의 프로토콜과 마찬가지로, IC 합의 프로토콜은 블록체인에 기반한다. 프로토콜이 진행됨에 따라, 트리의 루트인 특수한 *제네시스 블록*으로부터 블록의 트리가 성장한다. 트리의 제네시스 블록이 아닌 각 블록에는 (기타 내용 외에) 입력 시퀀스로 구성된 *페이로드*와 트리에서 부모 블록의 해시가 포함된다. 정직한 복제본들은 이 트리에 대한 일관된 관점을 가진다: 각 복제본은 이 트리에 대한 서로 다른, 부분적인 관점을 가질 수 있지만, 모든 복제본은 *동일한* 트리를 본다. 또한, 프로토콜이 진행됨에 따라, 이 트리에는 항상 *최종화된* 블록들의 경로가 존재한다. 다시 말해, 정직한 복제본들은 이 경로에 대해 일관된 관점을 가진다: 각 복제본은 이 경로에 대한 서로 다른, 부분적인 관점을 가질 수 있지만, 모든 복제본은 *동일한* 경로를 본다. 이 경로를 따라 있는 블록들의 페이로드에 있는 입력들이 인터넷 컴퓨터의 실행 계층(7절 참조)에 의해 처리될 순서화된 입력들이다.

프로토콜은 *라운드* 단위로 진행된다. 프로토콜의 라운드 h 에서는 *높이* h 의 하나 이상의 블록이 트리에 추가된다. 즉, 라운드 h 에 추가된 블록들은 항상 루트로부터 정확히 거리 h 에 있다. 각 라운드에서, 의사난수 프로세스를 사용하여 각 복제본에 고유한 *순위*가 할당되며, 이는 범위 $0, \dots, n - 1$ 의 정수이다. 이 의사난수 프로세스는 *랜덤 비콘*을 사용하여 구현된다(아래 5.5절 참조). 순위가 가장 낮은 복제본이 그 라운드의 *리더*다. 리더가 정직하고 네트워크가 동기적일 때, 리더는 블록을 제안하고, 그 블록은 트리에 추가될 것이다; 더구나, 이 라운드에서 트리에 추가되는 블록은 *오직* 그 블록 하나일 것이며, 이는 최종화된 경로를 확장할 것이다. 리더가 정직하지 않거나 네트워크가 동기적이지 않을 경우, 더 높은 순위의 다른 복제본들도 블록을 제안할 수 있으며, 그들의 블록도 트리에 추가될 수 있다. 어떤 경우에도, 프로토콜의 로직은 리더가 제안한 블록에 가장 높은 우선순위를 부여하며, 이 라운드에서 *어떤* 블록(들)이든 트리에 추가될 것이다. 프로토콜이 몇 라운드 동안 최종화된 경로를 확장하지 않더라도, 트리의 높이는 각 라운드마다 계속 증가하므로, 라운드 h 에서 최종화된 경로가 확장될 때, 최종화된 경로의 길이는 h 가 될 것이다. 이로부터의 결과는, 결함 있는 복제본이나 예기치 않게 높은 네트워크 지연으로 인해 *지연(latency)*이 때때로 증가하더라도, 프로토콜의 *처리량(throughput)*은 본질적으로 일정하게 유지된다는 것이다.

5.3 추가 속성

PBFT [CL99] 와 HotStuff [YMR+18] 과 마찬가지로(그리고 Tendermint [BKM18] 와는 달리), IC 합의 프로토콜이 가지는 추가 속성은 *낙관적 반응성(optimistic responsiveness)* [PS18] 이다. 이는 리더가 정직할 때, 프로토콜이 네트워크 지연의 상한이 아니라 실제 네트워크 지연의 속도로 진행될 수 있음을 의미한다.

우리는 IC 합의 프로토콜의 단순한 설계가, 비잔틴 실패가 실제로 발생할 때 그 성능이 매우 점진적으로 저하되도록 보장한다는 것도 지적한다. [CWA+09] 에서 지적했듯이, 최근의 합의 연구는 실패가 없는 “낙관적 경우”에서의 성능 향상에 지나치게 집중한 나머지, 결과 프로토콜들이 위험할 정도로 취약해졌고, 실패가 실제로 발생할 때는 사실상 사용 불가능해질 수 있다. 예를 들어, [CWA+09] 는 기존 PBFT 구현들의 처리량(꽤 단순한) 특정 유형의 비잔틴 행위 하에서 0으로 떨어짐을 보인다. 논문 [CWA+09] 는 *강건한 합의(robust consensus)* 를 옹호하는데, 여기서는(여전히 네트워크가 동기적이라는 가정 하에서) 일부 참여자가 실제로 손상되었을 때 *합리적인* 성능을 보장하기 위해 *최적* 조건 하의 *최고* 성능을 부분적으로 희생한다. IC 합의 프로토콜은 실제로 [CWA+09] 의 의미에서 강건하다: 리더가 손상된 라운드(그 자체가 확률 $1/3$ 미만으로 발생함)에서는, 프로토콜이 사실상 다른 참여자가 그 라운드의 리더 역할을 넘겨받아, 별다른 소란 없이, 적시에 프로토콜을 다음 라운드로 진행하도록 허용할 것이다.

5.4 공개키

프로토콜을 구현하기 위해, 각 복제본은 BLS 서명 체계 [BLS01] 를 위한 공개키와, 그에 대응하는 비밀 서명 키를 보유한다. 복제본과 공개키의 연관 관계는 NNS가 유지하는 레지스트리에서 얻는다(1.5절 참조). 이러한 BLS 서명은 복제본이 보낸 메시지를 인증하는 데 사용될 것이다.

프로토콜은 또한 BLS 서명의 *서명 집계(signature aggregation)* 기능 [BGLS03] 을 사용하며, 이는 동일한 메시지에 대한 많은 서명을 하나의 간결한 다중 서명으로 집계할 수 있게 한다. 프로토콜은 이러한 다중 서명을 *공증(notarizations)* (5.7절 참조) 및 *최종화(finalizations)* (5.8절 참조)에 사용하며, 이는 특정 형태의 메시지에 대한 $n - f$ 개 서명의 집계이다.

5.5 랜덤 비콘

위에서 논의한 BLS 서명과 다중 서명에 더해, 프로토콜은 상기 랜덤 비콘을 구현하기 위해 BLS 임계값 서명 체계를 사용한다. 높이 h 에 대한 랜덤 비콘은 높이 h 에 고유한 메시지에 대한 $(f + 1)$ -임계값 서명이다. 프로토콜의 각 라운드에서, 각 복제본은 다음 라운드를 위한 비콘의 자신의 조각을 브로드캐스트하여, 다음 라운드가 시작될 때 모든 복제본이 그 라운드의 비콘을 재구성하기에 충분한 조각을 갖도록 한다. 위에서 논의했듯이, 높이 h 의 랜덤 비콘은 프로토콜의 라운드 h 에서 사용될 각 복제본의 의사난수 순위를 할당하는 데 사용된다. 임계값 서명의 보안 속성 때문에, 적대자는 복제본들의 순위를 한 라운드 이상 미리 예측할 수 없으며, 이러한 순위는 사실상 무작위만큼 좋다. BLS 임계값 서명에 대한 더 많은 내용은 3절을 보라.

5.6 블록 생성

각 복제본은 서로 다른 시점에 *블록 메이커(block maker)*의 역할을 수행할 수 있다. 라운드 h 에서 블록 메이커로서, 복제본은 블록 트리에서 높이 $h - 1$ 의 어떤 블록 B' 의 자식이 될 높이 h 의 블록 B 를 제안한다. 이를 위해, 블록 메이커는 먼저 자신이 알고 있는 모든 입력으로 구성된 *페이로드*를 수집하되(다만 B' 에서 끝나는 트리를 통한 경로의 블록들에 이미 포함된 페이로드의 것들은 제외한다), 블록 B 는 다음으로 구성된다

- 페이로드,
- B' 의 해시,
- 블록 메이커의 순위,
- 블록의 높이 h .

블록 B 를 구성한 후, 블록 메이커는 다음으로 구성된 *블록 제안(block proposal)*을 만든다

- 블록 B ,
- 블록 메이커의 정체성, 그리고
- B 에 대한 블록 메이커의 서명.

블록 메이커는 자신의 블록 제안을 다른 모든 복제본에게 브로드캐스트할 것이다.

5.7 공증

블록은 공증이 되면 사실상 블록 트리에 추가된다. 블록이 공증되기 위해서는, 서로 다른 $n - f$ 개 복제본이 그 공증을 *지원*해야 한다.

높이 h 의 제안된 블록 B 가 주어지면, 복제본은 그 제안이 *유효한지* 판단할 것이며, 이는 B 가 위에서 설명한 구문 형식을 가진다는 뜻이다. 특히, B 는 이미 블록 트리에 있는(즉, 이미 공증된) 높이 h' 의 블록 B' 의 해시를 포함해야 한다. 또한, B 의 페이로드는 특정 조건을 만족해야 한다(특히, 페이로드의 모든 입력은 다양한 제약을 만족해야 하지만, 이러한 제약은 일반적으로 합의 프로토콜과는 독립적이다). 또한(블록 B 에 기록된 대로) 블록 메이커의 순위는(블록 제안에 기록된 대로) 해당 블록을 제안한 복제본에 대해 라운드 h 에서 랜덤 비콘이 할당한 순위와 일치해야 한다.

블록이 유효하고 다른 특정 제약들이 성립하면, 복제본은 다음으로 구성된 B 에 대한 *공증 조각(notarization share)*을 브로드캐스트함으로써 블록의 공증을 *지원*할 것이다

- B 의 해시,
- B 의 높이 h ,
- 지원하는 복제본의 정체성, 그리고
- B 의 해시와 높이 h 로 구성된 메시지에 대한 지원하는 복제본의 서명.

B 에 대한 임의의 $n - f$ 개 공증 조각 집합은 다음으로 구성된 B 에 대한 *공증(notarization)*으로 집계될 수 있다

- B 의 해시,
- B 의 높이 h ,
- $n - f$ 명의 지원 복제본들의 정체성 집합,
- B 의 해시와 높이 h 로 구성된 메시지에 대한 $n - f$ 개 서명의 집계.

복제본이 높이 h 의 검증된 블록을 얻는 즉시, 그 복제본은 라운드 h 를 완료하고, 이후에는 높이 h 의 다른 어떤 블록의 검증도 지원하지 않을 것이다. 이 시점에서, 그러한 복제본은 또한 0 의 공증을 다른 모든 복제본에게 중계(relay) 할 것이다. 이 복제본이 공증을 얻는 방법은 (1) 다른 복제본으로부터 공증을 수신했거나, (2) 자신이 수신한 $n - f$ 개의 검증 조각을 집계했기 때 문일 수 있음을 주목하라.

성장 불변식(growth invariant)은 각 정직한 복제본이 결국 각 라운드를 완료하고 다음 라운 드를 시작하여, 검증된 블록들의 트리가 계속 성장함을 말한다(그리고 이는 부분 동기성이 아니 라, 비동기적 최종 전달 가정만으로도 성립한다). 우리는 아래에서 성장 불변식을 증명한다 (5.11.4절 참조).

5.8 최종화

주어진 높이 h 에는 하나 이상의 검증된 블록이 존재할 수 있다. 그러나 블록이 **최종화**되면, 높 이 h 에 다른 검증된 블록이 없다는 것을 확신할 수 있다. 이를 **안전 불변식(safety invariant)** 이라고 부르자.

블록이 최종화되려면, 서로 다른 $n - f$ 개의 복제본이 그 최종화를 *지원*해야 한다. 복제본에 대해 라운드 h 가 끝나는 시점은 높이 h 의 검증된 블록 B 를 획득했을 때임을 상기하라. 높이 $h + 1$ 에서, 그 복제본은 자신이 블록 B 이외의 높이 h 의 어떤 블록의 검증도 지원했는지를 확인할 것이다(그 복제본이 B 자체의 공증을 지원했을 수도 있고 하지 않았을 수도 있다). 만약 아니라면, 그 복제본은 B 에 대한 **최종화 조각(finalization share)**을 브로드캐스트함으로써 B 의 최종화를 지원할 것이다. 최종화 조각은 형식적으로 검증 조각과 정확히 동일하다(그 러나 검증 조각과 최종화 조각이 혼동되지 않도록 태깅되어 있다). B 에 대한 $n - f$ 개의 최 종화 조각 임의의 집합은 **최종화(finalization)**로 집계될 수 있으며, 이는 형식적으로 검증과 정 확히 동일하다(그러나 다시 적절히 태깅되어 있다). 최종화된 블록을 획득한 어떤 복제본도 그 최종화를 다른 모든 복제본에게 브로드캐스트할 것이다.

우리는 아래에서 안전 불변식을 증명한다(5.11.5절 참조). 안전 불변식의 한 가지 귀결은 다음 과 같다. 두 블록 B 와 B' 가 최종화되었다고 가정하자. 여기서 B 의 높이는 h , B' 의 높이는 $h' \leq h$ 이다. 그러면 안전 불변식은 검증된 블록 트리에서 B' 에서 끝나는 경로가 B 에서 끝 나는 경로의 접두사임을 함의한다(그렇지 않으면, 높이 h' 에 두 개의 검증된 블록이 존재하게 되어 최종화 불변식에 모순된다). 따라서 복제본이 최종화된 블록 B 를 볼 때마다, B 의 모든 조상들을 **암묵적으로 최종화됨**으로 간주할 수 있으며, 안전 불변식 때문에, 이러한(명시적으로 그리고 암묵적으로) 최종화된 블록들에 대해서 안전 속성이 보장된다 — 즉, 모든 복제본이 이 러한 최종화된 블록들의 순서에 동의한다.

5.9 지연 함수

프로토콜은 블록 생성과 공증 활동의 타이밍을 제어하는 두 개의 지연 함수 Δ_m 과 Δ_n 을 사용한다. 이 두 함수 모두 제안하는 복제본의 순위 r 을 비음의 지연량으로 사상하며, 각 함수는 r 에 대해 단조 증가한다고 가정하고, 모든 $r = 0, \dots, n - 1$ 에 대해 $\Delta_m(r) \leq \Delta_n(r)$ 라고 가정한다. 이러한 함수들의 권장 정의는 $\Delta_m(r) = 2\delta r$ 및 $\Delta_n(r) = 2\delta r + \epsilon$ 이며, 여기서 δ 는 한 정직한 복제본에서 다른 정직한 복제본으로 메시지를 전달하는 시간의 상한이고, $\epsilon \geq 0$ 은 프로토콜이 너무 빠르게 실행되지 않도록 하는 “거버너(governor)” 이다. 이러한 정의 하에서, (1) 리더가 정직하고, (2) 메시지가 실제로 정직한 복제본들 사이에서 시간 δ 내에 전달되는 라운드들에서 활성성이 보장될 것이다. 실제로, 주어진 라운드에서 (1) 과 (2) 가 모두 성립하면, 그 라운드에서 리더가 제안한 블록은 최종화될 것이다. 이를 *활성성 불변식(liveness invariant)* 이라고 부르자. 우리는 아래에서 이를 증명한다(5.11.6절 참조).

5.10 예시

그림 2는 블록 트리를 보여준다. 각 블록은 그 높이(30, 31, 32, ...)와 블록 메이커의 순위로 라벨링되어 있다. 그림은 또한 트리의 각 블록이 $\diamond$ 기호로 표시된 바와 같이 공증되었음을 보여준다. 이는 트리의 각 공증된 블록에 대해, 적어도 서로 다른 $n - f$ 개의 복제본이 그 공증을 지원했음을 의미한다. 보듯이, 주어진 높이에서 트리에 하나 이상의 공증된 블록이 존재할 수 있다. 예를 들어, 높이 31에서는 순위 1 과 2 의 블록 메이커가 제안한 두 개의 공증된 블록이 있음을 볼 수 있다. 동일한 일이 높이 34 에서도 발생한다. 또한 높이 36 의 블록이 $\diamond$ 기호로 표시된 바와 같이 명시적으로 최종화되었음을 볼 수 있다. 이는 서로 다른 $n - f$ 개의 복제본이 이 블록의 최종화를 지원했음을 의미하며, 이는 이러한 복제본들(또는 최소한 이들 중 정직한 복제본들)이 다른 어떤 블록의 공증도 지원하지 않았음을 의미한다. 회색으로 음영 처리된 이 블록의 모든 조상들은 암묵적으로 최종화된 것으로 간주된다.

5.11 모두 합치기

이제 프로토콜이 어떻게 동작하는지에 대해 더 자세히 설명한다; 구체적으로, 복제본이 언제 블록을 제안하고 언제 블록의 공증을 지원할지를 보다 정확히 설명한다. 주어진 복제본 P 는 자신이 주어진 라운드 h 로 진입한 시각을 기록할 것이며, 이는 (1) 높이 $h - 1$ 의 블록에 대한 어떤 공증과, (2) 라운드 h 의 랜덤 비콘을 획득했을 때 발생한다. 라운드 h 의 랜덤 비콘이 결정되었으므로, P 는 라운드 h 에 대해 자신의 순위 r_P 와 각 다른 복제본 Q 의 순위 r_Q 를 결정할 수 있다.

5.11.1 랜덤 비콘 세부사항

복제본이 라운드 h 의 랜덤 비콘을 수신했거나, 라운드 h 의 랜덤 비콘을 구성하기에 충분한 조각을 수신하는 즉시, 라운드 h 의 랜덤 비콘을 모든 다른 복제본에게 중계(relay) 할 것이다. 복제본이 라운드 h 에 진입하는 즉시, 라운드 $h + 1$ 의 랜덤 비콘에 대한 자신의 조각을 생성하여 브로드캐스트할 것이다.

5.11.2 블록 생성 세부사항

복제본 P 는 (1) 라운드 시작 이후 적어도 $\Delta_m(r_P)$ 시간 단위가 경과했고, (2) 현재 P 가 보고 있는 더 낮은 순위의 유효한 블록이 없다는 조건 하에서만 자신의 블록 B_P 를 제안할 것이다.

P 가 라운드 h 에 진입할 때 높이 $h - 1$ 의 공증된 블록을 갖는 것이 보장되므로, 자신이 제안하는 블록을 이 공증된 블록(또는 자신이 가지고 있을 수 있는 높이 $h - 1$ 의 다른 공증된 블록)의 자식으로 만들 수 있음을 주목하라. 또한 P 가 B_P 에 대한 제안을 브로드캐스트할 때, B_P 의 부모의 공증을 모든 복제본에게 중계(relay)했음도 보장해야 한다는 점을 주목하라.

복제본 Q 가 순위 $r_P < r_Q$ 인 복제본 P 로부터, (1) 라운드 시작 이후 적어도 $\Delta_m(r_P)$ 시간 단위가 경과했고, (2) 현재 Q 가 보고 있는 r_P 보다 작은 순위의 블록이 없는, 유효한 블록 제안을 보게 되었다고 하자. 그러면 이 시점에서, 아직 그렇게 하지 않았다면, Q 는 이 블록 제안(제안된 블록의 부모 공증과 함께)을 모든 다른 복제본에게 중계(relay) 할 것이다.

5.11.3 공증 세부사항

복제본 P 는 순위 r_Q 인 복제본 Q 가 제안한 유효한 블록 B_Q 에 대해, (1) 라운드 시작 이후 적어도 $\Delta_n(r_Q)$ 시간 단위가 경과했고, (2) 현재 P 가 보고 있는 r_Q 보다 작은 순위의 블록이 없다는 조건 하에서, 그 공증을 지원할 것이다.

5.11.4 성장 불변식의 증명

성장 불변식은 각 정직한 복제본이 결국 각 라운드를 완료하고 다음 라운드를 시작함을 말한다. 모든 정직한 복제본이 라운드 h 를 시작했다고 가정하자. 라운드 h 에서 가장 낮은 순위의 정직한 복제본 P^* 의 순위를 r^* 라고 하자. 결국, P^* 는 (1) 자신의 블록을 제안하거나, (2) 더 낮은 순위의 복제본이 제안한 유효한 블록을 중계(relay)할 것이다. 어떤 경우든, 어떤 블록은 결국 모든 정직한 복제본에 의해 지원되어야 하며, 이는 어떤 블록이 공증되고 모든 정직한 복제본이 라운드 h 를 끝낸다는 것을 의미한다. 모든 정직한 복제본은 또한 라운드 $h + 1$ 의 랜덤 비콘을 구성하는 데 필요한 조각을 수신할 것이며, 따라서 라운드 $h + 1$ 을 시작할 것이다.

5.11.5 안전 불변식의 증명

안전 불변식은 주어진 라운드에서 블록이 최종화되었다면, 그 라운드에서 다른 어떤 블록도 공증될 수 없음을 말한다. 다음은 안전 불변식의 증명이다:

1. 손상된 복제본의 수가 정확히 $f^* \leq f < n/3$ 라고 가정하자.

2. 블록 B 가 최종화되었다면, 그 최종화는 적어도 $n - f - f^*$ 개의 정직한 복제본의 집합 S 에 의해 지원되었어야 한다(집계 서명에 대한 보안 속성에 의해).
3. (모순을 보이기 위해) 또 다른 블록 $B' \neq B$ 가 공증되었다고 가정하자. 그러면 그 공증은 역시 적어도 $n - f - f^*$ 개의 정직한 복제본의 집합 S' 에 의해 지원되었어야 한다(다시, 집계 서명에 대한 보안 속성에 의해).
4. 집합 S 와 S' 는 서로소이다(최종화 로직에 의해).
5. 그러므로 $n - f^* \geq |S \cup S'| = |S| + |S'| \geq 2(n - f - f^*)$ 이고, 이는 $n \leq 3f$ 를 함의하므로, 모순이다.

5.11.6 활성화 불변식의 증명

우리는 네트워크가 시각 t 에 δ -동기적이라고 말하기를, 정직한 복제본들에 의해 시각 t 에 또는 그 이전에 전송된 모든 메시지가 시각 $t + \delta$ 이전에 목적지에 도달하는 경우로 정의한다.

활성성 불변식은 다음과 같이 진술될 수 있다. $\Delta_n(1) \geq \Delta_m(0) + 2\delta$ 라고 가정하자. 또한 주어진 라운드 h 에서 다음이 성립한다고 가정하자:

- 라운드 h 의 리더 P 는 정직하고,
- 라운드 h 에 최초로 진입하는 정직한 복제본 Q 는 시각 t 에 그렇게 하며,
- 네트워크는 시각 t 와 $t + \delta + \Delta_m(0)$ 에 δ -동기적이다.

그러면 라운드 h 에서 P 가 제안한 블록은 최종화될 것이다.

다음은 활성화 불변식의 증명이다:

1. 시각 t 에 부분 동기성 하에서, 모든 정직한 복제본은 시각 $t + \delta$ 이전에 라운드 h 에 진입할 것이다(Q 의 라운드 $h - 1$ 을 종료시킨 공증과 라운드 h 의 랜덤 비콘이 이 시각 이전에 모든 정직한 복제본에게 도달할 것이다).
2. 라운드 h 의 리더 P 는 시각 $t + \delta + \Delta_m(0)$ 이전에 블록 B 를 제안할 것이며, 다시 부분 동기성에 의해, 이 블록 제안은 시각 $t + 2\delta + \Delta_m(0)$ 이전에 다른 모든 복제본에게 전달될 것이다.
3. $\Delta_n(1) \geq \Delta_m(0) + 2\delta$ 이므로, 프로토콜 로직은 각 정직한 복제본이 블록 B 의 공증만을 지원하고 다른 블록은 지원하지 않도록 보장하며, 따라서 B 는 공증되고 최종화될 것이다.

5.12 기타 이슈들

5.12.1 성장 지연(latency)

부분 동기성 가정 하에서, 우리는 성장 불변식의 정량적 버전을 또한 정식화하고 증명할 수 있다. 단순화를 위해, 위에서 권장한 대로 지연 함수를 정의한다고 가정하자: $\Delta_m(r) = 2\delta r$ 및 $\Delta_n(r) = 2\delta r + \epsilon$, 그리고 추가로 $\epsilon \leq \delta$ 라고 가정하자. 시각 t 에, 어떤 정직한 복제본이라도 진입한 가장 높은 번호의 라운드가 h 라고 하자. 라운드 h 에서 가장 낮은 순위의 정직한 복제본 P^* 의 순위를 r^* 라고 하자. 마지막으로, 네트워크가 구간 $[t, t + (3r^* + 2)\delta]$ 의 모든 시각에 대해 δ -동기적이라고 가정하자. 그러면 모든 정직한 복제본은 시각 $t + 3(r^* + 1)\delta$ 이전에 라운드 $h + 1$ 을 시작할 것이다.

5.12.2 로컬로 조정된 지연 함수

복제본이 여러 라운드에 걸쳐 어떤 최종화된 블록도 보지 못하면, 자신의 공증을 위한 지연 함수 Δ_n 을 증가시키기 시작할 것이다. 복제본들은 이러한 로컬로 조정된 공증 지연 함수들에 대해 합의할 필요가 없다.

또한, 복제본들이 지연 함수 Δ_p 를 명시적으로 조정하지는 않지만, 우리는 로컬 시계의 드리프트를 양 지연 함수를 로컬로 조정하는 것으로 수학적으로 모델링할 수 있다.

따라서, 복제본과 라운드에 의해 매개변수화된 많은 지연 함수들이 존재한다. 그럼으로써 활성성에 필요했던 임계 조건 $\Delta_n(1) \geq \Delta_m(0) + 2\delta$ 는 주어진 라운드의 모든 정직한 복제본들에 대해 최대와 최소를 취하여 $\max \Delta_n(1) \geq \min \Delta_m(0) + 2\delta$ 가 된다. 따라서, 충분한 라운드 동안 최종화가 실패하면, 모든 정직한 복제본은 결국 이 조건이 성립할 때까지 자신의 공증 지연을 증가시킬 것이고, 그러면 최종화는 다시 재개될 것이다. 일부 정직한 복제본이 다른 복제본들보다 자신의 공증 지연 함수(latency function)를 더 많이 증가시키더라도, 활성성 측면에서는 불이익이 없다(그러나 성장 지연 측면에서는 있을 수 있다).

5.12.3 공정성(fairness)

합의 프로토콜에서 중요한 또 다른 속성은 공정성이다. 일반적인 정의를 제시하기보다는, 우리는 활성성 불변식이 유용한 공정성 속성 또한 함의함을 단순히 관찰한다. 활성성 불변식은 기본적으로 리더가 정직하고 네트워크가 동기적인 모든 라운드에서, 리더가 제안한 블록이 최종화될 것이라고 말함을 상기하라. 이러한 라운드들에서는, 리더가 정직하다는 사실이(페이로드 크기에 대한 한계를 제외하고) 자신이 알고 있는 모든 입력을 자신의 블록의 페이로드에 포함할 것을 보장한다. 따라서, 매우 대략적으로 말해, 충분한 수의 복제본에 전파된 어떤 입력도 높은 확률로 합리적인 시간 안에 최종화된 블록에 포함될 것이다.

6 메시지 라우팅 계층

1.7절에서 논의했듯이, IC의 기본 계산 단위는 *캐니스터(canister)* 라고 불리며, 이는 *프로그램* 과 그 *상태* 를 모두 포함한다는 점에서 대략 *프로세스(process)* 의 개념과 동일하다. IC는 캐니스터에서 프로그램을 실행하고, 다른 캐니스터 및 외부 사용자와(메시지 전달을 통해) 통신하기 위한 런타임 환경을 제공한다.

합의 계층(5절 참조)은 입력들을 *페이로드*로 묶어 *블록*에 배치하며, 블록들이 최종화됨에 따라, 해당 페이로드들은 *메시지 라우팅 계층*에 전달되고, 이어서 *실행 환경*에 의해 처리되어, 복제 상태 기계 상의 캐니스터들의 상태를 업데이트하고 출력을 생성하며, 이러한 출력은 *메시지 라우팅 계층*에 의해 처리된다.

두 가지 유형의 입력을 구분하는 것이 유용하다:

- 인그레스 메시지(ingress messages): 외부 사용자로부터의 메시지;
- 크로스-서브넷 메시지(cross-subnet messages): 다른 서브넷의 캐니스터로부터 오는 메시지.

또한 두 가지 유형의 출력을 구분할 수 있다:

- 인그레스 메시지 응답(ingress message responses): 인그레스 메시지에 대한 응답(외부 사용자가 조회할 수 있음);
- 크로스-서브넷 메시지: 다른 서브넷의 캐니스터를 대상으로 하는 메시지.

합의로부터 페이로드를 수신하면, 그 페이로드에 있는 입력들은 다양한 *입력 큐(input queues)*에 배치된다. 서브넷에서 실행 중인 각 캐니스터 C 에 대해, 여러 입력 큐가 존재한다 — C 에 대한 인그레스 메시지 전용 큐가 하나 있으며, C 가 통신하는 각 다른 캐니스터 C' 는 자신만의 큐를 갖는다. (C' 가 C 와 동일한 서브넷에 있지 않은 경우, 이는 크로스-서브넷 메시지다.) 아래에 더 자세히 설명하듯이, 각 라운드에서 실행 계층은 이러한 큐들에 있는 일부 입력을 소비하고, 관련 캐니스터들의 복제 상태를 업데이트하며, 다양한 *출력 큐(output queues)*에 출력을 배치한다. 서브넷에서 실행 중인 각 캐니스터 C 에 대해, 여러 출력 큐가 존재한다 — C 가 통신하는 각 다른 캐니스터 C' 가 자신만의 큐를 갖는다. (C' 가 C 와 동일한 서브넷에 있지 않은 경우, 이는 크로스-서브넷 메시지다.) 메시지 라우팅 계층은 이러한 출력 큐에 있는 메시지들을 가져와 *서브넷-간 스트림(subnet-to-subnet streams)*에 배치하고, 이를 실제로 다른 서브넷으로 운반하는 역할을 하는 *크로스넷 전송 프로토콜(crossnet transfer protocol)*에 의해 처리되도록 한다.

이러한 출력 큐들에 추가로, *인그레스 이력(ingress history)* 데이터 구조도 존재한다. 인그레스 메시지가 캐니스터에 의해 처리되면, 그 인그레스 메시지에 대한 *응답*이 이 데이터 구조에 기록될 것이다. 그 시점에서, 해당 인그레스 메시지를 제공한 외부 사용자는 해당 응답을 조회할 수 있게 된다. (참고로, *인그레스 이력*은 모든 인그레스 메시지의 전체 이력을 유지하지 않는다.)

또한 크로스-서브넷 메시지에 더해, 동일한 서브넷 내의 한 캐니스터에서 같은 서브넷의 다른 캐니스터로 가는 *인트라-서브넷 메시지(intra-subnet messages)*도 존재함을 언급해야 한다. 메시지 라우팅 계층은 이러한 메시지를 출력 큐에서 해당 입력 큐로 직접 이동시킨다.

그림 3은 메시지 라우팅 및 실행 계층의 기본 기능을 보여준다. 복제 상태는 캐니스터의 상태뿐 아니라, 위에서 언급한 큐와 스트림 및 *인그레스 이력* 데이터 구조를 포함하는 “시스템 상태(system state)”도 포함함을 주목하라. 따라서, 메시지 라우팅과 실행 계층 모두가 서브넷의 복제 상태를 업데이트하고 유지하는 데 관여한다. 모든 이 상태가 완전히 *결정적*

(*deterministic*) 방식으로 업데이트되어, 모든 복제본이 *정확히* 동일한 상태를 유지하는 것이 필수적이다.

또한 합의 계층은 메시지 라우팅 및 실행 계층과 분리되어 있음을 주목하라. 이는 합의 블록체인에서의 어떤 포크도 그 페이로드가 메시지 라우팅으로 전달되기 전에 해결된다는 의미이며, 실제로 합의는 메시지 라우팅과 보조를 맞추는 필요가 없고 약간 앞서 실행될 수 있다.

6.1 라운드별 인증된 상태

각 라운드에서, 서브넷의 *일부* 상태가 *인증(certified)* 된다. *라운드별 인증된 상태(per-round certified state)* 는 체인 키 암호학(1.6절 참조), 구체적으로 3절에서 언급된 $(n - f)$ -out-of- n 임계값 서명 체계를 사용하여 인증된다. 보다 자세히 말하면, 각 복제본이 주어진 라운드에 대한 라운드별 인증된 상태를 생성한 후, 해당 임계값 서명의 자신의 조각을 생성하여 자신의 서브넷의 다른 모든 복제본에게 브로드캐스트한다. 그러한 조각 $n - f$ 개를 수집하면, 각 복제본은 해당 라운드의 라운드별 인증된 상태에 대한 *증명서(certificate)* 로 기능하는 결과 임계값 서명을 구성할 수 있다. 서명 전에, 라운드별 인증된 상태는 *머클 트리(Merkle tree)* [Mer87]로서 해시됨을 주목하라.

Figure 3: 메시지 라우팅 및 실행 계층

주어진 라운드의 *라운드별 인증된 상태*는 다음으로 구성된다

1. 최근에 서브넷-간 스트림에 추가된 크로스-서브넷 메시지들;
2. *인그레스 이력* 데이터 구조를 포함한 기타 메타데이터;
3. 이전 라운드의 라운드별 인증된 상태의 *머클 트리 루트 해시*.

라운드별 인증된 상태는 서브넷의 전체 복제 상태를 포함하지 않음을 주목하라. 일반적으로 이는 매우 방대하며, 매 라운드마다 모든 이 상태를 인증하는 것은 비실용적일 것이다.⁵

그림 4는 라운드별 인증된 상태가 트리로 조직될 수 있는 방식을 보여준다. 트리의 첫 번째 분기는 각 캐니스터에 대한 다양한 메타데이터를 저장한다(그러나 캐니스터의 전체 복제 상태는 아님). 두 번째 분기는 *인그레스 이력* 데이터 구조를 저장한다. 세 번째 분기는 서브넷-간 스트림에 대한 정보를 저장하며, 각 스트림에 대해 최근에 추가된 크로스-서브넷 메시지의 “윈도우”를 포함한다. 다른 분기들은 여기에서 논의하지 않는 다른 유형의 메타데이터를 저장한다. 그런 다음 이 트리 구조는 머클 트리로 해시될 수 있으며, 본질적으로 이 트리와 동일한 크기와 형태를 가진다.

라운드별 인증된 상태는 IC에서 여러 방식으로 사용된다:

⁵ 그러나 8.2절을 보라.

Figure 4: 트리로 구성된 라운드별 인증된 상태

- 출력 인증. 서브넷 간 메시지와 인그레스 메시지에 대한 응답은 라운드별 인증된 상태를 사용하여 인증된다. 머클 트리 구조를 사용하여, 개별 출력(서브넷 간 메시지 또는 인그레스 메시지 응답)은 머클 트리의 루트에 대한 임계값 서명과, 루트에서 그 출력을 나타내는 리프까지의 머클 트리 경로 위(및 인접)에 있는 해시 값들을 함께 제공함으로써 임의의 당사자에게 인증될 수 있다. 따라서 개별 출력을 인증하는 데 필요한 해시 값의 수는 머클 트리의 *깊이*에 비례하며, 머클 트리의 *크기*가 매우 크더라도 일반적으로 상당히 작다. 그러므로 단일 임계값 서명을 사용하여 많은 개별 출력을 효율적으로 인증할 수 있다.
- 비결정성의 예방 및 탐지. 합의는 각 복제본이 동일한 순서로 입력을 처리함을 보장한다. 각 복제본이 이러한 입력을 결정적으로 처리하므로, 각 복제본은 동일한 상태를 얻어야 한다. 그러나 IC는 (우발적인) 비결정적 계산이 발생할 경우 이를 예방하고 탐지하기 위한 추가적인 강건성 레이어로 설계되었다. 라운드별 인증된 상태는 이를 위해 사용되는 메커니즘 가운데 하나이다. 우리는 인증에 $(n - f)$ -out-of- n 임계값 서명을 사용하고, $f < n/3$ 이므로, 인증되는 상태의 시퀀스는 오직 하나만 존재할 수 있다.

상태 체이닝이 왜 중요한지 보기 위해, 다음 예시를 고려하자. 복제본이 4개, P_1, P_2, P_3, P_4 가 있고, 그중 하나가 손상되었다고 하자, 이를 P_4 라고 하자. 복제본 P_1, P_2, P_3 는 모두 동일한 상태에서 시작한다.

- 라운드 1 에서, 비결정적 계산 때문에, P_1, P_2 는 서브넷 A 로 보낼 메시지 m_1 을 계산하는 반면, P_3 는 서브넷 A 로 보낼 메시지 m'_1 을 계산한다.
- 라운드 2 에서, P_1, P_3 는 서브넷 B 로 보낼 메시지 m_2 를 계산하는 반면, P_2 는 서브넷 B 로 보낼 메시지 m'_2 를 계산한다.
- 라운드 3 에서, P_2, P_3 는 서브넷 C 로 보낼 메시지 m_3 를 계산하는 반면, P_1 은 서브넷 C 로 보낼 메시지 m'_3 를 계산한다.

이는 다음 표에 나타나 있다:

Replica	Round 1	Round 2	Round 3
P_1	$m_1 \rightarrow A$	$m_2 \rightarrow B$	$m'_3 \rightarrow C$
P_2	$m_1 \rightarrow A$	$m'_2 \rightarrow B$	$m_3 \rightarrow C$
P_3	$m'_1 \rightarrow A$	$m_2 \rightarrow B$	$m_3 \rightarrow C$

우리는 복제본 P_1, P_2 , 그리고 P_3 가 각각 개별적으로는 *유효한 시퀀스*의 계산을 수행한다고 가정하지만, 비결정성 때문에 이 시퀀스들이 동일하지 않다고 가정한다. (비결정성이 있어서는 안 되지만, 이 예시에서는 그러하다고 가정한다.)

이제 상태들을 체이닝하지 않는다고 가정하자. P_4 가 손상되어 있고 무엇이든 서명할 수 있으므로, 그는 “ $m_1 \rightarrow A$ ”라고 말하는 라운드 1 상태에 대해 3-중-4 서명을 만들 수 있고, 마찬가지로 “ $m_2 \rightarrow B$ ”라고 말하는 라운드 2 상태에 대해, 그리고 “ $m_3 \rightarrow C$ ”라고 말하는 라운드 3 상태에 대해 3-중-4 서명을 만들 수 있다. *비록 해당 시퀀스*

$$m_1 \rightarrow A, \quad m_2 \rightarrow B, \quad m_3 \rightarrow C \quad (10)$$

가 어떤 유효한 계산 시퀀스와의 양립하지 않을 수 있음에도 불구하고 더 나쁜 것은, 그러한 유효하지 않은 계산 시퀀스가 이후 다른 서브넷에서의 불일치한 상태로 이어질 수 있다는 점이다.

체이닝을 통해, 비결정성이 어느 정도 있더라도, 인증된 상태의 임의의 시퀀스가 정직한 복제본에 의해 실제로 수행된 어떤 유효한 시퀀스의 계산에 대응하도록 보장한다.

-
- 합의와의 조정. 라운드별 인증된 상태는 또한 두 가지 방식으로 실행 계층과 합의 계층을 조정하는 데 사용된다:
 - 합의 스로틀링(throttling). 각 복제본은 자신이 인증된 상태를 가진 가장 최근의 라운드를 추적할 것이며 — 이를 *인증 높이(certified height)*라고 한다. 또한 공증된 블록을 가진 가장 최근의 라운드도 추적할 것이며 — 이를 *공증 높이(notarized height)*라고 한다. 공증 높이가 인증 높이보다 유의미하게 크다면, 이는 실행이 합의보다 뒤처지고 있으며, 합의를 스로틀링해야 한다는 신호다. 이러한 지연은 비결정적 계산 때문일 수도 있고, 혹은 계층 간 보다 양성적인 성능 불일치 때문일 수도 있다. 합의는 5.9절에서 논의한 *지연 함수*를 통해 스로틀링되며 — 구체적으로, 각 복제본은 공증 높이와 인증 높이 사이의 격차가 커짐에 따라 “거버너” 값 ϵ 을 증가시킬 것이다(이는 5.12.2절의 “로컬로 조정된 지연 함수” 개념을 사용한다).
 - 상태-특정 페이로드 검증. 5.7절에서 논의했듯이, 페이로드의 입력은 특정 유효성 검사를 통과해야 한다. 사실, 이러한 유효성 검사는 어느 정도 상태에 의존할 수 있다. 우리가 생략한 세부사항 하나는 각 블록이 라운드 번호를 포함한다는 점이며, 해당 유효성 검사는 그 라운드 번호에 대한 인증된 상태를 기준으로 수행되어야 한다는 이해가 있다는 점이다. 이 검증을 수행해야 하는 복제본은 해당 라운드 번호에 대한 상태가 인증될 때까지 기다린 다음, 그 라운드의 인증된 상태를 사용하여 검증을 수행할 것이다. 이는 비결정적 계산이 있더라도, 모든 복제본이 동일한 유효성 검사를 수행하도록 보장한다(그렇지 않으면, 합의가 멈출 수 있다).
-

6.2 쿼리 콜 vs 업데이트 콜

지금까지 설명한 바와 같이, 인그레스 메시지는 서브넷의 모든 복제본에 의해 동일한 순서로 처리되도록 합의를 통과해야 한다. 그러나 서브넷의 복제 상태를 수정하지 않는 인그레스 메시지에 대해서는 중요한 최적화가 가능하다. 이러한 메시지는 *쿼리 콜(query calls)*이라고 불리며 — 다른 인그레스 메시지는 *업데이트 콜(update calls)*이라고 불린다. 쿼리 콜은 캐니스터의 상태를 읽고 가능하면 업데이트하는 계산을 수행할 수 있지만, 캐니스터 상태에 대한 어떤 업데이트도 복제 상태에 커밋되지 않는다. 따라서, 쿼리 콜은 합의를 통과하지 않고 단일 복제본에

의해 직접 처리될 수 있으며, 이는 쿼리 콜 응답을 얻는 지연을 크게 줄여준다.

쿼리 콜에 대한 응답은 *인그레스 이력* 데이터 구조에 기록되지 않음을 주목하라. 따라서, 이러한 쿼리 콜 응답을 인증하기 위해 라운드별 인증된 상태 메커니즘을 직접 사용할 수는 없다. 그러나 이러한 응답을 인증하기 위한 별도의 메커니즘이 제공된다: *인증 변수(certified variables)*. 라운드별 인증된 상태의 일부로서, 서버넷의 각 캐니스터에는 *그 캐니스터에 대한 인증 변수*로 불리는 소량의 바이트가 할당되며, 그 값은 업데이트 콜을 통해 갱신될 수 있고, 라운드별 인증된 상태 메커니즘을 사용하여 인증될 수 있다. 더 나아가, 캐니스터는 자신의 인증 변수를 사용하여 머클 트리의 루트를 저장할 수 있다. 이러한 방식으로, 캐니스터에 대한 쿼리 콜의 응답은, 그 응답이 해당 캐니스터의 인증 변수를 루트로 하는 머클 트리의 리프인 한, 인증될 수 있다.

6.3 외부 사용자 인증

인그레스 메시지와 서버넷 간 메시지 사이의 주요 차이점 중 하나는 이러한 메시지를 인증하는 데 사용되는 메커니즘이다. 우리는 이미 위에서(6.1절 참조) 서버넷 간 메시지를 인증하는 데 임계값 서명이 어떻게 사용되는지 보았다. NNS 레지스트리(1.5절 참조)는 서버넷 간 메시지 인증에 사용되는 임계값 서명의 공개 검증 키를 보유한다.

외부 사용자에게 대한 중앙 레지스트리는 없다. 대신, 외부 사용자는 *사용자 식별자(user identifier, 별칭 *principal*)* 를 사용하여 캐니스터에 자신을 식별하며, 이는 공개 서명-검증 키의 해시다. 사용자는 인그레스 메시지에 서명하는 데 사용되는 대응하는 비밀 서명 키를 보유한다. 이러한 서명과 해당 공개키는 인그레스 메시지와 함께 전송된다. IC는 서명을 자동으로 인증하고 사용자 식별자를 적절한 캐니스터로 전달한다. 그러면 캐니스터는 사용자 식별자와 인그레스 메시지에 지정된 연산의 다른 매개변수들을 바탕으로 요청된 연산을 승인할 수 있다.

첫 번째 사용자들은 IC와의 최초 상호작용 중에 키 쌍을 생성하고 공개 키로부터 사용자 식별자를 유도한다. 재방문 사용자들은 사용자 에이전트에 저장된 비밀 키를 사용하여 인증된다. 사용자는 서명 위임을 사용하여 여러 키 쌍을 단일 사용자 아이덴티티에 연계할 수 있다. 이는 한 사용자가 동일한 사용자 아이덴티티로 여러 기기에서 IC에 접근할 수 있게 하므로 유용하다.

7 실행 계층

실행 환경은 한 번에 하나의 입력을 처리한다. 이 입력은 입력 큐들 중 하나에서 가져오며 하나의 캐니스터를 대상으로 한다. 이 입력과 캐니스터의 상태에 기반하여, 실행 환경은 캐니스터의 상태를 업데이트하고, 추가로 출력 큐에 메시지를 추가하고 *인그레스 이력*을 업데이트할 수 있다(가능하다면 이전 인그레스 메시지에 대한 응답으로).

주어진 라운드에서, 실행 환경은 여러 입력을 처리할 것이다. *스케줄러*가 주어진 라운드에서 어떤 입력이 어떤 순서로 실행되는지를 결정한다. 스케줄러의 모든 세부사항에 들어가지는 않고, 몇 가지 목표를 강조한다:

- 결정적이어야 한다, 즉, 주어진 데이터에만 의존해야 한다;

- 캐니스터들 사이에 작업 부하를 *공정하게* 분배해야 한다(그러나 *지연(latency)* 보다 *처리량(throughput)* 최적화를 택한다);
- 각 라운드에서 수행되는 총 작업량(1.8절 참조의 *사이클*로 측정됨)은 미리 결정된 어떤 양에 가깝게 유지되어야 한다.

실행 환경(메시지 라우터와 함께)이 처리해야 하는 또 다른 과제는, 한 서브�트의 캐니스터가 다른 서브�트의 캐니스터가 소비할 수 있는 속도보다 더 빨리 서브�트 간 메시지를 생성하는 상황이다. 이를 위해, 생산 캐니스터를 *스로틀링* 하는 자체-조절 메커니즘이 구현되어 있다.

실행 환경이 처리하는 다른 많은 자원 관리 및 부기 작업들이 존재한다. 그러나 이러한 모든 작업들은 *결정적*으로 처리되어야 한다.

7.1 랜덤 테이프

각 서브�트는 *분산 의사난수 생성기(PRG)*에 접근할 수 있다. 3절에서 언급했듯이, 의사난수 비트들은 자체가 $(f + 1)$ -out-of- n BLS 서명인 시드로부터 파생되며, 이를 *랜덤 테이프(Random Tape)*라고 부른다. 합의 프로토콜의 각 라운드마다 서로 다른 랜덤 테이프가 존재한다. 이 BLS 서명은 합의에서 사용되는 랜덤 비콘(5.5절 참조)에 사용되는 것과 유사하지만, 메커니즘은 다소 다르다.

합의 프로토콜에서, 높이 h 의 블록이 최종화되자마자, 각 정직한 복제본은 높이 $h + 1$ 의 랜덤 테이프에 대한 자신의 조각을 공개(release)할 것이다. 이는 두 가지 함의를 갖는다:

1. 어떤 정직한 복제본에 의해 높이 h 의 블록이 최종화되기 전에, 높이 $h + 1$ 의 랜덤 테이프는 예측 불가능함이 보장된다.
2. 어떤 정직한 복제본에 의해 높이 $h + 1$ 의 블록이 최종화될 때쯤, 그 복제본은 일반적으로 높이 $h + 1$ 의 랜덤 테이프를 구성하는 데 필요한 모든 조각을 보유하게 될 것이다.

의사난수 비트를 얻기 위해, 서브�트는 이러한 비트에 대한 요청을 해야 한다. 이러한 의사난수-비트 요청은 어떤 라운드, 이를 h 라 하자, 실행 계층으로부터의 “시스템 콜”로 이루어질 것이다. 시스템은 이후, 높이 $h + 1$ 의 랜덤 테이프가 사용 가능할 때 그 요청에 응답할 것이다. 위의 성질 (1)에 의해, 요청이 이루어지는 시점에 요청된 의사난수 비트는 예측 불가능함이 보장된다. 위의 성질 (2)에 의해, 요청된 난수 비트는 일반적으로 다음 블록이 최종화되는 시점에 사용 가능할 것이다. 사실, 현재 구현에서는, 높이 h 의 블록이 최종화되는 시점에, 합의 계층(5절 참조)은 높이 h 의 블록(의 페이로드)과 높이 $h + 1$ 의 랜덤 테이프를 동시에 메시지 라우팅 계층으로 전달하여 처리하게 할 것이다.

8 체인 키 암호학 II: 체인-진화 기술

1.6.2절에서 언급했듯이, 체인 키 암호학에는 시간 경과에 따라 블록체인 기반 복제 상태 기계를 견고하고 안전하게 유지하기 위한 기술 모음이 포함되며, 이들이 함께 *체인-진화 기술* (chain-evolution technology) 이라 불린다. 각 서브넷은 여러 라운드(일반적으로 수백 라운드 규모)의 *에폭*(epochs) 으로 동작한다. 체인-진화 기술은 에폭에 연동된 주기로 주기적으로 실행되는 많은 필수 유지관리 활동(*가비지 컬렉션*, *패스트 포워딩*, *서브넷 멤버십 변경*, *비밀의 능동적 재공유*, 그리고 *프로토콜 업그레이드*)을 구현한다.

체인-진화 기술에는 두 가지 필수 요소가 있다: *요약 블록*(summary blocks) 과 *캐치업 패키지*(CUPs) 이다.

8.1 요약 블록

각 에폭의 첫 번째 블록은 *요약 블록* 이다. 요약 블록에는 다양한 임계값 서명 체계(3절 참조)의 공유를 관리하는 데 사용될 특수 데이터가 포함된다. 두 가지 임계값 체계가 있다:

- 하나는 에폭마다 새로운 서명 키가 생성되는 $(f + 1)$ -out-of- n 체계;
- 하나는 에폭마다 한 번씩 서명 키가 재공유(reshare)되는 $(n - f)$ -out-of- n 체계.

저임계값 체계는 *랜덤 비콘* 과 *랜덤 테이프* 에 사용되고, 고임계값 체계는 서브넷의 복제 상태를 인증하는 데 사용된다.

DKG 프로토콜(3.5절 참조)이 각 서명 키에 대해 거래 집합(dealings set)을 필요로 하며, 각 복제본이 이 거래 집합으로부터 자신의 서명 키 공유를 비대화식으로 획득할 수 있어야 함을 상기하라.

또한 NNS가 (그 밖의 것들과 함께) 서브넷의 멤버십을 결정하는 *레지스트리* 를 유지한다는 점(1.5절 참조)을 상기하라. 레지스트리(따라서 서브넷 멤버십)는 시간 경과에 따라 변경될 수 있다. 그러므로 서브넷들은 다양한 시점에 다양한 목적을 위해 어떤 *레지스트리 버전* 을 사용하는지에 합의해야 한다. 이 정보 또한 요약 블록에 저장된다.

에폭 i 의 요약 블록에는 다음 데이터 필드가 포함된다.

- currentRegistryVersion. 이 레지스트리 버전은 에폭 i 동안 사용될 *합의 위원회* (consensus committee) 를 결정할 것이다 — 합의 계층이 수행하는 모든 작업(블록 생성, 공증, 최종화)은 이 위원회에 의해 수행될 것이다.
- nextRegistryVersion. 각 합의 라운드에서, 블록 메이커는 자신이 알고 있는 최신 레지스트리 버전을 자신의 제안에 포함시킬 것이다(이는 제안된 블록이 확장하는 블록보다 과거일 수 없다). 이는 에폭 i 의 요약 블록에서 nextRegistryVersion 값이 상당히 최신 상태가 되도록 보장한다.

에폭 i 에서 currentRegistryVersion 의 값은 에폭 $i - 1$ 의 nextRegistryVersion 값으로 설정된다.

- currentDealingSets. 이것들은 에폭 i 에서 메시지에 서명하는 데 사용될 임계값 서

명 키를 결정하는 거래 집합들이다.

보게 되겠지만, 에폭 i 의 임계값 서명 위원회(threshold signing committee) (즉, 해당 임계값 서명 키 공유를 보유한 복제본들)는 에폭 $i - 1$ 의 합의 위원회이다.

- nextDealingSets. 이는 에폭 $i - 1$ 동안 수집(collected)된 거래들이 모여 저장되는 곳이다.⁶ 에폭 i 의 currentDealingSets 값은 에폭 $i - 1$ 의 nextDealingSets 값으로 설정될 것이며(그 자체가 에폭 $i - 2$ 에서 수집된 거래들로 구성된다).
- collectDealingParams. 이는 에폭 i 동안 수집될 거래 집합들을 정의하는 매개변수들을 기술한다. 에폭 i 동안, 블록 메이커들은 이러한 매개변수에 비추어 검증된 거래들을 자신들의 제안 블록에 포함시킬 것이다.

이러한 거래들의 수신 위원회(receiving committee)는 에폭 i 의 요약 블록의 nextRegistryVersion 값에 기반한다.

저임계값 체계의 경우, 거래 위원회(dealing committee)는 에폭 i 의 합의 위원회이다.

고임계값 체계의 경우, 재공유될 공유는 에폭 i 의 nextDealingSets 값에 기반한다. 그러므로 거래 위원회는 에폭 $i - 1$ 의 수신 위원회이며, 이는 또한 에폭 i 의 합의 위원회이다.

또한 에폭 i 의 임계값 서명 위원회는 에폭 $i - 2$ 의 수신 위원회이며, 이는 에폭 $i - 1$ 의 합의 위원회임을 관찰하라.

에폭 i 의 합의는 에폭 i 의 currentRegistryVersion과 currentDealingSets 값에 의존한다 — 특히, 합의 위원회 자체의 구성은 currentRegistryVersion에 기반하고, 합의에 사용되는 랜덤 비콘은 currentDealingSets에 기반한다. 더욱이, 다른 어떤 블록과 마찬가지로, 에폭 i 의 시작에서 검증된 요약 블록이 둘 이상 있을 수 있으며, 그 모호성은 에폭 i 의 합의에 의해 해소되어야 한다. 이 겹보기의 순환성은 에폭 i 가 시작되기 전에 에폭 $i - 1$ 의 시작에서의 요약 블록이 최종화되었다고 요구함으로써 해결된다. 이는 새로운 요약 블록의 관련 값들이 그 오래된 요약 블록에서 직접 복사되기 때문이다. 이는 실제로 *암묵적 동기성 가정*이지만, 꽤 학술적인 가정이다. 실제로, 활성성을 보장하기 위한 5.12.2절의 “합의 스로틀링” 때문이며, 또한 에폭의 길이가 상당히 길기 때문에, 이는 실제로 일어날 수 없다시피 하다: 에폭 $i - 1$ 의 요약 블록을 최종화하지 못한 채 에폭 $i - 1$ 의 끝에 도달하기 훨씬 이전에, 검증 지연 함수는 천문학적으로 커질 것이고, 따라서 최종화에 필요한 부분 동기성 가정은 (사실상) 확실하게(실무적으로는) 만족될 것이다.⁷

⁶ 우리가 생략한 한 가지 세부사항은, 에폭 $i - 1$ 에서 필요한 모든 거래들을 수집하는 데 실패하면, 폴백으로 에폭 i 의 nextDealingSets 값이 사실상 에폭 i 의 currentDealingSets 값으로 설정된다는 점이다. 이 경우, 프로토콜은 적절히 과거의 거래 위원회와 임계값 서명 위원회를 사용하게 될 것이다.

7 또한 에폭 i 에서 수집되는 거래들은 에폭 i 의 요약 블록의 데이터, 특히 *nextDealingSets* 와 *nextRegistryVersion* 의 값에 의존함을 주목하라. 따라서, 이러한 거래들은 에폭 i 의 요약 블록이 최종화되기 전에는 생성되어서도, 검증되어서도 안 된다.

8.2 CUPs

CUP을 설명하기 전에, 랜덤 비콘의 한 가지 세부사항을 먼저 지적한다: 각 라운드의 랜덤 비콘은 이전 라운드의 랜덤 비콘에 의존한다. 이는 필수적인 특징은 아니지만, CUP의 설계에 영향을 준다.

CUP은(블록체인 위가 아닌) 특수한 메시지로, 이전 에폭들에 대해 아무것도 알지 못해도 주어진 에폭에서 동작을 시작하는 데 복제본이 (대부분) 필요로 하는 모든 것을 담고 있다. 이는 다음 데이터 필드들로 구성된다:

- 전체/복제 상태(6.1절의 부분적, 라운드별 인증된 상태와는 대조됨)에 대한 머클 해시 트리의 루트.
- 해당 에폭의 요약 블록.
- 해당 에폭의 첫 라운드에 대한 랜덤 비콘.
- 위 필드들에 대해 서브넷의 $(n - f)$ -out-of- n 임계값 서명 키로 한 서명.

주어진 에폭에 대한 CUP을 생성하려면, 복제본은 해당 에폭의 요약 블록이 최종화되고 대응하는 라운드별 상태가 인증될 때까지 기다려야 한다. 이미 언급했듯이, 전체/복제 상태는 머클 트리로 해시되어야 한다 — 이 과정을 가속하기 위해 여러 기법이 사용되지만, 여전히 상당히 비용이 들며, 이것이 에폭당 한 번만 수행되는 이유다. CUP에는 이 머클 트리의 루트만 포함되므로, 복제본이 자신의 피어들로부터 필요한 임의의 상태를 가져올 수 있게 하는 특수한 *상태 동기화*(state sync) 하위 프로토콜이 사용된다 — 다시 말해, 이 과정을 가속하기 위해 여러 기법이 사용되지만, 여전히 상당히 비용이 든다. 우리가 CUP에 고임계값 서명을 사용하므로, 어떤 에폭에서도 유효한 CUP은 하나뿐임을 확신할 수 있으며, 더구나 상태를 끌어올 수 있는 피어들도 많을 것이다. 또한 임계값 서명 체계의 공개키는 시간 경과에 따라 변하지 않으므로, 서브넷의 현재 참여자를 알지 못하더라도 CUP은 검증될 수 있다.

8.3 체인-진화 기술의 구현

가비지 컬렉션: 주어진 에폭에 대한 CUP에 포함된 정보로 인해, 각 복제본은 그 에폭 이전에 처리된 모든 입력과, 그 입력들을 순서화하는 데 필요했던 모든 합의-수준 프로토콜 메시지들을 안전하게 제거(purge)할 수 있다.

Figure 5: 패스트 포워딩

패스트 포워딩: 서브넷의 한 복제본이 피어들에 비해 매우 뒤처지게 되는 경우(오랫동안 다운되었거나 네트워크에서 분리되었기 때문), 또는 새로운 복제본이 서브넷에 추가되는 경우, 해당 복제본은 그 시점까지의 모든 입력에 대해 합의 프로토콜을 실행하고 처리할 필요 없이, 가장 최근 에폭의 시작 지점으로 *패스트 포워딩* 될 수 있다. 이러한 복제본은 최신 CUP을 획득함으로써 이를 수행할 수 있다. CUP에 포함된 요약 블록과 랜덤 비콘, 그리고 다른 복제본들로부터의 프로토콜 메시지들(아직 제거되지 않은)을 함께 사용하여, 이 복제본은 해당 에폭의 시작부터 합의 프로토콜을 전진시킬 수 있다. 또한 이 복제본은 에폭의 시작에 대응하는 복제 상태를 획득하기 위해 상태 동기화 하위 프로토콜을 사용할 것이며, 이를 통해 합의에 의해 생성된 입력들도 처리할 수 있다.

그림 5는 패스트 포워딩을 설명한다. 여기서, 따라잡아야 하는 복제본이 어느 에폭의 시작 지점(예컨대 높이 101)에서의 CUP을 가지고 있다고 가정한다. CUP에는 높이 101에서의 복제 상태에 대한 머클 트리 루트, 높이 101의 요약 블록(녹색으로 표시됨), 그리고 높이 101의 랜덤 비콘이 포함되어 있다. 이 복제본은 CUP의 머클 트리 루트를 사용해 이 상태를 검증하면서, 상태 동기화 하위 프로토콜을 사용해 피어들로부터 높이 101의 전체 복제 상태를 획득할 것이다. 이 상태를 획득한 후, 복제본은 프로토콜에 참여할 수 있으며, 피어들로부터 높이 102, 103 등에서의 블록(및 합의와 관련된 다른 메시지들)을 획득하고, 자신의 복제 상태 사본을 업데이트할 것이다. 피어들이 이미 더 높은 높이의 블록들을 최종화한 경우, 이 복제본은 피어들로부터 그것들(및 그 공증과 최종화)을 획득하는 속도(그리고 실행 계층이 허용하는 속도)만큼 빨리 그 최종화된 블록들을 처리할 것이다.

서브넷 멤버십 변경: 특정 에폭에서 어떤 레지스트리 버전이 효력을 가지는지, 그리고 그것이 서브넷 멤버십을(더 구체적으로는 다양한 작업을 위한 위원회 멤버십을) 어떻게 결정하는지 인코딩하는 데 요약 블록이 어떻게 사용되는지는 이미 논의했다. 복제본이 서브넷에서 제거된 이후에도, 가능하다면 추가로 한 에폭 동안 자신에게 할당된 위원회 임무에 참여해야 한다는 점을 주목하라.

비밀의 능동적 재공유: 요약 블록이 서명 키를 생성하고 재공유하는 데 어떻게 사용되는지는 이미 논의했다. 필요하다면, 필요한 요약 블록은 CUP으로부터 획득될 수 있다.

프로토콜 업그레이드: CUP은 프로토콜 업그레이드를 구현하는 데도 사용된다. 프로토콜 업그레이드는 NNS에 의해 시작된다(1.5절 참조). 모든 세부사항에 들어가지는 않고 기본 아이디어만 말하면, 다음과 같다:

- 새로운 버전의 프로토콜을 설치할 시점이 되면, 에폭의 시작에서의 요약 블록이 이를 지시할 것이다;
- 구버전의 프로토콜을 실행 중인 복제본들은 요약 블록을 최종화하고 해당 CUP을 생성할 수 있을 만큼 충분히 합의를 계속 실행할 것이다; 그러나 그들은 빈 블록만 생성하고 어떤 페이로드도 메시지 라우팅과 실행으로 전달하지 않을 것이다;
- 새로운 버전의 프로토콜이 설치되고, 새 버전을 실행 중인 복제본들이 위의 CUP으로부터 전체 프로토콜 실행을 재개할 것이다.

참고문헌

- [BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
- [BKM18] E. Buchman, J. Kwon, and Z. Milosevic. The latest gossip on BFT consensus, 2018. arXiv:1807.04938, <http://arxiv.org/abs/1807.04938>.
- [BLS01] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9–13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.
- [But13] V. Buterin. Ethereum whitepaper, 2013. <https://ethereum.org/en/whitepaper/>.
- [CDH+21] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams. Internet Computer Consensus. *Cryptology ePrint Archive, Report 2021/632*, 2021. <https://ia.cr/2021/632>.
- [CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Advances in Cryptology – CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21–25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
- [CL99] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In M. I. Seltzer and P. J. Leach, editors, *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, USA, February 22–25, 1999, pages 173–186. USENIX Association, 1999.
- [CWA+09] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In J. Rexford and E. G. Sirer, editors, *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2009, April 22–24, 2009, Boston, MA, USA*, pages 153–168. USENIX Association, 2009.

http://www.usenix.org/events/nsdi09/tech/full_papers/clement/clement.pdf.

[Des87] Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO ’87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16–20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 1987.

[DLS88] C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Consensus in the Presence of Partial Synchrony. *J. ACM*, 35(2):288–323, 1988.

[Fis83] M. J. Fischer. The Consensus Problem in Unreliable Distributed Systems (A Brief Survey). In *Fundamentals of Computation Theory, Proceedings of the 1983 International FCT-Conference, Borgholm, Sweden, August 21–27, 1983*, volume 158 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 1983.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – CRYPTO ’86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GHM+17] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *Cryptology ePrint Archive*, Report 2017/454, 2017. <https://eprint.iacr.org/2017/454>.

[Gro21] J. Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, Report 2021/339, 2021. <https://ia.cr/2021/339>.

[JMV01] D. Johnson, A. Menezes, and S. A. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.

[Mer87] R. C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology – CRYPTO ’87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16–20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[MXC+16] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The Honey Badger of BFT Protocols. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24–28, 2016*, pages 31–42. ACM, 2016.

- [Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. <https://bitcoin.org/bitcoin.pdf>.
- [PS18] R. Pass and E. Shi. Thunderella: Blockchains with Optimistic Instant Confirmation. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018 – 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 – May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.
- [PSS17] R. Pass, L. Seeman, and A. Shelat. Analysis of the Blockchain Protocol in Asynchronous Networks. In *Advances in Cryptology – EUROCRYPT 2017 – 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 – May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673. Springer, 2017.
- [Sch90] F. B. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.
- [YMR+18] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham. HotStuff: BFT Consensus in the Lens of Blockchain, 2018. arXiv:1803.05069, <http://arxiv.org/abs/1803.05069>.