

에스프레소(Espresso)

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.

에스프레소 시퀀서: HotShot 합의와 Tiramisu 데이터 가용성

에스프레소 시스템(Espresso Systems)

2023년 9월 7일

초록

레이어 2(L2) 롤업은 레이어 1(L1) 블록체인의 확장성을 높이기 위한 대표적인 접근 방식이다. 롤업은 트랜잭션 처리를 오프체인으로 이전하고, L1은 롤업 상태만을 체크포인팅한다. 이러한 설계는 어떤 트랜잭션이 롤업에 포함되는지, 그리고 어떤 순서로 포함되는지에 대한 문제를 남긴다. 안타깝게도 현재의 모든 롤업은 트랜잭션 순서를 정하기 위해 자체적인 중앙화된 시퀀서(sequencer)를 사용한다. 이는 두 가지 한계를 초래한다. 첫째, 중앙화된 시퀀서는 단일 장애 지점(single point of failure)이 된다. 둘째, 서로 다른 L2 생태계에 속한 애플리케이션 간 상호운용성이 저해된다.

본 논문에서는 모든 L2 롤업이 공동으로 사용할 수 있는 탈중앙화 네트워크인 에스프레소 시퀀서(Espresso Sequencer)를 소개한다. 우리의 설계는 시퀀서의 두 가지 핵심 과제인 트랜잭션 순서 결정과 데이터 가용성 보장을 처리하기 위해 모듈적으로 분리된 두 가지 핵심 구성 요소, 즉 HotShot 합의(HotShot Consensus)와 Tiramisu 데이터 가용성(Tiramisu data availability)으로 구성된다. HotShot은 지분증명(proof-of-stake) 환경에서 동작하는 낙관적 반응성(optimistically responsive)과 통신 효율성을 갖춘 합의 프로토콜로, 매수 공격(bribing adversaries)에 저항성을 가지며 대규모 노드 수로 확장 가능하다. 계층화된 Tiramisu 데이터 가용성 프로토콜은 검증 가능한 정보 분산(verifiable information dispersal)과 소규모 무작위 위원회(small random committees)를 결합하여 선형 통신 복잡도를 유지하면서 데이터 가용성을 보장한다. 두 프로토콜 모두 네트워킹 계층에서 콘

텐츠 전송 네트워크(content distribution network, CDN)를 사용할 수 있도록 설계되어, 낙관적 경우에는 Web2 수준의 성능을 제공하면서도 비관적 경우에는 강력한 Web3 보안 보장을 유지한다.

목차

- 1. 서론
 - 1.1 HotShot: 고처리량·저지연의 확장 가능한 합의
 - 1.2 Tiramisu: 3계층 데이터 가용성 솔루션
 - 1.3 아키텍처 개요
 - 1.4 주요 기여
 - 1.4.1 HotShot 합의
 - 1.4.2 Tiramisu 데이터 가용성(DA)
 - 1.5 관련 연구
- 2. HotShot 합의
 - 2.1 표기법
 - 2.2 위협 모델
 - 2.3 HotStuff 개요
 - 2.4 HotShot 합의를 향하여
 - 2.4.1 스테이킹 키 처리
 - 2.4.2 에폭 간 재구성
 - 2.4.3 뷰 동기화
 - 2.4.4 탈중앙화 랜덤 비콘
 - 2.4.5 구성 요소 통합
 - 2.5 보안 개요
- 3. Tiramisu 데이터 가용성
 - 3.1 개요
 - 3.1.1 통신 복잡도
 - 3.2 구문
 - 3.3 구성
 - 3.3.1 커밋
 - 3.3.2 분산
 - 3.3.3 복구
 - 3.4 HotShot에서의 Tiramisu 활용
 - 3.4.1 활성 보장
 - 3.4.2 고비용 데이터 복구 강제
 - 3.4.3 최적화: Savoiardi 인증서 불필요
 - 3.4.4 페이로드 전파·복구·보존 정책
- 4. CDN을 활용한 반응성 극대화
 - 4.1 CDN + P2P: 하이브리드 블록체인 네트워킹
 - 4.2 에스프레소 시퀀서와 CDN 통합
- 5. 예비 평가

A. Savoirdi 검증 가능한 정보 분산

- A.1 커밋
- A.2 분산
- A.3 복구
- A.4 저장 쿼럼 크기
- A.5 벡터 커밋먼트의 필요성에 대하여
- A.6 점근적 복잡도
- A.7 최소 종료 보장
- A.8 강한 가용성 보장
- A.9 관련 연구

B. 구현 예시

- B.1 랜덤 비콘
- B.2 무작위 위원회 샘플링
- B.3 쿼럼 인증서
 - B.3.1 배경
 - B.3.2 정의
 - B.3.3 구성

1 서론

이더리움(Ethereum)은 범용 스마트 컨트랙트를 지원하는 대표적인 블록체인 시스템으로, 탈중앙화 금융(DeFi) [130, 36, 118], 대체 불가능 토큰(NFT) [117], 탈중앙화 게임 [29] 등 다양한 혁신적 애플리케이션을 가능하게 한다. 그러나 이 시스템은 우수한 탈중앙성과 경제적 보안을 제공하는 반면, 초당 약 25건의 트랜잭션 처리량과 약 13분에 달하는 지연 시간이라는 낮은 성능 한계를 가진다.

보다 최근에는 이더리움 위에서 트랜잭션 처리 성능을 개선하기 위해 롤업(rollup)과 같은 *레이어 2(layer-2, L2)* 솔루션 [66, 119, 109]이 도입되었다. 롤업의 핵심 아이디어는 트랜잭션 처리를 오프체인으로 이전하고, 이더리움 상에는 상태 체크포인트 컨트랙트만을 유지하는 것이다. 이를 위해 *L2 시퀀서(L2 sequencer)*가 사용자 트랜잭션을 수집하고 순서를 정한다. 순서가 정해진 트랜잭션은 롤업에서 실행되며, 갱신된 상태는 레이어 1(L1) 상의 롤업 컨트랙트를 통해 저장된다. 상태 전이의 정확성은 zkRollup [119]에서는 유효성 증명(Validity proof)을 통해, 옵티미스틱 롤업(optimistic rollup) [66]에서는 사기 증명(fraud proof)을 통해 보장된다. 이러한 접근 방식은 배치된 모든 트랜잭션에 대해 실행 및 저장에 필요한 가스 비용을 분산시키므로, 훨씬 높은 처리량과 더 낮은 가스 수수료를 가능하게 한다. 더 나아가 사용자가 L2 운영자를 신뢰하는 경우, 롤업 컨트랙트의 상태 변경을 확인하기 전에 낙관적으로 트랜잭션을 확정할 수 있어 지연 시간 또한 개선된다.

공유 탈중앙화 시퀀서의 필요성. 그러나 현재 존재하는 모든 L2 롤업은 L2 트랜잭션을 수집하고 순서를 정하기 위해 각자 자체적인 *중앙화된(centralized)* 시퀀서를 사용하고 있다. 이는 이더리움이 처음에 제공하고자 했던 주요 이점을 훼손하는 두 가지 근본적인 한계를 초래한다. 첫째, 중앙화된 시퀀서는 단일 장애 지점(single point of failure)이다. 시퀀서가 중단되면 전체 롤업이 작동을 멈추게 된다. 이는 이론적인 문제가 아니라 실제로 발생한 사례도 존재한다 [133]. 설령 시스템이 중단되지 않더라도, 시퀀서가 트랜잭션 순서를 임의로 변경하거나(equivocate), 사용자 트랜잭션을 검열하거나 차단하거나, 혹은 수수료를 독점적 수준으로 인상하지 않을 것이라는 막대한 신뢰를 요구하게 된다. 그 결과 L2 사용자는 더 이상 이더리움에서 누리던 수준의 보안을 공유하지 못하게 된다. 둘째, 이더리움의 가장 매력적인 특징 중 하나는 여러 애플리케이션 간의 조합성(composability)과 상호운용성(interoperability)이다. 그러나 L2 시스템마다 서로 다른 시퀀서를 사용하는 구조는 이러한 L2 생태계 간 상호작용을 훨씬 어렵게 만든다. 공유 시퀀서를 사용할 경우, 여러 롤업에 속한 트랜잭션 집합을 원자적으로 실행하는 것이 가능해지며, 이를 통해 위험 없이 크로스 롤업 차익거래 기회를 활용할 수 있다. 실제로 사용자가 요청한 원자성을 존중하지 않는 블록 빌더는 공개적으로 책임을 묻고 슬래싱할 수 있다. 이는 롤업마다 시퀀서가 서로 다른 경우에는 명백히 불가능하다. 공유 시퀀서는 또한 수수료를 대가로 계산된 위험을 감수하려는 참여자를 도입함으로써 플래시론(flash loan)과 같은 크로스 롤업 애플리케이션이라는 새로운 설계 공간을 열어준다. 이러한 설정에서는 추가 참여자가 빌더의 오작동에 따른 모든 결과를 부담하지만, 롤업의 보안 자체는 영향을 받지 않는다. 이는 다음과 같은 자연스러운 질문으로 이어진다.

모든 L2 롤업이 공유할 수 있는 탈중앙화되면서도 고성능의 시퀀서를 구축할 수 있는가?

본 논문에서는 에스프레소 시퀀서 네트워크(Espresso sequencer network)를 통해 이 질문에 대해 긍정적으로 답한다. 우리의 시퀀서가 갖는 속성을 이해하기 위해서는 먼저 *탈중앙화되면서도 고성능*이라는 개념이 무엇을 의미하는지 이해할 필요가 있다. 본질적으로 *탈중앙화*란 현실적이며 강력한 적대자를 허용하면서도 시퀀서 노드 집합 간 합의가 필요함을 의미하고, *고성능*이란 높은 처리량을 낮은 커밋 지연 시간으로 처리할 수 있음을 의미한다.

시퀀서는 레이어 1(L1)에 비해 설계 제약이 더 적다. 레이어 1과 같은 상태 머신 복제(state machine replication, SMR) 시스템의 설계 제약은 다음의 세 가지 측면을 포함한다. 이들 각각은 SMR의 잠재적인 병목 지점이 된다.

(데이터 브로드캐스트) 노드는 트랜잭션, 명령 등과 같은 데이터를 모든 다른 노드에게 브로드캐스트해야 한다.

(합의 및 순서 결정) 노드는 트랜잭션의 순서에 대해 합의에 도달해야 한다.

(실행) 노드는 각자의 상태 머신 상에서 이러한 트랜잭션을 실행해야 한다.

반면 시퀀서는 트랜잭션을 실행하지 않는다. 대신 실행은 일반적으로 롤업 자체가 생성하는 영지식 증명(ZK proof) 또는 사기 증명(fraud proof)을 통해 보장된다. 그 결과 시퀀서는 트랜잭션을 검증(validate)할 필요도 없으며, 따라서 트랜잭션 데이터를 모든 노드에 브로드캐스트할 필요도 없다. 대신 시퀀서는 트랜잭션 데이터의 *가용성(availability)*만을 보장하면 충분한데, 이는 해당 데이터가 이를 필요로 하는 누구에 의해서든 향후에 복구될 수 있음을 의미한다. 따라서 시퀀서는 완화된 형태의 데이터 가용성 문제와 합의 문제만을 해결하면 된다. 에스프레소 시퀀서는 HotShot 합의(HotShot Consensus)와 Tiramisu 데이터 가

용성(Tiramisu Data Availability) 프로토콜을 통해 이 두 가지 문제를 모듈적으로 해결한다.

1.1 HotShot: 고처리량·저지연의 확장 가능한 합의

분산 컴퓨팅 분야에서는 서로 다른 네트워크 조건과 다양한 유형의 적대자 하에서 안전성을 보장하고, 동적 가용성(dynamic availability), 책임성(accountability), 반응성(responsiveness) 등과 같은 여러 바람직한 속성 중 일부를 달성하는 다수의 합의 프로토콜이 제안되어 왔다. 이러한 접근법들과 비교할 때, HotShot을 설계함에 있어 핵심 설계 원칙은 강력한 적대자에 대해 보안을 확보하는 동시에 낙관적 경우에서 높은 효율성을 추구하는 것이다. 이는 공격과 네트워크 장애가 일반적으로 일시적인 사건에 그치며, 장기간 지속되는 공격의 경우 적대자가 얻는 이익이 그 비용을 정당화하지 못하도록 인센티브를 설정할 수 있다는 이론적·실무적 관찰에 근거한다. 보다 구체적으로 보안 관점에서의 핵심 목표는 네트워크가 이더리움 네트워크 자체와 동등한 수준의 보안을 갖추는 것이다. 이는 다수의 노드(또는 이더리움 스테이커)가 프로토콜을 실행하는 높은 수준의 탈중앙화를 달성해야 함을 의미한다. 또한 프로토콜은 스테이커를 대상으로 한 매수 공격(bribery attacks) [19, 84, 116]이 존재하더라도 안전성과 활성(liveness)을 유지해야 한다. 반면 성능 측면에서의 목표는 매우 높은 처리량, 매우 낮은 지연 시간, 그리고 이상적으로는 반응성(responsiveness)을 갖는 프로토콜을 구현하는 것이다. 처리량은 시퀀싱될 수 있는 전체 트랜잭션 수에 해당하고, 지연 시간은 트랜잭션이 네트워크에 제출된 시점부터 시퀀싱될 때까지의 지연을 의미하며, 반응성 속성은 고정된 블록 시간 없이 네트워크가 허용하는 한 최대 속도로 동작할 수 있는 시퀀서를 의미한다. 다시 말해, 우리의 성능 목표는 이더리움이 제공하는 성능을 보완하는 동시에 중앙화 시퀀서에 근접한 성능을 달성하려는 것이다.

보다 구체적으로, 우리의 합의 프로토콜은 허가형 비잔틴 장애 허용(permissioned Byzantine Fault Tolerant, BFT) 프로토콜을 지분증명(proof-of-stake) 환경에 맞게 적용한다. 이를 통해 스테이크를 예치하거나 해제함으로써 자유롭게 프로토콜에 참여하거나 이탈할 수 있는 동적 네트워크 참여자를 지원할 수 있다. 이 프로토콜은 전체 스테이크의 3분의 2 초과가 정직한 노드에 의해 제어되는 한 안전성과 활성을 만족해야 한다. 앞서 언급했듯이 완전한 탈중앙화를 달성하기 위해서는 수만 개에 달하는 스테이킹된 합의 노드를 지원할 수 있어야 한다. 전통적인 접근법은 일반적으로 허가형 BFT 프로토콜과 지분증명을 위원회 샘플링(committee sampling)을 통해 결합하는데, 이 방식에서는 소규모 무작위 위원회가 전체 스테이킹 사용자 집합을 대표하여 합의를 실행한다. 그러나 이러한 방식은 적응적 공격(adaptive attacks)에 취약한 경우가 많으며, 소량의 스테이크만을 보유한 적대자라 하더라도 선출된 위원회를 손상시켜 합의의 보안을 깨뜨릴 수 있다. 적응적 적대자에 대해 안전한 해결책으로는 Algorand [59], YOSO [58] 등이 존재한다. 가장 널리 알려진 방어책은 위원회 구성원이 투표를 공개할 때까지 선출된 위원회를 은닉하는 것으로, 이를 통해 적응적 적대자가 사후적으로 위원회의 행위를 변경하지 못하도록 한다. 그러나 이러한 방식에서도 매수 공격은 여전히 현실적인 우려 사항으로 남는다. 즉, 누구를 손상시켜야 하는지 알지 못하더라도 악의적인 적대자가 특정한 검증 가능한 악의적 행위에 대해 보상을 약속할 수 있기 때문이다 [12]. 예를 들어, 공격자는 선출된 위원회 구성원에게 특정 트랜잭션을 검열하도록 보상을 지급하는 스마트 계약을 생성할 수 있다. 따라서 효율성과 확장성을 확보하면서도 매수 공격에 대한 저항성을 유지하기 위해, 우리는 모든 스테이킹된 노드가 합의 프로토콜에 참여하도록 하고, 기본 BFT 프로토콜로 HotStuff-2 [82]를 사용한다.

HotStuff-2는 노드들이 충분한 시간 동안 동일한 뷰(view)에 진입할 수 있도록 하는 페이스메이커(pacemaker) 모듈이 주어질 경우 선형 통신 복잡도를 달성한다. 그러나 기존 HotStuff [125]에 포함된 일부 페이스메이커 구현은 이차 통신 복잡도와 비최적의 지연 시간을 초래한다. 이에 우리는 Naor와

Keidar [90]의 뷰 동기화(view sync) 프로토콜을 활용하여 낙관적 경우에서 선형 통신 복잡도를 달성한다. 이를 통해 합의 프로토콜은 수천 개의 물리적 노드로 확장 가능해지며, 이는 매수 공격의 비용을 매우 높게 만든다. 동시에 본 프로토콜은 낙관적 반응성(*optimistic responsiveness*)이라는 중요한 속성을 달성하는데, 이는 네트워크 조건이 양호한 경우 중앙화 시스템에 근접한 성능을 발휘할 수 있음을 의미한다. 더 나아가 우리는 집계된 쿼럼 인증서(aggregated quorum certificates), 스테이크 테이블(stake table), 탈중앙화 랜덤 비콘(decentralized random beacons)과 같은 효율적인 기본 구성 요소를 설계하여, 허가형 HotStuff-2 프로토콜을 성능 저하 없이 완전한 탈중앙화 환경에 적응시킨다.

1.2 Tiramisu: 3계층 데이터 가용성 솔루션

상태 머신 복제(state machine replication, SMR)는 크기가 $|B|$ 인 데이터 페이로드 B 가 네트워크 상의 모든 n 개 노드에게 브로드캐스트되어야 함을 요구한다. 이 문제에 대해 $O(|B|n)$ 의 통신량은 자명한 하한(lower bound)이며, 합의 프로토콜을 사용하여 B 의 내용에 대해 합의함으로써 $O(|B|n^2)$ 의 통신량으로 쉽게 해결할 수 있다.

그러나 앞서 설명한 에스프레소 시퀀서 네트워크에서와 같이, 우리의 목표는 데이터를 브로드캐스트하는 것이 아니라 데이터의 가용성(*availability*)만을 보장하는 것이다. 이러한 SMR의 완화된 형태는 통신 복잡도를 $O(|B|)$ 까지 크게 개선함으로써 달성될 수 있다.(^1)

우리의 데이터 가용성(data availability, DA) 솔루션인 Tiramisu는 효율성을 희생하지 않으면서 DA 보안을 보장하기 위해 세 가지 새로운 아이디어를 결합한다. 이러한 아이디어는 시대를 초월한 이탈리아 디저트의 층을 연상시키듯, 계층(layer)으로 볼 수 있다.

가장 아래에는 Savoiardi 계층이 있으며, 이는 [10]에서 제안된 검증 가능한 정보 분산(verifiable information dispersal, VID) 기법의 새롭게 도입된 변형을 사용하여 데이터 가용성을 보장한다. 기본 아이디어는 데이터 블록을 소거 부호화(erasure-coded)된 조각으로 인코딩하고, 각 노드마다 하나의 조각을 전송하는 것이다. 유효한 조각을 수신한 노드는 확인 서명(ack signature)을 반환하며, 합의 리더는 쿼럼에 해당하는 노드들로부터 확인 서명을 수신하는 한 다음 단계로 진행할 수 있다. Savoiardi VID 계층은 통신 효율성이 매우 높고 안전하다. 이를 손상시키기 위해서는 매수 공격을 수행하는 적대자라 하더라도 시스템 전체 스테이크의 3분의 1을 초과하는 비율을 통제해야 하는데, 이는 HotShot 합의 프로토콜을 깨뜨리는 것만큼이나 어렵다.

Savoiardi의 단점은 성능에 있다. 전체 페이로드를 복구하고자 하는 클라이언트는 다수의 저장 노드로부터 조각을 다운로드해야 하며, 페이로드를 디코딩하기 위한 계산 자원을 소모해야 한다. 이를 보완하기 위해 우리는 Mascarpone 계층을 도입한다. 이 계층은 데이터의 빠른 재구성과 부분 데이터에 대한 신속한 접근을 가능하게 한다. 우리는 HotShot 랜덤 비콘(random beacon)을 활용하여 에폭(epoch)마다 소규모 무작위 위원회를 선출한다. 이 위원회에 속한 각 노드는 데이터 조각이 아니라 전체 데이터 블록을 수신한다. 유효한 DA 인증서, 그리고 따라서 유효한 HotShot 블록에는 위원회의 일정 임계치, 예를 들어 80%의 서명이 포함되어야 한다. 이는 높은 확률로 하나 이상의 정직한 노드가 쿼럼에 포함되어, 데이터에 대한 빠른 접근을 제공하고 재구성을 지원할 수 있음을 보장한다. 무작위 선출은 위원회를 작게 유지할 수 있게 하지만, 동시에 매수 공격에 더 취약하게 만든다는 단점도 있다. Savoiardi 계층은 이러한 취약성이 없으며 가장 강력한 보안을 제공한다. Savoiardi와 Mascarpone을 결합함으로써 강력한 보안과 빠른 재구성을 동시에 달성

할 수 있다.

마지막으로, 우리는 *Cocoa 계층*이라 부르는 콘텐츠 전송 네트워크(content delivery network, CDN)를 추가함으로써 시스템의 성능을 낙관적으로 더욱 향상시킬 수 있다. CDN은 데이터를 캐시하고 효율적으로 분산시킬 수 있으며, 효율적인 브로드캐스트 계층으로 볼 수 있다. 이는 전통적인 Web2 인프라 수준의 성능을 제공할 수 있으나, 중앙화된 구성 요소로서 어떠한 보안 보장도 제공하지 않는다. 그러나 CDN은 완전히 선택적인 구성 요소이며 Mascarpone 및 Savoiardi 계층에 의해 백업되므로, 이는 보안 문제를 야기하지 않는다. CDN이 온라인 상태일 경우, 데이터의 신속한 전파와 빠른 접근을 제공함으로써 해당 계층들 및 전체 시스템의 성능을 크게 향상시킬 수 있다. Cocoa 계층은 모든 구성 요소가 정상적으로 작동하고 온라인 상태일 때 네트워크가 더 빠르게 동작하면서도, 적대자가 존재하는 상황에서는 여전히 높은 수준의 보안 보장을 제공하는 우리의 *낙관적 반응성(optimistic responsiveness)* 설계 원칙과 잘 부합한다.

1.3 아키텍처 개요

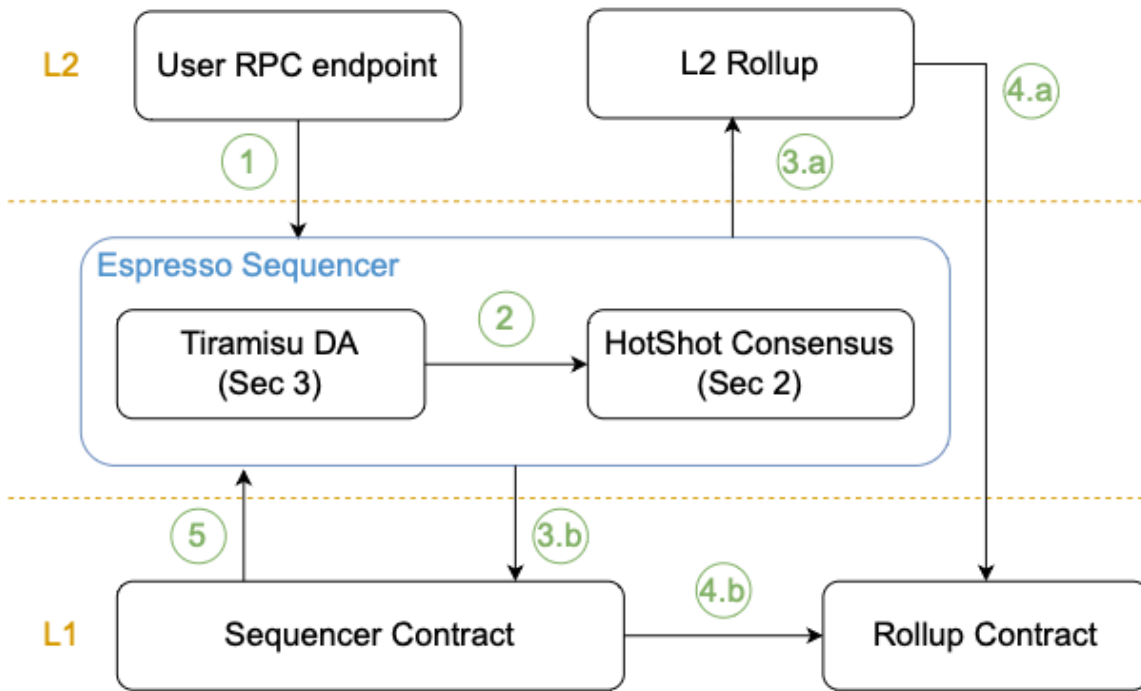
본 절에서는 에스프레소 시퀀서(Espresso Sequencer)의 주요 구성 요소와, 사용자, L2 롤업 및 L1 블록체인(예: 이더리움)과의 상호작용을 간략히 설명한다.

에스프레소 시퀀서는 수천 개의 이질적인 노드로 구성된 탈중앙화 네트워크로, *로그 복제(log replication)*, 즉 트랜잭션 시퀀싱을 서비스로 제공한다. 내부적으로는 다음의 두 가지 핵심 구성 요소로 이루어져 있다.

1. 제출된 트랜잭션에 대해 순차적인 순서를 제공하는 *HotShot 합의 계층(HotShot Consensus layer)*이다(실행은 수행하지 않음). 이는 허가 없는(permissionless) 리더 기반 멀티샷(multi-shot) BFT 프로토콜로, HotStuff-2 [82]를 지분증명(proof-of-stake) 환경으로 확장한 것이다. 자세한 내용은 [섹션 2](#)를 참조한다.
2. 클라이언트가 제출한 데이터의 가용성을 보장하는 *Tiramisu 데이터 가용성(DA) 계층*이다. 보다 구체적으로, 저장 노드(storage nodes) 집합이 원시 트랜잭션 데이터의 가용성과 복구 가능성을 담담한다. 시퀀서는 실행을 담당하지 않으므로, 데이터가 모든 풀 노드(full node)에 브로드캐스트될 필요는 없으며, 이는 더 높은 처리량을 달성하는 데 필수적이다. 자세한 내용은 [섹션 3](#)를 참조한다.

그림 1에 나타난 바와 같이, 에스프레소 시퀀서는 사용자, L2 롤업, 그리고 L1 블록체인과 적극적으로 상호 작용한다.

- **레이어 2(L2) 롤업.** 이는 오프체인 실행 엔진(VM)으로, 사용자의 트랜잭션을 수용한 뒤 시퀀서에 의해 순서가 정해지고 최종화된 이후 이를 결정론적으로 처리한다.



에스프레소 시퀀서와 연동되는 롤업의 실행 로직은 애플리케이션 특화된 형태부터 EVM 롤업과 같은 완전한 스마트 컨트랙트 플랫폼까지 다양할 수 있다. 또한 하위 구성 요소로서의 증명자 네트워크(prover network)는 주기적으로 L1 상의 롤업 컨트랙트에 상태 커밋먼트(state commitment)를 갱신하며, 이때 유효성 증명(Validity proof) (zkRollup의 경우) 또는 잠재적인 사기 증명(fraud proof) (옵티미스틱 롤업의 경우)을 함께 제출한다.

- 레이어 1(L1) 블록체인.** 에스프레소 시퀀서는 이 블록체인에 원장 상태를 체크포인트팅한다. L1의 주요 기능은 HotShot 합의 계층을 위한 항상 온라인(always-online) 상태의 최소 신뢰(minimally trusted) 검증 라이트 클라이언트로 동작하는 것이다. L1이 더 성숙해져 채택이 확대되고 포크 또는 재조직(reorg)에 대한 경제적 비용이 높아질수록, 이러한 체크포인트는 HotShot 합의에 대한 장거리 공격(long-range attacks)에 대해 방어 심층(defense in depth)을 제공한다. 내부적으로는 HotShot 시퀀서가 생성한 최종화된 원장을 기록하는 하나의 시퀀서 컨트랙트(sequencer contract)가 존재하며, 각 L2 롤업마다 하나씩 롤업 컨트랙트(rollup contract)가 존재하여 시퀀서 컨트랙트로부터 원장 상태를 읽고 롤업별 상태를 유지한다.

L2 롤업이 제출한 트랜잭션의 생명 주기는 다음과 같이 설명할 수 있다(그림 1 참조).

- 사용자는 RPC 엔드포인트/게이트웨이를 통해 원시 트랜잭션을 Tiramisu DA 멤폴로 제출한다. RPC 엔드포인트/게이트웨이는 누구나 어디서든 실행할 수 있다. 각 트랜잭션에는 또한 어느 L2 롤업에 속하는지를 나타내는 네임스페이스 ID가 함께 첨부된다.
- DA 계층의 리더는 서로 다른 롤업에 속할 수 있는 트랜잭션 데이터를 수집하고, 데이터 가용성 인증서(DA certificate)를 얻기 위해 해당 데이터를 Tiramisu DA 계층으로 분산(disperse)한다. 이때 DA 인증서의 형성은 반드시 모든 저장 노드가 전체 데이터를 획득할 것을 요구하지는 않는다. 동시에 HotShot 합의 계층의 리더는 트랜잭션 데이터에 대한 짧은 벡터 커밋먼트(short vector commitment)를 포함하는 제안을 브로드캐스트한다. HotShot 합의에 참여하는 노드들은 이에 대해 투표하고 블록 커밋먼트를 최종화한다. HotShot 합의에 참여하는 노드들은 DA 계층의 저장 노드이기도 하며, 블록 커밋먼트의 최종화는 해당 트랜잭션 데이터의 가용성을 함께 의미한다는 점에 유의

한다.

3. HotShot 합의 프로토콜에서 블록이 최종화된 것으로 간주되면, 에스프레소 시퀀서의 노드들은 새로운 블록 커밋먼트(및 쿼럼 인증서)를 L1 상의 시퀀서 컨트랙트로 전파한다([그림 1](#)의 단계 3.b). 이후 시퀀서 컨트랙트는 인증서를 검증하고 체크포인트된 원장 상태를 갱신한다. 시퀀서 컨트랙트는 현재의 에스프레소 구성과 스테이커에 대한 정보를 유지하고 있으므로, 커밋된 새로운 상태를 검증할 수 있다. 동시에 DA 노드들은 다중 롤업 블록 내용을 L2 롤업 증명자에게 전파하여, 각 롤업이 자신에게 해당하는 순서화된 트랜잭션 집합을 추출할 수 있도록 한다(단계 3.a). 우리는 지연 시간을 낮추고 처리량을 높이기 위해 데이터 전파를 합의의 임계 경로(critical path) 밖으로 이동시키는 설계 결정을 강조한다. 블록이 이미 최종화된 이후에도 전체 블록의 클라이언트(예: 롤업 증명자)에 대한 전파가 반드시 완료될 필요는 없으며, 임계 경로에서는 데이터 가용성을 보장하기 위한 최소한의 대역폭만 사용된다.
4. 증명자는 새로운 블록을 수신한 후, 갱신된 상태와 그 커밋먼트를 계산하고 상태 전이 증명(state-transition proof)을 생성하여 L1 상의 롤업 컨트랙트에 검증 가능한 상태 체크포인트로 게시한다([그림 1](#)의 단계 4.a). L2 상태 전이 증명을 검증하기 위해, 롤업 컨트랙트는 시퀀서 컨트랙트로부터 갱신된 에스프레소 원장 상태를 함께 읽는다(단계 4.b).

마지막으로, 에스프레소 네트워크 노드들은 L1 시퀀서 컨트랙트로부터 갱신된 스테이크 테이블(stake table)을 주기적으로 읽어 스테이킹 참여자 집합에 대한 전역적 관점을 획득한다(단계 5). 이는 일반적으로 스테이킹 참여자 집합이 재구성되는 에폭 전환 시점에 발생한다. 이러한 체크포인트는 HotShot 합의에 대한 장거리 공격에 대해 방어 심층을 제공한다.

1.4 주요 기여

본 절에서는 탈중앙화 공유 시퀀서를 구축하는 과정에서의 보안 및 확장성 문제를 해결하기 위해 HotShot 합의(HotShot Consensus)와 Tiramisu 데이터 가용성(DA)이 제공하는 핵심 기여를 제시한다.

1.4.1 HotShot 합의

HotShot의 노드들은 원시 트랜잭션 데이터에 대한 짧은 *벡터 커밋먼트(short vector commitment)*에 합의하는 프로토콜을 실행함으로써, 제출된 트랜잭션의 순차적 순서를 최종화한다. 탈중앙화 시퀀서의 맥락에서, 우리는 확장 가능한 합의를 가능하게 하는 다음과 같은 측면을 강조한다.

동적 참여. HotStuff [125], HotStuff-2 [82] 및 기타 전통적인 BFT 프로토콜은 합의 참여자 집합이 사전에 고정된 *허가형(permissioned)* 환경에서만 동작하는 것과 달리, 우리는 지분증명(proof-of-stake, PoS) 환경으로 이를 확장하여 참여자가 스테이킹 권한의 이동을 통해 합의 프로토콜에 동적으로 참여하거나 이탈할 수 있도록 한다. 이를 위해 HotStuff-2를 PoS 환경에 맞게 조정하였으며, 조정된 프로토콜은 손상된 스테이킹 키 뒤에 있는 적대적 가중치의 누적합이 전체 스테이킹 가중치의 3분의 1 미만인 한 안전성과 활성을 만족한다. 특히 도전적인 문제는 스테이킹 노드 집합이 변경되는 에폭(epoch) 전환을 처리하는 것이다. 예를 들어 극단적인 경우, 연속된 두 에폭 사이에서 스테이킹된 노드 집합이 완전히 달라질 수 있으며, 이 경우 HotStuff/HotStuff-2의 락킹(locking) 메커니즘은 더 이상 작동하지 않는다. 우리는 각 에폭의 끝에 상수 개수의 라운드를 추가하여 한 스테이킹 집합에서 다른 집합으로 뷰(view) 정보를 안전하게 전달할 수 있도록 HotStuff-2 프로토콜을 확장함으로써 이 문제를 해결한다. 자세한 내용은 [섹션 2.4.2](#)를 참조한다.

매수 공격 저항성. 허가형 프로토콜을 지분증명 환경으로 적응시키는 일반적인 접근 방식은 전체 스테이커 집합을 대표하는 소규모 무작위 위원회를 선출하고, 위원회 구성원들 사이에서 합의를 실행하는 것이다. 그러나 적응적 적대자는 선출 결과를 알게 된 이후 위원회를 손상시켜 합의를 깨뜨릴 수 있다. Algorand [59]는 주기적으로 비밀 선출(secret election) 프로토콜을 활용하여, 선출된 위원회 구성원이 자신이 위원회에 속해 있음을 증명하는 증거를 브로드캐스트할 때까지 적대자가 위원회 구성원을 알 수 없도록 함으로써 이 문제를 해결한다. 슬래싱(slash) 메커니즘을 통해 선출된 구성원은 자신의 당첨 티켓과 합의 메시지를 브로드캐스트하는 단 한 번의 발언만 허용되므로, 선출 결과를 안 이후에 적대자가 위원회를 손상시켜 상충된 메시지를 보내도록 만들기는 어렵다. 그러나 [12]에서 지적하듯이, Algorand의 접근 방식은 매수 공격을 수행하는 적대자가 특정한 검증 가능한 악의적 행위에 대해 보상을 보장하는 광고를 할 수 있는 상황에서는 충분하지 않다. 이 경우 적대자는 매수된 노드의 내부 상태나 정체를 알 필요조차 없다. 또한 [12, 22]는 이러한 매수 공격의 존재가 다수의 최장 체인 기반 지분증명(longest-chain PoS) 및 BFT 기반 지분증명 합의 프로토콜에서 안전성과 활성 위반으로 이어질 수 있음을 지적한다.

이에 따라 우리는 섹션 2.2에서 매수 공격을 포착하기 위한 새로운 위협 모델을 도입한다. 구체적으로, 우리는 네트워크에 공개된 메시지뿐만 아니라 정직한 노드가 전송을 준비 중인 “버퍼링된(buffered)” 메시지까지 관찰할 수 있는 가상 적대자를 고려한다. 이 적대자는 네트워크에 새로운 메시지가 게시되는 것을 본 이후 언제든지 정직한 노드를 적응적으로 손상시킬 수 있으며, 더 나아가 버퍼링된 메시지 목록을 확인한 뒤 정직한 노드를 적응적으로 손상시켜 전송 예정이던 버퍼링된 메시지를 수정하거나 폐기할 수 있다. *설정되지/알려지지 않은(unset) 버퍼링된 메시지*로부터 정직한 노드를 적응적으로 손상시킬 수 있는 이러한 능력이 매수 공격을 시뮬레이션하는 핵심이다. 전통적인 적응적 손상(adaptive corruption)과 달리, 매수 공격은 일반적으로 제한된 예산을 가진 적대자에게 지속적이며 회복 불가능한 비용을 발생시키므로, 우리는 매수 횟수에 대해 유한하지만 알려지지 않은 상한이 존재한다고 추가로 가정한다.

낙관적 반응성(Optimistic responsiveness). 새로운 위협 모델은 수만 개의 노드로 확장 가능한 고성능 합의 프로토콜을 달성하는 데 중대한 도전을 제기한다. 특히 매수 공격이 존재하는 상황에서는 소규모 위원회를 활용하는 합의 방식은 배제된다. 한편 시스템은 매수 공격이 발생하더라도 활성 및 안전성에 대한 부정적 영향을 최소화해야 한다. 즉, (i) 합의는 매수 공격의 존재 여부와 무관하게 항상 안전성을 만족해야 하며, (ii) 네트워크 상태가 정상으로 복구되고 매수 공격자가 예산을 소진하여 공격을 중단한 이후에는 프로토콜이 다시 진행을 이어갈 수 있어야 한다. 이는 애초에 적대자가 매수 공격을 개시하려는 유인을 약화시킨다. 따라서 대부분의 시간 동안 시스템은 매수 시도가 없는 양호한 시나리오에 놓여 있다고 안전하게 가정할 수 있다.

이러한 양호한 시나리오에서 뛰어난 성능을 갖는 확장 가능한 합의 프로토콜을 구축하기 위해, 우리는 HotStuff-2를 지분증명(PoS) 환경에 맞게 조정하고 다음과 같은 최적화를 도입한다.

- **전체 데이터가 아닌 짧은 커밋먼트에 대한 합의.** 우리는 처리량과 지연 시간을 개선하기 위해 데이터 가용성과 합의를 모듈적으로 분리하는 구조를 고안한다. 합의 계층은 데이터 블록에 대한 짧은 커밋먼트에 대해서만 합의하며, 전체 데이터의 가용성은 병렬로 동작하는 DA 계층에 위임된다. 다음 절에서 설명하는 Tiramisu DA는 모든 노드에 데이터를 전파하지 않고도 가용성을 달성할 수 있을 만큼 매우 효율적이므로, 합의 프로토콜의 처리량과 지연 시간은 크게 개선될 수 있다.
- **집계 서명과 비트 벡터를 이용한 퀴럼 인증서(QC) 구현.** 완전히 허가된 환경과 달리, 지분증명 환경에서는 투표자 집합이 시간에 따라 동적으로 변하기 때문에 간결하고 효율적으로 검증 가능한 퀴럼 인증서를 구축하는 것이 쉽지 않다. 이를 해결하기 위해 동적 스테이크 테이블은 신원(identity)을 서명

검증 키와 스테이크 양에 매핑하며, 우리는 효율적인 조화갱신 및 공개 키 샘플링을 위해 이를 영속적 머클 트리(persistent Merkle tree)로 구현한다. 또한 스테이크 키 등록 시에는 로그 키 공격(rogue-key attack)을 방지하기 위해 소유 증명(proof-of-possession)이 요구된다(L1 상의 계정 주소가 등록 키인 경우에는 이 단계는 제거할 수 있다). 이러한 스테이크 키 테이블을 바탕으로, 부록 B에서 우리는 쿼럼 인증서의 효율적인 구현을 제시한다. 개념적으로 쿼럼 인증서는 집계 서명과 투표자 집합을 나타내는 짧은 비트 벡터로 구성된다. 노드들은 스테이크 키 테이블과 비트 벡터로부터 집계된 서명 검증 키를 얻은 후 서명을 검증함으로써 QC를 확인한다. 점근적으로 QC의 크기는 여전히 스테이킹 노드 수에 선형이지만, 그 선형 항은 노드당 1비트에 불과하다. 따라서 QC는 실제로도 매우 효율적이며(예: 노드 10,000개 기준 약 1.25KB), 또한 슬래싱에 필수적인 책임성(accountability)을 만족한다.

- **네트워크 계층에서의 CDN 활용.** 낮은 지연 시간을 달성하기 위해서는 P2P 네트워크에만 의존하는 것보다 더 빠른 통신이 필요하다. 그러나 수만 개의 합의 노드가 존재하는 경우, 이들 모두와 직접 연결을 유지하는 것은 모든 합의 노드에 대해 비효율적이다. 우리는 Web2 아키텍처에서 사용되는 것과 유사하게 네트워크 계층에서 CDN을 활용한다. CDN은 고대역폭 연결을 갖춘 강력한 하드웨어로, 낮은 지연 시간에서 효율적인 정보 라우팅을 가능하게 한다. 이러한 인프라는 계산과 통신 양 측면 모두에서 많은 수의 노드를 훨씬 더 잘 확장할 수 있도록 한다. 통신 측면에서는 데이터가 모든 합의 노드로 더 빠르게 전파될 수 있으며, 계산 측면에서는 서명 집계와 같은 계산 집약적인 작업을 이러한 하드웨어로 오프로딩할 수 있다. 더 나아가, 우리의 프로토콜이 갖는 낙관적 반응성 속성은 네트워킹 계층에서의 이러한 수정과 결합되어 더욱 우수한 성능을 달성한다.
- **효율적인 뷰 동기화.** HotStuff-2에서 활성은 참여자들이 충분한 시간 동안 동일한 뷰(view)에 머무는 때만 보장된다. HotStuff는 이를 위해 페이스메이커(pacemaker) 모듈을 사용하지만, 이는 이차 통신 복잡도와 비최적의 지연 시간을 초래한다. 이 문제를 해결하기 위해 우리는 Naor와 Keidar [90]의 뷰 동기화기를 사용하여 낙관적 경우에서 선형 통신 복잡도와 상수 지연 시간을 달성한다.
- **효율적인 탈중앙화 랜덤 비콘.** 에스프레소 시퀀서는 합의/뷰 동기화 리더를 선출하고 소규모 DA 위원회를 샘플링하기 위해 주기적으로 시드를 생성할 수 있는 신뢰할 만한 무작위성 소스를 필요로 한다. 부록 B.1에서는 집계 서명과 지연 함수(delay function) [15]를 활용하여 탈중앙화 방식으로 매우 효율적인 랜덤 비콘을 구현하며, 이는 독립적인 연구 주제로서도 의미를 가질 수 있다.

1.4.2 Tiramisu 데이터 가용성(DA)

Tiramisu DA 계층은 합의에 도달하기 이전에 데이터 가용성을 보장하는 데 핵심적인 역할을 수행한다. DA 리더는 블록의 원시 데이터를 수집한 뒤, 이를 DA 네트워크 전반에 분산시키고 가용성에 대한 인증서를 획득한다. 바람직한 처리량과 낮은 지연 시간을 달성하기 위해서는 전체 통신량과 왕복(roundtrip) 횟수를 모두 최소화하는 것이 필수적이다. DA 계층은 다음의 세 가지 주요 구성 요소로 이루어진다.

- **Savoiardi: 매수 공격에 강인한 데이터 가용성.** 리더가 입력 데이터를 소거 부호화(erasure-coded)된 조각으로 인코딩한 뒤 *분산(dispersal)* 프로토콜을 통해 모든 저장 노드에 조각을 배포하는 검증 가능한 정보 분산(verifiable information dispersal, VID) 프로토콜이다. 각 저장 노드는 전체 블록 데이터가 아니라 크기가 $O(|B|/n)$ 인 단일 조각만을 저장하며, 해당 조각이 전체 데이터와 일관됨을 조각 수신자가 효율적으로 검증할 수 있음을 강조한다. 분산 프로토콜 이후, 리더는 임계치에 해당하는 저장 노드로부터 유효한 VID 조각이 가용함을 보장하는 인증서를 획득한다. 이후 이 인증서의 사본을 보유한 누구나 VID 복구(retrieve) 프로토콜을 실행함으로써 원래의 전체 데이터 블록을 복구할 수 있다. 우리의 VID 프로토콜은 [10]의 AVID-1과 DispersedLedger [123]를 결합한 것으로, 독립적인 연구 주제로서도 의미를 가질 수 있다. 분산 프로토콜은 매우 효율적이며, 리

더가 각 저장 노드에 VID 조각을 전송하고 저장 노드가 일관성 검증 후 서명을 반환하는 단 한 번의 통신 라운드만을 요구한다. 또한 네트워크 전체 통신량은 원래 블록의 원시 데이터 크기에 작은 상수 계수를 곱한 수준에 불과하며, 각 노드에 전송되는 소량의 페이로드 커밋먼트 데이터만이 추가된다. 다만 VID 복구 프로토콜은 복구자가 다수의 노드로부터 인코딩된 페이로드 조각을 가져와 이를 디코딩해야 하므로, 단일 소스로부터 전체 원시 페이로드를 다운로드하는 것보다 느리다. 다행히도 VID 복구 프로토콜은 다음에 설명할 CDN 계층과 무작위 DA 위원회 계층이 모두 사용 불가능한 최후의 수단으로만 실행된다. 이는 시스템의 보안과 성능에 치명적인 영향을 주지 않으면서도 높은 회복 불가능한 매수 비용(예: 매수된 사용자에게 지급된 금전적 보상)을 적대자에게 부과한다. 따라서 위원회 매수 공격을 시도할 유인은 거의 없으며, 대부분의 경우 데이터는 빠르게 복구될 수 있을 것으로 기대한다.

- **Mascarpone: 빠른 합의를 위한 소규모 DA 위원회.** 주기적으로 갱신되는 소규모 무작위 DA 위원회 노드 집합이다. DA 리더는 위원회에 속한 모든 노드에게 전체 데이터 블록을 전송하고, 위원회 노드 중 임계치에 해당하는 수가 이를 확인하고 서명을 반환하면 인증서를 획득한다. 정적 적대자 가정 하에서, 충분한 수의 위원회 노드는 높은 확률로 정직하며 데이터 가용성을 보장한다. 그러나 소규모 위원회는 매수 공격에 취약하다. 예를 들어, 적대자는 위원회가 데이터에 서명은 하지만 저장은 하지 않도록 매수할 수 있다. 이러한 상황에서는 Savoiardi 계층을 사용하여 데이터 가용성을 보장한다.
- **Cocoa: CDN을 통한 Web2 수준의 성능.** 정보 라우팅을 효율적으로 지원하는 상호 연결된 강력한 서버 집합으로 구성된 CDN 계층이다. CDN이 완전히 동작하고 네트워크가 공격을 받지 않는 낙관적 조건에서는, CDN은 중앙화된 Web2 네트워크에 필적하는 성능을 제공한다. CDN은 강력하지만 본질적으로 중앙화되어 있으므로 블록체인 프로토콜의 유일한 인프라로 간주될 수는 없다. 따라서 본 프로토콜은 CDN 계층 없이도 정상적으로 동작하며, CDN은 성능을 향상시키기 위한 선택적 구성 요소이다.

요약하면, 시퀀서 노드가 상태 머신을 실행하지 않는다는 점을 활용하여, 우리는 매우 효율적인 DA 계층을 구축함으로써 처리량과 지연 시간을 개선한다. 이는 모든 노드가 전체 트랜잭션 데이터를 저장할 필요가 없기 때문에 가능하다. 특히 우리의 DA 프로토콜은 전체 네트워크 통신량을 원래 블록 데이터 크기에 작은 상수를 곱한 수준으로 최소화하며, 낙관적 조건에서는 중앙화 네트워크에 필적하는 성능을 제공한다. 또한 DA 계층은 매수 공격에 대해서도 안전성을 유지한다.

HotShot 합의와 Tiramisu DA 간의 연결. 활성(liveness)을 보장하기 위해, HotShot 합의에 참여하는 노드들은 벡터 커밋먼트 제안에 대해 투표하기 전에 반드시 데이터 가용성을 확인해야 한다. 그렇지 않으면 새로운 리더가 기존 블록을 기반으로 새로운 트랜잭션 블록을 구성할 수 있는 방법이 불분명해진다. 구체적으로, 노드는 (i) 제안이 무작위 DA 위원회의 임계치에 해당하는 참여자가 전체 데이터를 수신했음을 증명하는 인증서를 포함하고 있으며, (ii) 해당 노드 자신이 해당 커밋먼트에 대한 VID 조각을 수신한 경우에만 투표한다. 이는 트랜잭션 순서가 최종화된 이후에도 데이터가 항상 가용함을 보장한다. 매수 공격을 수행하는 적대자가 무작위 DA 위원회 전체를 손상시켜 데이터는 저장하지 않으면서 서명만 하도록 만들 수 있다는 점에 유의해야 하며, 이 경우 프로토콜은 데이터를 복구하기 위해 VID 디코딩 알고리즘을 실행해야 한다. 그러나 이러한 상황에서는 적대자가 활성 저해를 위해 합의 리더를 손상시키는 편이 오히려 더 유리할 수 있다. 따라서 정직한 리더와 매수되지 않은 DA 위원회가 존재하는 낙관적 조건에서는, VID 복구 프로토콜이 실행되지 않으며 데이터는 CDN 노드나 소규모 DA 위원회로부터 신속하게 복구될 것으로 기대한다.

1.5 관련 연구

상태 머신 복제(state machine replication, SMR) 문제 [78, 79, 104]는 일부 머신이 적대적이거나 *비잔틴*(Byzantine)일 수 있는 상황에서도, 다수의 결정론적 분산 머신이 공통의 공유 상태에 합의하는 방법을 연구한다. 지난 40여 년 동안 SMR을 위한 비잔틴 장애 허용(Byzantine Fault Tolerant, BFT) 프로토콜 설계에 있어 많은 진전이 이루어져 왔다 [31, 80, 87, 86, 59, 40]. 이 분야의 모든 연구를 검토하는 것은 본 논문의 범위를 벗어나므로, 여기서는 HotShot과 밀접하게 관련된 측면만을 설명한다.

고처리량과 저지연 달성. 지난 20년간의 여러 연구는 대량의 트랜잭션을 낮은 지연 시간으로 처리할 수 있는 프로토콜 설계에 집중해 왔다. 여기에는 나카모토 스타일 프로토콜 [88, 27, 51, 99, 100, 38, 69, 122, 126, 72]과, 동기(synchrony) [34, 3, 4, 45, 59, 111, 101, 106, 2], 부분 동기(partial synchrony) [31, 24, 125, 33, 32, 57, 112], 비동기(asynchrony) [5, 61, 81, 108, 57] 등 다양한 네트워크 조건 하에서의 고전적 BFT 프로토콜이 포함된다. 처리량을 개선하기 위해, 최근의 연구 흐름에서는 트랜잭션에 대한 합의와 데이터 전파를 분리하려는 시도가 이루어지고 있다 [37, 41, 40, 108, 107, 100, 122]. 유사하게 지연 시간 측면에서는 낙관적 반응성(optimistic responsiveness) 개념이 도입되었으며, 다수의 프로토콜에서 이 속성이 성립함이 보였다 [99, 101, 106, 125, 33, 31]. 요컨대, 반응형 프로토콜은 네트워크 지연의 비관적 상한 Δ 와 무관하게 네트워크 속도에 맞추어 트랜잭션을 최종화한다.

이들 프로토콜 다수와 마찬가지로, HotShot 역시 높은 처리량과 낮은 지연 시간을 목표로 한다. 그러나 HotShot은 시퀀서로 사용된다는 점에서 고유한 제약을 가진다. 첫째, 처리량을 높이기 위해 데이터 전파와 합의를 분리한다. 또한 시퀀서 노드는 상태 머신의 실행을 담당하지 않으므로, HotShot은 데이터의 *가용성*만을 보장하면 되고 노드들이 *모든* 트랜잭션에 접근할 필요는 없다. 이로 인해 데이터 가용성은 전통적인 SMR 시스템에 비해 병목이 덜하다. 더 나아가, 우리의 목표는 HotShot이 매우 많은 수의 노드로 확장 가능하도록 하는 것이다.^(A2) 따라서 합의 과정에서 전송되는 데이터의 양은 시스템 성능과 직결되는 중요한 지표가 된다. 이에 우리는 HotShot을 HotStuff-2 프로토콜 [125, 82]에 기반하여 설계하였으며, 그 결과 낙관적 경우 선형, 최악의 경우 이차 통신 복잡도, 낮은 최종화 지연 시간, 그리고 낙관적 반응성과 같은 유리한 속성을 계승한다. 또한 실제 결함 수가 $f > n/3$ 인 상황에서 안전성 위반이 발생할 경우, 오작동 참여자를 탐지하고 슬래싱할 수 있도록 HotStuff-2의 책임성(accountability)을 갖춘 변형을 구현한다.

데이터 가용성. 개념적으로 데이터 가용성 프로토콜은 네트워크 노드들 사이에 공유된 데이터가 이후에도 신뢰할 수 있게 복구 가능함을 보장해야 한다. 소거 코드(eraser codes)는 이러한 목표를 달성하기 위한 핵심 도구이며, 특히 리드-솔로몬 코드(Reed Solomon codes) [121]는 (i) *사기 증명(fraud proof)* [127, 6, 7]을 통해 확률적으로 데이터가 올바르게 공유되었음을 보장하거나, (ii) 충분한 수의 참여자가 자신의 데이터 조각이 유효함을 확인할 경우 데이터 복구를 보장하는 *검증 가능한 정보 분산(Verifiable Information Dispersal, VID)* 프로토콜 [10, 93, 123, 9, 105, 92, 28, 124, 64]의 구성 요소로 사용된다. 일반적으로 이러한 연구들은 두 단계로 진행되는데, 첫 단계에서는 서로 다른 소거 코드 조각이 각 참여자에게 분배되어 $O(|B|)$ 비트의 통신 복잡도로 데이터 가용성을 보장하고, 두 번째 단계에서는 각 참여자가 자신의 데이터를 모든 참여자와 공유하여 $O(|B|n)$ 통신 복잡도로 모든 참여자가 데이터에 접근 가능하도록 한다. 우리는 데이터의 *가용성*만을 보장하면 되므로, $O(|B|)$ 통신만을 요구하는 첫 번째 단계만을 수행한다. 데이터에 접근이 필요한 어떤 참여자(예: 롤업)라도 다른 노드로부터 수집한 조각을 사용해 항상 데이터를 재구성할 수 있다. 이는 이더리움의 Danksharding 제안 [96]과도 유사하다.

소거 코드를 활용한 접근은 낮은 통신 복잡도를 달성하지만, 데이터 접근을 위해서는 여전히 다수의 노드와 통신해야 한다. 이에 따라 우리는 데이터 전체를 저장하는 두 개의 추가 데이터 가용성 계층, 즉 소규모 데이터 가용성 위원회와 콘텐츠 전송 네트워크(CDN)를 도입한다. 이러한 추가 계층은 보안을 위해 의존되지는 않는데, 이는 소규모 위원회는 매수될 수 있고 CDN은 장애를 일으킬 수 있기 때문이다. 그러나 낙관적 조건 하에서는 이들 계층이 반응적으로 쉽고 효율적인 데이터 접근을 제공한다.

매수 공격 저항성. 기존 연구들은 적대자를 정적(static), 적응적(adaptive), 또는 이동형(mobile)으로 분류해 왔다. 정적 손상은 개방형 블록체인 환경에서는 약한 가정이며, 강력한 이동형 적대자 하에서의 실용적 해법은 아직 알려져 있지 않다 [98, 128, 20, 54]. 이에 따라 많은 연구가 적응적 적대자 하에서의 보안을 목표로 하지만, 그 대가로 높은 통신 복잡도를 수반하는 경우가 많다 [21, 30, 70, 129, 3, 45, 1, 68]. Algorand [59]는 소규모 위원회를 사용함으로써 적응적 적대자에 대한 보안과 낮은 통신 복잡도를 동시에 달성한다. 그러나 [12]에서 언급되었듯이, 적대자는 매수 공격을 통해 무작위로 참여자를 손상시킬 수 있으며, 이로 인해 소규모 위원회 기반 해법은 안전하지 않다. 이에 반해 우리의 해법은 적응적이며 매수 공격을 수행하는 적대자 하에서도 보안을 유지하면서 바람직한 성능 지표를 달성한다.

CDN의 활용. 우리는 콘텐츠 전송 네트워크(CDN)와 P2P 네트워크를 결합한 하이브리드 네트워크를 활용한다. 낙관적 시나리오에서는 모든 메시지와 데이터가 CDN을 통해 교환되어, 처리량과 지연 시간 측면에서 거의 최적에 가까운 성능을 보장한다. CDN과 P2P 네트워크를 결합하여 콘텐츠 분산 성능을 개선하려는 아이디어는 기존 연구 [131, 65]에서도 탐구된 바 있으나, 그 목적은 중앙화 시스템의 성능 향상이었다. 반면 우리의 경우 목표는 대부분의 시간 동안 네트워크 성능을 극대화하면서도, CDN이 중단되거나 손상된 상황에서도 활성 및 검열 저항성을 유지하는 데 있다.

롤업(Rollups). 몇 년 전 개념화되기 시작한 이후 [66, 120], L2 롤업은 이제 현실이 되었으며 이더리움 생태계에서 빠르게 확산되고 있다. 사용자들은 가스 비용을 크게 절감하면서 결제를 수행하거나 선호하는 디앱(dApp)과 상호작용할 수 있는 다양한 대안을 선택할 수 있게 되었다 [75, 94, 109, 114, 97]. 가장 인기 있는 롤업 프로젝트들 사이에서는 탈중앙화 문제가 점점 더 많은 주목을 받고 있는데, 이는 특히 일부 실무적 이슈들 [133] 때문이다. 예를 들어 Aztec은 네트워크 탈중앙화 과정에 커뮤니티가 참여하도록 초대하고 있으며 [74], 다른 롤업 프로젝트들 또한 로드맵에 탈중앙화 관련 항목을 포함시키고 있다 [76, 113]. 이와 병행하여, 커뮤니티에서는 보다 일반화된 접근 방식에 대한 논의와 연구가 진행되고 있는데, 본 연구 역시 그 한 예이다. 즉, 각 롤업을 개별적으로 탈중앙화하는 대신, 모든 롤업의 트랜잭션을 신뢰할 수 있게 순서화하고 저장하는 *공유되고 탈중앙화된(shared and decentralized)* 시퀀서에 의존하자는 아이디어이다. 모든 롤업에 대해 단일 시퀀서를 사용하는 것의 수많은 이점과 함께, 그에 수반되는 미묘한 트레이드오프(예: 인프라 재사용, 크로스체인 MEV 포착 및 재분배, 상호운용성)는 활발히 논의되고 있으며 [110, 35], 유망한 연구 분야를 형성하고 있다. 실무적인 측면에서도, 여러 구체적인 이니셔티브 [110, 77]가 가까운 미래에 공유 시퀀서를 롤업과 최종 사용자에게 제공하는 것을 목표로 하고 있다. 공유 시퀀서를 설계할 때 핵심적인 고려 사항인 MEV [39] 처리와 관련하여, SUAVE [55]를 개발하는 팀은 MEV 포착과 분배를 보다 공정하게 만들기 위해 블록 빌딩을 탈중앙화하는 네트워크를 구축하고 있다.

2 HotShot 합의

본 절에서는 HotShot 합의 계층을 명세한다. 먼저 [섹션 2.2](#)에서 네트워크 및 적대자 모델을 정의한 후, [섹션 2.3](#)과 [섹션 2.4](#)에서 HotStuff 위에 우리의 합의 프로토콜이 어떻게 구축되는지를 설명한다. 프로토콜 실행에 대한 구체적인 내용은 [섹션 2.4.5](#)에서 명시한다. 마지막으로, 본 절에서 설명된 기본 구성 요소 (primitives)의 구체적인 구현(instantiation)은 [부록 B](#)에 위임한다.

2.1 표기법

HotShot 프로토콜은 *에폭(epoch)*의 연속으로 진행된다.

- $e \in \mathbb{N}$: 에폭.
- Γ : 각 에폭마다 커밋되는 블록의 개수.
- N_e (**굵게 표시**): 어떤 에폭 $e \in \mathbb{N}$ 에서 시스템 전체의 스테이크 총량.
- $t = \frac{1}{3}$ (**굵게 표시**): 비잔틴 스테이크의 최대 비율.
- n : 합의 노드의 수.
- f : 비잔틴 합의 노드의 수.

2.2 위협 모델

적대자의 유형. 우리는 다음과 같은 유형의 적대자를 구분한다. *정적(static)* 적대자는 프로토콜 시작 시점에 손상시킬 노드 집합을 선택해야 하는 반면, *적응적(adaptive)* 적대자는 프로토콜 실행 도중에 새로운 노드를 장악할 수 있다. 적대자가 손상된 노드의 메모리에 접근할 수는 있으나 해당 노드의 행위를 변경하지 않는 경우 이를 *수동적(passive)* 적대자라 하고, 반대로 손상된 노드가 임의로 행동하도록 만들 수 있는 경우를 *능동적(active)* 적대자라 한다. *매수(bribing)* 적대자 [12]는 경제적 인센티브를 활용하여 프로토콜에서 핵심적인 책임을 지는 노드(예: 다음 라운드의 리더)를 수동적인 방식으로 식별할 수 있다. *이동형(mobile)* 적대자 [128]는 프로토콜 실행 중에 노드를 손상시키거나 손상 상태를 해제할 수 있는 능력을 가진다. *합리적(rational)* 적대자 [60, 52]는 자신의 이익 극대화만을 목표로 하는 반면, 본 연구에서 고려하는 *비합리적(non-rational)* 적대자는 안전성이나 활성(liveness)과 같은 프로토콜의 핵심 속성을 깨뜨리기 위해 경제적 자원을 기꺼이 투입할 의지가 있다.

동기: 매수 공격 저항성. 허가형 프로토콜을 무허가(permissionless) 환경에 적응시키기 위한 일반적인 방법은 소규모 위원회를 사용하여 허가형 프로토콜의 단계를 실행하는 것이다. 각 위원회 구성원이 선택될 확률은 그가 보유한 자원(예: 지분증명 블록체인에서의 스테이크)에 비례한다. 그러나 이러한 접근을 따르는 PoS 프로토콜은 *완화된 적응적 적대자(mildly adaptive adversary)* [99]만을 견딜 수 있는데, 이는 표적 공격을 수행할 수는 있으나 그 지연이 회전 주기보다 길다는 가정에 의존한다. 즉각적인 손상이 가능한 완전 적응적 적대자는 선출 결과가 공개되는 즉시 모든 위원회 노드를 손상시켜 이러한 프로토콜을 깨뜨릴 수 있다. 완전 적응적 손상에 대응하기 위해 Algorand [59]는 비잔틴 합의의 각 하위 단계마다 비밀 선출(secret election)을 활용하는 *플레이어 대체 가능성(player replaceability)* 개념을 도입하였다. 이 방식에서는 복권 티켓이 로컬에서 계산되므로, 위원회 구성원은 해당 단계에서 당첨된 구성원이 공개적으로 검증 가능한 당첨 티켓을 공개하기 전까지는 적대자에게 예측 불가능하고 알려지지 않는다. 더 중요한 점은, 선출된 구성원이 자신의 당첨 티켓과 합의 관련 메시지를 브로드캐스트하는 단 한 번의 발언만을 수행한다는 것이다. 다음 하위 단계는 또 다른 무작위 위원회에 의해 수행되므로, 표적 공격을 수행할 수 있는 시간 창이 존

재하지 않는다.

그러나 Bagaria 등 [12]이 지적하였듯이, 특정한 검증 가능한 악의적 행위에 대해 보상을 보장하는 캠페인을 수행할 수 있는 매수 적대자가 존재하는 경우에는 플레이어 대체 가능성만으로는 충분하지 않다. 예를 들어, 공격자는 특정 유형의 트랜잭션을 검열하도록 선출된 위원회 구성원에게 보상을 지급하는 스마트 컨트랙트를 생성할 수 있다. 적응적 적대자와 비교했을 때 매수 적대자의 주요 특징은 다음과 같다.

- **적응적이면서 수동적인 손상:** 적대자는 공격을 개시하기 위해 표적의 신원을 알 필요가 없다. 또한 손상은 수동적인데, 이는 매수된 노드가 뇌물을 수령하기 위해 자발적으로 제어권을 넘기거나 지시된 대로 악의적으로 행동하기 때문이다. 수동적 손상의 중요한 특징은 공격이 성공하기 위해 프로토콜의 예측 가능성을 필요로 하지 않는다는 점이다. 그 결과, 사전적 적응 공격에 대한 대응책으로서의 플레이어 대체 가능성은 효용이 떨어진다.
- **지속적 매수에 대한 비용 증가:** 참여자를 매수하기 위한 예산은 유한하며, 일단 소진되면 향후에는 다시 접근할 수 없다.

이러한 매수 공격의 존재는 다수의 PoS 합의 프로토콜에서 안전성과 활성 위반으로 이어질 수 있음이 지적되어 왔다 [12, 22]. 한편 탈중앙화 거버넌스 시스템에서는 실제로 투표 매수 및 매수 플랫폼이 존재한다.^(A3) 일반적으로 우리는 합의 노드가 실무적으로는 합리적이라고 믿으며, 매수 공격은 매수된 사용자에게 거의 위험이 없을 경우 가능할 수 있다고 본다. 불행히도 현실 세계의 매수 공격에 대한 형식적 정의와 분석은 아직 부족하다. 이에 우리는 적응적 적대자뿐만 아니라 매수 가능성까지 포착할 수 있는 새로운 위협 모델을 고려한다.

네트워크 모델. 우리의 네트워크 조건은 표준적인 GST 기반의 *부분 동기(partial synchrony) 모델* [47]을 따른다. 이 모델에서는 알려지지 않은 전역 안정화 시간(Global Stabilization Time, GST) 이전에는 네트워크가 비동기적이며, 이후에는 알려진 전달 상한 Δ 를 갖는 동기 네트워크가 된다. 우리는 세 가지 유형의 네트워크 통신을 가정한다. (i) 피어 간 신뢰 가능하고 인증된 점대점(point-to-point) 연결, (ii) 네트워크 전반에 걸친 가십(gossip) 통신, (iii) 공유 통신 채널을 통해 하나의 노드가 여러 수신자에게 메시지를 전송하는 방식이다. 특히 전용 통신 채널을 통해 n 개의 노드에 메시지를 전송하는 것은 점대점 채널이나 가십 네트워크를 통해 순진하게 n 번 전송하는 것보다 실무적으로 훨씬 저렴하다. 예를 들어, 강력한 CDN 노드를 활용하여 메시지를 캐시하고 전달할 수 있다. 네트워크 계층의 세부 사항은 4절에서 다룬다.

프로토콜 실행. 우리는 전체 참여가 동적으로 변하는 시스템을 고려하며, 이는 노드의 공개 키를 네트워크 내 스테이크에 매핑하는 공개 접근 가능한 *스테이크 테이블(stake table)*로 표현된다. 각 노드는 비밀/공개 키 쌍을 생성하고, 공개 키는 스테이크 테이블에 게시되어 노드를 유일하게 식별한다. 또한 공개 키에는 해당 노드의 스테이크로부터 비음수가 되도록 결정론적으로 계산 가능한 *가중치(weight)*가 부여된다.

각 에폭 e 의 시작 시점마다, 적대자는 참여 노드 집합과 각 노드의 스테이크 가중치를 재구성할 수 있다. 각 에폭은 여러 개의 *뷰(view)*로 구성되며, 각 뷰에는 진행을 주도하는 고유한 전용 리더가 할당된다. 에폭, 블록, 뷰는 각각 단조 증가하는 에폭 번호 e , 블록 높이 h , 뷰 번호 v 로 표기된다. 합의 프로토콜의 실행은 일반적인 상태 머신으로 모델링할 수 있으며, 임의의 시점에서의 전체 상태는 각 프로세스/노드의 내부 상태의 합과 지금까지 전달된 모든 네트워크 메시지를 기록한 *트랜스크립트(transcript)*로 구성된다.^(A4) 우리는 모든 결함 노드의 내부 상태와 트랜스크립트를 합쳐 적대자의 실행 관점(adversarial view)이라 부른다. 각 정직한 노드의 행동은 그 내부 상태와 수신한 메시지에 의해 명확히 규정되는 반면, 각 비잔틴 노드는 임의로

행동할 수 있다. 모든 정직한 노드의 행위(예: 메시지 전송 또는 합의된 값의 커밋)는 새로운 메시지의 도착에 의해서만 촉발된다는 점에 유의한다. 다음으로, 적대자가 노드를 적응적으로 손상시키는 방식을 명시한다.

메시지 전달 및 적대자 모델. 사전적 손상(예: 표적 해킹)과 수동적 손상(예: 매수 공격)을 모두 포착할 수 있는 통합 모델을 위해, 우리는 다음과 같은 메시지 전달 모델을 고려한다. 우리는 전역 메시지 큐를 나타내는 *아웃박스 버퍼(outbox buffer)*라는 자연스러운 개념을 도입한다. 정직한 노드는 새로운 메시지의 도착에 의해 트리거된 이후, 수신자(들), 송신자, 타임스탬프가 명시된 메시지를 버퍼에 게시할 수 있다. 한편 적대자는 아웃박스 버퍼에 임의의 메시지를 주입할 수 있다. 아웃박스 버퍼의 메시지가 전달되기 전에는 다음의 세 단계가 존재한다.

손상 단계(corruption stage): 아웃박스 버퍼를 관찰한 이후, 적대자는 손상된 키 뒤에 있는 누적 적대적 가중치가 임계치 tN_e 미만인 한 새로운 비잔틴 노드 집합을 적응적으로 손상시킬 수 있다. 여기서 $t = 1/3$ 이며 N_e 는 현재 에폭 e 에서의 전체 스테이크 가중치를 의미한다. 다만, 이전에는 정직했으나 이 단계에서 비잔틴으로 전환된 노드가 이미 전송한 메시지는 적대자가 삭제하거나 수정할 수 없다. 즉, 새로운 비잔틴 집합은 다음 라운드부터 효력을 갖는다. 또한 적대자는 이전에 이미 손상시킨 비잔틴 노드를 *비손상 상태로 되돌릴 수 없다*는 점에 유의한다. 이 제약은 비고 2.2에서 설명하는 공격을 배제하기 위해 필요하다.

매수 단계(bribery stage): 손상 단계 이후, 적대자는 *새롭게 손상된 노드* 중 일부 S 를 선택하여 S 가 보낸 버퍼 메시지를 삭제하거나 수정할 수 있다. $|S|$ 를 S 의 노드 수라 할 때, 우리는 적대자가 이 매수 단계에서 $|S|$ 회의 매수를 수행했다고 말한다. 매수와 손상의 차이점은, 매수된 노드는 즉시 악의적으로 행동할 수 있는 반면, 매수되지 않은 손상 노드는 다음 라운드부터 악의적으로 행동할 수 있다는 점이다. 또한 적대자는 한 에폭 동안 노드를 비손상 상태로 되돌릴 수 없으므로, 에폭당 매수 횟수는 tN_e 로 제한된다.

네트워크 지연 단계(network delay stage): 메시지 전달을 제어하는 (네트워크) 적대자는 마지막으로 아웃박스 버퍼의 메시지를 네트워크 조건을 준수하는 범위 내에서 섞거나 재정렬할 수 있으며, 전달 없이 삭제할 수는 없다.(A5) 메시지는 성공적으로 전달되면 버퍼에서 제거된다.

비고 2.1. 본 모델은 적대자가 당사자의 신원이나 이미 게시된 메시지를 알지 못한 상태에서도 매수를 수행할 수 있는 공격 능력을 포착한다. 이는 적대자가 아직 전송되지 않은 버퍼 메시지를 관찰한 뒤 새로운 손상 집합을 결정하기 때문이다. 이 새롭게 손상된 집합의 일부가 현재 뷰에서 즉시 임의로 행동할 수 있는 매수 집합이 되며, 나머지는 다음 뷰부터 악의적으로 행동할 수 있다.

비고 2.2. [8]에서 지적되었듯이, 일시적으로 손상되었다가 이후 비손상 상태로 복구되는 당사자를 모델링하는 것은 매우 비자명하다. 예를 들어, 비손상 이후에도 적대자는 이전에 손상된 당사자의 내부 상태(예: 장기 비밀 키)를 여전히 기억할 수 있다. 반대로, 비손상된 당사자는 적대자가 이전에 저장해 두었던 메타데이터를 잊어버릴 수 있다. 이러한 양측의 문제는 보안 분석의 복잡성을 증가시킨다. 우리의 위협 모델과 비교하면, Algorand [59]는 적응적 적대자가 결코 비손상 상태로 되돌릴 수 없으며 매수 가능성도 존재하지 않는다는 점에서 훨씬 더 강한 가정을 둔다.

Let v be the current view number and node L_v be the leader in this view. Perform the following steps.

1. **Propose (leader only).** The leader L_v broadcasts $\langle \text{propose}, B_k, v, C_{v'}(B_{k-1}) \rangle_{L_v}$. Here, $B_k := (b_k, h_{k-1})$ is the block that should extend the highest certified block B_{k-1} with certificate $C_{v'}(B_{k-1})$ known to leader.
 2. **Vote (all nodes).** Upon receiving the first valid proposal $\langle \text{propose}, B_k, v, C_{v'}(B_{k-1}) \rangle_{L_v}$ in view v : If $C_{v'}(B_{k-1})$ is ranked no lower than the locked block, then send $\langle \text{vote}, B_k, v \rangle$ as a threshold signature share to L_v . Update lock to B_{k-1} and the certificate to $C_{v'}(B_{k-1})$ to leader L_{v+1} .
- Commit rule (all nodes).** Commit block B_k iff there exists an $l \geq k$ such that $C_{v'}(B_l)$ and $C_{v'+1}(B_{l+1})$ are formed.

Figure 2: View protocol for nodes in view v .

비고 2.3. 연속된 두 에폭 e 와 $e + 1$ 의 참여자들이 동시에 적극적으로 스테이킹하는 구간이 존재할 수 있다. 예를 들어, 에폭 e 가 종료되었으나 에폭 $e + 1$ 에서는 아직 어떠한 블록도 커밋되지 않은 기간이 이에 해당한다. 이 기간 동안 우리는 에폭 e 에서 손상된 키 뒤에 있는 누적 적대적 가중치가 임계치 tN_e 미만이며, 동시에 에폭 $e + 1$ 에서 손상된 키 뒤에 있는 가중치 또한 임계치 tN_{e+1} 미만이라고 가정한다.

안전성과 활성. 본 모델에서의 합의 프로토콜은 스테이킹 노드 집합이 제출된 트랜잭션 목록의 순차적 순서에 대해 합의하도록 하는 데 사용된다. 우리는 서로 다른 정직한 스테이킹 노드가 동일한 위치에서 서로 다른 값에 커밋하지 않는 경우, 해당 프로토콜이 *안전성(Safety)*을 제공한다고 말한다. 또한 각 제출된 트랜잭션이 어떤 에폭 $e \in \mathbb{N}$ 에서든 결국 모든 정직한 스테이킹 노드에 의해 커밋되는 경우, 해당 프로토콜이 *활성(Liveness)*을 제공한다고 말한다.

2.3 HotStuff 개요

본 절에서는 허가형(permissioned) 환경에서 HotStuff-2 프로토콜의 핵심에 대한 고수준 개요를 제시한다. 보다 자세한 설명은 HotStuff [125] 및 HotStuff-2 [82]를 참조한다. 이 두 작업 모두 HotShot과 호환되지만, 본 논문에서는 HotStuff-2를 기본 프로토콜로 가정한다. 프로토콜은 뷰(view) 단위로 동작하며, 정상 상태 프로토콜(steady-state protocol)과 뷰를 전환하기 위한 뷰 동기화(view synchronization, 또는 페이스메이커) 프로토콜의 두 부분으로 구성된다. 본 절에서는 모든 올바른 노드가 충분히 긴 시간 동안 동일한 뷰에 머무르고, 모든 노드에게 알려진 해당 뷰의 지정 리더가 정직한 경우에 커밋 결정을 이끄는 정상 상태 프로토콜에만 초점을 맞춘다. 뷰 동기화와 관련된 세부 사항은 [섹션 2.4.3](#)에서 설명한다.

먼저 몇 가지 데이터 구조와 용어를 정의한다. 허가형 환경은 n 개의 노드로 구성되며, 이 중 최대 t 개가 비잔틴일 수 있다.

블록 형식. 프로토콜은 값들의 체인을 형성한다. 우리는 체인 내의 각 값을 *블록(block)*이라 부른다. 블록이 체인에서 차지하는 위치를 해당 블록의 *높이(height)*라 한다. 높이 k 에 있는 블록 B_k 는 다음과 같은 형식을 가진다.

$$B_k := (b_k, h_{k-1})$$

여기서 b_k 는 높이 k 에서 제안된 값을 나타내며, $h_{k-1} := H(B_{k-1})$ 는 이전 블록의 해시 다이제스트이다. 첫 번째 블록 $B_0 = (b_0, \perp)$ 는 선행 블록을 갖지 않는다. 이후의 모든 블록 B_k 는 이전 블록 B_{k-1} 의 해시를 포함함으로써 선행 블록을 명시해야 한다. 우리는 (i) 선행 블록이 유효하거나 $\perp$ 이며, (ii) 제안된 값이 애플리케이션 수준의 유효성 조건을 만족하고 조상 블록 체인과 일관성을 유지하는 경우(예: 조상 블록 중 하나에서 이미 사용된 트랜잭션을 이중 지불하지 않는 경우), 해당 블록을 유효(valid)하다고 말한다.

블록 확장과 상충(equivocation). $l > k$ 일 때, B_k 가 B_l 의 조상이라면 우리는 B_l 이 B_k 를 확장(extend)한다고 말한다. 두 블록 B_l 과 $B_{l'}$ 이 서로 동일하지 않고, 어느 쪽도 다른 쪽을 확장하지 않는 경우, 우리는 이 두 블록이 서로 상충(equivocate)한다고 말한다.

인증서와 인증된 블록. 프로토콜에서 노드들은 집계 서명(aggregate signature)을 사용하여 블록에 투표한다. 우리는 $C_v(B_k)$ 를 뷰 v 에서 $2t + 1$ 개의 노드가 $h_k = H(B_k)$ 에 대해 서명한 서명 집합으로 정의한다. $C_v(B_k)$ 를 뷰 v 에서의 블록 B_k 에 대한 인증서 또는 퀴럼 인증서(quorum certificate, QC)라 부른다. 인증된 블록은 인증된 뷰 번호에 따라 순위가 매겨지며, $v > v'$ 일 경우 인증서 $C_v(B_k)$ 는 $C_{v'}(B_{k'})$ 보다 더 높은 순위를 가진다.

락된 블록. 임의의 시점에서 각 참여자는 자신이 알고 있는 가장 높은 순위의 인증된 블록을 락(lock)한다. 프로토콜 실행 동안 각 참여자는 모든 블록에 대한 모든 서명을 추적하고, 이에 따라 락된 블록을 지속적으로 갱신한다. 이후 설명하겠지만, 락된 블록의 개념은 커밋의 안전성을 보장하는 데 사용된다.

정상 상태 프로토콜. 그림 2는 정상 상태 프로토콜에서 실행되는 핵심 단계를 제시한다. 각 뷰에는 합의 진행을 담당하는 지정 리더가 존재한다. 우선 우리는 지정된 리더 집합을 알고 있다고 가정한다. 뷰 리더의 목표는 현재 시퀀스에서 가장 높은 순위의 인증된 블록을 확장하는 새로운 블록을 생성하고, 이를 인증받는 것이다.

- 뷰 v 의 리더는 제안 값, 뷰 번호, 그리고 자신이 알고 있는 가장 높은 순위의 인증서를 포함하는 높이 k 의 블록 B_k 를 구성한다. 리더는 제안된 블록을 모든 노드에게 전송한다.
- 각 참여자는 자신이 받은 인증서 중 가장 높은 순위의 인증서를 유지한다. 참여자는 블록 B_k 가 자신이 알고 있는 최고 인증서를 확장하는 경우에만, 다음 리더에게 서명된 투표를 전송함으로써 해당 블록에 투표한다.
- $2t + 1$ 개의 노드가 블록에 투표하면 해당 블록은 인증된다. $2t + 1$ 개의 서명된 투표로부터 형성된 인증서 $C_v(B_k)$ 는 다음 뷰의 리더에 의해 집계된다.

블록 커밋. 블록 B_k 는 $l \geq k$ 인 어떤 l 에 대해 $C_v(B_l)$ 과 $C_{v+1}(B_{l+1})$ 가 형성되고, B_l 이 B_k 를 확장하는 경우 커밋되었다고 한다. 즉, B_k 자체이거나 그 후속 블록 중 하나에 대해, 연속된 두 높이의 블록이 연속된 두 뷰에서 인증되면 커밋이 발생한다.

안전성과 활성화에 대한 직관. 단일 뷰 내에서의 안전성은 단순한 퀴럼 교차(quorum intersection) 논증에 의해 보장된다. 즉, 비잔틴 노드 수가 t 로 제한되는 한 서로 다른 두 블록이 동시에 인증될 수 없다. 뷰 간 안전성은 락킹 메커니즘에 의해 보장된다. 어떤 노드가 뷰 v 에서 블록을 커밋하면, $\geq t + 1$ 개의 정직한 노드가 해당 블록에 락된다. 이 조건은 상위 뷰에서 상충하는 블록에 대해 락 및 그에 따른 커밋이 발생하는 것을 방지한다. 활성화에 대한 논증은 보다 미묘한데, 개략적으로는 정직한 리더가 결국 가장 높은 순위의 인증된 블록을 확장하는 블록을 제안할 수 있으며, 이 블록은 모든 정직한 노드에 의해 투표될 수 있다는 사실에 의존한다.

2.4 HotShot 합의를 향하여

HotShot은 HotStuff 프로토콜을 지분증명(proof-of-stake, PoS) 환경으로 확장한 것이다. 본 절에서는 HotShot에서의 키 관리 방식(섹션 2.4.1), 스테이크 집합이 변경될 때의 재구성 절차(섹션 2.4.2), HotShot이 사용하는 뷰 동기화 프로토콜(섹션 2.4.3), 노드 선출을 위해 사용되는 무작위성의 출처(섹션 2.4.4)를 설명한 뒤, 이 모든 요소를 결합하여 전체 프로토콜을 기술한다(섹션 2.4.5).

2.4.1 스테이킹 키 처리

PoS 프로토콜의 핵심 구성 요소 중 하나는 공개 키 또는 참여자 식별자를 시스템 내 스테이크에 매핑하는 스테이크 테이블(stake table)이다. HotShot에서 노드들은 스테이크 테이블을 통해 누가 블록에 투표할 수 있는지를 판단한다. 일부 시스템과 달리 노드들이 공유 상태로 스테이크 테이블을 직접 관리하도록 하지 않고, 우리는 이 스테이크 테이블을 섹션 1.3에서 소개한 L1 상의 시퀀서 컨트랙트(sequencer contract)의 일부로 구현한다.

이 접근 방식에는 여러 가지 장점이 있다. 첫째, 스테이크 이전(stake transfer)이 HotShot 외부에서 처리되므로, 우리의 해법은 실행 계층을 포함하지 않는 순수한 비잔틴 원자적 브로드캐스트(Byzantine atomic broadcast)로 유지된다.

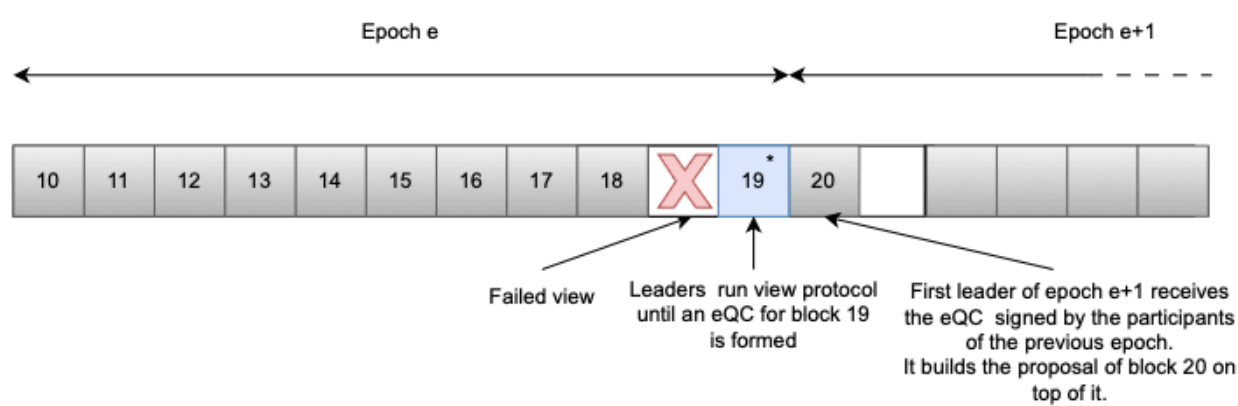
성능 측면을 넘어, 이러한 설계 결정은 EigenLayer [49]와 같은 리스테이킹(restaking) 프로토콜을 활용할 수 있게 하여, L1 스테이킹 토큰(예: 이더리움의 ETH 토큰)을 보유한 누구나 HotShot에 참여할 수 있도록 한다. 실무적으로 노드들은 L1 블록체인을 모니터링하여 스테이크 테이블의 상태를 파악한다. 이를 통해 각 에폭의 시작 시점마다 모든 정직한 노드는 다음 에폭에서 사용될 스테이크 테이블의 버전에 대해 합의할 수 있다. 효율성을 위해, 노드들은 쿼럼 인증서(QC) 검증 시 빠른 접근이 가능하도록 최적화된 스테이크 테이블의 로컬 사본을 유지한다. 마지막으로, 스테이크 테이블이 이더리움 상에 존재함으로써 장거리 공격(long-range attacks) [44, 56]이 방지된다는 점에 유의한다. 새로운 노드가 참여하여 시퀀서 체인을 따라잡고자 할 때, 이 노드는 이더리움으로부터 도출된 마지막 에폭의 참여자들에게 최신 상태를 요청할 수 있다. 이후 스테이크의 tN_e 초과 비율을 보유한 노드들로부터 동일한 응답을 수집하면, 해당 노드는 안전하게 블록 시퀀스를 다운로드하여 상태를 로컬에서 재계산할 수 있다.

2.4.2 에폭 간 재구성

확장된 쿼럼 인증서. 우리는 HotShot에서 스테이크 재구성을 지원하기 위해 쿼럼 인증서(quorum certificate, QC)의 개념을 확장한다. 블록 B_k 에 대해, 확장된 쿼럼 인증서(extended Quorum Certificate, eQC)는 연속된 세 개의 뷰에서 B_k 에 대해 생성된 세 개의 연속된 QC로 구성된다. 처음 두 개의 연속된 QC는 섹션 2.3의 커밋 규칙에 따라 블록 B_k 가 해당 에폭에서 커밋되었음을 보장하고, 마지막 QC는 해당 에폭에서 정직한 노드의 쿼럼이 블록 B_k 를 커밋했음을 보장한다.

재구성 메커니즘. HotStuff-2에서는 참여자 집합이 고정되어 있지만, 우리의 설정에서는 참여자가 프로토콜에 동적으로 참여하거나 이탈할 수 있도록 하고자 한다. 이를 위해 우리는 각 구성원의 투표 권한을 추적하는 스테이크 테이블에 의존한다. 스테이크 테이블은 체인에 블록이 추가될 때마다 갱신되지만, 참여자 집합을 매 블록마다 새로 고칠 수는 없다. 어떠한 결정이 내려지기 전에, 먼저 누가 투표할 수 있는지에 대한 합의가 필요하기 때문이다. 이러한 이유로 에폭 e 의 시작 시점마다 스테이크 테이블의 스냅샷을 주기적으로 생성한

다. 이 스냅샷은 다음 에폭 $e + 1$ 의 참여자 집합을 정의하며, 에폭 e 의 첫 번째 블록 시점의 테이블 상태에 해당한다. 프로토콜의 첫 번째 에폭에 대해서는 참여자 집합이 제네시스 블록에 하드코딩되어 있다.



에폭 $e + 1$ 의 스테이킹 노드 집합이 에폭 $e + 1$ 에서 커밋될 첫 번째 블록 B 에 대해 합의하기 위해서는, 먼저 B 가 확장할 수 있는 에폭 e 의 마지막 커밋 블록에 대해 합의해야 한다. 이를 달성하기 위해(그림 3의 예시 참조), 에폭 e 의 스테이킹 집합이 에폭 e 에서 마지막 블록 B' 를 커밋하려는 단계에서, 노드들은 B' 에 대한 확장된 QC가 형성될 때까지 HotStuff-2의 뷰 프로토콜을 계속 실행한다. 이후 에폭 e 의 노드들은 해당 eQC를 에폭 $e + 1$ 의 노드들에게 전송하고, 에폭 $e + 1$ 에서 정직한 노드의 쿼럼이 eQC를 수신하도록 보장한다. 이를 달성하기 위한 구체적인 메커니즘은 [섹션 2.4.3](#)에서 명시한다. eQC를 획득한 이후, 에폭 $e + 1$ 의 정직한 리더는 B' 가 이전 에폭에서 커밋된 유일한 블록임을 확신할 수 있으므로, 투표자들에게 받아들여질 수 있는 유효한 제안을 생성할 수 있다.

2.4.3 뷰 동기화(View Synchronization)

활성(liveness)은 GST 이후에 정직한 노드들이 충분히 긴 시간 동안 동일한 뷰에 머무르는 경우에만 보장될 수 있다 [89]. HotShot의 뷰 동기화 모듈은 이러한 조건을 만족하도록 HotStuff [125]의 페이스메이커(Pacemaker) 모듈을 확장한다. HotShot은 Naor–Keidar [90] 프로토콜을 사용하여 뷰 동기화를 달성한다. Naor–Keidar 프로토콜은 비잔틴 결함이 존재하더라도 기대값 및 낙관적 조건에서 선형 통신 복잡도와 상수 지연 시간을 제공하므로, HotShot이 대규모의 이질적인 네트워크로 확장될 수 있도록 한다. 이는 부분 동기(partially synchronous) 프로토콜로, 세 번의 투표 라운드를 거쳐 진행되며, 이 과정에서 노드들은 뷰 동기화 투표를 릴레이 노드(relay nodes)에게 전송한다. 릴레이 노드는 합의 리더와는 독립적으로 선택된 합의 노드로, 뷰 동기화 투표를 수집하여 인증서를 형성한다.

뷰 동기화 모듈은 합의 모듈과 병렬로 비동기적으로 실행된다. HotShot에서 노드는 네트워크 내 충분한 수의 정직한 노드가 동일한 뷰 번호에 합의하고 있다는 충분한 증거가 없는 한, 로컬에서 새로운 뷰로 진입하지 않는다. 이러한 증거는 다음 세 가지 형태 중 하나일 수 있다.

- 이전 뷰에 대한 쿼럼 인증서(quorum certificate). (또는 새로운 에폭의 첫 번째 뷰인 경우, 이전 에폭에서의 확장된 QC.)
- 이전 뷰에 대한 타임아웃 인증서(timeout certificate).
- 현재 뷰에 대한 뷰 동기화 인증서(view synchronization certificate).

합의 모듈은 쿼럼 인증서와 타임아웃 인증서의 생성을 담당한다. 타임아웃 인증서(앞선 단순화된 설명에서는 다루지 않았음)는 HotStuff에서도 사용되는 개념으로, 리더로부터 유효한 메시지를 수신하기 전에 노드의 타임아웃이 발생했음을 나타낸다는 점에서 QC와 유사하다. 뷰 동기화 모듈은 Naor-Keidar 프로토콜 동안 생성되는 뷰 동기화 인증서를 형성하는 역할을 담당한다. 뷰 동기화 인증서는 앞의 두 가지 인증서가 모두 존재하지 않는 경우에만 필요하다.

노드의 로컬 뷰 동기화 과정은 두 가지 방식으로 트리거될 수 있다. 첫째, 노드가 하나의 뷰를 종료했지만 다음 뷰로 진입하기에 충분한 증거를 보유하지 못한 경우, 합의 모듈로부터 직접 트리거될 수 있다. 둘째, 노드는 유효한 뷰 동기화 메시지를 수신함으로써, 이미 네트워크에서 진행 중인 뷰 동기화 프로토콜에 참여할 수도 있다. 두 번째 트리거는 노드가 로컬에서 뷰 타임아웃을 관측하지 못했더라도 뷰 동기화에 참여할 수 있도록 한다. 뷰 동기화 모듈이 유효한 뷰 동기화 인증서를 확인하면, 합의 모듈로 ViewChange 이벤트를 전송하며, 합의 모듈은 이에 따라 자신의 상태를 갱신한다.

예폭 간 ViewSync. [섹션 2.4.2](#)에서 설명한 바와 같이, 예폭 e 의 노드들이 마지막 블록을 커밋하고 eQC를 획득한 이후에는, 해당 eQC를 예폭 $e + 1$ 의 충분히 많은 정직한 노드에게 전달하고 예폭 $e + 1$ 에서 뷰 동기화를 유도해야 한다. 이를 위해 우리는 Naor-Keidar 프로토콜의 변형을 실행한다. 예폭 e 의 노드들이 뷰 v 에서 eQC를 획득했다고 가정하고, 예폭 $e + 1$ 에서 뷰 $v + 1$ 을 위한 의사무작위 릴레이 노드를 $R_1, R_2, \dots, R_n$ 이라 하자. 예폭 e 의 노드들은 먼저 eQC를 릴레이 R_1 에게 전송하고, R_1 은 예폭 $e + 1$ 의 노드 집합을 대상으로 Naor-Keidar 프로토콜에서의 무작위 릴레이 역할을 수행한다. 뷰 동기화 인증서를 획득한 이후, R_1 은 해당 인증서를 다시 예폭 e 의 노드들에게 전송하여, 충분히 많은 정직한 노드가 eQC를 수신하고 뷰 $v + 1$ 로 진입했음을 설득한다. 예폭 e 의 어떤 노드가 타임아웃 이후에도 뷰 동기화 인증서를 수신하지 못한 경우, 해당 노드는 eQC를 릴레이 R_2 에게 전송하고, R_2 는 R_1 과 동일한 알고리즘을 실행하며, 이러한 과정이 순차적으로 반복된다. 이 방식은 기대값 기준으로 선형 통신 복잡도를 가지지만, 최악의 경우에는 이차 통신 복잡도를 가진다. 이 방식이 Naor-Keidar 프로토콜과 다른 주요 차이점은, 뷰 동기화를 트리거하는 노드 집합(즉, 예폭 e 의 노드들)과 실제로 뷰 동기화 프로토콜을 실행하는 노드 집합(즉, 예폭 $e + 1$ 의 노드들)이 서로 다를 수 있다는 점이다.

2.4.4 탈중앙화 랜덤 비콘

우리는 합의 리더, 뷰 동기화 릴레이 노드, 그리고 데이터 가용성 위원회를 선택하기 위한 신뢰 가능한 무작위성의 출처로 *탈중앙화 랜덤 비콘(decentralized random beacon, DRB)*을 활용한다. 합의 리더나 뷰 동기화 릴레이를 결정론적 방식(예: 라운드 로빈 방식)으로 선택할 경우, 정적 적대자 하에서 선형 지연 시간이 발생할 수 있다. 이러한 이유로 우리는 안전한 무작위성 출처를 사용하여, 적응적 손상이 없는 낙관적 시나리오에서 다음 뷰로의 전환과 합의 진행이 상수 기대 시간 내에 이루어지도록 한다.

우리의 위협 모델에서는 예측 불가능성(unpredictability)과 비편향성(unbiasability)이라는 기존의 보안 요구사항만으로는 충분하지 않다. 실제로 DRB에 대한 매수 공격을 허용할 경우, 적대자는 시드를 편향시켜 다수의 연속된 뷰에서 반복적으로 선출될 수 있다. 이 경우 정적 적대자조차 추가적인 자원을 소모하지 않고도 시드를 지속적으로 통제할 수 있게 되어, 활성(liveness)을 무기한으로 저해할 수 있다. 또한 리더가 명시적 보상이나 MEV 추출의 형태로 수익을 획득한다는 점도 고려해야 한다. 적대자가 매수 공격을 통해 비콘을 조작하고 더 자주 선출될 수 있다면, 블록 생성으로부터 얻는 수익 역시 증가하게 된다. 일부 시나리오에서는 이러한 부당한 수익의 일부를 다시 매수 공격 자금으로 재투자하는 것도 가능하다. 매수 공격에 강인한

DRB 없이 리더 선출을 처리하려면, 결정론적 방식(예: 라운드 로빈)에 의존할 수밖에 없다. 그러나 이는 실무적으로 문제가 될 수 있는데, 에폭 동안 모든 스테이킹 단위를 순회해야 하므로 에폭의 길이를 사전에 제한할 수 없게 되며, 그 결과 새로운 스테이킹 키를 등록하는 데 오랜 시간 또는 가변적인 시간이 소요될 수 있어 사용자 경험을 저해할 수 있다.

한편, 다수의 참여자를 다루는 상황에서 낮은 통신 복잡도를 유지하는 것 또한 큰 도전 과제이다. 무작위 값을 브로드캐스트하는 단순한 방식은 과도한 통신을 초래하므로, 우리는 선형 통신을 달성하기 위해 하나의 리더가 모든 무작위 조각을 수집하는 구조를 요구한다. 그러나 이는 악의적인 리더가 포함할 조각을 임의로 선택함으로써 출력을 편향시킬 수 있는 여지를 남긴다. 이러한 문제를 해결하기 위해, 우리는 병렬 환경에서도 일정 시간이 반드시 소요되는 지연 함수(Delay Function)를 활용한 “커밋-리빌(commit-reveal)” 접근법을 채택한다. 리더가 비콘 값으로 사용될 출력값을 얻기 전에 지연 함수의 입력을 커밋하도록 함으로써, 두 가지 목표를 모두 달성할 수 있다. 자세한 내용은 [부록 B.1](#)에 제시한다.

마지막으로, 이론적으로는 이더리움의 랜덤 비콘 [48]을 활용할 수도 있음을 언급한다. 현재 RANDAO [103] 기반 구현은 매수 공격에 강인하지 않지만, 본 연구에서 제안한 것과 같은 VDF 기반 개선이 적용된다면 우리의 요구에 부합하는 해법이 될 수 있다.

2.4.5 구성 요소 통합

본 절에서는 [섹션 2.4.1](#), [섹션 2.4.2](#), [섹션 2.4.3](#), [섹션 2.4.4](#)에서 제시한 설계를 결합하여, 그림 4에 나타난 HotShot 프로토콜을 설명한다.

스테이크 등록 및 갱신. 스테이킹 키(및 스테이크 수량)는 스테이크 테이블 $\mathcal{T}$ 에 등록(및 갱신)될 수 있다. 키 등록 또는 스테이크 이전 시점에는 해당 키의 스테이크 수량이 일시적으로 변경되지 않는다. 스테이크 테이블의 갱신은 매 Γ 블록 주기마다만 활성화된다. 보다 구체적으로, Γ 는 하나의 에폭 동안 스테이크 테이블 구성이 변경되지 않는 블록 수를 의미한다. 에폭 $e \in \mathbb{N}$ 에 대해, $\tilde{\mathcal{T}}[e]$ 는 에폭 e 에서 투표에 참여하는 스테이킹 키 집합을 나타내며, 이는 에폭 $e - 1$ 이전에 수행된 모든 스테이킹 키 등록 및 갱신을 반영한 이후의 스테이킹 집합이다.

랜덤 비콘 시드. 이후 설명하겠지만, 합의 프로토콜은 진행을 주도할 무작위 리더를 지속적으로 샘플링해야 한다. 이를 위해 우리는 각 에폭마다 랜덤 비콘 시드를 생성하는 메커니즘이 필요하다. 보다 구체적으로, 각 에폭 $e + 1 \in \mathbb{N}$ 에 대한 랜덤 비콘 시드는 에폭 e 에서 처음 커밋된 블록의 QC를 입력으로 한 지연 함수의 출력으로 정의한다. [섹션 2.4.4](#)에서 설명한 바와 같이, DRB 시드는 매수 공격 하에서도 예측 불가능하고 비편향적이다. 또한 에폭 $e + 1$ 에 대한 스테이크 테이블 구성은 에폭 $e - 1$ 의 종료 시점에 이미 확정되므로, 에폭 $e + 1$ 의 리더들은 무작위로 분포하며, 적대자가 에폭 동안 노드를 적응적으로 손상시키지 않는 낙관적 시나리오에서는 합의 진행을 이끄는 정직한 리더를 선택하는 데 상수 기대 시간이 소요된다.

워크플로우. 프로토콜 시작 시점에, 제네시스 스테이킹 집합에 속한 각 노드는 다음을 포함하는 로컬 데이터 구조와 변수를 초기화한다. (i) 최근 및 현재 에폭에 대한 스테이크 테이블 스냅샷 목록, (ii) 노드의 QC 서명 키, (iii) 커밋된 블록 목록. 이 외에도 노드는 뷰 동기화 및 무작위 리더 선택을 위한 보조 정보를 저장한다.

합의 프로토콜은 활성 스테이킹 노드 집합에 의해 실행되며, $\text{뷰}(\text{view})$ 단위로 진행된다. 노드들은 [섹션 2.4.3](#)에 명시된 메커니즘에 따라 새로운 뷰에 진입한다. 각 뷰 v 에서 노드들은 활성 스테이크 키 테이블로부터 무작위로 샘플링된 뷰 리더(*view leader*)에 대해 합의한다. 샘플링에 사용되는 무작위성은 뷰 번호 v 와 현재 에폭에 대한 랜덤 비콘 시드로부터 도출된다. 뷰 리더는 DA 계층에서 가용한 트랜잭션 데이터에 대한 짧은 커밋먼트(*short commitment*)를 선택하고, 해당 커밋먼트에 대한 제안을 브로드캐스트한다.

Parameters: Let Γ denote the number of blocks after which the list of stake table snapshots $\tilde{T}$ is updated, i.e., every Γ blocks form an *epoch*. Denote $\tilde{T}[e]$ as the snapshot used for epoch e . Denote $\text{seeds}[e]$ as the decentralized random seed used in epoch e .

Initialization: For every node in the initial staking set, initialize the stake table snapshots $\tilde{T}$, random beacon seeds seeds , and the list of committed blocks to default values.

View protocol: Let $v \in \mathbb{N}$ be the current view number. Perform the following steps.

1. **Stake table update.** For every node, let h be the block height of its last committed block before view v . If $h \bmod \Gamma = 0$, i.e., it comes to an end of an epoch, the node will update the stake table given the stake updates from L1, and append the stake table for the new epoch to the list of stake table snapshots.
2. **Seed update.** If a node commits a first block in epoch e , it computes the beacon seed $\text{seeds}[e + 1]$ for epoch $e + 1$ as $\text{DF.Eval}(C_{v'}(B_{(e-1)\Gamma+1}))$, i.e., the delay function on input the QC of the first committed block in epoch e . Note that the delay function hardness parameter is set so that the node is guaranteed to know $\text{seeds}[e + 1]$ before the end of epoch e . For newcomers in epoch $e + 1$, they can extract the seed from a committed block in epoch e .
3. **ViewSync.** Let $e = \lceil (h + 1)/\Gamma \rceil$ be the current epoch. Let L_v denote the leader of view v sampled from the stake key table $\tilde{T}[e]$ using randomness $H_1(\text{seeds}[e], v)$ (where H_1 is a hash function modeled as a random oracle). The nodes enter view v if one of the following happens:
 - L_v receives a valid QC (or extended QC from the last epoch). The leader will send the (e)QC to other nodes, who would enter view v and vote after receiving the certificates.
 - If L_v does not receive a valid QC but receives a timeout certificate (TC) from the last view v , similarly, the leader sends the TC to other nodes who then enter view v afterward.
 - If any honest node reaches the end of view $v - 1$ but doesn't receive any evidence to enter view v , it triggers the Naor-Keidar protocol [90] for view synchronization.^a
 - For the special case where view $v - 1$ is the last view in epoch $e - 1$, the honest nodes in epoch $e - 1$ will trigger the variant of the Naor-Keidar protocol and send the eQC to the nodes in epoch e (Sect. [2.4.3](#)).
4. **Propose.** The leader L_v broadcasts $\langle \text{propose}, B_k, v, C_{v'}^*(B_{k'}) \rangle_{L_v}$. Here,
 - B_k is the proposed block. As its data, it contains a short commitment to a block of raw transaction data, and an optimistic DA certificate dac from the small DA committee.
 - $C_{v'}^*(B_{k'})$ is the certificate for the block $B_{k'}$ to be extended. There are three cases:
 - **Case 1:** B_k is the first block in an epoch (i.e., $k \bmod \Gamma = 1$). Then $k' = k - 1$ and B_{k-1} is the last block committed in the previous epoch and $C_{v'}^*(B_{k-1})$ is the extended QC for block B_{k-1} (See Section [2.4.2](#)).
 - **Case 2:** B_k is not the first block in any epoch and $k' = k - 1$. Then $C_{v'}^*(B_{k-1}) = C_{v'}(B_{k-1})$ is the QC for the highest certified block B_{k-1} known to leader.
 - **Case 3:** $k' = k$ and B_k is the last block in an epoch (i.e., $k \bmod \Gamma = 0$). Then $v' = v - 1$ and $C_{v'}^*(B_k) = C_{v'}(B_k)$ is the previous QC for block B_k .
 - If L_v obtains $\text{seeds}[e + 1]$ (Section [2.4.4](#)), it also includes it in the proposal.
5. **Vote.** Upon receiving $\langle \text{propose}, B_k, v, C_{v'}^*(B_{k'}) \rangle_{L_v}$, a node sends vote $\langle \text{vote}, B_k, v \rangle$ to L_v if:
 - the node received the VID share for the block commitment inside B_k , and the DA committee certificate dac inside B_k is valid;
 - the QCs in $C_{v'}^*(B_{k'})$ are valid given the stake table $\tilde{T}[\lceil (k')/\Gamma \rceil]$ for height k' ;
 - if $k \bmod \Gamma = 1$, then $k' = k - 1$ and $C_{v'}^*(B_{k-1})$ is the extended QC of B_{k-1} ; if $k \bmod \Gamma = 0$ and $k' = k$, then $v' = v - 1$ and $C_{v'}^*(B_k) = C_{v-1}(B_k)$ is the previous QC for B_k ; otherwise $C_{v'}^*(B_{k'})$ is the QC for B_{k-1} , which is ranked no lower than the node's locked block;

- the random beacon seed field in the proposal is either empty or $\text{DF.Eval}(C_{v'}(B_{(e-1)\Gamma+1}))$ (i.e., the valid seed for epoch $e + 1$).

If the conditions hold, the node updates lock to B_{k-1} if $k \bmod \Gamma \neq 1$ and $k' = k - 1$. The node also sends a vote $\langle \text{vote}, C_{v'}^*(B_{k'}), v \rangle$ to leader L_{v+1} where $C_{v'}^*(B_{k'})$ is included as part of the vote. If the node does not receive any valid message from the leader L_v after timeout, the node sends a timeout vote $\langle \text{vote}, \perp, v \rangle$ to L_{v+1} .

Commit rule. Commit block B_k iff $\exists l \geq k$ such that $C_{v'}(B_l), C_{v'+1}(B_{l+1})$ are formed.

^aThe NK protocol setups multiple random relays so that at least one of them is honest. The nodes first send votes to the first random relay, who aggregates votes and sends back the threshold signature as a viewsync certificate. If a node does not receive a response after a timeout, it sends vote to the next random relay and runs similar procedures, etc. We refer to Section 5.2 of [90] for more details of the algorithm specification.

Figure 4: The HotShot Consensus Protocol.

제안에는 커밋먼트에 대한 DA 위원회 인증서와, 해당 커밋먼트가 확장할 인증된 블록이 함께 첨부된다. 이때 다음의 세 가지 경우가 존재한다.

- **경우 1:** 리더가 제안한 블록이 새로운 에폭 e 의 첫 번째 블록인 경우이다. 이때 인증된 블록은 이전 에폭 $e - 1$ 에서 마지막으로 커밋된 블록이어야 하며, 인증서는 해당 커밋 블록에 대한 확장된 QC(섹션 2.4.2)여야 한다.
- **경우 2:** 리더가 제안한 블록이 에폭의 첫 번째 블록이 아니며, 인증된 블록이 제안된 블록과 동일하지 않은 경우이다. 이때 인증된 블록은 동일한 에폭에 속해야 하며, 리더가 알고 있는 것 중 가장 높은 뷰를 갖는 쿼럼 인증서를 포함해야 한다.
- **경우 3:** 리더가 제안한 블록이 에폭의 마지막 블록이며, 인증된 블록이 제안된 블록과 동일한 경우이다. 이때 인증된 블록은 이전 뷰 $v - 1$ 에서의 블록이어야 한다.

다른 스테이킹 노드들은 제안을 수신한 후, 제안의 유효성을 검증하고 다음 뷰의 리더에게 투표를 전송한다. 제안은 다음의 조건을 모두 만족하는 경우에만 유효하다. (i) DA 위원회 인증서가 유효하고, 해당 노드 자신이 해당 커밋먼트에 대한 VID 조각을 수신한 경우, 그리고 (ii) 인증된 블록의 QC가 요구사항을 충족하는 경우이다. 즉, 인증된 블록이 이전 에폭의 마지막 커밋 블록인 경우에는 유효한 확장된 QC여야 하며, 그렇지 않은 경우에는 노드의 락된 블록보다 낮은 순위를 갖지 않는 유효한 QC이거나, 이전 뷰에서 제안된 블록에 대한 유효한 QC여야 한다. 에폭 e 에서 QC를 검증하기 위해, 검증 알고리즘은 QC에 포함된 투표자의 스테이킹 키를 사용하여 집계 검증 키를 구성한 뒤, (1) QC 내부의 집계 서명이 올바르게 검증되는지, 그리고 (2) QC에 해당하는 전체 스테이크 양이 에폭 e 의 전체 스테이크 N_e 에 대해 $2N_e/3$ 보다 크지를 확인한다. 자세한 내용은 [부록 B.3](#)를 참조한다.

제안이 유효한 경우, 노드는 자신의 락된 블록을 갱신한다. 다음 뷰 $v + 1$ 의 리더는 노드들이 전송한 투표를 수집하며, 충분한 수의 투표를 수신하면 자신의 로컬 정보를 갱신하고 $v + 1$ 뷰로 진행한다.

2.5 보안 스케치

우리는 HotShot의 안전성과 활성에 대해 분석한다. 증명 아이디어는 HotStuff [125] 및 HotStuff-2 [82]의 그것과 유사하지만, (i) 동적 스테이크 변경, (ii) 매수 공격을 수행하는 적대자, (iii) 적응적 적대자가 노드를 점진적으로 손상시킬 수 있다는 점에서 추가적인 도전 과제를 다루어야 한다. 도전 과제 (i)에 대한 해법은 기존 HotStuff 프로토콜에 대한 수정이 필요하며, (ii)와 (iii)은 다루기 비교적 쉽고 보안 증명에 소폭의 변경만을 요구한다.

보조정리 2.1. 스테이킹 노드가 뷰 v 에서 블록 B_k 를 커밋했다면, 뷰 번호가 v 이상인 어떤 뷰에서든 QC를 획득하는 모든 블록은 B_k 를 확장하거나 B_k 와 동일하다.

증명. 뷰 $v' \geq v$ 에서 QC를 획득하는 임의의 블록 $B_{k'}$ 를 고려하자. 여기서 해당 QC는 에폭 $e' = \lceil k'/\Gamma \rceil$ 에서의 스테이킹 노드 집합에 관한 것이다. 먼저 정의에 따라 $B_{k'}$ 는 뷰 $v' \geq v$ 에서 QC를 획득하므로, $B_{k'}$ 가 에폭 $e' < e$ 에 속할 수는 없다.

다음으로 $B_{k'}$ 가 에폭 $e' > e$ 에 속하는 경우를 고려한다. 우리는 뷰 v' 시점에는 에폭 e 의 모든 블록이 이미 커밋되었으며 포크가 불가능함을 주장한다. 연속된 두 에폭 a 와 $a + 1$ 사이에서 스테이킹 노드 집합은 완전히 달라질 수 있으므로, HotStuff의 락킹 메커니즘은 더 이상 작동하지 않는다. 다행히도 새로운 에폭의 스테이킹 집합에 대해서는 (예: 이더리움과 같은) 단일한 진실의 원천(source of truth)을 통해 모든 참여자가 에폭 $a + 1$ 의 스테이킹 노드 집합에 대해 일치된 전역 관점을 갖는다. 또한 우리의 구성은 각 에폭의 마지막 블록을 커밋하기 위해 에폭 종료 시점에 필요한 모든 QC를 수집한다. 이는 에폭 a 의 참여자가 에폭 $a + 1$ 의 스테이크 집합으로 안전성 속성을 전달할 수 있게 한다. 보다 구체적으로, 에폭 $a + 1$ 의 정직한 노드들은 에폭 a 에서 마지막으로 커밋된 블록을 확장하는 블록에 대해서만 투표하며, 에폭 a 에서 마지막 커밋 블록은 유일하므로 안전성이 성립한다.(A6)

따라서 $B_{k'}$ 는 에폭 e 의 마지막 커밋 블록을 확장해야 하며, 보조정리가 에폭 e 에서 QC를 획득하는 임의의 블록에 대해 성립한다면, 마지막 커밋 블록은 B_k 를 확장하거나 B_k 와 동일하다. 종합하면, $B_{k'}$ 와 B_k 가 동일한 에폭에 속하는 경우만 고려하면 충분하다.

마지막으로 $B_{k'}$ 와 B_k 가 같은 에폭에 속하는 경우(즉, $e' = e = \lceil k/\Gamma \rceil$)에 보조정리가 성립함을 보인다. 증명 아이디어는 [82]의 보조정리 4.1과 유사하다. 다만 [82]의 증명은 HotStuff-2의 비파이프라인(non-pipelined) 버전에만 해당하므로, 완결성을 위해 여기서 다시 증명한다. 먼저 $v' = v$ 라면, 뷰 v 에서의 쿼럼 교차 논증에 의해 $B_{k'}$ 는 B_k 와 동일하거나 B_k 를 확장해야 한다. 이제 뷰 v 부터 $v' - 1$ 까지 QC를 획득하는 모든 블록이 B_k 를 확장하거나 B_k 와 동일하다고 가정하자. 귀납 단계인 $v' > v$ 를 고려하면, B_k 가 뷰 v 에서 커밋되었으므로 커밋 규칙에 따라, 에폭 e 의 스테이킹 노드 중 스테이크 가중치가 최소 $2tN_e + 1$ 인 부분집합 S 가 뷰 v 종료 시점에 B_k 와 동일하거나 B_k 를 확장하는 블록에 락되었음이 성립한다(여기서 tN_e 는 에폭 e 의 비잔틴 스테이크 임계치이다). 귀납 가정과 락킹 규칙에 의해, 뷰 $v' - 1$ 종료 시점에도 S 의 정직한 노드들은 여전히 B_k 와 동일하거나 B_k 를 확장하는 블록에 락되어 있다. 뷰 v' 에서, w 를 S 내 비잔틴 노드가 통제하는 스테이크 가중치라 하고, w' 를 S 내에서 적대자에 의해 매수된 노드들의 스테이크 가중치라 하자. 위협 모델의 정의에 따라 $w + w' \leq tN_e$ 가 성립하므로, S 내에서 매수되지 않은 정직한 노드들의 가중치는 최소 $tN_e + 1$ 이다. 또한 적대자는 결코 노드를 비손상 상태로 되돌리지 않으므로, S 내의 매수되지 않은 정직한 노드들은 여전히 뷰 $v' - 1$ 에서의 최고 락 블록을 기억한다. 이제 모순을 위해, 뷰 v' 에서 제안된 $B_{k'}$ 가 B_k 와 동일하지도 확장하지도 않는다고 가정하자. 귀납 가정에 의해, $B_{k'}$ 가 확장하는 블록이 가질 수 있는 QC의 뷰는 v 보다 작은 값뿐이다. $B_{k'}$ 와 뷰 v' 가 여전히 에폭 e 에 속한다는 가정하에, 뷰 번호가 v 이상인 락 블록을 가진 매수되지 않은 정직한 노드들이 최소 $tN_e + 1$ 의 가중치로 존재한다. 따라서 $B_{k'}$ 는 뷰 v' 에서 QC를 획득하기에 충분한 투표를 얻을 수 없으며, 이는 모순이다. □

비고 2.4. 에폭마다 적대자가 비잔틴 노드를 비손상 상태로 되돌릴 수 없다는 제약은 위의 증명에 필수적이다. 그렇지 않으면 비손상된 정직한 노드들이 이전에 보유하던 중요한 메타데이터(예: 최고 락 블록)를 기억하지 못할 수 있고, 안전성을 지키는 정직한 노드의 수가 충분하지 않게 된다. 또한 적대자는 비손상된 노드들의 장기 비밀 키를 기억한 채 그들을 대신하여 메시지를 위조할 수도 있다. 적대자 모델의 선택에 대한 추가 논

의는 비고 2.2를 참조한다.

정리 2.2 (안전성). 임의의 블록 높이 h 에 대해, 모든 두 스테이킹 노드는 동일한 블록을 커밋한다.

증명. 모순을 위해 두 노드가 높이 h 에서 서로 다른 두 블록 B_h 와 B'_h 에 커밋했다고 가정하자. 각각 B_h (및 B'_h)와 동일하거나 이를 확장하며, 이후 연속된 2개의 QC를 획득하는 최초의 인증된 블록을 B_k (및 $B_{k'}$)라 하자. B_k 와 $B_{k'}$ 가 커밋되는 뷰를 각각 v 와 v' 라 하자. 일반성을 잃지 않고 $v \leq v'$ 라고 가정하고, $v = v'$ 인 경우에는 $k' \geq k$ 라고 가정하자. 보조정리 2.1에 의해 $B_{k'}$ 는 B_k 를 확장하거나 B_k 와 동일해야 하며, 이는 $B_h = B'_h$ 를 함의하므로 모순이다. $\square$

정리 2.3 (활성). GST 이후 충분히 긴 시간 동안 모든 정직한 노드가 연속된 뷰 $v, v + 1, v + 2$ 에서 동기화되어 있고, 세 뷰의 리더 L_v, L_{v+1}, L_{v+2} 가 모두 정직하다면, 정직한 노드들은 뷰 $v + 2$ 에서 새로운 블록을 커밋한다.

증명. GST 이후의 시점을 고려하자. 뷰 v 에서 정직한 리더 L_v 가 제안한 블록을 블록 높이 h 의 B_h 라 하고, $e = \lfloor h/\Gamma \rfloor$ 를 블록 B_h 의 에폭이라 하자. 우리는 두 가지 경우를 고려한다.

- $h \bmod \Gamma = 1$ 인 경우, 즉 B_h 가 에폭 e 에서 커밋될 첫 번째 블록이며, B_h 가 에폭 $e - 1$ 에서 마지막으로 커밋된 블록 B_{h-1} 을 확장하는 경우이다. 이 경우 보조정리 2.1의 증명과 유사한 논증에 따라, 에폭 e 와 $e - 1$ 에 대해 만장일치로 합의된 스테이킹 집합이 존재한다. 또한 에폭 $e - 1$ 의 정직한 노드들은 에폭 e 에서 첫 번째 블록이 커밋될 때까지 블록 B_{h-1} 에 대한 eQC를 에폭 e 의 리더들에게 전송한다. 따라서 정직한 리더 L_v 는 B_{h-1} 을 수신하고 B_h 를 제안할 수 있음이 보장된다. 마지막으로 에폭 e 의 모든 정직한 스테이킹 노드들은 B_h 가 마지막 커밋 블록 B_{h-1} 을 확장하므로 B_h 를 수용하고 투표한다. ($\wedge 7$)
- B_h 가 에폭에서 커밋될 첫 번째 블록이 아닌 경우이다. 이 경우 충분한 동기화 시간이 주어진다면, 정직한 노드들은 자신의 최고 락 블록을 L_v 에게 전송하고, 리더는 GST 이후이므로 이를 뷰 v 에서 모두 수신한다. 따라서 제안 B_h 는 모든 정직한 노드들 사이에서의 최고 락 블록을 확장하게 되며, 모든 정직한 노드는 이 블록에 투표한다.

두 경우 모두에서 쿼럼 인증서 $C_v(B_h)$ 가 형성된다. 뷰 $v + 1$ 과 $v + 2$ 에 대해서도 유사한 논증을 적용하면, B_h 를 확장하는 블록 B_{h+1} 과 B_{h+2} 가 존재하여, 각각 뷰 $v + 1$ 과 $v + 2$ 에서 QCs $C_{v+1}(B_{h+1})$ 및 $C_{v+2}(B_{h+2})$ 가 형성된다. (특수한 경우로, B_h 가 에폭의 마지막 커밋 블록인 시나리오가 있으며, 이때는 $B_{h+2} = B_{h+1} = B_h$ 가 된다.) 커밋 규칙에 의해 정직한 노드들은 뷰 $v + 2$ 종료 시점에 B_h 를 커밋하게 되며, 이것으로 증명이 완료된다.

비고 2.5. 에폭마다 적대자가 비잔틴 노드를 비손상 상태로 되돌릴 수 없다는 제약은 활성 증명에도 필수적이다. 그렇지 않으면 B_h 가 에폭의 첫 블록이 아닌 경우, 최고 락 블록을 전송하는 정직한 노드 집합과 블록에 투표하는 정직한 노드 집합이 완전히 달라질 수 있다. 이때 후자의 집합에 속한 노드들이 더 높은 락 블록을 보유하고 있어 리더 L_v 가 제안한 블록에 투표하지 않는 상황이 발생할 수 있다.

비고 2.6. 세 개의 연속된 정직한 리더라는 가정은 합리적이다. 에폭마다 손상 가중치의 총량은 제한되어 있으며, 적대자가 손상 예산을 소진한 경우(즉, 손상 비율 임계치가 $t = 1/3$ 일 때의 N_e/t)에는 리더의 신원이 알려진 직후 세 뷰에서 정직한 리더 중 하나를 즉시 손상시키는 것이 불가능하다. 또한 리더는 무작위로 선택되므로, 세 리더가 연속으로 정직할 확률은 최소 $(1 - t)^3$ 이다.

비고 2.7. GST 이후 모든 정직한 노드가 충분한 시간 동안 동일한 뷰에 머무른다는 가정은 합리적이다. GST 이후, 정직한 리더 L_v 가 이전 뷰의 유효한 QC 또는 타임아웃 인증서를 수신하거나, 이전 에폭의 마지막 블록에 대한 유효한 확장된 QC를 수신하면, L_v 는 해당 인증서를 다른 노드들에게 전송하고, 정직한 노드들은 인증서를 확인한 뒤 새로운 뷰로 진입한다. 타임아웃 이후에도 어떤 증거 인증서도 수신되지 않으면, 정직한 노드는 Naor–Keidar 프로토콜 [90]을 트리거한다.(^8) NK 프로토콜은 서로 다른 에폭에 걸쳐 실행되지 않으며, 참여자 집합은 고정되어 있고(즉, 에폭 스테이킹 집합), 그 안에서 악의적 노드의 가중치는 tN_e 미만이다. 따라서 Naor–Keidar 프로토콜의 보안에 의해 정직한 노드들은 동기화 상태를 유지하게 된다. 적대자가 선형 개수의 무작위 릴레이를 적응적으로 손상시키면 NK 프로토콜의 통신 복잡도는 이차가 될 수 있다는 점에 유의한다. 그러나 적대자는 결코 노드를 비손상 상태로 되돌릴 수 없고, 에폭당 누적 적대적 가중치는 tN_e 로 제한되므로, 전체 에폭 동안의 적응적 손상 총량은 여전히 선형이며, 뷰 동기화의 총 통신 복잡도도 최악의 경우 이차로 유지된다. 이는 에폭당 뷰 수가 충분히 크다면, 상각(amortized) 복잡도가 선형으로 유지될 수 있음을 의미한다.

3 Tiramisu 데이터 가용성

3.1 개요

기존의 합의 프로토콜에서는 각 노드가 새로운 블록을 최종화하기 위해 투표하기 전에, 해당 블록의 전체 데이터 페이로드(payload)를 확인한 이후에만 투표한다. 이 순진한 전략은 대형 페이로드가 블록을 최종화하기 위해 초다수(supermajority) 노드에게 전달되어야 하므로, 프로토콜의 처리량과 지연 시간을 제한한다. 네트워크에 n 개의 노드가 있고 페이로드의 크기가 $|B|$ 라면, 이 전략에서의 전체 네트워크 통신량은 $O(n|B|)$ 이며, 이는 $O(|B|)$ 통신을 목표로 하는 우리의 목적과 충돌한다. 처리량과 지연 시간의 추가적인 개선을 실현하기 위해서는, 각 투표 노드가 최종화 이전에 전체 페이로드를 확인해야 한다는 기존의 요구 사항을 완화해야 한다.

그러나 이러한 성능 개선 욕구는, 네트워크에 대한 강력한 매수 공격에 대해서도 보안을 확보하고자 하는 우리의 욕구와 긴장 관계에 있다. 블록이 최종화되기 전에 전체 페이로드가 쿼럼에 도달하도록 요구하지 않는다면, 매수 적대자는 최종화 이전에 전체 페이로드를 확인한 소수의 노드를 손상시킬 수 있다. 그 경우 블록은 최종화되지만 페이로드는 영원히 소실되며, 이는 활성의 치명적 실패로 이어진다.

이러한 긴장 관계가 바로 *데이터 가용성(data availability, DA) 문제*의 본질이다. 본 절에서는 DA 문제에 대한 효율적인 3계층 해법인 Tiramisu를 제안한다. 각 계층은 보안–성능 트레이드오프 곡선 상의 한 지점을 나타낸다(그림 5 참조).

기본 계층. Savioardi: 매수 공격에 강인한 데이터 가용성. 드물게 네트워크는 *비관적(pessimistic)* 조건에서 동작한다. 즉, 강력한 능동적 또는 매수 적대자가 네트워크를 적극적으로 공격하고 있으며, Tiramisu의 다른 계층들이 이 공격 기간 동안 작동을 중단할 수 있다.

이 경우 우리는 이더리움의 Danksharding 제안 [96]과 유사한 해법에 의존한다. 새로운 블록의 제안자 (proposer)는 페이로드를 소거 코드(eraser code)로 인코딩한다. 그 다음, 크기 $|B|$ 의 전체 페이로드를 n 개의 모든 노드에게 브로드캐스트하여 $O(n|B|)$ 통신을 수행하는 대신, 제안자는 인코딩된 페이로드를 작은 조각(shares)으로 분할하고, 네트워크 내 $O(|B|)$ 크기의 노드 부분집합에 이 작은 조각들만을 분배한다($O(|B|)$ 통신). Savoiardi에 대한 전체 설명은 [부록 A](#)에 제시한다.

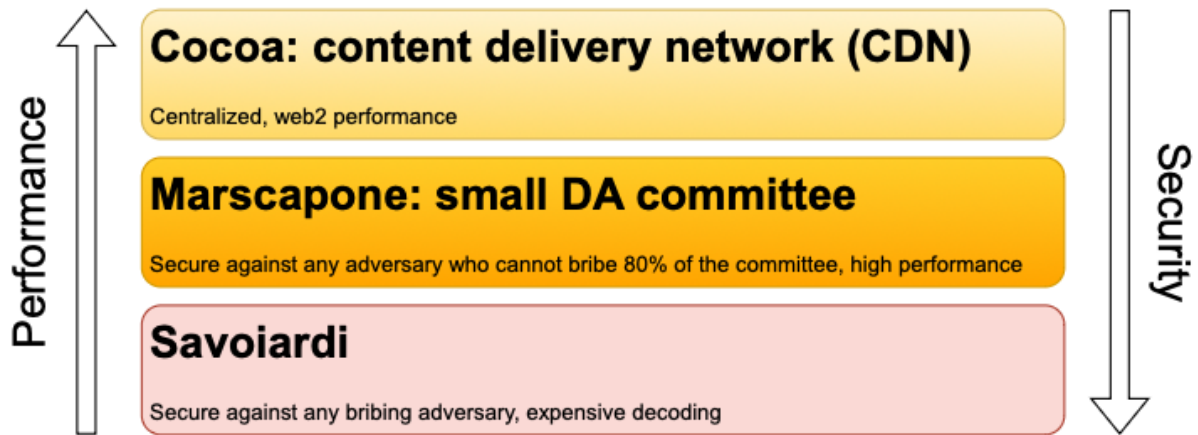


Figure 5: Tiramisu: a three-layer solution for data availability (DA).

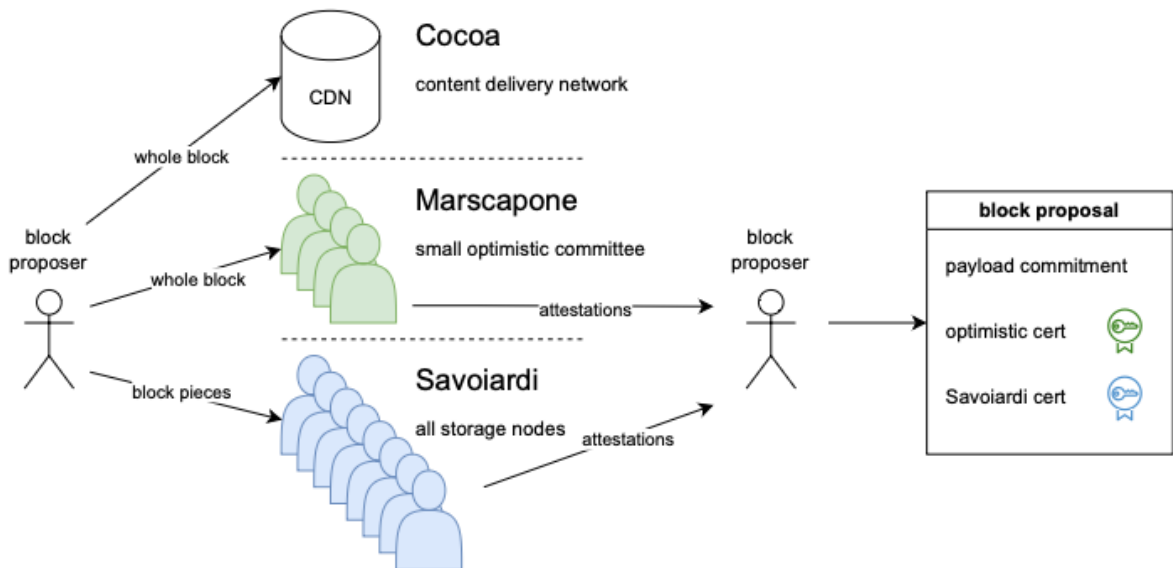


Figure 6: Tiramisu flow. Block proposer disperses block among storage nodes, aggregates attestations into certificates, builds a candidate HotShot block.

보안. 소거 코드로 인코딩된 페이로드는 충분히 큰 임의의 조각 부분집합으로부터 전체 페이로드를 복구할 수 있다는 성질을 가진다. 따라서 강력한 적대자가 다수의 노드를 손상시키더라도, 남아 있는 정직한 노드들이 페이로드를 복구할 수 있다.

성능. Savoiardi의 단점은 어느 단일 노드도 전체 페이로드를 보유하지 않는다는 점이다. 전체 페이로드를 복구하려는 클라이언트는 다수의 저장 노드로부터 조각을 다운로드하고, 페이로드를 디코딩하기 위해 계산 자원을 투입해야 한다. 따라서 Savoiardi는 강력한 적대자에 대한 궁극적 방어 수단이지만, 성능상의 이유로 최악의 네트워크 조건을 제외한 대부분의 상황에서는 사용을 피해야 한다.

중간 계층. Mascarpone: 빠른 합의를 위한 소규모 DA 위원회. 대부분의 시간 동안 우리는 네트워크가 낙관적(*optimistic*) 조건에서 동작할 것으로 기대한다. 즉, 네트워크가 강력한 능동적 또는 매수 적대자의 공격을 받고 있지 않은 상황이다.

이러한 경우, 네트워크는 *Savoirdi*의 비용이 큰 페이로드 복구 과정을 회피하고 대신 일정한 상수 크기의 소규모 DA 위원회에 의존할 수 있다. 블록 제안자 또는 다른 참여자가 새로운 블록의 페이로드를 위원회에 업로드하고, 위원회 구성원 중 쿼럼으로부터 증명(*attestation*)을 수신한 이후 블록이 최종화된다.(^9)

보안. 위원회 구성원은 무작위로 선택되며, 위원회는 각 에폭의 시작 시점마다 교체된다. 이 선택에 필요한 무작위성은 [섹션 2.4.4](#)에서 설명한 탈중앙화 랜덤 비콘(DRB)으로부터 공급된다. 위원회 크기는 수동적 적대자가 전체 스테이크의 최대 t 비율만 손상시키는 경우, 적어도 한 명의 위원회 구성원이 압도적 확률로 정직하도록 선택된다. 따라서 Mascarpone 계층은 가장 강력한 적대자를 제외한 대부분의 적대자에 대해 안전하다. 이 계층은 위원회가 새 위원회로 교체되기 전에 빠르게 위원회를 손상시킬 수 있는 적응적 적대자, 또는 위원회를 즉시 손상시킬 수 있는 매수 적대자에 의해서만 침해될 수 있다.

최상위 계층. Cocoa: CDN을 통한 Web2 수준의 성능. 우리는 중앙화된 콘텐츠 전송 네트워크(*content distribution network*, CDN)를 통해 낙관적 네트워크 조건에서 성능을 추가로 개선할 수 있다.

이 해법은 단순하다. 블록 제안자는 새로운 블록의 페이로드를 CDN에 업로드하고, 블록 페이로드가 필요한 누구나 Web2 스트리밍 서비스 이용자가 비디오를 다운로드하듯이 CDN에 요청하여 이를 획득할 수 있다.

또한 CDN은 [섹션 4.2](#)에서 설명한 쿼럼 인증서 수집과 같은 활용 사례를 위해, 블록 제안자와 다른 노드들 사이에서 메시지를 초고속으로 전달하는 채널로도 기능할 수 있다.

중앙화된 CDN은 공격자에게 쉬운 표적이 될 수 있으며, 악의적 행위자가 없더라도 간헐적인 다운타임을 겪을 수 있다. 따라서 CDN이라는 Cocoa 계층은, CDN을 사용할 수 없지만 네트워크 조건이 그 외에는 낙관적인 경우에, 그 아래의 Mascarpone 계층에 의해 충분히 보강된다.

3.1.1 통신 복잡도

Tiramisu는 프로토콜의 세 계층 전체를 합산했을 때 점근적으로 최적의 $O(|B|)$ 전체 네트워크 통신 복잡도를 달성한다.

- **기본 계층(Savoirdi).** *Savoirdi* 소거 코드 분산(*disperse*) 기법은 [부록 A.6](#)에 설명된 바와 같이 $O(|B|)$ 통신 복잡도를 달성한다.
- **중간 계층(Mascarpone).** 전체 블록 페이로드는 Mascarpone 위원회의 각 구성원에게 전송된다. 이 위원회의 크기는 상수(예: 10–200개 노드)이므로, 이 계층에서의 전체 네트워크 통신량은 $O(|B|)$ 이다.
- **최상위 계층(Cocoa).** Mascarpone과 마찬가지로 전체 블록 페이로드는 상수 개수의 노드에게만 전송된다. Cocoa에서는 페이로드가 단 하나의 노드, 즉 CDN에만 전송된다. 따라서 이 계층에서의 전체 네트워크 통신량 역시 $O(|B|)$ 이다.

3.2 구문

Tiramisu는 검증 가능한 정보 분산(verifiable information dispersal, VID)을 위한 프로토콜로 기술하는 것이 편리하다. VID 스킴은 다음과 같은 구문을 가진다.

Commit: $B \mapsto C$.

알고리즘 *Commit*은 블록 페이로드 B 를 입력으로 받아, 간결한 커밋먼트 C 를 출력으로 생성한다.

Disperse: $B \mapsto \text{cert}(C)$.

프로토콜 *Disperse*는 페이로드 송신자(payload sender)와 다수의 저장 노드(storage nodes)에 의해 실행된다. 이는 송신자로부터 블록 페이로드 B 를 입력으로 받아, 출력으로 “fail” 또는 복구 가능성 인증서(retrievability certificate) $\text{cert}(C)$ 를 송신자에게 제공한다.

Retrieve: $(C, \text{cert}(C)) \mapsto B$.

프로토콜 *Retrieve*는 클라이언트(client)와 저장 노드들에 의해 실행된다. 이는 클라이언트로부터 커밋먼트 C 와 해당 C 에 대한 복구 가능성 인증서 $\text{cert}(C)$ 를 입력으로 받아, 출력으로 “fail” 또는 블록 페이로드 B 를 클라이언트에게 제공한다.

VID 스킴이 가질 수 있는 보안 속성의 조합은 다양하다. 모든 VID 스킴은 어떤 형태로든 정확성(correctness)과 가용성(availability)을 필요로 한다. 정확성은 정직한 참여자가 정직하게 분산된 블록을 항상 복구할 수 있음을 의미하고, 가용성은 악의적인 송신자가 존재하더라도 정직한 참여자가 항상 어떤 블록이든 복구할 수 있음을 의미한다. VID 보안 속성에 대한 논의는 참고문헌 [93, 섹션 1.4]를 참조한다.

Tiramisu에 요구되는 보안 속성은 다음과 같다.

- **커밋먼트 결합성(commitment binding).** *Commit*은 블록 페이로드 B 에 대한 결정론적 결합 커밋먼트이다.
- **정확성(correctness).** 정직한 송신자가 블록 페이로드 B 에 대해 *Disperse*를 개시하면, 해당 프로토콜은 결국 종료되며 $\text{cert}(\text{Commit}(B))$ 형태의 복구 가능성 인증서를 획득한다.
- **가용성(availability).** 정직한 클라이언트가 유효한 커밋먼트 C 와 이에 대한 복구 가능성 인증서 $\text{cert}(C)$ 를 입력으로 *Retrieve*를 개시하면, 해당 프로토콜은 결국 종료되며 $\text{Commit}(B) = C$ 를 만족하는 블록 페이로드 B 를 획득한다. (이 속성은 $\text{cert}(C)$ 가 악의적인 송신자에 의해 생성된 경우에도 성립한다.)

3.3 구성

본 절에서는 Tiramisu VID 연산인 *Tiramisu.Commit*, *Tiramisu.Disperse*, *Tiramisu.Retrieve*를 설명한다. 이러한 연산들은 *Savoirdi* VID 하위 프로토콜을 블랙박스 사용하며, 그 외의 구조는 비교적 단순하다. 따라서 본 절에서는 *Savoirdi* 블랙박스를 기준으로 한 비형식적 설명만을 제공한다. *Savoirdi* 블랙박스에 대한 보다 형식적인 설명은 [부록 A](#)에 제시되어 있다(그림 6 참조).

3.3.1 Commit

Tiramisu는 *Savoirdi* 블랙박스가 요구하는 조건 외에는 *Commit*에 대해 추가적인 제약을 두지 않는다. 따라서 본 절의 목적상 *Tiramisu.Commit*은 *Savoirdi.Commit*과 동일하다고 이해해도 충분하다. 논의를 구체화하기 위해, *Tiramisu.Commit*을 블록 페이로드 B 의 해시로 생각해도 무방하다.

3.3.2 Disperse

*Tiramisu.Disperse(B)*는 블록 송신자(block sender)와 저장 노드(storage nodes) 사이에서 수행되는 단일 라운드의 상호작용 프로토콜이다. 저장 노드는 세 가지 유형으로 구분된다. 즉, 일반 저장 노드(Savoiardi), 소규모 DA 위원회 노드(Mascarpone), 그리고 콘텐츠 전송 네트워크(CDN) 노드(Cocoa)이다.

블록 송신자. 블록 페이로드 B 가 주어지면 다음을 수행한다.

1. 페이로드 커밋먼트 $C \leftarrow \text{Commit}(B)$ 를 계산한다.
2. Tiramisu의 세 계층을 모두 동시에 개시한다.
 - **Savoiardi.** 모든 저장 노드와 함께 *Savoiardi.Disperse(B)*를 실행한다.
 - **Mascarpone.** (B, C) 를 소규모 DA 위원회에 업로드한다.
 - **Cocoa.** (B, C) 를 콘텐츠 전송 네트워크(CDN)에 업로드한다.

모든 저장 노드(Savoiardi). [부록 A](#)에 명시된 *Savoiardi.Disperse*에 따라 동작한다.

소규모 DA 위원회 노드(Mascarpone). 송신자로부터 (B, C) 를 수신하면 다음을 수행한다.

1. $C = \text{Commit}(B)$ 인지 검증한다. 그렇지 않으면 중단한다.
2. C 에 대한 서명된 증명(attestation)을 블록 송신자에게 반환한다.
3. (B, C) 를 소규모 DA 위원회의 다른 노드들에게 가십(gossip) 방식으로 전파한다.

콘텐츠 전송 네트워크(CDN)(Cocoa). CDN의 역할은 단순하다. 송신자로부터 (B, C) 를 수신하면, 이를 요청하는 누구에게든 (B, C) 를 제공할 수 있도록 준비한다. CDN은 Tiramisu에 대해 어떠한 형식적 보안도 제공하지 않으므로, (B, C) 를 수신한 이후 송신자에게 응답할 필요는 없다.

물론 CDN이 송신자에게 응답해야 할 다른 이유가 있을 수 있다. 예를 들어 성능상의 이유로 CDN이 (B, C) 의 수신을 확인해 주는 것이 도움이 될 수 있다. 이러한 응답은 구현상의 세부 사항으로 간주한다.

블록 송신자. 저장 노드들로부터의 응답을 대기한 후, 가용성 인증서를 구성한다.

1. 동시에 다음을 수행한다.
 - **Mascarpone.** 소규모 DA 위원회 노드의 쿼럼으로부터 C 에 대한 증명을 대기한다. C 에 대한 Mascarpone 인증서는 이러한 증명들의 집계로 구성된다.
 - **Savoiardi.** *Savoiardi.Disperse*에 따라 저장 노드 쿼럼으로부터 C 에 대한 증명을 대기한다. C 에 대한 Savoiardi 인증서는 이러한 증명들의 집계로 구성된다.
2. C 에 대한 Tiramisu 인증서 $\text{cert}(C)$ 는 C 에 대한 Mascarpone 인증서와 Savoiardi 인증서를 모두 포함한다.

3.3.3 Retrieve

*Tiramisu.Retrieve(C, cert(C))*는 클라이언트와 저장 노드 사이에서 수행되는 단일 라운드의 상호작용 프로토콜이다. 클라이언트는 다음을 수행한다.

1. CDN(Cocoa)에 블록 페이로드를 요청한다. CDN이 $Tiramisu.Commit(B) = C$ 를 만족하는 페이로드 B 를 반환하면, B 를 반환한다.
2. 소규모 DA 위원회(Mascarpone)에 블록 페이로드를 요청한다. 소규모 DA 위원회가 $Tiramisu.Commit(B) = C$ 를 만족하는 페이로드 B 를 반환하면, B 를 반환한다.
3. 저장 노드(Savoiardi)와 함께 $Savoiardi.Retrieve(B)$ 를 실행한다.

3.4 HotShot에서의 Tiramisu 활용

HotShot에서 Tiramisu를 사용하는 방식은 단순하다. 각 뷰(view)마다 HotShot 합의 리더는 단일 블록 B 를 최종화하려고 시도한다. 블록은 데이터 가용성 인증서(certificate of availability)가 수반되지 않으면 최종화될 수 없다. 이러한 인증서는 리더가 저장 노드들과 함께 $Tiramisu.Disperse(B)$ 를 실행함으로써 획득한다. 그 결과로 생성된 블록 커밋먼트 $C = Tiramisu.Commit(B)$ 와 이에 대응하는 인증서 $cert(C)$ 가 온체인에 포함된다. 앞서 논의했듯이, 페이로드 B 는 온체인에 담기에는 너무 크지만, $Tiramisu.Disperse$ 의 보안 속성에 의해 B 는 이를 필요로 하는 누구에게든 가용함이 보장된다.

본 절에서는 HotShot이 Tiramisu를 사용하는 방식에 대한 몇 가지 세부 사항을 명확히 한다.

3.4.1 활성화

$Tiramisu.Disperse$ 는 다음 두 조건이 모두 충족될 때에만 성공한다는 점을 상기하자.

1. 블록 제안자가 소규모 DA 위원회로부터 충분한 수의 서명을 수집하여 Mascarpone 인증서를 생성한 경우, 그리고
2. $Savoiardi.Disperse$ 가 Savoiardi 인증서를 성공적으로 생성한 경우이다.

Savoiardi는 Savoiardi 스킴의 속성과 HotShot 위협 모델에 의해 항상 성공이 보장된다. 그러나 Mascarpone은 충분히 강력한 적대자가 낙관적 DA 위원회를 손상시켜 인증서를 생성할 수 없게 만드는 경우 실패할 수 있다. 이 경우 해당 HotShot 뷰에서는 어떠한 블록도 최종화될 수 없으며, 이는 활성화의 일시적인 실패를 의미한다.

낙관적 DA 위원회는 무작위로 선택되며 자주 교체되므로, 적대자가 이 위원회를 지속적으로 실패하게 만들 수 있는 유일한 방법은 비용이 큰 적응적 공격 또는 매수 공격을 수행하는 것이다. [섹션 2.2](#)에서 논의했듯이, 이러한 공격은 적대자의 예산을 소진시키며, 예산이 소진된 이후에는 활성화는 즉시 회복된다.

이러한 공격을 관점에서 보면, 적대자는 동일한 활성화 실패를 훨씬 더 낮은 비용으로 해당 뷰의 리더만 공격함으로써 유발할 수도 있다. 따라서 이 공격 경로는 HotShot의 활성화 특성을 추가적으로 약화시키지 않는다.

3.4.2 고비용 데이터 복구 강제

어떤 이유로 인해 Tiramisu의 Cocoa 계층(CDN)이 작동하지 않는다고 가정하자. CDN이 정상적으로 작동하는 경우에는 페이로드 데이터를 복구하는 데 문제가 발생하지 않기 때문이다. Tiramisu의 모든 계층 중에서 Savoiardi만이 적응적 적대자 또는 매수 적대자에 대해서도 안전하다. 따라서 적대자가 네트워크가 비용이 큰 Savoiardi 스킴에 의존하여 블록 페이로드를 복구하도록 강제하는 공격이 존재할 것으로 예상할 수 있다.

이러한 공격의 유일한 형태는, 소규모 DA 위원회를 손상시켜 인증서는 생성하지만 요청 시 페이로드를 제공하지 않거나 제공할 수 없게 만드는 것이다. [섹션 3.4.1](#)에서 관찰했듯이, 활성화에 대한 공격은 적대자의 예산을 소진시킨다. 동일한 원리가 고비용 데이터 복구를 강제하는 공격에도 적용된다.

3.4.3 최적화: Savoirdi 인증서 불필요

설명의 편의를 위해 *Tiramisu.Disperse*가 생성하는 인증서 $cert(C)$ 에 Mascarpone(소규모 DA 위원회) 인증서와 Savoirdi 인증서를 모두 포함한다고 가정하는 것이 편리하다. 그러나 HotShot에서는 Savoirdi 인증서가 엄밀히 말해 필요하지 않다. 대부분 또는 모든 HotShot 노드는 Savoirdi 저장 노드이기도 하므로, 이러한 노드들은 (i) Mascarpone 인증서와 (ii) 자신이 보유한 인코딩된 블록에 대한 Savoirdi 조각만을 확인한 상태에서도 후보 블록에 대해 HotShot 투표를 수행할 수 있다. 이 방식에서는 HotShot 투표가 통과되었다면, Savoirdi 저장 노드의 쿼럼이 각자 자신의 Savoirdi 조각을 이미 확인했음이 자동으로 성립하므로, 이러한 증명들을 별도의 인증서로 집계할 필요가 없다.

3.4.4 페이로드 전파, 복구, 보존에 대한 정책

HotShot은 Tiramisu를 사용하여 블록이 최종화되는 시점에 *일부 참여자*가 해당 블록의 전체 페이로드를 복구할 수 있음을 보장할 뿐이다.

왜 일부 참여자만이면 충분한가? 이 질문은 *전파(dissemination)*와 *복구(retrieval)*에 관한 것이다. 즉, 누가 전체 페이로드를 *반드시* 다운로드해야 하며, 누가 전체 페이로드를 다운로드할 수 *있는지*에 대한 문제이다.

HotShot은 자신이 시퀀싱하는 트랜잭션을 실행하지 않으므로, 네트워크 상의 대부분의 노드는 모든 블록에 대해 전체 페이로드를 다운로드할 필요가 없다. 전체 페이로드의 복구는 DA 위원회(Mascarpone)나 CDN(Cocoa)에 속한 소수의 강력한 네트워크 노드에게만 요구된다.

이러한 느슨한 요구사항의 긍정적인 결과는, HotShot이 보안을 달성하기 위해 느린 노드들을 고려할 필요가 없다는 점이다. 실제로 우리는 대부분의 HotShot 노드가 대부분의 블록에 대해 페이로드를 다운로드하지 않을 것으로 예상한다.

그렇다면 누가 전체 페이로드를 다운로드할 수 있는가? 이 질문은, 페이로드 복구를 개시할 수 있는 주체에 제한이 없다면, 악의적인 행위자가 데이터 요청을 대량으로 보내 네트워크를 마비시키는 서비스 거부(DoS) 공격이 가능하지 않은지에 대한 우려와 관련된다. 이 문제는 HotShot에만 국한된 것이 아니라, 모든 퍼미션리스 시스템과 모든 퍼블릭 블록체인에 공통적으로 해당된다. 이 문제에 대한 해결책은 본 문서의 범위를 벗어난다. 여기서는 각 프로토콜이 자신의 요구에 맞게 서로 다른 복구 정책을 채택한다는 점을 관찰하는 것으로 충분하다. 일반적인 접근 방식은, 각 네트워크 노드가 초기에는 누구에게나 데이터를 제공하는 관대한 정책을 취하되, 남용의 징후를 감지하면 일방적으로 요청 처리를 중단할 수 있도록 하는 것이다.

마지막으로, 왜 블록이 최종화되는 시점에만 가용성을 보장하는가? 이 질문은 *보존(retention)*에 관한 것이다. 과거에 최종화된 페이로드 데이터의 가용성은, 새로운 후보 블록에 대한 현재 페이로드 데이터의 가용성과는 별개의 문제이다. 명시적인 보존 정책은 본 문서의 범위를 벗어난다. 여기서는 과거 데이터의 복구가 아카이빙 서비스나 탈중앙화 데이터 저장 플랫폼과 같은 다른 행위자에 의해 제공될 수 있음을 관찰하는 것으로 충분하다.

4 CDN을 활용한 반응성 극대화

본 절에서는 HotShot 합의(섹션 2)와 Tiramisu 데이터 가용성 솔루션(섹션 3) 전반에 걸쳐 적용되는 핵심 기법인 *콘텐츠 전송 네트워크(Content Delivery Network, CDN)*와 *피어 투 피어(peer-to-peer, P2P)*를 결합한 하이브리드 네트워킹 계층을 설명한다. CDN의 활용은 단순한 엔지니어링 최적화처럼 보일 수 있으나, 우리는 이것이 본 프로토콜과 매우 높은 정합성을 가지며 실제 성능을 크게 증폭시킬 수 있음을 보일 것이다. 본 절의 나머지 부분에서는 먼저 블록체인에서 널리 사용되는 기존 네트워킹 선택지와 그 한계를 설명하고, 이를 통해 “CDN+P2P” 설계의 동기를 제시한다. 이후 CDN의 기본 개념과 핵심 속성을 정리하고, 마지막으로 왜 CDN이 반응형 프로토콜에 이상적으로 부합하는지, 그리고 이를 어떻게 구체적으로 활용하는지를 설명한다.

4.1 CDN + P2P: 하이브리드 블록체인 네트워킹

네트워킹 계층은 블록체인의 성능과 보안에 있어 핵심 구성 요소이다. 대부분의 블록체인 프로토콜은 탈중앙화된 P2P 네트워크 위에서 실행되며 [46], 이 네트워크는 서로 다른 연산 능력, 저장 용량, 대역폭을 가진 노드들로 구성되고, 노드들은 언제든지 참여하거나 이탈할 수 있다. 이러한 이질성, 다중 홉 라우팅, 그리고 높은 이탈률(churn rate)은 지연 시간을 유발하여 트랜잭션과 블록의 전파를 느리게 만든다. 예를 들어 비트코인 P2P 네트워크는 대역폭 효율성이 낮다는 점이 관찰되었으며 [91], 처리량을 높이려는 시도 과정에서 안전성이 훼손될 수 있음이 보고되었다 [43]. 한편 DiemBFT, Aptos [10]와 같은 일부 블록체인은 각 합의 노드가 전체 멤버십 뷰를 유지하고 모든 다른 노드와 직접 연결되는 풀 메쉬(full-mesh) 네트워크를 구성하여 낮은 지연 시간을 달성한다. 그러나 이러한 방식은 연결 수가 이차적으로 증가하므로, 훨씬 더 큰 네트워크 규모로 확장하는 것은 비현실적이다.

반면 Web2 비즈니스에서 사용되는 현대적인 네트워킹은 CDN 인프라를 활용하여 수백만 명의 사용자에게 대량의 데이터(예: 비디오)를 1초 미만의 지연 시간으로 제공하는 것을 목표로 한다. CDN은 지리적으로 분산된 클라이언트에게 정보를 효율적으로 라우팅할 수 있는 상호 연결된 서버 집합으로 구성되며, 강력한 하드웨어, 고대역폭 연결, 그리고 고급 캐싱 기법을 활용한다 [102]. CDN은 견고하고 거의 최적에 가까운 서비스를 제공하지만, 본질적으로 중앙화되어 있으므로 블록체인 네트워킹을 위한 단독 대안으로는 적합하지 않다.

CDN의 내부 구조와 엔지니어링 세부 사항은 복잡하지만, 본 연구의 목적상 단순화된 추상 모델로 충분하다. 우리는 CDN을 모든 노드 사이에 위치하여 빠른 메시지 전달과 위임된 연산을 제공하는 강력하지만 중앙화된 허브로 간주한다. 위임된 연산은 CDN 서버의 강력한 하드웨어 덕분에 가능하며, 특히 리더 기반 프로토콜에서 서명 집계와 같이 계산 집약적인 리더의 작업 일부를 위임하는 데 유용하다.

네트워크 계층에서의 “블록체인 트릴레마” [26]를 극복하기 위해, 우리는 CDN과 P2P 네트워크를 결합한 하이브리드 인프라를 채택한다. 핵심 아이디어는 네트워크 조건이 양호한 대부분의 시간 동안 CDN에 의존하여 최상의 처리량과 지연 시간을 확보하고, 드물게 CDN 장애나 심각한 성능 저하가 발생하는 경우에는 성능은 떨어지지만 더 견고한 P2P 네트워크로 폴백하여 CDN이 복구될 때까지 활성(liveness)을 보장하는 것이다. 이 하이브리드 접근 방식은 다음과 같은 속성들로 요약되는 양쪽의 장점을 모두 제공한다.

- **최적의 낙관적 성능(고처리량, 저지연).**

CDN이 정상 동작하는 낙관적 네트워크 조건에서는 Web2 수준의 최적 성능을 누릴 수 있다. 낮은 지연 시간은 지리적으로 분산된 PoP(Point of Presence) [11]의 배치를 통해 클라이언트와 데이터 간의 라우팅 거리가 크게 단축되는 데서 기인한다. CDN 솔루션에 일반적으로 적용되는 최적화된 캐싱 메커니즘은 동적 콘텐츠에 대해서도 높은 캐시 적중률(cache hit ratio)을 달성하여 평균 전달 시간을 더욱 줄인다. 높은 처리량은 소프트웨어 차원의 지능적 부하 분산과 하드웨어 차원의 고대역폭에서 비롯된다.

- **보안(안전성과 활성).**

현대적인 CDN은 DDoS 공격, 트래픽 하이재킹, 스푸핑 공격, 중간자 공격에 대한 다양한 완화 기법을 내장하고 있으며, 따라서 P2P 네트워크에 비해 일식(eclipse) 공격 [63]에 덜 취약하고 우수한 보호를 제공한다. CDN이 실패하더라도, 노드들은 견고한 P2P 네트워크로 자연스럽게 폴백하여 계속해서 진행을 이어가며, 네트워킹 관점에서는 오늘날 대부분의 블록체인과 유사한 수준의 성능을 유지한다.

- **검열 저항성.**

우리는 CDN과 P2P를 통한 전파가 *병렬적으로* 동작함을 강조한다. CDN에 게시된 모든 데이터는 P2P를 통해서도 독립적으로 전파될 수 있다. 이는 패킷을 드롭하는 손상된 CDN이 존재하더라도, 더 느리지만 견고한 P2P 경로를 통해 결국 전달이 이루어지므로 검열 저항성을 보장하는 데 필수적이다.

- **확장성 저해 없음.**

성숙한 CDN 클러스터는 성능이나 보안을 저해하지 않으면서 수평 확장을 통해 인터넷 규모의 트래픽을 처리할 수 있다.

- **반응성과 선형성의 효용 극대화.**

이는 CDN을 채택한 가장 핵심적이고 설득력 있는 이유이다. 반응형 프로토콜은 낙관적 경우 사전 정의된 타임아웃을 기다리지 않고 네트워크 속도에 맞추어 진행할 수 있으므로, 매우 빠른 하부 네트워크로부터 최대의 이점을 얻는다. 일반적으로 더 빠른 네트워크는 어떤 합의 프로토콜에도 유리하지만, 예를 들어 Tendermint [23]와 같이 라운드마다 고정된 뷰 전환 타임아웃을 가지는 프로토콜에서는 이러한 이점이 상쇄된다. 또한 HotStuff-2 및 그 위에 구축된 HotShot 합의의 선형 통신 복잡도는, P2P 네트워크를 사용하거나 네트워크 규모가 작은 경우(예: 수백 노드)에는 그 중요성이 상대적으로 낮다. P2P 네트워크에서는 목적지에 도달하기까지의 기대 홉 수로 인해 통신에 로그 비용이 발생하고, 플러딩 기반 브로드캐스트는 중복 메시지로 인해 유효 처리량을 저하시킨다. 반면 CDN은 기대 값 기준으로 상수 홉 수(또는 P2P보다 더 적은 홉 수)로 데이터를 전달하므로, 선형성의 장점이 실제 환경에서 그대로 발현된다. 그 결과 구체적인 상수 계수가 작은 상태에서 실질 복잡도는 선형으로 유지되며, 수만 개의 노드로 확장될수록 그 절감 효과는 더욱 두드러진다.

마지막으로, bloXroute [71]는 일반적인 블록체인을 위한 네트워킹 계층 개선이라는 점에서 관련 선행 연구이다. 우리는 중앙화된(잠재적으로 분산된) 인프라를 성능을 위해 사용하고, P2P 네트워크를 복원력과 보안을 위한 폴백으로 사용하는 하이브리드 접근 방식에서 유사성을 공유한다. bloXroute의 릴레이 네트워크는 현대 인터넷 인프라의 다양한 최적화 기법을 통합한 우리의 CDN 계층에 해당하며, 양쪽 설계 모두에서 P2P 네트워크는 견고한 폴백으로 존재한다. 그러나 우리의 네트워킹은 특정 합의 프로토콜을 염두에 두고 설계된 반면, bloXroute는 블록체인에 독립적인 네트워크 계층을 목표로 한다. 이로 인해 우리는 단순한 메시지 전달을 넘어 서명 집계와 같은 연산까지 CDN에 맞게 추가적으로 맞춤화할 수 있는 반면, bloXroute의 CDN에 해당하는 네트워크에서는 전송되는 콘텐츠가 암호화되어 있어 연산이 불가능하다. 또한 bloXroute는 CDN에 의한 송신자 기반 및 콘텐츠 기반 검열을 완화하고 더 강한 중립성 보장을 제공하는 대신, 모든 패킷에 대해 약간의 지연 시간 증가를 감수한다. 반면 우리는 낙관적 경우에만 CDN을 사용하므로 이러한 강한 속성을 요구하지 않는다. 향후 연구 과제로서, bloXroute를 사용할 경우의 정성적 비교 및

시스템 평가를 보다 심도 있게 수행하는 것이 남아 있다.

4.2 에스프레소 시퀀서와 CDN의 통합

섹션 4.1에서 CDN의 속성과 HotShot 합의와의 시너지를 설명했으므로, 본 절에서는 설계 전반에 걸친 구체적인 통합 방식을 상세히 설명한다.

P2P 네트워크. 우리의 P2P 네트워크는 GossipSub [115]의 한 구현이다. GossipSub는 대역폭 사용을 최적화하면서도 시빌(Sybil), 검열, 일식(eclipse) 공격과 같은 일반적인 적대적 행위를 완화할 수 있어 블록체인 P2P 프로토콜에 적합하다. 고수준에서 GossipSub는 네트워크를 여러 개의 메시(mesh) 집합으로 구성하며, 각 메시 내에서는 노드들이 서로 직접 연결된다. 메시 간 또는 특정 메시에도 직접 속하지 않는 노드와의 통신은 가십 방식으로 이루어지며, 메시지는 무작위로 선택된 피어의 부분집합에게 전송된다. 또한 각 노드는 악의적 행위를 탐지하고 이에 따라 네트워크를 재구성하기 위해 피어에 대한 점수 함수(score function)를 유지한다. 우리의 프로토콜은 PoS 기반이면서 리더 기반이므로, GossipSub 네트워킹 인프라를 다음과 같이 확장한다. (1) 피어에 대한 분산 해시 테이블(Distributed Hash Table) [12] 정보를 인증하는데, 특히 노드의 IP 주소는 해당 스테이킹 키로 서명된다. (2) 노드(특히 리더)에 해당하는 IP 주소가 메시 전반에 걸쳐 고르게 분산되도록 보장한다.

CDN. 우리의 CDN은 현재 하나의 마스터 서버와 여러 개의 PoP(Point of Presence) 서버로 구성되어 있으며, 마스터 서버의 역할은 다음과 같다.

- 쿼럼 인증서 구성과 같은 계산 집약적 작업에서 리더를 지원한다(부록 B.3.3 참조).
- 리더의 제안과 같은 브로드캐스트 메시지 또는 부분 서명이나 투표와 같이 리더에게 되돌아가는 점대점 메시지를 전달한다.
- 지리적 위치 및 기타 매개변수에 따라 노드 클러스터에 PoP를 할당한다.

브로드캐스트 메시지는 마스터 서버로 전송된 후 모든 노드로 전달된다. 노드들은 초기 연결 시 전용 네이밍 서버에 접속하여 자신에게 할당된 PoP를 획득하며, 이후에는 주로 해당 PoP와 상호작용한다.

메시지 전달. CDN이 정상적으로 동작하는 경우 모든 메시지는 CDN을 통해 전달되며, 송신자가 추가적인 검열 저항성과 중복성을 원할 경우 선택적으로 P2P로도 전송될 수 있다. P2P 네트워크로의 전환은 명시적인 합의 없이 이루어지며, CDN 장애를 감지한 노드들이 점차 모든 메시지에 대해 P2P를 주 경로로 사용하기 시작하면서 자연스럽게 발생한다. 이 두 네트워킹 옵션은 서로 다른 처리량-지연 특성을 가지며, 어느 하나 또는 둘 모두의 실패나 성능 저하는 안전성을 위반하지 않는다. 또한 CDN의 실패는 우리의 위협 모델 하에서 활성(liveness)을 일시적으로 정체시키는 데 그친다.

5 예비 평가

본 절에서는 현재 구현에 대한 (매우) 예비적인 평가를 제시한다.

구현 세부 사항.

우리는 HotShot과 Tiramisu의 한 버전에 대한 오픈소스 구현을 Rust로 제공하고 있다 [13]. 현재의 HotShot 구현은 HotStuff 프로토콜을 변형한 것으로, 크기가 n 인 고정된 허가형(permissioned) 시퀀서 노드 집합을 가정하며, 이들의 공개 키는 서로에게 알려져 있다. 노드들은 정상 상태 프로토콜을 실행하며, 리더는 라운드 로빈 방식으로 전환되고 동기화가 깨질 경우 뷰 동기화를 수행한다. 특히 현재 구현은 이더리움으로부터 스테이킹 키를 가져오지 않으며, 에폭을 구현하지 않고 재구성을 수행하지도 않는다. 쿼럼은 스테이크 가중치가 아닌 $\geq \frac{2}{3}n$ 크기의 임계치로 정의되며, 현재는 서명 집계를 수행하지 않는다. Tiramisu 구현은 Cocoa와 Mascarpone 계층으로 구성되어 있으나 Savoiardi는 포함하지 않는다. Mascarpone 계층에 참여하는 노드 집합은 고정되어 있으며 모든 노드에게 알려져 있다. 네트워킹 측면에서는 현재 모든 통신이 낙관적 CDN 계층을 통해 라우팅되며, P2P 풀백 네트워크는 존재하지 않는다. 또한 현재 CDN은 계산 집약적 작업에는 사용되지 않는다.

실험 환경.

모든 실험은 Amazon AWS의 us-east-2 리전에서 수행되었다 [11]. 실험에서는 HotShot 메시지와 Tiramisu 메시지로 각각 하나씩, Nginx [95]로 프런트된 두 개의 CDN 인스턴스를 사용하였다. 각 CDN은 m6a.xlarge EC2 인스턴스로, 4 vCPU, 16GB 메모리, 최대 12.5Gbit/s 대역폭을 제공한다. 노드들은 100ms 간격으로 HTTP/1.1을 사용하여 CDN을 폴링한다. HotShot 및 Tiramisu 네트워크를 구성하는 노드들은 각각 2 vCPU와 16GB 메모리를 갖는 ECS Fargate 태스크로 실행된다. AWS는 ECS Fargate 태스크에 대한 대역폭 사양을 공개하지 않는다. 이 노드 하드웨어는 이더리움 노드의 요구사항 [50]보다 더 강력하지 않다. ECS 태스크는 us-east-2 리전의 3개 가용 영역에 균등하게 분산 배치되었다.

방법론.

각 실행(run)마다 10개의 노드를 클라이언트로 선택하여, Rust의 별도 비동기 태스크로 사전에 생성된 트랜잭션을 다른 모든 노드에 제출한다. 실행은 네트워크의 모든 노드가 100개의 블록을 커밋할 때까지 지속된다. 평균 뷰 시간은 전체 실행 시간과 실행에 필요한 뷰 수의 비율로 계산한다. 보고되는 지연 시간은 평균 뷰 시간의 세 배이며, 처리량은 모든 데이터 블록 크기의 합을 실행 시간으로 나눈 값이다. 각 데이터 포인트마다 이러한 실행을 세 번 수행하고, 세 실행 및 커밋된 블록 전반에 걸친 평균값을 보고한다.

서로 다른 구성에서 시스템의 최대 처리량을 찾기 위해, 트랜잭션 제출 간격은 고정된 채 네트워크에 제출되는 트랜잭션의 크기를 변화시켰다. 리더는 자신의 큐에 있는 트랜잭션을 집계하여 다음 블록을 생성한다. 실험적으로 작은 트랜잭션 크기에서 시작하여, 네트워크 처리량이 계속 증가하는 동안 트랜잭션 크기를 두 배씩 증가시켰다. 처리량이 감소하기 시작하는 변곡점에 도달하면, 변곡점과 그 양쪽에서 가장 가까운 지점 사이의 중간값에 해당하는 트랜잭션 크기를 선택하였다. 이후 변곡점 기준으로 양쪽에서 이진 탐색에 가까운 방식으로 이 과정을 재귀적으로 반복하였으며, 트랜잭션 크기의 증가 또는 감소가 원래 변곡점 트랜잭션 크기 대비 두 자릿수(order of magnitude) 미만이 될 때까지 수행하였다. 그 결과 새로운 변곡점과 최대 처리량을 도출하였다.

평가 결과.

주어진 실험 환경에서 우리는 두 가지 핵심 매개변수, 즉 노드 수 n 과 Mascarpone 위원회 크기를 변화시키며 시스템 성능을 평가하였다. 그림 7은 Mascarpone 위원회 크기를 고정된 상태에서 노드 수를 변화시켰을 때의 처리량과 지연 시간을 보여준다. 관찰 결과, 평균 처리량은 일관되게 약 25–30MB/초 범위를 유

지하며, 평균 지연 시간은 약 2.9–3.5초 범위에서 안정적으로 유지된다. 즉, 네트워크 내 노드 수가 증가하더라도 성능은 저하되지 않는다. 흥미롭게도 10개 노드로 구성된 네트워크의 처리량은 노드 수가 100을 초과하는 경우보다 현저히 낮다. 이는 현재 구현의 특성에 따른 결과로, 고정된 10개의 노드가 동시에 트랜잭션 생성, 데이터 가용성 역할, 합의 역할을 모두 수행하기 때문이다. n 이 증가하면 합의에 참여하는 대부분의 노드가 나머지 두 작업에는 관여하지 않게 된다.

그림 8은 노드 수를 고정한 상태에서 Mascarpone 위원회 크기를 변화시켰을 때의 성능을 제시한다. 현재 구현에서는 위원회 크기가 증가할수록 처리량이 비례적으로 감소하는 것을 확인할 수 있다. 이는 예상대로 Mascarpone 계층에서의 데이터 전파가 현재 병목 지점임을 강하게 시사한다. 위원회 크기가 증가함에 따라 지연 시간이 크게 변동하는데, 이는 처리량을 최적화하기 위해 실행마다 블록 크기를 조정한 데서 비롯된 인공적인 효과이다. 이러한 문제들은 이미 인지하고 있으며, 이후 구현 단계에서 해결할 예정이다.

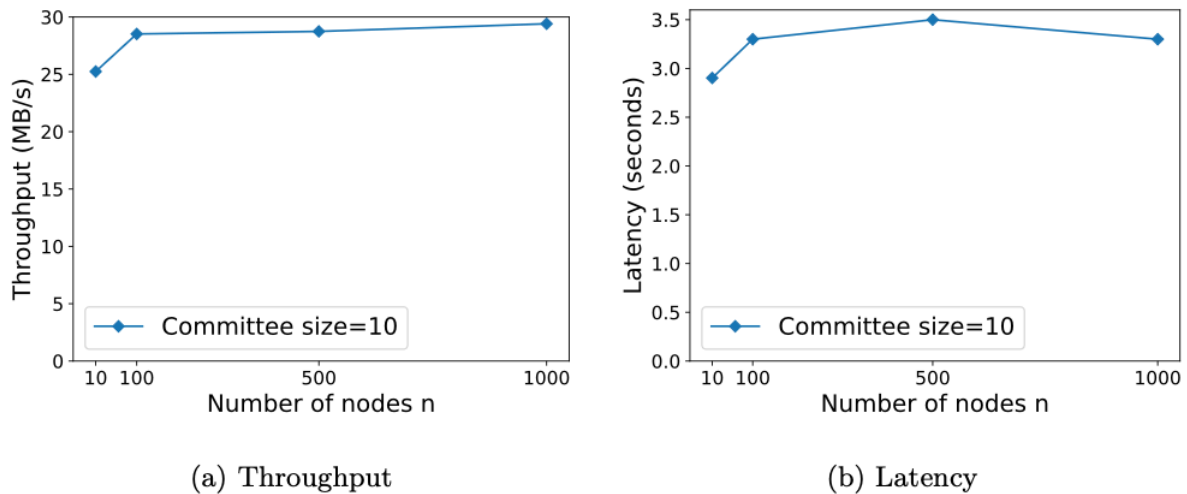


Figure 7: Throughput and latency at varying n and a fixed Mascarpone committee

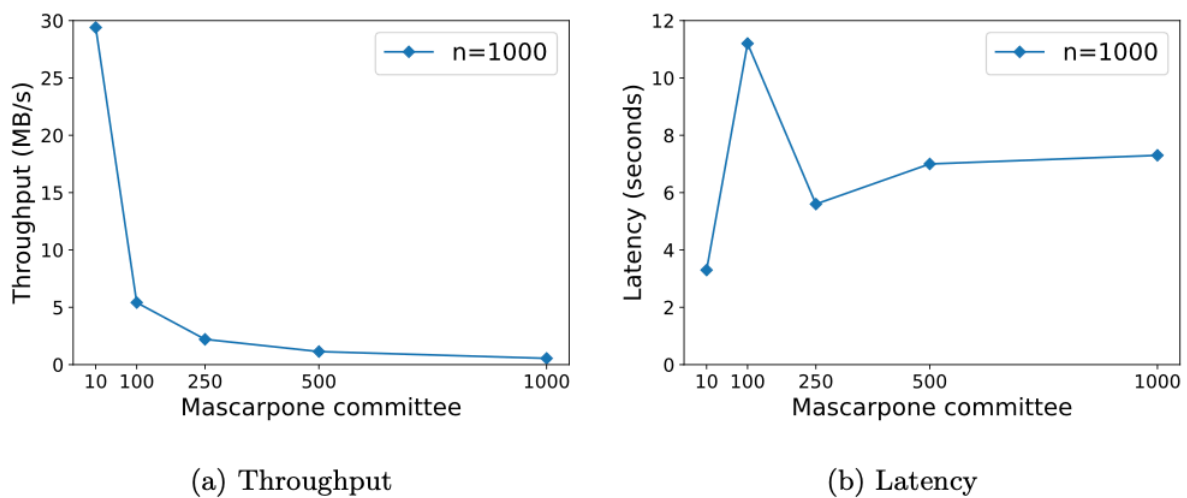


Figure 8: Throughput and latency at varying Mascarpone committee sizes and a fixed n

참고 문헌

- [1] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. *Synchronous byzantine agreement with expected $o(1)$ rounds, expected $o(n^2)$ communication, and optimal resilience*. Cryptology ePrint Archive, Paper 2018/1028, 2018. <https://eprint.iacr.org/2018/1028>
- [2] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. *Synchronous byzantine agreement with expected $o(1)$ rounds, expected communication, and optimal resilience*. In *International Conference on Financial Cryptography and Data Security*, pages 320–334. Springer, 2019.
- [3] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, and Ling Ren. *Dfinity consensus, explored*. Cryptology ePrint Archive, 2018.
- [4] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. *Sync hotstuff: Simple and practical synchronous state machine replication*. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 106–118. IEEE, 2020.
- [5] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. *Asymptotically optimal validated asynchronous byzantine agreement*. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.
- [6] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. *Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities*. arXiv preprint arXiv:1809.09044, 160, 2018.
- [7] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. *Fraud and data availability proofs: Detecting invalid blocks in light clients*. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 279–298. Springer, 2021.
- [8] Andreea B Alexandru, Erica Blum, Jonathan Katz, and Julian Loss. *State machine replication under changing network conditions*. Cryptology ePrint Archive, 2022.
- [9] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. *Asynchronous verifiable information dispersal with near-optimal communication*, 2022. <https://eprint.iacr.org/2022/775>
- [10] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. *Succinct erasure coding proof systems*, 2021. <https://eprint.iacr.org/2021/1500>

- [11] AWS. *General purpose network performance*.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/general-purpose-instances.html#general-purpose-network-performance>, Accessed: 2023-08-11.
- [12] Vivek Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. *Proof-of-stake longest chain protocols: Security vs predictability*. In *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, pages 29–42, 2022.
- [13] Mihir Bellare and Gregory Neven. *Multi-signatures in the plain public-key model and a general forking lemma*. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399, 2006.
- [14] Alexandra Boldyreva. *Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme*. In *Public Key Cryptography—PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography Miami, FL, USA, January 6–8, 2003 Proceedings 6*, pages 31–46. Springer, 2002.
- [15] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. *Verifiable delay functions*. In *Advances in Cryptology—CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I*, pages 757–788. Springer, 2018.
- [16] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. *Aggregate and verifiably encrypted signatures from bilinear maps*. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.
- [17] Dan Boneh and Chelsea Komlo. *Threshold signatures with private accountability*. In *Advances in Cryptology—CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV*, pages 551–581. Springer, 2022.
- [18] Dan Boneh, Ben Lynn, and Hovav Shacham. *Short signatures from the weil pairing*. In *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 7*, pages 514–532. Springer, 2001.
- [19] Joseph Bonneau. *Why buy when you can rent? bribery attacks on bitcoin-style consensus*. In *International Conference on Financial Cryptography and Data Security*, pages 19–26, 2016.

- [20] Silvia Bonomi, Antonella Del Pozzo, Maria Potop-Butucaru, and Sébastien Tixeuil. *Optimal mobile byzantine fault tolerant distributed storage*. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 269–278, 2016.
- [21] Gabriel Bracha. *An asynchronous $[(n-1)/3]$ -resilient consensus protocol*. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162, 1984.
- [22] Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. *Formal barriers to longest-chain proof-of-stake protocols*. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, page 459–473. Association for Computing Machinery, 2019.
- [23] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, University of Guelph, 2016.
- [24] Ethan Buchman, Jae Kwon, and Zarko Milosevic. *The latest gossip on bft consensus*. *arXiv preprint arXiv:1807.04938*, 2018.
- [25] Vitalik Buterin. *Reed-solomon erasure code recovery in $n \log^2 n$ time with ffts*, August 2018. <https://ethresear.ch/t/reed-solomon-erasure-code-recovery-in-n-log-2-n-time-with-ffts>
- [26] Vitalik Buterin. *Why sharding is great: demystifying the technical properties*. <https://vitalik.ca/general/2021/04/07/sharding.html>, 2021.
- [27] Vitalik Buterin et al. *A next-generation smart contract and decentralized application platform*. *white paper*, 3(37):2–1, 2014.
- [28] Christian Cachin and Stefano Tessaro. *Asynchronous verifiable information dispersal*. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS’05)*, pages 191–201. IEEE, 2005.
- [29] Wei Cai, Zehua Wang, Jason B Ernst, Zhen Hong, Chen Feng, and Victor CM Leung. *Decentralized applications: The blockchain-empowered software system*. *IEEE access*, 6:53019–53033, 2018.
- [30] Ran Canetti and Tal Rabin. *Fast asynchronous byzantine agreement with optimal resilience*. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 42–51, 1993.
- [31] Miguel Castro and Barbara Liskov. *Practical byzantine fault tolerance*. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, pages 173–186. USENIX Association, 1999.

- [32] Benjamin Y Chan and Rafael Pass. *Simplex consensus: A simple and fast consensus protocol*. Cryptology ePrint Archive, 2023.
- [33] TH Hubert Chan, Rafael Pass, and Elaine Shi. *Pala: A simple partially synchronous blockchain*. Cryptology ePrint Archive, 2018.
- [34] TH Hubert Chan, Rafael Pass, and Elaine Shi. *Pili: An extremely simple synchronous blockchain*. Cryptology ePrint Archive, 2018.
- [35] Jon Charbonneau. *Rollups aren't real*.
<https://joncharbonneau.substack.com/p/rollups-arent-real>, Accessed: 2023-05-11.
- [36] Yan Chen and Cristiano Bellavitis. *Blockchain disruption and decentralized finance: The rise of decentralized business models*. *Journal of Business Venturing Insights*, 13:e00151, 2020.
- [37] Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, Matteo Monti, and Manuel Vidigueira. *Every bit counts in consensus*. *arXiv preprint arXiv:2306.00431*, 2023.
- [38] Phil Daian, Rafael Pass, and Elaine Shi. *Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake*. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 23–41. Springer, 2019.
- [39] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. *Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability*. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. IEEE, 2020.
- [40] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. *Narwhal and tusk: a dag-based mempool and efficient bft consensus*. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.
- [41] Sourav Das, Zhuolun Xiang, and Ling Ren. *Asynchronous data dissemination and its applications*. Cryptology ePrint Archive, Paper 2021/777, 2021.
<https://eprint.iacr.org/2021/777>
- [42] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. *Practical asynchronous distributed key generation*. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2518–2534. IEEE, 2022.
- [43] Christian Decker and Roger Wattenhofer. *Information propagation in the bitcoin network*. In *IEEE P2P 2013 Proceedings*, pages 1–10. IEEE, 2013.

- [44] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. *A survey on long-range attacks for proof of stake protocols*. *IEEE access*, 7:28712–28725, 2019.
- [45] Danny Dolev and H. Raymond Strong. *Authenticated algorithms for byzantine agreement*. *SIAM Journal on Computing*, 12(4):656–666, 1983.
- [46] Maya Dotan, Yvonne–Anne Pignolet, Stefan Schmid, Saar Tochner, and Aviv Zohar. *Survey on blockchain networking: Context, state-of-the-art, challenges*. *ACM Computing Surveys (CSUR)*, 54(5):1–34, 2021.
- [47] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. *Consensus in the presence of partial synchrony*. *J. ACM*, 35:288–323, 1988.
- [48] Ben Edgington. *Upgrading ethereum: A technical handbook on ethereum’s move to proof of stake and beyond*, 2022.
- [49] EigenLabs. *Eigenlayer: shared security to hyperscale ethereum*. <https://www.eigenlayer.xyz/>, Accessed: 2023-07-18.
- [50] Ethereum.org. *Run a node*. <https://ethereum.org/en/developers/docs/nodes-and-clients/run-a-node/#requirements>, Accessed: 2023-08-08.
- [51] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. *{Bitcoin–NG}: A scalable blockchain protocol*. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 45–59, 2016.
- [52] Ittay Eyal and Emin Gün Sirer. *Majority is not enough: Bitcoin mining is vulnerable*. *Communications of the ACM*, 61(7):95–102, 2018.
- [53] Dankrad Feist and Dmitry Khovratovich. *Fast amortized kzg proofs*, 2023. <https://eprint.iacr.org/2023/033>.
- [54] Orr Fischer and Mevar Parter. *Distributed congest algorithms against mobile adversaries*. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 262–273, 2023.
- [55] Flashbots. *The future of mev is suave*. <https://writings.flashbots.net/the-future-of-mev-is-suave/>, Accessed: 2023-05-11.
- [56] Peter Gazi, Aggelos Kiayias, and Alexander Russell. *Stake-bleeding attacks on proof-of-stake blockchains*. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 85–92. IEEE, 2018.

- [57] Rati Gelashvili, Lefteris Kokoris–Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. *Jolteon and ditto: Network–adaptive efficient consensus with asynchronous fallback*. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 296–315. Springer, 2022.
- [58] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. *Yoso: You only speak once: Secure mpc with stateless ephemeral roles*. In *Annual International Cryptology Conference*, pages 64–93, 2021.
- [59] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. *Algorand: Scaling byzantine agreements for cryptocurrencies*. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.
- [60] Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. *Byzantine agreement with a rational adversary*. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9–13, 2012, Proceedings, Part II 39*, pages 561–572. Springer, 2012.
- [61] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. *Dumbo: Faster asynchronous bft protocols*. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.
- [62] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. *Aggregatable distributed key generation*. In *Advances in Cryptology—EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*, pages 147–176. Springer, 2021.
- [63] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. *Eclipse attacks on Bitcoin’s Peer-to-Peer network*. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144. USENIX Association, August 2015.
- [64] James Hendricks, Gregory R Ganger, and Michael K Reiter. *Verifying distributed erasure-coded data*. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 139–146, 2007.
- [65] Hai Jiang, Jun Li, Zhongcheng Li, and Xiangyu Bai. *Efficient large-scale content distribution with combination of cdn and p2p networks*. *International Journal of Hybrid Information Technology*, 2(2):4, 2009.
- [66] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. *Arbitrum: Scalable, private smart contracts*. In *27th USENIX Security Symposium*, pages 1353–1370, 2018.

- [67] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. *Constant-size commitments to polynomials and their applications*. In *Advances in Cryptology—ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5–9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.
- [68] Jonathan Katz and Chiu-Yuen Koo. *On expected constant-round protocols for byzantine agreement*. In *Annual International Cryptology Conference*, pages 445–462. Springer, 2006.
- [69] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. *Ouroboros: A provably secure proof-of-stake blockchain protocol*. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.
- [70] Valerie King and Jared Saia. *Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary*. *Journal of the ACM (JACM)*, 58(4):1–24, 2011.
- [71] Uri Klarman, Soumya Sankar Basu, Aleksandar Kuzmanovic, and Emin Gün Sirer. *bloXroute: A scalable trustless blockchain distribution network whitepaper*, 2018.
<https://bloxroute.com/wp-content/uploads/2018/03/bloXroute-whitepaper.pdf>
- [72] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. *Enhancing bitcoin security and performance with strong consistency via collective signing*. In *25th USENIX Security Symposium (USENIX security 16)*, pages 279–296, 2016.
- [73] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. *Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures*. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.
- [74] Aztec Labs. *Request for proposals: Decentralized sequencer selection*.
<https://discourse.aztec.network/t/request-for-proposals-decentralized-sequencer-selection/350/1>, Accessed: 2023-05-11.
- [75] Offchain Labs. *Arbitrum*. <https://arbitrum.io/>, Accessed: 2023-05-11.
- [76] Offchain Labs. *Arbitrum decentralization update*.
<https://offchain.medium.com/arbitrum-decentralization-update-39f093768c42>, Accessed: 2023-05-11.
- [77] Settler Labs. *Astria*. <https://www.astria.org/>, Accessed: 2023-05-11.
- [78] Leslie Lamport. *Time, clocks, and the ordering of events in a distributed system*. *Commun. ACM*, 21:558–565, 1978.

- [79] Leslie Lamport. *The part-time parliament*. *ACM Transactions on Computer Systems*, 16:133–169, 1998.
- [80] Leslie Lamport. *Paxos made simple*. *ACM SIGACT News (Distributed Computing Column)* 32, 4 (Whole Number 121, December 2001), pages 51–58, 2001.
- [81] Yuan Lu, Zhenliang Lu, and Qiang Tang. *Bolt-dumbo transformer: Asynchronous consensus as fast as the pipelined bft*. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2159–2173, 2022.
- [82] Dahlia Malkhi and Kartik Nayak. *Hotstuff-2: Optimal two-phase responsive bft*. *Cryptology ePrint Archive*, 2023.
- [83] Petar Maymounkov and David Mazieres. *Kademlia: A peer-to-peer information system based on the xor metric*. In *Peer-to-Peer Systems: First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers*, pages 53–65. Springer, 2002.
- [84] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. *Smart contracts for bribing miners*. In *IACR Cryptology ePrint Archive*, 2018.
- [85] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. *Accountable-subgroup multisignatures*. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 245–254, 2001.
- [86] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. *The honey badger of bft protocols*. *Cryptology ePrint Archive, Paper 2016/199*, 2016. <https://eprint.iacr.org/2016/199>
- [87] Iulian Moraru, David G Andersen, and Michael Kaminsky. *There is more consensus in egalitarian parliaments*. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 358–372, 2013.
- [88] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. *Decentralized business review*, page 21260, 2008.
- [89] Oded Naor, Mathieu Baudet, Dahlia Malkhi, and Alexander Spiegelman. *Cogswort: Byzantine view synchronization*. *arXiv: Distributed, Parallel, and Cluster Computing*, 2019.
- [90] Oded Naor and Idit Keidar. *Expected linear round synchronization: The missing link for linear byzantine smr.*, 2020. <https://arxiv.org/abs/2002.07539>

- [91] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Sasha Fedorova, and Ivan Beschastnikh. *Bandwidth-efficient transaction relay for bitcoin*. arXiv preprint arXiv:1905.10518, 2019.
- [92] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H Vaidya, and Zhuolun Xiang. *Improved extension protocols for byzantine broadcast and agreement*. arXiv preprint arXiv:2002.11321, 2020.
- [93] Kamilla Nazirkhonova, Joachim Neu, and David Tse. *Information dispersal with provable retrievability for rollups*. arXiv preprint arXiv:2111.12323, 2021.
- [94] Aztec Network. Aztec. <https://aztec.network/>, Accessed: 2023-05-11.
- [95] Nginx. *Deliver modern applications at scale with nginx*. <https://www.nginx.com/>, Accessed: 2023-08-08.
- [96] Valeria Nikolaenko and Dan Boneh. *Data availability sampling and danksharding: An overview and a proposal for improvements*. <https://a16zcrypto.com/posts/article/an-overview-of-danksharding-and-a-proposal-for-improvement-of-das/>, Accessed: 2023-07-12.
- [97] Optimism. Optimism. <https://www.optimism.io/>, Accessed: 2023-05-11.
- [98] Rafail Ostrovsky and Moti Yung. *How to withstand mobile virus attacks*. In *Proceedings of the tenth annual ACM symposium on Principles of distributed computing*, pages 51–59, 1991.
- [99] Rafael Pass and Elaine Shi. *Hybrid consensus: Efficient consensus in the permissionless model*. *Cryptology ePrint Archive, Paper 2016/917*, 2016. <https://eprint.iacr.org/2016/917>
- [100] Rafael Pass and Elaine Shi. *Fruitchains: A fair blockchain*. In *Proceedings of the ACM symposium on principles of distributed computing*, pages 315–324, 2017.
- [101] Rafael Pass and Elaine Shi. *Thunderella: Blockchains with optimistic instant confirmation*. In *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* 37, pages 3–33. Springer, 2018.
- [102] Al-Mukaddim Khan Pathan, Rajkumar Buyya, et al. *A taxonomy and survey of content delivery networks*. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 4(2007):70, 2007.
- [103] Youcai Qian. *Randao: A dao working as rng of ethereum*. <https://github.com/randao/randao>, 2018. Accessed: 26-July-2023.

- [104] Fred B Schneider. *Implementing fault-tolerant services using the state machine approach: A tutorial*. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.
- [105] Peiyao Sheng, Bowen Xue, Sreeram Kannan, and Pramod Viswanath. *Aced: Scalable data availability oracle*. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II* 25, pages 299–318. Springer, 2021.
- [106] Nibesh Shrestha, Ittai Abraham, Ling Ren, and Kartik Nayak. *On the optimality of optimistic responsiveness*. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 839–857, 2020.
- [107] Alexander Spiegelman, Balaji Arun, Rati Gelashvili, and Zekun Li. *Shoal: Improving dag-bft latency and robustness*, 2023.
- [108] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. *Bullshark: Dag bft protocols made practical*. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2718, 2022.
- [109] Starkware. *Starknet*. <https://www.starknet.io/>, Accessed: 2023-05-11.
- [110] Espresso Systems. *The espresso sequencer*. Accessed: 2023-05-11.
- [111] DFINITY Team et al. *The internet computer for geeks*. *Cryptology ePrint Archive*, 2022.
- [112] TD Team et al. *State machine replication in the diem blockchain.*, 2021.
- [113] Scroll Tech. *Scroll*. <https://scroll.io/>, Accessed: 2023-05-11.
- [114] Polygon Technology. *Polygon zkEVM*. <https://polygon.technology/polygon-zkevm>, Accessed: 2023-05-11.
- [115] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. *Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks.*, 2020.
- [116] Anton Wahrstätter, Liyi Zhou, Kaihua Qin, Davor Svetinovic, and Arthur Gervais. *Time to bribe: Measuring block construction market*. *arXiv preprint arXiv:2305.16468*, 2023.
- [117] Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. *Non-fungible token (nft): Overview, evaluation, opportunities and challenges*. *arXiv preprint arXiv:2105.07447*, 2021.

- [118] Sam M Werner, Daniel Perez, Lewis Gudgeon, Arian Klages-Mundt, Dominik Harz, and William J Knottenbelt. *Sok: Decentralized finance (defi)*. *arXiv preprint arXiv:2101.08778*, 2021.
- [119] Barry Whitehat. *zkrollup*, 2018. https://github.com/barryWhiteHat/roll_up, Accessed: 2023-05-11.
- [120] Barry Whitehat, Alex Gluchowski, Yondon Fu, and Philippe Castonguay. *Rollup rollback snark side chain 17000 tps*, 2018. <https://ethresear.ch/t/roll-up-roll-back-snark-side-chain-17000-tps/3675>, Accessed: 2023-05-11.
- [121] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.
- [122] Lei Yang, Vivek Bagaria, Gerui Wang, Mohammad Alizadeh, David Tse, Giulia Fanti, and Pramod Viswanath. *Prism: Scaling bitcoin by 10,000 x*. *arXiv preprint arXiv:1909.11261*, 2019.
- [123] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. *DispersedLedger: High-Throughput byzantine consensus on variable bandwidth networks*. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 493–512, Renton, WA, April 2022. USENIX Association.
- [124] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. *Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks*. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 493–512, 2022.
- [125] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. *Hotstuff: Bft consensus with linearity and responsiveness*. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.
- [126] Haifeng Yu, Ivica Nikolić, Ruom Hou, and Prateek Saxena. *Ohie: Blockchain scaling made simple*. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 90–105. IEEE, 2020.
- [127] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. *Coded merkle tree: Solving data availability attacks in blockchains*. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, pages 114–134. Springer, 2020.
- [128] Moti Yung. *The "mobile adversary" paradigm in distributed computation and systems*. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 171–172, 2015.

[129] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. *Rapidchain: Scaling blockchain via full sharding*. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[130] Dirk A Zetsche, Douglas W Arner, and Ross P Buckley. *Decentralized finance*. *Journal of Financial Regulation*, 6(2):172–203, 2020.

[131] Ge Zhang, Wei Liu, Xiaojun Hei, and Wenqing Cheng. *Unreeling xunlei kankan: Understanding hybrid cdn-p2p video-on-demand streaming*. *IEEE Transactions on Multimedia*, 17(2):229–242, 2014.

[132] Haibin Zhang, Sisi Duan, Chao Liu, Boxin Zhao, Xuanji Meng, Shengli Liu, Yong Yu, Fangguo Zhang, and Liehuang Zhu. *Practical asynchronous distributed key generation: Improved efficiency, weaker assumption, and standard model*. *Cryptology ePrint Archive*, 2022.

[133] @zksync. *Post-mortem: Zksync liveness issue*. Twitter, May 2023. <https://twitter.com/zksync/status/1642277573678090626>, Accessed: 2023-05-11.

A Savoiardi 검증 가능한 정보 분산

HotShot은 Alhaddad–Duan–Varia–Zhang(ADVZ)에 의해 제안된 VID 스킴의 변형을 사용하며, 해당 저자들은 이를 **AVID-1** [10]이라 부른다. 우리의 AVID-1 변형은 **Savoiardi**라 불리며, 원래 스킴과 다음과 같은 점에서 차이가 있다.

1. AVID-1의 *Disperse*에서 발생하는 점근적 통신 부담을 저장 노드 수에 대해 이차(quadratic)에서 선형(linear)으로 감소시킨다. AVID-1 *Disperse*의 이차 통신은 해당 스킴의 “echo” 및 “ready” 단계에서 저장 노드들 간의 전원 간(all-to-all) 메시징으로 인해 발생한다. HotShot은 이러한 단계를 제거함으로써 *Disperse*에 대해 선형 통신을 달성한다(섹션 A.7 참조).
2. 아래에서 설명하는 바와 같이, *Commit*(B)에 특정 다항식 평가에 대한 상수 크기의 벡터 커밋먼트를 포함하도록 확장한다. 이 확장의 목적은 KZG 증명을 준선형(quasi-linear) 알고리즘으로 배치 계산할 수 있도록 하기 위함이다(섹션 A.5 참조).

다음으로 Savoiardi를 비형식적으로 설명한다 — 의사코드는 *알고리즘 9*를 참조한다. 저장 노드의 수를 n , 소거 코드의 비율(rate)을 r (예: $r = 1/4$)라 하고, 블록 페이로드 B 가 분할되는 조각(fragment)의 수를 $m = rn$ 이라 하자. 일반성을 잃지 않고, 블록 페이로드 B 는 어떤 적절한 소수 체(prime field) 상의 스칼라들의 리스트로 구성되어 있으며, 이 리스트의 크기는 m 의 배수라고 가정한다. 따라서 어떤 k 에 대해 B 의 크기는 km 이 된다.

A.1 Commit

블록 페이로드 B 를 각각 m 개의 스칼라로 이루어진 k 개의 부분 리스트로 본다. $i = 1, \dots, k$ 에 대해, 부분 리스트 i 에 속한 m 개의 스칼라를 차수 $m - 1$ 의 다항식 p_i 의 계수로 보고, $\hat{p}_i$ 를 p_i 에 대한 KZG [67] 커밋먼트라 하자. 각 $j = 1, \dots, n$ 에 대해,

$$e_j = (p_1(j), \dots, p_k(j))$$

를 이들 다항식을 j 에서 평가한 값들로 이루어진 k -튜플이라 하자. 임의의 상수 크기 벡터 커밋먼트 스킴을 vc 라 하자. $Commit(B)$ 는 다음 쌍 (h, v) 로 정의된다.

$$h = \text{hash}(\hat{p}_1, \dots, \hat{p}_k)$$

$$v = vc(e_1, \dots, e_n)$$

(원한다면 $Commit(B)$ 의 비트 길이는 (h, v) 쌍을 다시 해싱함으로써 더 줄일 수 있다.)

A.2 Disperse

$Disperse(B)$ 는 블록 송신자와 저장 노드 사이에서 수행되는 단일 라운드의 상호작용 프로토콜이다. $j = 1, \dots, n$ 에 대해 송신자는 다음 데이터를 저장 노드 j 에게 전송한다.

1. 다항식 커밋먼트 $\hat{p}_1, \dots, \hat{p}_k$ 와 벡터 커밋먼트 v . (이 데이터는 모든 저장 노드에 대해 동일하다.)
2. 평가 튜플 $e_j = (p_1(j), \dots, p_k(j))$ 와 e_j 에 대한 v 의 벡터 커밋먼트 오프닝 v_j .
3. 다항식 커밋먼트에 상대적인 다항식 평가에 대한 상수 크기의 집계 KZG 증인 w_j .

KZG 증인 $w_1, \dots, w_n$ 은 다음과 같이 계산된다.

1. 의사무작위 스칼라를 계산한다.

$$t = \text{hash-to-field}(Commit(B)) \tag{1}$$

2. 의사무작위 선형 결합으로 다항식 p 를 계산한다.

$$p = \sum_{i=1}^k t^i p_i \tag{2}$$

3. 각 w_j 는 다항식 평가 $p(j)$ 에 대한 KZG 증인이다. Feist–Khovratovich 알고리즘 [53]을 사용하여 모든 KZG 증인 $w_1, \dots, w_n$ 을 준선형 시간에 배치 계산한다.

Disperse(B)	Commit(B)
▷ Sender 1: $p_1, \dots, p_k \leftarrow$ interpret B as polynomials 2: $\hat{p}_1, \dots, \hat{p}_k \leftarrow$ KZG commitments to $p_1, \dots, p_k$ 3: for $j = 1, \dots, n$ 4: $e_j \leftarrow (p_1(j), \dots, p_k(j))$ evaluate polynomials 5: endfor 6: $h \leftarrow \text{hash}(\hat{p}_1, \dots, \hat{p}_k)$ 7: $v \leftarrow \text{vc}(e_1, \dots, e_n)$ 8: $t \leftarrow \text{hash-to-field}(h, v)$ 9: $p \leftarrow \sum_{i=1}^k t^i p_i$ (random lin combo) 10: $(w_1, \dots, w_n) \leftarrow \text{batch-KZG-prove}(p)$ 11: for $j = 1, \dots, n$ 12: $v_j \leftarrow$ open v at e_j 13: Send to storage node j : 14: $\hat{p}_1, \dots, \hat{p}_k, v$ and e_j, w_j, v_j 15: endfor ▷ Storage node j 16: Receive $\hat{p}_1, \dots, \hat{p}_k, v$ and e_j, w_j, v_j 17: Verify vector opening v_j of v at e_j 18: $h \leftarrow \text{hash}(\hat{p}_1, \dots, \hat{p}_k)$ 19: $\text{Commit}(B) \leftarrow (h, v)$ 20: $t \leftarrow \text{hash-to-field}(\text{Commit}(B))$ 21: $\hat{p} \leftarrow \sum_{i=1}^k t^i \hat{p}_i$ 22: $p(j) \leftarrow \sum_{i=1}^k t^i p_i(j)$ 23: KZG-verify($\hat{p}, j, p(j), w_j$) 24: Store $\hat{p}_1, \dots, \hat{p}_k, v$ and e_j, w_j, v_j 25: indexed by $\text{Commit}(B)$ 26: Send to Sender: $\text{sign}(\text{Commit}(B))$ ▷ Sender 27: Wait for q valid sigs $s_1, \dots, s_q$ 28: of message $\text{Commit}(B)$ from storage nodes 29: $s \leftarrow$ aggregate sigs $s_1, \dots, s_q$ 30: return certificate of retrievability 31: $\text{cert}(\text{Commit}(B)) = (s, \text{Commit}(B))$	1: $p_1, \dots, p_k \leftarrow$ interpret B as polynomials 2: $\hat{p}_1, \dots, \hat{p}_k \leftarrow$ KZG commitments to $p_1, \dots, p_k$ 3: $h \leftarrow \text{hash}(\hat{p}_1, \dots, \hat{p}_k)$ 4: for $j = 1, \dots, n$ 5: $e_j \leftarrow (p_1(j), \dots, p_k(j))$ evaluate polynomials 6: endfor 7: $v \leftarrow \text{vc}(e_1, \dots, e_n)$ 8: return commitment (h, v)
	Retrieve($c, \text{cert}(c)$)
	▷ Client 1: Check validity of $\text{cert}(c)$ 2: Retrieve $\hat{p}_1, \dots, \hat{p}_k, v$ from somebody 3: $h \leftarrow \text{hash}(\hat{p}_1, \dots, \hat{p}_k)$ 4: Verify $c = (h, v)$ 5: $t \leftarrow \text{hash-to-field}(h, v)$ 6: $\hat{p} \leftarrow \sum_{i=1}^k t^i \hat{p}_i$ 7: Send to all storage nodes: “retrieve $c, \text{cert}(c)$ ” ▷ Storage node j 8: Receive “retrieve $c, \text{cert}(c)$ ” 9: Retrieve e_j, w_j, v_j and send to Client ▷ Client 10: Receive e_j, w_j, v_j from storage node j 11: Verify vector opening v_j of v at e_j 12: $p(j) \leftarrow \sum_{i=1}^k t^i p_i(j)$ 13: KZG-verify($\hat{p}, j, p(j), w_j$) 14: Store j, e_j 15: Retrieve from m storage nodes $j_1, \dots, j_m$ 16: for $i = 1, \dots, k$ 17: $p_i \leftarrow$ Interpolate from $p_i(j_1), \dots, p_i(j_m)$ 18: endfor 19: $B \leftarrow$ Interpret $p_1, \dots, p_k$ as a block payload 20: return B

Figure 9: VID

송신자로부터 이 데이터를 수신한 각 저장 노드 j 는 자신의 데이터의 무결성을 검사한다. 무결성 검사가 성공하면 저장 노드는 해당 데이터를 이후 사용을 위해 저장하고, 성공을 나타내기 위해 $\text{Commit}(B)$ 에 대한 서명을 송신자에게 반환한다.

저장 노드 j 에서의 무결성 검사는 다음과 같이 진행된다.

1. v 에 대해 e_j 의 벡터 오프닝 v_j 를 검증한다.
2. (1)에 따라 t 를 계산하고, 다음과 같이 커밋먼트 $\hat{p}$ 와 평가값 $p(j)$ 를 계산한다.

$$\hat{p} = \sum_{i=1}^k t^i \hat{p}_i \quad (3)$$

$$p(j) = \sum_{i=1}^k t^i p_i(j) \quad (4)$$

3. KZG 검증을 실행하여 증인 w_j 가 $\hat{p}, j$, 그리고 $p(j)$ 와 일관되는지 확인한다.

송신자는 최소 q 개의 저장 노드로부터 $\text{Commit}(B)$ 에 대한 서명을 대기한다. (q 의 선택은 [섹션 A.4](#)에서 논의한다.) 블록 페이로드 B 에 대한 복구 가능성 인증서는 이 q 개의 서명과 $\text{Commit}(B)$ 의 집계로 구성된다.

A.3 Retrieve

$\text{Retrieve}(c, \text{cert}(c))$ 는 클라이언트와 저장 노드 사이에서 수행되는 단일 라운드의 상호작용 프로토콜이다. 클라이언트는 다항식 커밋먼트 $\hat{p}_1, \dots, \hat{p}_k$ 와 벡터 커밋먼트 v 를 어딘가에서 — 예를 들어 저장 노드 중 하나로부터 — 가져오고, 다음을 검증하여 이 커밋먼트들의 정확성을 확인한다.

$$c = (\text{hash}(\hat{p}_1, \dots, \hat{p}_k), v).$$

다음으로 클라이언트는 (1)에 따라 스칼라 t 를 계산하고, (3)에 따라 다항식 커밋먼트 $\hat{p}$ 를 계산한다.

클라이언트는 $\text{cert}(c)$ 로부터 최소 q 개의 저장 노드의 식별자를 추출하고, 각 저장 노드에 대해 커밋먼트 c 에 대한 블록 데이터 요청을 전송한다. 저장 노드 j 는 자신의 데이터 튜플 e_j 와 증인 w_j, v_j 를 조회하여 이를 클라이언트에게 전송한다.

저장 노드 j 로부터 이 데이터를 수신한 클라이언트는 데이터의 무결성을 다음과 같이 검사한다.

1. v 에 대해 e_j 의 벡터 오프닝 v_j 를 검증한다.
2. (4)에 따라 $p(j)$ 를 계산하고, $\hat{p}, j$, 그리고 $p(j)$ 에 대해 KZG 증인 w_j 를 검증한다.

클라이언트는 저장 노드 $j_1, \dots, j_m$ 로부터 최소 m 개의 성공적인 복구를 대기한다. 각 $i = 1, \dots, k$ 에 대해, 클라이언트는 m 개의 평가값 $p_i(j_1), \dots, p_i(j_m)$ 로부터 보간(interpolation)을 통해 차수 $m - 1$ 의 다항식 p_i 를 복구한다. $p_1, \dots, p_k$ 의 계수들이 바로 블록 페이로드 B 의 데이터이다.

A.4 저장 쿼럼 크기

복구 가능성 인증서에 포함되는 저장 노드의 수 q 는 최대 f 개의 악의적인 저장 노드가 존재하더라도 *Disperse* 송신자와 *Retrieve* 클라이언트가 모두 성공함을 보장하도록 선택된다. 따라서 $m + f \leq q \leq n - f$ 를 요구한다. 이 조건을 만족하는 (f, m, q) 의 선택은 다양하다. 예를 들어 소거 인코딩으로 인한 오버헤드는 소거 코드 비율 $r = m/n$ 에 반비례하며, 이는 $m = n - 2f$ 에서 최대화되고, 이 경우 $q = n - f$ 가 된다. 또는 더 작은 r 을 선택하면 더 큰 f 또는 더 작은 q 를 허용할 수 있다.

A.5 벡터 커밋먼트의 필요성에 대하여

섹션 A.1에서 $\text{Commit}(B)$ 를 다항식 평가 튜플 $(e_1, \dots, e_n)$ 에 대한 커밋먼트 v 를 포함하도록 정의하였다. 그 이유는 무엇인가. 각 j 에 대해 의사무작위 스칼라 t 는 반드시 평가값 $p_1(j), \dots, p_k(j)$ 에 의존해야 하며, 그렇지 않으면 악의적인 송신자가 잘못된 평가값에 대해서도 유효한 KZG 증인을 생성할 수 있기 때문이다.

벡터 커밋먼트의 필요성을 피하는 대안으로는, 각 저장 노드 j 에 대해 서로 다른 스칼라 t_j 를 $t_j = \text{hash}(\text{Commit}(B), p_1(j), \dots, p_k(j))$ 로 정의하고, t 대신 t_j 를 사용하여 각 저장 노드마다 서로 다른 방식으로 $\hat{p}, p$, 그리고 $p(j)$ 를 계산하는 방법이 있다.

그러나 이러한 방식은 Feist–Khovratovich 알고리즘의 사용을 불가능하게 하며, 송신자의 실행 시간에 저장 노드 수 n 에 대한 이차(quadratic) 의존성을 도입한다. HotShot에서는 준선형 실행 시간이 우선 과제이므로, 송신자의 $O(n^2)$ 실행 시간 대신 벡터 오프닝 v_j 에 따른 추가 통신 오버헤드를 선택한다.

A.6 점근적 복잡도

$|B|, |open|$ 을 각각 블록 페이로드 B 의 크기와 벡터 오프닝 v_j 의 크기라 하자. 오버헤드를 제외한 페이로드 B 에 대한 전체 네트워크 통신량은 $O(|B|)$ 이다. 노드당 오버헤드는 $O(k + |open|)$ 이다. kn 이 $O(|B|)$ 임을 상기하자. 따라서 $|open|$ 이 상수라면 Savioardi는 최적의 점근적 통신 복잡도 $O(|B|)$ 를 달성한다.

*Disperse*와 *Retrieve*의 점근적 계산 복잡도에는 벡터 커밋먼트의 계산 및 검증과 같은 여러 비용이 포함되지만, 이러한 비용은 이들 프로토콜에서 수행되는 이산 푸리에 변환(DFT)에 의해 지배된다. 체(field) 연산은 군(group) 연산보다 저렴하므로, 우리는 이 두 비용을 분리하여 고려한다.

Disperse. 각 $i = 1, \dots, k$ 에 대해 다항식 평가 $p_i(1), \dots, p_i(n)$ 는 $O(n \log n)$ 의 비용이 들며, 총 비용은 $O(|B| \log n)$ 의 체 연산이다. 배치 KZG 증명은 $O(n \log n)$ 의 군 연산 비용이 든다.

Retrieve. k 개의 다항식 보간 $p_1, \dots, p_k$ 는 각각 $O(n \log^2 n)$ 의 비용이 들며 [25], 총 비용은 $O(|B| \log^2 n)$ 의 체 연산이다.

	Field ops	Group ops
Disperse	$ B \log n$	$n \log n$
Retrieve	$ B \log^2 n$	1

Table 1: Dominant DFT costs of two main procedures in Savoirdi.

표 1: Savoirdi의 두 주요 절차에서 지배적인 DFT 비용.

A.7 최소 종료 보장

왜 참고문헌 [10]의 AVID-1 Disperse에는 “echo” 및 “ready” 단계가 존재하며, 왜 이러한 단계들이 HotShot에서는 안전하게 제거될 수 있는가? 이러한 단계들은 AVID-1에서 강한 종료 보장을 달성하기 위해 필요하다. 구체적으로 AVID-1은 **종료(termination)**(송신자가 정직한 경우 모든 정직한 저장 노드가 Disperse를 완료함)와 **합의(agreement)**(어떤 정직한 저장 노드가 Disperse를 완료하면 모든 정직한 저장 노드가 Disperse를 완료함)를 모두 달성한다.

참고문헌 [93]에서 관찰되었듯이, 이러한 강한 종료 보장은 HotShot과 같은 프로토콜에서는 필요하지 않다. 대신, Disperse에 대해 정직한 송신자만이 프로토콜을 완료하고 유효한 복구 가능성 인증서를 획득할 수 있음을 보장하면 충분하다. 이러한 더 약한 보장은 저장 노드 간의 전원 간(all-to-all) 메시징 없이도 달성될 수 있으므로, HotShot에서는 이러한 단계를 안전하게 제거할 수 있다.

A.8 강한 가용성 보장

일부 VID 프로토콜은 약한 가용성 보장만을 제공한다. 즉, 정직한 클라이언트가 $Retrieve(C, cert(C))$ 를 게시하면 결국 종료되어 어떤 블록 페이로드 B' 를 획득할 수는 있으나, $Commit(B') = C$ 임은 보장되지 않는다. 이러한 약한 가용성 보장은 훨씬 더 단순하고 빠른 VID 프로토콜을 가능하게 하며, 임의의 소거 코드와 임의의 해시 함수로 구현될 수 있다. 이러한 유형의 최신 VID 프로토콜의 예로는 AVID-M [123]과 참고문헌 [9]의 무명 프로토콜이 있다.

약한 가용성 보장은 악의적으로 분산된 페이로드가 Disperse 단계에서는 발견되지 않을 수 있음을 의미한다. 대신 이러한 불일치는 Retrieve 단계에서, 클라이언트가 $Commit(B')$ 를 재계산하여 커밋먼트 C 와의 일관성을 확인할 때에야 발견된다.

HotShot 적대자가 VID Disperse 송신자와 HotShot의 낙관적 DA 위원회 전체를 모두 손상시켜, DA 위원회가 어떠한 데이터도 저장하지 않게 되고 복구된 블록 페이로드 B' 가 커밋먼트 C 와 일관되지 않은 경우에는 데이터 가용성이 상실될 수 있다. 다행히도 $Commit(B') \neq C$ 라는 사건 자체가 손상된 Disperse 송신자를 식별하고 처벌하기 위한 증거가 될 수 있다.

- **예시:** 누군가 저장 노드의 부분집합 S 를 구성하여 B' 를 복구하고, S 가 C 와 불일치함을 증명하는 SNARK 증명을 생성할 수 있다.
- **예시:** 저장 노드의 쿼럼이 각자 B' 가 자신의 조각과는 일관되지만 C 와는 일관되지 않음을 증명할 수 있다.

그러나 이러한 완화 기법들은 복잡하고 비용이 크다. 복잡하고 비용이 크며 드물게 사용되는 완화 절차는 특히 오류에 취약하다. 약한 가용성 VID가 제공하는 성능상의 이점이 이러한 위험을 감수할 만큼 가치가 있는지는 명확하지 않다.

A.9 관련 연구

앞서 언급했듯이, Savoiardi는 종료 보장이 더 약한 AVID-1 [10]의 변형이다. 유사한 최신 프로토콜로는 Nazirkhanova–Neu–Tse [93]의 *Semi-AVID-PR*이 있다. *Semi-AVID-PR*은 투명한 설정과 빠른 이산 로그 기반 곡선에 대한 지원과 같은 특징을 가지므로, 우리의 VID 프로토콜로 대체 사용될 수도 있다. 그러나 저장 노드 수가 큰 경우, *Semi-AVID-PR*은 AVID-1에 비해 검증 및 통신 복잡도가 더 높다는 트레이드오프를 가진다.

섹션 A.8에서는 약한 가용성 보장을 갖는 최신 VID 프로토콜의 예로 참고문헌 [123, 9]를 인용하였다.

B 구현(Instantiations)

본 절에서는 HotShot 합의 프로토콜에서 사용되는 구성 요소와 데이터 구조의 구체적인 구현을 명시한다.

B.1 랜덤 비콘

우리의 랜덤 비콘은 탈중앙화된 방식으로 예측 불가능하고 비편향적인 무작위성을 생성하도록 설계되었다. 이는 리더 선출 프로토콜, 데이터 가용성 위원회 샘플링, 그리고 합의 노드의 뷰 동기화를 위한 입력으로 주기적으로 새로운 무작위성을 제공해야 한다.

순진한 해법으로는 블록 해시나 최근 블록의 서명으로부터 엔트로피를 추출한 뒤 이를 원하는 분포로 해싱하는 방법이 있다. 그러나 이러한 해법은 모든 엔트로피 소스에 영향을 미칠 수 있는 적대자에 의해 편향되거나 예측 가능해질 수 있다.

적대자가 최대 f 단위의 스테이크만 통제할 수 있다고 가정하므로, 우리는 총 가중치가 최소 $f + 1$ 인 서명 집합으로부터 비콘을 생성할 수 있으며, 이 집합에는 반드시 하나 이상의 정직하게 생성된 서명이 포함된다. 그러나 이러한 서명들을 브로드캐스트하면 과도한 통신 오버헤드가 발생하므로, 선형 통신을 달성하기 위해 하나의 리더가 모든 서명을 수집하도록 한다. 서명을 수집하는 리더가 어떤 서명을 포함할지 임의로 선택함으로써 출력을 편향시키는 것을 방지하기 위해, 우리는 “커밋–리빌(commit–reveal)” 접근법을 채택한다. 즉, 비콘의 엔트로피 소스는 최종 비콘 값을 누구도 알기 전에 고정되고 커밋되어야 한다. 이러한 성질은 대규모 병렬 연산 능력을 가진 자원 풍부한 참여자라 하더라도 일정 시간이 경과하기 전에는 출력을 얻을 수 없는 지연 함수(delay function)를 통해 달성된다.

지연 함수

지연 함수(delay function, DF)는 지연 매개변수 $\mathbf{D}$ 가 주어졌을 때 $\mathbf{D}$ 번의 순차적 단계를 필요로 하며, 병렬 프로세서를 사용하더라도 계산을 가속할 수 없는 함수이다. 단순화를 위해 지연 매개변수 $\mathbf{D}$ 는 명시적으로 언급하지 않고, 지연 함수를 호출하는 경우를 $y \leftarrow \text{DF.Eval}(x)$ 로 표기한다.

구성

알고리즘 1에 나타난 바와 같이, 우리의 프로토콜은 다음과 같은 속성을 가진다.

- 각 블록 리더는 이미 총 가중치가 최소 $2f + 1$ 인 집계 서명을 포함하는 쿼럼 인증서를 계산한다는 점을 상기하자. 우리는 이 서명을 랜덤 비콘의 엔트로피 소스로 재사용할 수 있다.
- 블록 높이 h 에서의 새로운 비콘 값은 블록 $h - k$ 의 쿼럼 인증서로부터 생성되며, 블록 $h - k$ 는 이미 최종화되었다고 가정한다. 따라서 해당 비콘 값은 k 개 블록 앞에서는 예측될 수 없다.
- 지연 함수 DF의 출력을 계산하는 데 필요한 시간은 비편향성을 보장하기 위해 합의 진행 시간보다 충분히 길어야 한다. 한편, 새로운 비콘 값이 필요할 시점에는 이미 계산이 완료되어야 하므로, k 개의 블록을 처리하는 데 걸리는 시간보다는 짧아야 한다.
- 우리는 특정 비콘 계산 주체에 대한 DDoS 공격으로 적대자가 활성을 저해하는 것을 방지하기 위해, 모든 참여자가 이 지연 함수를 계산하도록 요구한다. 따라서 검증 가능한 지연 함수(verifiable delay function)를 사용하는 해법과 달리, 비콘 값은 단순히 값의 동일성만을 확인함으로써 검증할 수 있으며, 시간이 많이 소요되는 증명 생성 단계는 생략할 수 있다.

알고리즘 1: 탈중앙화 랜덤 비콘 프로토콜

Algorithm 1: Decentralized Random Beacon protocol

```

  ▷ When block  $\mathbf{h} = \Gamma, 2\Gamma, \dots$  is decided
1:  as everyone
2:    Let  $\text{qc.sig}$  be the quorum certificate aggregate signature of block  $\mathbf{h}$ 
3:    Begin computing  $\text{candidate\_seed} \leftarrow \text{DF.Eval}(\text{qc.sig})$  asynchronously
      We assume that the computation will be done before block  $\mathbf{h} + \mathbf{k}$ 
  ▷ On block  $\mathbf{h} + \mathbf{k}$ 
4:  as a leader
5:    Let  $\text{candidate\_seed}$  be the output of DF computed in block  $\mathbf{h}$ 
6:    Propose a block that contains  $\text{candidate\_seed}$ 
  as a node
7:    Fetch  $\text{candidate\_seed}$  from the leader's block proposal
8:    Let  $\text{candidate\_seed}'$  be the output of DF computed in block  $\mathbf{h}$ 
9:    Proceed to accept the block if  $\text{candidate\_seed} = \text{candidate\_seed}'$ 
  ▷ When block  $\mathbf{h} + \mathbf{k}$  is decided
10: as everyone
11:   Update the random seed to be the most recent finalized  $\text{candidate\_seed}$ 

```

B.2 무작위 위원회 샘플링

Tiramisu 데이터 가용성 솔루션은 각 에폭 $e \in \mathbb{N}$ 마다 Mascarpone 계층을 위해 고정된 크기 N 의 무작위 위원회를 선출하도록 규정한다.

이 위원회는 §B.1에서 설명한 랜덤 비콘 프로토콜이 생성한 시드를 사용하여 *알고리즘 2*에 따라 선출된다.

샘플링되는 모든 키는 에폭 e 에 대한 스테이크 테이블 스냅샷으로부터 선택되며, 해당 스냅샷은 에폭 $e - 2$ 의 종료 시점에 확정된다. 따라서 적대자가 위원회 노드를 적응적으로 손상시키거나 매수하지 않는 낙관적 시나리오 하에서는, 무작위 위원회가 높은 확률로 충분한 수의 정직한 노드를 포함하게 된다.

또한 스테이크 테이블 스냅샷을 생성하는 시간 간격 Γ 는, 비콘 출력을 계산하는 데 필요한 시간 간격인 k 이 상임을 다시 한 번 유의한다.

마지막으로, 리더가 새로운 블록을 제안할 때 위원회 구성원들이 데이터를 미리 확보할 수 있도록, 샘플링된 위원회는 충분한 사전 시간을 두고 노드들에게 알려져야 한다.

Algorithm 2: SampleRandomCommittee($curBlock, N$)

```
// root_active corresponds to the latest stake table snapshot which is at least  $\Gamma$  blocks ahead
1: for  $i \in [1..N]$ 
2:    $j \leftarrow \text{SampleRandom}(seed||i, |keys\_list|)$ 
3:    $key \leftarrow \mathcal{T}.\text{Sample}(\text{root}_{\text{active}}, j)$ 
4:    $committee.append(key)$ 
5: endfor
6:  $res \leftarrow \text{set}(committee)$  // Remove duplicates
7: return  $res$ 
```

B.3 쿼럼 인증서

배경

쿼럼 인증서(quorum certificate, QC)는 그 이름이 시사하듯이, 충분한 수의 서로 다른 참여자가 어떤 메시지나 진술에 대해 투표했음을 증명하는 인증서이다. QC는 HotShot의 신뢰 가능한 데이터 전파 및 트랜잭션 순서 결정 프로토콜에서 중요한 역할을 하며, 그 간결성은 기반 프로토콜에서 바람직한 통신-계산 복잡도를 달성하는 데 필수적이다. 예를 들어 다음과 같은 시나리오에서 쿼럼 인증서가 사용된다.

합의.

합의 리더는 자신의 커밋먼트 제안에 대해 충분한 수의 투표를 수신한 후, 이 투표들을 하나의 QC로 조합하고, 해당 QC를 다른 합의의 노드들에게 전송하여 커밋먼트 제안을 프리커밋/커밋하도록 한다.

데이터 가용성.

Savioardi 프로토콜에서는 블록 제안자가 저장 노드들에게 페이로드 조각을 분산시킨다. 저장 노드들은 조각을 수신했음을 확인하는 서명을 제안자에게 반환한다. 제안자는 이 투표들을 하나의 QC(이 경우 DA 인증서)로 조합한 뒤, 해당 DA 인증서를 전체 네트워크에 *신뢰성 있게 브로드캐스트*한다.

정직한 참여자는 DA 인증서를 수신한 이후, 대응되는 블록 데이터가 결국 모든 정직한 참여자에게 가용해질 것임을 확신할 수 있다.

체크포인트.

HotShot이 L1과 시퀀서 컨트랙트(sequencer contract)를 통해 상호작용하며 HotShot 합의 상태를 무신뢰적으로 체크포인트한다는 점을 상기하자. 시퀀서 스마트 컨트랙트는 커밋된 블록에 대한 QC(합의용 QC와 DA 인증서를 모두 포함)를 입력으로 받아, 블록 커밋먼트에 대해 QC를 검증하고 검증이 성공하면 블록 커밋먼트를 추가한다.

QC를 구성하는 순진한 접근 방식은 각 참여자가 메시지에 서명하고, 집계자가 서명 목록을 그대로 연결하여 QC로 사용하는 것이다. 그러나 이 방식은 증명 크기와 검증 시간이 선형으로 증가하므로, 쿼럼 크기가 큰 경우(예: 1000)에는 실용적이지 않다. 대안으로는 임계 서명(threshold signature) 스킴을 사용하여, 임의의 쿼럼이 협력하여 단일 서명을 생성하고 이를 QC로 사용하는 방법이 있다. 그러나 이 방식은 참여자들이 비밀리에 서명 키를 공유할 수 있도록 하는 비용이 큰 분산 키 생성(distributed key generation, DKG) [132, 62, 42, 73] 프로토콜을 요구한다.

또한 스테이크 키 테이블이 동적이며 자주 갱신될 수 있으므로, DKG는 주기적으로 다시 실행되어야 한다. 더 나아가 전통적인 임계 서명은 QC로부터 서명자의 신원을 추출할 수 없는데, 이는 참여자에게 보상을 지급하거나 악의적 행위를 한 참여자를 슬래싱하려는 경우 바람직한 속성이 아니다. 책임성을 갖춘 임계 서명(accountable threshold signature) 스킴 [17, 85, 13, 18, 14]은 이러한 속성을 제공하지만, 추가적인 복잡성을 수반한다.

본 절에서는 우리의 요구사항에 부합하는 두 가지 단순한 QC 구성 방식을 설명한다. 이 스킴들은 쿼럼 크기가 대략 1만(10K) 수준인 경우에도 잘 동작한다. [섹션 B.3.2](#)에서는 QC 스킴의 API와 효율성·보안 요구사항을 명시하고, [섹션 B.3.3](#)에서는 구체적인 구성을 설명한다. 첫 번째 스킴은 합의에서 사용되며, 두 번째 스킴은 시퀀서 스마트 컨트랙트에서 사용된다.

B.3.2 정의

직관적으로, 우리는 쿼럼 인증서(QC)가 간결하며 효율적인 생성 및 검증을 허용하기를 요구한다. 선택적으로, QC가 책임성(accountable)을 갖기를 원하기도 하는데, 이는 어떤 참여자가 서로 충돌하는 두 메시지(예: 서로 충돌하는 두 블록)에 대해 두 개의 QC에 서명했을 경우 이를 검증 가능하게 슬래싱할 수 있음을 의미한다.

$\mathcal{T} = \{(pk_i, w_i)\}_{i \in [n]}$ 를 동적 스테이크 키 테이블이라 하자. 여기서 n 은 등록된 전체 키의 수이며, $N_e = \sum_{i=1}^n w_i$ 는 전체 스테이크의 총합이다. 각 공개 키 pk_i 는 대응되는 비밀 키 sk_i 에 매핑된다. 우리는 QC와 관련된 알고리즘들의 구문을 다음과 같이 형식적으로 정의한다.

- $\text{sig} \leftarrow \text{QC.Sign}_{\mathcal{T}}(sk, m)$: 비밀 키 sk 와 메시지 m 을 입력으로 받아 서명 sig 를 출력한다. sig 는 비밀 키 sk 를 보유한 참여자가 메시지 m 에 대해 행사한 가중 투표(weighted vote)로 이해할 수 있다.
- $qc \leftarrow \text{QC.Assemble}_{\mathcal{T}}(\{\text{sig}_i\}, S)$: 서명들의 리스트와 부분집합 $S \subseteq [n]$ 을 입력으로 받아 QC 또는 $\perp$ 를 출력한다.
- $b \leftarrow \text{QC.Check}_{\mathcal{T}}(qc, m)$: QC qc 와 메시지 m 을 입력으로 받아, 해당 QC가 유효한지를 나타내는 비트 값을 출력한다.
- $\{pk_i\} \leftarrow \text{QC.Trace}_{\mathcal{T}}(m, qc)$: 메시지 m 과 QC qc 를 입력으로 받아, qc 의 서명에 참여한 공개 키들의 리스트를 출력한다.

QC 스킴의 효율성은 위 알고리즘들의 실행 시간과 QC의 크기로 측정한다.

QC 스킴의 쿼럼 크기 임계치를 T 라 하자. 합리적인 QC 스킴이 만족해야 할 속성들을 비형식적으로 다음과 같이 기술한다.

- **완전성(Completeness).** 총 스테이크가 최소 T 이상인 임의의 참여자 부분집합은 (임의의 메시지에 대해) QC 검사를 통과하는 유효한 QC를 구성할 수 있다.
- **위조 불가능성(Unforgeability).** 총 스테이크가 T 미만인 임의의 참여자 집합은, 총 스테이크가 T 이상인 부분집합 $S \subseteq [n]$ 과 메시지 m 을 입력으로 QC 오라클을 질의하여 대응되는 유효한 QC를 얻을 수 있더라도, *새롭고 서로 다른* 유효한 QC를 위조할 수 없다.

선택적으로, 책임성을 갖춘 QC 스킴은 다음 속성도 추가로 만족한다.

- **책임성(Accountability).** 메시지 m 에 대한 QC qc 의 생성에 참여한 임의의 참여자 집합 S 에 대해, 알고리즘 $QC.Trace(qc, m)$ 은 집합 S 에 대응되는 공개 키들을 출력한다.
- **비모함성(Non-frameability).** 임의의 정직한 참여자 $i \in [n]$ 에 대해, 집합 $[n] \setminus \{i\}$ 는 (qc, m) 쌍을 위조하여 $QC.Trace(qc, m)$ 의 출력에 pk_i 가 포함되도록 만들 수 없다.

B.3.3 구성

QC 구성은 집계 서명(*aggregate signature*) 스킴을 기본 구성 요소로 사용한다.

집계 서명 스킴 $\Pi_{\text{Agg}} = (\text{Sign}, \text{Verify}, \text{Aggregate})$ 는 일반적인 서명 스킴과 유사하지만, 추가적인 API인 $\Pi_{\text{Agg}}. \text{Aggregate}$ 를 제공한다.

이 집계 API는 서명들의 리스트와 대응되는 공개 키들을 입력으로 받아 단일 서명 sig_{Agg} 를 출력한다.

구현 예로는 BLS 집계 서명 [16]이 있다.

서명자 식별 벡터를 포함한 집계 서명

다음으로, 첫 번째 QC 구성을 설명한다. 직관적으로 QC는 집계 서명과, 해당 QC 하에서 서명한 참여자 집합을 나타내는 비트 벡터로 구성된다.

$\text{sig} \leftarrow \text{QC.Sign}_{\mathcal{T}}(\text{sk}, m)$:

1. 오라클 $\mathcal{T}$ 로부터 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$ 를 획득한다.
2. $\text{sig} \leftarrow \Pi_{\text{Agg}}. \text{Sign}(\text{sk}, (cm_{\mathcal{T}} \parallel m))$ 을 반환한다.

$qc \leftarrow \text{QC.Assemble}_{\mathcal{T}}(\{\text{sig}_i\}, S)$:

1. 총 스테이크 가중치 $\sum_{i \in S} w_i$ 가 쿼럼 임계치 T 보다 작으면 $\perp$ 를 반환한다.
2. 스테이크 테이블 $\mathcal{T}$ 로부터 공개 키 집합 $pk_S := \{pk_i\}_{i \in S}$ 를 추출한다.
3. $sig_{\text{Agg}} \leftarrow \Pi_{\text{Agg}}. \text{Aggregate}(\{\text{sig}_i\}, pk_S)$ 를 계산한다.
4. $\vec{v}_S \in \{0, 1\}^n$ 인 비트 벡터를 설정하되, $\vec{v}_S[i] = 1$ 인 경우는 오직 $i \in S$ 일 때에 한한다.
5. $qc := (sig_{\text{Agg}}, \vec{v}_S)$ 를 반환한다.

$\mathbf{b} \leftarrow \text{QC.Check}_{\mathcal{T}}(\mathbf{qc}, \mathbf{m})$:

1. $\mathbf{qc} = (\text{sig_Agg}, \mathbf{v}_S)$ 로 파싱한다.
2. 비트 벡터 $\vec{v}_S$ 로부터 S 를 추출한다.
3. 총 스테이크 가중치 $\sum_{i \in S} w_i$ 가 쿼럼 임계치 T 보다 작으면 0을 반환한다.
4. 스테이크 테이블 $\mathcal{T}$ 로부터 공개 키 집합 $pk_S := \{pk_i\}_{i \in S}$ 를 추출한다.
5. 오라클 $\mathcal{T}$ 로부터 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$ 를 획득한다.
6. $\mathbf{b} \leftarrow \Pi_{\text{Agg}}.\text{Verify}(pk_S, \text{sig_Agg}, (cm_{\mathcal{T}} \parallel \mathbf{m}))$ 를 반환한다.

$\{pk_i\} \leftarrow \text{QC.Trace}_{\mathcal{T}}(\mathbf{qc}, \mathbf{m})$:

1. $\mathbf{qc} = (\text{sig_Agg}, \mathbf{v}_S)$ 로 파싱한다.
2. 스테이크 테이블 $\mathcal{T}$ 와 비트 벡터 $\vec{v}_S$ 로부터 공개 키 집합 pk_S 를 추출한다.
3. 오라클 $\mathcal{T}$ 로부터 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$ 를 획득한다.
4. $\mathbf{b} := \Pi_{\text{Agg}}.\text{Verify}(pk_S, \text{sig_Agg}, (cm_{\mathcal{T}} \parallel \mathbf{m}))$ 를 계산한다.
5. $\mathbf{b} = 1$ 이면 pk_S 를 반환하고, 그렇지 않으면 $\perp$ 를 반환한다.

위 스킴에서 QC는 하나의 서명과 n 비트 벡터로 구성되며, n 이 중간 규모일 경우 실제로 매우 작다.

예를 들어 $n = 1000$ 일 때 비트 벡터는 125B이고, $n = 10000$ 일 때는 1.25KB이다.

QC 검증은 공개 키 집계 단계와 단일 일반 서명 검증으로 이루어지며, 일반적인 데스크톱 환경에서 충분히 실용적이다.

그러나 공개 키 집계 단계는 시퀀서 스마트 컨트랙트에서 실행하기에는 비용이 너무 크므로, 우리는 아래에서 가스 효율적인 검증을 달성하는 수정된 스킴을 제안한다.

SNARK 기반 집계 서명

직관적으로, 두 번째 스킴에서 QC는 집계 서명, 집계된 공개 키, 그리고 해당 공개 키가 올바르게 집계되었음을 증명하는 SNARK 증명으로 구성된다.

$\Pi_{\text{SNARK}} := (\cdot, \text{Prove}, \text{Verify})$ 를 SNARK 스킴이라 하자. 먼저 키 집계를 위한 SNARK 회로 C_{agg} 를 설명한다.

$C_{\text{agg}}(pk_{\text{agg}}, T, cm_{\mathcal{T}}, \vec{v}_S; \mathcal{T})$:

- **공개 입력:** 집계된 공개 키 pk_{agg} , 쿼럼 크기 임계치 T , 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$, 그리고 서명자 식별 벡터 $\vec{v}_S$
- **비밀 증인:** 스테이크 키 테이블 $\mathcal{T}$
- **SNARK 제약 조건:**
 1. $pk_S = \{pk_i\}_{i \in S}$ (및 $\{w_i\}_{i \in S}$)가 스테이크 테이블 $\mathcal{T}$ 내에서 벡터 $\vec{v}_S$ 가 나타내는 집합 S 에 대응되는 공개 키(및 스테이크 가중치)의 부분집합이 되도록 제약한다. 이때 $\mathcal{T}$ 는 커밋먼트 $cm_{\mathcal{T}}$ 에 바인딩되어 있다.
 2. 가중치 합 $\sum_{i \in S} w_i$ 가 쿼럼 임계치 T 를 초과하도록 제약한다.
 3. pk_{agg} 가 공개 키 집합 pk_S 의 집계 결과와 동일하도록 제약한다.

비고 B.1.

비고 B.2에서 언급하듯이, QC 스킴이 시퀀서 스마트 컨트랙트에서만 사용되는 경우에는 $\vec{v}_S$ 를 비밀 증인으로 이동시킬 수 있다.

두 번째 QC 스킴 QC_2 는 첫 번째 QC 스킴 QC_1 과 동일한 *Sign, Trace* API 구현을 사용한다.

여기서는 *Assemble* 및 *Check* 알고리즘에서의 수정 사항만을 설명한다.

아래 설명에서 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$ 와 쿼럼 임계치 T 는 공개적으로 알려져 있다고 가정한다.

$qc \leftarrow QC_2.Assemble_{\mathcal{T}}(\{\text{sig}_i\}, S)$:

1. $qc' \leftarrow QC_1.Assemble_{\mathcal{T}}(\text{sig}_i, S)$ 를 계산한다.
 2. $qc' = \perp$ 이면 $\perp$ 를 반환하고, 그렇지 않으면 $qc' := (\text{sig}_{\text{agg}}, \vec{v}_S)$ 로 파싱한다.
 3. 스테이크 테이블 $\mathcal{T}$ 와 벡터 $\vec{v}_S$ 로부터 공개 키 집합 $pk_S := \{pk_i\}_{i \in S}$ 를 추출한다.
 4. 공개 키 집합 pk_S 로부터 집계된 공개 키 pk_{agg} 를 계산한다.
 5. 회로 $\mathcal{C}_{\text{agg}}(pk_{\text{agg}}, T, cm_{\mathcal{T}}, \vec{v}_S; \mathcal{T})$ 에 대한 SNARK 증명 π_{SNARK} 를 계산한다.
 6. $qc := (qc', pk_{\text{agg}}, \pi_{\text{SNARK}})$ 를 반환한다.
-

$b \leftarrow QC_2.Check_{\mathcal{T}}(qc, m)$:

1. $qc = \perp$ 이면 0을 반환한다.
2. $qc = (\text{sig}_{\text{agg}}, \vec{v}_S, pk_{\text{agg}}, \pi_{\text{SNARK}})$ 로 파싱한다.
3. 오라클 $\mathcal{T}$ 로부터 스테이크 테이블 커밋먼트 $cm_{\mathcal{T}}$ 를 획득한다.
4. SNARK 공개 입력을 $PI := (pk_{\text{agg}}, T, cm_{\mathcal{T}}, \vec{v}_S)$ 로 설정한다.
5. 다음을 반환한다.

$$b := \Pi_{\text{Agg}}.Verify(pk_{\text{agg}}, \text{sig}_{\text{agg}}, (cm_{\mathcal{T}} || m)) \wedge \Pi_{\text{SNARK}}.Verify(\pi_{\text{SNARK}}, PI)$$

비고 B.2 (최적화).

주의 깊은 독자는 비트 벡터 $\vec{v}_S$ 가 QC에 포함되는 이유가 책임성을 달성하기 위함이며, 이를 통해 도전자도 오작동한 노드 집합을 식별할 수 있음을 관찰할 수 있다.

합의에서는 첫 번째 QC 스킴 QC_1 을 사용하므로, 책임성은 이미 달성된다.

따라서 시퀀서 스마트 컨트랙트 적용의 경우 다음과 같은 최적화를 적용할 수 있다.

(i) $QC_2.qc$ 에서 비트 벡터 $\vec{v}_S$ 를 제거한다.

(ii) 키 집계 SNARK 회로에서 $\vec{v}_S$ 를 공개 입력에서 비밀 증인으로 이동시킨다.