https://doi.org/10.7840/kics.2023.48.3.362

지분증명 블록체인에서의 난수 생성 기법

임 종 철◆, 오 진 태*

Random Number Generation in PoS Blockchains

Jong-choul Yim◆, Jin-tae Oh*

요 약

최근 퍼블릭 블록체인은 에너지 낭비 문제와 성능 문제를 극복하기 위해 작업증명 방식에서 지분증명 방식으로의 변환이 이루어지고 있다. 지분증명 블록체인에서는 전체 노드 중 일부 노드를 선출하여 위원회를 구성하고 위원회가 BFT 계열 합의 프로토콜을 이용하여 블록을 합의하여 생산하는 방식을 주로 사용한다. 따라서 위원회를 어떻게 선출하느냐가 블록체인의 안전성, 가용성뿐만 아니라 탈중앙성 및 공정성을 가늠하는 매우 중요한 문제가 될 수 있다. 위원회 선출에 있어서 난수의 활용은 필수적이다. 블록체인의 난수는 탈중앙화라는 특성으로 인해 비잔틴 행위를 할 수 있는 노드를 포함한 독립적인 동작을 하는 다수의 노드에 의해 협력적으로 생성된다는 특징으로 인해 공공에 의해 검증 가능해야 한다는 검증 가능성, 예측 불가능성, 편향 저항성, 가용성과 같은 요구사항을 가진다. 본 고에서는 블록체인에서 사용되는 난수의 요구사항 및 대표적인 난수 생성 기법에 대해 살펴보고자 한다.

키워드 : 블록체인, 지분증명, 난수, 비콘
Key Words : Blockchain, Proof of Stake, Randomness, Beacon

ABSTRACT

Recently, the scheme of public blockchains has been changed from Proof of Work to Proof of Stake for the purpose of overcoming the energy waste problem and the performance issue in terms of transaction processing capacity. In the PoS blockchains, a committee, which is a subset of entire nodes, produces a block by mean of BFT consensus protocol in many cases. Therefore, it is of great significance to elect a committee well in order to achieve availability, decentralization and fairness of a blockchain. To use random numbers in the election process is imperative. Randomness for a blockchain requires Verifiability, Unpredictability, Bias-resistance and Availability(Liveness) because a blockchain is operated in a decentralized manner, and needs to be work correctly in the presence of byzantine faults. This paper summarizes the requirements of randomness as well as the representative randomness generation methods used in the PoS blockchains.

Ⅰ. 서 론

비트코인[1]으로부터 유래한 퍼블릭 블록체인 기술은 최근까지 분화·발전하고 있다. 특히 대규모 노드가 동시에 참여하는 작업증명(PoW: Proof of Work) 기반의 경쟁적 블록 생성 방식이 가지는 에너지 낭비

※ 본 연구는 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임. (No. 2021-0-00118, 대규모 노드를 위한 탈중앙화 합의체 구성 기술 개발)
◆ First Author : Electronics and Telecommunications Research Institute, hektor@etri.re.kr, 종신회원
* Electronics and Telecommunications Research Institute, showme@etri.re.kr, 정회원
논문번호 : 202210-256-C-RN, Received October 19, 2022; Revised December 9, 2022; Accepted December 14, 2022

문제와 TPS(Transaction Per Second) 측면에서의 성능 문제는 새로운 형태의 블록체인을 등장시키는 원동력이 되어 왔다고 할 수 있다. 이러한 배경하에 최근에는 지분증명(PoS: Proof Of Stake) 방식을 사용하는 블록체인[2-5]이 주류로 떠오르고 있다.

일반적으로 말해, 지분증명 방식은 블록체인에 참여하는 전체 노드 중 일부 노드를 무작위로 선택하여 위원회를 구성하고 해당 위원회가 BFT(Byzantine Fault Tolerant) 계열의 합의 프로토콜을 이용하여 블록을 생산하는 구조를 가진다고 할 수 있다. 이때 위원회 노드로 선택될 확률은 사용자가 가진 지분에 비례하도록 하여 시빌 어택(sybil attack)을 방지한다.

BFT 합의 프로토콜은 퍼블릭 블록체인의 안전성과 가용성뿐만 아니라 성능을 좌지우지할 수 있는 매우 중요한 요소라고 할 수 있다. BFT 합의 프로토콜이 올바르게 동작하기 위해서는 검증인 노드 중에서 위원회 노드를 무작위로 선택하여 위원회를 구성하는 것이 필수적이다. 또한 무작위 선출은 퍼블릭 블록체인의 매우 중요한 요소인 탈중앙성 및 공정성과 직접 관련이 있다.

전술한 무작위성은 소프트웨어로 구현된 일명 PRNG(Pseudo Random Number Generator)에 의해 생성된 난수에 의존한다. 퍼블릭 블록체인에서 사용되는 난수는 탈중앙화라는 특징으로 인해, 예측이 힘들어야 하고 균일 분포(uniform distribution)를 따라야 한다는 기본 요구사항 외에 가용성 및 공공에 의해 검증될 수 있어야 한다는 요구사항을 더 가질 수 있다.

본 고에서는 블록체인에서 사용되는 난수의 요구사항 및 대표적인 난수 생성 기법에 대해 살펴보고자 한다.

Ⅱ. 배경 기술

2.1 지분증명 블록체인

지분증명 블록체인이란 사용자의 지분에 비례하여 블록 생산 자격을 부여하는 블록체인을 일컫는다. 일반적으로 사용자는 스테이킹 등의 자발적 의사표현에 의해, 블록 생산에 관여할 수 있는 검증인(validator) 역할을 획득한다. 노드 중 검증인 역할을 하는 노드를 검증인 노드라고 한다.

전형적인 지분증명 방식에서는 블록체인 네트워크를 구성하는 검증인 노드 중 일부 노드를 무작위로 선택하여 위원회를 구성하고, 그 위원회가 BFT 합의 프로토콜을 이용하여 블록을 생산한다.

퍼블릭 블록체인은 블록체인에 참여하는 노드에 대한 아무런 제약이 없기 때문에 비잔틴 행위를 할 수 있는 악의적인 노드의 위협이 항상 존재한다. 또한, 인터넷 환경에서 동작하므로 통신 네트워크가 비동기적(asynchronous)이라고 가정하는 것이 합리적이다. 단 악의적인 노드의 비율은 전체 노드에 비해 비교적 작아야 한다. 일반적으로 20%~30%를 가정한다.

이러한 가정하에 비동기 BFT 합의 프로토콜은 블록체인의 안전성과 가용성을 담보하는 핵심적인 요소라고 할 수 있다.

BFT 합의 프로토콜은 합의에 참여하는 노드의 전체 수가 n일 때, 최대로 감내할 수 있는 폴트의 수 f는 $\lfloor (n-1)/3 \rfloor$이다. 따라서 위원회가 BFT 합의 프로토콜에 기반하여 성공적으로 블록을 생산하기 위해서는, 위원회 선출 과정에서 f개 이하의 비잔틴 노드가 위원회 멤버로 선출되어야만 한다.

무작위 선출 방법은 확률적으로 정직한 노드들이 위원회의 큰 비중을 차지하도록 하여 BFT 합의 프로토콜이 정상적으로 동작하도록 데 있어 매우 중요하다고 할 수 있는데, 좋은 무작위 선출은 궁극적으로 난수 생성 기술에 크게 의존한다.

2.2 PRNG

일반적으로 SW에서의 난수 생성은 PRNG를 사용한다고 볼 수 있다. 블록체인의 난수 생성도 예외는 아니다. PRNG는 시드(seed)를 입력으로 받아 결정적(deterministic) 방식으로 난수열을 생성한다. 시드는 난수의 입력 소스 역할을 한다고 할 수 있다. PRNG에 의해 생성된 난수는 무작위성의 정도가 시드에 의존하게 되기 때문에, 예측하기 어려운 난수를 생성하기 위해서는 시드의 엔트로피(entropy)가 높아야 한다.

엔트로피가 높다는 말은 시드가 무작위적이어야 하고, 시드 값의 범주가 매우 커야 한다는 것을 의미한다. 시드 값을 m 비트의 이진 문자열로 표현할 경우 m의 크기가 시드 값의 범주를 결정한다.

엔트로피가 높은 시드를 얻기 위해서는 하드웨어의 도움이 필요할 수 있다. RFC 4086[6]에 따르면, 강력하고 신뢰할만한 하드웨어 엔트로피 소스가 없을 경우, 예측 불가능한 난수를 얻기 위해서 상호연관(correlation)되지 않은 다수의 입력 소스를 강력한 혼합 함수(mixing function)를 통해 결합하는 것이 매우 좋은 방법이라고 설명하고 있다. 강력한 혼합 함수란 여러 입력을 결합하여 하나의 출력을 생성함에 있어 각 출력 비트가 입력 비트들에 대해 복합 비선형(complex nonlinear) 함수를 적용한 값으로 생성되는 것을 의미한다. 여러 입력을 비트 단위로 xor하는 방법도 간단한 혼합 함수로 사용이 가능하고, 보다 강력

한 방법으로는 해시나 암호화 스킴을 사용하는 방법이 있다.

2.3 VRF

VRF(Verifiable Random Function)[22,26]는 두 사용자 간에 난수를 증명 가능한 형태로 공유하고자 할 때 사용할 수 있다. 사용자 A는 입력 값 x와 자신의 비밀키를 이용하여 난수로 쓰일 수 있는 결과 값 y와 y를 증명할 수 있는 증거 π를 생성한다. 사용자 B는 y를 사용자 A의 공개 키와 π를 이용하여 검증할 수 있다. 이 검증은 입력 값 x에 대해 A가 난수를 생성할 경우 항상 y라는 출력 값을 만들어낸다는 것을 보증한다.

2.4 VSS/PVSS/DKG

Shamir에 의해 제안된 기본적인 비밀 공유(secret sharing) 프로토콜[27]은 다자간 비밀(예를 들어, 암호화 키)을 공유하기 위한 것으로 신뢰할 수 있는 딜러(dealer)가 공유 대상인 비밀을 n개의 share로 나누어 참여자에게 분배하고 그 중 t개의 share를 모으면 원래의 비밀을 복원할 수 있도록 한다.

신뢰할 수 있는 정직한 딜러의 필요성으로 인해 악의적인 노드가 있을 수 있는 블록체인과 같은 분산 시스템에서 원형 그대로 사용하는 것은 바람직하지 않다.

VSS(Verifiable Secret Sharing)[17]는 비잔틴 폴트를 감내할 수 있는 비밀 공유 방법으로써, 딜러가 보낸 share가 올바른지 검증할 수 있도록 함으로써 비잔틴 행위에 대응한다. 이 과정에서 참여자 간 여러 라운드의 메시지 교환을 필요로 하기 때문에 대화형(interactive) 프로토콜이라고 할 수 있다. 대화형 프로토콜의 경우 통신 복잡도(communication complexity)가 polynomial이거나 지수적일 경우, 참여자 수나 네트워크 상황에 따라 프로토콜이 종료하는데 많은 시간이 걸릴 수 있다는 단점이 있어, 비대화형(non-interactive) 프로토콜에 대한 연구도 진행되었는데, 대표적인 것으로 Feldman이 제안한 연구[28]를 들 수 있다.

PVSS(Publicly Verifiable Secret Sharing)[18]는 VSS에 공개 키 암호화 스킴을 결합하여, 비밀 공유에 참여하는 당사자뿐만 아니라 제 3자도 당사자의 공개키만 안다면, 비밀 공유가 올바르게 이루어졌는지 검증할 수 있도록 만든 비대화형 프로토콜이다. 제 3자 검증 가능성은 특히 퍼블릭 블록체인과 같은 탈중앙화된 시스템에 매우 유용할 수 있다.

전술한 비밀 공유 기법들은 PRNG에 필요한 시드를 다자간 협력하여 생성하거나, 암호화에 사용할 공동의 비밀을 생성하는 등의 용도로 사용할 수 있다. 더불어, 후술하는 (t, n) 임계치 서명에 사용되는 공유 비밀 키를 배분하는데 사용할 수 있다.

VSS를 공개 키 암호화 스킴의 키를 공유하는 용도로 사용할 때, 여전히 하나의 딜러에 의해 생성된 비밀을 사용해야 한다는 한계를 가진다. DKG(Distributed Key Generation)[21]는 이 문제를 극복하기 위해 여러 참여자의 협력을 통해 공개 키와 비밀 키를 생성한다.

2.5 (t, n) 임계치 서명(Threshold Signature)

(t, n) 임계치 서명이란 n명의 참여자가 협력적으로 공동의 전자서명을 하는 기술로써, n명의 참여자 중 t명 이상이 동의하게 되면 성공적으로 하나의 서명을 만들어 낼 수 있는 기법이다.

지분증명 블록체인 시스템에서는 전형적으로 n명으로 구성된 위원회가 BFT 합의 프로토콜을 이용하여 블록을 생산하게 되는데, 이때 (t, n) 임계치 서명을 사용할 수 있다. 이 경우 임계치 t는 f+1이다.

임계치 서명을 하기 위해서는 서명에 사용되는 공통의 키를 공유하여야 하는데, 이를 위해 VSS/PVSS/DKG와 같은 비밀 공유 프로토콜을 사용한다.

대표적인 (t, n) 임계치 서명의 구현 방법으로 ec-schnorr 임계치 서명[25]과 Dfinity에서 사용하는 BLS 서명[24] 기반의 임계치 서명[29]이 있다. Ec-schnorr 임계치 서명은 RandHound에서 사용된다. Ec-schnorr 임계치 서명이 참여자 간 여러 라운드에 걸친 메시지 교환을 해야 하는 대화형 스킴인데 반해 BLS 임계치 서명은 비대화형으로 동작하는 것이 가능하다. 즉 임계치 서명 참여자가 단지 자신의 비밀키 share로 서명하여 서명 조각을 만들어 전송하면, 다른 참여자와 상호 연동 없이 어느 참여자라도 t개의 서명 조각을 모아 합치기만 하면 온전한 서명을 조립할 수 있다.

Ⅲ. 블록체인의 난수 생성 기법

3.1 난수의 요구사항

탈중앙화 블록체인에서 난수는 블록을 생성하는 위원회 노드들을 뽑거나 dApp에서 활용되는 등 블록체인을 보안(security)과 공정성(fairness) 측면에서 건전하게 유지하는데 필수적이라 할 수 있다. 난수는 보통 여러 노드가 관여하여 만들게 되는데, 이때 어느 한 노드가 자기가 원하는 방향으로 난수가 만들어지도록

유도하는 것을 방지해야함은 물론 난수를 미리 예측하는 것을 매우 어렵게 하여야 한다. 또한 만들어진 난수는 블록체인 상의 어느 노드라도 쉽게 검증할 수 있어야 한다.

난수를 검증 가능하도록 하는 방법은 난수 생성에 필요한 시드를 블록체인의 상태(state)에 기록하고, 해당 시드를 인자로 한 PRNG를 이용하여 난수를 생성함으로써, 어느 노드라도 같은 시드와 PRNG를 사용한다면 동일한 난수를 획득할 수 있도록 하는 방식에 의해 일반적으로 구현된다. 블록이 동기화된 상태라면 블록체인의 상태는 동일하기 때문에 블록체인 노드는 누구나 쉽게 블록체인의 난수를 검증할 수 있다. 이때 VRF(Verifiable Random Function)[22]와 같은 암호학적 기법을 가미한 난수를 사용하는 것도 가능하다.

퍼블릭 블록체인은 비잔틴 노드가 있을 수 있기 때문에 난수 생성시 비잔틴 폴트 감내를 할 수 있어야 하며, 특히 비잔틴 노드에 의해 난수가 정상적으로 생성되지 못하게 되는 문제를 극복하여야 한다.

블록체인의 난수의 요구사항은 다음과 같이 정리할 수 있다[7-10].

1) 예측 불가: 어떠한 한 노드라도 다음 난수를 예측하는 것이 불가능해야 한다.
2) 편향 저항: 어떠한 한 노드 또는 야합한 노드들이 다음 난수가 자신들에게 유리한 쪽으로 만들어지는데 영향을 줄 수 없어야 한다.
3) 공공 검증: 난수 생성에 참여하지 않은 노드들이 생성된 난수가 옳은지 공공에 공개된 정보(예: 블록체인 상태)에 근거하여 검증할 수 있어야 한다.
4) 가용성: 어떠한 한 노드 또는 야합한 노드들이 다음 난수가 생성되는 것을 막을 수 없어야 한다.

3.2 기본적인 난수 생성 방법

블록체인에서의 난수 생성은 일반적으로 다수의 노드가 관여하여 이루어진다고 할 수 있다. 다자 참여 난수 생성에서는 미리 정해진 N 명의 참여자가 있어야 하는데, 지분증명 블록체인의 경우 무작위 추첨을 통해 검증인 중 일부를 난수 생성 참여자로 선발할 수 있다. 단, 난수 생성 참여자의 자격 여부를 누구든지 검증할 수 있어야 한다.

아래에서는 블록체인에 적용될 수 있는 다자 참여 난수 생성 방법에 대해 간략히 설명한다.

3.2.1 기본 스킴

1) 참여자 i는 자신의 개인 시드[1] S_i를 생성하고 포스트한다.
2) 혼합 함수를 사용하여 포스트된 모든 S_i를 결합하여 최종 결과를 생성한다. 혼합 함수로 xor(⊕)를 사용하게 되면 무작위 출력(Random Output) R은 다음과 같이 구할 수 있다.

$$\cdot R = S_0 \oplus S_1 \cdots \oplus S_{n-1}$$

포스트한다는 의미는 다른 노드가 그 값을 알 수 있도록 블록체인 상태에 기록한다는 의미로, 스마트 컨트랙트나 별도로 정의된 프로토콜을 통해 할 수 있다.

이 방식에서는 만약 마지막 참여자가 모든 이전 값(즉, $S_0..S_{n-2}$)을 알 수 있다면 자신의 개인 시드 값을 조정함으로써, R 값을 자신이 원하는 방향으로 조작하는 것이 가능하다. 이를 보통 마지막 참여자 문제라고 부른다.

3.2.2 커미트/리빌(commit/reveal) 스킴

전술한 기본 스킴은 예측 불가 및 편향 저항 측면에서 큰 문제가 있다. 따라서 이를 방지하기 위해 커미트/리빌 스킴을 사용할 수 있다. 이 스킴에서는 커미트 단계에서 시드를 생성하였다는 서약(commitment)을 먼저 하고, 실제 시드는 리빌을 통해 나중에 공개하기 때문에, 마지막 참여자가 모든 이전 값을 알았다 하더라도 자신의 시드 값을 조정하는 것이 불가능하다. 다음과 같이 동작을 요약할 수 있다.

1) 각 참여자는 자신의 시드 S_i를 생성하고, 서약 값을 다음과 같이 계산한다.

$$\cdot C_i = \text{Message Digest}(S_i)$$

2) 각 참여자는 C_i를 포스트한다.
3) 각 참여자는 모든 참여자의 C_i가 포스트가 된 후에, 자신의 S_i를 포스트한다.
4) 무작위 출력 R은 기본 스킴과 같은 방법으로 구한다.

이 방식에서는 마지막 참여자가 자신의 시드에 대해 리빌 여부를 선택할 수 있게 되므로, 다음과 같은 두 개의 다른 결과가 생길 수 있다. 결과적으로 제한적이지만 마지막 참여자 문제가 존재한다고 할 수 있다.

a. $R = S_0 \oplus S_1 \cdots \oplus S_{n-2}$
b. $R = S_0 \oplus S_1 \cdots \oplus S_{n-1}$

만약 R를 구할 때에 참여자 전원의 개인 시드를 필

1) 시드가 아닌 난수를 직접 생성하는 방법도 가능하다.

요로 할 경우, 알고리즘이 가용적이지 않게 된다. 특히 비잔틴 행위를 할 수 있다는 가정에서는 사용하기 힘들다. 마지막 참여자의 나쁜 행동을 방지하기 위해 스테이킹 기반의 슬래싱(slashing)을 하는 방식을 사용할 수 있지만, 만약 마지막 참여자의 이익이 슬래싱에 의한 손해보다 크다면 나쁜 행동에 대한 억제력을 잃게 될 수 있다.

3.3 RANDAO

RANDAO 방식은 개념적으로는 DAO(Decentralized Autonomous Organization)에 의해 난수가 생성되는 스킴이라고 볼 수 있다. 즉 여러 독립적인 참여자에 의해 협력적으로 난수를 생성하는 방식이다. 기본 동작 원리는 커미트/리빌 스킴에 기반한다고 할 수 있으나, 실제로는 다양한 방식으로 구현된다.

3.3.1 커미트/리빌 RANDAO

가장 기본적인 유형의 스킴으로 DAO를 구성한 참여자들이 RANDAO 스마트 컨트랙트를 통해 난수를 협력적으로 생성하는 방법이다[11,12]. 스마트 컨트랙트가 난수 생성과 관련한 일종의 룰을 강제한다고 볼 수 있다. 아래와 같은 3개의 단계로 구성된다.

1) 1 단계: C_i(=sha3(S_i)) 값 수집

각 참여자는 일정 금액의 담보금을 걸고, 정해진 기간 안에 각자 생성한 비밀(secret: 개인 seed) S_i에 대한 해시를 계산하여 스마트 컨트랙트를 통해 등록한다. 커미트/리빌 스킴 상에서 커미트를 하는 과정이다.

2) 2단계: 유효한 s 값 수집

각 참여자는 S_i를 스마트 컨트랙트를 통해 리빌한다. S_i가 유효한지 여부는 C_i와 대조함으로써 확인할 수 있다. 유효한 S_i는 최종 난수를 계산하는데 필요한 시드로 포함된다.

3) 3 단계: 난수 생성

유효한 S_i들로부터 난수를 생성한다. 이때 S_i들을 xor하는 형태의 혼합 함수를 사용한다. 이때 난수 생성에 참여한 참여자에게 담보금과 인센티브를 돌려줄 수 있다.

3.3.2 Hash Onion RANDAO

전술한 방식의 문제 중 하나는 난수 생성 프로토콜이 대화형이라는 점에서 생기는 오버헤드라고 할 수 있다. 이 오버헤드를 경감하기 위해 hash onion이 사용될 수 있다. 이 기법은 이더리움의 비콘 체인[2]에서 사용하기 위해 연구가 진행되었으나, 현재의 비콘 체인에는 쓰이지 않는다. 이 기법에서는 난수 생성 참여자(예: 검증인)가 하나의 비밀로부터 hash onion을 생성하고, 그 hash onion을 그 참여자가 생성한 난수 자체와 난수에 대한 서약으로 사용하는 방식이다.

Hash onion의 구조는 아래와 같이 기술할 수 있다. s는 비밀, H는 해시 함수를 각각 의미한다.

$$C_0 = H(H(H(\dots (s) \dots)))$$
$$C_0 = H(C_1),\ C_1 = H(C_2),\ \dots\ C_{n-1} = H(s)$$

이때, 연속으로 해시를 n번 진행한 것을 간략하게 H^n으로 표현할 수 있다. 예를 들어 H^{100}은 s에 대해 100번 해시한 것을 의미한다. 여기서 C_0의 역상(preimage)을 언급할 때에는 H^{-1}로 표기할 수 있다. 즉 $H^{-1}(C_0)=C_1$이다. 따라서, $C_0=H^{100}(s)$라면, $H^{-100}(C_0)=s$이다.

참여자는 서약 C_0를 포스트하고, 이후 C_1, 그 다음엔 C_2를 포스트한다. C_1과 C_2가 시드로 쓰일 때, C_0는 C_1에 대한 서약, C_1는 C_2에 대한 서약으로 작동한다. 일반화시키면 C_i가 시드로 쓰일 때, C_{i-1}은 C_i에 대한 서약으로 쓰인다고 할 수 있다. C_i에 대한 검증은 C_i==$H(C_{i-1})$로 할 수 있다. 참여자는 C_i를 순차적으로 포스트하는 것만으로 커미트/리빌 스킴과 같은 효과를 얻을 수 있다. 만약 n이 크고, 참여자가 C_0를 포스트하는 것에 대한 시빌 어택을 할 수 없다면, 이 방식은 개별 참여자가 난수를 조작하는 것을 더욱 어렵게 할 수 있다.

Hash onion RANDAO 알고리즘의 동작은 아래와 같이 기술할 수 있다[13,14].

- 블록체인 상태는 R (난수 값, 엄밀히는 PRNG의 시드)을 가지고 있다.
- 검증인이 되고자 하는 자(노드)는 등록 과정에서 비밀 s를 생성하고 그에 기반하여 hash onion을 생성하고, 그 값(C_0)을 등록한다. 해당 검증인의 C_0는 블록체인 상태에 저장된다. 검증인 i가 등록한 C_0를 C_0V_i라고 하자.
- 블록체인의 상태에는 각 검증인의 현재 서약 값(CV_i)이 저장된다.
- 각 검증인은 난수 생성에 참여할 자격 조건을 얻게 되었을 경우(예를 들어 블록 제안자가 된 경우), 새로운 CV_i($CV_i=H^{-1}(CV_i)$)를 계산하여 포스트한다.

- 검증인 i로부터 신규 서약을 받게 되면 블록체인 상태는 다음과 같이 변경된다.

  · update CVi for validator i
  · $R = CV_i \oplus R$

이 방식은 검증인이 자신의 값을 공개하지 않는 공격(마지막 참여자 문제)이 존재할 수 있는데, 실제 사용시에는 큰 문제가 되지 않는다고 알려져 있다[13,14].

3.3.3 이더리움 비콘체인 RANDAO

이더리움 비콘체인에서 RANDAO에 기반한 난수를 어떻게 얻는지를 이해하기 위해서는 먼저 비콘체인에서 블록이 어떻게 만들어지는가를 대략적으로 이해하는 것이 필요하다. 이더리움 비콘체인은 에폭(epoch) 단위로 블록을 생성할 블록 제안자(proposer)와 생성된 블록에 대해 검증 및 투표(attestation)를 진행할 검증인을 선출한다. 현재[2] 규격에 따르면 하나의 에폭은 32개의 슬롯(slot)으로 구성되고, 슬롯 별로 하나의 블록이 생성되는 구조를 가진다. 각 슬롯은 12초의 간격을 가진다. 특정 에폭을 담당할 검증인과 제안자는 해당 에폭 이전에 RANDAO 스킴으로 만들어진 난수를 이용하여 결정된다. 그림 1은 대략적으로 비콘체인 난수가 어떻게 생성되는지를 보여준다.

비콘체인 규격[2]에 따르면 Beacon Block Body에 BLS[24] 시그너처를 저장하는 randao_reveal이라는 필드가 정의되어 있고, Beacon State에는 randao_mixes라는 필드가 정의되어 있다. randao_reveal은 블록 제안자가 블록을 생성할 때 계산하여 값을 할당하고, randao_mixes는 randao_reveal 값들을 xor 연산을 이용해 혼합한 값을 저장한다. randao_reveal 값은 에폭 번호에 대한 블록 제안자의 시그너처이기 때문에, 에폭 번호와 블록 제안자의 공개 키를 이용하여 누구나 검증할 수 있다. 공개된 고정 값에 대한 시그너처을 사용하기 때문에 커미트없이 리빌만 하여도 난수가 조작될 수 없다. 에폭 경계에 있는 randao_mixes 값은 다음 에폭의 블록 제안자 및 위원회 멤버를 선택하는 함수인 랜덤 셔플링 함수의 시드로써 활용된다. 랜덤 셔플링 함수는 블록체인 상태에 저장되어 있는 검증인을 무작위로 섞는 역할을 수행하고, 셔플링 결과에 따라 결정적(deterministic)으로 블록 제안자 및 위원회 멤버가 선택된다.

이 스킴에서도 마지막 블록 제안자가 자신이 유리한 쪽으로 리빌 여부를 결정할 수 있는 문제가 있다. 이를 극복하기 위해 VDF(Verified Delay Function)[23]를 같이 사용하는 것을 고려하고 있다.

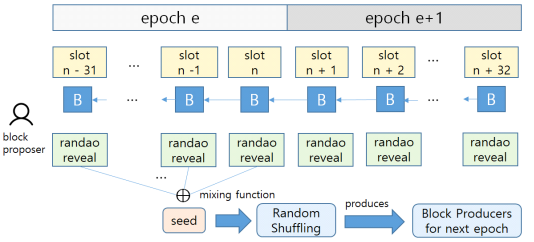

그림 1. 비콘체인 난수 생성
Fig. 1. Random number in the beacon chain

3.4 알고랜드(Algorand)

알고랜드의 시드는 블록 제안자와 위원회 멤버를 선출하는 암호학적 정렬(cryptographic sortition)의 무작위 과정에 사용되는 랜덤 함수의 엔트로피 소스 역할을 수행한다. 기본적인 메커니즘은 블록 제안자가 블록 생성 시에 시드도 같이 제시하는 방식이라고 할 수 있다. 이때 블록 제안자가 시드를 조작(manipulation)할 수 없도록 하는 방법이 필요한데, 이를 위해 조작이 불가능한 퍼블릭 값에 대해 VRF를 적용하는 방식을 취한다. 조작이 불가능한 값이란 블록 제안자가 어떤 수단을 사용하더라도 그 값을 바꿀 수 없는 것을 의미한다. 그러한 값에 대해 VRF를 통해 시드를 생성하면, 누구라도 블록 제안자의 공개 키를 알고 있다면, 난수를 검증하는 것이 가능하다. 알고랜드 논문[3]에 의하면 라운드 r에 대한 시드 값은 다음과 같이 계산한다[3]. 이때 VRF는 블록 제안자의 비밀 키를 이용하여 시드 및 증거(π)를 생성한다.

$$\langle seed, \pi \rangle \leftarrow VRF(seed_{r-1} \parallel r)$$

알고랜드의 각 검증인은 모두 독립적으로 암호학적 정렬 함수를 호출하여 자신이 다음 블록의 블록 제안자 또는 위원회 멤버가 되었는지를 확인한다. 암호학적 정렬 함수에서는 이를 위해 각 검증인의 비밀 키에 의해 동작하는 검증 가능한 랜덤 함수 즉, VRF를 사용한다. 이때 VRF의 엔트로피는 앞서 생성된 시드에 의존하게 된다.

3.5 RandShare/RandHound/RandHerd

RandShare, RandHound 및 RandHerd[7]는 (t, n) 임계치 서명과 BFT 프로토콜에 기반한 편향 저항 난

2) 2022년 9월.

3) Algorand specification[15]에서는 논문과 약간 다르다.

수 생성 기법이다. RandShare는 BFT 기반으로 난수를 생성하는 $O(n^3)$ 통신 복잡도를 가진 기본 프로토콜을 정의하고 있어, 참여자가 늘어나는 것에 대한 확장성(scalability)를 제공할 수 없는 방식이라고 할 수 있다. RandHound는 RandShare의 확장성 문제를 해결하여 대규모의 분산 노드가 참여할 경우에도 난수를 생성해 내는 것을 가능하게 하였다. 프로토콜 동작이 위원회 기반으로 블록을 합의하여 생산하는 퍼블릭 블록체인에 적합하도록 되어 있으며, 실제로 Omniledger[16]에 적용되어 사용된다. RandHerd는 미리 정해진 주기로 난수를 지속적으로 생성해 낼 수 있는 방법을 제시한다. RandHerd에 대해서는 자세히 다루지 않는다.

3.5.1 RandShare

RandShare는 N명의 참여자가 각자 자신의 비밀을 제시하고, 그 값들을 모두 혼합하여 난수를 얻는 구조이다. 예측 불가, 편향 저항 및 가용성을 확보하기 위해 VSS 기법[17]과 BA(Byzantine Agreement) 프로토콜을 같이 사용한다.

f명의 비잔틴 참여자에 대응할 수 있도록 N은 3f+1 이상이어야 한다. 이때, N명의 참여자는 각자 자신의 비밀 s를 VSS 기법을 통해 N개의 share로 쪼개어 다른 N-1 참여자에게 나누어주고, 각각의 share가 BA 프로토콜을 통해 모든 참여자들에게 올바르게 전달되도록 한다. 각 참여자의 비밀에 대한 t개 이상의 share가 모였을 때 원래 비밀 s를 복원할 수 있다. 이때 t는 f+1이다. 성공적으로 복원한 비밀의 개수가 t개보다 많으면 해당 비밀들을 xor함으로써 최종 난수(시드)를 계산한다.

난수 생성 참여자 i(피어 i)는 다음과 같은 스텝에 따라 최종 난수를 생성한다.

1) Share Distribution

피어는 비밀 s와 임의의 t-1차 다항식 $(s_i(x) = \sum_{k=0}^{t-1} a_{ik}\, x^k)$을 생성한다. 피어의 비밀 s는 $s_i(0)$로써 다항식의 0차 계수인 a_{i0}이다[4]. 피어는 주어진 다항식을 이용하여 피어의 비밀 s에 대한 n개의 검증 가능한 share를 생성한다. 이 share를 share($s_i(j)$)로 표기한다. 이때 i와 j는 1..N이고, N은 참여하는 피어의 개수이다. share($s_i(j)$)는 피어 i의 피어 j를 위한 share라고 이해할 수 있다. 피어는 생성된 share를 자신을 제외한 다른 모든 피어에게 안전하게 전송한다.

4) Shamir 비밀 공유법에 따라 비밀 s와 다항식을 만든다.

2) Share Verification

피어 i는 전송 받은 share를 먼저 검증하고, 옳다면 BA 프로토콜(prepare/commit) 메시지 교환을 통해 해당 share에 대응하는 피어의 비밀 ($s_j(0)$)이 올바르게 합의되었는지 확인한다. 피어의 비밀이 올바르게 합의되었다는 기준은 2f+1 이상의 commit 메시지가 있을 경우이다.

3) Share Disclosure

피어 i가 받은 다른 피어의 비밀에 대한 share을 공개하는 단계로, 만약 그 값이 유효하다면 해당 share를 브로드캐스트한다.

4) Randomness Recovery

만약 특정 피어에 대한 t개 이상의 share를 받았다면, 해당 피어에 대한 비밀을 복원한다. 정상적으로 복원된 비밀들을 xor하여 최종 난수를 계산한다.

그림 2는 전술한 바에 따른 RandShare의 난수 생성 방법을 도시한다.

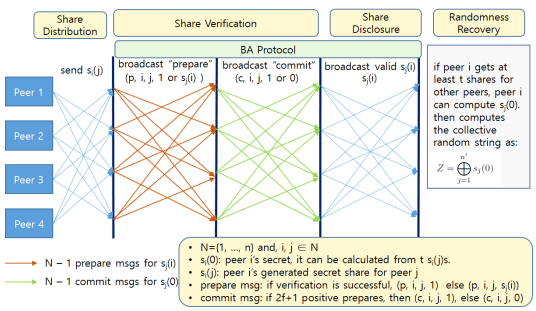

그림 2. RandShare 프로토콜
Fig. 2. RandShare Protocol

3.5.2 RandHound

RandHound는 N(>=3f+1)개로 구성된 서버들이 협력하여 클라이언트의 요청에 따라 난수를 생성하는 스킴이라고 할 수 있다. RandHound는 RandShare의 확장성 문제를 해결하기 위해 전체 서버 군을 그룹으로 나누고, 그룹 멤버들끼리만 비밀을 공유하도록 한다. 이에 따라 통신 복잡도를 $O(n^3)$에서 $O(nc^2)$로 줄일 수 있다. 여기서 c는 그룹의 평균 멤버 수이다. 난수 생성의 기본은 PVSS 스킴[18]을 따른다고 할 수 있는데, 통신 비용을 줄이기 위해 클라이언트가 난수 생성을 관장(coordination)하는 형태로 변형했다고 볼 수 있다. 이때 클라이언트의 이퀴보케이션(equivocation) 문제를 해결하기 위해 Cosi[19]를 같이 사용한다. 이로

인해 통신 복잡도는 줄었지만, 기존의 PVSS의 비대화형(non-interactive) 스킴이 대화형(interactive) 스킴으로 변하게 되어 라운드 수가 증가하는 부작용이 생긴다. 모든 정보가 다시 클라이언트로 모여서 처리되기 때문에 생기는 문제라고 볼 수 있다. 프로토콜 동작은 단일 클라이언트와 다수의 서버들이 몇 라운드에 걸친 프로토콜 메시지 교환을 하는 형태이다.

PVSS는 비밀 공유를 함에 있어 그 과정과 결과를 어느 누구라도 검증할 수 있도록 하는 메커니즘으로, 기본적으로 공개 키 암호 스킴과 더불어 NIZK(Non Interactive Zero Knowledge) 기법이 같이 쓰인다. PVSS는 3개의 단계로 구성된다고 할 수 있다. 초기화(initialization) 단계는 시스템 파라메터들을 생성하는 단계이다. 이때, 시스템 파라메터는 모든 노드에게 똑같이 적용된다. 분배(distribution) 단계는 딜러가 비밀에 대한 share를 생성하고 이를 참여자에게 전송하는 단계이다. 이때, 각 참여자를 위한 share를 보낼 때는 share를 해당 참여자의 공개 키를 이용하여 암호화하여 보낸다. 이때 암호화된 share 뿐만 아니라 해당 암호화된 share가 올바르다는 증거를 같이 전송한다. 암호화된 share와 증거, 그리고 공개 키만 알고 있다면, 누구라도 암호화된 share가 올바르다라는 것을 검증할 수 있다. 복원(reconstruction) 단계는 암호화된 share에 사용된 공개 키에 대응하는 비밀 키를 가진 참여자에 의해서 이루어진다. 해당 참여자는 암호화된 자신의 share(자신의 공개 키로 암호화된 share)를 복호화한 후 해당 share를 다른 참여자와 공유하여 원래의 비밀을 복원한다. 이때 복호화한 share와 더불어 해당 share가 올바르다는 것을 증명할 수 있는 증거도 공개한다. (t, n) 임계치 서명 스킴이 사용되어, 어느 참여자라도 t개 이상의 올바른 share를 모으면 딜러가 생성한 비밀을 복원할 수 있다.

그림 3은 RandHound의 동작 구조를 도시한다. 서버는 난수 생성에 참여하는 노드로써, 자신의 비밀을 각자 생성한다. 만약 비밀이 올바르게 공유되었다는 것이 확인된다면 최종 난수를 계산하는데 사용할 수 있다. 서버들의 비밀을 공유하는데 드는 비용을 줄이기 위해 서버들을 PVSS 그룹별로 묶고, 각 서버의 비밀은 서버가 속한 그룹 내에서만 변형된 PVSS를 사용하여 공유한다. 원래의 PVSS는 서버끼리 직접 통신을 하지만, 변형된 방식에서는 클라이언트를 경유하여 비밀을 공유하는 방식을 사용한다.

Omniledger[16]에서는 VRF를 통해 선출된 리더가 RandHound의 클라이언트 역할을 하고, 다른 검증인 노드들이 서버 역할을 하도록 하여 난수를 생성한다.

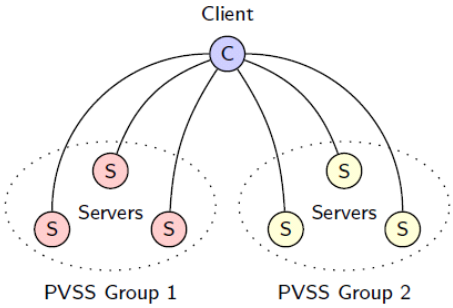

그림 3. RandHound 설계 개념
Fig. 3. An overview of the RandHound design

3.6 Ouroboros

Ouroboros[20]는 에폭에 슬롯이 존재하고, 각 슬롯별로 리더가 지정되어 슬롯 리더가 블록을 생성하는 구조를 가진다. 이때 에폭 단위로 슬롯 리더들이 협력하여 시드를 생성하게 되는데, 슬롯 리더들이 자신의 시드를 제시하면 그 시드들을 혼합하여 최종 시드를 생성하게 된다. 각 슬롯 리더가 제공하는 시드는 블록체인에 기록된다. 슬롯 리더의 시드 생성 및 공개는 PVSS 및 커미트/리빌 스킴을 이용한다. 특정 에폭의 슬롯 리더들은 최소 한 에폭 전에 알려져 있고, 그들의 공개 키도 알려져 있다. 다음과 같은 3개의 단계에 따라 난수가 생성된다.

1) Commitment 단계

이 단계에서는 슬롯 리더들이 각각 자신의 랜덤 스트링을 생성하고, 이를 share로 쪼갠다. 쪼개진 share를 다른 슬롯 리더들의 공개 키를 이용하여 암호화하고 또한 랜덤 스트림에 대한 서약을 생성하여 share와 함께 블록체인에 등록한다.

2) Reveal 단계

슬롯 리더들이 커미트가 이루어진 자신의 랜덤 스트링을 블록체인에 등록함으로써 공개한다. 정상적으로 리빌이 된다면, 각 슬롯 리더가 생성한 랜덤 스트링을 모든 노드가 알 수 있게 된다.

3) Recovery 단계

슬롯 리더 중에 리빌을 제대로 수행하지 않은 노드가 있을 경우, 블록체인에 등록된 share를 통해 해당 노드의 랜덤 스트링을 복원하는 단계이다. 리빌이 되지 않은 슬롯 리더의 share를 가진 모든 노드들이 해당 share를 복호화하여 블록체인에 등록한다. 복호화한 share들을 이용하여 원래의 랜덤 스트링을 복원할

수 있다.

요약하면, 정상적인 상황에서는 커미트/리빌 방식에 따라 동작하지만, 만약 특정 슬롯 리더가 자신의 시드를 공개하지 않음으로써 이익을 얻고자 할 경우를 대비하기 위해, PVSS를 사용하여 이를 방지하는 방식이라고 할 수 있다.

3.7 DFINITY

Dfinity의 난수 생성은 마지막 참여자 문제 해결하기 위해 BLS 기반의 임계치 서명을 사용하는 것이 핵심이라고 할 수 있다. (t, n) 임계치 서명을 사용하게 되면 마지막 t번째 노드가 난수 생성에 참여하지 않더라도 n - t개의 노드가 남아 있기 때문에 마지막 참여자 문제를 억제할 수 있다.

블록에는 난수 r이 포함되는데, 이전 블록에 포함된 r 값과 신규로 생성할 블록의 번호를 대상으로 임계치 서명을 생성하고, 그 서명에 대한 해시 값을 신규로 생성할 블록의 r 값으로 저장한다. (t, n) 임계치 서명을 수행하기 위해서는 먼저 전자서명에 사용할 공개키와 비밀 키가 서명에 참여하는 노드들에게 공유가 되어야 하는데, 이를 위해 Joint-Feldman DKG[21]를 사용한다.

DKG는 비밀을 분배하는 신뢰할 수 있는 딜러에 의존하지 않고도, 여러 노드들이 협력하여 비밀을 분배할 수 있는 방법이다. 여기에서는 임계치 서명에 사용되는 비밀 키가 비밀에 해당한다. DKG가 끝나게 되면, 각 노드는 임계치 서명에 사용하게 될 자신만의 비밀 키 share를 가지고 되고, 임계치 서명을 검증할 공개 키를 알게 된다. 각 노드는 서명 시 비밀 키 share를 이용하여, 자신의 서명 조각을 만들게 되는데, t개의 서명 조각을 모으면 하나의 최종 서명을 얻게 된다. 이 최종 서명은 공개 키를 이용해 검증할 수 있다.

그림 4는 Dfinity의 난수 생성 과정을 도시한다.

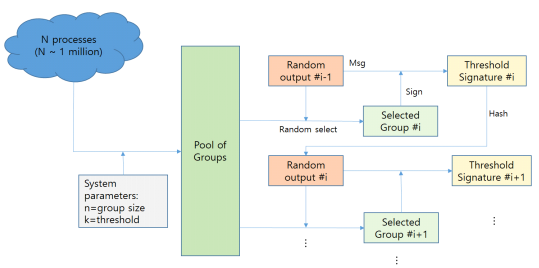

그림 4. DFINITY의 난수 생성 과정
Fig. 4. Randomness generation of DFINITY

Ⅳ. 난수 생성법 비교

표 1은 3장에서 다룬 난수 생성 방법을 요약한다.

예측 불가는 PRNG나 PRF(Pseudo Random Function)의 입력으로 사용되는 시드의 엔트로피에 의존한다. 열거한 방법이 모두 다수 참여자가 개별적으로 생성한 입력 소스에 대해 암호학적 스킴(해시/공개키 암호)을 사용한 혼합 함수를 사용하기 때문에 예측 불가를 만족한다고 할 수 있다.

입력 소스를 섞는 방식에 따라 편향 저항에 영향을 줄 수가 있다. 순차적으로 입력 소스를 섞게 되면, 뒤 순서의 참여자는 앞서 공개된 입력 소스를 미리 볼 수 있기 때문에 시드를 자기가 유리한 방향으로 조작할 수 있다. 이를 방지하기 위해 커미트/리빌 스킴을 이용할 수 있다. 그러나 미미하지만 여전히 마지막 참여자 문제가 발생할 수 있다. 비콘 체인과 알고랜드가 여기에 해당한다. 이 문제를 극복하기 위해 Ouroboros는 recovery 단계를 두었고, Dfinity는 (t, n) 임계치 서명을 이용하였다.

입력 소스를 병렬적으로 섞을 경우에 만약 네트워크가 동기적이라면 앞서 언급한 마지막 참여자 문제가 발생하지 않을 수 있으나, 인터넷 환경에서는 동일한 문제가 발생할 수 있다. RandShare와 RandHound는 (t, n) 임계치 서명을 통해 이 문제를 극복하였다.

공공 검증은 난수 생성 참여자의 입력 소스가 블록체인 상태에 기록되고 검증에 필요한 정보도 모두 공

표 1. 난수 생성 방법 비교
Table 1. Comparison of randomness generation

Protocol	U/B/V/A	Communication Complexity	Cryptographic Scheme
Beacon chain (randao)	√/x/√/√	O(n)	BLS
Algorand	√/x/√/√	$O(n^2)$	VRF
RandShare	√/√/x/√	$O(n^3)$*	VSS, ec-scnorr thr-sig.
RandHound	√/√/√/√	$O(nc^2)$	PVSS, ec-scnorr thr-sig.
Ouroboros	√/√/√/√	$O(n^3)$	PVSS
DFINITY	√/√/√/√	$O(n^2)$	DKG, BLS thr-sig.

U: Unpredictability, B: Bias-Resistance, V: Verifiability by Public, A: Availability
* 메시지 교환 회수 입장에서는 $O(n^2)$이지만, 메시지 크기를 고려하면 $O(n^3)$이다.

공에 공개하는 비콘 체인, 알고랜드, Ouroboros 및 Dfinity의 경우에는 달성된다고 할 수 있다. RandHound는 PVSS를 사용하기 때문에 누구든지 생성된 난수(시드)를 검증할 수 있으므로 공공 검증을 만족하지만, RandShare는 VSS로 인해 난수 생성 참여자만 검증을 할 수 있어 공공 검증을 만족시킬 수 없다.

가용성은 각 블록체인에서 사용하는 합의 프로토콜의 liveness와 관련이 있다. 비콘 체인, 알고랜드, Ouroboros 및 Dfinity 모두가 BFT 합의 프로토콜에 기반하여 블록을 차례대로 생성하고, 생성된 블록에 기록된 입력 소스에 의거하여 시드를 계산하기 때문에 가용성을 만족한다고 볼 수 있다. RandShare와 RandHound도 BFT 프로토콜에 의해 시드를 생성하므로 가용성을 제공한다고 볼 수 있다.

V. 결 론

에너지 낭비 문제와 트랜잭션 처리 속도 측면에서의 성능 문제를 극복하기 위해 최근 블록체인 스킴이 작업증명 방식에서 지분증명 방식으로 바뀌어 가고 있다. 지분증명 블록체인에서는 많은 경우에 전체 노드 중 일부 노드를 선출하여 위원회를 구성하고 그로 하여금 블록 생산을 책임지도록 하고 있다. 이때, 특히 탈중앙화 및 안전성을 담보하기 위해서는 좋은 속성을 가진 난수를 통해 위원회를 선출하는 것이 필수적이다. 본 고에서는 블록체인에서 사용되는 난수의 요구사항 및 대표적인 블록체인의 난수 생성 및 활용 방법에 대해 정리하였다.

References

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008, Retrieved Sep. 1, 2022, from www.bitcoin.org.

[2] Ethereum, *Ethereum Proof-of-Stake Consensus Specifications*, Retrieved Sep. 1, 2022, from https://github.com/ethereum/consensus-specs.

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling byzantine agreements for cryptocurrencies,” in *Proc. 26th Symp. Operating Syst. Principles*, ACM, pp. 51-68, 2017. (https://doi.org/10.1145/3132747.3132757)

[4] A. Yakovenko, *Solana: A new architecture for a high performance blockchain v0. 8.13. Whitepaper*, 2018, Retrieved Sep. 1, 2022, from https://solana.com/solana-whitepaper.pdf

[5] T. Rocket, M. Yin, K. Sekniqi, R. van Renesse, and E. G. Sirer, “Scalable and probabilistic leaderless BFT consensus through metastability,” *arXiv preprint arXiv:1906.08936*, 2019.

[6] D. Eastlake, J. Schiller, and S. Crocker, “*Randomness requirements for security*,” IETF RFC 4086, Jun. 2005.

[7] E. Syta, P. Jovanovic, E. Kokoris-Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, “Scalable bias-resistant distributed randomness,” in *38th IEEE Symp. Secur. and Privacy*, May 2017. (https://doi.org/10.1109/SP.2017.45)

[8] M. Raikwar and D. Gligoroski, “SoK: Decentralized randomness beacon protocols,” *Inf. Secur. and Privacy, ACISP 2022*, pp. 416-431, 2022. (https://doi.org/10.1007/978-3-031-22301-3_21)

[9] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl, “Hydrand: Efficient continuous distributed randomness,” in *2020 IEEE Symp. Secur. and Privacy*, pp. 73-89, 2020. (https://doi.org/10.1109/SP40000.2020.00003)

[10] M. Raikwar, “Competitive decentralized randomness beacon protocols,” in *Proc. Fourth ACM Int. Symp. Blockchain and Secure Critical Infrastructure*, 2022. (http://dx.doi.org/10.1145/3494106.3528679)

[11] randao.org, *Randao: Verifiable Random Number Generation*, Retrieved Sep. 1, 2022, from https://www.randao.org/whitepaper/Randao_v0.85_en.pdf

[12] Randao, *RANDAO: A DAO working as RNG of Ethereum*, Retrieved Sep. 1, 2022, from https://github.com/randao/randao.

[13] V. Buterin, *RNG exploitability analysis assuming pure RANDAO-based main chain*, Retrieved Sep. 1, 2022, from https://ethresear.ch/t/rng-exploitability-analysis-assuming-pure-randao-based-main-chain/1825.

[14] V. Buterin, *RANDAO beacon exploitability an*

alysis round 2, Retrieved Sep. 1, 2022, from https://ethresear.ch/t/randao-beacon-exploitability-analysis-round-2/1980.

[15] Algorand, *Algorand Byzantine Fault Tolerance Protocol Specification*, Aug. 23, 2022, Retrieved Sep. 1, 2022, from https://github.com/algorand foundation/specs/releases/.

[16] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," *2018 IEEE Symp. Secur. and Privacy*, pp. 583-598, 2018. (https://doi.org/10.1109/SP.2018.000-5)

[17] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *SFCS '85*, pp. 383-395, Washington, DC, USA, 1985. (https://doi.org/10.1109/SFCS.1985.64)

[18] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *IACR CRYPTO*, pp 784-784, 1999. (https://doi.org/10.1007/3-540-48405-1_10)

[19] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities "Honest or bust" with decentralized witness cosigning," in *37th IEEE Symp. Secur. and Privacy*, May 2016. (https://doi.org/10.1109/SP.2016.38)

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *CRYPTO 2017*, vol. 10401, pp. 357-388, 2017. (https://doi.org/10.1007/978-3-319-63688-7_12)

[21] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *EUROCRYPT '99*, pp. 295-310, Prague, Czech Republic, May 1999. (https://dx.doi.org/10.1007/3-540-48910-X_21)

[22] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annu. Symp. Foundations of Comput. Sci.*, pp. 120-130, 1999. (https://doi.org/10.1109/SFFCS.1999.814584)

[23] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," *Advances in Cryptology - CRYPTO*, pp. 757-788, Springer International Publishing Cham, 2018. (https://doi.org/10.1007/978-3-319-96884-1_25)

[24] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *ASIACRYPT*, Dec. 2001. (https://doi.org/10.1007/3-540-45682-1_30)

[25] D. R. Stinson and R. Strobl, "Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates," in *ACISP*, pp. 417-434, Berlin, Heidelberg, 2001. (https://doi.org/10.1007/3-540-47719-5_33)

[26] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Int. Wkshp. Public Key Cryptography*, pp. 416-431, Springer, 2005. (https://doi.org/10.1007/978-3-540-30580-4_28)

[27] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612-613, Nov. 1979. (https://dx.doi.org/10.1145/359168.359176)

[28] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *IEEE FOCS'87*, pp. 427-437, 1987. (https://doi.org/10.1109/SFCS.1987.4)

[29] T. Hanke, M. Movahedi, and D. Williams, "Dfinity technology overview series, consensus system," *arXiv preprint arXiv:1805.04548*, 2018.

임 종 철 (Jong-choul Yim)

1997년 2월 : 서울시립대학교 전산통계학화 졸업
2000년 2월 : 서울시립대학교 전산통계학화 석사
2000년 10월~현재 : 한국전자통신연구원 근무
2022년~현재 : 충남대학교 컴퓨터공학과 박사과정
<관심분야> 블록체인, 분산 시스템, 합의 알고리즘, 네트워크 보안

오 진 태 (Jin-tae Oh)

1990년 2월 : 경북대학교 전자공학 졸업
1992년 2월 : 경북대학교 전자공학 석사
2011년 2월 : 충남대학교 컴퓨터공학 박사
2003년 3월~현재 : 한국전자통신연구원 근무
<관심분야> 블록체인 분산합의, 네트워크 보안, 고속하드웨어 설계