REAP CHAIN Co., Ltd.

REAP CHAIN Yellow Paper

Ver 0.7

REAP CHAIN

목차

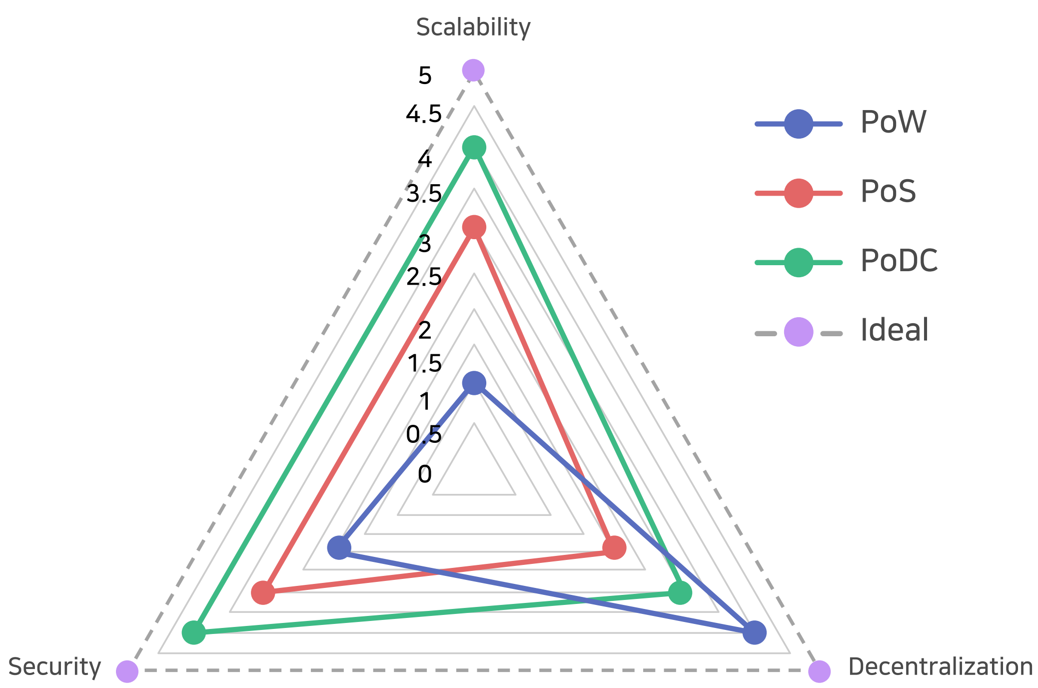
REAP CHAIN의 기본 설계 방향

REAP CHAIN은 현존하는 블록체인 플랫폼의 여러 가지 문제점을 혁신적으로 해결하고자 다음과 같은 방향으로 블록체인 생태계를 위한 향상된 방안을 제시합니다.

REAP CHAIN은 기존에 다양한 블록체인 메인넷에서 제시된 합의알고리즘의 단점을 극복하기 위하여, 허가된 노드와 일반 노드에서 퀀텀난수를 적용하여 선정된 노드로 구성된 합의 과정으로, 속도와 안정성, 그리고 보안성을 동시에 향상시키고자 합니다.

Trilemma 해결을 위한 방안

블록체인의 3중 딜레마 즉 트릴레마는 확장성, 보안, 탈중앙화 입니다. 블록체인의 트릴레마는 이 모든 세 가지 이슈를 한번에 해결할 수가 없다는 것으로, 한 가지를 향상하기 위해 새로운 시도를 하면 다른 나머지가 약화되는 문제점이 발생합니다. 예를 들어 PoS또는 DPoS는 PoW의 약점인 확장성과 보안을 높이기 위해 지분증명 방식을 도입하지만 이 때문에 중앙화되는 문제점이 발생합니다.



Comparison of Blockchain Trilemma Solutions

확장성(scalability) 측면

일반적으로 사용자 수가 증가함에 따라 거래건수가 증가하고 이에 따른 전송처리용량 증대의 문제가 발생합니다. REAP CHAIN에서는 Proposal Block을 전파하는 단계인 Pre-prepare 단계를 제외하고는 일정한 형태의 내부 구성을 가지기 때문에 네크워크 확장시에도 처리용량의 증대와 같은 문제가 별도로 발생하지 않습니다. 네트워크의 노드 숫자가 증가하더라도 합의과정에서는 항상 일정 수의 노드만이 참여하여 진행하기 때문에 네트워크 확장시에도 일정한 처리속도를 보장할 수 있습니다.

탈중앙화(decentralization) 측면

REAP CHAIN에서는 블록체인 네트워크에서 모든 통신처리가 자발적이며 자율적으로 연결되며, Block 생성시 마다 코디네이터와 임의의 후보군을 동적 선정 방식으로 지정하여 상호간 연결이 자유로운 방법으로 처리됩니다. 또한 각 블록이 생성될 때마다 매번 합의 과정에 참여하는 주체가 달라집니다. 모든 블록생성 및 블록 확정 과정에서 외부의 개입없이 자체적이고 자발적으로 진행하는 방식으로 후보군의 지정과 후보군의 확정이 별도로 알고리즘에 의해서 이루어지기 때문에 중앙 또는 외부의 관여가 없습니다.

보안(security) 측면

보안의 핵심 목표는 기밀성, 무결성 및 가용성입니다. REAP CHAIN에서는 블록체인 내의 데이터나 프로그램의 권한이 없는 사용자가 블록생성시 접근할 수 없도록 하기 위해서 공개키 방식을 이용하여 데이터를 처리합니다. 퀀텀난수를 이용하여 선정된 노드만이 열람할 수 있는 각 노드별 공개키를 제공하여 블록을 확정하는 단계에서 허가된 노드만 투표에 접근이 가능하도록 기밀성을 유지합니다. 외부의 개입이 없고 예측이 불가능한 퀀텀난수를 이용하여 선정된 노드의 공개키 서명으로 블록을 확정하므로 데이터의 무결성을 유지합니다. 또한 전체 네트워크의 모든 사용자가 정보에 함부로 접근할 수 없도록 하고 보안키를 이용하여 해당 블록이 생성시 선정된 노드만 접근할 수 있도록 하여 서비스가 원활하도록 하는 가용성을 제공합니다.

특히 컴퓨팅 파워 공격이 가능하기 위해서는 악의적으로 제어할 수 있는 노드를 늘려서 장악하고 이를 통하여 공격하는 경우인데, REAP CHAIN에서는 블록 생성시에 블록 확정을 위한 노드를 후보군으로 균등하게 배분하며, 노드가 늘어나는 경우에 안전성이 더 높아지며, 신뢰성이 있는 상시 관리되는 노드로 구성된 합의 그룹과 결합하여 합의가 도출되기 때문에 안전합니다.

컴퓨팅 파워 공격으로 후보군으로 선정된다고 하더라도 최종 투표 그룹의 결정은 예측불가능한 퀀텀난수를 이용하여 별도로 이루이지기 때문에 외부 특정인의 의도적인 공격이 불가능합니다. 다른 회원 노드에게 각각 다른 권한을 부여할 수 있으며, 이는 거버넌스에 등록 및 관리 투표를 통하여 신분상승을 진행하고 평판 또는 권위를 제공합니다. 합의과정에 참여할 후보군 노드로 신원을 등록하기 위해서는 등록 비용을 지불하고 상임위 및 운영위 후보군의 다른 수준의 신원 규정을 두어 의무, 책임 그리고 권한을 부여합니다. 네트워크 참여 전에 신원을 확인하고, 합의시에는 각 역할에 따라서 거버넌스 및 Qmanager 그리고 코디네이터에서 한번 더 확인합니다.

따라서 REAP CHAIN은 신뢰를 확보한 내부 및 외부 노드로 균형있게 구성하고 [Highly Secured], 그러한 노드들이 보안이 강화된 퀀텀난수를 적용한 알고리즘으로 간섭없이 선정되고 [Decentralized], 퀀텀난수로 선정된 코디네이터에게 최소의 메시지로 직접적인 투표로 합의를 이끌어 낸 후 [Highly Scalable], 지불된 모든 거래 비용을 모아서 코인을 보유한 모든 참여자들에게 공평하게 배분합니다.

퀀텀난수(Quantum Random Number) 선정 방식 도입

일반적으로 난수를 발생시키는 방법은 여러 가지가 있습니다. 컴퓨터 소프트웨어에서 수학적으로 난수를 발생시키는 함수를 사용하여 유사 난수(pseudo random number)를 생성합니다. 유사 난수는 일반적인 산업 분야와 특히 금융, 항공, 보안 분야 등에서도 사용하고 있습니다. 하지만 유사 난수를 자주 사용하다 보면 일정한 패턴이 발생하여 다음에 발생할 난수를 확률상으로 예측할 수 있습니다. 어떤 합의알고리즘에서는 노드가 수학적으로 검증한 유사난수를 사용하여 투표에 참여할 수 있는 권한을 가지게 되는 데 이것을 검증하기 위해서 많은 시간을 필요로 합니다. 또한 이 합의알고리즘은1000여 개의 합의 노드가 필요하고 이는 초기 네트워크가 확장되지 않으면 작동할 수 없습니다.

이와 다르게 시도되고 있는 방법이 하드웨어적으로 난수를 발생시키는 것입니다. 자연현상에 기반한 특성을 활용하는 경우, 예를 들어 빛의 양자 또는 동위원소의 원소 움직임은 완전 무작위적이기 때문에 전혀 예측할 수 없다고 합니다. 이런 아날로그적인 움직임을 디지털로 수치화하여 난수를 생성하는 방법이 가장 예측불가능하다고 합니다. 유사난수를 사용하는 다른 합의알고리즘과는 달리 REAP CHAIN에서는 신뢰가 있는 퀀텀난수생성기를 도입합니다. 이를 사용하여 얻어진 난수에 대해 공개키로 암호화하여 활용하므로 안전하게 보호가 됩니다.

PoDC (Proof of Double Committee) 합의알고리즘 도입

REAP CHAIN에서는 내부와 외부의 균형을 위하여 상임위 노드와 운영위 노드를 운영하고, 이런 노드들로 이중 위원회(double committee)를 구성하여 합의과정을 진행합니다. 상임위 노드는 REAP CHAIN 블록체인 네트워크를 상시 운영되도록 하기 위해 별도로 구축된 내부 그룹입니다. 상임위 노드는 REAP코인의 총 발행량의 2%를 보유해야 합니다. 이에 반해 운영위 노드는 일반 노드 중에서 100,000 REAP코인 보유 조건에 맞으면서 거버넌스에 등록과정을 거쳐서 허가된 외부 그룹으로 구성됩니다. 운영위 노드는 해당 조건을 갖춘 노드이면 누구나 등록가능합니다. 사실 이렇게 등록된 그룹은 실제 합의 과정에서는 운영위 후보군으로 불립니다.

PoDC는 상임위와 운영위 구성에서 탈중앙화를 지향합니다. 상임위 노드는 초기에는 거버넌스에 의해 커뮤니티에서 주도하여 운영되지만, 기존 상임위 노드 중 중지되거나 조건이 맞지 않는 경우, 운영위 노드 중 상임위 노드의 조건을 갖춘 후보군에서 추가로 선정될 수 있습니다. 이 때 거버넌스의 별도 투표 절차를 거쳐야 운영위 노드에서 상임위 노드로 승격이 될 수 있습니다.

REAP CHAIN은 상시 운영되는 14개의 상임위 노드와 일반 노드 중에서 보안에 강한 퀀텀난수를 활용한 방식으로 예측불가능한 무작위 추출로 선택된 15개의 운영위 노드로 구성되며, 이러한 29개의 노드가 블록 생성을 위한 합의 과정에 참여합니다. 합의에 참여하는 노드 중에서 운영위 노드가 참여하는 비율은 52% 이상 유지하여야 상시 운영되는 상임위 노드가 공정하게 운용되고 있는 지 감시할 수 있습니다.

<그림 – PoDC 이중위원회 노드 구성>

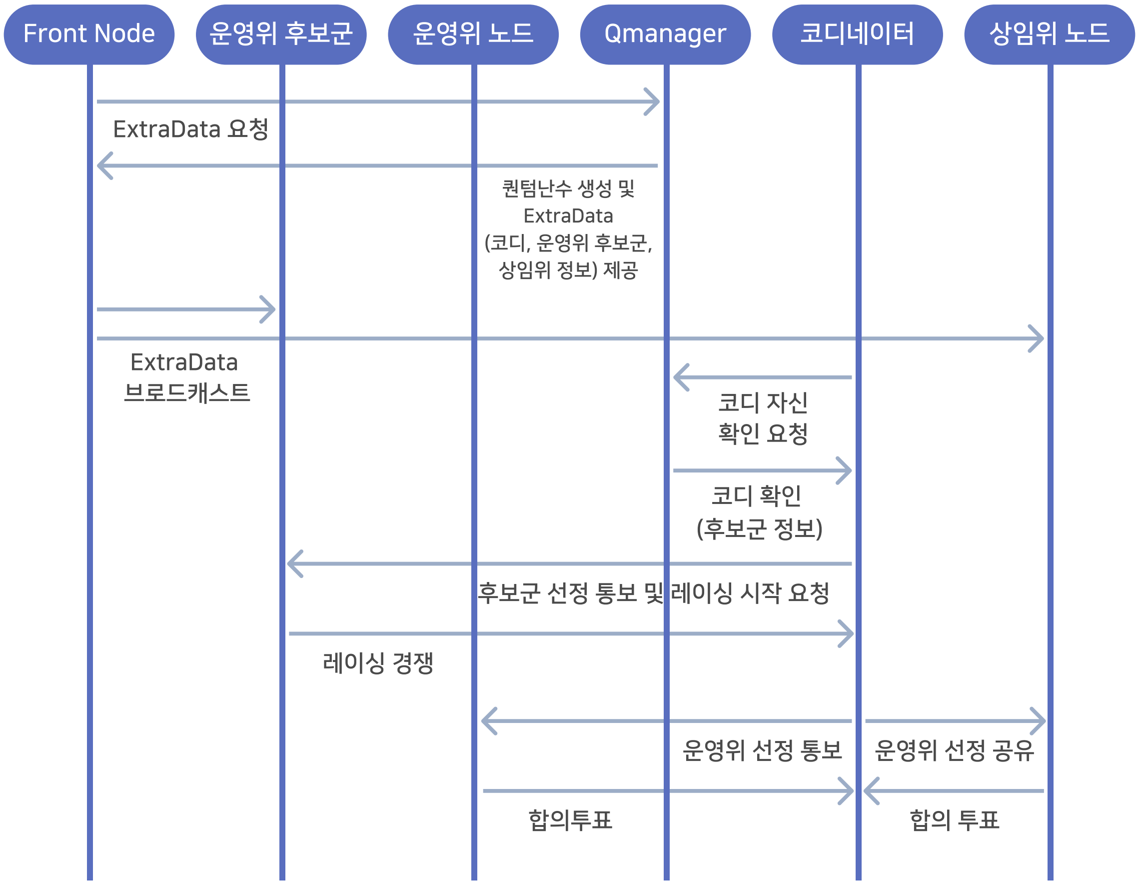
▣ 합의과정 및 블록생성

REAP CHAIN은 기본적인 블록 생성 주기는 2초이지만 임시 원장에 해당하는 이중체인 미들웨어에서는 1초 미만으로 가승인 통보를 해줌으로써, 소비자가 느낄 수 있도록 카드 결제 승인과 같은 효과를 낼 수 있습니다. 실제 REAP CHAIN에서 블록이 2초 이내로 생성되므로 소비자는 결제 승인 지연을 거의 느끼지 못합니다. 이를 통해서 REAP CHAIN은 상용화 거래에 적합합니다. REAP CHAIN은 2초 이내에 상시 운영되는 상임위 중에서 코디네이터를 선정하여 빠르게 합의에 이를 수 있는데, 이는 선정된 코디네이터를 중심으로 나머지 28개의 노드들은 합의과정 중에 노드간의 직접 통신 상태로 최소의 네트워크 트래픽으로 합의에 도달하기 때문입니다. 구체적으로는, P2P네트워크에서는 broadcast되는 확인 메시지를 모으는 과정에서 시간 지연과 네트워크 부하가 발생하지만, 내부 네트워크(내부 조직)에서는 코디네이터와 메시지가 한번만 발생합니다. 이러한 코디네이터는 결국 오직 한 개의 블록 생성자 (Block Producer)가 되어 블록을 생성할 수 있습니다. 2초 이내에 블록이 생성되지 못한 경우에는 해당 주기의 블록 생성은 건너뛰게 됩니다.

REAP CHAIN에서는 2초 이내로 짧은 시간에 블록이 생성되면서 매 블록마다 확정(Finality)을 하게 되면 시간이 좀 더 걸릴 여지가 있으므로, 이더리움 캐스퍼 방식처럼 매 20번째 블록에서 Checkpoint를 두어서 일정 수의 블록을 모아서 거래를 확정하는 부분을 향후 개선할 예정입니다.

REAP CHAIN은 블록 생성을 위해 하나의 라운드 테이블을 생성합니다. 하나의 라운드에서는 일반 노드 중에서 보안이 강화된 랜덤 선정방식으로 선택된 15개의 운영위 노드와 14개 상임위 노드는 함께 트랜잭션의 유효성 검사에 참여합니다. 이때는 REAP CHAIN의 QSN (Quantum Safety Net) 네트워크 내에서 이루어지는 합의의 신뢰를 보장합니다.

블록 제안 (Proposal)	연속되어 들어오는 트랜잭션 Tx가 저장되는 Tx pool을 가진 임의의 프론트 노드는 신규 블록에 포함될 마지막 Tx를 수신하여 블록 Bx를 생성하는 시점에, 퀀텀난수 관리 Qmanager에 접속해서 퀀텀난수 생성을 요청
↓	
Qmanager 통지	Qmanager는 프론트 노드의 요청을 받고 블록생성 작업과 합의를 위한 코디네이터 및 운영위 후보군과 이들의 인식을 위한 퀀텀난수 정보를 암호화하여 이를 프론트 노드에 전송. 선정된 코디네이터에게는 운영위 후보군 정보를 전송
↓	
프론트 노드의 전체 통지	프론트 노드는 Qmanager로부터 전송받은 암호화 정보를 블록 Header에 Extra data로 구성하여 전체 노드에 브로드캐스팅
↓	
코디 및 후보군 확인	각 노드는 전송받은 블록의 Extra data에서 자기자신의 해시를 검증하고 자신의 정보인 경우에만 복호화하여 코디네이터와 후보군 확인
↓	
운영위 등록 레이싱	운영위 후보로 확인된 운영위 후보 노드들은 선착순으로 진행되는 운영위 등록 레이싱을 진행
↓	
운영위 확정	코디네이터는 선착순 15개의 노드의 운영위를 확정. 상임위 및 운영위에게 확정결과를 통보
↓	
합의 진행	상임위 및 운영위 합의 진행
↓	
블록 생성	코디네이터는 합의에 따른 블록 생성



REAP CHAIN 블록체인 네트워크에서 일부 장애로 인하여 참여 노드가 부족한 경우에도 블록생성을 위해 네트워크 내에서 반드시 합의가 이루어집니다. 특히 거버넌스에 의해서 상시 운영되는 신뢰할 수 있는 상임위 노드는 지속적으로 healthcheck를 진행하고, 상임위 노드 중에서 선정된 코디네이터가 합의 과정을 이끌어갑니다. 코디네이터가 선정된 이후에도 순간적인 장애가 발행할 확률이 있는 경우를 대비하여 2f + 1 개의 코디네이터 후보를 선정하는 방법을 고려할 수 있습니다.

▣ 코디네이터 및 운영위 선정 방법

REAP CHAIN에서는 상임위의 코디네이터 결정 및 일반노드 중에서 운영위 후보를 공정하게 결정하는데 예측 불가하며, 무편향적이고, 숫자간 무관계한 퀀텀난수를 이용합니다. Qmanager는 퀀텀난수를 이용하여 운영위 후보의 결정을 위한 후보군을 제공하지만 실제 후보가 되는 것은 어떤 노드인지를 퀀텀난수를 발생시킨 Qmanager에서는 알 수 없습니다. 실제 운영위 결정은 순간순간 블록체인 네트워크 상에서 후보군의 레이싱을 통하여 결정하게 되어 무결성을 보장하고 있습니다.

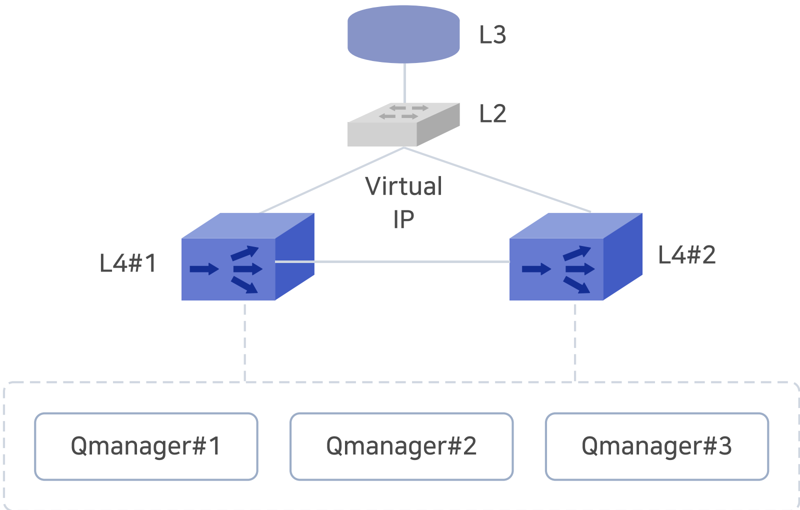
퀀텀난수를 통하여 선정된 운영위 후보군의 각 노드 공개키를 이용하여 퀀텀난수를 암호화하고 블록체인 네트워크 상에 배포되지만 운영위 후보군으로 선정된 노드만이 개인키를 사용하여 퀀텀난수를 복호화하고 확인할 수 있습니다.

복호화한 내용에는 상임위노드에서 퀀텀난수를 이용하여 선정된 코디네이터의 공개키 및 코디네이터 정보가 있습니다. 이 정보를 이용하여 퀀텀난수를 암호화하고 코디네이터에 레이싱을 통하여 전달하며, 코디네이터는 Qmanager로부터 미리 수신한 운영위 후보군의 각 노드를 위한 퀀텀난수를 비교를 수행하여 선출된 후보군의 무결성을 확인하게 됩니다.

▣ Qmanager 에 의한 선정 방식의 안정성

보안이 강화된 랜덤 선정방식을 구현하기 위하여, REAP CHAIN은 유사난수보다 보안 측면에서 우수한 퀀텀난수를 생성하는 Qmanager를 도입합니다. Qmanager는 기본적으로 퀀텀난수를 하드웨어적으로 생성하여 소프트웨어 알고리즘으로 생성하는 방식보다 안전합니다. Qmanager는 퀀텀난수를 생성 및 관리하고, 거버넌스에서 제공받은 노드 정보를 관리하면서 퀀텀난수를 이용하여 코디네이터 및 운영위 후보군을 선정하는 일을 주로 합니다.

Qmanager는 DOS 공격에 대비하고 보다 안정적인 운영을 위하여 이중화 구조를 적용하여 최대 2f+1개의 시스템으로 운영됩니다. Qmanager 가 f=1개 가 필요한 시점에는 2f+1을 적용하여 3대를 운영하여 이중화 구조를 완성합니다.



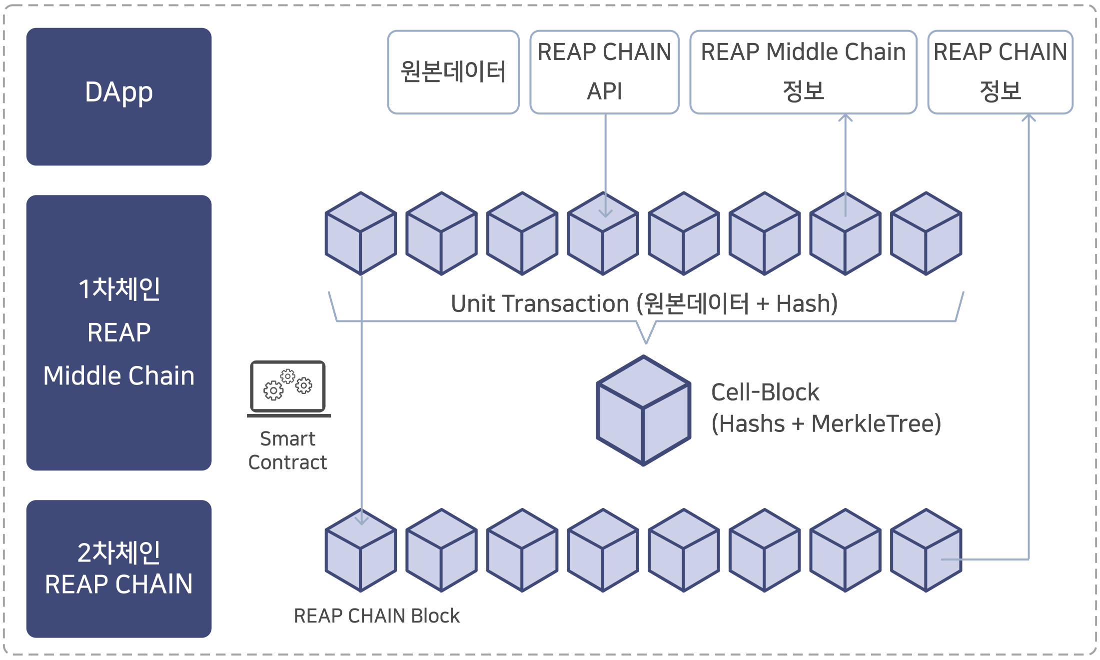
코디네이터와 운영위 후보군의 정보가 Qmanager에서 적절하게 선정된 것인지를 판단하기 위하여, 전송된 Block Extra Header에 Qmanager의 서명이 암호화되어 포함되어 있는 확인하며, 또는 코디네이터가 선정된 후에 투표에 참여할 후보군들이 Qmanager로 해당 정보를 전송하여 확인 과정을 거치게 됩니다. 따라서 현재 코디네이터가 현 라운드에 선정된 코디네이터인지, 정당한 Qmanager으로부터 선정된 것인지를 확인하고 이를 통하여 운영위 후보군의 정보를 전송 받아서 운영위 후보에게 레이싱 신호를 전송하게 되며 이때 각자는 이전에 수신한 Block Extra Header내의 각자의 정보에서 코디네이터의 공개키를 통해 이를 검증합니다. 또한 레이싱을 수행한 운영위 후보군은 각자의 정보를 전송하여 확인하게 됩니다.

Qmanager의 정보가 중간에 변조되지 않고 모든 노드에 도달되었다는 것을 보장하기 위하여, 모든 노드에 도달해야만 코디네이터가 Qmanager의 서명 정보를 포함하여 이를 Qmanager 로 전송하게 되고 이를 확인하여 중간에 변조가 되지 않은 것을 1차적으로 확인하며 이후 과정이 진행됩니다. 이 과정이 진행되지 않으면 다음 절차가 수행되지 않으며, Qmanager 서명확인 방법은 외부에 노출하지 않기 때문에 다른 Qmanager를 만들어서 생성할 수는 없습니다. 또한 무작위로 이루어지는 선정에 대해서는 최종적으로 블록을 저장하기 위해 합의시에 서명을 하기 때문에 실제 서명한 것이 인증된 블록인지 여부를 모두에게 공개되고 검증될 수 있습니다. 거버넌스에서는 공개된 블록 기록으로 Qmanager가 선정한 코디네이터 및 운영위 노드가 동일한 패턴으로 발생되는 지 여부 등 healthcheck를 진행하여 Qmanager의 안정성을 관리합니다. Qmanager의 보완이나 업그레이드가 필요한 경우에는 거버넌스의 투표 절차에 따라 승인을 얻어서 진행합니다.

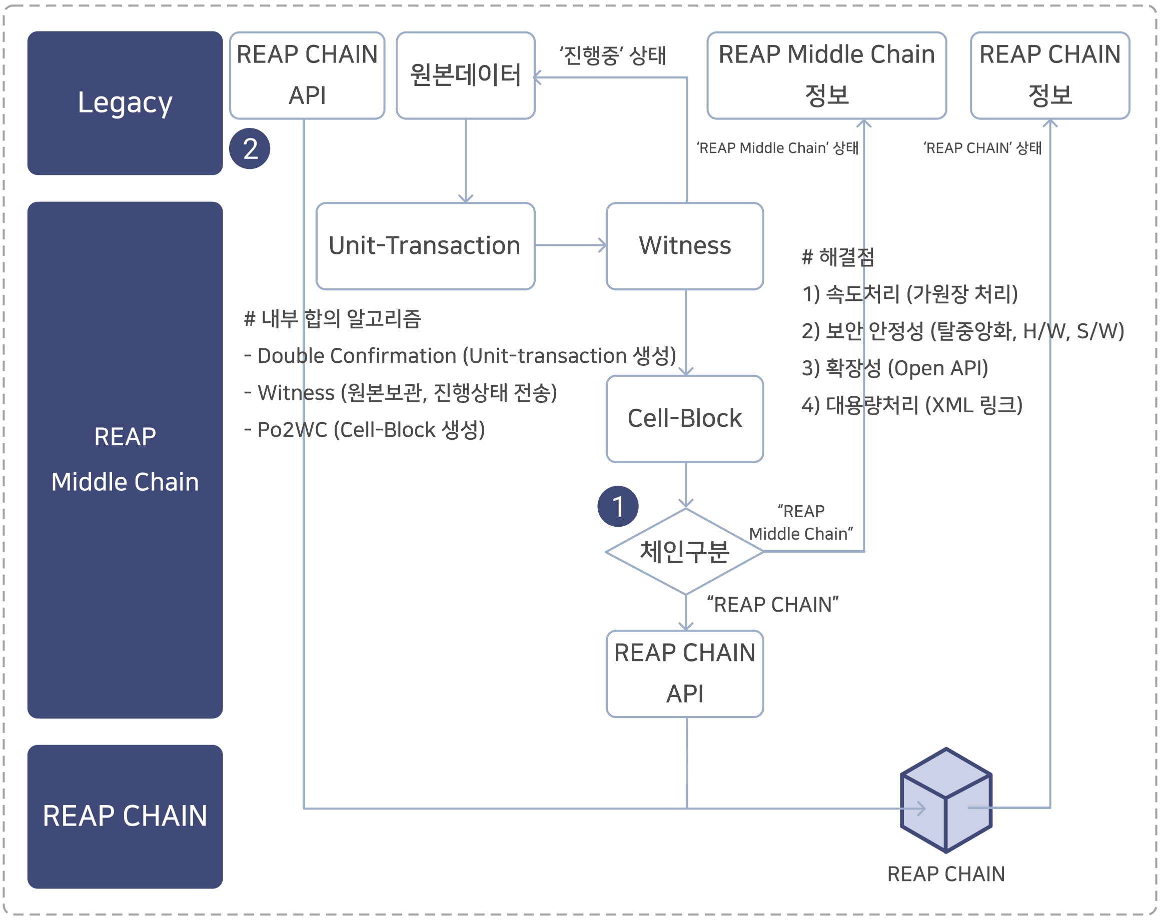
따라서 REAP CHAIN은 매번 새로운 블록을 생성하는 과정마다 새로운 투표인단을 구성하기 위하여, 퀀텀난수와 같은 예측불가능한 방법으로 코디네이터를 선정하고 또한 불특정 다수로부터 레이싱과 같은 자체 경쟁방식을 추가적으로 도입하여 운영위 노드를 구성합니다. 이를 통해 적은 숫자의 상임위 노드와 운영위 노드로 구성하여도 결국 블록 확정시에 임의성이 극대화된 민주적인 방식을 제공하기 때문에 선출의 정당성을 인정받을 수 있습니다. EOS는 21개의 적은 수의 투표 노드로 운영되어 성공적인 사례로 증명하고 있으나, 노드를 구성하기 위해서 커뮤니티에 의존하여 점차 중앙화되는 경향을 보이고 있습니다.

이중 체인 구조 및 미들웨어 Double Chain and Middleware

REAP CHAIN에서는 금융시스템에서 사용하는 임시 원장(Temporary Ledger)과 영구 원장(Permanent Ledger) 개념을 기술적으로 구현합니다.
1차적으로 REAP Middle Chain에 데이터를 저장하고 임시 원장으로 사용하고, 이후 REAP CHAIN에 블록 데이터를 기록하여 영구 원장으로 관리합니다.



REAP Middle Chain은 트랜잭션이 발생하면 그 원천데이터를 기록하여 원본을 관리합니다. 금융권 특히 증권분야에서는 주문이 발생하는 시점부터 주문 정정, 주문취소, 체결 완료 시점까지 모두 관리되면서 그 원본의 중요성이 강조되고 있습니다. 이와 동시에 블록체인 처리과정을 적용하기 위해 트랜잭션을 블록에 넣을 수 있는 형태로 관리합니다. 트랜잭션의 데이터와 트랜잭션의 해시값을 관리하고 머클트리를 관리합니다. 트랜잭션을 Cell Block이라는 미니 블록 형태로 블록체인화하여 기록하고, 이후 트랜잭션들을 단대단 보안이 적용된 API를 통해서 REAP CHAIN 플랫폼으로 전송하고 REAP CHAIN에서는 Cell Block의 트랜잭션 데이터와 해시값 및 머클트리를 참조하여 위변조 연부를 확인 후 최종 REAP CHAIN 블록체인을 완성합니다. 미들웨어 단계에서 미니 블록체인을 관리하고, 보다 확정적으로 REAP CHAIN에 블록체인을 저장하고, 이중 체인간의 위변조 여부를 확인하는 데이터 정합성을 관리함으로써 이중체인을 완성하게 됩니다. REAP Middle Chain에 기록되는 블록은 수수료가 없는 반면, REAP CHAIN 블록체인에 기록되는 블록은 수수료가 있습니다. 종합적으로 봐서, REAP CHAIN은 원천데이터, 트랜잭션 Cell Block, REAP CHAIN 블록체인 전체를 이중 체인 구조를 통해서 관리하게 됩니다. 이 모든 데이터는 보안 하드웨어에 기록되어 보안이 뛰어납니다.

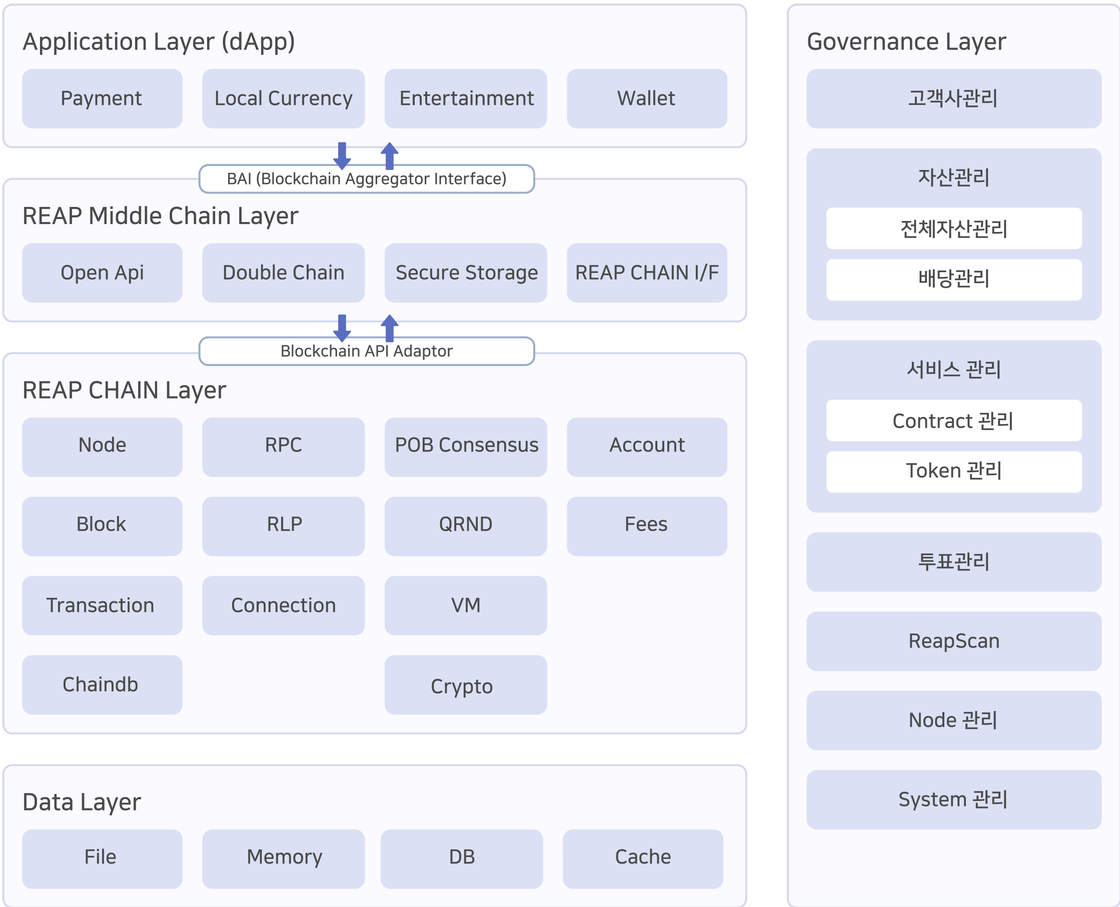


REAP CHAIN에서는 비즈니스 서비스를 제공하는 DApp의 레거시와 블록을 기록하는 메인넷 사이에, 가벼운 형태의 가속처리된 Cell Block을 처리하는 미들웨어를 두어 DApp은 독립적으로 서비스를 유지하면서 메인넷에는 부하를 줄여줍니다.

계층적인 아키텍처의 가장 큰 장점은 시스템 전체를 수정하지 않고 특정한 계층의 기능이나 성능을 개선하는 것이 용이하다는 점입니다.

REAP CHAIN은 다양한 애플리케이션과의 효과적인 연동을 위하여 다음과 같은 계층적인 아키텍처 구조를 제공합니다.

<그림 – REAP CHAIN 계층적 아키텍처 >

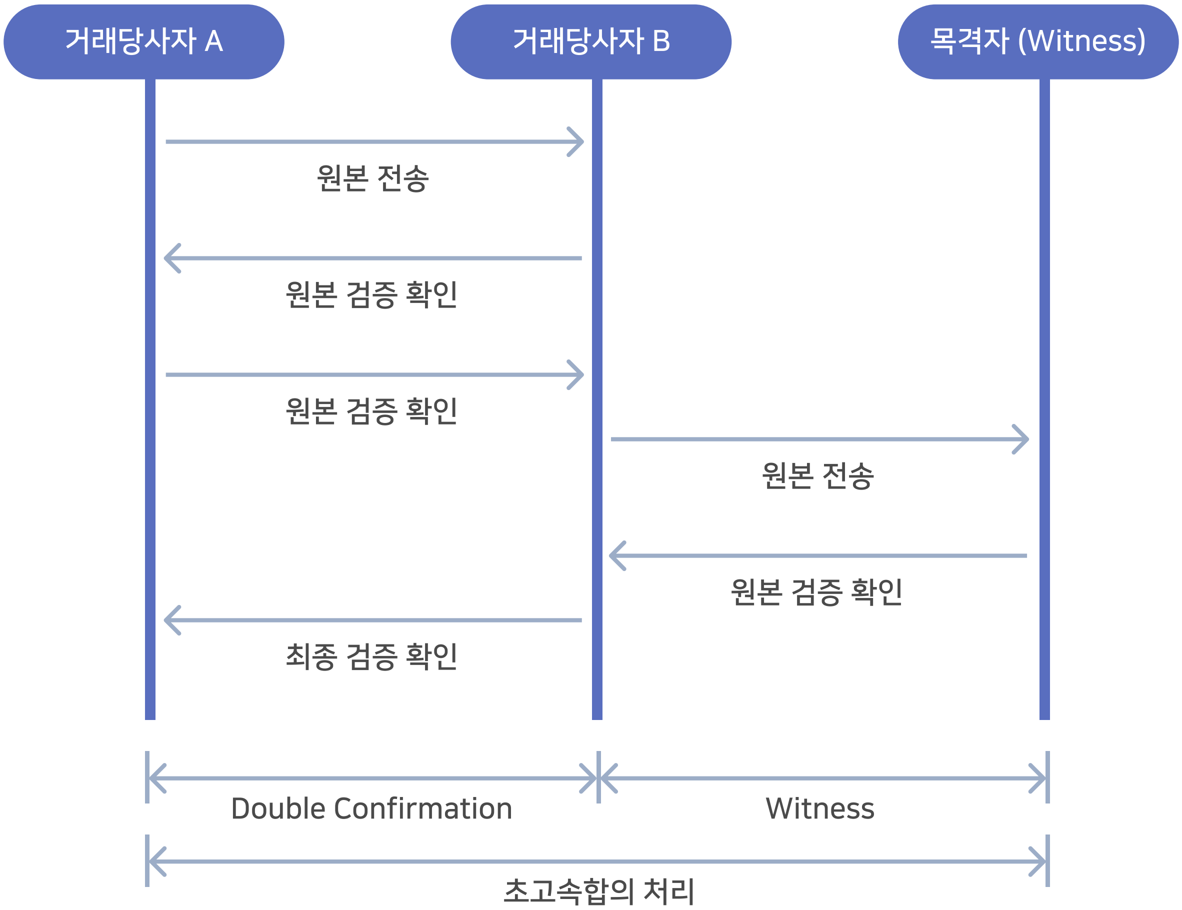


- **DApp Layer**: REAP CHAIN을 활용하는 다양한 형태의 비즈니스 애플리케이션이 제공되는 계층입니다.

- **REAP Middle Chain Layer**: 멀티 계층 트랜잭션 아키텍처를 지원하고 트랜잭션을 Cell Block으로 초고속처리하는 계층입니다. 중간계층에서 효율적이고 빠른 트랜잭션을 처리할 수 있도록 하면서, 상단 계층에는 기존 시스템 연동이 편리한 프론트 엔드 인터페이스를 제공하고, 하단 계층에서는 메인넷과 연동하여 효율적인 인터페이스를 제공합니다.
  - **BAI(Blockchain Aggregator Interface)**: 애플리케이션이 REAP CHAIN과 일관되고(Seamless) 손쉽게(easy) 연동할 수 있도록 해주는 환경을 제공합니다.
  - **Blockchain API Adaptor**: REAP CHAIN 코어 기능과 연동할 수 있도록 합니다.

- REAP CHAIN Layer
  - **Core Layer**: REAP CHAIN 핵심 역할인 합의과정, 블록 생성, 원장 관리, 토큰 관리 등으로 구성되어 있습니다.
  - **Governance**: REAP CHAIN이 유지되기 위해 필요한 수수료 배당 정책과 랜덤 사용자 노드 선발 정책, 그리고 모든 종류의 노드 관리 정책 등 관리 기능을 제공하는 특수한 계층입니다.

초고속처리



트랜잭션이 발생하면 먼저 거래 당사자 쌍방이 트랜잭션의 내용의 정확성을 2번씩 확인하는 double confirmation 과정을 거치고 바로 목격자(Witness)를 증인으로 하는 공증 과정을 거치는 3자 합의 과정을 거칩니다. 핵심 참여자로서 책임 있는 거래 당사자와 공증인이 거래에 대해 빠른 합의를 이끌어냅니다. 기본적으로 트랜잭션은 국제표준 XML 구조를 활용하기 때문에 XML parsing 과정을 통해 프로토콜의 문법 검증과 내용 특히 숫자의 정합성을 확인합니다. 트랜잭션 데이터는 해당 VIEW를 가지고 있지 않은 경우에는 열람이 불가하도록 보안 장치가 마련되어 있습니다. 트랜잭션의 내용과 해시값을 확인한 이후 임시 원장 생성을 위한 REAP Middle Chain 노드간의 합의 과정을 거칩니다. 이를 통해 연속적인 트랜잭션들이 Cell Block으로 생성됩니다. 이러한 트랜잭션과 블록들은 보안이 강화된 스토리지에 실시간으로 저장되어 보안이 보장됩니다.

합의 구조 Consensus Model

REAP CHAIN의 합의알고리즘은 퀀텀난수(QRN; Quantum Random Number)를 활용하여 구성된 상임위 및 운영위에서 투표를 진행하여 빠르게 블록을 생성합니다.

- REAP CHAIN은 상시운영되는 상임위 노드들 중 48%와 일반 노드들 중 52%가 선택 되고, 각 노드들은 합의 과정에 참여하는 블록 생성의 주체와 검증자의 역할을 합니다.
- 합의에 참여하는 노드 중에서 운영위 노드의 참여하는 비율은 52% 이상 유지하여 상시 운영되는 상임위 노드가 공정하게 운용되고 있는지 감시합니다.
- 합의 참여자 선정은 퀀텀난수를 이용하여 검증자 노드들을 공정하게 선택 배분하고 다시 퀀텀난수와 공개키를 이용하여 암호화 기능이 내재된 난수 티켓 부여 방식을 이용합니다. 해킹이 불가능한 메시지 전달 방식과 고속 인증과 합의가 구현된 하드웨어 의존적 호출함수들의 모임을 퀀텀난수 매니징 서버(Qmanager)라 명명합니다.
- 퀀텀난수 생성의 하드웨어 의존적인 특징 때문에 Qmanager는 노드와 별도로 존재합니다.
- Qmanager의 역할에 대해, 퀀텀난수를 접목한 양자확률함수기법(Quantum Random Function Mechanism)을 도입합니다. REAP CHAIN의 차별적인 특징은 QRF Mechanism에 담겨있습니다.

합의를 위한 QRF Mechanism

● QRF Mechanism: REAP CHAIN에서 합의가 이루어지는 절차는 퀀텀난수와 공개키 암호 그리고 빠른 인식기능을 위한 공개키 해시를 활용하여 이루어집니다. 일련의 인증 및 정보전달 체계를 양자확률함수기법(Quantum Random Function Mechanism)이라 명명합니다.

QRF는 퀀텀난수 서버인 Qmanager를 이용하여 기능적인 요소를 제공합니다.
Qmanager는 블록체인망에 참여한 모든 노드들의 네트워크 정보, 공개 키를 갖고 있습니다. Qmanager의 네트워크 주소는 모든 노드들에게 공개되어 있으며 체인에 참여하는 노드는 REAP CHAIN에서 지정하는 공개키 암호화를 이용하여 해당 노드의 공개키와 개인키를 생성한 후 개인키는 각 노드가 안전하게 보관하고 공개키를 Qmanager에 등록합니다.
Qmanager는 암호화된 노드 마스타 테이블을 인메모리 DB형태로 갖고 노드 아이디, 노드 네트워크정보, 공개키, 할당퀀텀난수값을 저장하고 있습니다.

<node_master_table>

NODETYPE	TRNO	NODEID	NETINFO	PUBKEY	QRND
상임위,일반구분	해당트랜잭션	아이디	네트워크주소	공개키	할당난수

- QRF Mechanism의 함수 구성

(1) **QRF_REGISTER(NODEID, PUBKEY)**: 진입 노드 또는 공개키 갱신 요청을 받은 노드가 Qmanager에 접속하여 공개키를 등록합니다.

입력: 노드 아이디, 노드의 공개키

출력: 결과 정수값

(2) **QRF_FINDKEY(NODEID)**: 노드 아이디를 이용하여 공개키를 획득

입력: 노드 아이디

출력: 노드의 공개키

(3) **QRF_GENQ(NODEID)**: 노드에게 퀀텀난수를 할당

입력: 노드 아이디

출력: 퀀텀난수

(4) **QRF_SLIST()**: 현재 블록체인망에 활동중인 모든 상임위의 아이디, 공개키, 네트워크 정보를 담고 있는 노드 인식정보nni를 배열로 리턴합니다.

입력: None

출력: 활동중인 모든 상임위의 아이디, 네트워크 정보

(5) **QRF_QRLIST()**: 현재 블록체인망에 활동중인 모든 일반 노드에서 퀀텀난수를 이용한 확률적 선택 룰을 이용하여 입력 개수에 맞는 운영위 후보들의 노드 인식정보nni를 배열로 리턴합니다.

입력: 시드, 개수

출력: 활동중인 모든 운영위의 아이디

(6) **QRF_RID**$(PK^i,\ PK^c)$: 임의의 노드 i가 공개키를 통하여 퀀텀난수를 할당받고 특정노드의 공개키와 네트워크 정보를 인식하게되는 메카니즘과 이를 위한 메시지 생성을 의미한다. 메시지는 노드 i 인식자로 명명하며 다음과 같은 구조로 표현됩니다.

입력: PK^i : 노드 i의 공개키, PK^c : 노드 c의 공개키

출력: $\left[Q^{i}_{pk^i} \mid N^{c}_{pk^i} \mid h_{pk^i}\right]$ ~ [노드 i 인식자]

$Q^{i}_{pk^i}$: 노드 i 의 퀀텀난수를 노드 i 의 공개키로 암호화한 값.

$N^{c}_{pk^i}$: 노드 c의 공개키, 네트워크정보를 노드 i의 공개키로 암호화한 값.

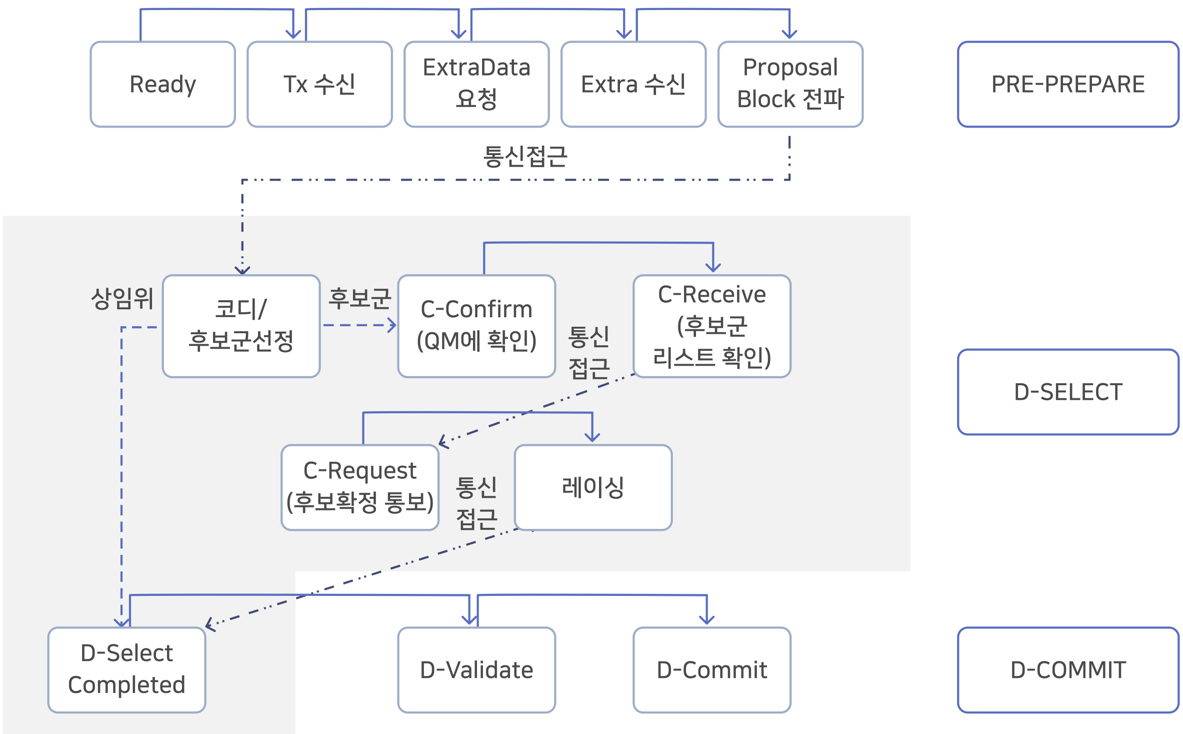
h_{pk^i} : 노드 i 의 공개키 해시값

인식과정은 먼저 h_{pk^i} 를 통하여 자신에게 할당되었음을 해시값 대조를 통하여 즉시 확인하게 됩니다. 그 다음 개인키를 이용한 복호화를 통해 할당된 퀀텀난수 Q^i 및 코디네이터 노드의 네트워크 정보 (node-net-info, nni)를 입수합니다.

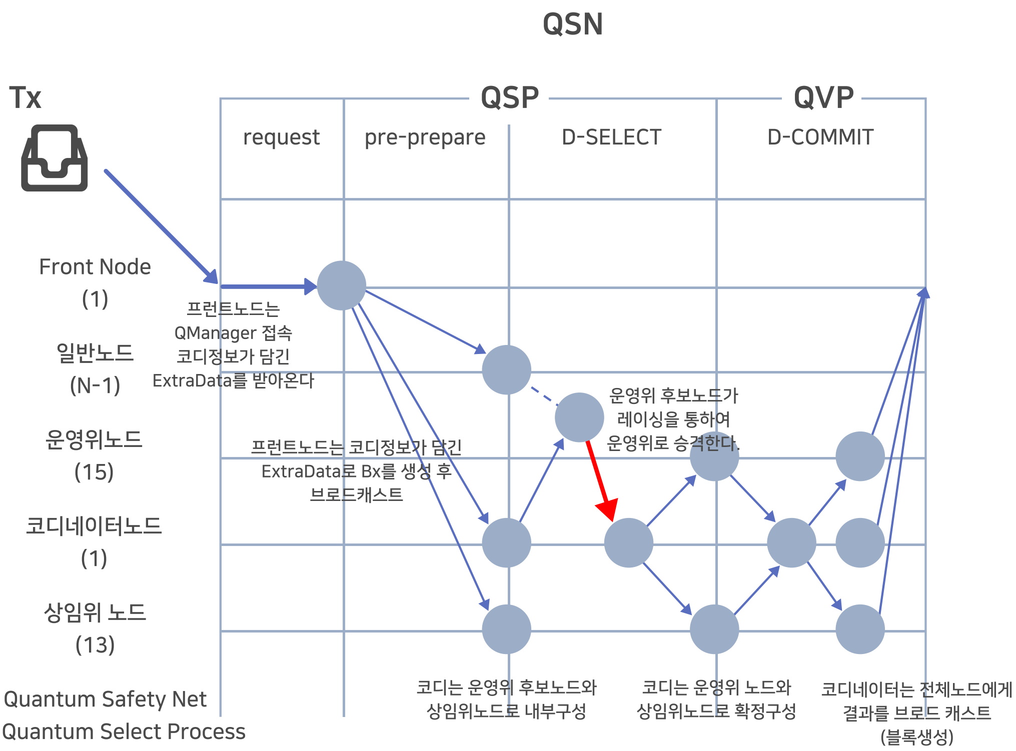
(7) **QRF_CID** $(Q^1, \cdots, Q^n,\ PK^c)$: 노드 c가 n개의 노드1,2,⋯n에게 할당된 퀀텀난수값 $Q^1, \cdots, Q^n$을 노드 c의 공개키로 암호화한 값 $Q^1_{pk^c}, \cdots, Q^n_{pk^c}$과 노드 c의 공개키 해시값 h_{pk^c} 와의 접합을 구하는 함수입니다. 코디네이터 노드가 각 운영위 후보그룹의 양자난수값을 인식하는 구조를 의미합니다.

입력: 노드1,2,⋯ n에게 할당된 퀀텀난수값 $Q^1, \cdots, Q^n$ 노드 c 의 공개키 PK^c

출력: $[Q^1_{pk_c} | Q^2_{pk_c} | \cdots | Q^n_{pk_c} | h_{pk_c}]$ ~ [코디 인식자] 각 운영위 후보의 퀀텀난수가 코디의 공개키로 암호화된 값을 접합시킨 퀀텀난수리스트와 선택된 코디네이터의 공개키 해시값으로 이루어 집니다.



<그림 - 시간 진행에 따른 합의과정>



※ QSN : Quantum Safety Net
QSP : Quantum Select Process
QVP : Quantum Voting Process

합의알고리즘 상세설명

▣ REAP CHAIN의 합의 알고리즘

● 용어 및 기술기호

- QM서버:

QM 서버는 현재 블록체인망에 연결된 모든 노드들의 네트워크 정보와 공개키를 갖고 있으며 퀀텀난수의 분포적 성질을 이용한 노드의 확률적 선택, 선택된 노드들에게 퀀텀난수 생성 후 공개키 암호화에 의한 할당, 공개키의 해시함수 생성 기능을 담당한다. 물리적, 기능적, 역할적 특징을 갖고 있습니다.

물리적 : 특정 주소를 갖는 퀀텀난수 생성기가 설치된 서버로서 미리 모든 노드에게 공개됨

기능적: (1) 퀀텀난수의 생성 (2) 암호 키의 생성 (3) 확률적 선택 (4) 블록 검증 함수가 구현되어 있다. 이를 QM서버의 기능 함수라 정의함

역할적: 기능함수들이 결합되어 (1) 코디네이터의 선정 (2) 운영위 후보군의 선정을 담당하는 역할(Roll)을 구현함

- QRND: 퀀텀난수발생기에서 생성한 난수를 의미합니다. Qmanager의 라이브러리 함수 GETQRNARR()을 이용하여 49151바이트 이내의 원하는 크기의 바이트 난수를 한번에 생성 가능합니다. 비트연산을 적용하여 큰 자연수 N이내의 균등분포를 얻을 수 있습니다.
- 트랜잭션: $tx_i,\ i = 0,1,\cdots$
- 블록: $Bx = (\cdots,(tx_0,tx_1,\cdots),\cdots)$
- 노드 n_* : 프런트(단일) n_F, 코디네이터(단일) n_C , 상임위(복수) n_S , 운영위후보군(복수) n_{RR} , 운영위 n_R, 일반(복수, 상임위제외) n_G

- PK^i : 노드 i의 공개키
- pk^j : 객체의 아래첨자로 사용되어 해당 객체를 노드 j의 공개키로 암호화함을 의미함
- Q^i : 노드 i에게 할당되는 퀀텀난수
- $Q^i_{pk^j}$: 노드 i에게 할당되는 퀀텀난수 Q^i 의 노드 j 의 공개키로 암호화된 문자열
- h_{pk^j} : 노드 j의 공개키 해시 값
- N^i : 노드 i의 인식 정보(node-net-info nni): 노드의 아이디(공개키), 네트워크 정보
- $N^i_{pk^j}$: 노드 i 의 인식 정보를 노드 j의 공개키로 암호화한 값

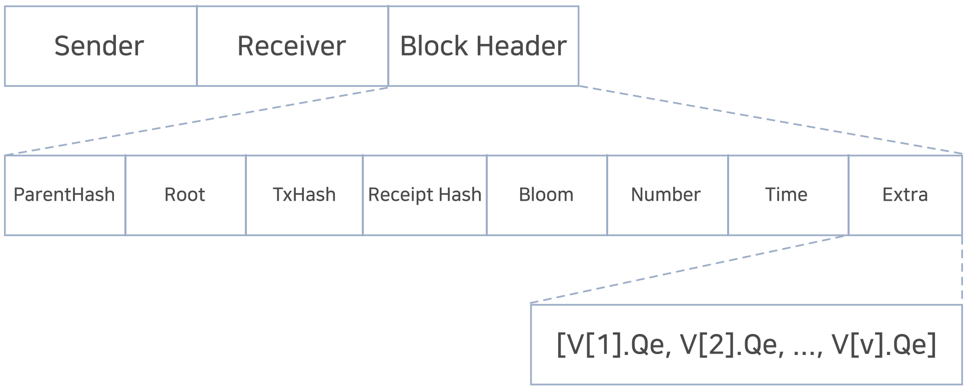
위의 기술에서 첨자 i 또는 j가 c 이면 코디네이터 노드를 뜻함

● PoDC 알고리즘의 용어 및 상태 정의

- 환경: E는 QM서버, n (노드: 프런트, 코디, 상임위, 운영위후보, 운영위, 일반)
- 행위(액션, A): 블록체인망내부에서 발생하는 블록의 생성과 관련하여 환경이 관련된 모든 능동적 또는 수동적인 행위를 의미
  1) 행위자Actor: $QM, n_F, n_C, n_S, n_{RR}, n_R, n_G$
  2) 수용자 Receiver: $Bx, QM, n_F, n_C, n_S, n_{RR}, n_R, n_G$

     Action에 의하여 수용자의 상태 변화가 발생합니다. 진행순서대로 0을 포함하는 자연수 깊이 (Depth)로 나타냅니다.

- 메시지의 구조: 갱신되기를 요구하는 블록의 ExtraData항목에는 액션이 필요한 노드들의 정보를 담아서 전파됩니다.
  - ■ 송신(Sender): 송신하는 노드의 정보
  - ■ 수신(Receiver): 수신하는 노드의 정보
  - ■ 블록 헤더(Block Header): 앞으로 생성할 블록의 헤더
    - ◆ ParentHash: 부모 블록 헤더의 해시값
    - ◆ Root: 어카운트의 상태 정보가 모여 있는 머클 패트리시아 트리의 루트 노드에 대한 해시값
    - ◆ TxHash: 현재 블록 내에 모든 트랜잭션의 머클 트리의 루트 노드에 대한 해시값
    - ◆ ReceiptHash: 현재 블록 내 모든 트랜잭션의 리시트들의 머클 트리의 루트 노드에 대한 해시값
    - ◆ Bloom: 로그 정보들을 검색하는 데 사용하는 32바이트 블룸 필터
    - ◆ Number: 현재 블록의 번호(제네시스 블록은 블록 번호가 0번으로, 이후에 1씩 증가)
    - ◆ Time: 현재 블록의 최초 생성 시간을 기록하여 블록들의 체인에 연결될 때 시간 순으로 연결되는 기준으로 사용되며, 블록 생성 시간을 임의로 조작할 수 없도록 함
    - ◆ Extra: 현재 블록과 관련된 기타 추가 정보
      - ● V[1],V[2], ⋯ , V[v]: v 명의 운영위 후보
      - ● V[i].Qe: 운영위 후보 i 의 공개키로 암호화한 운영위 후보 i의 QRND와 코디네이터 정보, 운영위 후보 i의 공개키의 해시값을 갖는 맴버 변수



Stage 4 메세지 예시

- 상태: S, 상태의 변화는 블록의 변화와 환경(QM서버, 노드)의 변화로 나눕니다.

 상태함수1: $S(QM,\ n_F,\ n_C,\ n_S,\ n_{RR},\ n_R,\ n_G)$, 각 변수는 환경의 상태가 입력이 되며 결과는 깊이로 표현되는 0보다 크거나 같은 정수 값을 갖는 상태입니다.

 상태함수2: $S(Bx)$, 블록의 상태변화에 따른 함수 (1) 블록 제안 (2) ExtraData의 추가 (3) 블록갱신의3가지 상태가 존재

- 스테이지(Stage) 또는 깊이(Depth): 하나의 블록이 상태변화가 완성될 때까지 일어나는 모든 상태를 의미합니다.

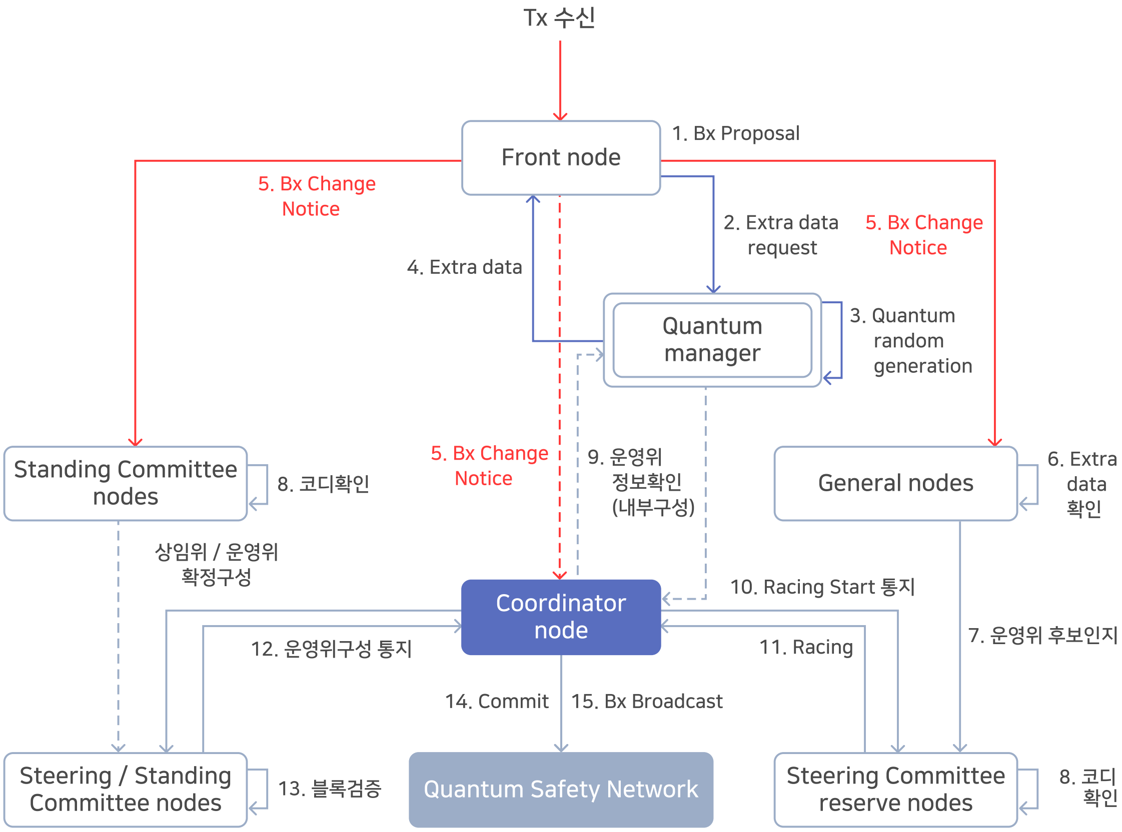
깊이	행위자	수용자	환경상태	블록상태
0	-	all	대기상태	
1	-	n_F	프런트 노드의 블록의 상태변화 제안	블록의 제안
2	n_F	QM	프런트 노드에 의한 Qmanager로의 블록 전송	
3	QM	-	Qmanager의 코디 및 운영위 후보군 선정 ⇒ ExtraData 생성	
4	QM	n_F	Qmanager에 의하여 프런트 노드로 ExtraData 전송,	블록의 ExtraData갱신
5	n_F	all	프런트 노드에 의한 갱신된 블록의 브로드 캐스팅	
6	-	n_C ,n_{RR}	1차 선출자 인지(코디, 운영위 후보들)	
7	n_C	QM	코디네이터(코디) 등록(선정)	
8	n_C	n_S ,n_{RR}	코디: 내부망 구성	
9	n_C	n_{RR}	코디: 레이싱 통지	
10	n_C	n_{RR}	코디: 운영위 등록	
11	n_C	n_S,n_R	코디: 운영위 확정	
12	n_S ,n_{RR}	n_C	블록 검증	
13	n_C	all	블록 전파	블록의 갱신

- 환경 객체의 상태 전이도

QM	nf	nc	ns	nrr	nr	ng
-	-	-	-	-	-	-
-	1	-	-	-	-	-
2	2	-	-	-	-	-
3	-	-	-	-	-	-
4	4	-	-	-	-	-
0	5	5	5	5	5	5
-1	-1	6	-	6	-	-
-1	-1	7	-	-	-	-
-	-1	8	8	8	-	-
-	-1	9	-	9	-	-
-	-1	10	-	-1	10	-
-1	-1	11	11	-1	11	-
-1	-1	12	12	-1	12	-
-1	-1	13	13	-1	13	13

- : 대기상태, 1,…,11 : 위에서 기술한 각각의 상태, -1 : 소멸상태

환경의 처리 단계 전이 상세:



1. 대기상태 : 새로운 블록의 상태변화 제안을 기다립니다.

2. 블록 상태 변경 요청 제안(T-Proposal): 프런트 노드는 제안자(Proposal)로부터 상태 변경을 요청하는 트랜잭션(Tx)을 수신합니다.

3. 프런트 노드는 블록 생성 전에 Qmanager에 접속하여 합의를 위한 ExtraData 생성 요청 (자신의 서명이 포함됨)을 하고 수신을 기다립니다.

4. Qmanager (Q-Work): Qmanager는 합의를 위한 ExtraData생성 작업을 합니다. 구체적으로 코디 및 운영위 후보군과 이들의 인식을 위한 퀀텀난수정보를 암호화하여 이를 ExtraData로 만든 다음 대기하고 있는 프런트 노드에 전송합니다.

   Qmanager 통지(Q-Request): Qmanager는 Extra Data를 생성 후 프런트 노드의 공개키를 이용하여 암호화한 후 전송하고 프런트 노드에 전송 프런트 노드는 자신의 개인키를 이용하여 자신의 서명값을 검증을 통해서 Extra Data를 검증합니다.

5. 프런트 통지(F-Request): 프런트 노드는 Extra Data를 검증한 후 이상이 없으면 제안된 블록의 Extra Data항목을 추가 업데이트하여 전체 노드를 대상으로 신규 Bx를 전송합니다.

6. 코디 및 운영위 후보군 선정
   - 코디선정: 각 (상임위)노드는 Extra Data의 마지막 항목인 [퀀텀난수리스트 인식자]를 복호화 하여 먼저 공개키 해시값을 확인하여 본인의 코디 여부를 확인합니다. 확인된 코디는 항목의 내용을 공개키로 복호화하여 운영위 후보군의 퀀텀난수 리스트를 확보합니다.
   - 운영위 후보군 선정: 각 (일반)노드는 전송 받은 블록의 ExtraData를 복호화 하여 코디의 네트워크 정보(node net-info, nni)와 해당 노드가 후보군에 속하는지를 확인 한 후 운영위 후보 군으로 확인된 노드는 코디에게 운영위 등록 작업을 준비합니다.

7. 코디 확인(C-Confirm): 확정된 코디는 Qmanager에 접속하여 코디로 등록되었음을 통지 및 확인

8. 코디 통지(C- Request): 코디는 대상 노드 (상임위 + 운영위 후보군)들의 공개 키를 사용하여 코디로 선정되었음을 통지하고 내부망을 구성합니다.

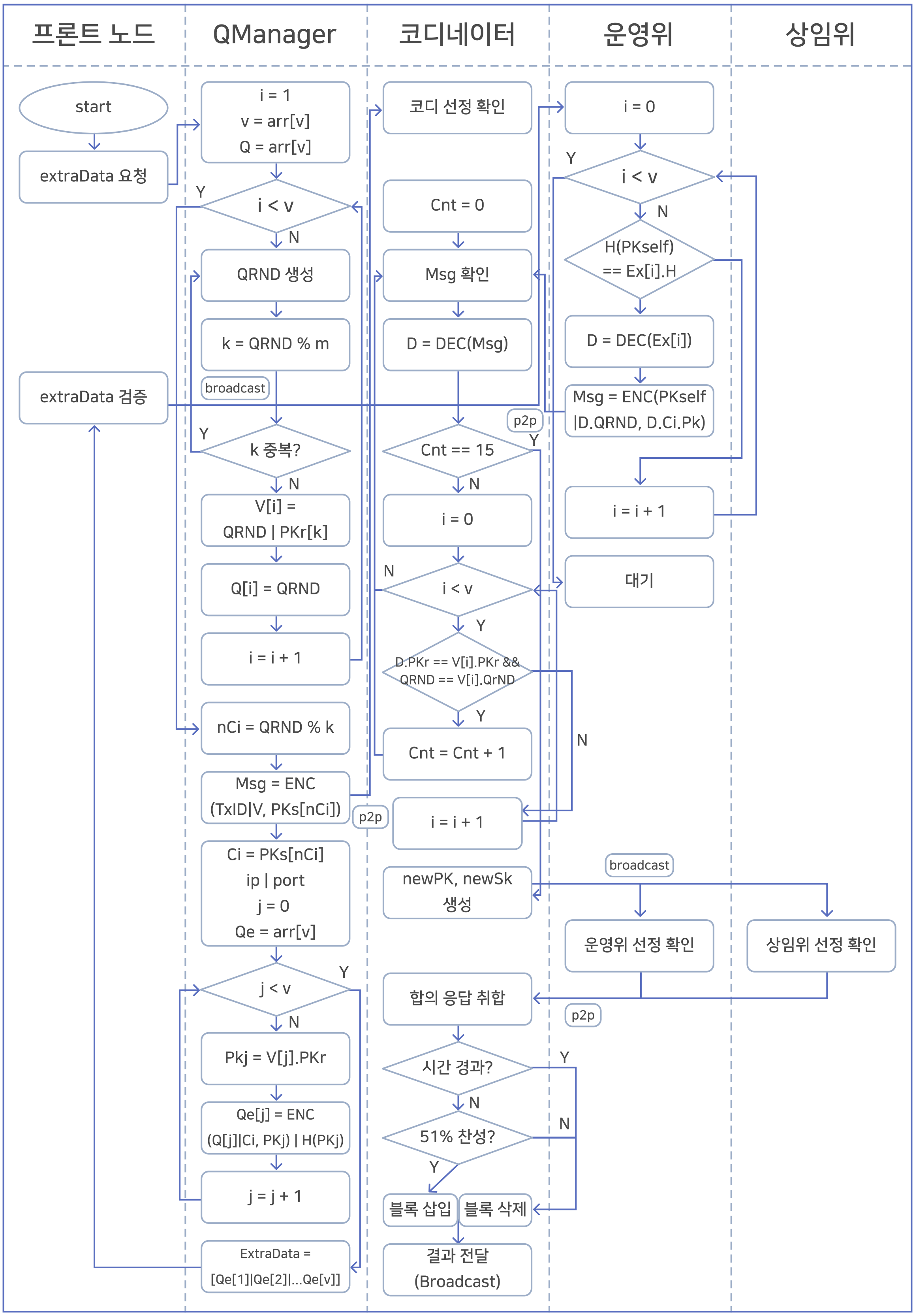
9. 레이싱: 코디에게 운영위 등록 통지를 받은 즉시 운영위 후보노드들은 선착순으로 진행되는 운영위 등록(레이싱)을 진행합니다.

10. 운영위 확정(D-Select Completed): 코디는 등록된 노드들을 검증한 후 선착순 15노드의 운영위를 확정 합니다.

11. 블록 검증(D-Validate): 검증자 노드들(상임위 + 운영위)은 검증 후 코디에게 블록 검증 메시지를 통보하고 코디는 2/3의 검증이 완료되는지 기다립니다.

12. 검증(D-Commit): 코디는 검증 노드들에게 검증이 이상없음으로 보고되면 블록의 검증이 완료된 결과(즉, 제네시스 블록으로부터 현재 블록까지의 해쉬검증)를 전체 노드에 전파합니다.

<합의알고리즘 순서도>


프론트 노드
QManager
코디네이터
운영위
상임위
start
extraData 요청
extraData 검증
i = 1
v = arr[v]
Q = arr[v]
i < v
Y
N
QRND 생성
k = QRND % m
broadcast
k 중복?
V[i] =
QRND | PKr[k]
Q[i] = QRND
i = i + 1
nCi = QRND % k
Msg = ENC
(TxID|V, PKs[nCi])
p2p
Ci = PKs[nCi]
ip | port
j = 0
Qe = arr[v]
j < v
Pkj = V[j].PKr
Qe[j] = ENC
(Q[j]|Ci, PKj) | H(PKj)
j = j + 1
ExtraData =
[Qe[1]|Qe[2]|...Qe[v]]
코디 선정 확인
Cnt = 0
Msg 확인
D = DEC(Msg)
p2p
Cnt == 15
i = 0
i < v
D.PKr == V[i].PKr &&
QRND == V[i].QrND
Cnt = Cnt + 1
i = i + 1
newPK, newSk
생성
합의 응답 취합
시간 경과?
51% 찬성?
블록 삽입
블록 삭제
결과 전달
(Broadcast)
i = 0
i < v
H(PKself)
== Ex[i].H
D = DEC(Ex[i])
Msg = ENC(PKself
|D.QRND, D.Ci.Pk)
i = i + 1
대기
broadcast
운영위 선정 확인
상임위 선정 확인
p2p

트랜잭션 및 블록생성 Transaction and Block Building

트랜잭션에 확정성(finality)를 부여하기 위해서 REAP CHAIN은 이중체인 구조를 적용하여 임시 원장의 Cell Block 이후에 영구 원장의 블록생성으로 최종 확인되며 트랜잭션이 포함된 블록은 보안 하드웨어에 저장되어 바뀔 수 없습니다.

블록의 크기는 동적(dynamic)으로 변화될 수 있으며 트랜잭션의 기대치를 나타내는 TPS는 RC Node의 성능 향상을 통해 높은 수치의 TPS 달성이 가능합니다.

< 표 – 타 블록체인의 블록생성 주기 및 확정회수 비교>

	컨센서스	블록생성주기(초)	확정횟수	확정시간(초)
EOS	DPoS	3	15	45
STEEM	DPoS	3	15	45
NEO	dBFT	15 ~ 20	1	15 ~ 20
이더리움	PoW	14	12	180
비트코인	PoW	600	6	3,600
REAP CHAIN	PoDC	2	12	24

REAP CHAIN의 상시운영노드가 전체 합의 노드 중에서 48%를 차지하고 있고, 사용자는 노드는 예측할 수 없도록 퀀텀난수를 이용한 랜덤으로 선발되기 때문에 악의적인 사용자 노드의 비율이 52% 를 지속적으로 점유하는 경우는 통계적으로 지극히 낮다고 할 수 있습니다.

▣ REAP CHAIN 노드의 구성

REAP CHAIN은 역할과 권한에 따라 다음과 같이 다양한 형태의 노드로 구성됩니다.

1. **상임위원회 노드**: REAP CHAIN에서 상시 운영되는 노드로서, 합의 과정에 참여하여 블록 생성 권한을 가지고 있음
2. **일반 노드**: 누구나 설치할 수 있는 노드로서, 퀀텀난수를 이용한 랜덤 선정방식에 의해 트랜잭션 검증 과정에 선택적으로 참여할 수 있음. 트랜잭션 검증에 참여하는 노드를 운영위원회 및 검증인(Validator)로 정의함
3. **거버넌스 노드**: 수수료 배당 정책과 랜덤 사용자 노드 선발 정책, 그리고 모든 종류의 노드 관리 정책 기능 등을 포함된 특수한 노드임
4. **Qmanager**: 퀀텀난수를 생성 및 관리하고, 거버넌스에서 제공받은 노드 정보를 활용하여 퀀텀난수로 코디네이터 및 운영위 후보군 선정 과정을 지원함

< 표 – 노드 종류별 역할과 권한 >

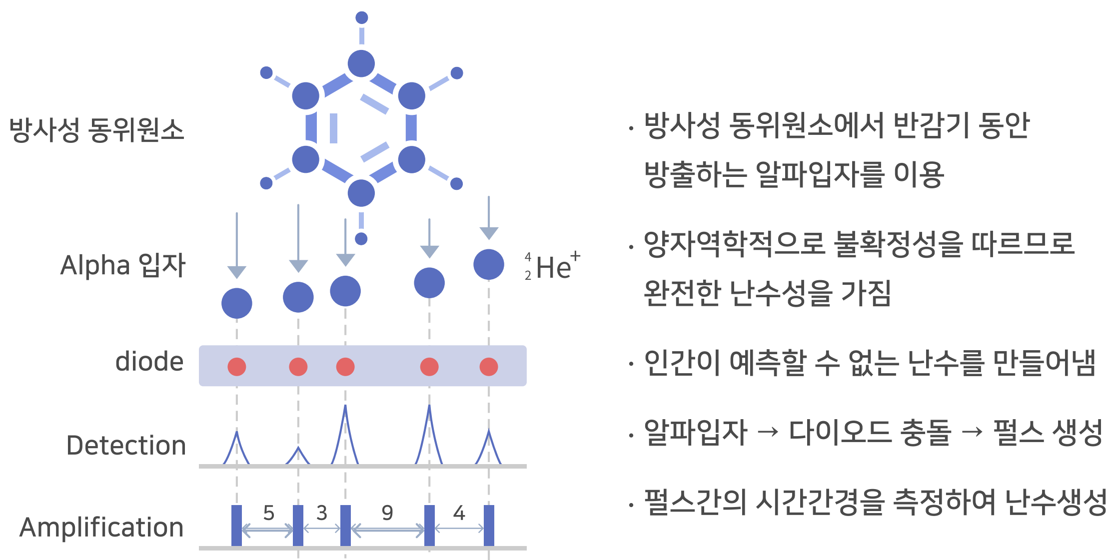
	Qmanager	거버넌스 노드	상임위원회노드	일반노드 (Validator)	일반노드
노드 관리	X	○	X	X	X
블록 생성 권한	X	X	○	X	X
합의 참여	X	X	○	○	X
트랜잭션 처리	X	X	○	○	○
블록 검사	X	X	○	○	○

Quantum Random Number를 통한 보안 강화 Randomness

REAP CHAIN의 합의 과정에 참여하는 노드들은 퀀텀난수 생성기 (QRNG; Quantum Random Number Generators)를 통해 예측할 수 없는 방식으로 선택됩니다.

암호기술을 적용하는 보안 분야에서 난수생성기의 난수 패턴 예측은 보안에 큰 영향을 미칠 수 있기 때문에 안정적인 난수생성기의 역할은 매우 중요합니다. 특히 퍼블릭 블록체인과 같이 개방된 시스템에서는 암호화 키 생성과 유효성 검사 과정 등은 쉽게 예측할 수 없어야 합니다.

Micro QRNG의 원리

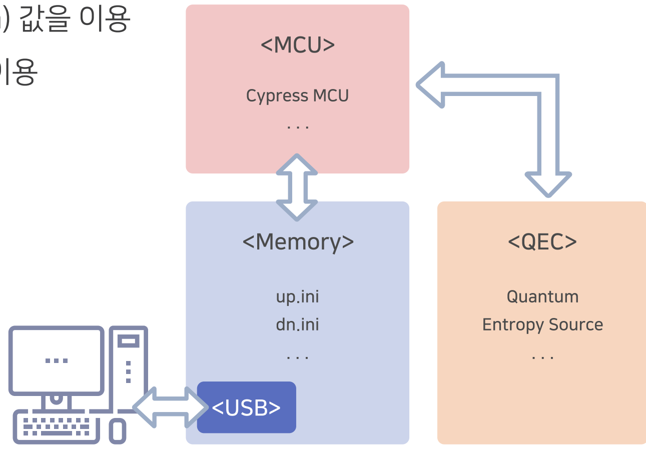


*펄스간의 시간간격 측정 : Random number 5, 3, 9, 4

암호화 시스템을 구성하는 데 필수적인 조건은 예측 불가능해야 하며(Unpredictable), 무편향성(Unbiased), 그리고 숫자간 무관성(Uncorrelated)이 보장되어야 합니다. 컴퓨팅 기술이 발전하면서 수학적인 알고리즘으로 만들어 내는 유사난수(Pseudo Random Number)가 많이 사용되고 있으나, 유사난수를 사용한 암호화 기술은 암호화의 패턴을 찾아내는 해킹 공격에 취약합니다.

기존 유사난수 생성기(PRNG; Pseudo Random Number Generator) 방식의 경우,
다음과 같은 암호화 패턴으로 시드(seed)값을 추적할 수 있습니다.

- 과거 블록 (old block)의 블록해시(blockhash) 값을 이용
- 현재 블록 (current block)의 블록해시 값을 이용
- 마지막 블록의 블록해시 값을 이용
- 블록 변수 (block variables)를 이용



유사난수생성 방식과 비교해 보았을 때, 퀀텀 역학적인 불확정성의 성질을 이용한 QRNG는 진정한 의미의 RNG로 볼 수 있으며, REAP CHAIN은 이러한 QRNG를 이용하여 컨센시스 과정에서 필요한 보안성을 강화합니다. 기술이 더욱 발전하면 블록체인에서 합의 과정은 랜덤 증명(PoR; Proof of Randomness)을 수행해야 하는 날이 올 수도 있습니다. 립체인은 그 진화과정을 선도합니다.

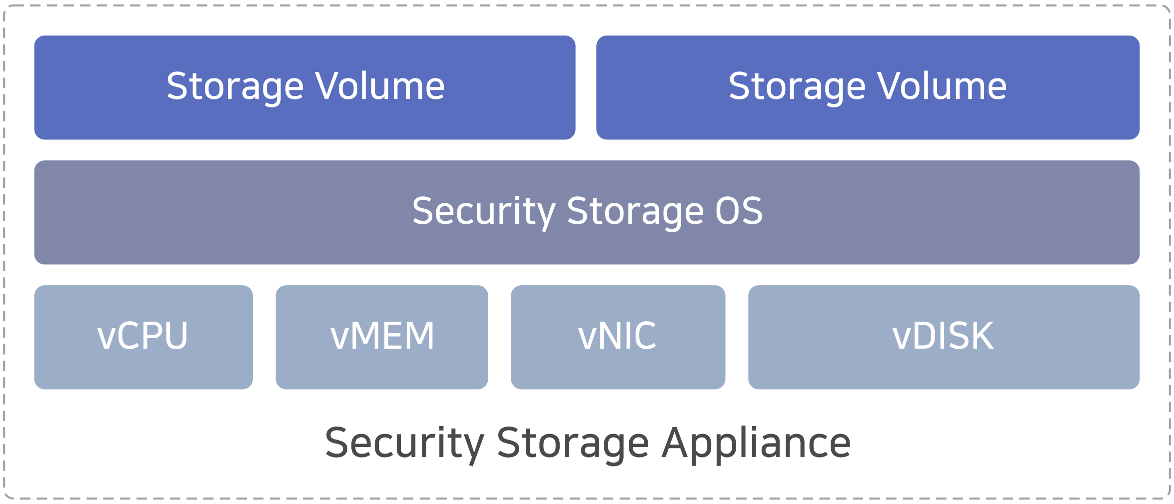
REAP CHAIN에서는 상임위의 코디네이터 결정 및 일반노드 중에서 운영위 후보를 공정하게 결정하는데 예측 불가하며, 무편향적이고, 숫자간 무관계한 퀀텀난수를 이용합니다. Qmanager는 퀀텀난수를 이용하여 운영위 후보의 결정을 위한 후보군을 제공하지만 실제 후보가 되는 것은 어떤 노드인지를 퀀텀난수를 발생시킨 Qmanager에서는 알 수 없습니다. 실제 운영위 결정은 순간순간 블록체인 네트워크 상에서 후보군의 레이싱을 통하여 결정하게 되어 무결성을 보장하고 있습니다.

퀀텀난수를 통하여 선정된 운영위 후보군의 각 노드 공개키를 이용하여 퀀텀난수를 암호화하고 블록체인 네트워크 상에 배포되지만 운영위 후보군으로 선정된 노드만이 개인키를 사용하여 퀀텀난수를 복호화하고 확인할 수 있습니다.

복호화한 내용에는 상임위노드에서 퀀텀난수를 이용하여 선정된 코디네이터의 공개키 및 코디네이터 정보가 있습니다. 이 정보를 이용하여 퀀텀난수를 암호화하고 코디네이터에 레이싱을 통하여 전달하며, 코디네이터는 Qmanager로부터 미리 수신한 운영위 후보군의 각 노드를 위한 퀀텀난수를 비교를 수행하여 선출된 후보군의 무결성을 확인하게 됩니다.

보안 하드웨어에 데이터 저장

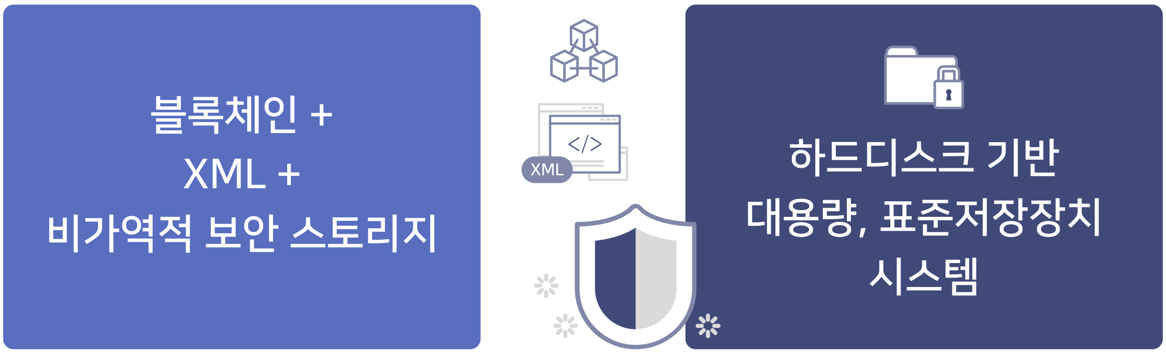
REAP CHAIN은 블록체인에 특화된 보안 스토리에 원본성을 확인할 필요가 있는 로그, 트랜잭션, Cell Block, 시스템 사용기록 등의 데이터를 저장합니다. 데이터가 기록된 DVD에 기록된 데이터를 지울 수가 없듯이 보안 스토리지에 기록된 REAP CHAIN 데이터는 위변조 및 삭제가 원천적으로 불가능합니다. 이를 위해 블록체인 데이터에 대한 최적의 보안 기술을 적용합니다. 또한 접근권한이 없이는 저장된 데이터에 대한 접근이 불가합니다.



REAP CHAIN에 적용된 보안 스토리지 기술은 Read-Only 저장 매체인 광디스크와 같은 개념을 스토리지 장비에 적용한 것으로, 특히 컴퓨팅 자원관리 및 권한관리를 수행하는 OS 수준의 보안기술을 적용하였습니다. 결국 블록체인의 소프트웨어적인 개념과 보안 스토리지의 하드웨어적인 것을 결합하여 보다 강력한 위변조 시도를 차단합니다.

기존 블록체인에서 주장하는 것은 데이터를 포함한 블록 간의 블록해시를 활용하여 데이터간의 견고한 연결고리를 만들면 위변조가 불가능할 것으로 여긴다는 것입니다. 실제 컴퓨팅 파워가 강력한 해커 세력이라면 위변조가 가능할 수 있습니다. 또한 데이터 블록을 삭제하여 서비스를 지연하거나 불능상태로 만들 수 있습니다.

REAP CHAIN의 보안 데이터 관리 기술은 블록해시 및 암호화된 데이터를 특수 스토리지에 저장하여 절대적으로 접근제어 및 해킹 행위 자체를 원천적으로 차단합니다. 블록체인의 데이터 블록해시를 적용하고, 데이터를 암호화하고, 물리적인 하드웨어에 저장하여, 절대 비가역적 3중 보안 처리를 실현하였습니다.



REAP CHAIN에서는 국가기록, 금융거래내역, 역사적 증권데이터, 학생부기록 및 졸업증명 등 전쟁, 자연재해, 해킹 등으로부터 안전하게 원본 데이터를 보관할 수 있으며, 랜섬웨어 해킹과 같은 부분에서도 접근 권한을 원천적으로 통제하여 원본 데이터에 대한 장악력을 해킹 세력에게 빼앗기지 않습니다.

수익 및 배당 Revenue and Dividend

일반적으로 PoW(Proof-of-Work)방식은 블록을 채굴한 노드들만이 보상을 받으며, PoS(Proof-of Stake)방식은 암호화폐를 일정규모 이상 예치해야만 보상을 받는 구조이나 ReapChain은 자체 토큰인 REAP을 상임위에 위임을 통한 생태계에 참여를 통해 보상을 받게 됩니다.

서비스 공급사의 서비스에서는 다음과 같은 종류의 트랜잭션이 발생할 수 있습니다.

- Reap Pay 에서 발생되는 트랜잭션
- 중고거래 플랫폼에서 발생되는 트랜잭션
- Reap Bank 에서 발생되는 트랜잭션
- REAP CHAIN 범용 투표시스템 트랜잭션
- 기타 REAP CHAIN에 블록체인을 사용하는 모든 트랜잭션

총 수수료 수입은 기본 수수료와 추가 수수료 및 컨트랙트 수수료로 구성되며, 다음과 같습니다.

{총수수료 = 0.02 Reap + (0.02Reap × OpCode) + 10,000 Reap}

REAP CHAIN은 생태계 활성화를 위해 Reap 보유자와 상임위원회, 운영위원회 및 후보군에 대해서도 배당금을 지급합니다. 배당금은 총 수수료 수입이며, 수수료 적립액 및 블록주기 등 배당조건이 충족되는 경우 Reap으로 배당합니다. 배당 시점에 배당금이 배당 상한액을 초과하는 경우 모두 합산하여 지급합니다.

총 배당금의 배당비율은 전체 Reap 보유자에 대해 70%, 운영위원회 및 후보군 20%, 상임위원회 10%로 하며, 중복지급을 허용합니다. 배당금의 산정방식은 다음과 같습니다.

▣ 전체 Reap 보유자

$$Dividend(n) = \sum_{i=1}^{m} D_i \times 0.7 \times \frac{B(n)}{R_{Issue}}$$

$\mathrm{Dividend}(n)$: $Dividend\ of\ user\ n$

D_i : $Total\ Dividedn\ of\ i-th\ block$

$\mathrm{B(n)}$: $Balance\ of\ user\ n(The\ total\ reap\ amount\ of\ user\ n)$

R_{Issue} : $The\ amount\ of\ Reap\ Issue\ Amount$

m : $Block\ number\ per\ dividend\ cycle$

▣ **상임위원회(Standing Committee) 노드 14개**

$$Dividend(cn) = \frac{\sum_{i=1}^{m} D_i \times 0.1}{\sum_{i=1}^{14} C_i}$$

$\text{Dividend}(cn) : Dividend\ of\ standing\ committee\ n$

$D_i :\ Total\ Dividedn\ of\ i - th\ block$

$C_i :\ Number\ of\ standing\ committee\ i = 1 \cdots 14$

$\text{m} : Block\ number\ per\ dividend\ cycle$

▣ **운영위원회(Steering Committee) 15개 노드 및 후보군**

$$Dividend(scn) = \frac{\sum_{i=1}^{m} D_i \times 0.2}{\sum_{i=1}^{m} SC_i}$$

$\text{Dividend}(scn) : Dividend\ of\ steering\ committee\ n$

$D_i :\ Total\ Dividedn\ of\ i - th\ block$

$SC_i :\ Total\ Number\ of\ standing\ committee\ i = 1 \cdots m$

$\text{m} : Block\ number\ per\ dividend\ cycle$

총 수수료 수입에 대한 배당과 별도로 Reap 총 발행량 중 10%는 생태계 참여에 대한 보상으로 특정 블록 주기 생성 시 전체 Reap 보유자들에게 개별 Reap 보유량에 비례하여 지급하며, 배당 산식은 다음과 같습니다.

$$\text{A Supplementary Dividend(n)} = R_i \times \frac{B(n)}{R_{Issue}}$$

$\text{A Supplementary Dividend(n)} : A\ Supplementary\ Dividend\ of\ user\ n$

$\text{R}_\text{i} : Total\ Reap(Reap\ Chain\ Coin)\ Issue\ Amont\ of\ i - th\ block$

$\text{B(n)} : Balance\ of\ user\ n(The\ total\ reap\ amount\ of\ user\ n)$

$\text{R}_{Issue} : The\ amount\ of\ Reap\ Issue\ Amount$

$$R_i = \sum_{n=1}^{i} RB_n$$

$$RB_n = \frac{0.1 \times \sum_{i=1}^{n} RB_i}{Total\ number\ of\ Block}$$

$$\text{Total number of Block} = \frac{10\ years}{Block\ generation\ Cycle}$$

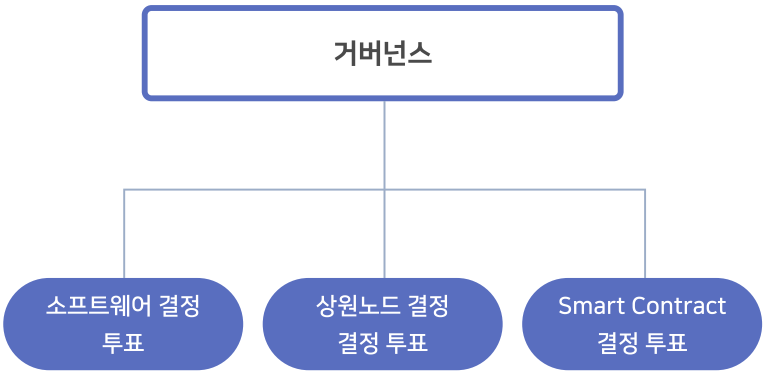
$RB_n: Reap\ Issue\ Amount\ of\ per\ block$

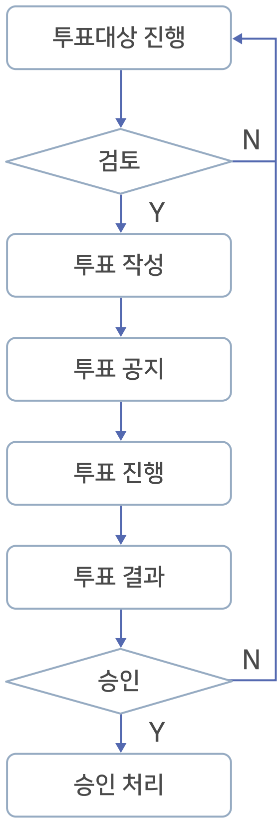
$RB_i: Reap\ Issue\ Amount\ of\ block\ i = 1 \cdots n$

거버넌스 Governance

REAP CHAIN은 상시 운영되는 상임위 노드들의 합의에 의하여 REAP CHAIN 블록체인의 변경이 허가됩니다.

상임위 노드를 운영하는 자는 REAP CHAIN에서 제공되는 투표 소프트웨어 (Voting) 를 통해 선택되며, 기본 프로토콜을 변경하거나 결함이 있는 소프트웨어를 업데이트할 수 있는 권한이 부여됩니다. 또한 Smart Contract의 승인 여부를 결정할 수 있습니다.





소프트웨어 변경에 대한 의사결정은 상임위 노드의 투표 중에서 2/3 이상의 승인 결과를 통해 결정됩니다.

만일 상임위 노드를 운영하는 자가 거버넌스의 의사결정을 거부할 경우, REAP 코인을 보유한 참여자들의 투표에 의해 상임위에서 탈락할 수 있습니다.

상임위 노드를 운영하는 자는 토큰 발행과 소각 승인에 대한 의사결정에 참여할 수 있습니다.

REAP CHAIN 암호알고리즘 기술 개요

Rechain의 암호알고리즘은 단말간, 단말과 서버 간에 단일 장치에서 전체키를 보유하지 않고서도 암호키를 생성하고 서명하고 업데이트 할 수 있는 기능을 제공합니다.

REAP CHAIN 암호알고리즘의 구조는 아래와 같습니다.

- 암호화키는 키를 생성하거나 사용하는 어느 시점에서 완전한 키를 가지지 않고 있음
  통신을 통하여 키를 공유하게 되며 이때에도 전체 키가 노출되지 않음
- 일반적인 메시지 전달은 현재 진행 중이 작업에 대한 정보가 포함되어 있음
- 키서명과 같은 동작에 대한 정의는 여러 메시지를 포함하는 컨텍스트에 의해 이루어짐
- 키에 대한 공유, 메시지, 다수의 메시지를 포함하는 컨텍스트에 대해 동작함

REAP CHAIN 암호알고리즘은 아래의 동작 기능을 제공합니다.

- 양자간 ECDSA secp256k1 생성 및 서명
- 양자간 EdDSA ed25519 생성 및 서명
- 양자간 BIP32 생성
- 양자간 키공유 분배 및 저장
- 양자간 제로 지식 백업 저장 및 복구

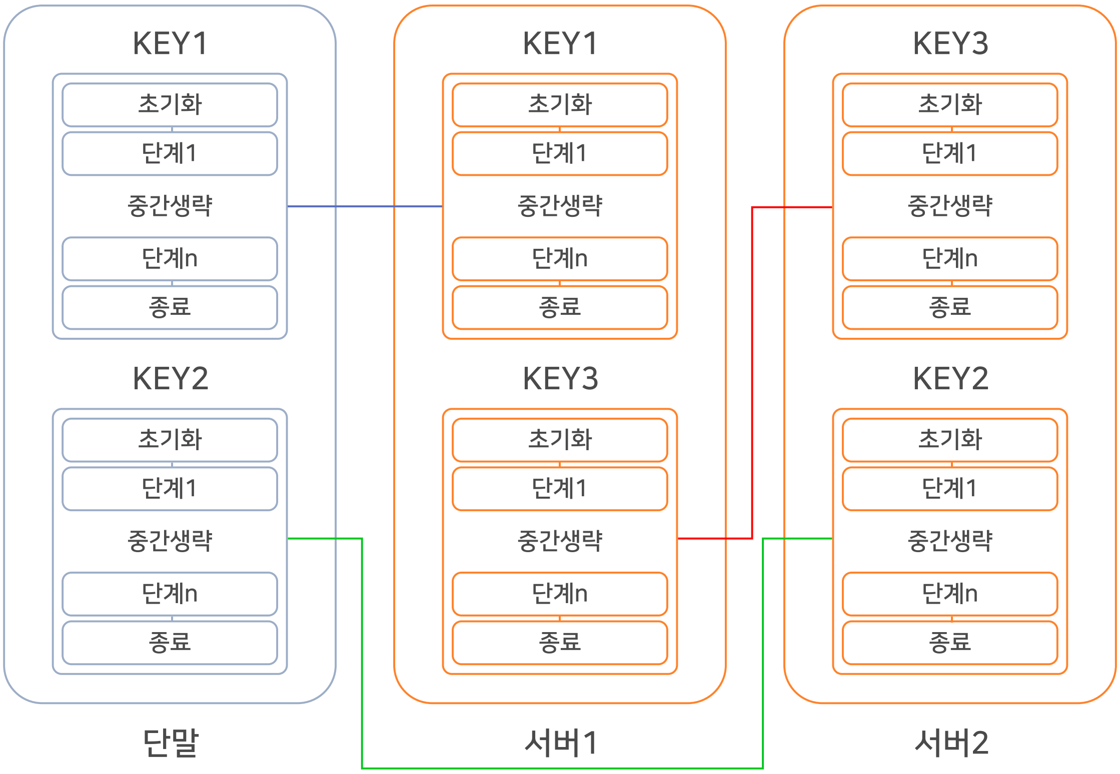
키 공유, 메시지 및 컨텍스트 구조에 대한 직렬화, 역 직렬화 및 메모리 관리를 처리하는 내용을 포함합니다.

시스템에서의 동작 흐름은 다음과 같이 정의합니다.

첫단계는 초기화 단계로 동작을 위한 필요한 정보를 입력하고 컨텍스트를 생성하는 단계로 각 대상은 자체적으로 초기화를 수행합니다. 각 단계에서 컨텍스트 내에는 일련의 단계가 존재하며 각 단계에서 입력 메시지를 입력 받아 일정한 작업을 수행하고 결과를 출력하며, 이 결과를 다시 전송하여 수신된 상태 측에서는 켄텍스트와 연관시키는 작업을 수행합니다.

전체가 마지막 상태를 완료되며 완료상태로 전환하고 메모리를 확보하고 공유된 키 정보를 복사하여 사용하는 단계로 전환합니다.

각 단말별 역할은 요청자가 키를 생성하였고 서명자가 서명을 수행하려면 요청자에게 프로세스 시작을 요청하는 단계를 수행하고 완료되면, 요청자는 결과를 서명자에게 보냅니다. 서명자는 확인을 수행합니다. 자세한 절차는 다음과 같습니다.



- 요청자는 초기화 함수를 호출
- 요청자는 입력파라메터를 입력하고 내부처리를 수행하여 출력 메시지를 생성
- 요청자는 세부정보 및 출력 메시지를 서명자에게 보냄
- 서명자는 세부 정보를 확인하고 실행 동의 수행
- 서명자는 초기화 함수 호출
- 서명자는 요청자로부터 수신한 메시지를 입력 메시지로 하여 단계 함수를 호출
- 서명자는 단계함수로부터 출력된 출력 메시지를 요청자에게 보냄
- 요청자와 서명자 간의 입력 메시지를 통하여 단계함수를 호출하고 상호 출력 메시지를 전달하고 수신하는 과정을 반복

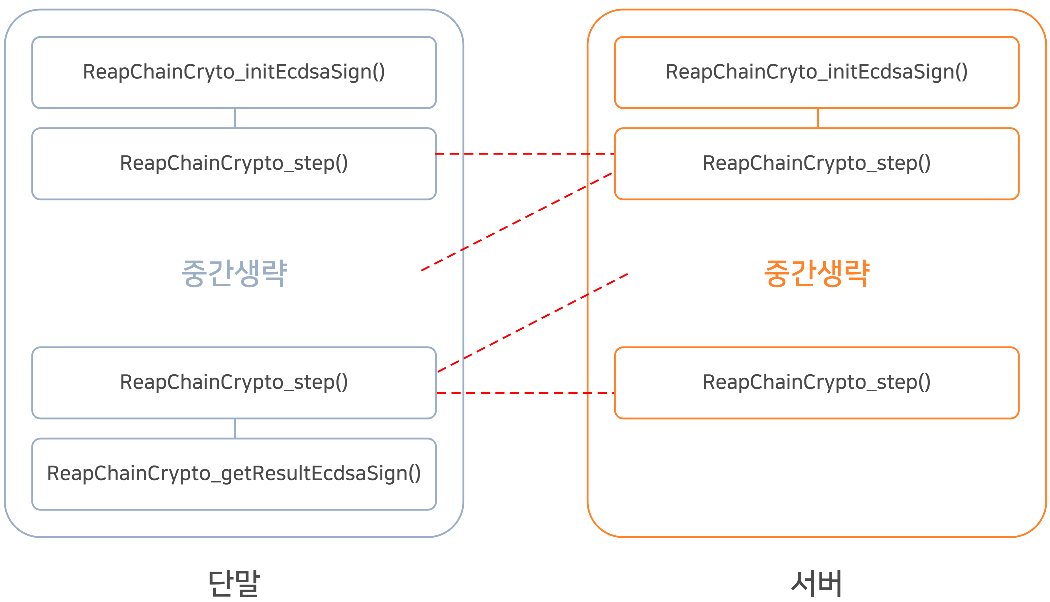
● 요청자와 서명자가 모두 단계함수를 완료한 경우에는 결정수행
- fgREAP CHAIN_MPC_PROTOCOL_FINISHED 플래그는 요청자와 서명자의 마지막 단계 의미
- fgREAP CHAIN_MPC_SHARE_CHANGED 플래그도 포함하면 새로 변경 조치 와 같이 로컬 키 공유가 변경됨. 키 공유는 getShare 함수를 사용하여 컨텍스트에서 검색하고 나중에 사용하기 위해 저장해야 함
- 마지막 단계의 출력 메시지가 비어 있지 않으면 상대에게 메시지 전송하여야 함
- 요청자와 서명자는 작업 유형에 따라 getResult 함수 를 호출해야 할 수도 있음
예를 들어, 서명 동작에는 요청자만 결과가 있지만, 새로 고침 동작에는 서명자 대한 새로운 키 공유

전체 프로세스에서 동일한 컨텍스트를 사용해야 합니다. 컨텍스트를 저장해야 하는 경우 직렬화 기능을 사용한 다음 역 직렬화기능을 사용하여 다시 읽을 수 있습니다.

REAP CHAIN 알고리즘의 동작

REAP CHAIN에서 ECDSA 서명 조치의 예시는 아래와 같습니다.

● 요청자와 서명자는 초기화를 위해 ReapChainCrypto_initEcdsaSign() 함수를 호출하여 시작
● 초기화 후 요청자와 서명자는 서명 프로세스가 완료 될 때까지 ReapChainCrypto _step() 함수를 여러 번 호출 수행
● 서명 프로세스의 결과인 서명은 서명 프로세스가 완료된 후 최종 함수 ReapChainCrypto_finalEcdsaSign() 을 호출하여 수신



REAP CHAIN 암호 솔루션 개요

REAP CHAIN 암호알고리즘은 서명을 생성하고 자금을 이체하는데 사용되는 개인 키를 보유한 단일 장치를 기반으로 하지 않습니다.

보관하려는 개인키는 단말 및 서버 간에 공유되기 때문에 실제 키에 대한 정보를 가지고 있지 않게 되며 서명을 생성하는 하는 경우에도 서로 키를 공유하지 않고 서명을 생성하는 안전한 양자간 계산 프로토콜을 사용하게 됩니다. 각 단말에서 사용되는 계정은 같은 사람에 국한되지는 않고 다양한 형태로 구성될 수 있습니다.

REAP CHAIN 암호 솔루션은 BIP32 / BIP44 키 생성을 지원합니다. BIP키에 의해 키를 생성하는 경우에는 BIP 마스터 시드 또는 키가 사용됩니다. BIP키에 의해 생성되는 키는 단말 및 서버 간에 REAP CHAIN 암호 알고리즘에 의해 수행되고 서명을 통하여 통합되고 해당 키에 대한 일부만을 수신하고 전체키가 공개되지는 않습니다.

BIP를 적용하는 이유는 일부의 키가 손상되었다고 하더라도 키를 복구할 수 있는 데 중요하게 사용됩니다. 개인키가 손상되면 백업을 수행해야 하는 RSA 키 페어가 생성되고 개인 RSA키는 콜드 백업에 저장되며 ECDSA / EdDSA 개인키는 RSA 코드 백업 공개키로 암호화 수행합니다. 이 암호화는 각 당사자가 자신의 공유에 대해 개별적으로 수행하며, 각 당사자는 개인 키의 올바른 공유가 암호화되었다는 공개적으로 검증 가능한 제로 지식 증명을 생성합니다.

제로 지식 증명은 악의적인 당사자가 콜드 백업을 쓸모 없게 만드는 것을 방지하기 위해 필요합니다. 이 방법은 BIP 또는 다른 키 생성 방법을 사용하여 생성된 키와 동일하게 작동합니다. 백업은 실제 키이므로 BIP 생성을 사용할 때는 마스터 시드가 아닌 루트 개인 키로 전달되며 필요한 경우 생성된 모든 BIP 키를 복구가 가능합니다. 루트 노드 개인 키에 대해보다 효율적인 제로 지식 증명이 존재하기 때문에 BIP 마스터 시드 자체를 직접 백업하는 방법을 제공하지 않습니다.

REAP CHAIN 암호 솔루션은 두 당사자간에 실행되는 MPC 프로토콜로 구성됩니다. 그러나 동일한 프로토콜을 사용하여 2-out-of- n 당사자 (n 개의 엔터티가 있고 다른 모든 작업에 서명하고 수행하기에 충분할 경우)를 설정하는 MPC를 달성할 수 있으며, 확장은 두 당사자가 처음에 키를 생성하고 n 사이의 모든 쌍에 공유를 다시 공유함으로써 수행합니다.

일반적이며 secp256k1 곡선에서 ECDSA를 지원하고 ed25519 곡선(또는 Schnorr)에서 EdDSA를 지원합니다. EdDSA의 경우 비밀 키를 추가로 공유하는 Schnorr의 임계 값 서명 프로토콜을 사용합니다.

키 생성에 사용되는 제로 지식 증명이 보다 효율적이고, BIP 키를 통하여 생성된 키는 MPC를 ECDSA 키 생성과 통합하며 키 공유 업데이트가 포함됩니다.

지원되는 기능

REAP CHAIN 암호 알고리즘은 아래의 내용을 지원합니다.

- **일반 비밀 생성**: BIP를 통하여 개인키 생성에 사용되는 초기 비밀을 생성하는 데 사용
- **BIP 생성**: BIP32 표준에 따라 초기 비밀을 사용하여 MPC에서 키를 생성. 이 단계의 결과는 ECDSA에서 사용할 수 있는 키이며, 두 당사자는 생성된 공개 키를 받고 관련 개인 키의 임의 공유를 보유함 MPC가 사용되므로 어느 당사자도 전체 개인 키를 알고 있지 않음
- **ECDSA / EdDSA 키 생성**: ECDSA 또는 EdDSA 키를 직접 생성하기 위해 두 당사자에 대해 MPC를 사용. 그 결과 양 당사자에게 알려진 공개키와 준 개인 키를 임의로 공유. MPC가 사용되므로 어느 당사자도 전체 개인 키를 알고 있지 않음
- **백업 메카니즘**: 백업 방법은 RSA 백업 공개 키를 수신하고 올바른 것으로 보장 된 당사자의 공유에 대해 공개적으로 검증 가능한 백업을 생성. 이 절차는 제로 지식 증명을 사용하여 정확성을 검증할 수는 있지만 개인 키 공유에 대해 아무것도 공개하지 않도록 함. 백업 검증 기능은 제로 지식 증명을 검증하고 모든 엔티티가 백업이 유효한지 확인하기 위해 사용(이 기능은 공개 ECDSA / EdDSA 키를 수신하여 백업이 제공된 공개 키의 개인 키인지 검증 함). 마지막으로 백업 복원 방법은 백업 정보와 RSA 백업 개인 키를 가져와 ECDSA / EdDSA 개인 키를 출력
- **ECDSA / EdDSA 서명**: 이 방법은 이전에 생성 된 키를 사용하여 둘 다 승인 된 메시지 m에서 MPC에 로그인 수행. MPC가 사용되기 때문에 어느 당사자도 그러한 서명만 생성하거나 상대방이 승인한 메시지와 다른 메시지에 서명하도록 속일 수는 없음
- **키 공유 업데이트**: 당사자가 기존 공유 키의 새로운 랜덤 공유를 공동으로 생성하여 이전 공유가 쓸모 없게 되는 데 사용. 공격자가 새로 고치기 전에 한 장치에서, 새로 고침 후에 다른 장치에서 공유를 훔치는 경우 개인 키에 대해서는 아무것도 알지 못함

사용하는 암호화 기법

① ECDSA 서명

ECDSA 서명 알고리즘 정의는 다음과 같습니다.

앨리스는 키쌍(d_A, Q_A)을 만듭니다.

- d는 무작위로 선택된 1부터 n-1사이의 정수로서, 개인키
- Q는 Q=dg를 만족하는 정수로서, 공개키

- 앨리스가 메시지 m을 다음 절차를 따라 서명합니다.
  1) e=H(m)이고, H는 암호학적 해쉬함수
  2) z는 e의 이진 값에서 왼쪽으로부터 Ln번째 까지를 잘라낸 값 Ln leftmost bits of e),
     Ln는 n의 비트 길이(bit length)
  3) 암호학 적으로 안전한 난수 k를 [1, n-1] 사이에서 무작위로 선택
  4) 곡선 위의 점 (x1, y1) = k * g을 계산
  5) r = x1 (mod n)을 계산. 만약 r=0이면 3으로 되돌아가 다른 k를 다시 선택하여 순서대로 진행
  6) s = k-1(z + rdA) (mod n)을 계산. 만약 s=0이면 3으로 되돌아가 다른 k를
     다시 선택하여 순서대로 진행
- 완성된 서명은 (r, s)

② EdDSA 서명

EdDSA 서명 알고리즘은 Edwards 곡선에 대한 Schnorr 서명 버전입니다.

간단히하기 위해 여기서 Schnorr의 서명을 설명합니다.

G는 기준점 (생성기)이 G 인 순서 q의 타원 곡선 그룹으로 합니다.

개인 키는 임의의 값 x ∈ Zq이고 공개 키는 Q = x G입니다. 메시지 서명 m은 다음과 같습니다.

1) 무작위 **r ∈ Z q ***를 선택 (표준 EdDSA에서는 의사 난수 기능을 사용하는 개인 키 및 메시지에서 생성되지만 임의로 선택)
2) **E** ← **H (R, Q, m**)
3) s ← r + x · e mod q
4) Output (R, s)

③ Paillier 암호화

Paillier 암호화 방식은 많은 수의 인수 분해의 어려움에 의존하는 RSA와 같은 공개 키 암호 시스템입니다.

1) Paillier의 공개 및 개인 키는 두 개의 무작위 소수 **p** 및 **q**를 선택하여 생성.
   개인 키 **(p, q)** 의 경우 공개 키는 **N = pq**
2) 메시지를 암호화하는 **m** , 랜덤 샘플 값 **(R)**을 하고 계산 **ENCPK (m, R) = (1 + N) X R N mod N 2**

Paillier 암호화의 중요한 특성은 부가 적으로 동질적이라는 것입니다. 평문 m1, m2의 암호화 인 암호문 c1, c2가 주어지면 (c1) v mod N2는 m·v mod N의 암호화이고 c1·c2 mod N2는 m1 + m2 값의 암호화하며, 암호화 된 값은 복호화없이 알려진 스칼라에 의해 합산되고 곱해질 수 있습니다.

④ 확정 체계(Commitment Schemes)

확약 방식은 나중에 암호화된 값을 공개할 수 있는 능력으로 다른 사람에게 숨기고 선택된 값(또는 선택된 문장)에 커밋할 수 있는 암호화 기본 요소입니다. 약정 체계는 당사자가 약정을 한 후에 가치나 진술을 변경할 수 없도록 설계되었습니다. 즉, 약정 체계는 구속력이 있습니다. 확약 체계는 안전한 동전 뒤집기, 제로 지식 증명 및 안전한 계산을 포함하여 여러 암호화 프로토콜에서 적용되고 있습니다.

약정 체계를 시각화하는 방법은 보낸 사람이 메시지를 잠긴 상자에 넣고 상자를 받는 사람에게 주는 것으로 생각하는 것입니다. 상자의 메시지는 수신기에서 숨겨져 있으며, 잠금을 직접 열 수 없습니다. 수신자가 상자를 가지고 있기 때문에, 발신자가 나중에 키를 제공하기로 선택한 경우 내부 메시지를 변경할 수 없습니다.

약정 체계의 상호 작용은 다음 두 단계로 이루어집니다.

- 값이 선택되어 지정되는 커밋 단계
- 값이 공개되고 확인되는 공개 단계

간단한 프로토콜에서 커밋 단계는 발신자에서 수신자로의 단일 메시지로 구성됩니다.
이 메시지를 약속이라고 합니다.

- 숨김속성
  그 때 수신자가 선택한 특정 값을 알 수 없어야 합니다. 간단한 공개 단계는 단일 메시지, 발신자에서 수신자로의 열기, 수신자가 수행한 확인으로 구성됩니다.

- 바인딩 속성
  커밋 단계에서 선택한 값은 보낸 사람이 계산할 수 있고 공개 단계 중에 유효성을 검사할 수 있는 유일한 값이어야 합니다.

확약 체계는 암호화 해시 기능을 사용하여 구현될 수 있습니다.

이 경우 발신자는 문자열 **r** 을 선택하여 메시지 **m** 에 커밋할 수 있습니다. (길이는 128 비트) 수신기에 **H (m || r)**를 보냅니다. 그런 다음 확약은 송신자 **(m, r)**만 송신하고 수신자는 두 값의 해시가 실제로 송신된 값인지 확인합니다. 이것은 해시 함수의 충돌 방지 속성에 의해 바인딩되며, 해시 함수의 속성에 대한 가정하에 숨어 있습니다. 특히, 해시 함수가 랜덤 함수로 모델링되는 랜덤 오라클 모델에서 값 H (m || r)는 m을 유추하여 파악할 수는 없습니다.

⑤ 제로 지식 증명

제로 지식 증명는 증명자와 검증자 사이의 프로토콜이며, 입증자는 정보를 공개하지 않고 진술이 사실임을 검증자에게 증명합니다. 제로 지식 증명은 세 가지 특성을 충족해야 합니다.

- (건전성) 진술이 거짓이면 검증자는 검증자에게 진술이 참임을 입증할 수 없으며,
- (완전성) 진술이 참이면 검증자가 검증자가 진술이 사실이며,
- (제로 지식) 증명자는 사실이 사실이라는 사실을 넘어 증거로부터 정보를 배울 수 없어야 합니다.

제로 지식 증명은 증명이 주어진 진술에 대한 실제 NP 증인을 알고 있는 경우에만 진술을 증명할 수 있게 하는 제로 지식 증명입니다. REAP CHAIN 프로토콜에서 사용하는 제로 지식 증명을 사용합니다.

1) 커밋된 값과 암호화된 값이 동일하다는 증거
2) 커밋된 값이 일부 값 범위에 있음을 증명
3) 플레이어가 타원 곡선 포인트 (ZK-DL)의 이산 로그를 알고 있다는 증거
4) Paillier 공개 키가 올바르게 생성되었음을 증명합니다.(ZK-PPK)
5) 주어진 Paillier 암호문의 암호 해독이 주어진 타원 곡선 점 (ZK-PDL)의 불연속 로그임을 증명
6) RSA 암호문 ci가 Qi가 알려진 xi = xi·G가되도록 값 xi를 암호화한다는 증거. 콜드 백업에 사용

제로 지식 증명이 비대화 형인 경우에는 공개적으로 검증할 수 있는 속성을 갖습니다. 이는 누구든지 증명을 읽고 진술이 실제로 정확한지 확인할 수 있습니다. 자금을 주소로 이체하기 전에 백업이 유효한지 확인하는 데 사용할 수 있습니다.

⑥ 중간에 노출되지 않는 망각 전송(Oblivious transfer)

왜곡 회로(Grabled Circuits) 프로토콜에서 명백한 전송을 사용합니다. 명백한 전송에서는 다음과 같은 방식으로 발신자와 수신자 간에 문자열이 전송됩니다.

- 발신자는 두 개의 문자열 S0와 S1을 갖습니다.
- 수신자가 i ∈{ 0, 1 } 을 선택하고 발신자는 망각 전송 프로토콜과 함께 Si를 전송합니다.
- 수신자는 S(1-i)에 대한 정보를 얻지 못합니다.
- i의 값은 발신자에게 노출되지 않습니다. (영지식)

망각 전송은 RSA 암호화 시스템과 같은 비대칭 암호화를 사용하여 구현됩니다.

OT(Oblivious Transfer)는 공개키를 사용해야 하며, OT 확장을 사용하여 공개키를 기반으로 고정된 수의 OT를 실행한 다음 해싱만 사용하여 더 많은 실제 OT를 얻을 수 있습니다.

전체적으로 이 프로토콜은 설정에 3 라운드의 상호 작용이 필요하고 OT 실행을 수행하기 위해 2 라운드가 필요합니다. (실제 OT의 첫 번째 라운드는 설정의 세 번째 라운드와 함께 전송될 수 있으므로 전체적으로 4 라운드가 필요합니다.)

⑦ 왜곡 회로(Grabled Circuits)

왜곡 회로(Grabled Circuits)는 두 명의 불신하는 당사자가 개인 입력에 대한 기능을 공동으로 평가할 수 있고 신뢰할 수 있는 양자 보안 계산을 가능하게 하는 암호화 프로토콜을 의미합니다. 왜곡 회로 프로토콜의 기능은 부울 회로로 설명해야 합니다. 안전한 양당 계산 프로토콜에서 사용되는 암호화 알고리즘입니다.

평가자에게 입력의 암호화를 사용하여 잘못된 회로에 대한 액세스 권한이 부여되면 다른 정보를 제공하지 않고 주어진 입력에서 함수의 출력을 알 수 있습니다. 당사자 중 하나가 나중에 다른 당사자에 의해 평가되는 왜곡 회로를 생성하도록 하여 안전한 양자간 계산에 사용됩니다.

⑧ 이중 실행

두 명의 플레이어가 정직한 경우(지정된 프로토콜을 실행할 수 있음을 의미) 단일 깨진 왜곡회로 만으로 도 안전한 계산을 실행할 수 있습니다. 그러나 악의적인 경우 출력을 통해 정보를 유출할 수 있는 잘못된 왜곡 회로를 생성할 수 있습니다.

당사자가 악의적일때 보안을 달성하는 것은 중요한 과제이며 비용이 발생합니다. 당사자들이 부정 행위를 방지하는데 사용되는 방법 중 하나를 "이중 실행"이라고 합니다.

이것은 각 당사자가 왜곡회로가 한 번 계산을 수행하고, 회로의 평가자로 한 번 계산을 실행하도록 하여 작동합니다.

두 평가 후 당사자는 결과를 공개하지 않고 결과를 비교합니다. 두 회로가 동일한 출력을 생성하는 경우 각각은 그것이 옳다는 것을 알고 있습니다. 각 당사자가 회로 중 하나를 생성하기 때문에 정직한 경우 회로와 계산이 정확하다는 것을 알고 있습니다.

이중 실행에서 공격자에게 유출된 유일한 정보는 중단이 있기 때문입니다. 즉, 결과의 비교가 "같지 않다"면 단일 비트가 누출될 수 있습니다. 예를 들어, 손상된 당사자는 키의 첫 번째 비트가 0 인 경우 garbage를 계산하는 잘못된 회로를 생성하고 그렇지 않으면 올바른 출력을 계산할 수 있습니다.

회로가 깨져서 감지되지 않습니다. 그런 다음 중단이 있었는지 여부에 대한 단순한 비트를 누출시킵니다.

그러나 여기에서 키 생성에 대한 사용법에서 결과가 같지 않을 때 공격이 감지되었기 때문에 그러한 공격으로 인한 누출이 적다는 것을 알 수 있습니다. (키의 평균 2 비트 공개) 따라서 공격이 탐지되면 공격자를 제거하고 자금을 새로운 주소로 이전하는 것이 현명합니다.

동등성 테스트(ElGamal in-the-exponent 부가 적으로 동질적인 암호화 사용)를 기반으로 3 라운드 통신으로 새로운 동등성 테스트를 구성했습니다. 이 테스트는 보안 계산의 이상적인 / 실제 모델 시뮬레이션 패러다임에서 보안 증명을 가능하게 하고 양 당사자에게 확인된 출력을 제공하도록 고안되었습니다.

확장을 기반으로 필요한 OT의 두 라운드를 사용하여 전체 이중 실행에는 동등성 테스트를 포함하여 5 회의 통신이 있습니다.

이중 실행 프로토콜은 당사자가 사기를 당하면 단일 비트를 알 수 있지만 확률 ½ (비공식적으로 말하기)을 잡는 비용으로 속성을 가지고 있습니다. 따라서 공격이 탐지되면 키 사용을 중지됩니다.

전략 중 하나는 부정 행위를 시도하지만, 정직한 당사자에게 평등 테스트의 결과를 제공하지 않고(이 결과를 먼저 수신할 때) "충돌" 또는 네트워크 오류가 있다고 주장하는 것입니다. 이것이 처리되지 않으면, 상대방은 비밀 키의 모든 비트를 알 수 있습니다. 따라서 실행은 항상 완료해야 하며 (완료하는 데 필요한 메시지를 디스크에 저장하고 충돌 또는 실패 후 계속) 또는 일부 부정 행위를 가정해야 합니다.

또한 공격자가 각 병렬 실행에서 조금 알 수 있기 때문에 동일한 입력 키에서 많은 실행을 병렬로 실행할 수 없습니다. 이것이 필요한 경우 모든 실행에 대해 단일 동등성 테스트가 실행되도록 코드를 변경할 수 있으며 이것이 안전합니다.

Block Crypto MPC

REAP CHAIN의 암호 알고리즘은 Secure MPC(Multiparty Computing)을 적용하고 있으며, 키관리/보호를 위해서 Yehuda Lindell 교수와 Samuel Ranellucci 박사가 고안한 기술을 채용하며 암호화 서명 키 및 시드(Seed) 비밀 보호를 위해서 사용하고 있습니다.

REAP CHAIN이 사용하고 있는 Secure MPC 기술은 암호화 키와 비밀 보호를 위해서 다수의 사용자가 각자의 키를 생성하고 이를 조합한 마스터키를 사용하는 방식인 Multi-sig 형태와는 다르게 단일 서명으로 원장에 대한 의존성이 없이 사용할 수 있는 장점을 가집니다.

아래는 3가지 장점에 대해서 기술합니다.

- **분리의 성질** : 비밀을 유지하여야 하는 개인키는 두개의 공간에 분리되어 저장됩니다.
  이는 별도의 물리공간을 의미하며, 보유하고 있는 정보에 대해서는 서로 모르게 됩니다.
- **기밀성의 성질** : 중요한 부분으로 분리된 공간에 저장한 개인키는 조합/결합되지 않고 모든 암호화 작업을 수행하게 되며, 두 개의 장치를 동시에 공격하지 않은 이상에는 이를 손상시킬 수 없다고 알려져 있습니다. 또한 키 자체가 물리적으로 분리되어 있기 때문에 하나가 손상된다고 해도 필요한 정보를 얻을 수는 없게 됩니다.
- **예측 불가의 성질** : 키에 대한 공유 방식은 계속 값을 변경하며 전송하게 되고 동시에 물리적으로 분리된 두개의 시스템을 손상시키지 않으면 알아낼 수가 없는 형태입니다.

REAP CHAIN에서 사용하는 주요 알고리즘은 아래의 내용을 포함합니다.

- 양자간 ECDSA secp256k1 생성 및 서명
- 양자간 EdDSA ed25519 생성 및 서명
- 양자간 BIP32 생성
- 양자간 키 공유 분배 및 저장
- 양자간 제로 지식 백업 제공

주요 알고리즘은 강력한 보안을 제공하기 위해서 두 개 이상의 별도의 시스템에 배포되어 사용되는 암호화 방식입니다.

REAP CHAIN의 보안 알고리즘은 블록체인의 앱에서 보안을 제공하기 위해서 사용할 수 있습니다.

① 개별 단말기와 서버간의 사용 방법

REAP CHAIN 이용자는 개별 단말의 모바일 지갑을 이용하여 서버와 통신을 수행하고 이를 통해서 암호화 알고리즘의 서비스를 받을 수 있습니다.

BIP32 시드 값과 모든 서명값은 하나의 단말에 존재하지 않고 두 개 이상의 분리된 디바이스에 존재합니다. 따라서 암호화 작업을 수행하기 위해서는 통신을 통하여 암호화 작업을 수행하여야 합니다.

1) 일반적인 표준들
   - Mnemonic code words, based on BIP-39
   - HD wallets, based on BIP-32
   - Multipurpose HD wallet structure, based on BIP-43
   - Multicurrency and multiaccount wallets, based on BIP-44

2) 표준은 변경되거나 폐지될 수는 있지만 REAP CHAIN과 암호화폐를 위한 de facto 지갑 표준으로 상용됩니다.

3) 각 표준은 광범위한 소프트웨어, 하드웨어 지갑에 사용되는 서로 연동할 수 있도록 만들어 줍니다. 사용자는 지갑 중 하나에서 니모닉을 생성할 수 있고, 다른 지갑에 올릴 수 있고, 모든 키와 주소를 복구할 수 있습니다.

개별 단말기와 서버간의 사용 방법의 특징 및 장단점은 다음과 같습니다.

● 트랜잭션이 발생한 경우에 이를 서명하기 위해서는 단말과 서버가 같이 서명을 수행하지 않으면 서비스가 제공되지 않는 부인방지 효과
● 키를 탈취하거나 손상시키기 위해서는 동시에 단말과 서버를 공격하여 동시에 손상을 시켜야 하기 때문에 이중안전 장치 제공 효과
● 키를 생성하여 사용하는 중에 시드 값 및 생성 키에 대한 정보가 외부로 노출되지 않아 보안성의 3요소 중 하나인 기밀성 보장
● 직접적인 암호 공격 외에 시스템에 가해지는 부채널공격에 대해서 강력한 안전성을 제공
● 단말 및 서버에 모두 안정된 서비스를 제공하는 암호화 방식으로서 암호화 자산 보호 효과

② 단말/단말간의 사용 방법

사용자가 여러 개의 단말을 가지고 있는 경우에 이를 통하여 블록체인 지갑 및 서명 거래에 안전성을 확보하기 위해서 적용이 가능합니다.

단말과 단말간에서 BIP32 시드 및 모든 서명키에 대해 통신을 통하여 암호화 작업을 수행할 수 있습니다.

단말/단말간의 사용 방법의 특징 및 장단점은 다음과 같습니다.

- 단일 장치에서는 거래를 승인할 수 없으며 서로 다른 두 개의 장치에서 거래를 승인해야만 거래가 이루어지게 됨
- 거래에 대한 주요 내용을 손상시키기 위해서는 두 개의 다른 장치를 동시에 공격하여야만 손상시킬 수 있음
- 암호작업을 수행하는 중에 시드키나 개인키가 노출되지 않아 기밀성을 보장
- 키에 대한 직접적인 공격이 아닌 부채널공격에도 강함
- 단말과 단말 간의 안전한 거래를 위해서 제공되는 안전한 서비스

암호알고리즘 백업

백업을 수행하여 안정성을 확보하는 방안이 중요하며 아래에서는 이에 대해 설명하겠습니다.

기본적인 특성으로 공개적으로 검증이 허용되어 있는 암호화된 콜드 백업 기능을 제공하고 있으며, 서로 다른 참여자가 백업을 별도로 해독하거나 하지 않고도 백업의 정확성을 확인할 수 있도록 합니다. 검증을 통한 백업의 잘못된 생성 및 저장에 대한 원천적인 방어책을 제공합니다.

① 백업 사용시 사용자 정보 관리

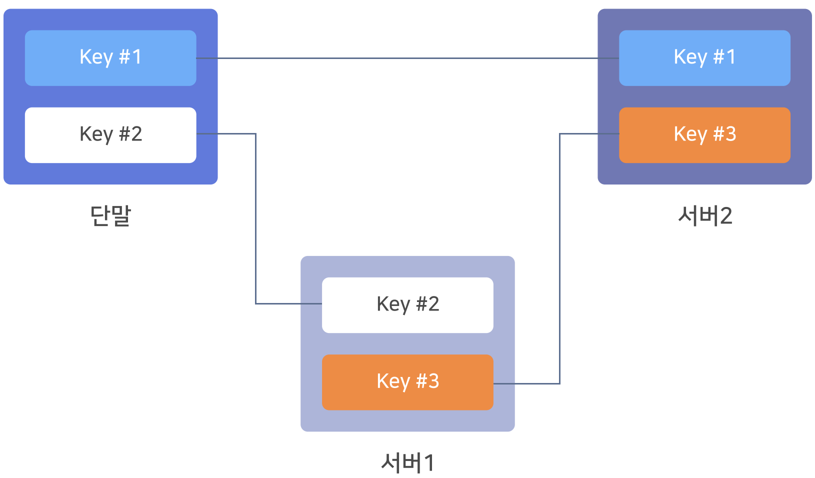
일반적으로 최종 사용자에 대한 백업 관리 역할을 하는 일반적인 백업 형식입니다.

지갑의 암호화 된 백업은 중복성을 위해 여러 위치에 저장될 수 있습니다.(예 : 모바일 / PC / 서버 사용 사례에 설명된 대로 서비스 제공 업체가 저장할 수 있음)이 백업의 개인 키는 사용자가 단독으로 소지해야 하며 콜드 백업이어야 합니다. 백업 복구 프로세스는 재해 복구에만 사용해야 합니다.

지갑의 암호화된 백업은 중복성을 위해서 서로 다른 사용자가 보유하거나 서버에 보관할 수 있습니다. 이 백업에 대한 개인키는 사용자가 단독으로 소유하고 있어야 하고 콜드 백업형태로 보관되어야 합니다. 백업 복구 프로세스에 대해서는 키가 손상된 경우에만 접근할 수 있도록 하여야 합니다.

② 별도의 관리 서버를 이용한 백업

별도의 백업서버를 활용하여 사용자 단말 간 혹은 서버에서 사용하여 각자의 공유키를 잃은 경우에 사용할 수 있도록 합니다.



이 경우에는 중복하여 복원 정보를 가지게 되며, 단말과 서버간, 단말과 백업서버 간, 백업서버와 서버간 복원을 위한 정보를 보유하게 됩니다.

참고문헌

[1] kblock, “[케블리] #48. 합의 알고리즘 이해하기 - PBFT Consensus Algorithm,” 2018. [온라인]. Available: https://steemit.com/consensus/@kblock/48-pbft-consensus-algorithm.

[2] vasa, “ConsensusPedia: An Encyclopedia of 30+ Consensus Algorithms,” 2 7 2018. [온라인]. Available: https://hackernoon.com/consensuspedia-an-encyclopedia-of-29-consensus-algorithms-e9c4b4b7d08f.

[3] MabelOza12, “Hyperledger Consensus Algorithms,” 7 4 2018. [온라인]. Available: https://www.slideshare.net/MabelOza12/hyperledger-consensus-algorithms.

[4] G. Konstantopoulos, “Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance,” 1 12 2017. [온라인]. Available: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419.

[5] N. S., “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [온라인]. Available: https://bitcoin.org/bitcoin.pdf. [액세스: 2019].

[6] M. &. G.-E. J. C. Herrero-Collantes, “Quantum Random Number Generators. Reviews of Modern Physics. 89. 10.1103/RevModPhys.89.015004.,” 2016.

[7] A. D. Adamk , “Internet Tools and Services,” 2014. [온라인]. Available: https://gyires.inf.unideb.hu/GyBITT/08/ch04.html.

[8] 김선미, 임종철, 유현경 그리고 곽지영, 블록체인과 합의 알고리즘, ETRI, 2018.

[9] A. Asayag, G. Cohen, I. Grayevsky, M. Leshkowitz, O. Rottenstreich, R. Tamari 그리고 D. Yakira, “Helix Consensus Algorithm,” 4 2018. [온라인]. Available: https://www.orbs.com/white-papers/helix-consensus-whitepaper/. [액세스: 2019].

[10] NEM Foundation, “NEM Technical Reference,” 2018.

[11] FUTUREPIA, “Whitepaper V1.2,” 2019.

[12] SYMVERSE INC., “Whitepaper,” 2018.

[13] Juan Carlos Garcia, “Quantum Random Number Generators,” 2016. [온라인].

[14] 최낙준, 이지은 그리고 김광조, “진난수생성기에 대한 부채널 공격 및 난수성 분석 사례 조사,” 2018.

[15] 과학기술정보통신부 그리고 한국과학기술기획평가원, “블록체인의 미래,” 2019.

[16] PegaSys, “Scaling Consensus for Enterprise: Explaining the IBFT Algorithm,” 6 2018. [온라인]. Available: https://media.consensys.net/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm-ba86182ea668.