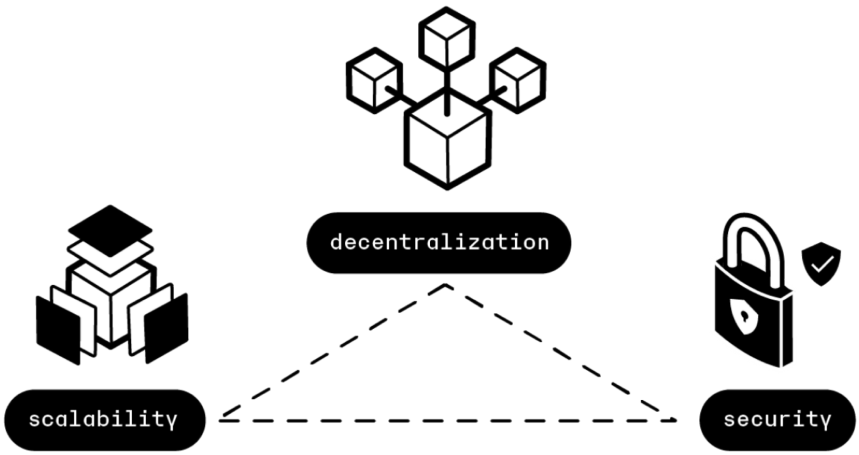
zkEVM

1조 유인선, 박보현, 노종찬

Abstract

최근 여러 레이어2 프로젝트에서 빠른 속도로 발전하고 있는 zkEVM이 무엇인지 영지식 증명의 이론적인 기반과 함께 알아보고 현재 zkEVM을 사용하고있는 ZK 롤업들의 특징들을 비교한다. 이 아티클은 이더리움과 레이어2, 영지식증명에 대한 기초적인 지식을 가진 독자들을 대상으로 작성되었다.

1. Motivation



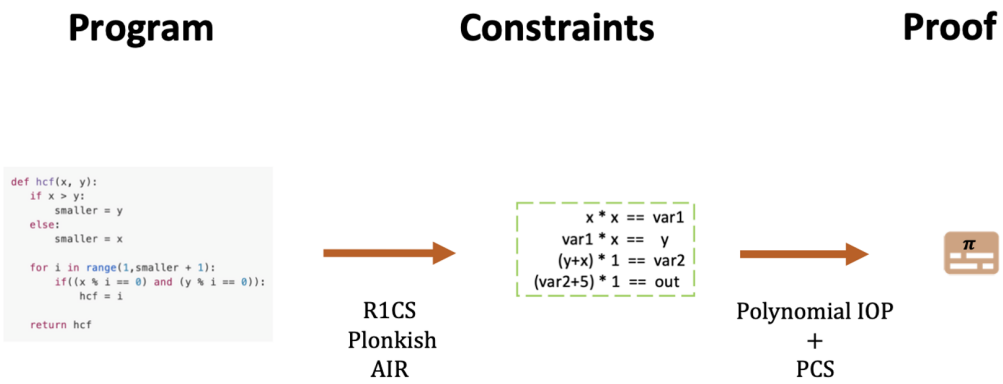
Source: https://coinloan.io/blog/what-is-blockchain-trilemma/

여전히 블록체인 트릴레마는 L1 블록체인의 풀리지않는 숙제이다. L1 의 확장성을 위한 솔루션으로 여러가지 방안들이 제시되었는데, 이더리움의 경우 스케일링 솔루션으로는 L2, 그 중에서도 ZK 롤업을 Execution layer로써 사용하고자 하는 움직임이 활발해지고있다.

ZK 롤업과 옵티미스틱 롤업의 유일한 차이는 Validity proof 제출의 유무이다. Optimistic Rollup의 경우 “Presumption of innocent”를 전제로 L2에서 어떤 증명값도 제출하지 않고, 수동적으로 Challenge를 받는 방식이다. 반면 ZK 롤업의 경우 L2 데이터를 L1 블록에 올릴때마다 해당 데이터에 대한 Validity proof를 같이 제출한다.

ZK 롤업에도 여러가지 형태가 있는데, 이 아티클에서는 Universal ZK 롤업을 중점적으로 살펴볼 예정이다. Universal 롤업은 어떤 특정 기능만 가능한 App-specific 롤업(가령 송금의 기능만 있는 롤업)과 달리 zkEVM 또는 zkVM을 사용해 Proof를 생성하는 General purpose 롤업을 말한다.

2. ZK-SNARK



ZK-SNARK workflow | source: https://zk-learning.org/

a. ZK-SNARK Structure

zkEVM에서 생성되는 Validity Proof는 ZK-SNARK라는 증명방식을 사용한다. ZK-SNARK는 Zero Knowledge Succinct Non-interactive Argument of Knowledge의 약자로 ZK의 속성인 Hiding과, SNARK의 속성인 Non-interactive를 모두 만족하면서 간결한 검증을 할 수 있는 증명방식이다.

ZK-SNARK는 흔히 엔지니어링에서 사용하는 Front-end와 Back-end라는 용어를 사용하는데 널리 알려진 개념과는 다른 형태이다. ZK-SNARK의 work flow는 다음과 같다.

1) Setup : 합의된 형태를 통해 Prover와 Verifier가 증명과 검증을 위해 각각 가지고있는 파라미터이다.

2) Prover

1. 증명하고자 하는 Function을 Arithmetic circuit의 형태로 바꾼다.
2. Arithmetic circuit을 Oracle(주로 다항식)의 형태로 바꾼다.
3. Oracle을 암호학적인 컴파일을 통해 특정 형태의 Commitment 값으로 Verifier에게 제출한다.

3) Verifier : Prover로부터 받은 증명값을 검증로직으로 연산 후 Accept or Reject한다.

b. ZK-SNARK Traits

ZK-SNARK에는 여러가지 스킴들이 존재하는데 각 스킴들마다의 장단점이 있기 때문에, zkEVM의 형태와 trade-off를 잘 고려하여 proving scheme을 선택해야한다. 고려해야할 특성은 크게 3가지가 있다.

1) Setup

Trusted setup per circuit : Circuit 별로 매번 다른 Trusted setup이 필요하다. (e.g. Groth16)

Universal Trusted Setup : 한번의 Trusted setup으로 모든 Circuit에 사용 가능하다. (e.g. KZG)

Transparent Setup : Trusted party가 필요하지 않은 Transparent setup이다. (e.g. FRI)

2) Proving cost : Prover가 proof를 생성하기 위한 complexity이다.

3) Verifying cost : ZK 롤업의 Verify는 보통 온체인에서 이루어지기 때문에 가스비와 직결되는 특성이다.

Proof size : Circuit의 크기 대비 증명값의 크기이다. (e.g. KZG - O(1), IPA - O(log N))
Verifier time : Verifier proof 검증 로직의 time complexity다. (e.g. KZG - O(1), IPA - O(N))

c. Different types of proving schemes

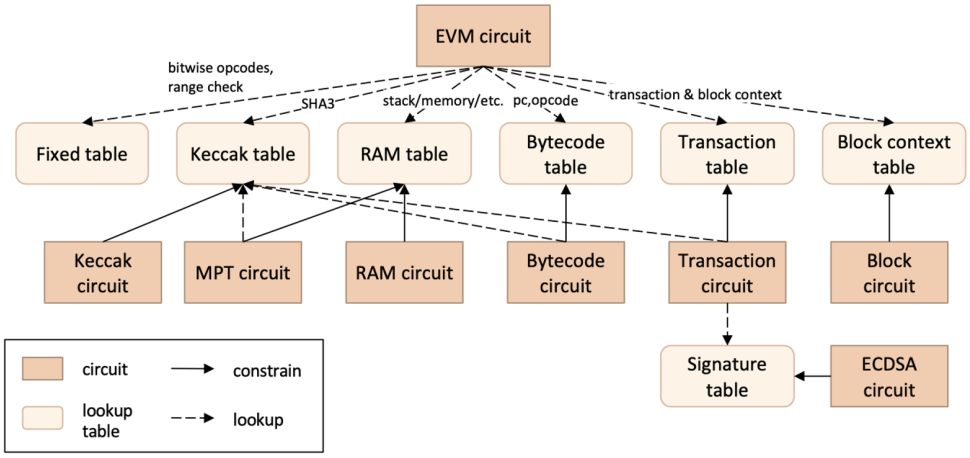
Proving scheme들은 사용하는 암호학적 가정들에 따라 나눌 수 있다. 각각의 기반 테크닉에 따라 Setup의 형태나 코스트가 달라진다.

1) Pairing based : 타원곡선의 Bilinear pairing을 이용한 스킴 (e.g. KZG)

2) Discrete log based : 타원곡선의 이산로그문제(DLP)를 이용한 스킴 (e.g. Bulletproofs)

3) Hash based : 해시를 사용하는 스킴 (e.g. FRI)

3. Architecture of zkEVM



Architecture of zkEVM | source: https://zk-learning.org/

a. zkEVM proving data

zkEVM은 모든 트랜젝션들을 올바르게 실행시켰는지, 즉 EVM의 Execution Trace가 모두 올바른지를 증명하는 Circuit이다. 여기서 증명해야하는 것들은 EVM word가 256비트인지, 옵코드의 실행결과가 모두 올바른지, 스택과 메모리에서 값이 올바르게 읽고 쓰여졌는지 등 EVM 실행에 대한 모든 validity이다.

zkEVM Prover가 가지는 input 데이터는 다음과 같다:

1) Input : 이더리움의 기존 World state t, 새로운 World state t+1, L2 트랜젝션

2) Witness : Execution trace

이 데이터들을 이용해 모든 Execution trace의 Codehash, Gas usage, R/W of PC & Stack pointer, Opcode 등 모든 실행 유효성을 증명한다.

b. zkEVM structure

zkEVM의 구조는 프로젝트마다 모두 상이하지만, 이런 거대한 크기의 circuit의 증명을 가능하게 한 요인 중 하나는 Lookup이다. Lookup은 circuit 내부에서 직접 연산을 처리하지 않고 미리 프리컴파일 되어있는 테이블에서 값이 “존재”하는지만을 증명하는 획기적인 방법이다.

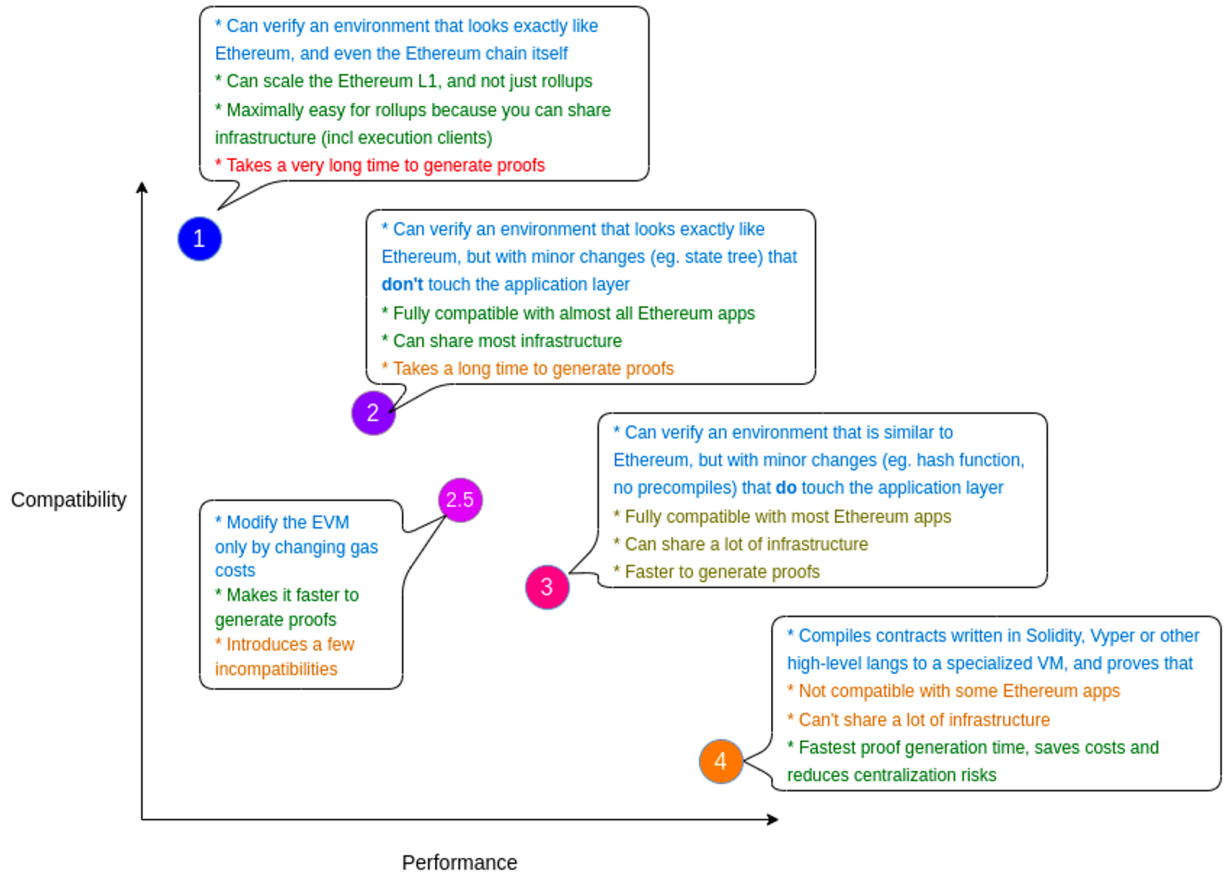
대부분의 zkEVM들은 기본적으로 EVM trace가 올바른지 증명하기 위한 EVM circuit이 있고 증명하고자 하는 값들이 프리컴파일된 여러 Lookup 테이블들이 존재한다. 가령 이더리움에서 사용하는 해시함수인 Keccak의 경우 Circuit 내부에서 연산하려면 코스트가 커지는데 해시값을 프리컴파일한 Keccak table에서 lookup하는 방식으로 constraints를 획기적으로 줄인다.

따라서 zkEVM은 보통 Execution trace를 검증하기 위한 메인 EVM circuit과 Lookup을 위한 Auxiliary circuit들로 구성되어있다.

zkEVM에서 중요한 또다른 미션은 여러개의 Proof들을 어떻게 줄이느냐인데, 이는 사용하는 proving scheme에 따라 다양한 어프로치들이 있다. ZCash에서 사용하는 Halo2(IPA)의 경우 Pasta curve를 사용한 recursion으로 proof를 accumulation하고, Scroll의 경우 Halo2(KZG)를 사용하는데

cycling이 불가능한 BN254를 사용하기 때문에 proof들을 합치기 위한 또다른 Aggregation circuit이 존재한다.

c. Types of zkEVM



The different types of ZK-EVMs | source: [Vitalik's blog]

비탈릭 부테린이 제시한 zkEVM의 분류체계는 총 네 가지로 나눌 수 있다.

1) **Type1** : 이더리움 컨센서스 레이어까지 모두 호환되는 이더리움 호환 zkEVM. 이더리움의 보안과 모든 인프라를 그대로 상속받을 수 있는 장점이 있는 반면, 그만큼 proving 코스트가 매우 크다. 현재 대표적으로 Taiko가 Type1 zkEVM을 개발 중이다.

2) **Type2** : EVM과 호환되는 EVM 호환 zkEVM. 이더리움의 MPT를 ZK-friendly structure인 ZK-Trie로 변경하는 경우와 같이 ZK를 위한 변경이 있지만 여전히 EVM레벨에서는 동등한 호환성을 가지고있다. 대표적인 프로젝트로 스크롤이 있다.
3) **Type3** : ZK-unfriendly Opcode나 구조들을 없앤 형태로 이더리움 디앱들이 그대로 온보딩하지 못하고 약간의 수정이 필요한 허들이 존재한다.
4) **Type4** : 자체적인 ZK-friendly 언어와 Virtual Machine을 사용하는 경우로 이더리움과 호환이 되지 않지만 다른 타입에 비해 proof 생성이 훨씬 빠르다는 장점이 있다. 대표적으로 StarkNet과 zkSync가 있다.

4. Conclusion

ZK 롤업은 아직 미완의 기술이고 가야할 길이 멀다. 기술의 특성상 진입장벽이 높아 아직 이 분야를 이끌어갈 수 있는 전문가의 절대적인 수도 부족하고 지원도 부족하다. 그럼에도 불구하고 현재 매우 빠른속도로 발전중이고 이더리움의 확장성에 기여할 수 있는 좋은 솔루션 중에 하나임은 분명하다.