zk-SNARK and Plonk

최원재

2024. 11. 9

Virtual Machine & Optimization Laboratory

Dept. of Electrical and Computer Engineering

Seoul National University

Table of Contents

I. Zero Knowledge Proof

II. SNARK

III. Polynomial Commitment Scheme

IV. Polynomial IOP

Zero Knowledge Proof

Zero Knowledge Proof

Definition

- 증명자가 어떤 정보를 알고 있음을 검증자에게 증명하는 과정에서, 해당 정보에 대한 어떠한 내용도 직접적으로 노출하지 않고 증명하는 기술

Scenario #1: Where’s Waldo

문제 상황

- 월리를 찾아라 게임을 하는데,
  다른 사람들에게 정답을 노출하지 않고,
  내가 월리를 찾았다는 것을 증명하고 싶습니다.

Scenario #1: Where’s Waldo

문제 상황

- 월리를 찾아라 게임을 하는데,
  다른 사람들에게 정답을 노출하지 않고,
  내가 월리를 찾았다는 것을 증명하고 싶습니다.

해결 방법

- 아무것도 없는 빈방에서 그림과 가위만 가지고 들어갑니다.
- 월리를 찾습니다.
- 오른쪽과 같이 월리만 보이도록 잘라서 보여줍니다.

검증 방법

- 전체 그림에서 어디인지 알 필요 없이 잘라낸 그림이
  월리가 맞는 지만 확인하면 됩니다.

Scenario #2: Sudoku Puzzle

문제 상황

- 스도쿠 퍼즐을 푸는데, 정답을 노출하지 않고,
  내가 해결했다는 사실만을 증명하고 싶습니다.

	1	2	3	4	5	6	7	8	9
A							6	8	
B					7	3			9
C	3		9					4	5
D	4	9							
E	8		3		5		9		2
F								3	6
G	9	6					3		8
H	7			6	8				
I		2	8						

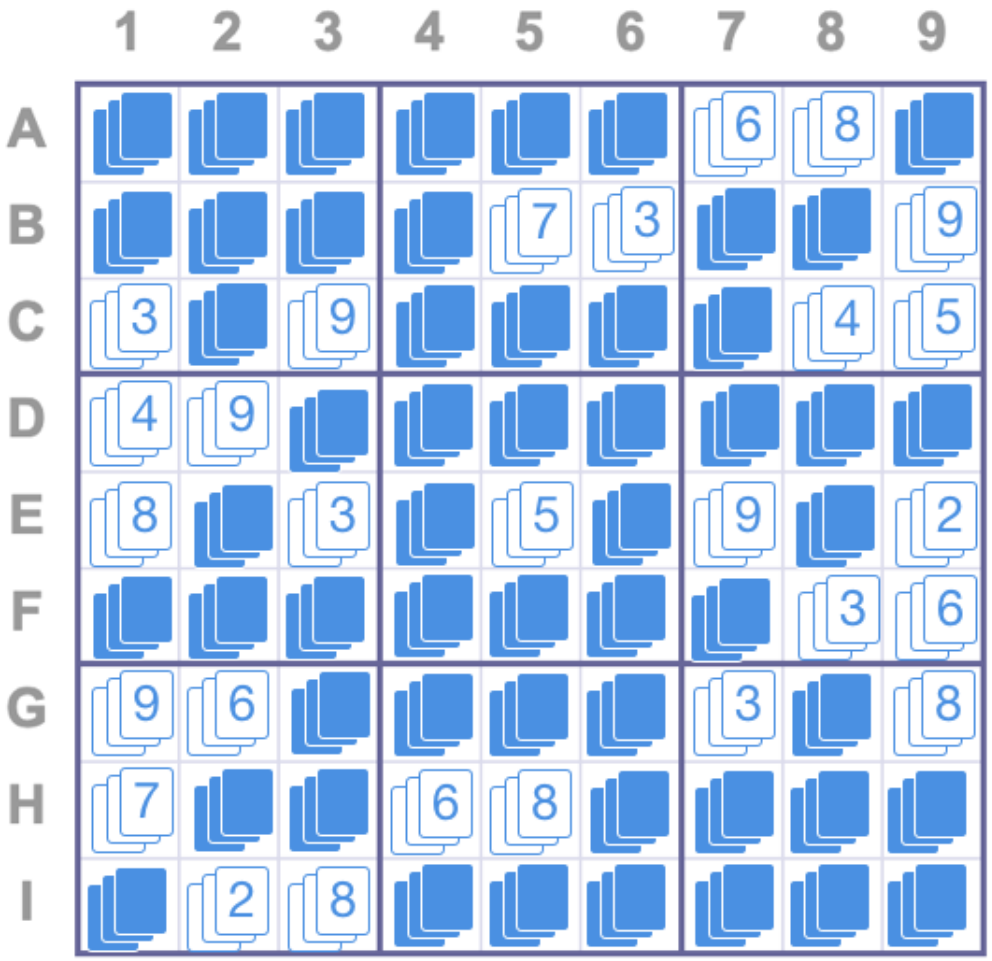
Scenario #2: Sudoku Puzzle

문제 상황

- 스도쿠 퍼즐을 푸는데, 정답을 노출하지 않고,
  내가 해결했다는 사실만을 증명하고 싶습니다.

해결 방법

- 스도쿠 퍼즐을 풉니다.
- 정답은 보이지 않도록 뒤집어 놓고,
  모든 칸에 정답 카드를 세 장씩 놓습니다.



Scenario #2: Sudoku Puzzle

문제 상황

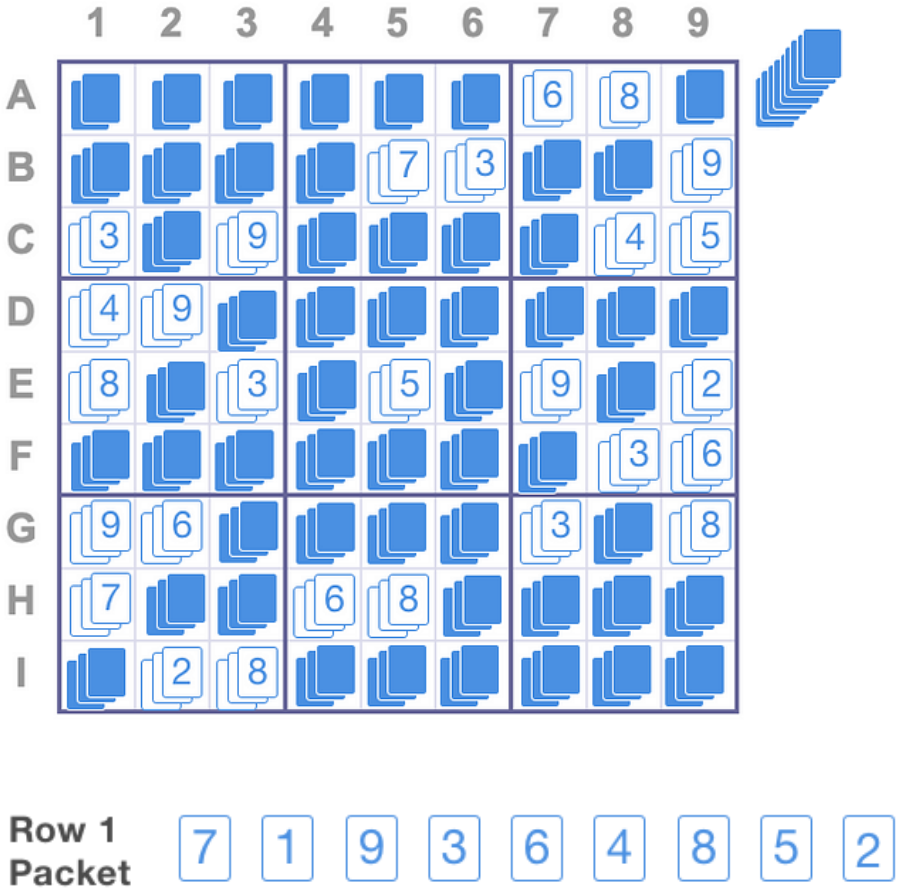
- 스도쿠 퍼즐을 푸는데, 정답을 노출하지 않고,
  내가 해결했다는 사실만을 증명하고 싶습니다.

해결 방법

- 스도쿠 퍼즐을 풉니다.
- 정답은 보이지 않도록 뒤집어 놓고,
  모든 칸에 정답 카드를 세 장씩 놓습니다.

검증 방법

- 모든 행, 열, 3*3 격자를 차례로 9장씩 가져가 섞습니다.
- 섞은 카드가 1~9 숫자가 모두 있는지 확인합니다.



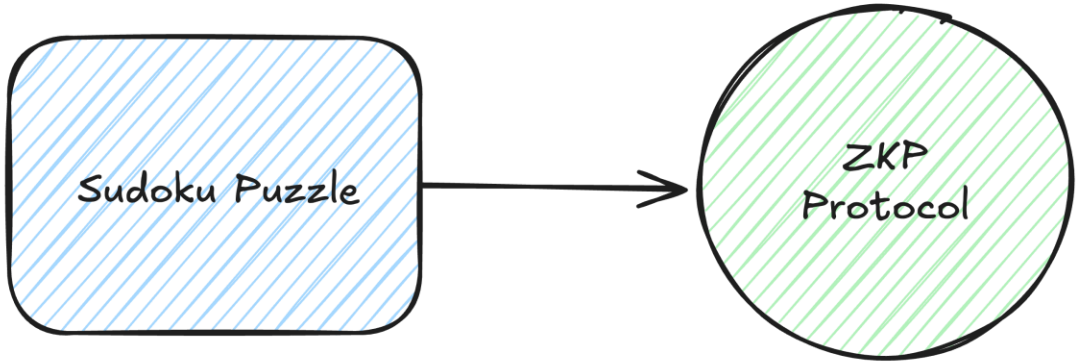
Can it be generalized?

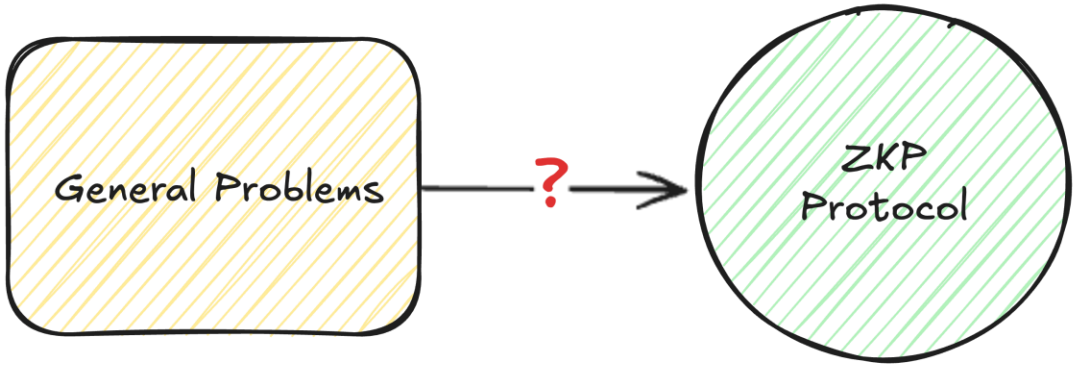
Sudoku Puzzle ZKP Protocol

- 앞서 스도쿠 퍼즐에 대해 영지식 증명을 생성할 수 있다는 것을 알 수 있었음.

For General Problems?

- 일반화하여 다양한 상황에서 사용할 수 있는 영지식 증명을 만들려면?





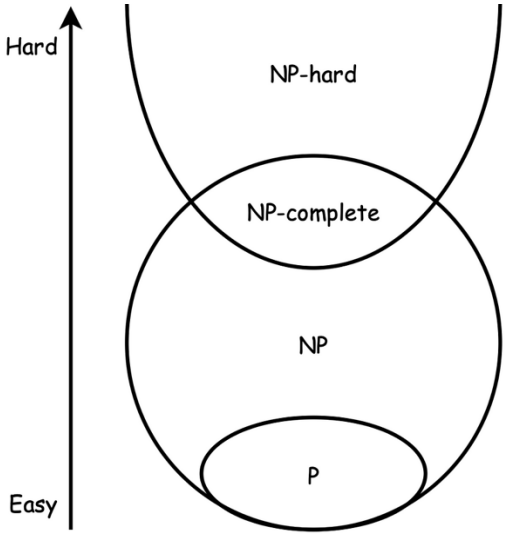
NP-Completeness

NP Problem

- 문제의 검증이 다항 시간 안에 가능한 문제의 집합
- 해결은 다항 시간 안이 아니어도 됨 (가능하면 P)
- 예: 배열 정렬, 인수분해, 스도쿠 퍼즐…

NP-Complete

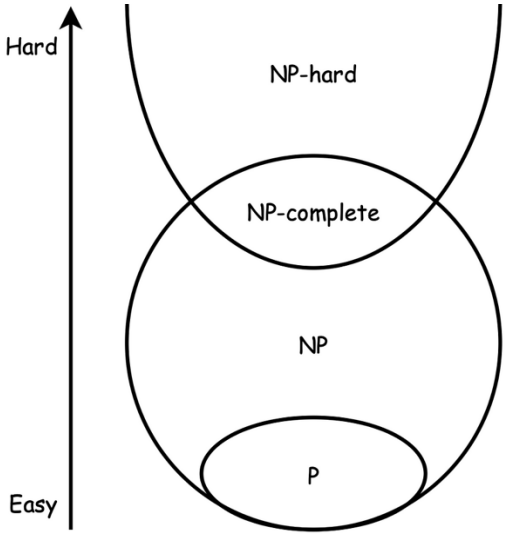
- 모든 NP 문제들은 NP-Complete 문제로 다항 시간 안에 환원 가능
- 예: 스도쿠 퍼즐, SAT, …



NP-Completeness

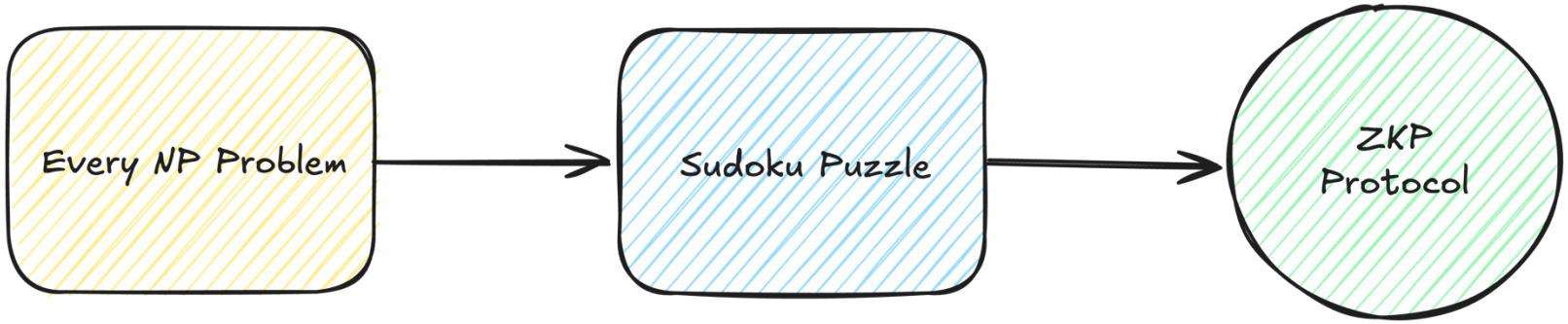
NP Problem

- 문제의 검증이 다항 시간 안에 가능한 문제의 집합
- 해결은 다항 시간 안이 아니어도 됨



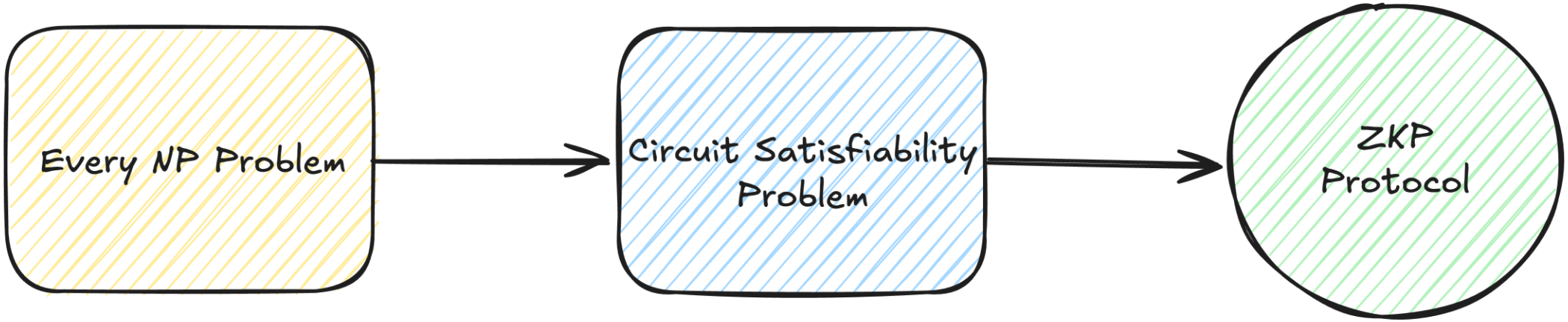
NP-Complete

- 모든 NP 문제들은 NP-Complete 문제로 다항 시간 안에 환원 가능
- <u>스도쿠 퍼즐은 NP-Complete 문제이므로, 즉, 모든 NP 문제는 스도쿠 퍼즐로 환원 가능</u>



Generalization

- 즉, 모든 NP 문제의 검증에 대한 영지식 증명 프로토콜을 구성할 수 있음.
- 실제로는 산술 회로 만족성 문제에 대한 영지식 증명 프로토콜을 만들어 내어,
  다른 문제들을 산술 회로 만족성 문제로 치환하여 영지식 증명을 수행.
- Universal



Sudoku Puzzle to Arithmetic Circuit

변수 정의

- 스도쿠의 각 셀의 값을 $X_{i,j}$ 로 나타냄.

셀 값 제약

- $(X_{i,j} - 1)(X_{i,j} - 2)(X_{i,j} - 3) \ldots (X_{i,j} - 9) = 0$

고유성 제약

- $$\prod_{k=1,k\neq j}^{9} (X_{i,j} - X_{i,k}) \neq 0$$
- $$\prod_{k=1,k\neq i}^{9} (X_{i,j} - X_{k,j}) \neq 0$$
- $$\prod_{(i',j')\in 3\text{x}3 \text{ grid},(i',j')\neq(i,j)} (X_{i,j} - X_{i',j'}) \neq 0$$

퍼즐 초기 조건

- $X_{4,2} = 5, \ldots$

Sudoku Puzzle to Arithmetic Circuit

변수 정의

- 스도쿠의 각 셀의 값을 $X_{i,j}$ 로 나타냄.

셀 값 제약

- $(X_{i,j\text{ -1}})(X_{i,j\text{ -2}})(Xi_{,j\text{ -3}}) \ldots (X_{i,j\text{ -9}}) \ = \ 0$

고유성 제약

- $$\prod_{k=1,k\neq j}^{9} (X_{i,j} - X_{i,k}) \neq 0$$
- $$\prod_{k=1,k\neq i}^{9} (X_{i,j} - X_{k,j}) \neq 0$$
- $$\prod_{(i',j')\in 3\text{x}3\text{ grid},(i',j')\neq(i,j)} (X_{i,j} - X_{i',j'}) \neq 0$$

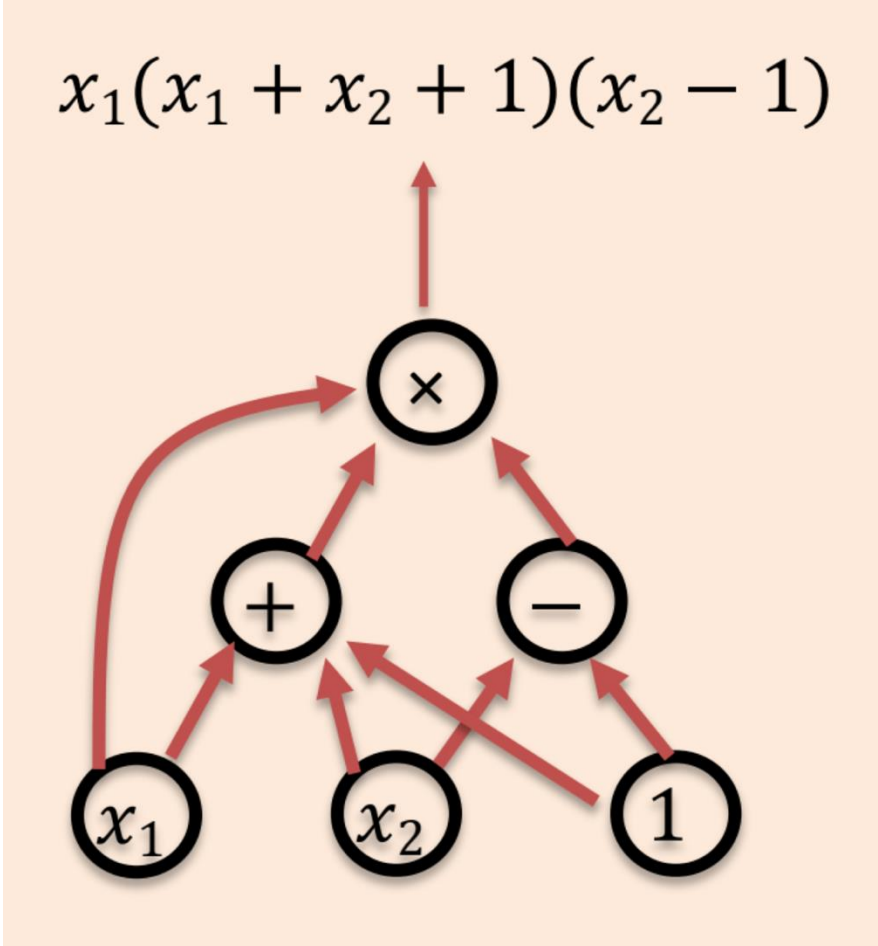
퍼즐 초기 조건

- $X_{1,7} \ = 6, X_{2,5} \ = 7, \ldots$

	1	2	3	4	5	6	7	8	9
A							6	8	
B					7	3			9
C	3		9					4	5
D	4	9							
E	8		3		5		9		2
F								3	6
G	9	6					3		8
H	7			6	8				
I		2	8						

Sudoku Puzzle to Arithmetic Circuit

	1	2	3	4	5	6	7	8	9
A							6	8	
B					7	3			9
C	3		9					4	5
D	4	9							
E	8		3		5		9		2
F								3	6
G	9	6					3		8
H	7			6	8				
I		2	8						



SNARK

Argument System

Definition

- 검증자가 증명자를 완전히 신뢰하지 못하는 상황에서, 증명자가 제공한 증명이 신뢰할 만한지 확인하는 체계

Entities

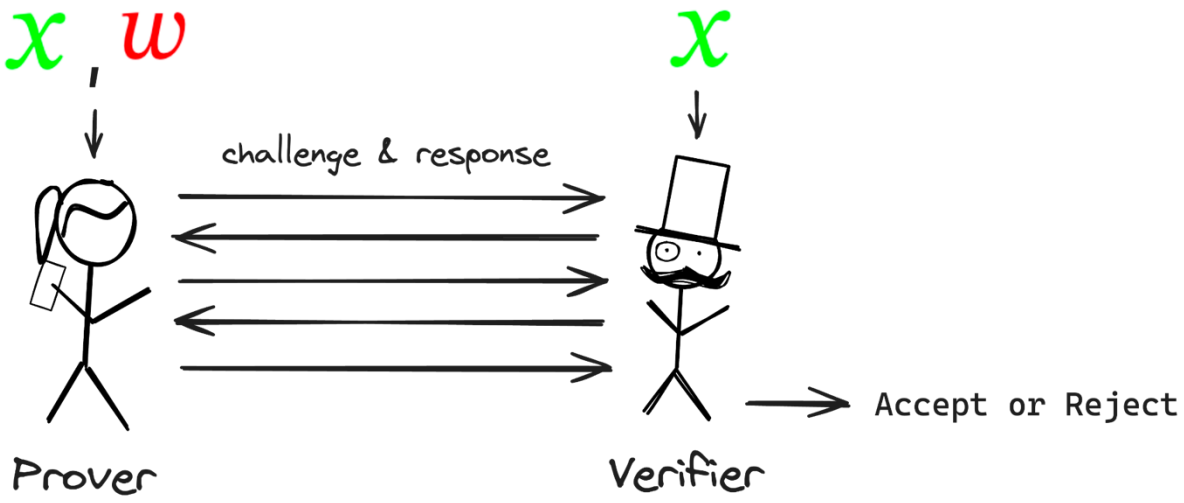
- Prover and Verifier

Prover's Goal

- Convince V that $\exists w$ s.t. $C(x, w) = 0$

Components

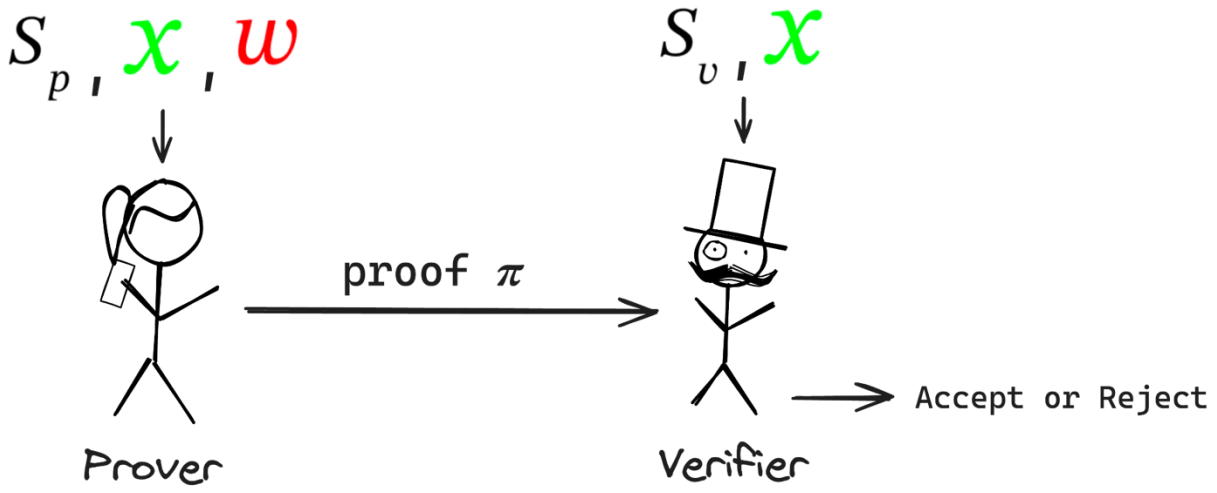
- Arithmetic Circuit $C: \mathbb{F}^{n+m} \rightarrow \mathbb{F}$
  예시: 스도쿠를 검증하기 위한 회로
- Public Statement x in $\mathbb{F}^n$
  예시: 스도쿠에 미리 채워져 있는 숫자 값들
- Secret Witness w in $\mathbb{F}^m$
  예시: P가 빈칸에 채운 숫자 값들



SNARK

Definition

- Succinct Non-interactive Argument of Knowledge
- 짧고 간결한 형태로 증명 가능한 비대화형 지식 논증 시스템
- 스도쿠 퍼즐의 답을 알고 있음을 짧고 간결한 하나의 증명으로 증명하는 시스템



Succinctness (간결성)

- 증명이 매우 짧고, 검증하는 데 걸리는 시간이 매우 빠름
- Plonk의 경우
  - Proof Size: $O(1)$, ≈ 400 Bytes
  - Verifier Time: $O(1)$, ≈ 6 ms

Non-Interactive (비대화성)

- 증명자가 하나의 증명만 보내고, 검증자는 증명자와 상호작용 없이 독립적으로 증명을 검증할 수 있음

Zero-Knowledge (영지식성)

- 증명하고자 하는 지식에 대해 필요한 것 이외에 어떠한 정보도 노출하지 않음
- Optional하게 적용할 수 있음, 만족하는 경우를 zk-SNARK라 함

ZKP Downside

Proof Generation

- 증명 생성 시에 Hardware Requirement가 큼
- Computation Intensive, 오래 걸리고 비용이 비쌈

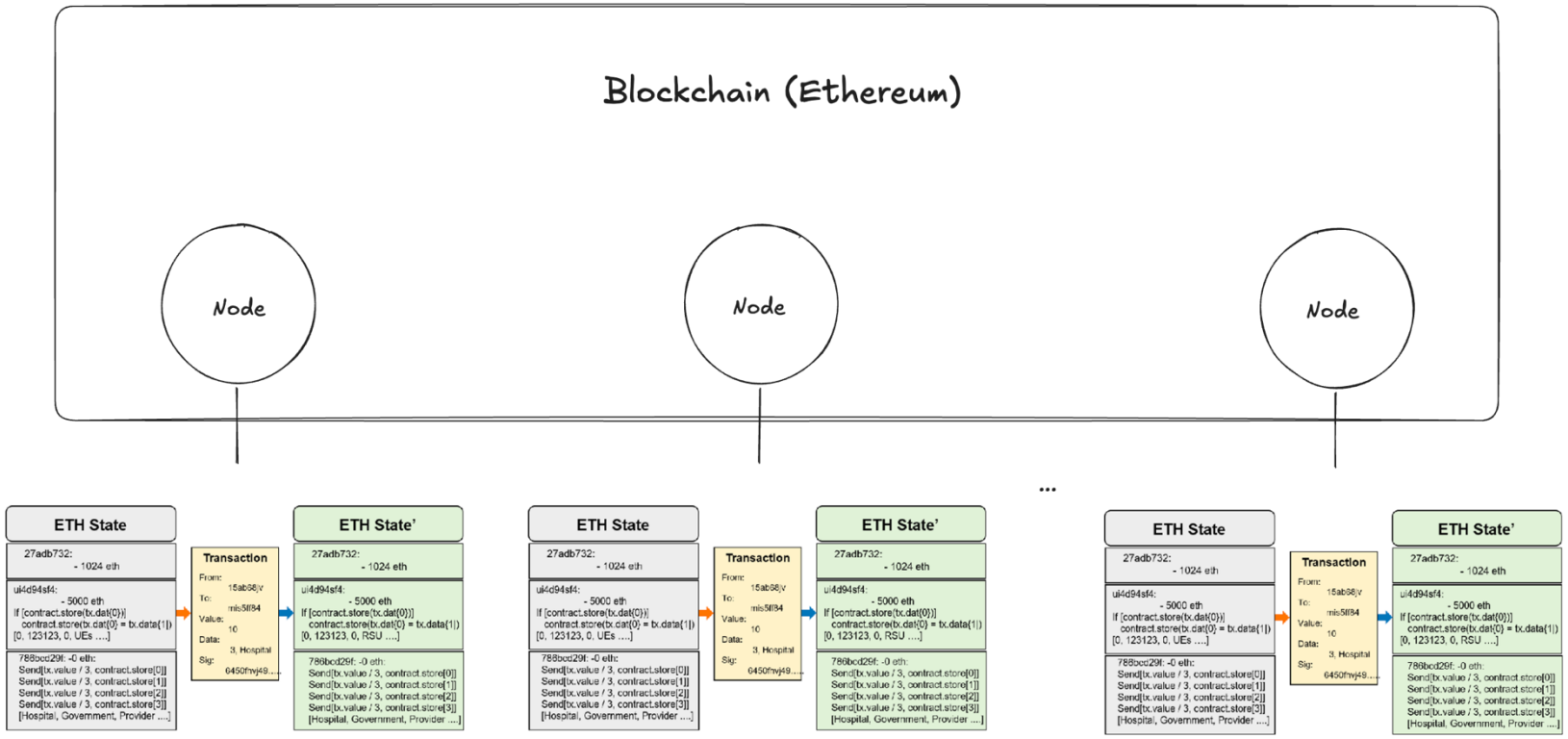
Development Difficulty

- 새로운 문제마다 Circuit을 새로 만들어야 함.
  - Circuit 생성을 위한 프로그래밍 언어 (Circom, ZoKrates, Leo, Cairo, Noir, …)
- ZKP Engineer의 숫자와 숙련도가 부족
  - But, 올바른 Circuit을 짜는 것이 보안에 직결됨

SNARK Application in Blockchain – ZK Rollup

Transaction Verification in Blockchain

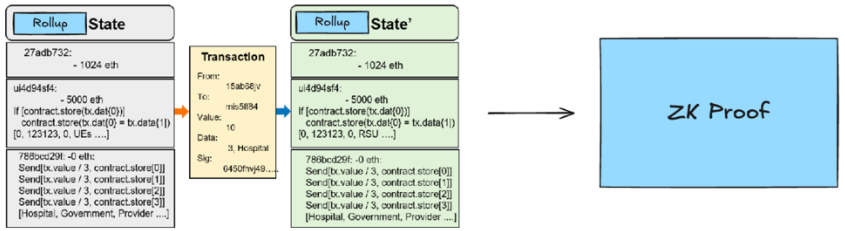
- 블록체인은 기본적으로 모든 노드들이 모든 트랜잭션(상태전이함수)를 직접 실행해서 검증하고 합의

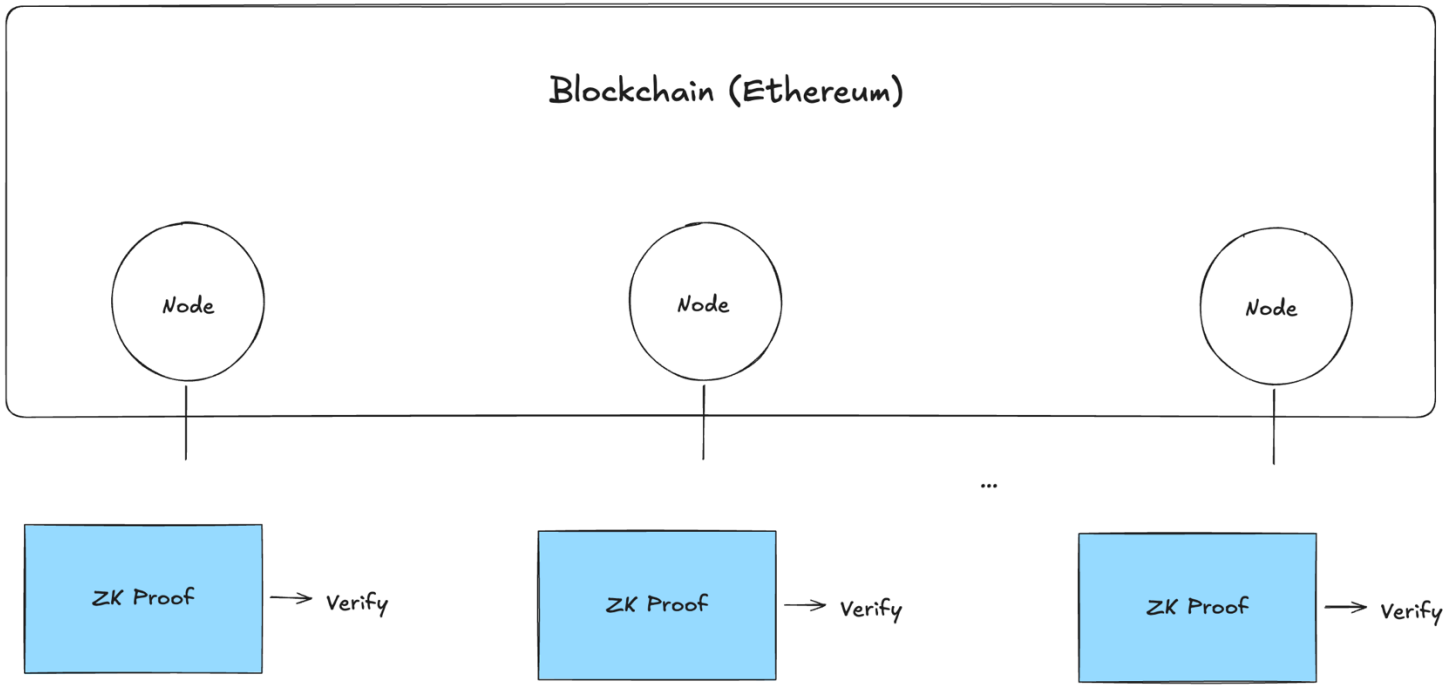


ZK Rollup

Generate Proof on a Single Super Computer

- 어떤 블록체인의 트랜잭션들이 실행되었을 때 상태가 State -> State'이 된다는 것을 SNARK로 증명
- 하나의 증명으로 만들어 이더리움에서 모든 노드들이 검증





ZK Rollup

Computation

- 트랜잭션들을 원래는 모든 노드들이 전부 실행해야 했지만, 한 컴퓨터가 대신 증명을 만듦
- 일반 노드들은 이 간결한 증명만 검증함으로써 네트워크 전체적으로 연산 이득
- 전에는 증명 생성이 매우 오래 걸렸지만, 현재는 많이 개선 되었음

Security

- 큰 확장성을 제공하면서도 이더리움에서 검증되기 때문에 이더리움의 보안을 상속받음

Total Value Locked in Rollups

Rollups 52 | Validiums & Optimiums 67 | Others

#	NAME	RISKS	TYPE	STAGE	TOTAL VALUE LOCKED	PAST DAY UOPS
1	Arbitrum One		Optimistic Rollup	STAGE 1	$18.01B ▲13.2%	28.79 ▲6.66%
2	Base		Optimistic Rollup	STAGE 0	$10.64B ▲3.64%	83.35 ▼2.46%
3	OP Mainnet		Optimistic Rollup	STAGE 1	$7.71B ▲15.4%	12.03 ▼2.55%
4	Blast		Optimistic Rollup	STAGE 0	$1.52B ▲7.53%	5.94 ▲2.55%
5	ZKsync Era		ZK Rollup	STAGE 0	$1.08B ▲14.1%	1.76 ▲16.1%
6	Linea		ZK Rollup	STAGE 0	$1.08B ▲11.7%	2.56 ▼12.7%
7	Scroll		ZK Rollup	STAGE 0	$998.70M ▲2.40%	0.95 ▲2.37%
8	Starknet		ZK Rollup	STAGE 0	$960.07M ▲23.3%	2.57 ▲6.88%
9	Mode		Optimistic Rollup	STAGE 0	$493.69M ▲9.79%	0.65 ▲67.6%
10	Fuel Ignition		Optimistic Rollup	STAGE 0	$361.82M ▲8.70%	1.54 ▼4.68%
11	World Chain		Optimistic Rollup	STAGE 0	$287.34M ▲9.64%	16.49 ▲1.90%
12	Lisk		Optimistic Rollup	STAGE 0	$186.53M ▲15.7%	0.36 ▲72.9%
13	Zircuit		Optimistic Rollup	STAGE 0	$177.58M ▲9.63%	0.08 ▼42.7%
14	BOB		Optimistic Rollup	STAGE 0	$143.59M ▲8.21%	0.46 ▲21.7%
15	Taiko		Optimistic Rollup	STAGE 0	$121.09M ▲5.12%	49.09 ▼1.93%
16	Polygon zkEVM		ZK Rollup	STAGE 0	$93.66M ▲31.6%	0.16 ▲2.89%

zk-SNARK Application - Privacy

Zcash, Tornado Cash, Railgun

- 100% 익명성이 보장되는 거래가 가능
- 규칙에 맞는 트랜잭션이라는 것을 ZKP로 증명, 100% 익명
- 범죄, 돈세탁에 악용

Verifiable Credentials

- 모바일 신분증, 운전면허증
- SKT 이니셜

AI

- AI로부터 나온 결과가 해당 모델로부터 나왔다는 것을 증명
- AI 학습을 해당 데이터로 했다는 것을 증명

SNARKs Landscape

	Size of Proof	Verifier Time	Setup	Post-Quantum Resistant
Groth16	~ 200bytes ($O_\lambda(1)$)	~ 1.5ms ($O_\lambda(1)$)	Trusted per Circuit	No
Plonk	~ 400bytes ($O_\lambda(1)$)	~ 3ms ($O_\lambda(1)$)	Universal Trusted Setup	No
Bulletproofs	~ 1.5Kbytes ($O_\lambda(\log\|C\|)$)	~ 3ms ($O_\lambda(\|C\|)$)	Transparent	No
zk-STARK	~100Kbytes ($O_\lambda(\log^2\|C\|)$)	~ 10ms ($O_\lambda(\log^2\|C\|)$)	Transparent	Yes

Polynomial Commitment

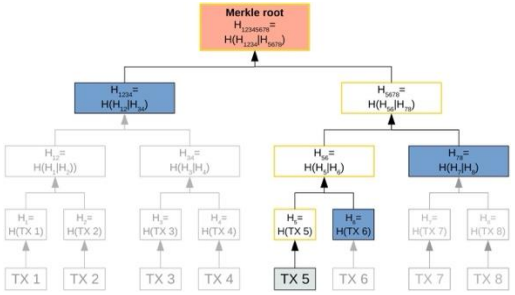
Commitment Scheme

What is Commitment Scheme?

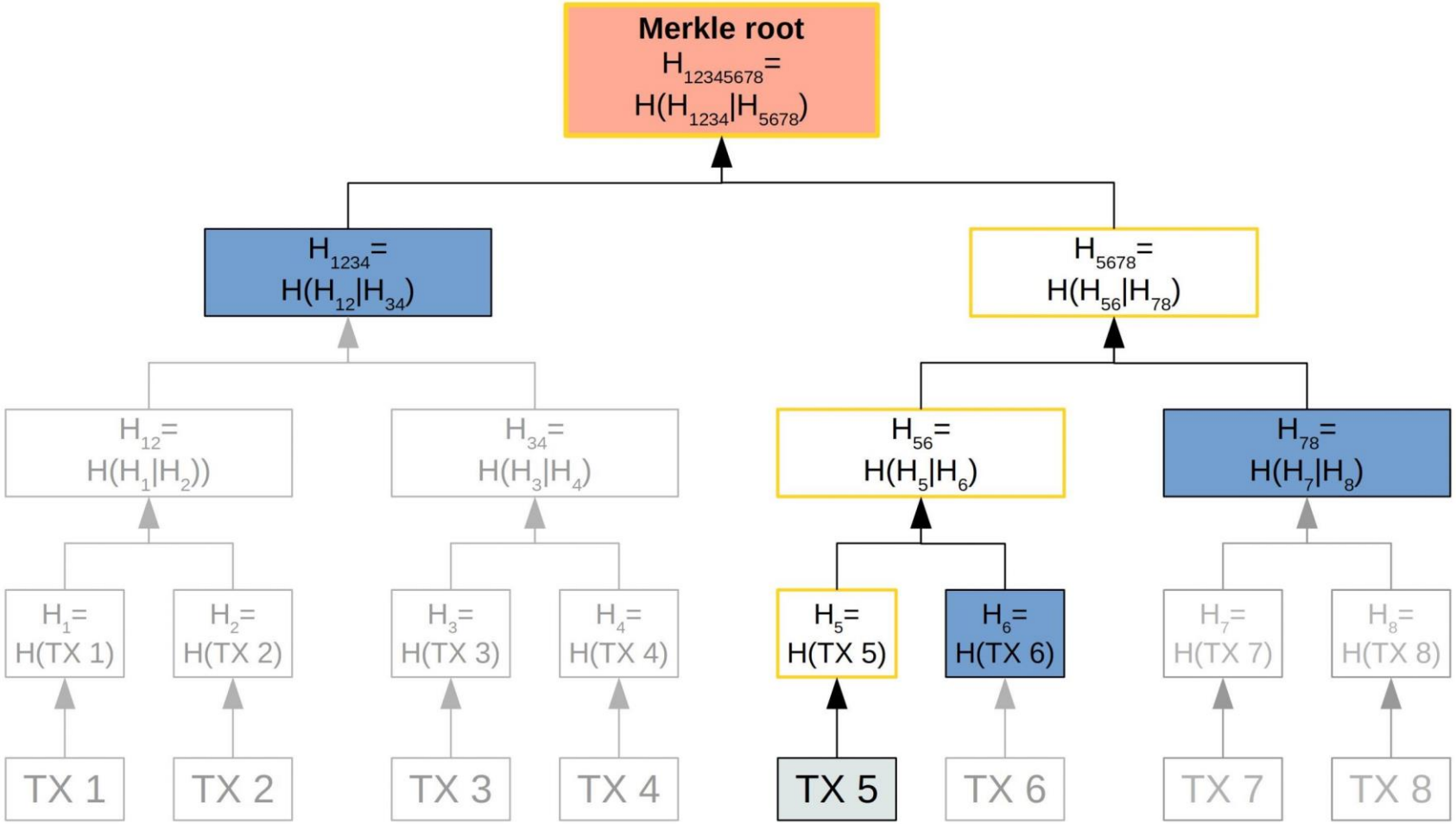
- 메시지의 원본은 감춘채로 Commitment의 형태로 공개하고, 나중에 메시지를 공개하는 암호학적 방법
- $commit(m, r) \rightarrow \boldsymbol{com}$
- $verify(m, \boldsymbol{com}, r) \rightarrow accept\ or\ reject$

Example

- **Hash Function**
  - 메시지를 해시(SHA-256)
  - 커밋먼트 공개
    C6338ED76C1311A2F2B7371A1AC06117F97E00AC4466794CC1358DE8538F4810
  - 원본 메시지 공개
    최원재
  - SHA-256(최원재) == C6338ED76C1311A2F2B7371A1AC06117F97E00AC4466794CC1358DE8538F4810
    Accept
- **Merkle Tree**
  - Partial Reveal



Merkle Tree


Merkle root
$H_{12345678}$=
$H(H_{1234}|H_{5678})$
H_{1234}=
$H(H_{12}|H_{34})$
H_{5678}=
$H(H_{56}|H_{78})$
H_{12}=
$H(H_1|H_2))$
H_{34}=
$H(H_3|H_4)$
H_{56}=
$H(H_5|H_6)$
H_{78}=
$H(H_7|H_8)$
H_1=
H(TX 1)
H_2=
H(TX 2)
H_3=
H(TX 3)
H_4=
H(TX 4)
H_5=
H(TX 5)
H_6=
H(TX 6)
H_7=
H(TX 7)
H_8=
H(TX 8)
TX 1
TX 2
TX 3
TX 4
TX 5
TX 6
TX 7
TX 8

Properties of Commitment Scheme

Binding

- 하나의 커밋먼트에 대해 두가지 유효한 opening을 만들어낼 수 없음
- 데이터를 조작할 수 없기 때문에 무결성 보장

Hiding

- 커밋먼트가 커밋된 메시지에 대해 어떤 정보도 제공하지 않음
- 정보를 노출하지 않고도 정보를 고정할 수 있음

Perfect Hiding (Information-Theoretic Hiding)

- 무한한 계산 능력에서도 원래의 메시지를 알 수 없음
- 예시: Pedersen Commitment

Statistical Hiding

- 수학적 확률로 커밋 값이 원래의 메시지와 통계적으로 거의 독립적이어서, 거의 완벽하게 숨김 보장
- 예시: Goldwasser-Micali Bit Commitment Scheme

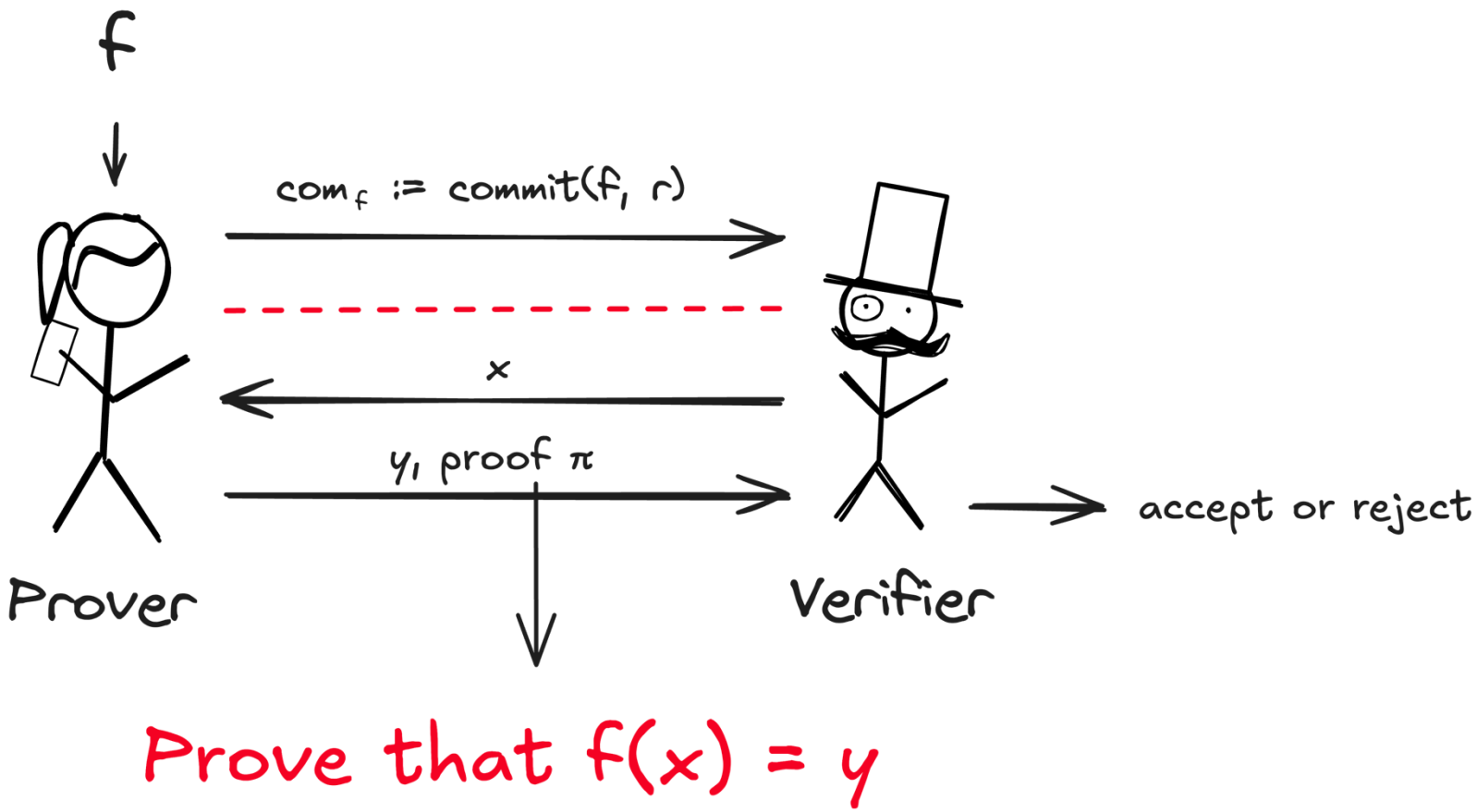
Computational Hiding

- 계산 능력이 제한된 경우에만 보안을 보장, 암호학적 가정에 의존
- 예시: KZG Commitment, Hash Based Commitments

Security

- Perfect Hiding > Statistical Hiding > Computational Hiding

Functional Commitment Scheme



Functional Commitment Scheme

Three Types of Functional Commitments

- Polynomial Commitment
- Multilinear Commitment
- Linear Commitment

Functional Commitment Scheme

Three Types of Functional Commitments

- Polynomial Commitment
- Multilinear Commitment
- Linear Commitment

KZG Commitment Scheme (Kate-Zaverucha-Goldberg ’10)

Goal

- Prover proves that polynomial $f(u) = v$ to the Verifier

4-Step Process

- 1. Trusted Setup - Anyone
  - 비밀 값을 생성하여 Public Parameter 생성 비밀 값은 파기
  - 보통 MPC(Multi-Party Computation)을 통해 비밀 값 생성

- 2. Commitment Creation - Prover
  - 증명하고 싶은 함수 f를 커밋하여 커밋 값 $\mathbf{com}_f$ 공개

- 3. Proof Generation - Prover
  - 특정 평가 지점 u에 대해 $f(u) = v$임을 증명하는 증명 π 생성

- 4. Verification - Verifier
  - 커밋 값 $\mathbf{com}_f$ 와 증명 π 를 사용해 $f(u) = v$임을 검증

KZG Commitment Scheme (Kate-Zaverucha-Goldberg ’10)

Important Note: I have ommitted the elliptic curve pairings for simplicity

Trusted Setup & Commitment Creation

$\mathbb{G} := \{0, G, 2 \cdot G, 3 \cdot G, \ldots, (p - 1) \cdot G\}$

- Cyclic Group of order p
- $p \cdot G = 0$

<u>setup</u>$(\lambda) \rightarrow pp$:

- Sample random $\alpha \in \mathbb{F}_p$
- $pp = (H_0 = G, H_1 = \alpha \cdot G, H_2 = \alpha^2 \cdot G, \ldots, H_d = \alpha^d \cdot G) \in G^{d+1}$
- Delete α (Trusted Setup)
- Also Known as KZG Ceremony
  - https://ceremony.ethereum.org

<u>commit</u>$(pp, f) \rightarrow \mathbf{com}_f$

- $\mathrm{com}_f := f(\alpha) \cdot G \in \mathbb{G}$
- $f(X) = f_0 + f_1 X + \ldots + fdXd$
- $\mathbf{com}_f = f(\alpha) \cdot G = f_0 + f_1 \cdot \alpha \cdot G + \ldots + fd \cdot \alpha^d \cdot G$
- $\mathbf{com}_f = f_0 \cdot H_0 + f_1 \cdot H_1 + \ldots + f_d H_d$
  - Pippenger's Algorithm

Proof Generation & Verification

Goal: prove f(u) = v

- f(u) = v
- $\leftrightarrow$ $\mathrm{q}(x) \cdot (x - u) = f(x) - v$ 를 만족하는 다항식 q가 존재
- $\leftrightarrow \exists\, \mathrm{q} \in \mathbb{F}_p[x] \quad s.t. \;\; \mathrm{q}(x) \cdot (x - u) = f(x) - v$
- Goal: prove $\exists\, \mathrm{q} \in \mathbb{F}_p[x] \quad s.t. \;\; \mathrm{q}(x) \cdot (x - u) = f(x) - v$

<u>Prover$(pp, f, u, v) \rightarrow \pi$</u>

- Compute $\mathrm{q}(x)$ and $\mathbf{com}_q = q(\alpha) \cdot G$
- $\pi \coloneqq \mathbf{com}_q$

<u>Verifier$(pp, \mathbf{com}_f, u, v) \rightarrow$ accept or reject</u>

- $\mathbf{com}_q \cdot (\alpha - u) = \mathbf{com}_f - v \cdot G$
- If it holds → accept, else reject
- How to compute $\mathbf{com}_q \cdot (\alpha - u)$? By bilinear pairing on elliptic curves (Not today ..)

$\forall i \in \{1, 2, \ldots, t-1\}, \qquad r_i \in \mathbb{F}_p$

Computation Evaluation

Trusted Setup

- Requires a lot of computation and process to ensure secret value is not exposed
- But it is one time process and can be reused

Prover Side

- Commit
  - Pippenger's Algorithm
  - Time Complexity: $O(\frac{d}{logd})$
- Proof Size
  - $\mathbf{com}_q$ is a value on cyclic group $\mathbb{G}$ (single point on an elliptic curve if pairing applied)
  - Very small constant size
- Proof Generation
  - Polynomial Division for computing the quotient polynomial q
  - Euclidean Polynomial Division, FFT (Fast Fourier Transform)
  - $O(n),\ O(nlogn)$
  - + Pippenger's Algorithm

Computation Evaluation

Verifier Side

- Smart contract is the verifier in the context of blockchain
- Efficiency is important since it is performed on-chain
- Space Complexity
  - Proof is constant size
  - Only needs two points G and H_1 not all public parameters
- Time Complexity
  - $e(\mathbf{com}_f - [v]_1, [v]_2) = e(\mathbf{com}_q, [\alpha - u]_2)$
  - Two pairing operations
  - $O(logp)$
  - Operations using curves like BLS12-381 can be optimized with precompiled contracts

Plonk: Poly-IOP for General Circuits

Plonk

발표 정보

- 학회: 없음 (Arxiv)
- 발표일: 2019년 10월

논문 정보

- 현대적인 영지식 증명 시스템 중 가장 기초가 되는 논문

$\mathcal{P}$lon$\mathcal{K}$: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon* Aztec

Zachary J. Williamson Aztec

Oana Ciobotaru

February 23, 2024

Abstract

zk-SNARK constructions that utilize an updatable universal structured reference string remove one of the main obstacles in deploying zk-SNARKs[GKM+]. The important work of Maller et al. [MBKM19] presented Sonic - the first potentially practical zk-SNARK with fully succinct verification for general arithmetic circuits with such an SRS. However, the version of Sonic enabling fully succinct verification still requires relatively high proof construction overheads. We present a universal SNARK construction with fully succinct verification, and significantly lower prover running time (roughly 7.5-20 times fewer group exponentiations than [MBKM19] in the fully succinct verifier mode depending on circuit structure).

Similarly to [MBKM19] we rely on a permutation argument based on Bayer and Groth [BG12]. However, we focus on "Evaluations on a subgroup rather than coefficients of monomials"; which enables simplifying both the permutation argument and the arithmetization step.

Polynomial IOP (Interactive Oracle Proof)

Goal: Prove $\exists w$ s.t. $C(x, w) = 0$

- Our original goal for SNARK

Recall Sudoku Puzzle Example

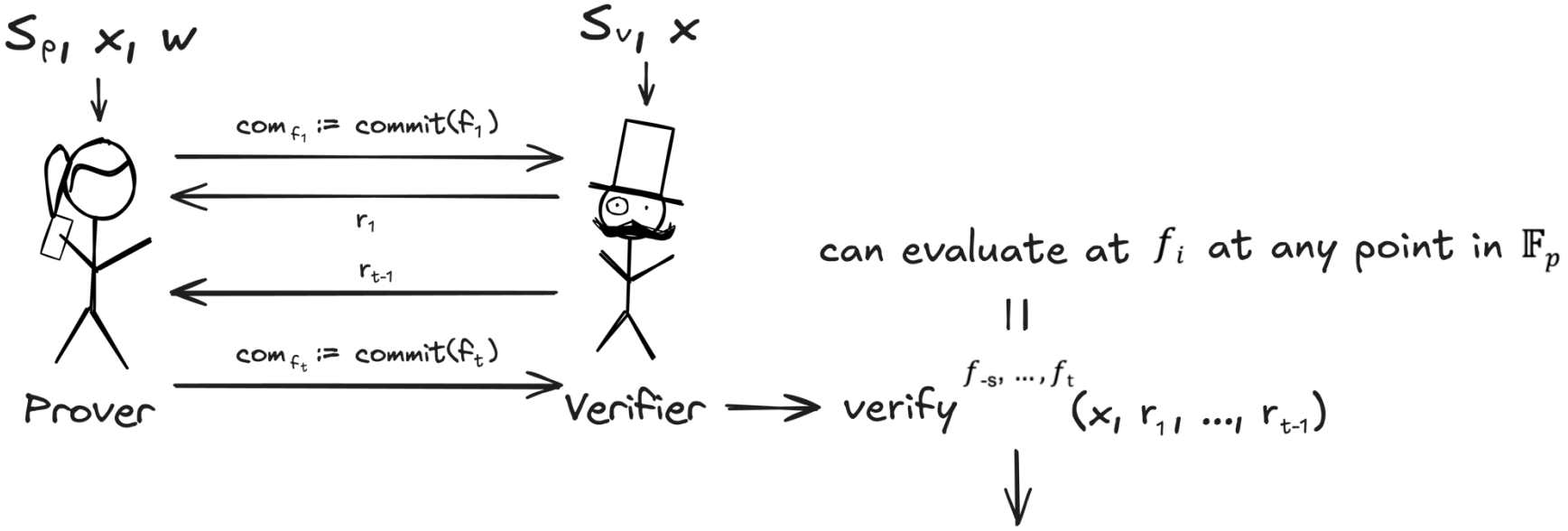
- C: Sudoku puzzle verifying circuit
- x: Problem, Provided numbers
- w: Answer, Numbers for the blanks

	1	2	3	4	5	6	7	8	9
A							6	8	
B					7	3			9
C	3		9					4	5
D	4	9							
E	8		3		5		9		2
F								3	6
G	9	6					3		8
H	7			6	8				
I		2	8						

Setup → public parameters S_p and $S_v = (f_0, f_{-1}, f_{-2}, \dots, f_{-s})$

- Let's denote a commitment of f as f
- (KZG Commitment)

Polynomial IOP (Interactive Oracle Proof)



$\forall i \in \{-s, \dots, 0, 1, 2, \dots, t\}, \quad f_i \in \mathbb{F}_p^{(\leq d)}$

$\forall i \in \{1, 2, \dots, t-1\}, \quad r_i \in \mathbb{F}_p$

Plonk, Finally

Plonk is Polynomial IOP

Polynomial IOP + PCS= SNARK

Plonk + KZG = SNARK

Zero Test for a Polynomial on Finite Field

$for\ 0 \neq f \in F_p^{(\leq d)}[X]$:
$for\ r \leftarrow F_p$: $\Pr[f(r) = 0] \leq d/p$

- 어떤 함수 f가 0이 아닌 다항식일 때, 어떤 값 r에 대해 $f(r) = 0$ 를 만족할 확률은 d/p
- 유한체의 소수 p는 일반적으로 256비트(2^{256}) 정도의 값 또는 512비트 정도의 값을 사용하고, 다항식의 차수 d는 데이터의 크기에 비례하지만 보통 40비트를 넘지 않음
- 즉, $d/p \approx 0$, negligible 하다

== PCS에서 평가 값이 0이면, 함수가 zero와 identical하다.

- 동일한 논리로 $f(r) = g(r)$ 이면, $f = g$

Root of Unity

Let $\omega \epsilon \mathbb{F}_p$ be a primitive k-th root of unity $(\omega^k = 1)$

Set $H := \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \subseteq \mathbb{F}_p$

Let $f \epsilon \mathbb{F}_p^{(\leq d)}$ and $b, c \epsilon \mathbb{F}_p$. $(d \geq k)$

Three poly-IOPs

- Zero-test: prove that f is identically zero on H
- Sum-check: prove that $\sum_{a \epsilon H} f(a) = b$
- Product-check: prove that $\prod_{a \epsilon H} f(a) = c$

Root of Unity

Let $\omega \in \mathbb{F}_p$ be a primitive k-th root of unity $(\omega^k = 1)$

Set $H := \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \subseteq \mathbb{F}_p$

Let $f \in \mathbb{F}_p^{(\leq d)}$ and $b, c \in \mathbb{F}_p$. $(d \geq k)$

Three poly-IOPs

- Zero-test: prove that f is identically zero on H
- Sum-check: prove that $\sum_{a \in H} f(a) = b$
- Product-check: prove that $\prod_{a \in H} f(a) = c$

Zero-test on H

Lemma

- f is zero on H if and only if $f(x)$ is divisible by $x^k - 1$
  - $H := \{1, \omega, \omega^2, \ldots, \omega^{k-1}\}$, $\omega^k = 1$

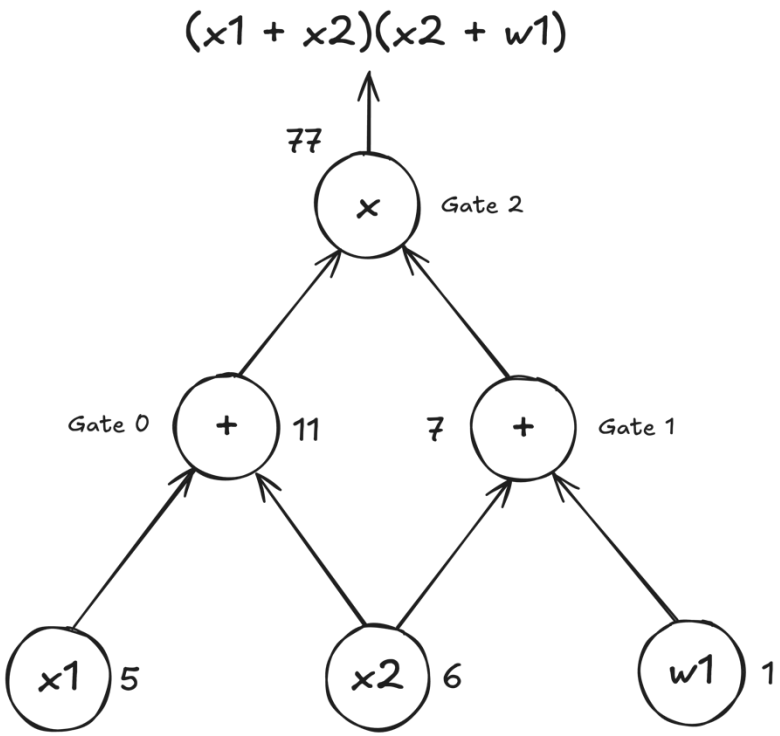
Prover		**Verifier (knows f)**
$q(x) \leftarrow f(x)/(xk - 1)$		
	Commits to q $\longrightarrow$	
	Open at r $\longleftarrow$	$r \leftarrow \mathbb{F}_p$
Evaluate $q(r)$ and $f(r)$	Open $q(r)$ and $f(r)$ $\longrightarrow$	
		Accept if $f(r) = q(r) \cdot (rk - 1)$ $\Leftrightarrow f \text{ is zero on } H$

Plonk: Poly-IOP for General Circuits

Step 1-1: Compile Circuit to a Computation Trace

Compile circuit to a computation trace

Arithmetic Circuit



Computation Trace

	left inputs	right inputs	outputs
inputs:	5,	6,	1
Gate 0:	5,	6,	11
Gate 1:	6,	1,	7
Gate 2:	11,	7,	77

Step 1-2: Encoding the Trace as a Polynomial

- $|C|$:= total # of gates in C
- $|I|$:= $|Ix| + |Iw|$ = # inputs to C
- Let $d := |C| + |I|$ and $H := \{1, \omega, \omega^2, \ldots, \omega^{d-1}\}$

Prover interpolates $P \in \mathbb{F}_p^d$ such that

- P encodes all inputs: $P(\omega^{-j})$ = input # j for $j = 1, \ldots, |I|$
- P encodes all wires: for $\forall l = 0, \ldots, |C| - 1$
  - $P(\omega^{3l})$: left input to gate $\#l$
  - $P(\omega^{3l+1})$: right input to gate $\#l$
  - $P(\omega^{3l+2})$: output to gate $\#l$

inputs:	5,	6,	1
Gate 0:	5,	6,	11
Gate 1:	6,	1,	7
Gate 2:	11,	7,	77

Step 1-2: Encoding the Trace as a Polynomial

inputs:	$P(\omega^{-1}) = 5,$	$P(\omega^{-2}) = 6,$	$P(\omega^{-3}) = 1,$
gate 0:	$P(\omega^{0}) = 5,$	$P(\omega^{1}) = 6,$	$P(\omega^{2}) = 11,$
gate 1:	$P(\omega^{3}) = 6,$	$P(\omega^{4}) = 1,$	$P(\omega^{5}) = 7,$
gate 2:	$P(\omega^{6}) = 11,$	$P(\omega^{7}) = 7,$	$P(\omega^{8}) = 77$

inputs:	5,	6,	1
Gate 0:	5,	6,	11
Gate 1:	6,	1,	7
Gate 2:	11,	7,	77

Step 2: Proving Validity of P

Prover commits to the polynomial P

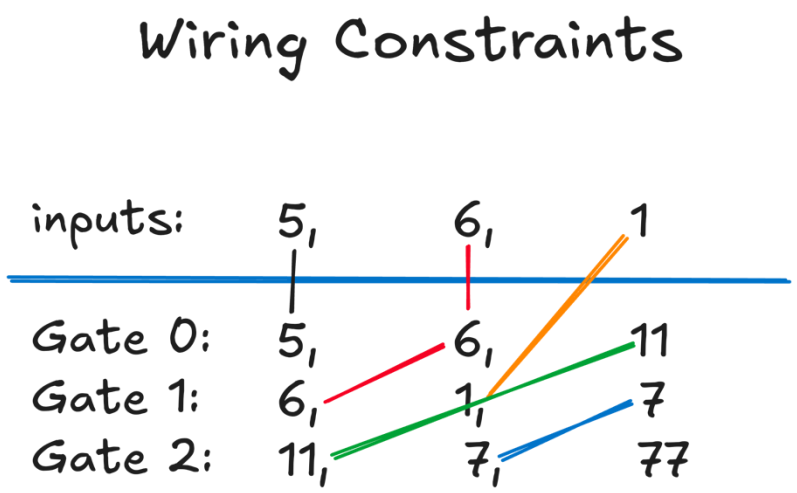
How to prove that P is a correct computation trace

(1) P encodes the correct inputs

(2) Every gates are evaluated correctly

(3) Wiring is implemented correctly

(4) Output of the last gate is 0



Proving (1): P encodes the correct inputs

Goal

- Prove $P(\omega^{-j}) =$ input $\#j$ for $j = 1, \ldots, |I_x|$

How

- Both Prover and Verifier interpolate a polynomial $v(x)$ s.t. $v(\omega^{-j}) =$ input $\#j$ for $\forall j = 1, \ldots, |I_x|$
- $H_{input} := \{1, \omega, \omega^2, \ldots, \omega^{-|Ix|}\}$
- Zero-test $P - v = 0$ on H_{input}

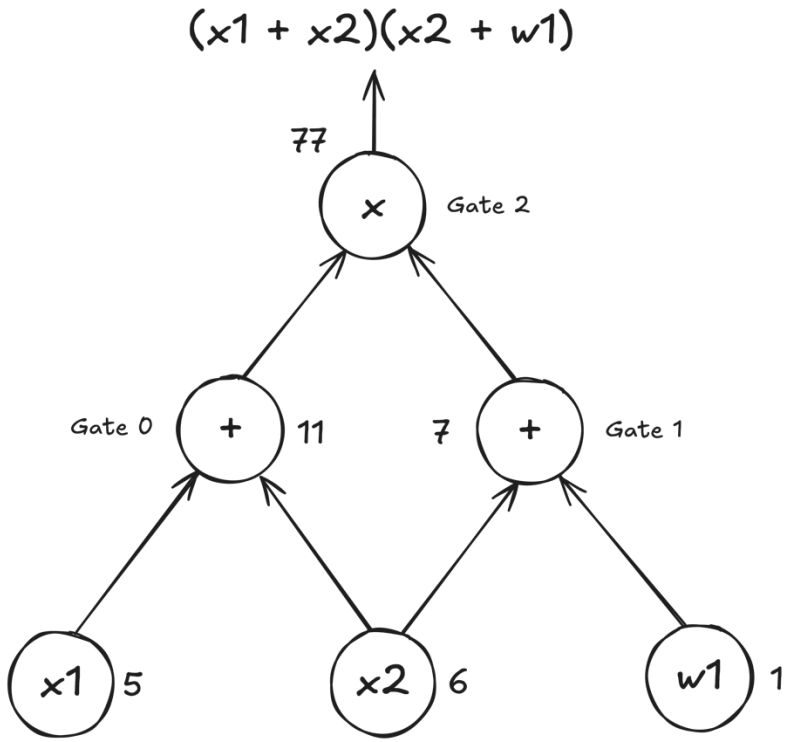
Proving (2): Every gate is evaluated correctly

Selector Polynomial

- Define $S(x)$ such that $\forall\ l = 0, \dots, |C| - 1$:
  - $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
  - $S(\omega^{3l}) = 0$ if gate $\#l$ is an multiplication gate

inputs:	$P(\omega^{-1}) = 5,$	$P(\omega^{-2}) = 6,$	$P(\omega^{-3}) = 1,$
gate 0:	$P(\omega^{0}) = 5,$	$P(\omega^{1}) = 6,$	$P(\omega^{2}) = 11,$
gate 1:	$P(\omega^{3}) = 6,$	$P(\omega^{4}) = 1,$	$P(\omega^{5}) = 7,$
gate 2:	$P(\omega^{6}) = 11,$	$P(\omega^{7}) = 7,$	$P(\omega^{8}) = 77$

$$S(\omega^0) = 1, \qquad S(\omega^3) = 1, \qquad S(\omega^6) = 0$$



Proving (2): Every gate is evaluated correctly

$\forall\, y \in Hgates := \{1, \omega^3, \omega^6, \omega^9, \ldots, \omega^{3(|c|-1)}\}$:

$$S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) = P(\omega^2 y)$$

- $P(y)$: Left Input
- $P(\omega y)$: Right Input
- $P(\omega^2 y)$: Output

inputs:	$P(\omega^{-1}) = 5,$	$P(\omega^{-2}) = 6,$	$P(\omega^{-3}) = 1,$
gate 0:	$P(\omega^0) = 5,$	$P(\omega^1) = 6,$	$P(\omega^2) = 11,$
gate 1:	$P(\omega^3) = 6,$	$P(\omega^4) = 1,$	$P(\omega^5) = 7,$
gate 2:	$P(\omega^6) = 11,$	$P(\omega^7) = 7,$	$P(\omega^8) = 77$

$$S(\omega^0) = 1, \qquad S(\omega^3) = 1, \qquad S(\omega^6) = 0$$

Proving (2): Every gate is evaluated correctly

Selector Polynomial

- Define $S(x)$ such that $\forall\, l = 0, \ldots, |C| - 1$:
  - $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
  - $S(\omega^{3l}) = 0$ if gate $\#l$ is an multiplication gate

- Selector Polynomial을 만드는 것과 커밋하는 과정은 Setup 과정에서 수행
- Public Parameter S_v에 Selector Polynomial S의 커밋 값 S가 포함

$$S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) = P(\omega^2 y)$$

- P와 S가 모두 Commit 되어 있으므로, 위의 수식을 Zero-test를 이용하여 증명할 수 있음

Proving (2): Every gate is evaluated correctly

Selector Polynomial

- Define $S(x)$ such that $\forall\, l = 0, \ldots, |C| - 1$:
  - $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
  - $S(\omega^{3l}) = 0$ if gate $\#l$ is an multiplication gate

- Selector Polynomial을 만드는 것과 커밋하는 과정은 Setup 과정에서 수행
- Public Parameter S_v에 Selector Polynomial S의 커밋 값 S가 포함

$$S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) = P(\omega^2 y)$$

- P와 S가 모두 Commit 되어 있으므로, 위의 수식을 Zero-test를 이용하여 증명할 수 있음

Proving (3): Wiring is correct

Goal

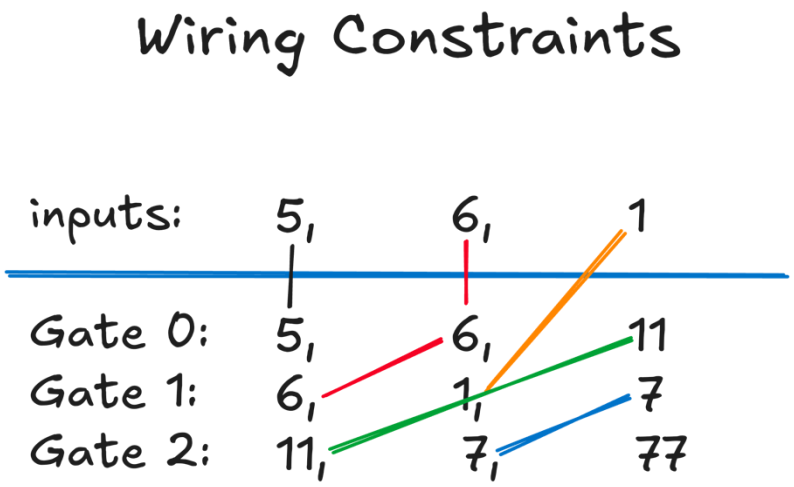
- Check the wirings

$P(\omega^{-2}) = P(\omega^{1}) = P(\omega^{3})$

$P(\omega^{-1}) = P(\omega^{0})$

$P(\omega^{2}) = P(\omega^{6})$

$P(\omega^{-3}) = P(\omega^{4})$



How

- Define a polynomial $W: H \rightarrow H$ such that
  $W(\omega^{-2}, \omega^{1}, \omega^{3}) = (\omega^{1}, \omega^{3}, \omega^{-2})$, $W(\omega^{-1}, \omega^{0}) = (\omega^{0}, \omega^{-1})$
- Rotation Polynomial

Lemma

- $\forall\, y \epsilon H$: $P(y) = P\big(W(y)\big)$ $\Longrightarrow$ wire constraints are satisfied

Proving (4): Output of the last gate is 0

Goal

- Prove $P(\omega^{3|C|-1}) = 0$

How

- Just one opening for the KZG commitment

Full Picture of Plonk

Setup(C) → $Sp := (S, W)$ and $S_v := (S, W)$

- Untrusted Setup
- S, W is derived from the circuit C which is public

Prover(S_p, x, w)

Build P from the circuit C

Commits to P

Build $v(x) \in \mathbb{F}_p^{|Ix|}$

Verifier (S_v, x)

Build $v(x) \in \mathbb{F}_p^{|Ix|}$

Proves (1), (2), (3), (4)

Accept if

Eveything is correct

What I didn’t Cover

- Pairings
- Fiat-Shamir Transform
- Zero-Knowledge Formal Definition
- KZG with Blinding
- Batch Proofs

Thank You