

Recursion zkp

재귀적 영지식 증명 이해하기

정현 (tariz@theradius.xyz)

목차

재귀적 영지식 증명은 이용당했습니다..

오늘은 영지식 증명에서 많이 사용되는 수학 문장을 읽는 방법을 더 많이 배우게 됩니다..

- 재귀적 영지식 증명 컨셉 이해하기
- Polygon Zero의 재귀적 영지식 증명 알고리즘
- 영지식 증명 알고리즘 이해하기
- Plonky 이해하기
- Plonky2 이해하기

재귀적 영지식 증명 컨셉 이해하기

영지식 증명의 효율성을 높이는 도구

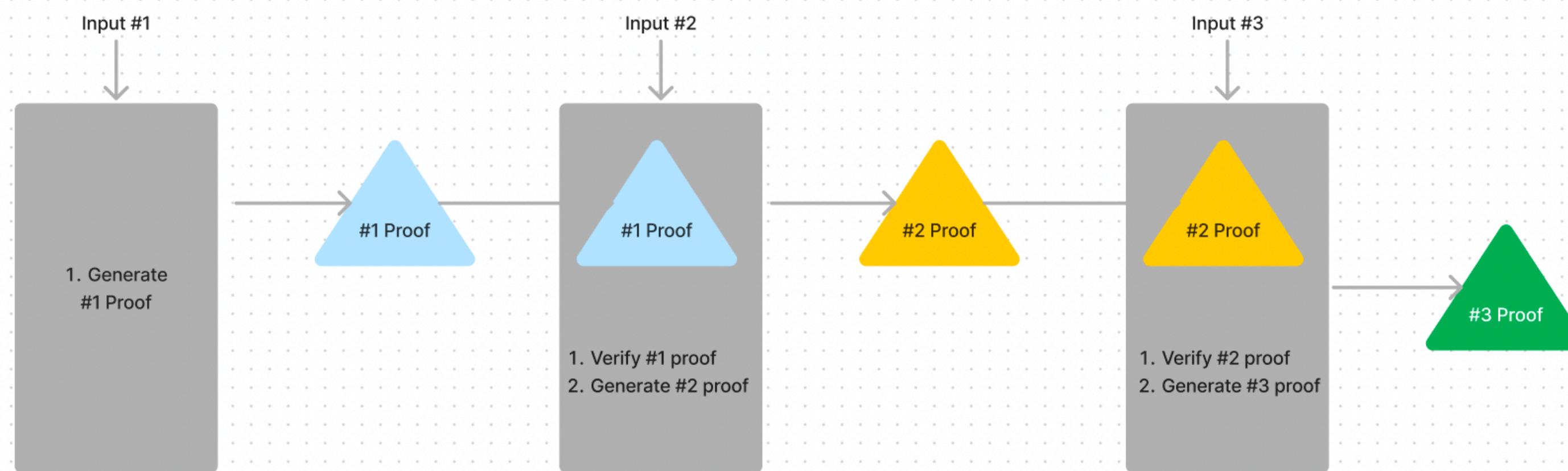
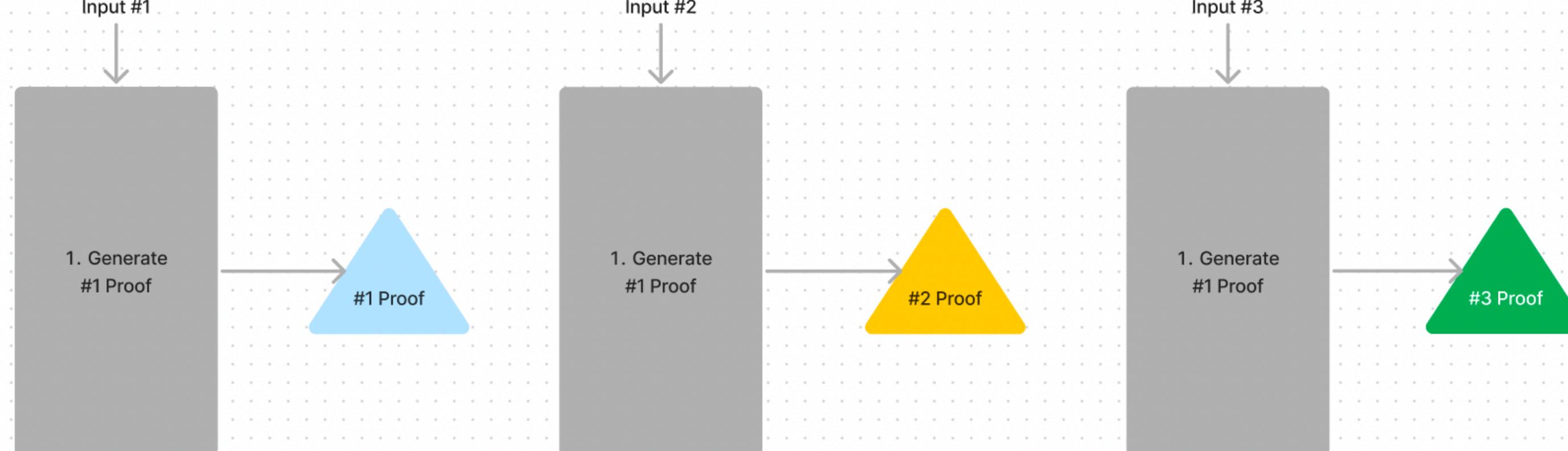
- 목표 : 검증 비용 절감하기
- 방법 : 증명 개수 줄이기
- 컨셉
 - 1000개의 트랜잭션을 검증하기 위해서는 1000개의 증명이 필요함
 - 이더리움에서 1000개의 증명을 검증한다 = 비용 폭탄
 - 1000개의 증명을 1개로 압축하자

재귀적 영지식 증명 컨셉 이해하기

- 재귀 함수: 특정 함수가 자기 자신을 부르는 것을 의미
- 영지식 증명에서 재귀란
 - 영지식 증명 생성 코드안에 이전에 생성된 증명의 검증 코드를 포함하는 것

재귀적 영지식 증명 컨셉 이해하기

방법 예시



- #3 Proof의 검증 결과가 참이라면,
- #2 Proof, #1 Proof의 검증 결과가 모두 참이다.

Polygon Zero의 재귀적 영지식 증명 알고리즘

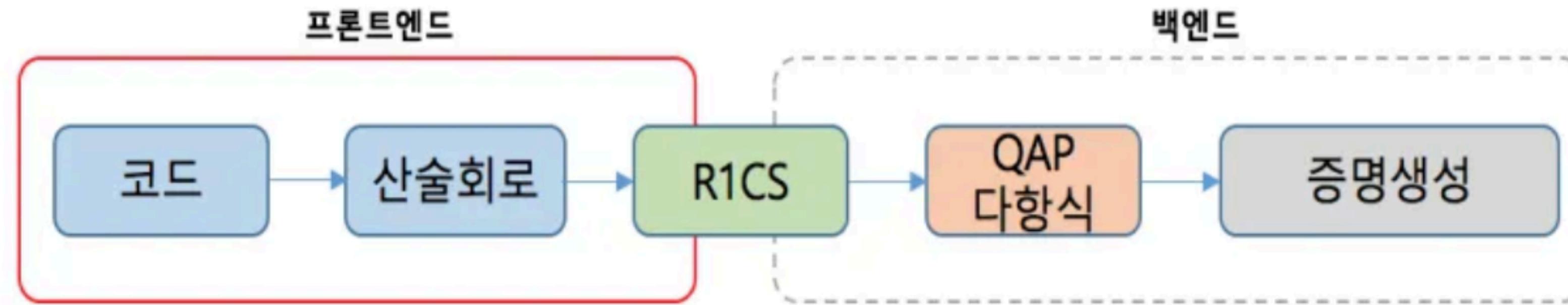
- Mir Protocol : 영지식 증명을 기반으로 블록 공간을 줄여 더 많은 노드가 참여할 수 있는 가벼운 블록체인 만들기
 - Mir Protocol의 각 노드들은 트랜잭션과 블록을 zkP로 검증
 - 각 노드들의 검증 비용을 줄이기 위해 재귀적 증명 알고리즘을 연구
 - Plonky : 기존 블록체인 시스템보다 validator의 현재 상태를 약 1000배 줄임
 - Plonky2 : Plonky보다 더 개선된 재귀적 증명 알고리즘
 - 증명 생성 시간: 170ms

Polygon Zero의 재귀적 영지식 증명 알고리즘

Plonky vs Plonky2

- Plonky = Plonk + Kate pairing + Halo
- Plonky2 = Plonk + FRI

영지식 증명 알고리즘 이해하기



- zkP는 프론트엔드와 백엔드 조합을 통해 다양한 알고리즘을 구성할 수 있음
- 프론트엔드: 함수에 대해 산술 회로식을 만드는 과정
 - Sonic, Marlin, Plonk
- 백엔드: 산술 회로에 대해 증명을 생성하는 과정
 - Kate pairing, DARK, FRI, IPA, Bulletproof

영지식 증명 알고리즘 이해하기

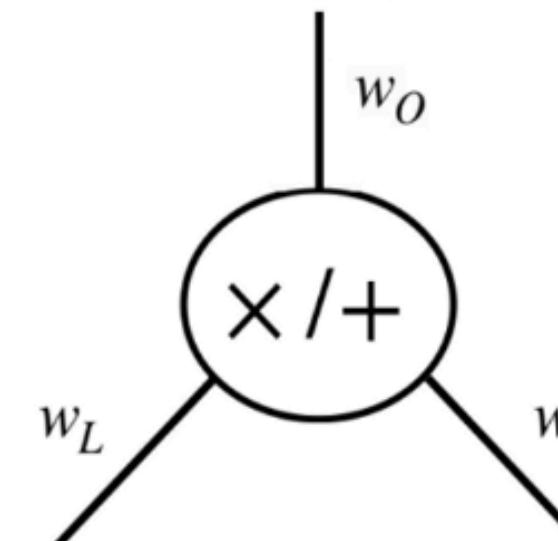
프론트엔드 : 다항식으로 어떻게 항등식을 만들까?

- 다항식을 증명하기 위해서는 이를 항등식으로 표현해야함

QAP와 Plonk 비교

곱셈 게이트와 덧셈 게이트의 표현

- 와이어 값 : w_L, w_R, w_O
- 게이트 상수 : q_M, q_L, q_R, q_O, q_C



$$q_M \cdot w_L \cdot w_R + q_L \cdot w_L + q_R \cdot w_R + q_O \cdot w_O + q_C = 0$$

Lagrange-base 형태로의 변환

와이어 값 다항식

- $w_L(X) = \sum_{i=1}^n w_{L,i} L_i(X)$
- $w_R(X) = \sum_{i=1}^n w_{R,i} L_i(X)$
- $w_O(X) = \sum_{i=1}^n w_{O,i} L_i(X)$

회로 다항식

- $q_M(X) = \sum_{i=1}^n q_{M,i} L_i(X)$
- $q_L(X) = \sum_{i=1}^n q_{L,i} L_i(X)$
- $q_R(X) = \sum_{i=1}^n q_{R,i} L_i(X)$
- $q_O(X) = \sum_{i=1}^n q_{O,i} L_i(X)$
- $q_C(X) = \sum_{i=1}^n q_{C,i} L_i(X)$

$$\begin{aligned} & q_M(X)w_L(X)w_R(X) + q_L(X)w_L(X) + q_R(X)w_R(X) \\ & + q_O(X)w_O(X) + q_C(X) = 0 \mod Z_H(X) \end{aligned}$$

〈자료〉 Gabizon, Ariel, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.

영지식 증명 알고리즘 이해하기

백엔드 : 다항식을 증명값으로 어떻게 변환할까?

- 다항식을 검증하기 위해서는 짧은 값의 증명으로 만들어 검증자에게 전송하고 검증자는 이를 정해진 알고리즘으로 검증한다.

PC Schemes	KZG10	IPA	FRI	DARKS
Low level tech	Pairing group	Discrete log group	Hash function	Unknown order group
Setup	G1, G2 groups g1, g2 generators e pairing function s_k secret value in F	G elliptic curve gⁿ independent elements in G	H hash function w unity root	N unknown order g random in N q large integer
Commitment	$(a_0s^0 + \dots + a_ns^n)g_1$	$a_0g_0 + \dots + a_ng_n$	$H(f(w^0), \dots, f(w^n))$	$(a_0q^0 + \dots + a_dq^d)g$

	FRI	KZG	IPA	DARK
Proof size	$O(\log^2(d))$	$O(1)$	$O(\log(d))$	$O(\log(d))$
Verify time	$O(\log^2(d))$	$O(1)$	$O(\log(d))$	$O(d)$
Prove time	$O(d * \log^2(d))$	$O(d)$	$O(d)$	$O(d)$

영지식 증명 알고리즘 이해하기

Plonky = Plonk + Kate pairing + Halo

- Plonk : 다항식으로 어떻게 항등식을 만들까?
- Kate Pairing : 다항식을 증명으로 어떻게 만들까?
- Halo : 왜 Plonky2에서는 사용되지 않을까?

영지식 증명 알고리즘 이해하기

다항식을 항등식으로 만들기

1. $f(x)$ 가 F 상의 원소를 근으로 갖는 다항식이다.

2. $f(x)$ 가 $Z_H(x)$ 로 나누어 떨어진다.

3. $f(x) = t(x) * Z_H(x) \quad x \in F$

▶ $Z_H(x)$ 는 vanishing함수로 H 상의 원소를 근으로 갖는 다항식이다. 즉, H 상의 원소를 $Z_H(x)$ 에 대입하면 항상 0이다.

4. $f(x) = t(x) * Z_H(x) \quad x \in H \subset F$

F 의 subgroup H 상의 모든 x 원소에 대해 위 방정식을 만족한다면, F 상의 모든 원소에 대해서도 이 방정식을 만족한다.

5. $f(z) = t(z) * Z_H(z) \quad z \leftarrow H$

최종 statement로 schwarz-sipple lemma에 의거하여 H 상의 한 점에서 위의 방정식이 만족하면 H 상의 모든 원소에 대하여 위 방정식을 만족함을 증명할 수 있다.

영지식 증명 알고리즘 이해하기

- 검증해야하는 것 : F 상의 모든 x 를 $f(x)$ 에 대입했을 때 정말 0이니?
- $f(x)$ 가 F 상의 원소를 근으로 갖는 다항식이다.
 - 증명자가 아는 것: $f(x), F$
 - 검증자가 아는 것: F
 - 검증을 위해 필요한 것: $f(x), F$
 - 검증자가 해야하는 것: 증명자가 $f(x)$ 를 보내주면, F 상의 원소를 다 대입해본다.
 - 주의: 증명자는 $f(x)$ 를 검증자에게 알려주기 쉽다.

영지식 증명 알고리즘 이해하기

1단계 : $f(x)$ 가 F 상의 원소를 근으로 갖는 다항식이다.

- $f(x)$: 다항식
 - 예: $f(x) = x^2 - 4$
- F : 체 (Field)
 - $x \in F$
 - 예: $F = \{2, -2\}$, $2 \in F$
- $f(x)$ 가 F 상의 원소를 근으로 갖는 다항식이다.
 - 예: $f(2) = 0, f(-2) = 0$

영지식 증명 알고리즘 이해하기

2단계: $f(x)$ 가 $Z_H(x)$ 로 나누어 떨어진다.

- $H : F$ 의 부분집합 (subgroup)
 - 예: $H = \{2\} \subset F = \{2, -2\}$
- $Z_H(x)$: vanishing function => H 상의 원소를 근으로 갖는 다항식
 - 예: $Z_H(x) = x - 2 \Rightarrow Z_H(2) = 0$
- 다항식끼리 나누어 떨어진다.
 - $f(x) = x^2 - 4, Z_H(x) = x - 2 \Rightarrow f(x) = (x-2)(x+2)$
 - $f(x)$ 는 $Z_H(x)$ 로 나누어 떨어지고, 몫은 $(x+2)$ 이다.

영지식 증명 알고리즘 이해하기

3단계: $f(x) = t(x) * Z_H(x)$, $x \in F$

- $t(x)$: 다항식, 몫
- $f(x)$ 가 $Z_H(x)$ 로 나누어 떨어진다.
 - $f(x) = (x-2)(x+2)$
 - $Z_H(x) = x-2$
 - $t(x) = x+2$
 - $f(x) = t(x) * Z_H(x)$ 이 성립하고, 여기서 모든 x 는 F 의 원소이다
 - $F = \{-2, 2\}$

영지식 증명 알고리즘 이해하기

4단계: $f(x) = t(x) * Z_H(x)$, $x \in H \subset F$

- F 상의 모든 원소에 대해서 위 항등식이 만족한다면,
- F 의 부분군인 H 상의 모든 원소 x 에 대해서도 위 항등식이 만족한다.

영지식 증명 알고리즘 이해하기

5단계: $f(z) = t(z) * Z_H(z)$, $z \leftarrow H$

- $z \leftarrow H$: z 는 H 에서 뽑은 원소이다.
- [Schwarz-sipple lemma](#)에 의거하여 H 상의 한 점에서 위 항등식이 만족하면
- H 상의 모든 원소에 대해 위 항등식이 만족한다.

영지식 증명 알고리즘 이해하기

Schwarz-Zipple lemma

- Schwartz-Zippel lemma는 두 다항식이 같은지 검사하는데 사용되며, 확률적으로 알고리즘의 정확성을 보장하는 정리이다.
 - $f(x)$
 - $t(x) * Z_H(x)$
- 기존 방법: 모든 원소 x 에 대해 값을 대입하여 항등식이 성립하는지 확인해야함. 비싸고 오래걸림
- Schwartz-Zippel lemma
 - 두 다항식이 서로 다를 때, 랜덤한 수를 대입했을 때 서로 같은 값이 나올 확률이 매우 작다.
 - $\text{degree}(f)/|F|$
 - $\text{degree}(f)$: $f(x)$ 의 최고차항 차수 – $f(x) = x^2 - 4$ 인 경우 $\text{degree}(f)$ 는 2이다.
 - $|F|$: F 의 원소의 개수
 - 우리는 F 가 매우 큰 경우를 가정하고 있기 때문에, 확률이 매우 작은 것을 알 수 있음
- 다항식의 차수가 매우 커서 일반적인 방법으로 검사하기 어려운 경우에 유용하게 사용됨

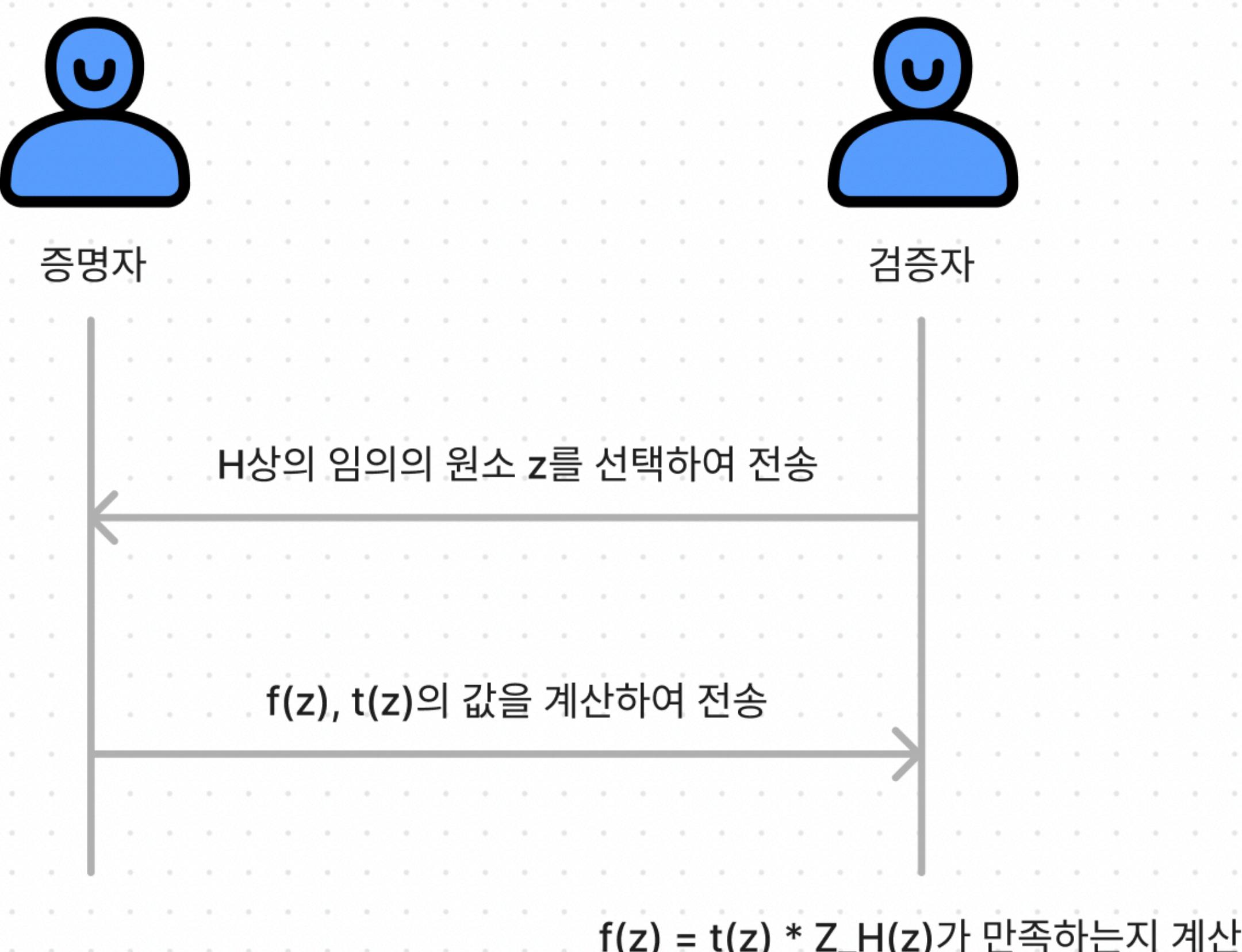
영지식 증명 알고리즘 이해하기

1단계 -> 5단계

- **f(x)**가 F상의 원소를 근으로 갖는 다항식이다.
 - $f(x)$ 가 F의 원소를 대입했을 때 항상 0이 나오는지 확인하기 위해서는
 - 무수히 많은 F의 모든 원소 x 를 $f(x)$ 에 대입하면 된다.
 - 모든 x 에 대해 항상 0이 나오면 위 조건에 대해서 $f(x)$ 는 참이다.
- **$f(z) = t(z) * Z_H(z)$, $z \leftarrow H$**
 - $f(x)$ 가 F의 원소를 대입했을 때 항상 0이 나오는지 확인하기 위해서는
 - F의 부분집합 H에서 원소 z를 선택한 후 이를 위 항등식에 대입한다
 - 만약 항등식이 만족하면, $f(x)$ 는 참이다

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기



- 검증해야하는 것 : z 를 $f(x)$ 에 대입했을 때 정말 0이니?
- 검증자가 아는 것: $Z_{\mathbb{H}}(x), \mathbb{H}, F$
- 증명자가 아는 것: $f(x), Z_{\mathbb{H}}(x), \mathbb{H}, F$
- 검증해야하는 것: 항등식이 만족하는가?
$$f(z) = t(z) * Z_{\mathbb{H}}(z), z \leftarrow \mathbb{H}$$

Plonky 이해하기

H 를 알고 있는 사람은 $Z_H(x)$ 를 만들 수 있다

- $H = \{1, 2, 3, 4, 5\}$
- $Z_H(x) = (x-1)(x-2)(x-3)(x-4)(x-5)$

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- 사실 $t(z)$ 의 값은 매우매우매우 크다
- Maller's optimization을 사용해 전송 데이터의 크기를 줄이자
- $r(x) = f(x) - t(x) * Z_H(x)$

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- $\text{com}(t)$ 는 숫자이다.
- 그 숫자도 커서 $t(x)$ 다항식을 3개로 나눈다

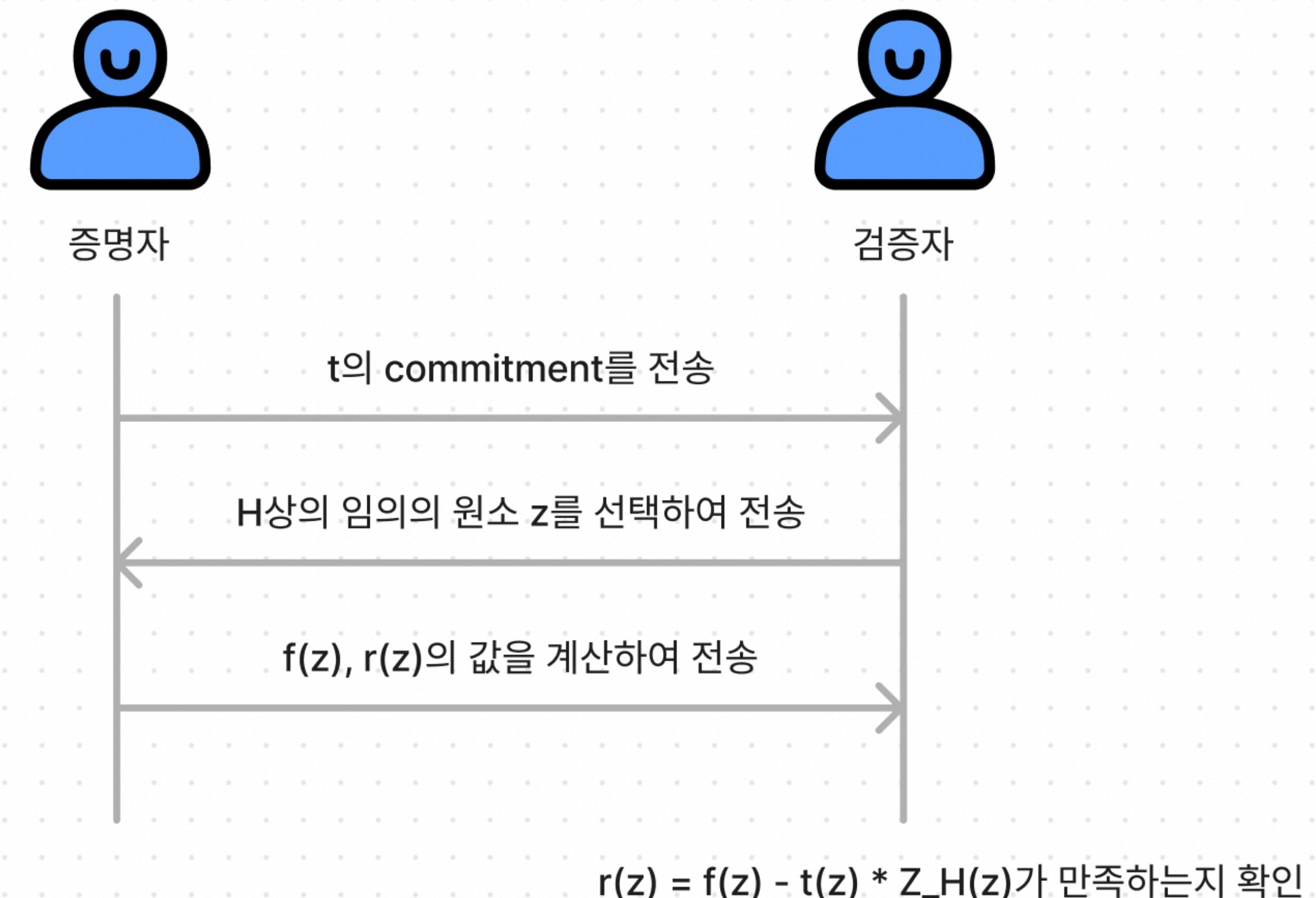
$$t \rightarrow t_{lo}, t_{mid}, t_{hi}$$

$$t(x) = t_{lo}(x) + x^n * t_{mid}(x) + x^{2n} * t_{hi}(x)$$

- 이로인해, $r(x)$ 와 $\text{com}(r)$ 도 변경된다

$$r(x) = f(x) - [t_{lo}(x) + x^n * t_{mid}(x) + x^{2n} * t_{hi}(x)] \cdot Z_H(x)$$

$$\text{com}(r) = \text{com}(f) - [\text{com}(t_{lo}) + z^n * \text{com}(t_{mid}) + z^{2n} * \text{com}(t_{hi})] \cdot Z_H(z)$$



Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- Plonk의 핵심 : $f(x)$ 와 항등식 $r(x) = \dots$ 가 어떻게 만들어지는가
 - Plonky-style circuit 만드는 방법, Polynomial interpolation, root of unity, copy constraints...
- Kate pairing의 핵심 : commitment을 만들기, 페어링 연산으로 검증하기

$$f(x) = a(x)q_L(x) + b(x)q_R(x) + a(x)b(x)q_M(x) + c(x)q_o(x) + q_c(x)$$

$$\text{com}(f) = a(z)\text{com}(q_L) + b(z)\text{com}(q_R) + a(z)b(z)\text{com}(q_M) + c(z)\text{com}(q_O) + \text{com}(q_C)$$

$$\begin{aligned} \text{com}(r) = & a(z)\text{com}(q_L) + b(z)\text{com}(q_R) + a(z)b(z)\text{com}(q_M) + c(z)\text{com}(q_O) + \text{com}(q_C) \\ & - [\text{com}(t_{lo}) + z^n \cdot \text{com}(t_{mid}) + z^{2n} \cdot \text{com}(t_{hi})] \cdot Z_H(z) \end{aligned}$$

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- 증명자와 검증자가 모두 아는 것: $q(x)$ 들
- 증명자가 숨겨야하는 것: $a(x), b(x), c(x)$
- 증명자가 검증자에게 전송하는 것 : $\text{com}(a), \text{com}(b), \text{com}(c), a(z), b(z), c(z)$
- 검증자가 확인해야하는 것 : $\text{com}(r)$ 이 아래 항등식을 만족하는가

$$\begin{aligned} \text{com}(r) = & a(z)\text{com}(q_L) + b(z)\text{com}(q_R) + a(z)b(z)\text{com}(q_M) + c(z)\text{com}(q_O) + \text{com}(q_C) \\ & - [\text{com}(t_{lo}) + z^n \cdot \text{com}(t_{mid}) + z^{2n} \cdot \text{com}(t_{hi})] \cdot Z_H(z) \end{aligned}$$

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- 실제 검증을 위해 증명자가 검증자에게 전송하는 값은 타원곡선상의 점이다.

$$\begin{aligned} [q_M] &= [q_M(s)] = 5(1, 2) + 16(68, 74) + 13(65, 98) + 1(18, 49) = 12(1, 2) = (12, 69) \\ [q_L] &= [q_L(s)] = 13(1, 2) + 1(68, 74) + 4(65, 98) + 16(18, 49) = 6(1, 2) = (32, 42) \\ [q_R] &= [q_R(s)] = 13(1, 2) + 1(68, 74) + 4(65, 98) + 16(18, 49) = 6(1, 2) = (32, 42) \\ [q_0] &= [q_0(s)] = 16(1, 2) = 16(1, 2) = (1, 99) \\ [q_c] &= [q_c(s)] = 0(1, 2) = 0(1, 2) = \text{inf} \\ [s_{\alpha_1}] &= [s_{\alpha_1}(s)] = 7(1, 2) + 13(68, 74) + 10(65, 98) + 6(18, 49) = 2(1, 2) = (68, 74) \\ [s_{\alpha_2}] &= [s_{\alpha_2}(s)] = 4(1, 2) + 0(68, 74) + 15(65, 98) + 1(18, 49) = 13(1, 2) = (65, 3) \\ [s_{\alpha_3}] &= [s_{\alpha_3}(s)] = 6(1, 2) + 7(68, 74) + 3(65, 98) + 14(18, 49) = 8(1, 2) = (18, 49) \end{aligned}$$

$$\begin{aligned} [a] &= (91, 66) \\ [b] &= (26, 45) \\ [c] &= (91, 35) \\ [z] &= (32, 59) \\ [t_{10}] &= (12, 32) \\ [t_{mid}] &= (26, 45) \\ [t_{hi}] &= (91, 66) \\ [W_3] &= (91, 35) \\ [W_{300}] &= (65, 98) \end{aligned}$$

Plonky 이해하기

Plonk + Kate pairing으로 증명 검증하기

- 검증을 위한 페어링 연산

$$e([W_{\mathfrak{z}}]_1 + u \cdot [W_{\mathfrak{z}\omega}]_1, [x]_2) \stackrel{?}{=} e(\mathfrak{z} \cdot [W_{\mathfrak{z}}]_1 + u\mathfrak{z}\omega \cdot [W_{\mathfrak{z}\omega}]_1 + [F]_1 - [E]_1, [1]_2)$$

Plonky 이해하기

Halo : 왜 Plonky2에서는 사용되지 않는가?

- Recursion zkp의 증명 생성 코드(circuit)에는 검증 코드가 포함되어 있다.
- Kate pairing : 검증 과정에서 페어링(pairing)연산을 해야한다
 - 페어링 연산을 위한 검증 코드가 매우 크기 때문에 실질적인 구현이 어렵다
- Halo : 검증을 위한 페어링 연산을 circuit 밖에서 수행한다.
 - 모든 증명의 검증 연산(페어링)은 밖에서 계산된 후 누적되고,
 - circuit에는 이전 단계의 누적 계산과 그 결과값이 유효하다는 것만 검증한다.
 - 즉, 실제 circuit에는 검증 페어링 연산이 포함되지 않는다.

Plonky 이해하기

Before Halo vs After Halo

- Circuit 안에 포함되어야 했던 검증 페어링 연산
- Halo로 인해 circuit 안에 있는 검증 연산 (ECC 곱 연산)

$$e(g^{f(x)}/g^{f(z)}, H) = e(g^{q(x)}, H^x/H_z)$$

```
R(A', B'; (A, B), (P, Q) {  
    α ← Hash(A, B, P, Q)
```

$A' \leftarrow A * P^a$

$B' \leftarrow B * Q^a \}$

Plonky2 이해하기

Halo : 왜 Plonky2에서는 사용되지 않는가?

- Plonky2 : Plonk + FRI
 - FRI(백엔드) : 검증에 pairing 연산을 사용하지 않음
 - Halo를 사용할 필요가 없음
 - FRI의 검증을 위한 항등식

$$f(r\omega^2) - f(r\omega) - f(r) = \frac{Z_H(r)}{(r - \omega^{n-2})(r - \omega^{n-1})} * g(r)$$

$$f(r) - B(r) = D(r) * g'(r)$$

앞으로 더 이야기 하고 싶은 것

- Plonk 다향식 만드는 방법
- Kate로 commitment 계산하는 방법
- 검증을 위한 페어링 연산
- Halo의 계산 누적 연산
- FRI