

# Polynomial Commitment Strategies in Hash-Based Proof Systems for Small Statements

Thibauld Feneuil

*Workshop « On the Mathematics of PQC »*

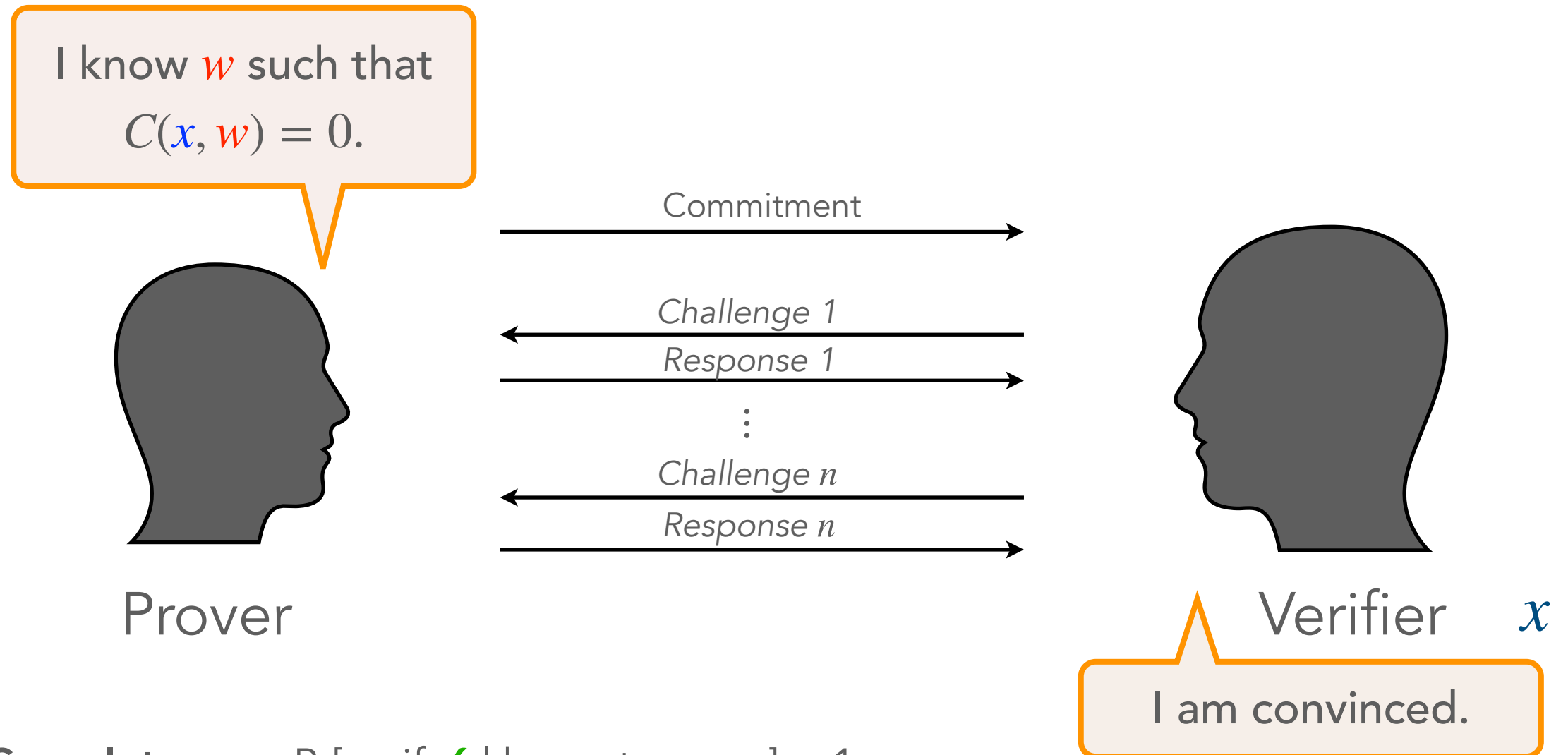
June 5, 2025 — Zürich (Switzerland)

# Table of Contents

- Context / Motivation
- Hash-based polynomial commitments
  - Using GGM trees (a.k.a. seed trees)
  - Using Merkle trees (a.k.a. hash trees)
- Applications
- Conclusion

# Context / Motivation

# (Zero-Knowledge) Proofs of Knowledge

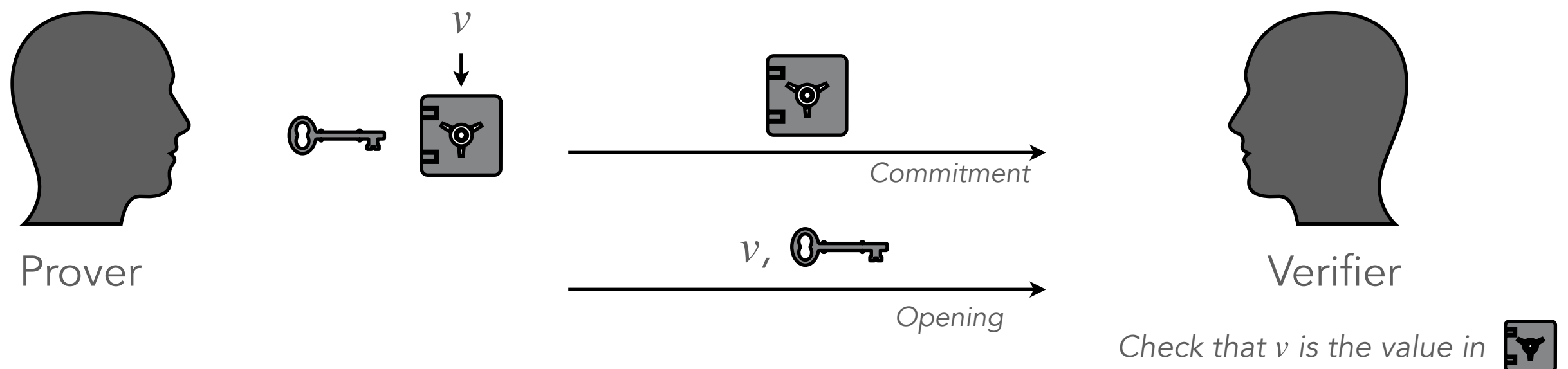


- **Completeness:**  $\Pr[\text{verif } \checkmark \mid \text{honest prover}] = 1$
- **Soundness:**  $\Pr[\text{verif } \checkmark \mid \text{malicious prover}] \leq \varepsilon$  (e.g.  $2^{-128}$ )
- **Zero-knowledge (optional):** verifier learns nothing on  $w$ .
- **Succintness (optional):** verifying the proof is faster than computing  $C$ .

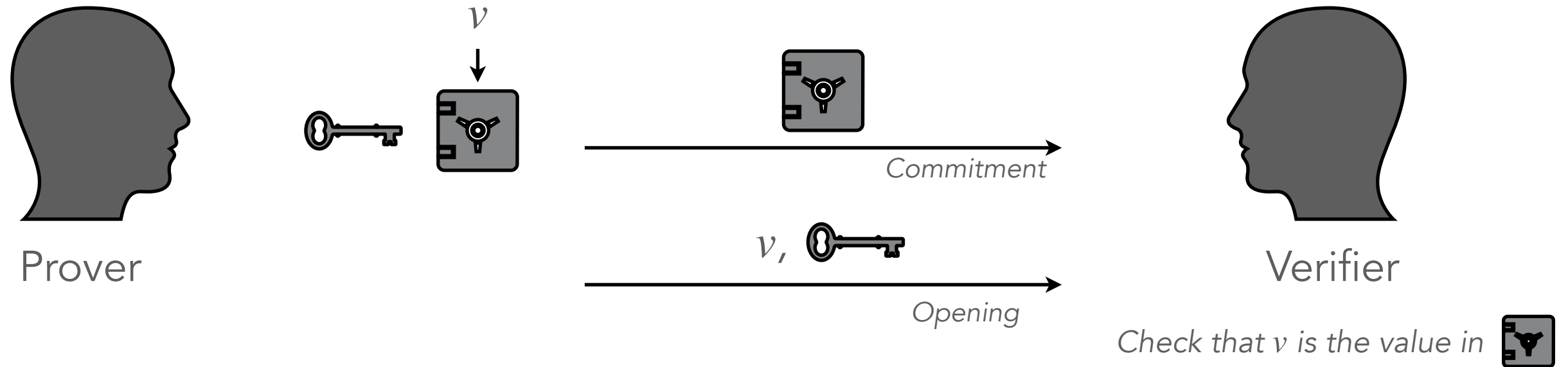


# Commitment Scheme

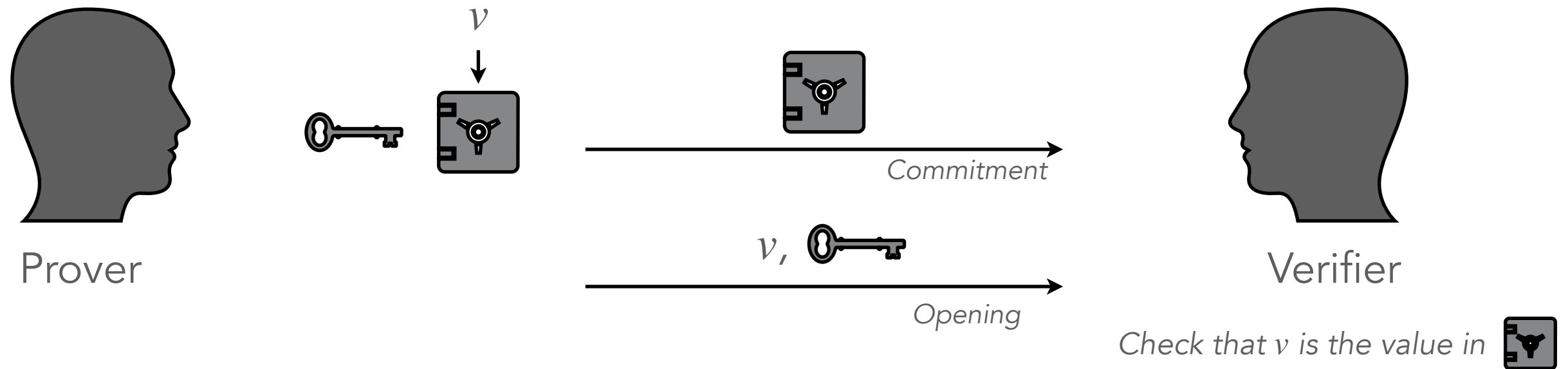
- **Commitment** algorithm/procedure. A prover can commit to a chosen value  $v$  while keeping it *hidden* to other people (*hiding property*).
- **Opening** algorithm/procedure. The prover can reveal the value  $v$  and prove that the revealed value is the one which has been committed through the commitment procedure. It should be impossible for the prover to reveal a value  $v' \neq v$  while convincing the verifier that  $v'$  is the committed value (*binding property*).



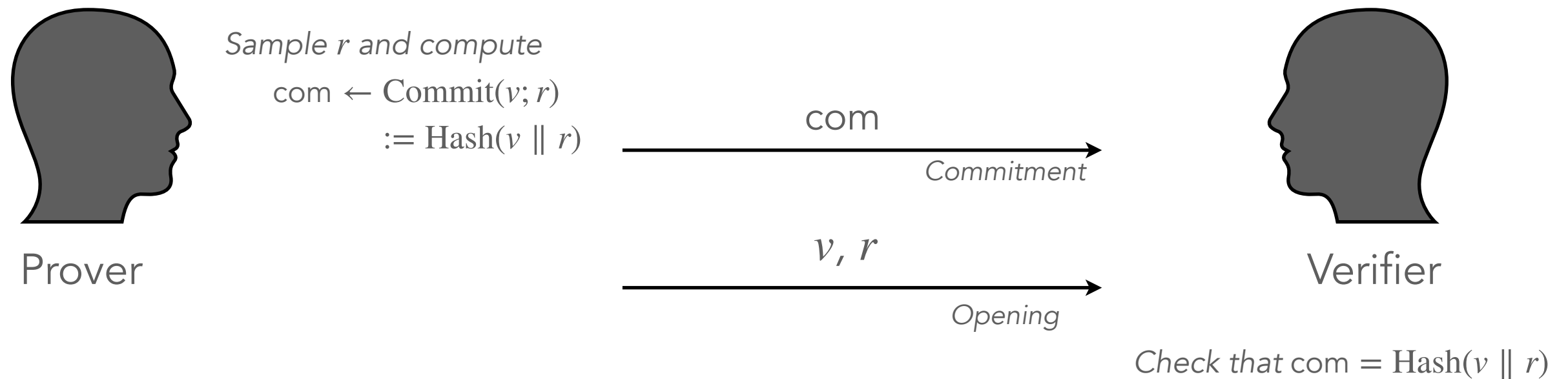
# Commitment Scheme



# Commitment Scheme



## Hash-based Commitment Scheme:



# Polynomial Commitment Scheme

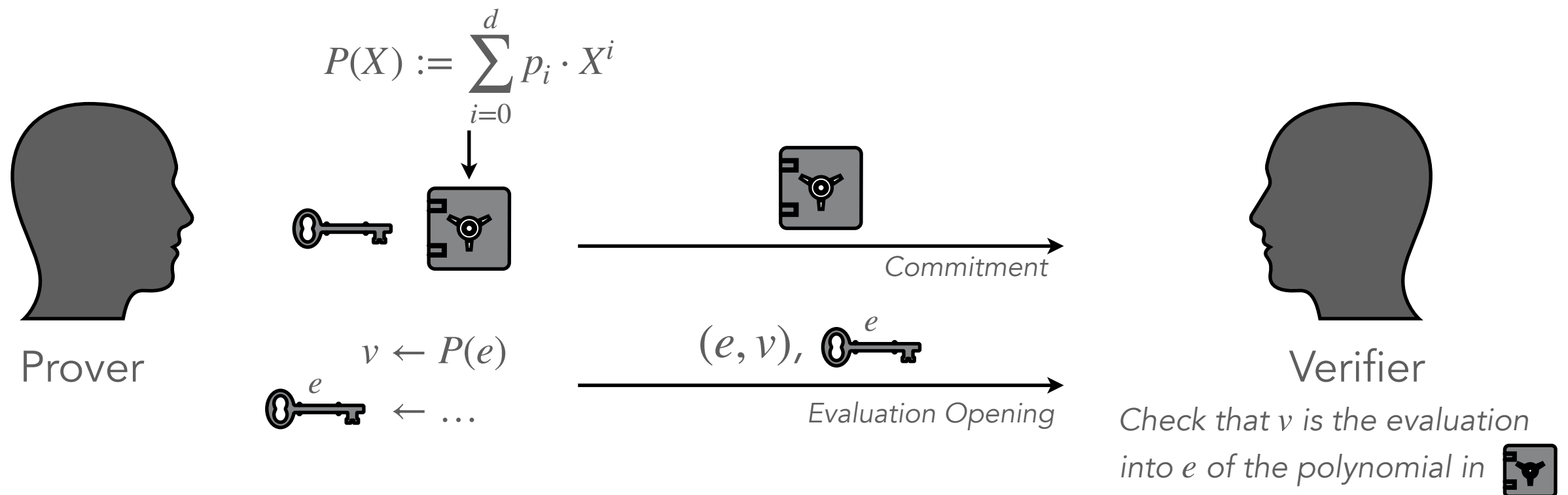
- We want to commit a **polynomial**  $P(X) := p_0 + p_1 \cdot X + \dots + p_d \cdot X^d$ :
  - Using a standard commitment scheme, the opening procedure would consist of revealing the *entire polynomial* in a verifiable way.

# Polynomial Commitment Scheme

- We want to commit a **polynomial**  $P(X) := p_0 + p_1 \cdot X + \dots + p_d \cdot X^d$ :
  - Using a standard commitment scheme, the opening procedure would consist of revealing the *entire polynomial* in a verifiable way.
  - Using a **polynomial commitment scheme**, the opening procedure would consist of **some evaluations of the committed polynomials** in a verifiable way, while keeping the other evaluations **hidden** to the verifier.

# Polynomial Commitment Scheme

- We want to commit a **polynomial**  $P(X) := p_0 + p_1 \cdot X + \dots + p_d \cdot X^d$ :
  - Using a standard commitment scheme, the opening procedure would consist of revealing the *entire polynomial* in a verifiable way.
  - Using a **polynomial commitment scheme**, the opening procedure would consist of **some evaluations of the committed polynomials** in a verifiable way, while keeping the other evaluations **hidden** to the verifier.



# Basic Proof System for Polynomial Constraints

I know  $w_1, \dots, w_n$  such that

$$\begin{cases} f_1(w_1, \dots, w_n) &= 0 \\ \vdots \\ f_m(w_1, \dots, w_n) &= 0, \end{cases}$$

where  $f_1, \dots, f_m$  are public **degree- $d$  polynomials**.

Prover

Prove it!

Verifier

# Basic Proof System for Polynomial Constraints

---

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .

Verifier

Prover



# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

Verifier

Prover

# Basic Proof System for Polynomial Constraints

① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .

② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$X \cdot Q_1(X) = f_1(P_1(X), \dots, P_n(X))$$

$$\vdots$$

$$X \cdot Q_m(X) = f_m(P_1(X), \dots, P_n(X))$$

Well-defined!

$$\forall j, f_j(P_1(0), \dots, P_n(0)) = f_j(w_1, \dots, w_n) = 0$$

Verifier

Prover

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

Verifier

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$



Prover

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

Verifier

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

Prover

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

Prover

Verifier

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .
- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$X \cdot Q_{j^*}(X) \neq f_{j^*}(P_1(X), \dots, P_n(X))$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .
- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

Malicious Prover 🐱

Verifier

## Soundness Analysis

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$r \cdot v_1^Q = f_1(v_1^P, \dots, v_n^P)$$

$\vdots$

$$r \cdot v_m^Q = f_m(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .
- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$\underline{X \cdot Q_{j^*}(X)} \neq \underline{f_{j^*}(P_1(X), \dots, P_n(X))}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations

- for all  $i$ ,  $v_i^P := P_i(r)$
- for all  $j$ ,  $v_j^Q := Q_j(r)$

*Evaluation into 0*

$= 0$

$\neq 0$

Malicious Prover 🤖

Verifier

## Soundness Analysis

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$r \cdot v_1^Q = f_1(v_1^P, \dots, v_n^P)$$

$\vdots$

$$r \cdot v_m^Q = f_m(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .

- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$X \cdot Q_{j^*}(X) \neq f_{j^*}(P_1(X), \dots, P_n(X))$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations
- for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

Malicious Prover 🐱

Verifier

## Soundness Analysis

$\xrightarrow{\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)}$

$r$

$\xleftarrow{}$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

$\xrightarrow{}$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$r \cdot v_1^Q = f_1(v_1^P, \dots, v_n^P)$$

$\vdots$

$$r \cdot v_m^Q = f_m(v_1^P, \dots, v_n^P)$$

$$r \cdot v_{j^*}^Q = f_{j^*}(v_1^P, \dots, v_n^P)$$



# Basic Proof System for Polynomial Constraints

Verifier

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .

- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$X \cdot Q_{j^*}(X) \neq f_{j^*}(P_1(X), \dots, P_n(X))$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

$\xrightarrow[\text{r}]{\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)}$

**Soundness Analysis**

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

**Schwartz-Zippel Lemma:** Let  $D$  be the **non-zero** degree- $(d \cdot \ell)$  polynomial defined as

$$D := X \cdot Q_{j^*}(X) - f_{j^*}(P_1(X), \dots, P_n(X))$$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

$$r \cdot v_{j^*}^Q = f_{j^*}(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

Verifier

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .

- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$X \cdot Q_{j^*}(X) \neq f_{j^*}(P_1(X), \dots, P_n(X))$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

$\xrightarrow{\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)}$   
 $r$

**Soundness Analysis**

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

**Schwartz-Zippel Lemma:** Let  $D$  be the **non-zero** degree- $(d \cdot \ell)$  polynomial defined as

$$D := X \cdot Q_{j^*}(X) - f_{j^*}(P_1(X), \dots, P_n(X))$$

We have

$$\Pr[\text{verification passes}] = \Pr[D(r) = 0 \mid r \leftarrow_{\$} \mathcal{C}] \leq \quad .$$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

$$r \cdot v_{j^*}^Q = f_{j^*}(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

Verifier

- ① For all  $i$ , choose a degree- $\ell$  polynomial  $P_i(X)$ . There exist  $j^*$  such that  $f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0$ .

- ② Choose some polynomials  $Q_1, \dots, Q_m$ . We know that

$$X \cdot Q_{j^*}(X) \neq f_{j^*}(P_1(X), \dots, P_n(X))$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$   
 $r$

## Soundness Analysis

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

**Schwartz-Zippel Lemma:** Let  $D$  be the **non-zero** degree- $(d \cdot \ell)$  polynomial defined as

$$D := X \cdot Q_{j^*}(X) - f_{j^*}(P_1(X), \dots, P_n(X))$$

We have

$$\Pr[\text{verification passes}] = \Pr[D(r) = 0 \mid r \leftarrow_{\$} \mathcal{C}] \leq \frac{d \cdot \ell}{|\mathcal{C}|}.$$

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

$$r \cdot v_{j^*}^Q = f_{j^*}(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that
 
$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$
- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .
- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

Prover

Verifier

## Zero-Knowledge Analysis

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$r \cdot v_1^Q = f_1(v_1^P, \dots, v_n^P)$$

$\vdots$

$$r \cdot v_m^Q = f_m(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .

② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$X \cdot Q_1(X) = f_1(P_1(X), \dots, P_n(X))$$

$$X \cdot Q_m(X) = f_m(P_1(X), \dots, P_n(X))$$

③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$

⑤ Reveal the evaluations

- for all  $i$ ,  $v_i^P = P_i(r)$
- for all  $j$ ,  $v_j^Q = Q_j(r)$

Revealing an evaluation of  $P_i(X)$   
leaks no information about  $w_i$ .

Prover

Verifier

## Zero-Knowledge Analysis

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P)$   $(v_1^Q, \dots, v_m^Q)$

④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$r \cdot v_1^Q = f_1(v_1^P, \dots, v_n^P)$$

$\vdots$

$$r \cdot v_m^Q = f_m(v_1^P, \dots, v_n^P)$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

The evaluation  $Q_j(r)$  is fully determined by  $r$  and  $(v_1^P, \dots, v_n^P)$ .

Prover

Verifier

## Zero-Knowledge Analysis

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$r$

$(v_1^P, \dots, v_n^P)$   $(v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

# Basic Proof System for Polynomial Constraints

- ① For all  $i$ , sample a random degree- $\ell$  polynomial  $P_i(X)$  such that  $P_i(0) = w_i$ .
- ② Build the polynomials  $Q_1, \dots, Q_m$  such that

$$\begin{aligned} X \cdot Q_1(X) &= f_1(P_1(X), \dots, P_n(X)) \\ &\vdots \\ X \cdot Q_m(X) &= f_m(P_1(X), \dots, P_n(X)) \end{aligned}$$

- ③ Commit the polynomials  $(P_1, \dots, P_n)$  and  $(Q_1, \dots, Q_m)$ .

- ⑤ Reveal the evaluations
  - for all  $i$ ,  $v_i^P := P_i(r)$
  - for all  $j$ ,  $v_j^Q := Q_j(r)$

**Hiding Polynomial  
Commitment Scheme**

Prover

Verifier

**Zero-Knowledge Analysis**

$\text{PCom}(P_1, \dots, P_n, Q_1, \dots, Q_m)$

$(v_1^P, \dots, v_n^P), (v_1^Q, \dots, v_m^Q)$

- ④ Choose a random evaluation point  $r \in \mathcal{C} \subset \mathbb{F}$

- ⑥ Check that  $(v_1^P, \dots, v_n^P)$  and  $(v_1^Q, \dots, v_m^Q)$  are consistent with the commitment.

Check that

$$\begin{aligned} r \cdot v_1^Q &= f_1(v_1^P, \dots, v_n^P) \\ &\vdots \\ r \cdot v_m^Q &= f_m(v_1^P, \dots, v_n^P) \end{aligned}$$

# Basic Proof System for Polynomial Constraints

I know  $w_1, \dots, w_n$  such that

$$\begin{cases} f_1(w_1, \dots, w_n) &= 0 \\ \vdots \\ f_m(w_1, \dots, w_n) &= 0, \end{cases}$$

where  $f_1, \dots, f_m$  are public **degree- $d$  polynomials**.

Prove it!

Prover

Degree of the witness polynomials  
 $P_1(X), \dots, P_n(X)$

Verifier

$$\text{Soundness Error} = \frac{d \cdot \ell}{|\mathcal{C}|}$$

Probability that a malicious prover  
can convince the verifier.

Size of the challenge space that  
contains all the possible opened  
evaluations



# Blueprint of Hash-based Proof Systems

## Prover

Build some polynomials

$$P_1(X), \dots, P_n(X)$$

satisfying some relations

$$\begin{cases} F_1(P_1(X), \dots, P_n(X)) = 0 \\ \vdots \\ F_m(P_1(X), \dots, P_n(X)) = 0, \end{cases}$$

Build an opening proof  $\pi$  for

$$v_1 := P_1(e), \dots, v_n := P_n(e).$$

## Verifier

Sample a point  $e$  from  $\mathcal{C}$ .

Check that  $\pi$  is a valid opening proof for  $e$ .

Check that

$$\begin{cases} F_1(v_1, \dots, v_n) = 0 \\ \vdots \\ F_m(v_1, \dots, v_n) = 0, \end{cases}$$

Hash-based SNARK: Ligero, Aurora, STARK, Brakedown, ...

Post-quantum signatures: FAEST, MQOM v2, SDitH v2, ...

# Blueprint of Hash-based Proof Systems

## Prover

Build some polynomials

$$P_1(X), \dots, P_n(X)$$

satisfying some relations

$$\begin{cases} F_1(P_1(X), \dots, P_n(X)) &= 0 \\ \vdots \\ F_m(P_1(X), \dots, P_n(X)) &= 0, \end{cases}$$

Build an opening proof  $\pi$  for

$$v_1 := P_1(e), \dots, v_n := P_n(e).$$

Without packing:  $P(0) = w$

With packing:

$$P(1) = w_1, \dots, P(s) = w_s$$

## Verifier

$\xrightarrow{\text{PCom}(P_1, \dots, P_n)}$

$\xleftarrow{e}$

Sample a point  $e$  from  $\mathcal{C}$ .

$\xrightarrow{(v_1, \dots, v_n), \pi}$

Check that  $\pi$  is a valid opening proof for  $e$ .

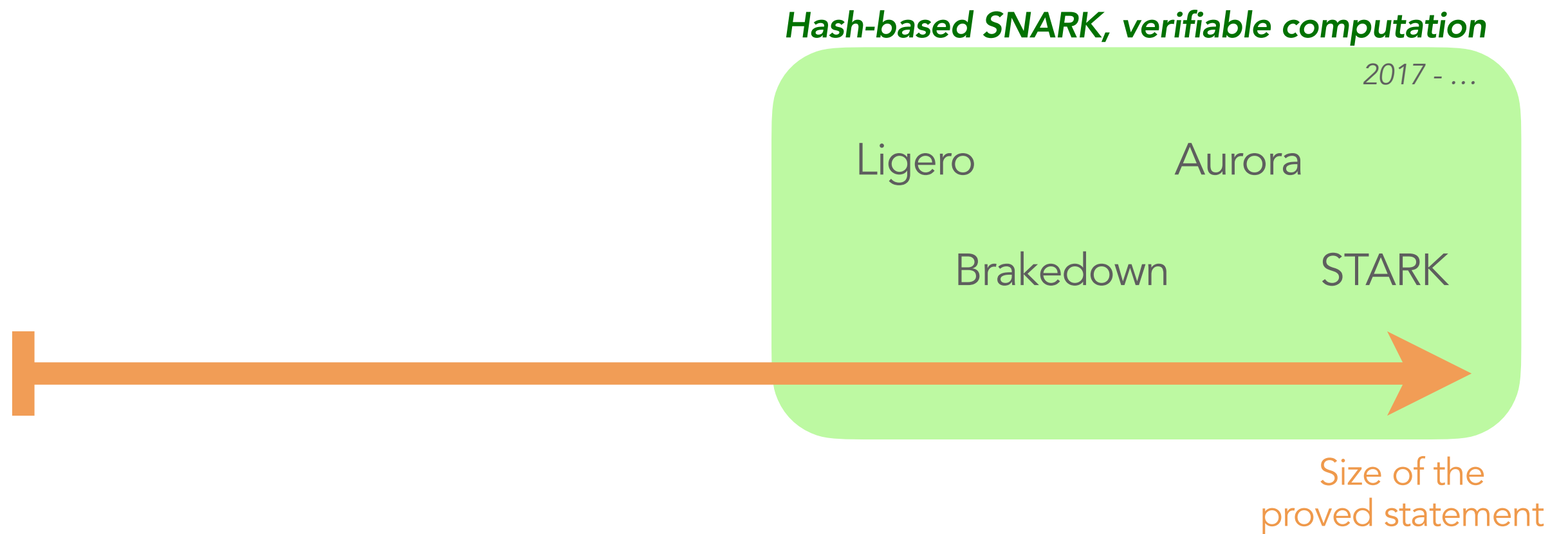
Check that

$$\begin{cases} F_1(v_1, \dots, v_n) &= 0 \\ \vdots \\ F_m(v_1, \dots, v_n) &= 0, \end{cases}$$

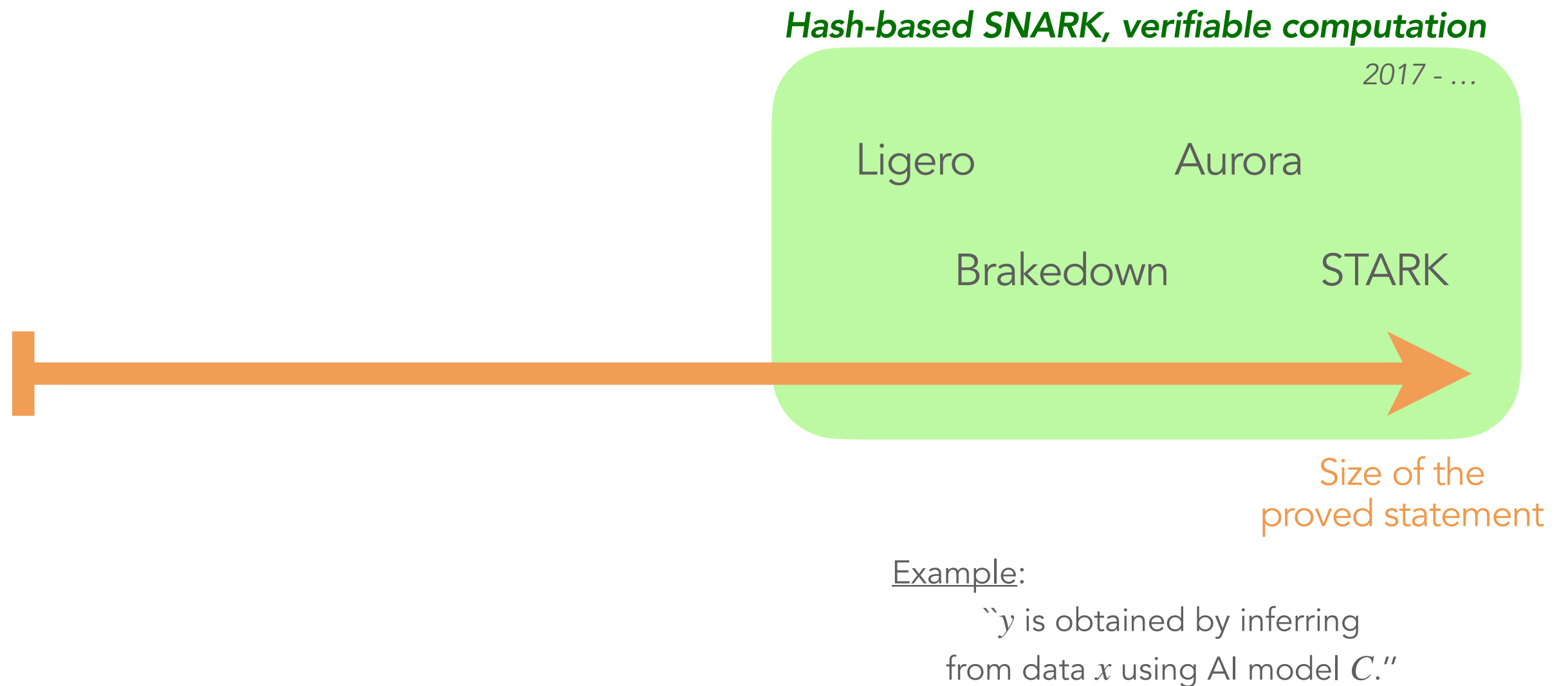
Hash-based SNARK: Ligero, Aurora, STARK, Brakedown, ...

Post-quantum signatures: FAEST, MQOM v2, SDitH v2, ...

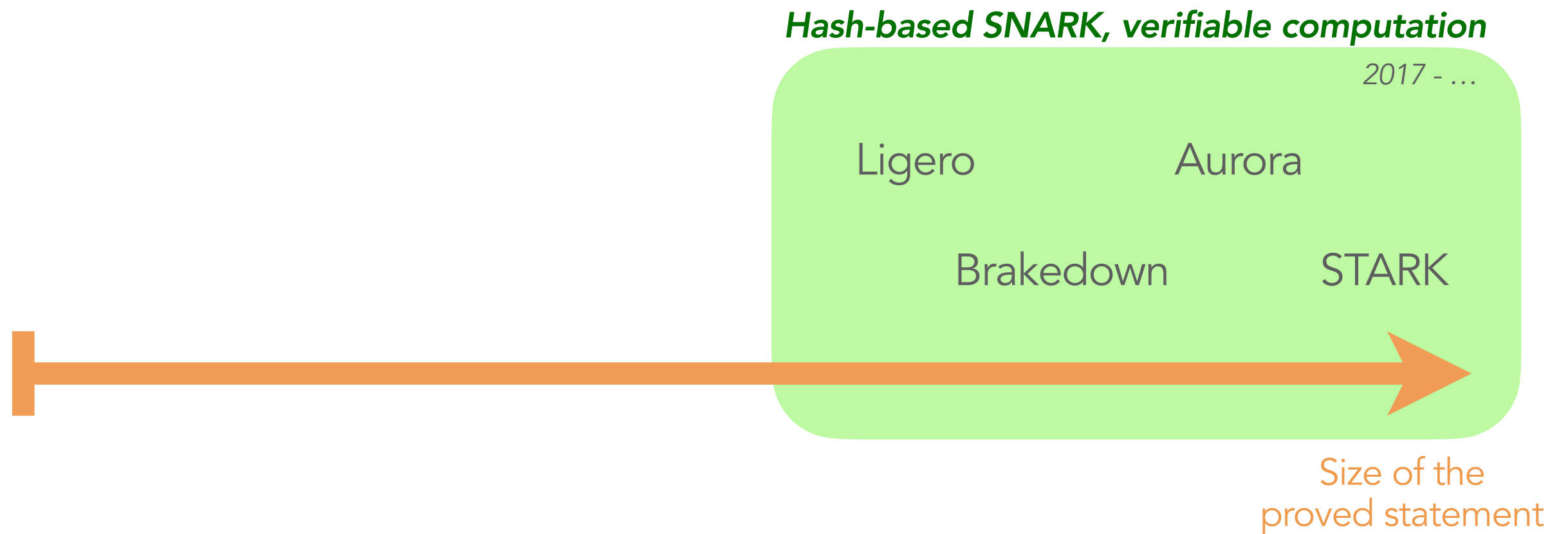
# Blueprint of Hash-based Proof Systems



# Blueprint of Hash-based Proof Systems



# Blueprint of Hash-based Proof Systems



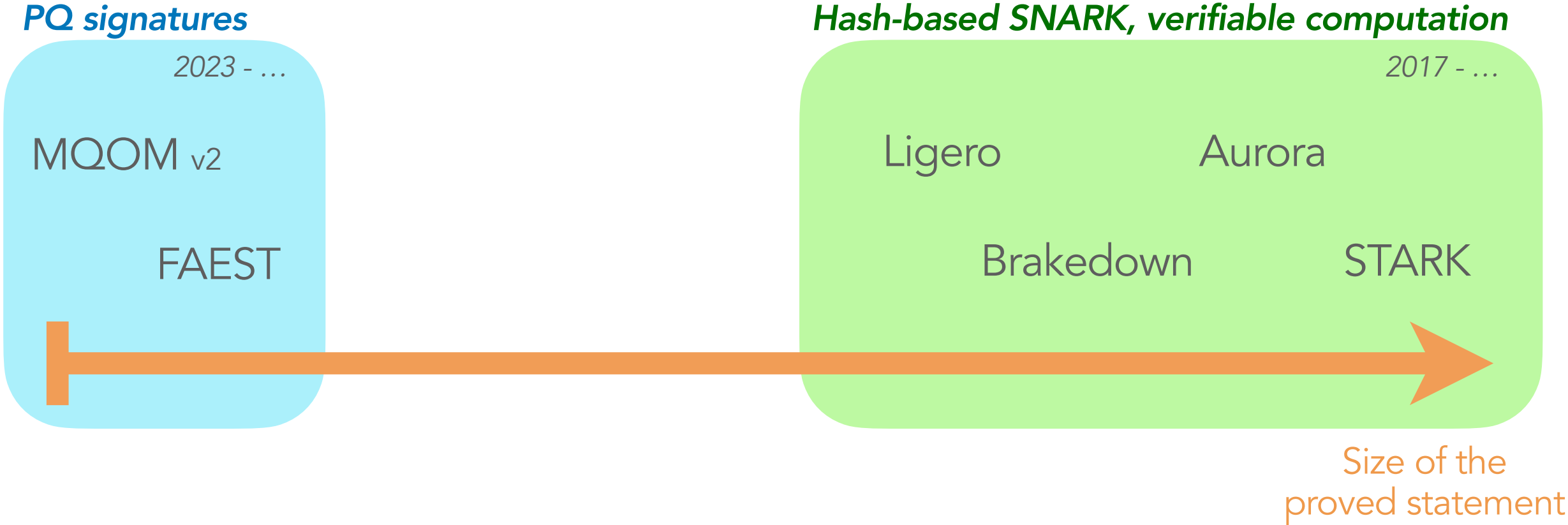
Example:

" $y$  is obtained by inferring  
from data  $x$  using AI model  $C$ ."

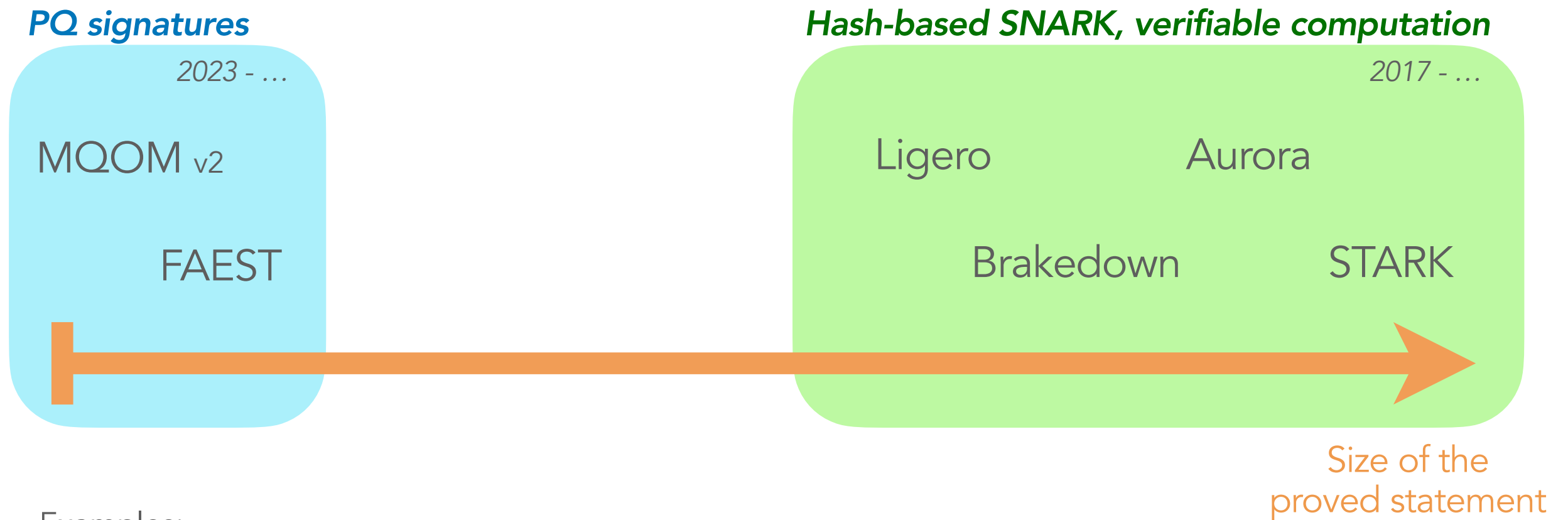
Properties:

- Succinctness: ***required***
- Zero-Knowledge: ***optional***

# Blueprint of Hash-based Proof Systems



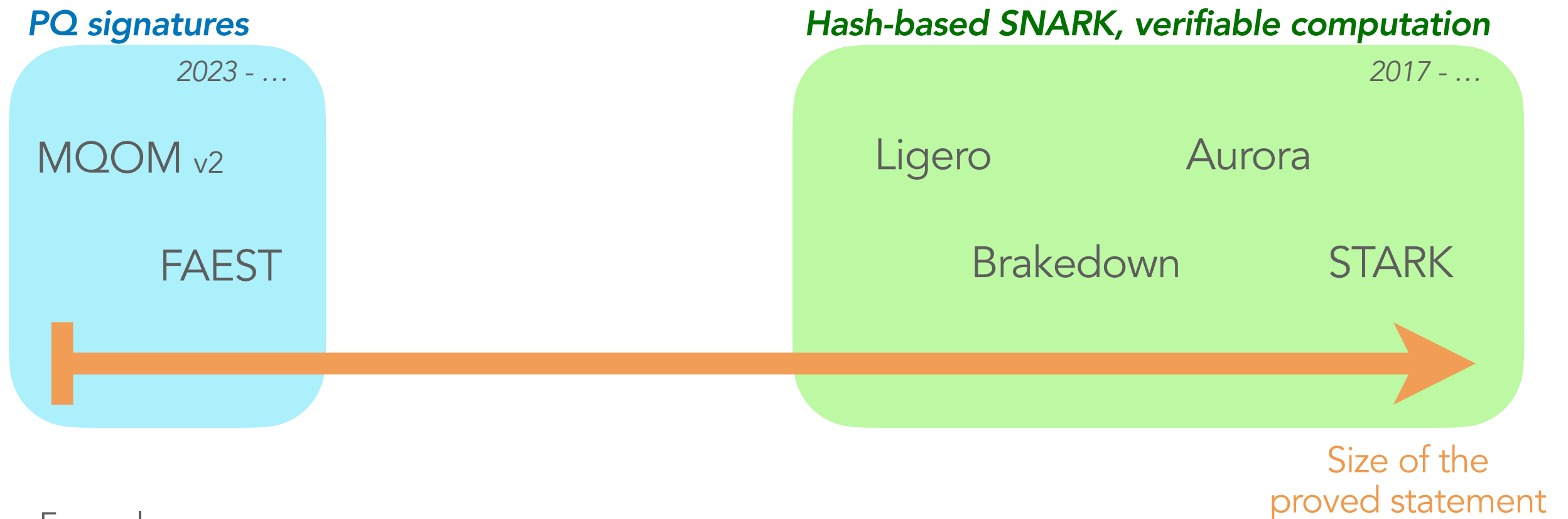
# Blueprint of Hash-based Proof Systems



## Examples:

- **FAEST**: Given  $(x, y)$ , I know  $k$  such that  $y = \text{AES}_k(x)$
- **MQOM**: Given a multivariate system  $\mathcal{F}$ , I know  $x$  such that  $\mathcal{F}(x) = 0$

# Blueprint of Hash-based Proof Systems



## Examples:

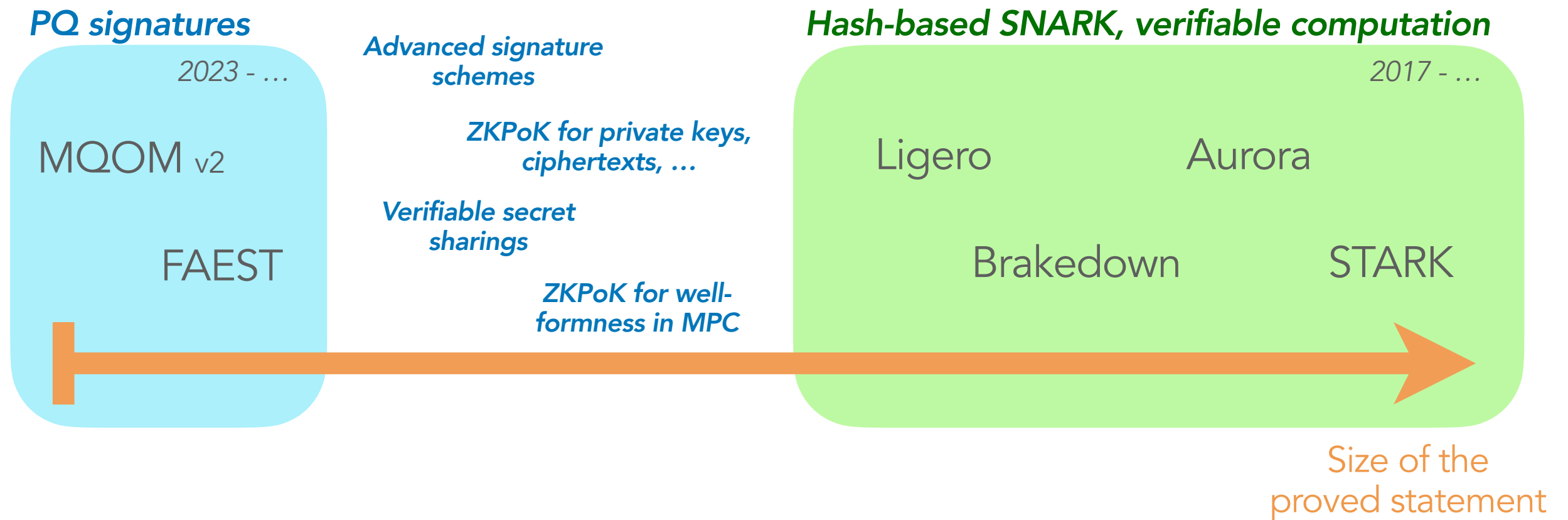
- **FAEST:** Given  $(x, y)$ , I know  $k$  such that  $y = \text{AES}_k(x)$
- **MQOM:** Given a multivariate system  $\mathcal{F}$ , I know  $x$  such that  $\mathcal{F}(x) = 0$

## Properties:

- Zero-Knowledge: **required**
- Succinctness: **optional**



# Blueprint of Hash-based Proof Systems



# Blueprint of Hash-based Proof Systems

Small statements

*PQ signatures*

2023 - ...

MQOM v2

FAEST

*Advanced signature schemes*

*ZKPoK for private keys, ciphertexts, ...*

*Verifiable secret sharings*

*ZKPoK for well-formness in MPC*

*Hash-based SNARK, verifiable computation*

2017 - ...

Ligero

Aurora

Brakedown

STARK

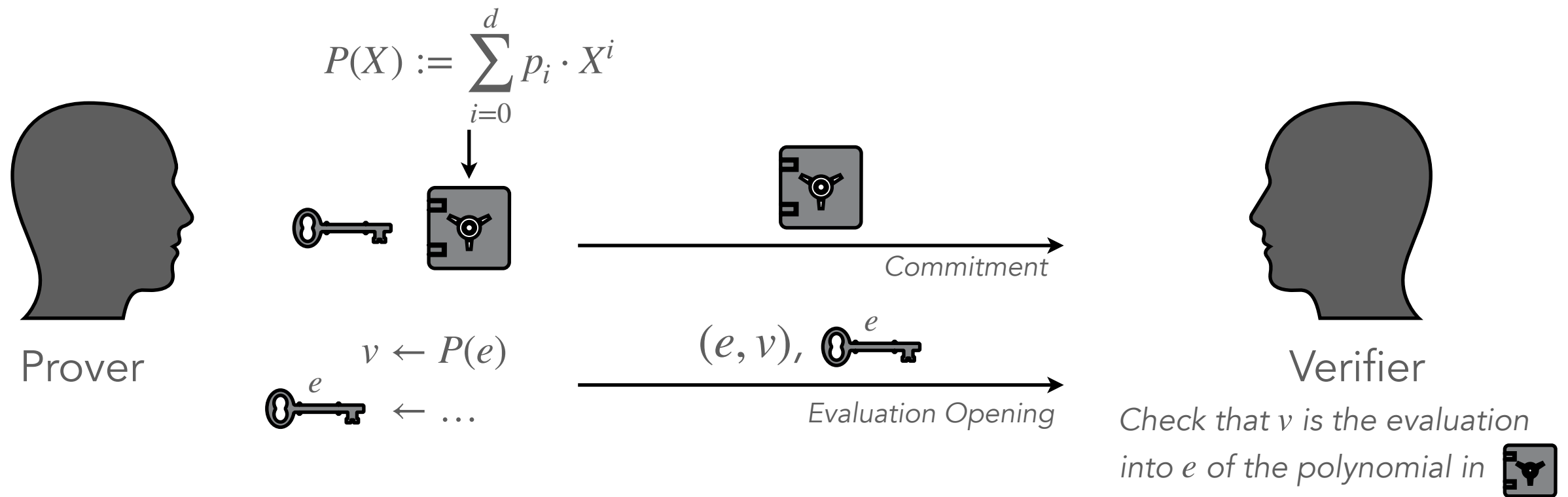
Size of the  
proved statement

Zero-Knowledge is **required**  
Succinctness is **optional**

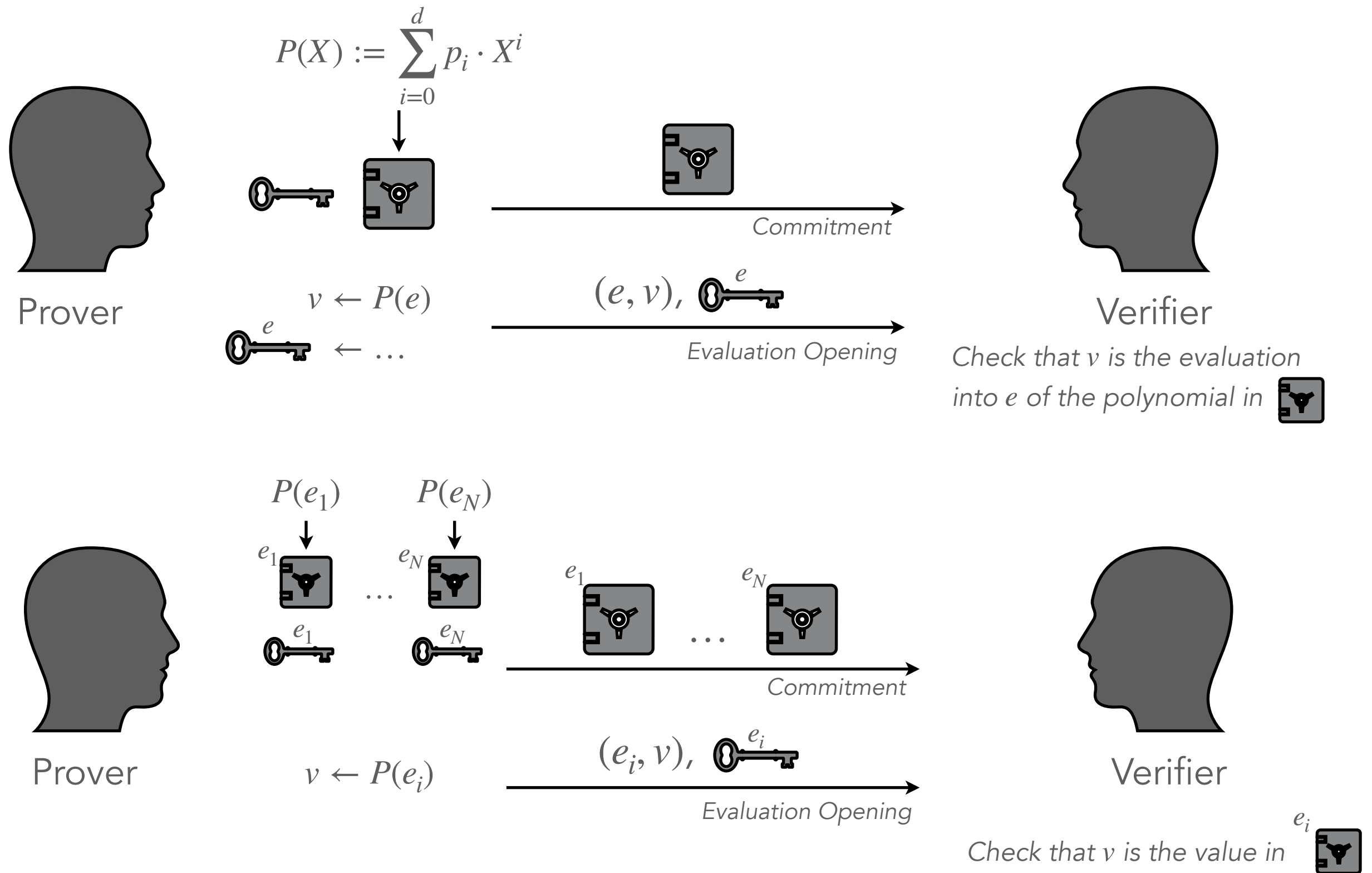
Succinctness is **required**  
Zero-Knowledge is **optional**

# Hash-based Polynomial Commitments

# Polynomial Commitment Scheme



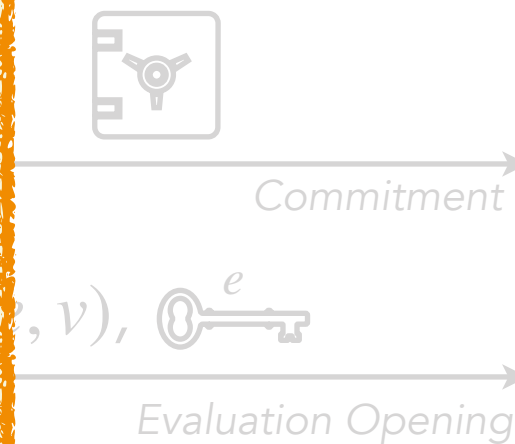
# Polynomial Commitment Scheme



# Polynomial Commitment Scheme

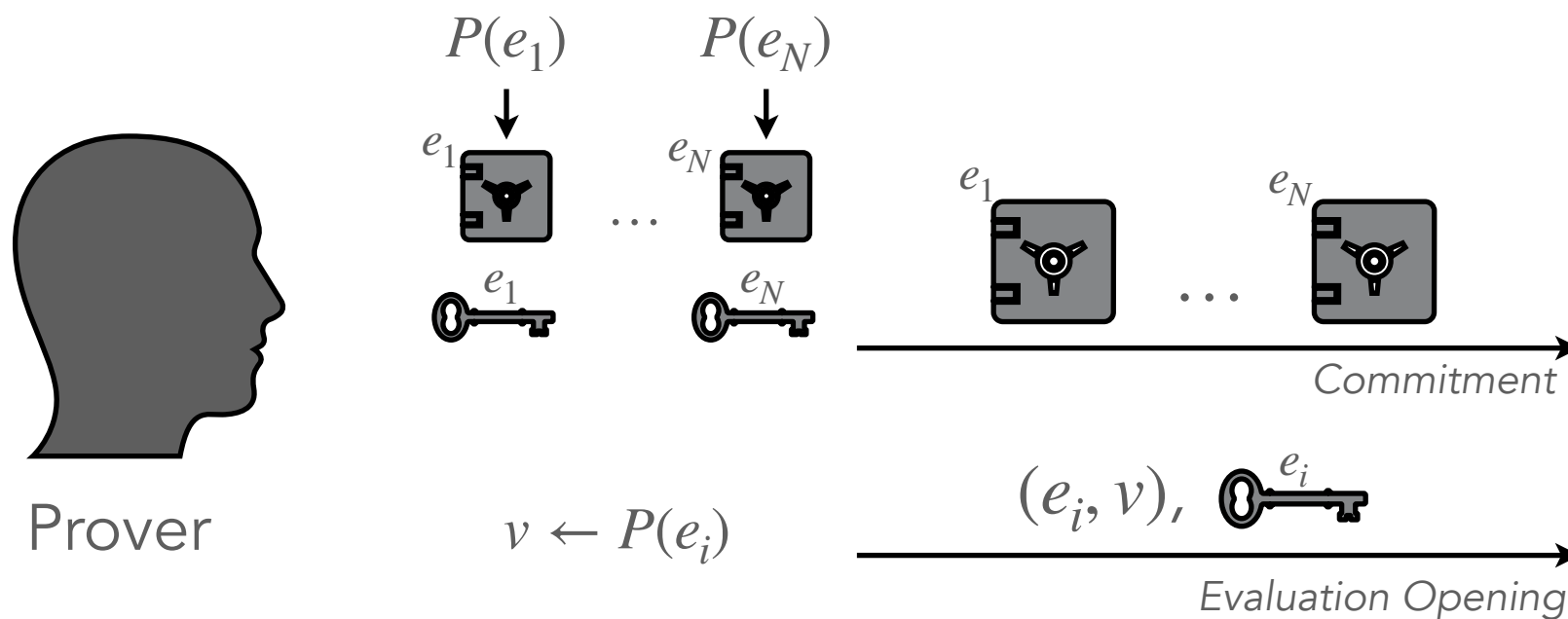
## Performance issue:

If the number  $N$  of possible evaluations is **large**, it will be impracticable.



Verifier

Check that  $v$  is the evaluation into  $e$  of the polynomial in



Check that  $v$  is the value in  $e_i$

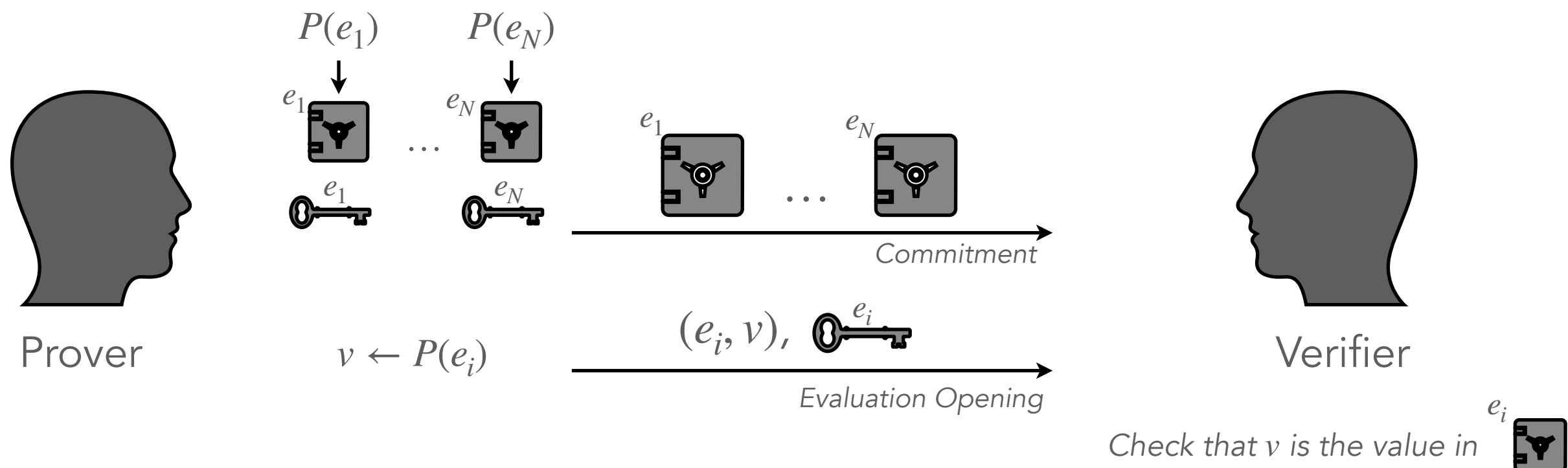
# Polynomial Commitment Scheme

## Performance issue:

If the number  $N$  of possible evaluations is **large**, it will be impracticable.

## Security issue:

The verifier has **no guarantee** that the committed evaluations form a polynomial of the right degree.



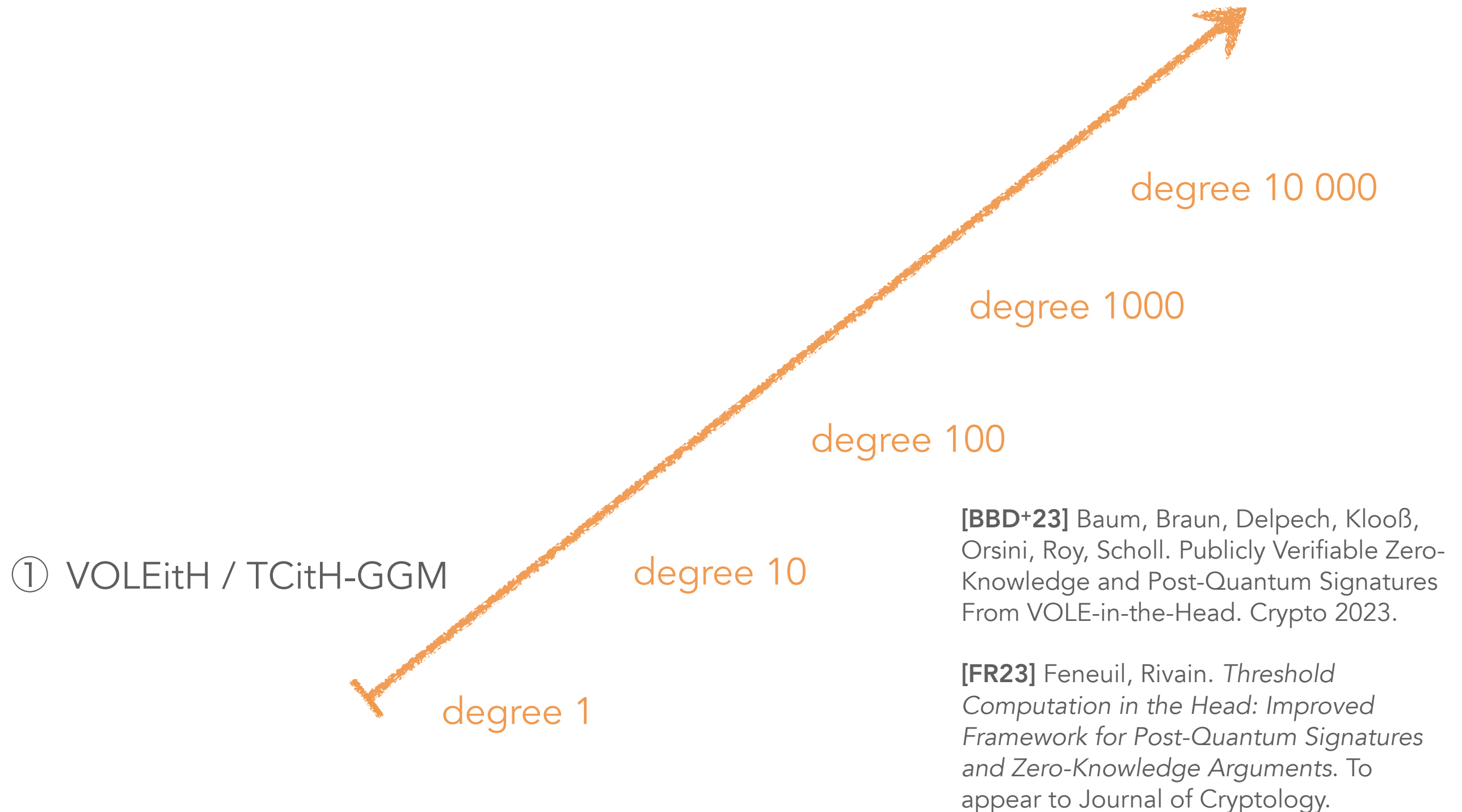
# Polynomial Commitment Scheme

- **Small-domain PCS**: the prover can open only  $N$  evaluations of the committed polynomial, where  $N \ll |\mathbb{F}|$ .
  - For example, the prover commits to  $P(X) \in \mathbb{F}_q$  with  $q = 2^{32} - 5$ , but the prover can only open the evaluations  $P(e)$  for  $e$  in  $\{0, 1, \dots, 1023\}$ .
- **Full-domain PCS**: the prover can open all the evaluations of the committed polynomial, i.e. he can open  $P(e)$  for all  $e \in \mathbb{F}$ .
  - For example, the prover commits to  $P(X) \in \mathbb{F}_q$  with  $q = 2^{32} - 5$  and the prover can open the evaluations  $P(e)$  for  $e$  in  $\mathbb{F}_q = \{0, 1, \dots, 2^{32} - 6\}$ .



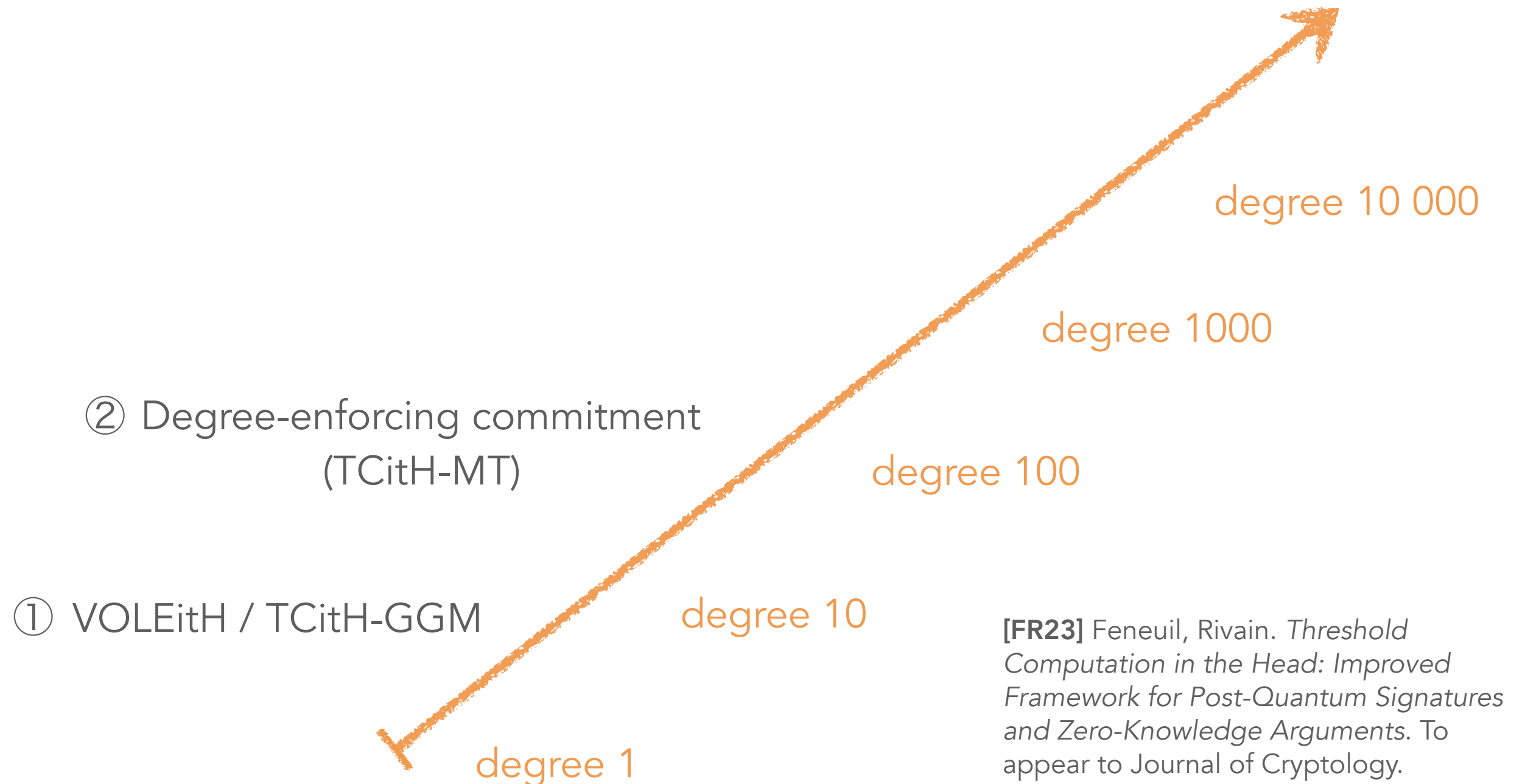
# How to commit to polynomials?

(using symmetric primitives)



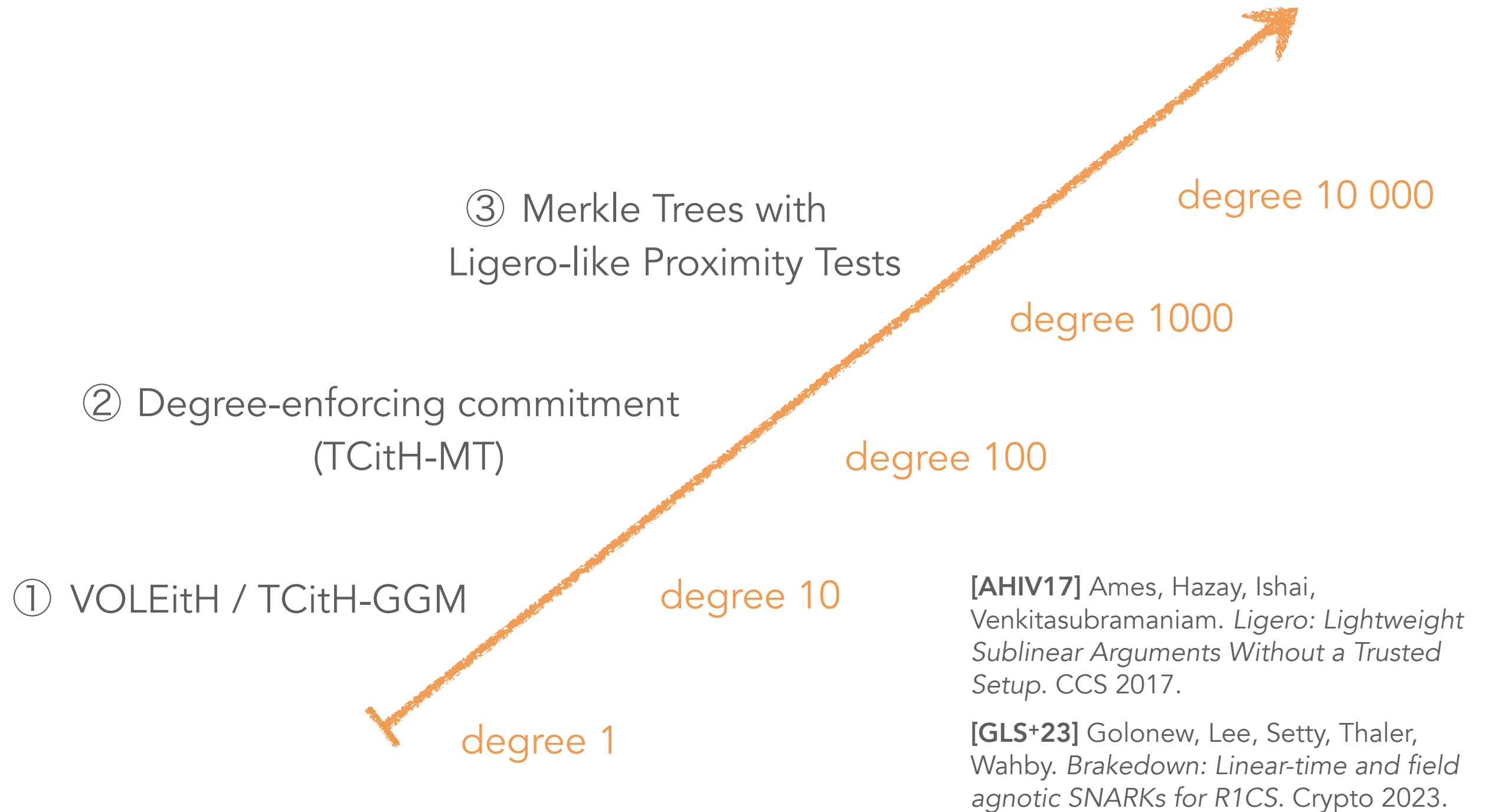
# How to commit to polynomials?

(using symmetric primitives)



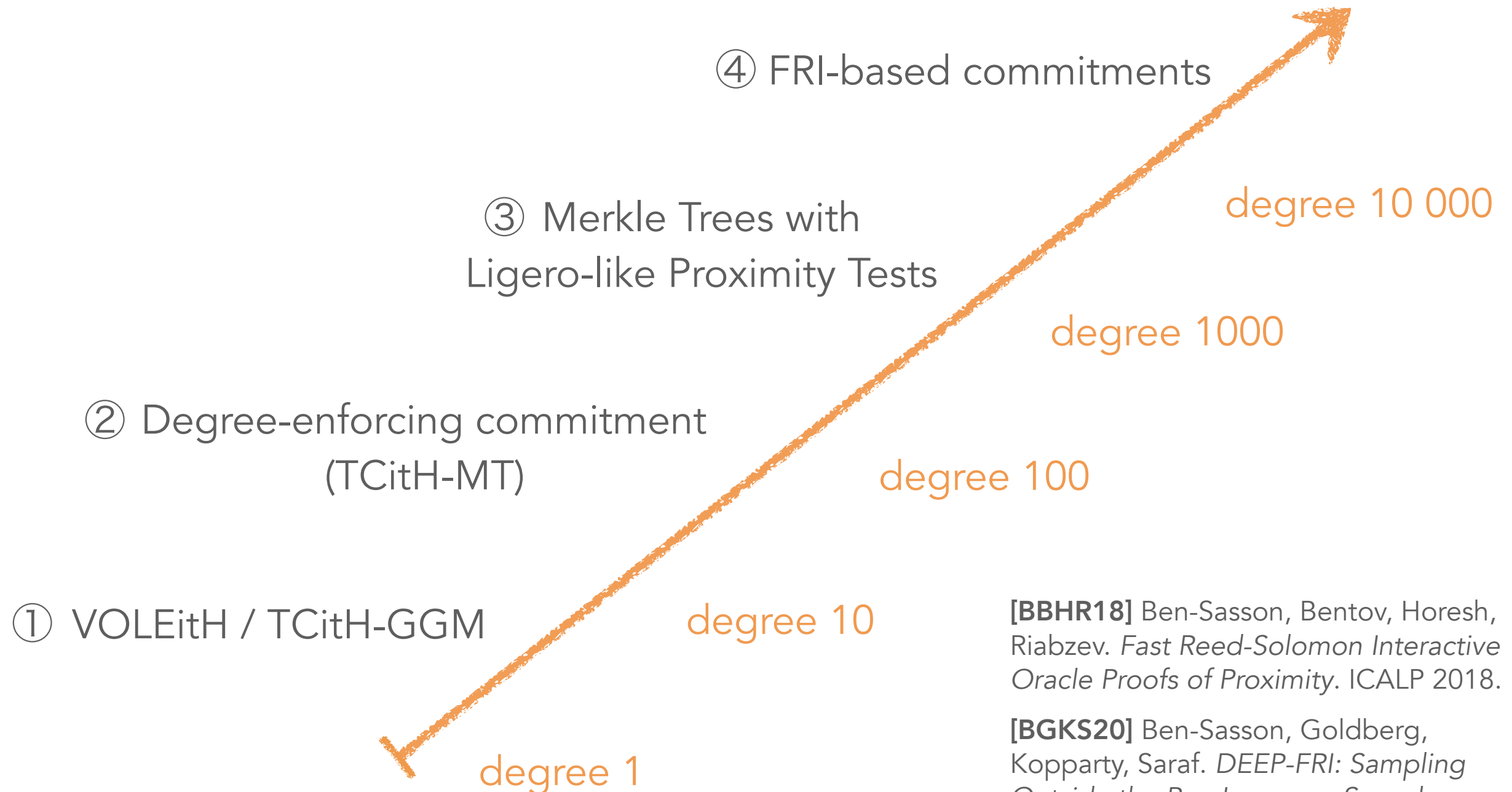
# How to commit to polynomials?

(using symmetric primitives)



# How to commit to polynomials?

(using symmetric primitives)



**[BBHR18]** Ben-Sasson, Bentov, Horesh, Riabzev. *Fast Reed-Solomon Interactive Oracle Proofs of Proximity*. ICALP 2018.

**[BGKS20]** Ben-Sasson, Goldberg, Kopparty, Saraf. *DEEP-FRI: Sampling Outside the Box Improves Soundness*. ITCS 2020.

# How to commit to polynomials?

(using symmetric primitives)

**Merkle Tree**

④ FRI-based commitments

③ Merkle Trees with  
Ligero-like Proximity Tests

② Degree-enforcing commitment  
(TCitH-MT)

① VOLEitH / TCitH-GGM

**GGM Tree**

degree 1

degree 10

degree 100

degree 1000

degree 10 000

*Natively, those techniques  
lead to **small-domain**  
polynomial commitment scheme*

# How to commit to polynomials?

(using symmetric primitives)

**Merkle Tree**

④ FRI-based commitments

③ Merkle Trees with  
Ligero-like Proximity Tests

② Degree-enforcing commitment  
(TCitH-MT)

① VOLEitH / TCitH-GGM

**GGM Tree**

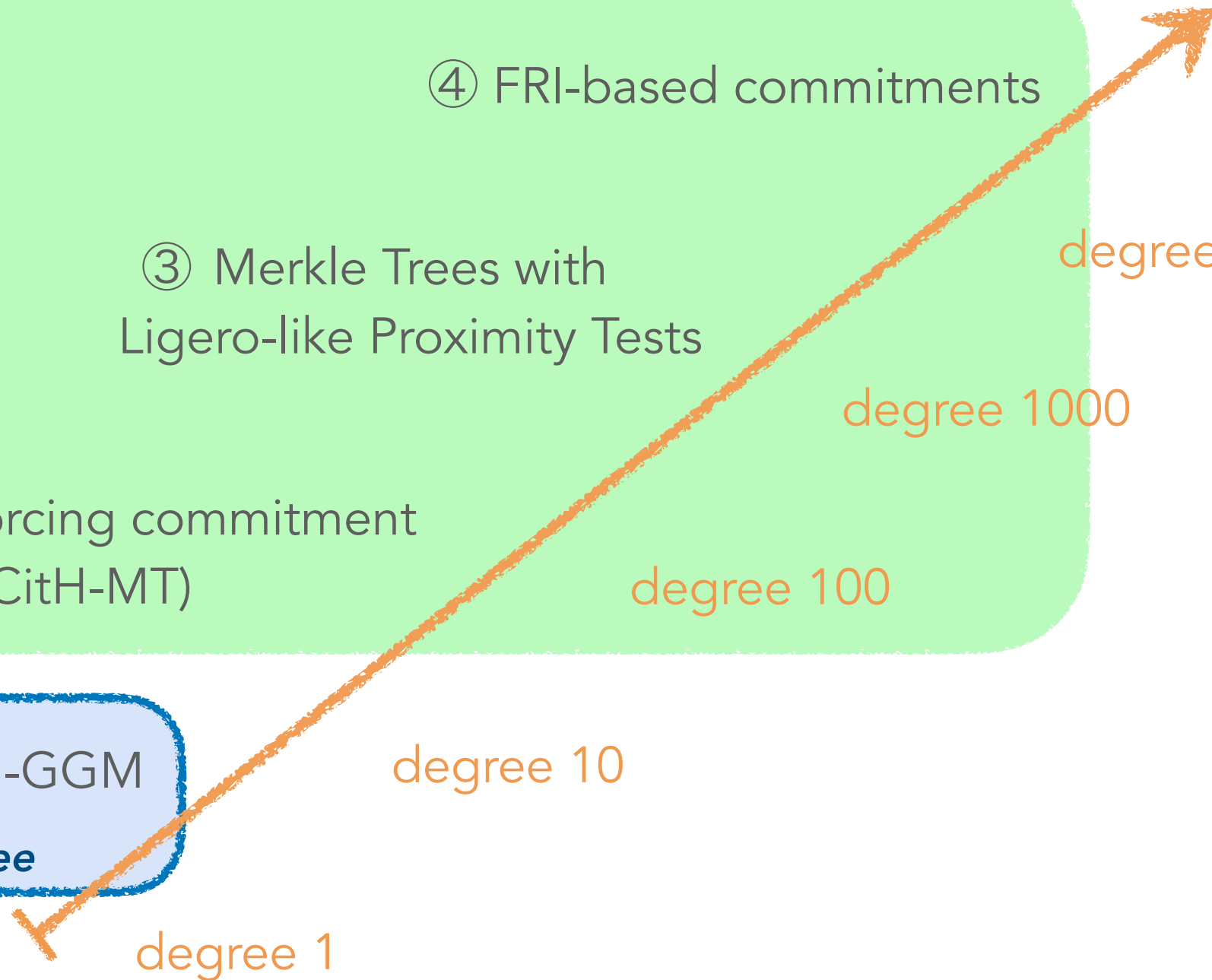
degree 1

degree 10

degree 100

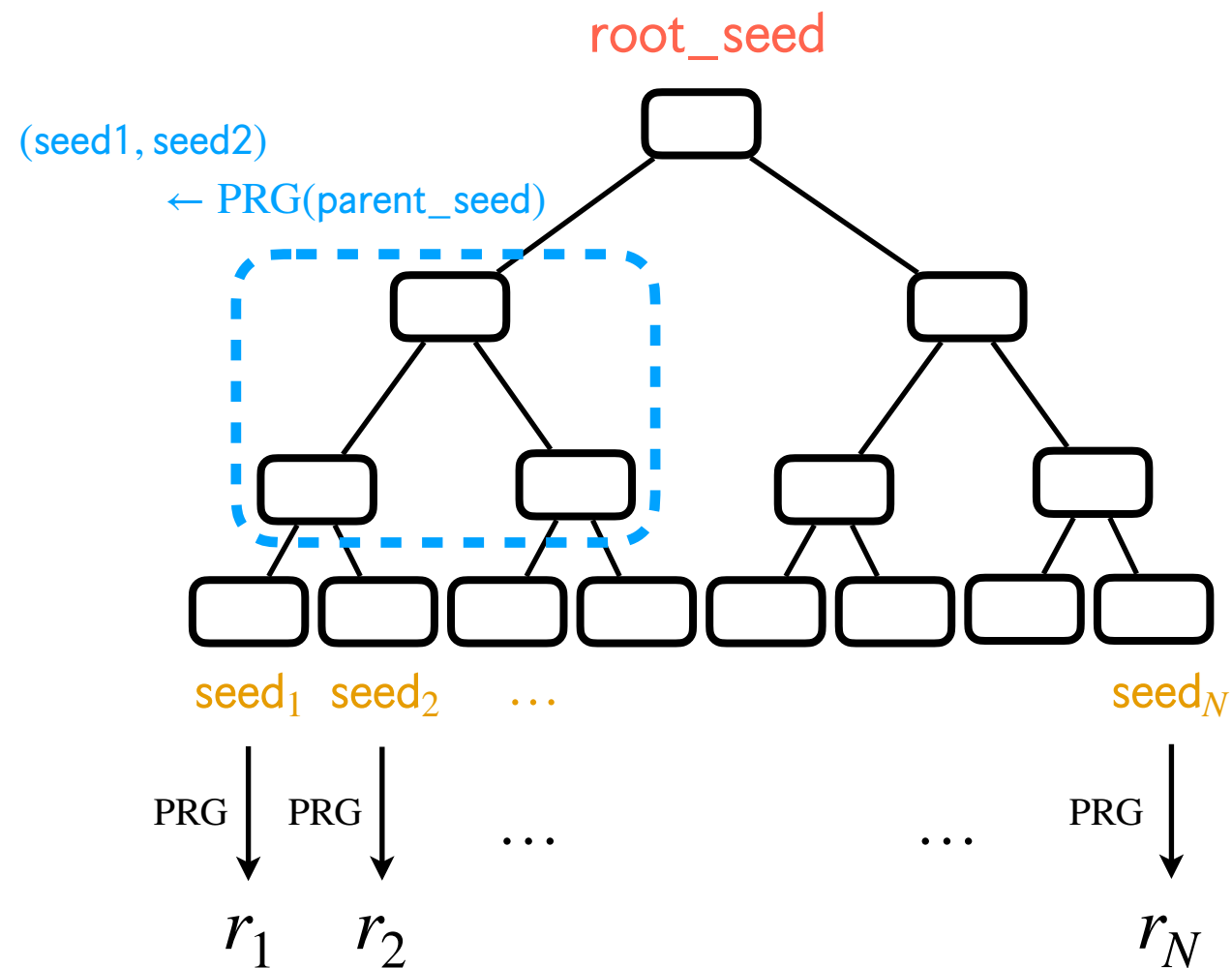
degree 1000

degree 10 000



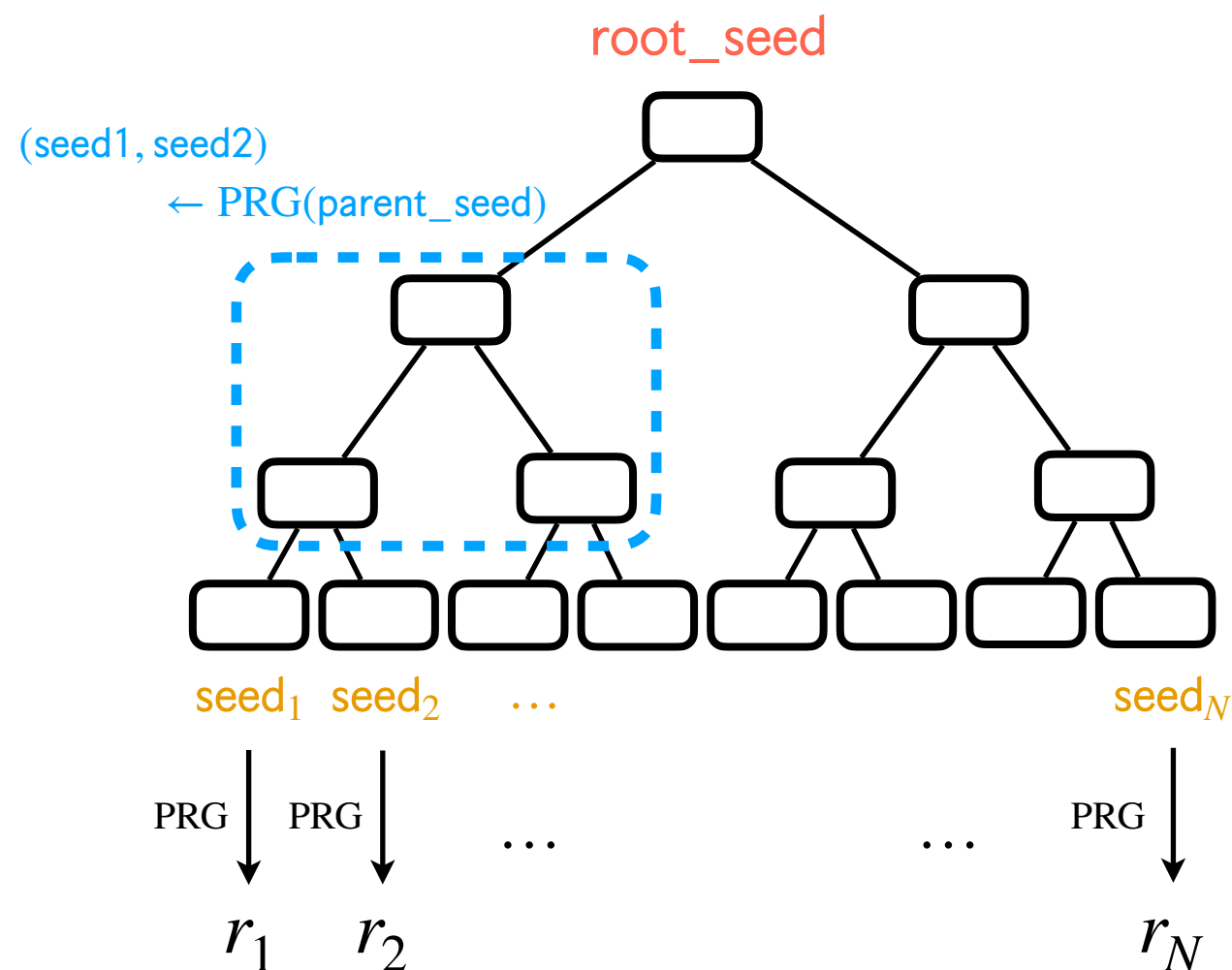
# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



Build  $\Delta P(X)$  as

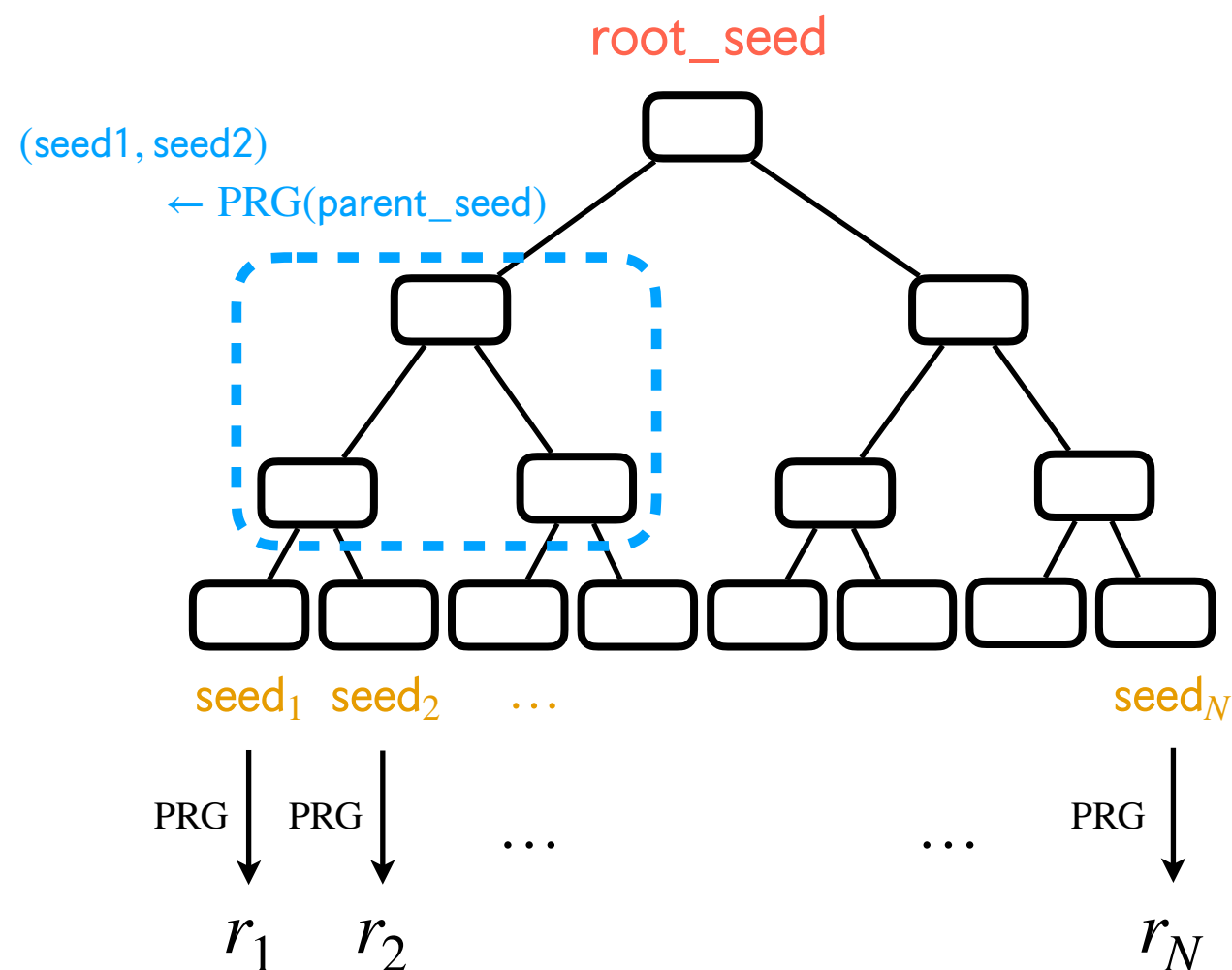
$$\Delta P(X) := P(X) + \sum_{i=1}^N r_i \cdot (X - e_i)$$

(assuming  $\deg P = 1$ )



# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



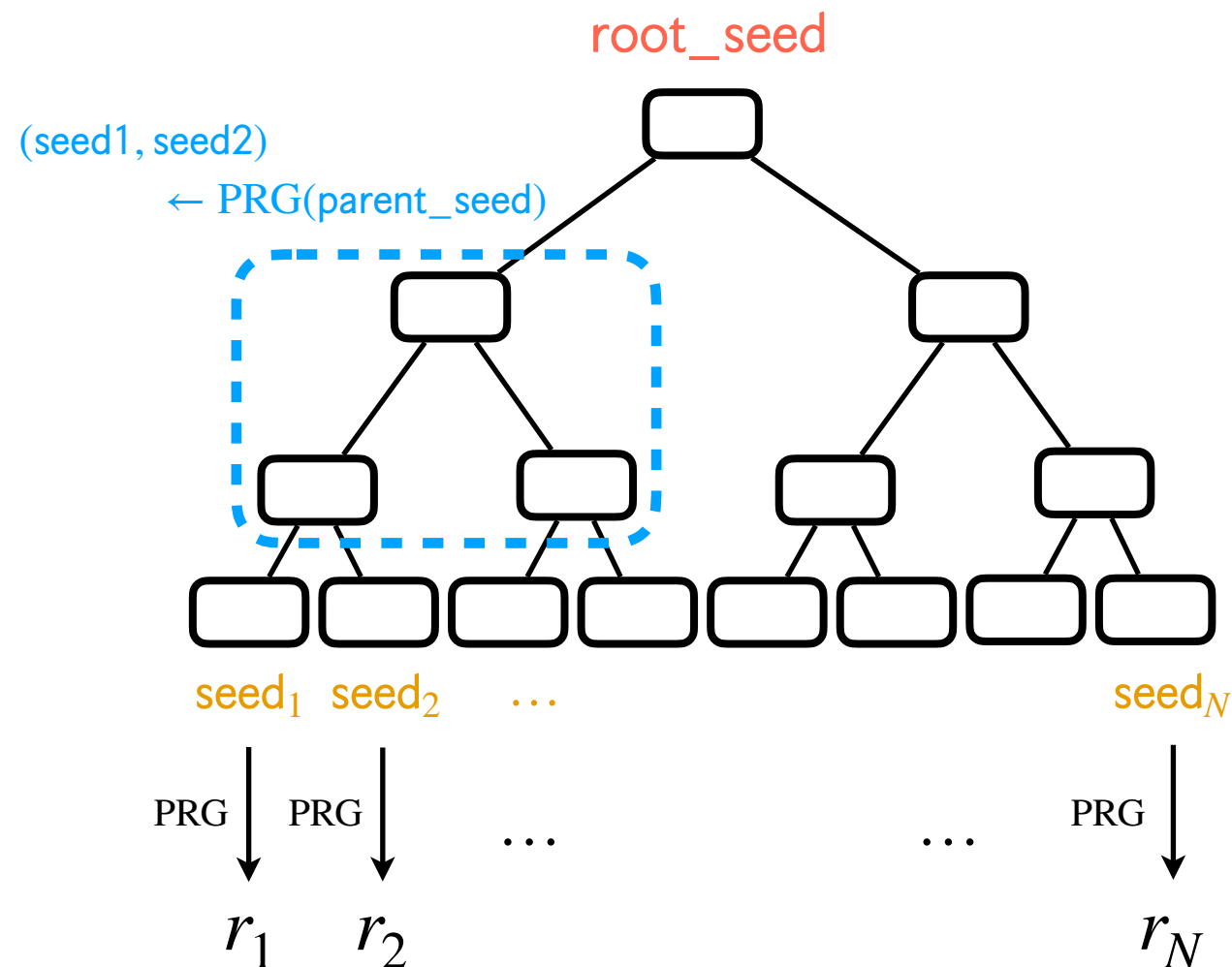
Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



## Commitment:

- Commit to each seed **independently**
- Reveal the masked polynomial  $\Delta P(X)$

## Open $P(e_{i^*})$ :

Reveal all  $\{r_i\}_{i \neq i^*}$  since

$$P(e_{i^*}) = -\Delta P(e_{i^*}) + \sum_{i \neq i^*} r_i \cdot (e_{i^*} - e_i)$$

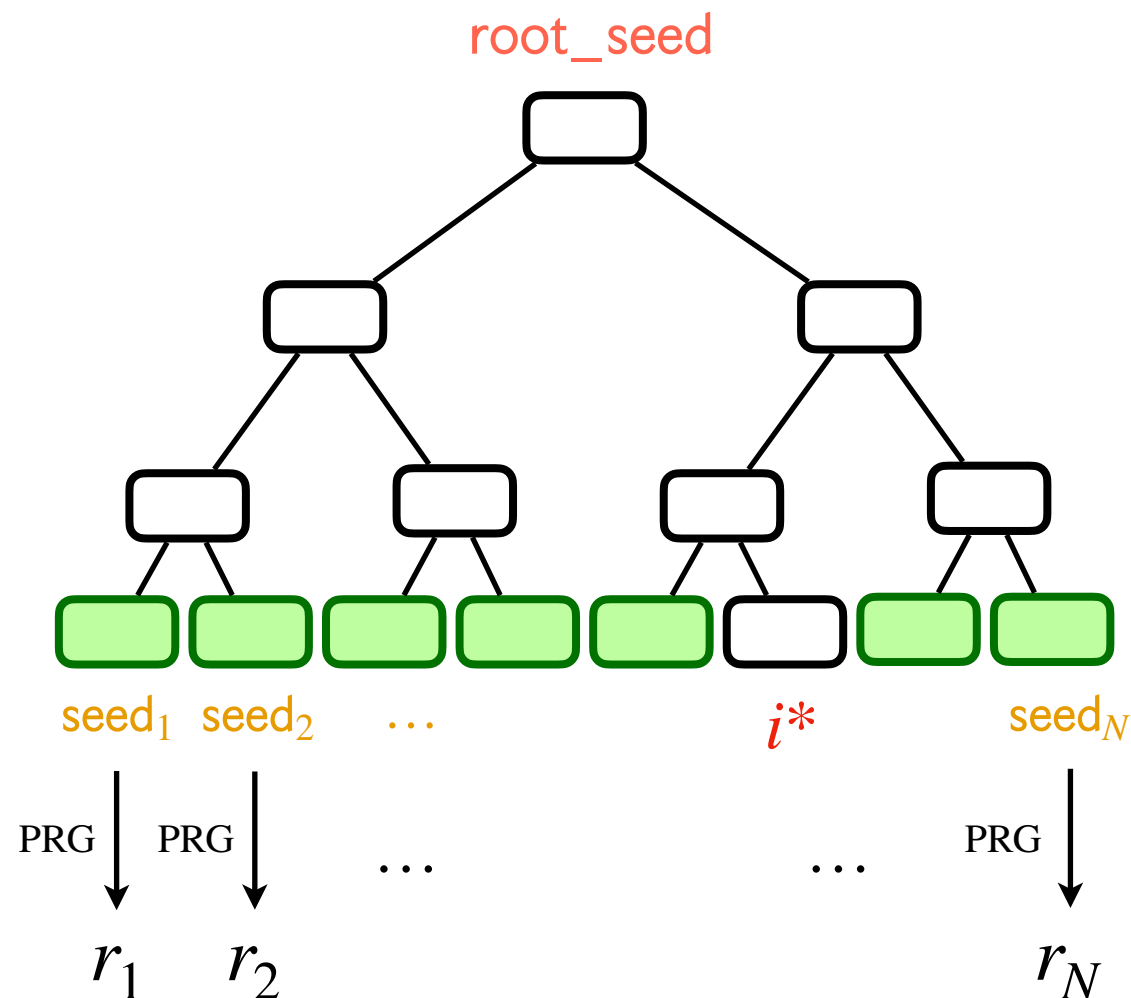
Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



## Commitment:

- Commit to each seed **independently**
- Reveal the masked polynomial  $\Delta P(X)$

## Open $P(e_{i*})$ :

Reveal all  $\{r_i\}_{i \neq i^*}$  since

$$P(e_{i^*}) = -\Delta P(e_{i^*}) + \sum_{i \neq i^*} r_i \cdot (e_{i^*} - e_i)$$

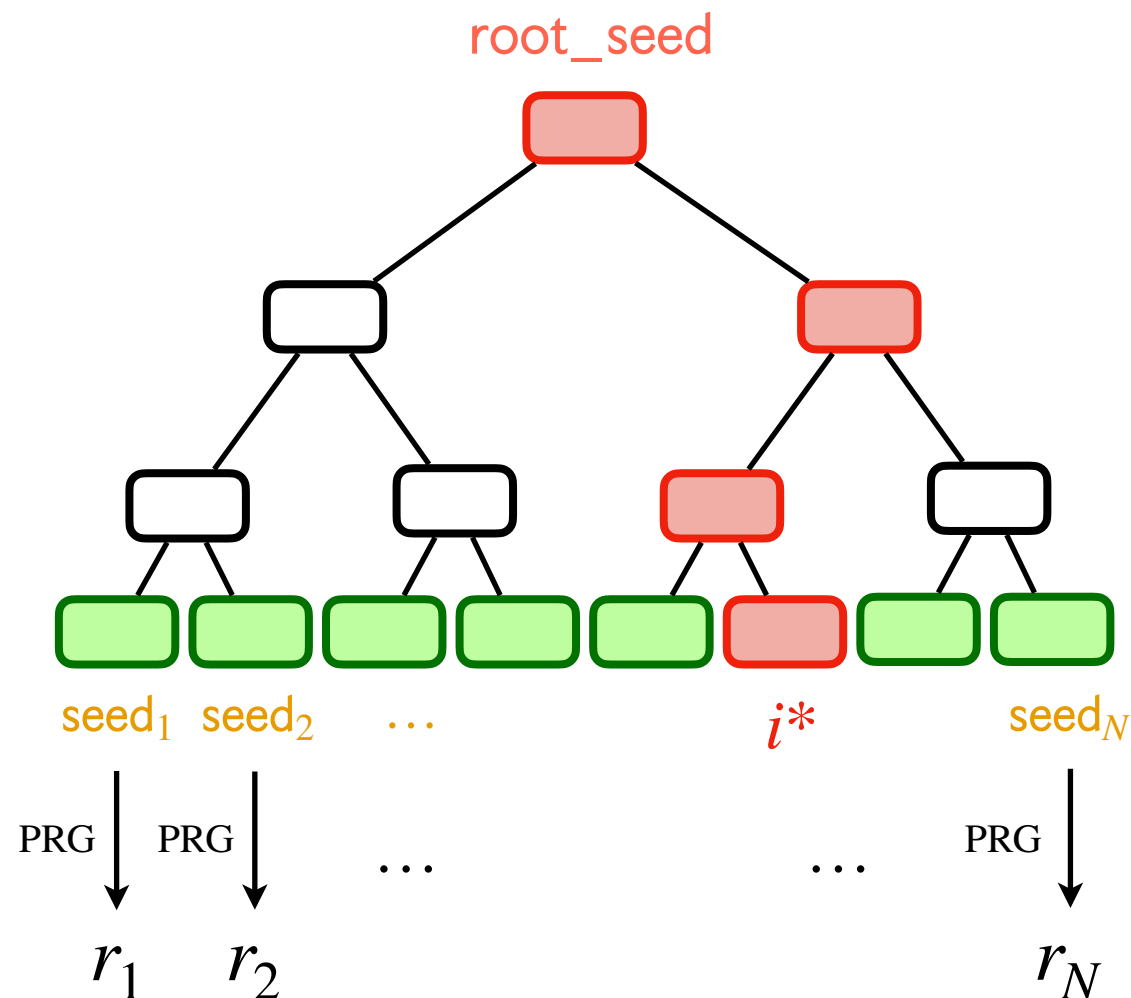
Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



## Commitment:

- Commit to each seed **independently**
- Reveal the masked polynomial  $\Delta P(X)$

## Open $P(e_{i^*})$ :

Reveal all  $\{r_i\}_{i \neq i^*}$  since

$$P(e_{i^*}) = -\Delta P(e_{i^*}) + \sum_{i \neq i^*} r_i \cdot (e_{i^*} - e_i)$$

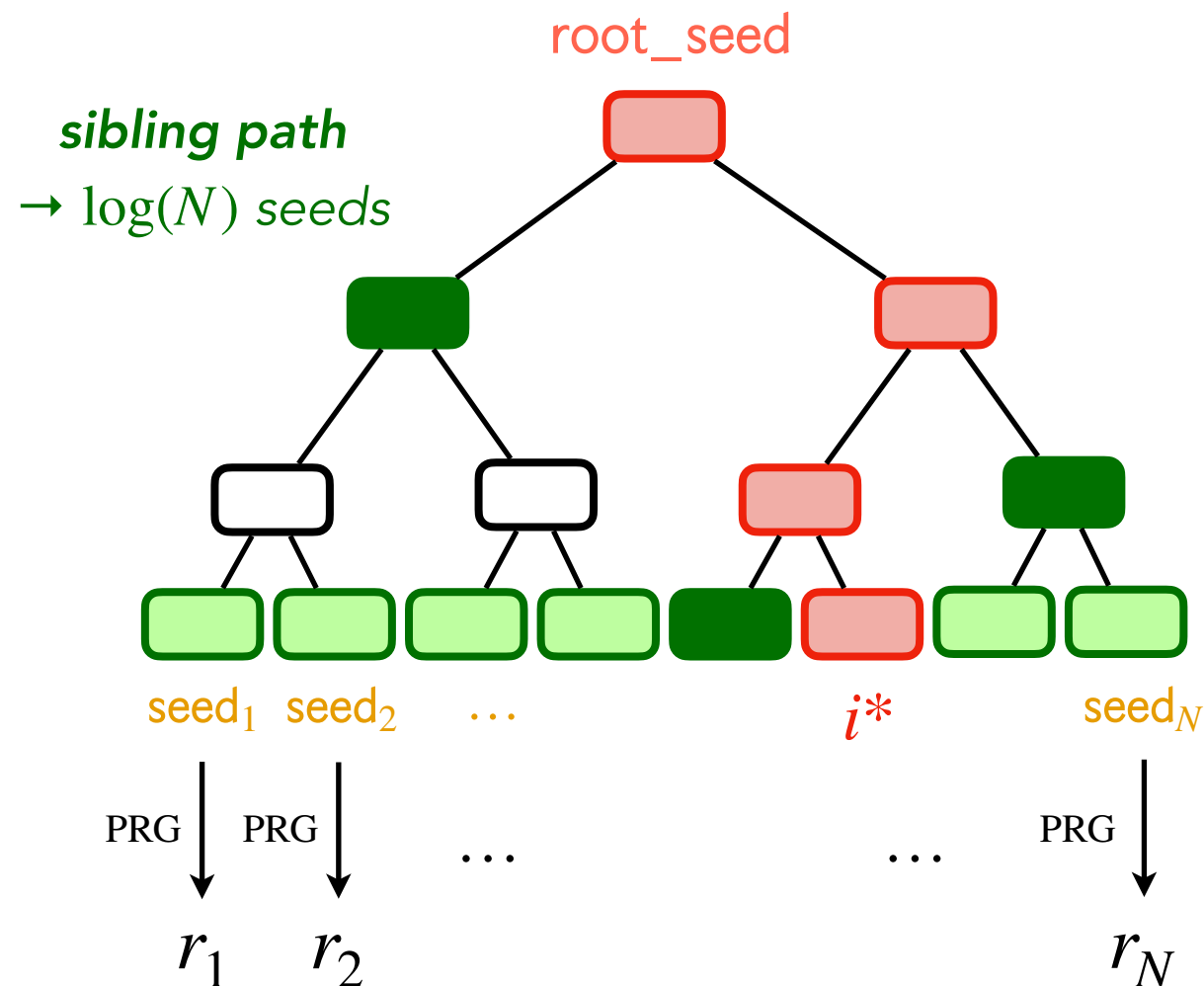
Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



## Commitment:

- Commit to each seed **independently**
- Reveal the masked polynomial  $\Delta P(X)$

## Open $P(e_{i^*})$ :

Reveal all  $\{r_i\}_{i \neq i^*}$  since

$$P(e_{i^*}) = -\Delta P(e_{i^*}) + \sum_{i \neq i^*} r_i \cdot (e_{i^*} - e_i)$$

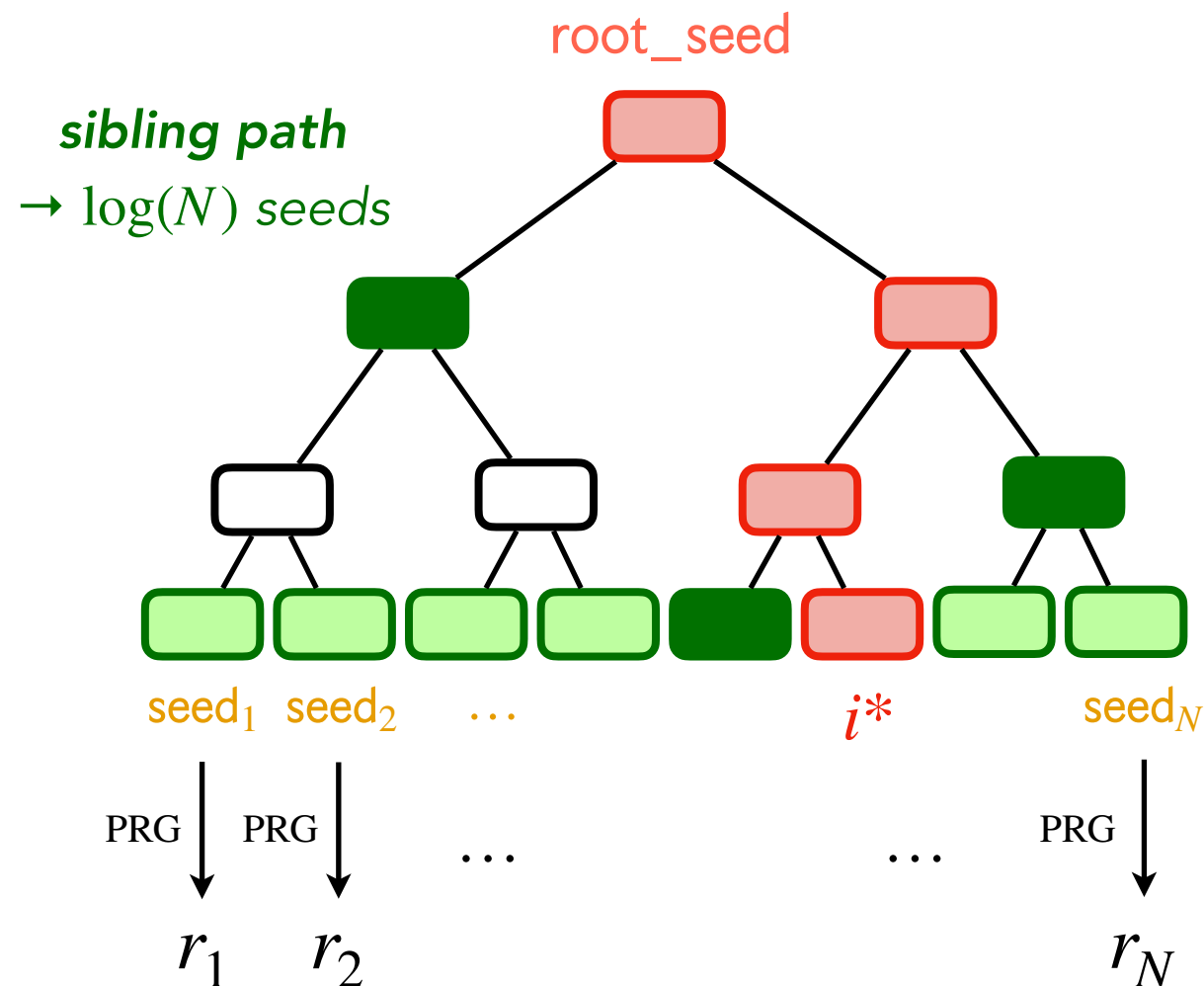
Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

# Option 1: Using a GGM tree (ie. a seed tree)

[GGM84] Goldreich, Goldwasser, Micali: "How to construct random functions (extended extract)" (FOCS 1984)



Build  $\Delta P(X)$  as

$$\Delta P(X) := P(X) + \underbrace{\sum_{i=1}^N r_i \cdot (X - e_i)}_{\text{Mask}}$$

(assuming  $\deg P = 1$ )

## Commitment:

- Commit to each seed **independently**
- Reveal the masked polynomial  $\Delta P(X)$

## Open $P(e_{i*})$ :

Reveal all  $\{r_i\}_{i \neq i*}$  since

$$P(e_{i*}) = -\Delta P(e_{i*}) + \sum_{i \neq i*} r_i \cdot (e_{i*} - e_i)$$

## Properties:

- Cost of sending a tree node:  $\lambda$  bits
- Verification complexity:  $O(N)$
- Nodes contain sensitive information
- Commitment cost:  $O_\lambda(\#P \times \deg P)$
- The committed polynomial  $P$  is naturally of the right degree

# How to commit to polynomials?

(using symmetric primitives)

**Merkle Tree**

④ FRI-based commitments

③ Merkle Trees with  
Ligero-like Proximity Tests

② Degree-enforcing commitment  
(TCitH-MT)

① VOLEitH / TCitH-GGM

**GGM Tree**

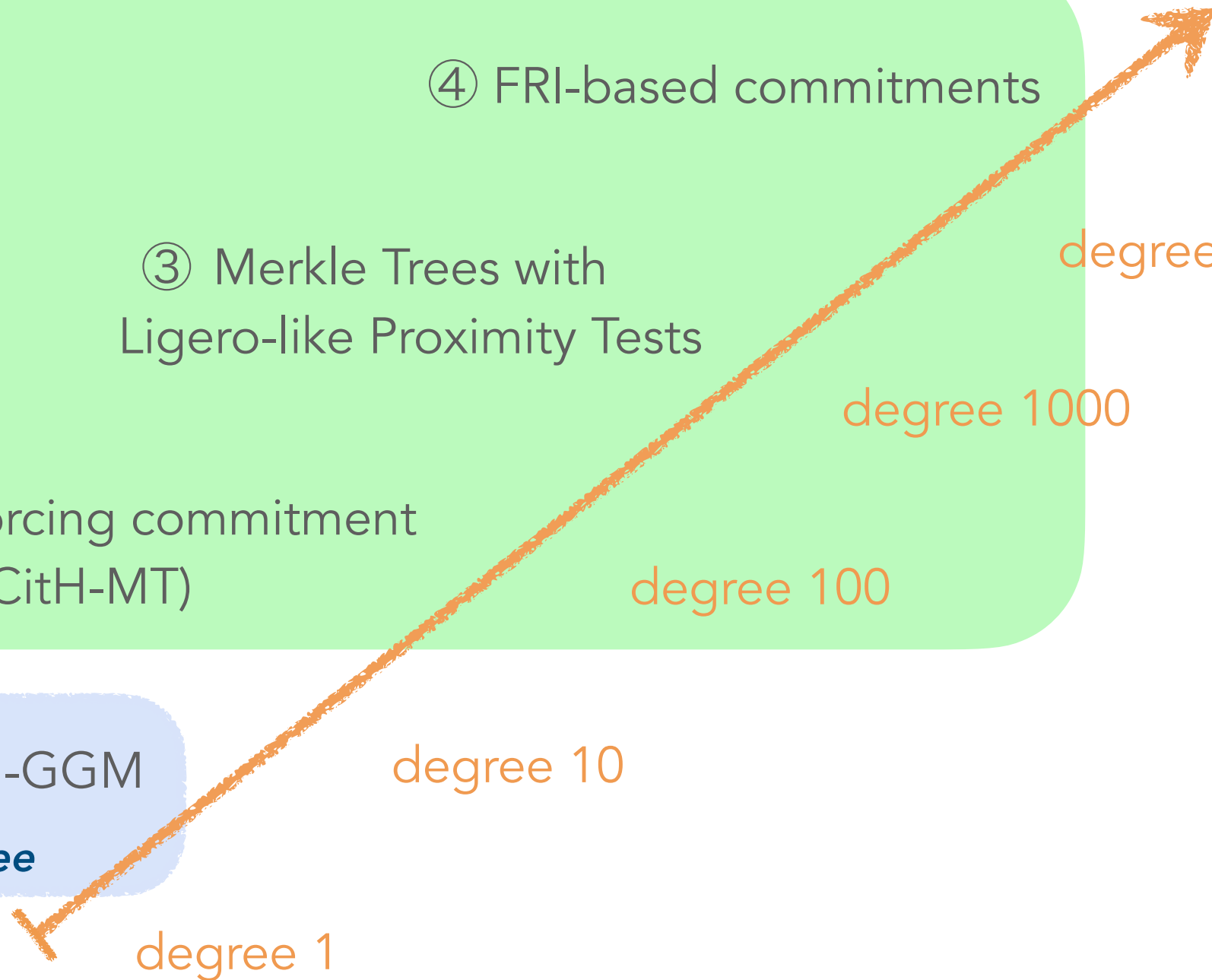
degree 1

degree 10

degree 100

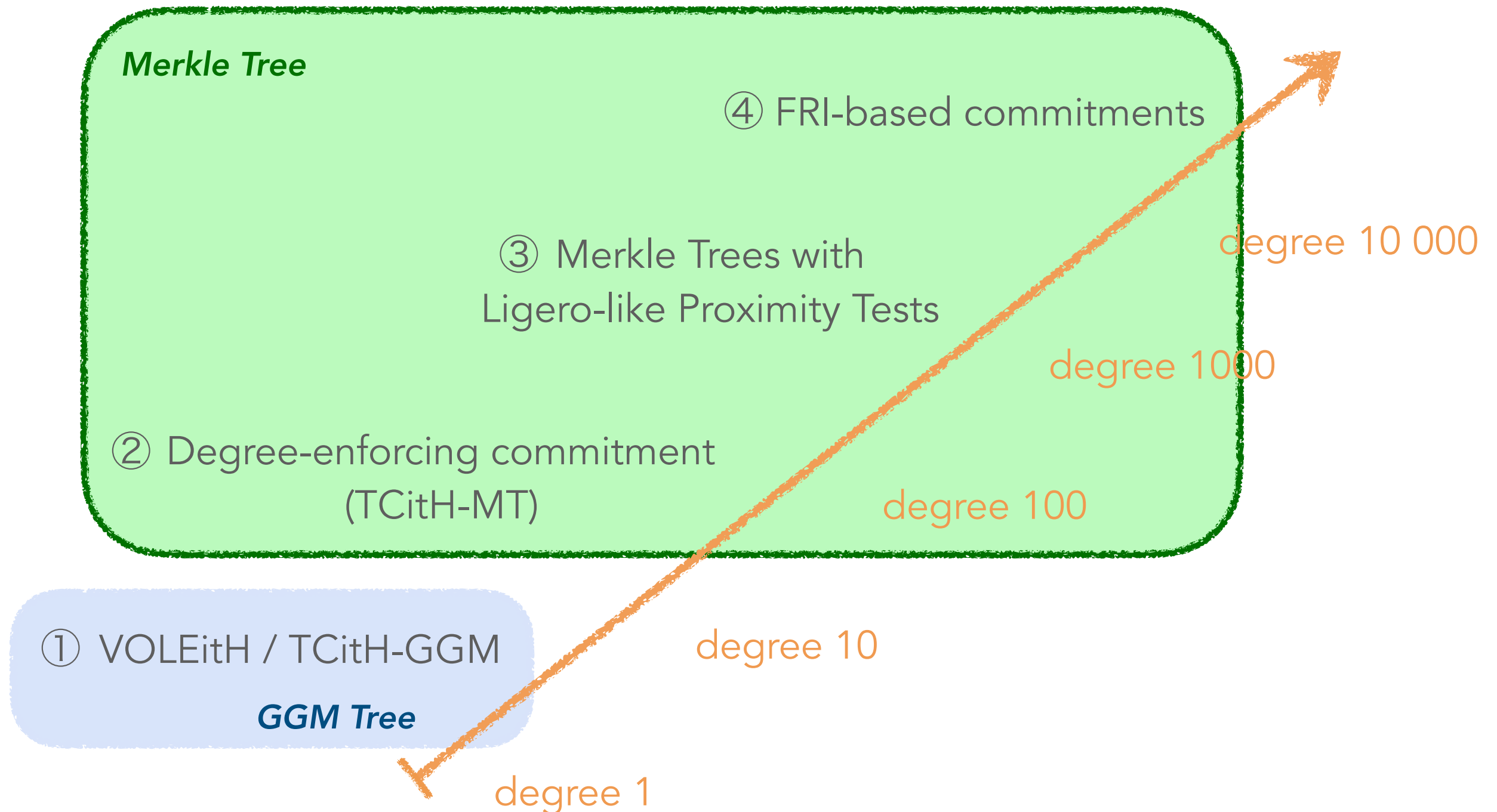
degree 1000

degree 10 000



# How to commit to polynomials?

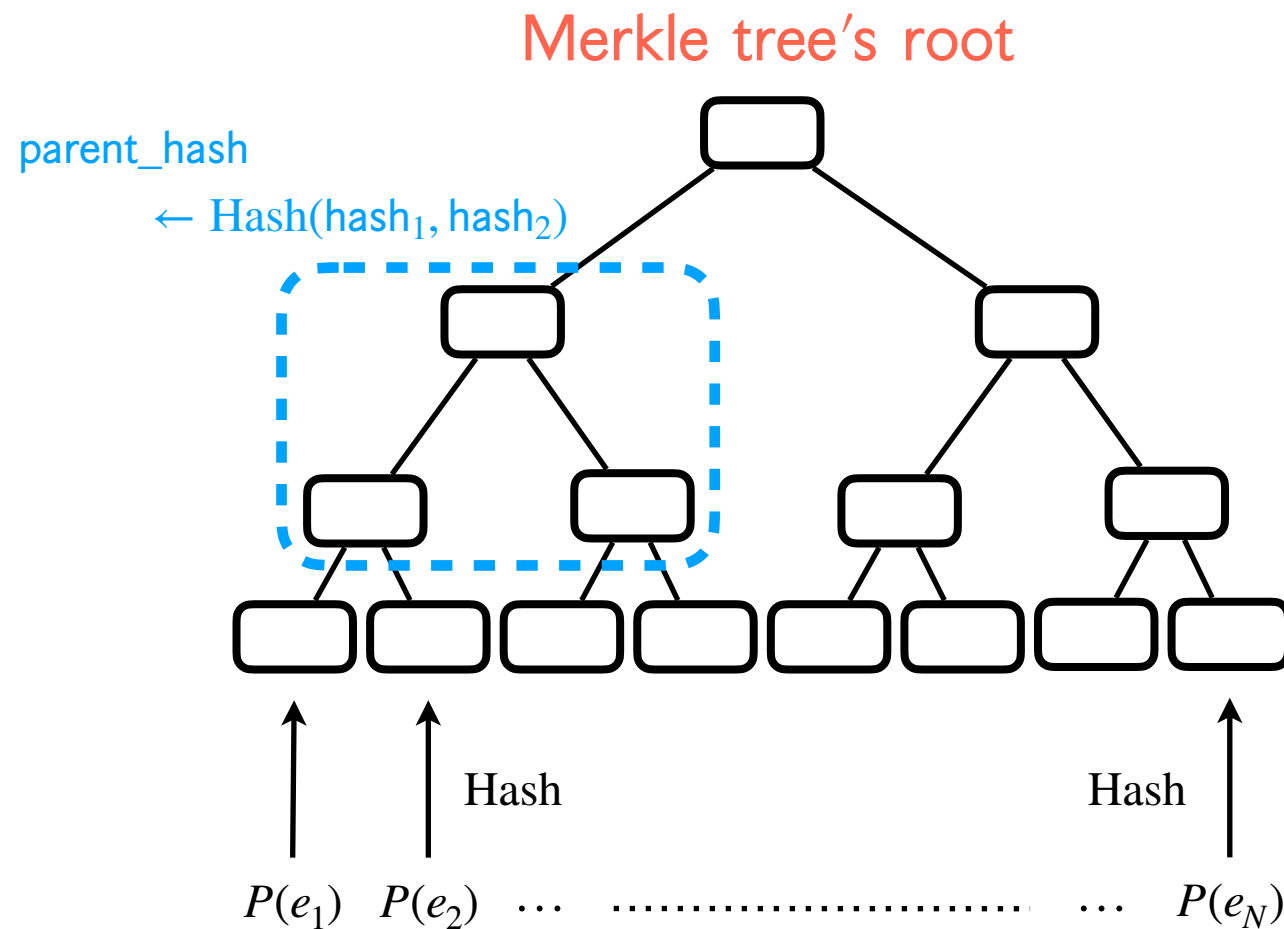
(using symmetric primitives)





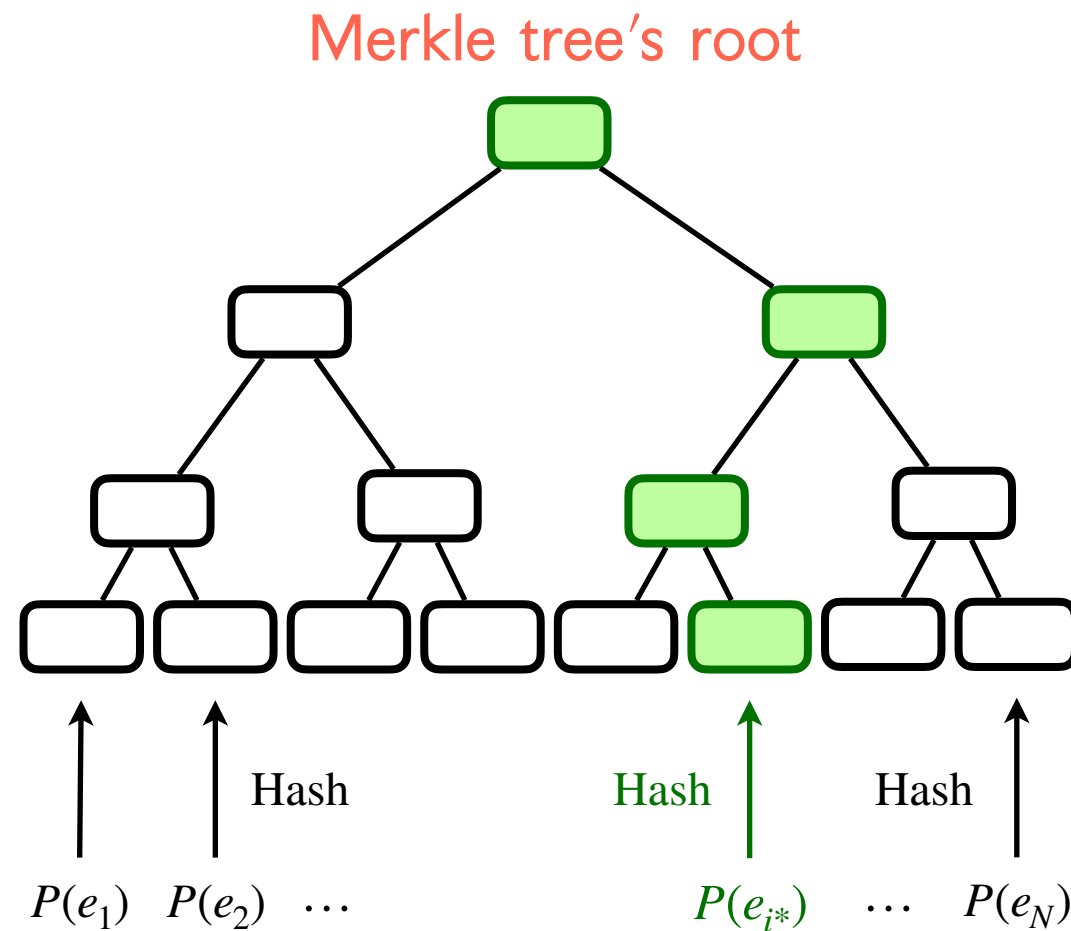
# Option 2: Using a Merkle tree (ie. a hash tree)

[Mer79] Merkle: "Secrecy, authentication, and public key systems" (Ph.D. Thesis, 1979)



# Option 2: Using a Merkle tree (ie. a hash tree)

[Mer79] Merkle: "Secrecy, authentication, and public key systems" (Ph.D. Thesis, 1979)

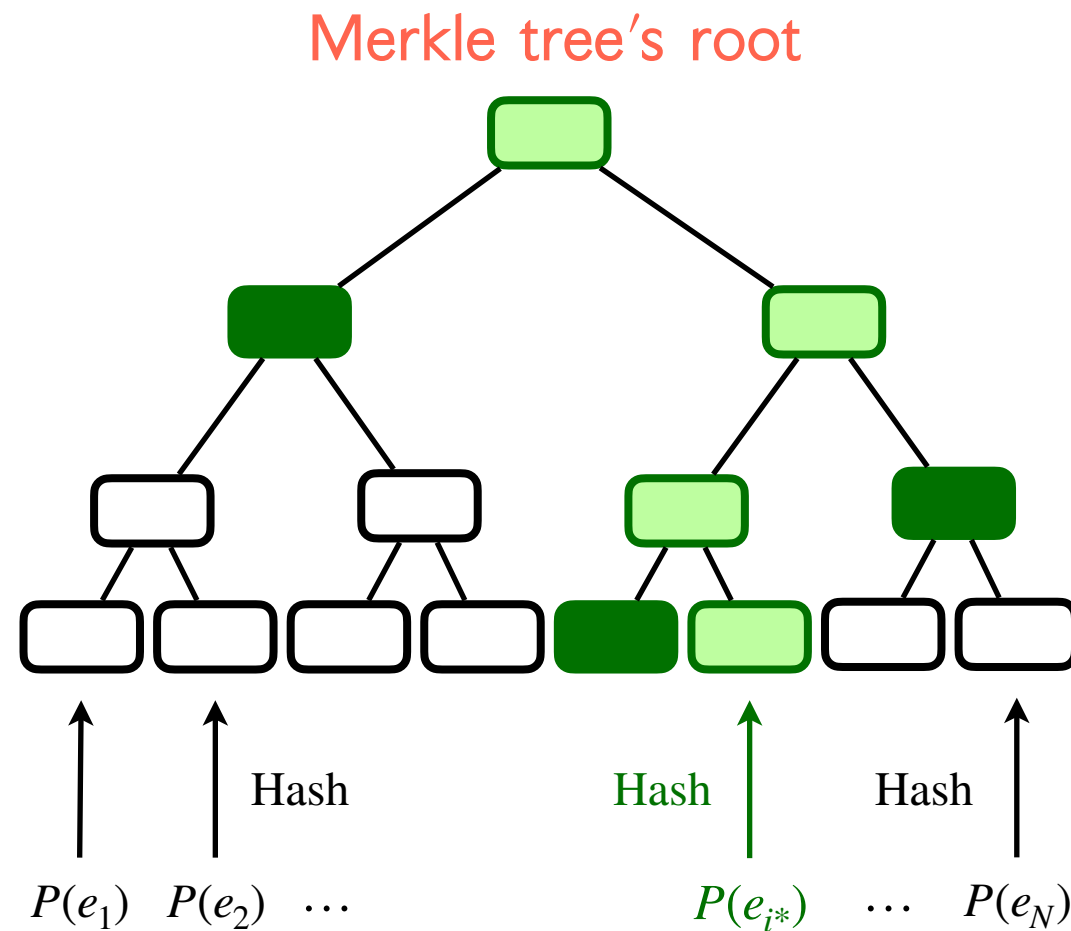


Open  $P(e_{i*})$ :

*Reveal the authentication path of  $P(e_{i*})$*

# Option 2: Using a Merkle tree (ie. a hash tree)

[Mer79] Merkle: "Secrecy, authentication, and public key systems" (Ph.D. Thesis, 1979)



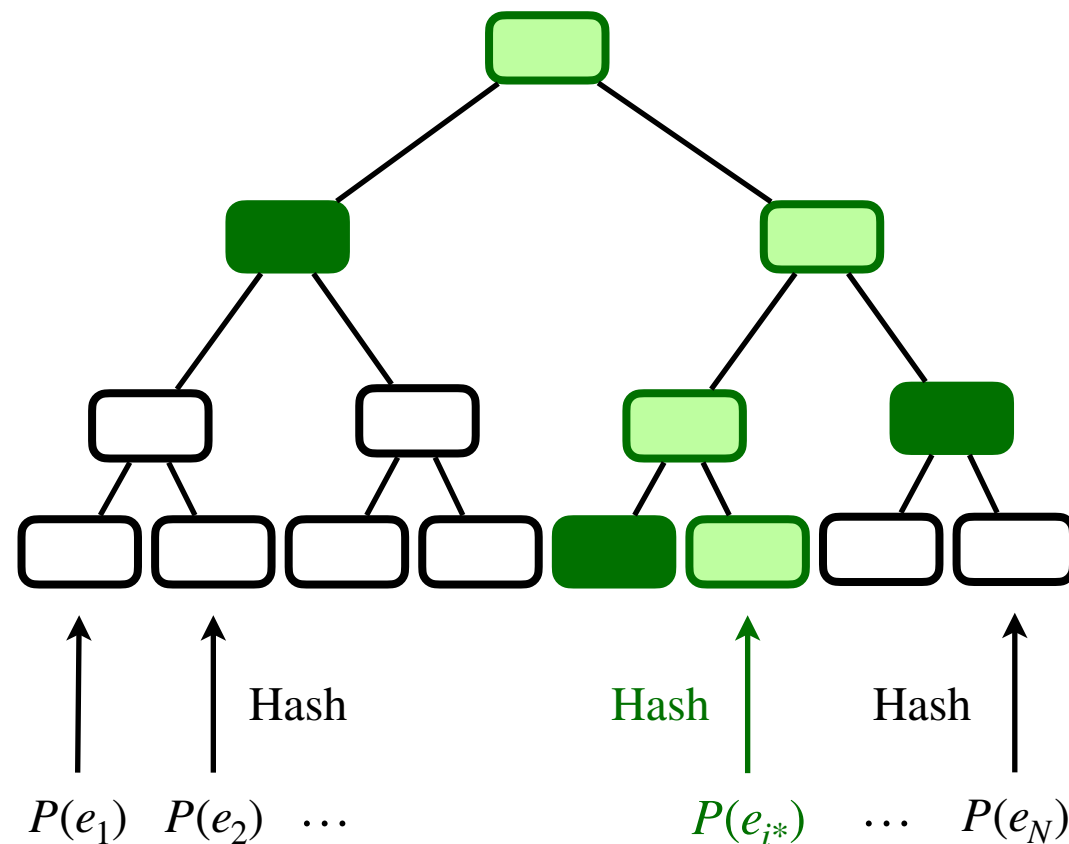
Open  $P(e_{i^*})$ :

*Reveal the authentication path of  $P(e_{i^*})$*

# Option 2: Using a Merkle tree (ie. a hash tree)

[Mer79] Merkle: "Secrecy, authentication, and public key systems" (Ph.D. Thesis, 1979)

Merkle tree's root



Commitment:

- Reveal the Merkle root
- Use a mechanism to ensure that the committed polynomial is of the right degree

Open  $P(e_{i*})$ :

Reveal the authentication path of  $P(e_{i*})$

⚠ Need to ensure that the committed evaluations correspond to a polynomial of the right degree:

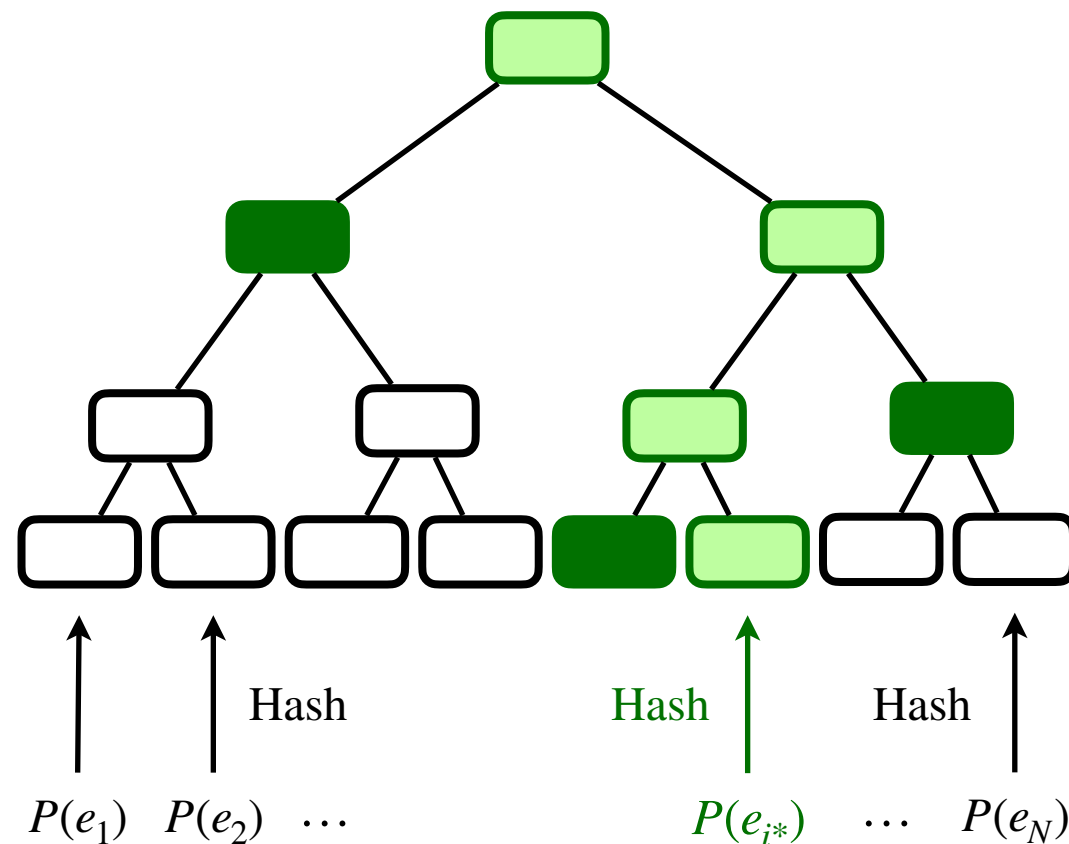
Large polynomials: Proximity Test (Ligero-like or FRI)

Small polynomials: Degree-Enforcing Test (TCitH)

# Option 2: Using a Merkle tree (ie. a hash tree)

[Mer79] Merkle: "Secrecy, authentication, and public key systems" (Ph.D. Thesis, 1979)

Merkle tree's root



Commitment:

- Reveal the Merkle root
- Use a mechanism to ensure that the committed polynomial is of the right degree

Open  $P(e_{i*})$ :

Reveal the authentication path of  $P(e_{i*})$

Properties:

- Cost of sending a tree node:  $2\lambda$  bits
- Verification complexity:  $O(\log_2 N)$
- Nodes contain non-sensitive information
- Commitment cost:  $O_\lambda(\#P + \deg P)$
- Require a mechanism that provides some guarantee on the degree of the committed polynomial

⚠ Need to ensure that the committed evaluations correspond to a polynomial of the right degree:

Large polynomials: Proximity Test (Ligero-like or FRI)

Small polynomials: Degree-Enforcing Test (TCiH)

# How to commit to polynomials?

(using symmetric primitives)

## Merkle Tree

④ FRI-based commitments

③ Merkle Trees with  
Ligero-like Proximity Tests

② Degree-enforcing commitment  
(TCitH-MT)

① VOLEitH / TCitH-GGM

## GGM Tree

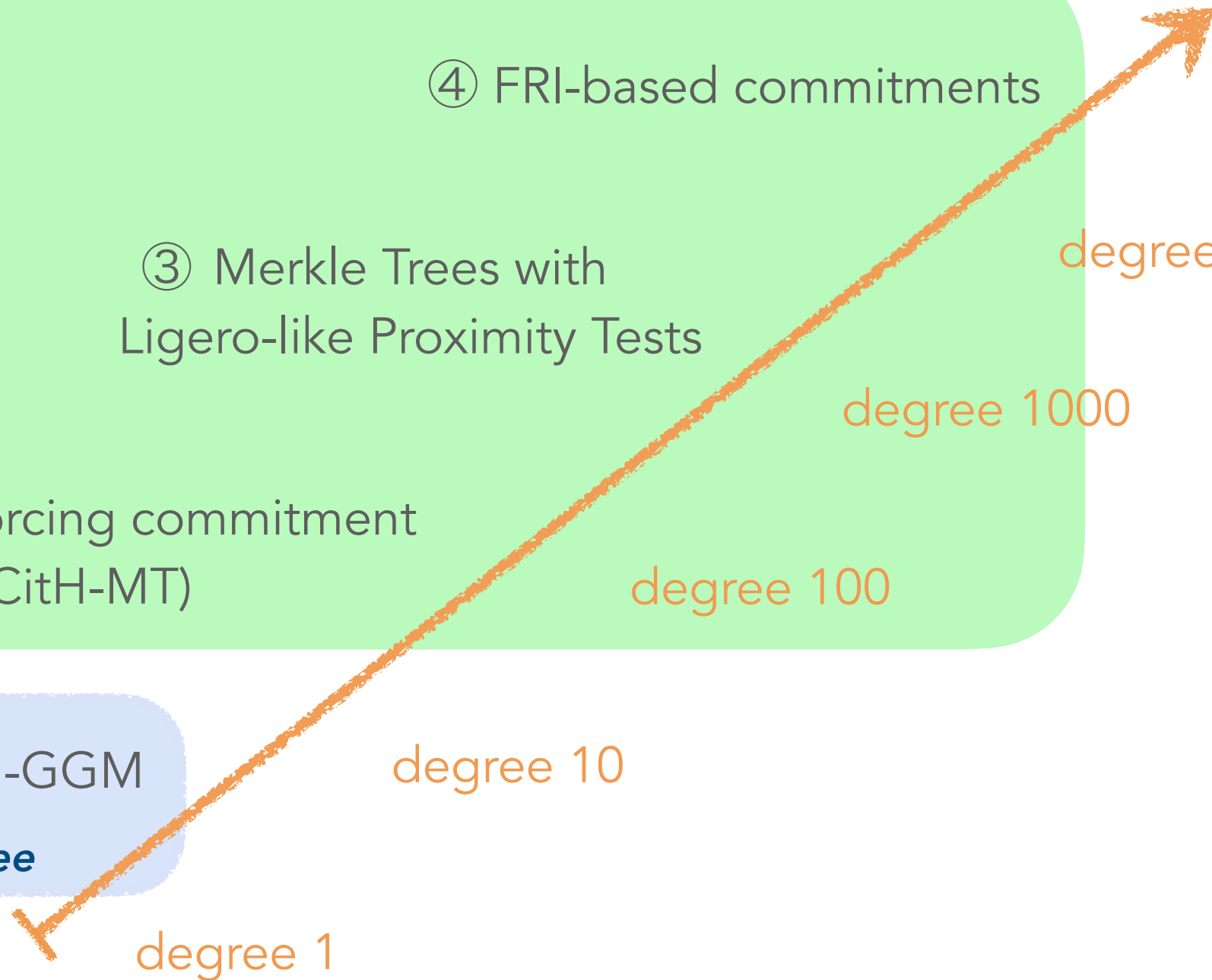
degree 1

degree 10

degree 100

degree 1000

degree 10 000



# How to commit to polynomials?

(using symmetric primitives)

## Merkle Tree

④ FRI-based commitments

③ Merkle Trees with  
Ligero-like Proximity Tests

② Degree-enforcing commitment  
(TCitH-MT)

① VOLEitH / TCitH-GGM

## GGM Tree

degree 1

degree 10

degree 100

degree 1000

degree 10 000

Natively, those techniques  
lead to **small-domain**  
polynomial commitment scheme

# Basic Proof System for Polynomial Constraints

I know  $w_1, \dots, w_n$  such that

$$\begin{cases} f_1(w_1, \dots, w_n) &= 0 \\ \vdots \\ f_m(w_1, \dots, w_n) &= 0, \end{cases}$$

where  $f_1, \dots, f_m$  are public **degree- $d$  polynomials**.

Prove it!

Prover

Degree of the witness polynomials  
 $P_1(X), \dots, P_n(X)$

Verifier

$$\text{Soundness Error} = \frac{d \cdot \ell}{|\mathcal{C}|}$$

Probability that a malicious prover  
can convince the verifier.

Size of the challenge space that  
contains all the possible opened  
evaluations



# Building a full-domain PCS from a small-domain one

- Small-domain PCS:  $|\mathcal{C}| = N$   
where  $N$  is the size of the tree
- Full-domain PCS:  $|\mathcal{C}| = |\mathbb{F}|$  or  $|\mathcal{C}| = |\mathbb{K}|$

# Building a full-domain PCS from a small-domain one



Out-of-sampling  
Technique

Rely on the equivalence:

$P(e) = z$  iff there exists  $Q(X)$  s.t.

$$P(X) - z = (X - e) \cdot Q(X)$$

- Small-domain PCS:  $|\mathcal{C}| = N$   
where  $N$  is the size of the tree
- Full-domain PCS:  $|\mathcal{C}| = |\mathbb{F}|$  or  $|\mathcal{C}| = |\mathbb{K}|$

**[BGKS19]** Ben-Sasson, Goldberg, Kopparty, Saraf. DEEP-FRI: Sampling outside the box improves soundness. ITCS 2020.

# Building a full-domain PCS from a small-domain one

Out-of-sampling  
Technique

Using  
Tensor codes

Rely on the equivalence:

$P(e) = z$  iff there exists  $Q(X)$  s.t.

$$P(X) - z = (X - e) \cdot Q(X)$$

- Small-domain PCS:  $|\mathcal{C}| = N$   
where  $N$  is the size of the tree
- Full-domain PCS:  $|\mathcal{C}| = |\mathbb{F}|$  or  $|\mathcal{C}| = |\mathbb{K}|$

[BCG20] Bootle, Chiesa, Groth. Linear-time arguments with sublinear verification from tensor codes. TCC 2020.

[Lee21] Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. TCC 2021.

[GLS+23] Golovnev, Lee, Setty, Thalers, Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. Crypto 2023.

# Building a full-domain PCS from a small-domain one

Out-of-sampling  
Technique

Using  
Tensor codes

VOLE-in-the-Head  
Technique

Support only degree 1  
Large-domain PCS

Rely on the equivalence:

$P(e) = z$  iff there exists  $Q(X)$  s.t.

$$P(X) - z = (X - e) \cdot Q(X)$$

- Small-domain PCS:  $|\mathcal{C}| = N$   
where  $N$  is the size of the tree
- Full-domain PCS:  $|\mathcal{C}| = |\mathbb{F}|$  or  $|\mathcal{C}| = |\mathbb{K}|$

[BBD+23] Baum, Braun, Delpech, Klooß, Orsini, Roy, Scholl. Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures From VOLE-in-the-Head. Crypto 2023.

# Applications

# Comparison of the approaches

## GGM Tree

- The nodes contain sensitive information.
- The complexity of the tree verification is in  $O(N)$ , where  $N$  is the number of leaves.
- A node is of  $\lambda$  bits

## PCS from GGM Tree

- The committed is naturally of the right degrees.
- Commitment cost:  $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- The nodes do not contain sensitive informations.
- The complexity of the tree verification is in  $O(\log N)$ , where  $N$  is the number of leaves.
- A node is of  $2\lambda$  bits

## PCS from Merkle Tree

- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost:  $O_\lambda(\#P + \deg P)$

# Minimizing the proof sizes

## GGM Tree

- A node is of  $\lambda$  bits
- The committed is naturally of the right degrees.
- Commitment cost (PCS):  
 $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- A node is of  $2\lambda$  bits
- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost (PCS):  
 $O_\lambda(\#P + \deg P)$

# Minimizing the proof sizes

## GGM Tree

- A node is of  $\lambda$  bits
- The committed is naturally of the right degrees.
- Commitment cost (PCS):  
 $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- A node is of  $2\lambda$  bits
- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost (PCS):  
 $O_\lambda(\#P + \deg P)$

Scheme	Using GGM Tree	Using Merkle Tree
Signature schemes	2.5 - 6 KB	7 - 12 KB



# Minimizing the proof sizes

## GGM Tree

- A node is of  $\lambda$  bits
- The committed is naturally of the right degrees.
- Commitment cost (PCS):  
 $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- A node is of  $2\lambda$  bits
- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost (PCS):  
 $O_\lambda(\#P + \deg P)$

Scheme	Using GGM Tree	Using Merkle Tree
Signature schemes	2.5 - 6 KB	7 - 12 KB
ZKPoK of Kyber512's secret key	$\approx$ 12 KB	$\approx$ 14 KB

Using VOLEith

Using SmallWood

# Minimizing the proof sizes

## GGM Tree

- A node is of  $\lambda$  bits
- The committed is naturally of the right degrees.
- Commitment cost (PCS):  
 $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- A node is of  $2\lambda$  bits
- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost (PCS):  
 $O_\lambda(\#P + \deg P)$

Scheme	Using GGM Tree	Using Merkle Tree
Signature schemes	2.5 - 6 KB	7 - 12 KB
ZKPoK of Kyber512's secret key	$\approx 12$ KB	$\approx 14$ KB
ZKPoK of <b>four</b> Kyber512's secret keys	$\approx 36$ KB	$\approx 21$ KB

Using VOLEith

Using SmallWood

# Minimizing the proof sizes

## GGM Tree

- A node is of  $\lambda$  bits
- The committed is naturally of the right degrees.
- Commitment cost (PCS):  
 $O_\lambda(\#P \cdot \deg P)$

## Merkle Tree

- A node is of  $2\lambda$  bits
- Need to add an additional mechanism to ensure the degree of the committed polynomials
- Commitment cost (PCS):  
 $O_\lambda(\#P + \deg P)$

Scheme	Using GGM Tree	Using Merkle Tree
Signature schemes	2.5 - 6 KB	7 - 12 KB
ZKPoK of Kyber512's secret key	$\approx 12$ KB	$\approx 14$ KB
ZKPoK of <b>four</b> Kyber512's secret keys	$\approx 36$ KB	$\approx 21$ KB
ZKPoK of LWE (binary secret, $q \approx 2^{61}$ , $n=4096$ , $m=1024$ )	$\approx 102$ KB	$\approx 21$ KB

Using VOLEith

Using SmallWood

# Efficient Verification Algorithm

---

## GGM Tree

- The complexity of the tree verification is in  $O(N)$ , where  $N$  is the number of leaves.

## Merkle Tree

- The complexity of the tree verification is in  $O(\log N)$ , where  $N$  is the number of leaves.

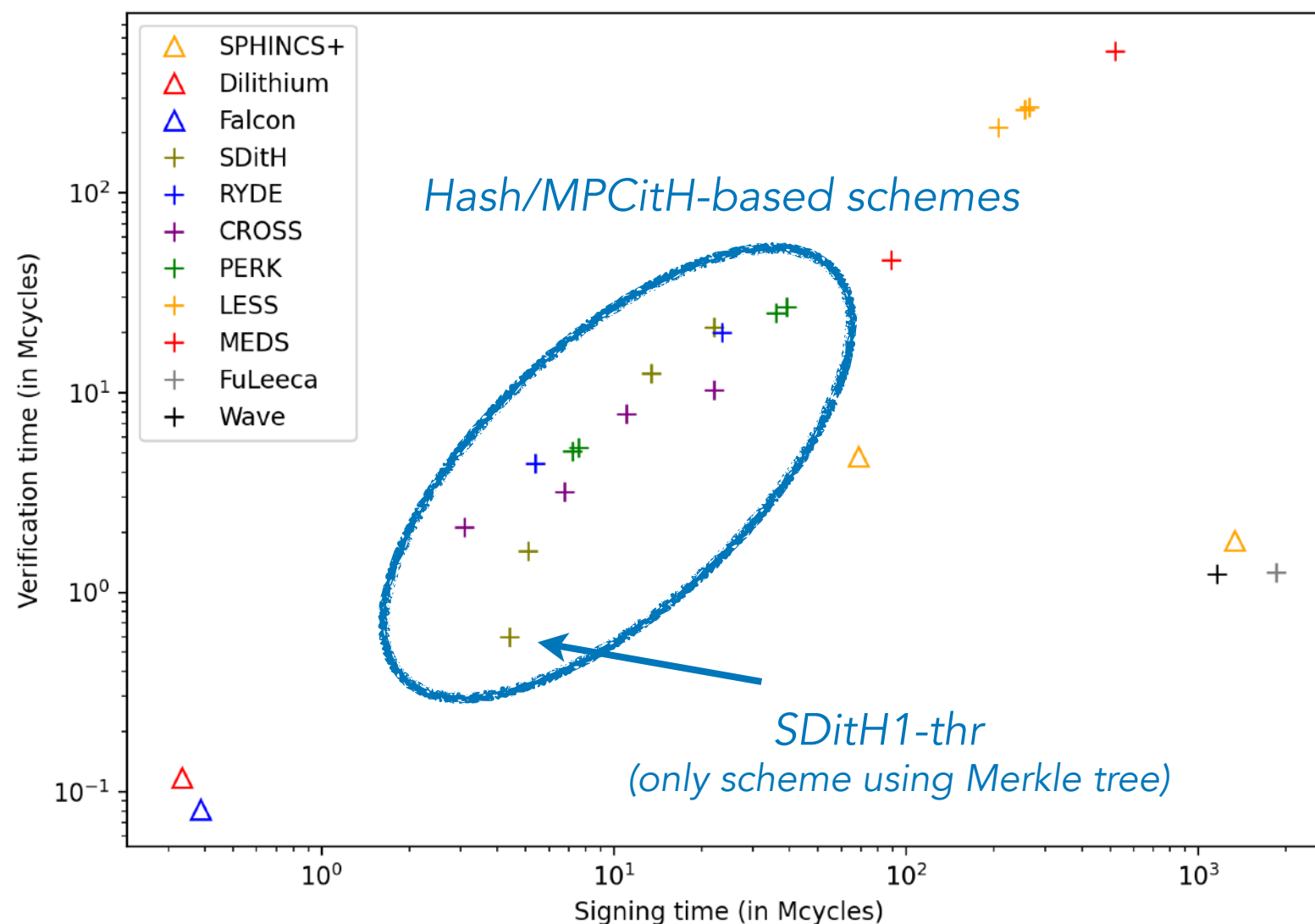
# Efficient Verification Algorithm

## GGM Tree

- The complexity of the tree verification is in  $O(N)$ , where  $N$  is the number of leaves.

## Merkle Tree

- The complexity of the tree verification is in  $O(\log N)$ , where  $N$  is the number of leaves.



# Efficient Verification Algorithm

---

## GGM Tree

- The complexity of the tree verification is in  $O(N)$ , where  $N$  is the number of leaves.

## Merkle Tree

- The complexity of the tree verification is in  $O(\log N)$ , where  $N$  is the number of leaves.

- Fast verification algorithm (for example, SDitH1-thr)

# Efficient Verification Algorithm

## GGM Tree

- The complexity of the tree verification is in  $O(N)$ , where  $N$  is the number of leaves.

## Merkle Tree

- The complexity of the tree verification is in  $O(\log N)$ , where  $N$  is the number of leaves.

- Fast verification algorithm (for example, SDitH1-thr)
- The verification algorithm can be efficiently represented as an arithmetic circuit, *i.e.* leading to SNARK-friendly signatures.

For example,

**[FR25]** Feneuil, Rivain. CAPSS: A Framework for SNARK-Friendly Post-Quantum Signatures. ePrint 2025/061.

<i>Signature Scheme</i>	<i>Signature size</i>	<i>Nb R1CS Constraints</i>
VOLeith-based signatures	2.5 - 5 KB	$\geq 10\,000\,000$
CAPSS-Anemoi ( $2^{256}$ )	$\approx 11$ KB	$\approx 19\,000$
CAPSS-RescuePrime ( $2^{256}$ )	$\approx 12$ KB	$\approx 36\,000$

# Masking-Friendly Scheme

---

## GGM Tree

- The nodes contain sensitive information.

## Merkle Tree

- The nodes do not contain sensitive informations.



# Masking-Friendly Scheme

## GGM Tree

- The nodes contain sensitive information.

## Merkle Tree

- The nodes do not contain sensitive informations.

- Since the nodes does not contain secret information, one does not need to mask Merkle trees, in a context where the secret values are *shared*.

For example, in the context of side-channel attacks:

**[FRW25]** Feneuil, Rivain, Warmé-Janville. Masking-Friendly Post-Quantum Signatures in the Threshold-Computation-in-the-Head Framework. ePrint 2025/520.

# Conclusion

# Conclusion

- ***Polynomial commitment schemes*** (PCS) is the cornerstone of all the recent hash-based proof systems, including the MPCitH ones.

# Conclusion

- **Polynomial commitment schemes** (PCS) is the cornerstone of all the recent hash-based proof systems, including the MPCitH ones.
- There are **two main approaches** to commit to polynomials using only symmetric cryptography, each of them has its own advantage:
  - Using **GGM trees** (a.k.a seed trees)
    - Smaller internal nodes ( $\lambda$  bits)
    - The committed polynomial is ensured to have the right degree
  - Using **Merkle trees** (a.k.a hash trees)
    - Internal nodes are not sensitive information
    - Sublinear verification
    - Asymptotically-better communication cost

# Conclusion

- **Polynomial commitment schemes** (PCS) is the cornerstone of all the recent hash-based proof systems, including the MPCitH ones.
- There are **two main approaches** to commit to polynomials using only symmetric cryptography, each of them has its own advantage:
  - Using **GGM trees** (a.k.a seed trees)
    - Smaller internal nodes ( $\lambda$  bits)
    - The committed polynomial is ensured to have the right degree
  - Using **Merkle trees** (a.k.a hash trees)
    - Internal nodes are not sensitive information
    - Sublinear verification
    - Asymptotically-better communication cost
  - We can enhance the soundness/performance by converting the small-domain PCS into a full-domain PCS.

# Conclusion

- **Polynomial commitment schemes** (PCS) is the cornerstone of all the recent hash-based proof systems, including the MPCitH ones.
- There are **two main approaches** to commit to polynomials using only symmetric cryptography, each of them has its own advantage:
  - Using **GGM trees** (a.k.a seed trees)
    - Smaller internal nodes ( $\lambda$  bits)
    - The committed polynomial is ensured to have the right degree
  - Using **Merkle trees** (a.k.a hash trees)
    - Internal nodes are not sensitive information
    - Sublinear verification
    - Asymptotically-better communication cost
  - We can enhance the soundness/performance by converting the small-domain PCS into a full-domain PCS.
- **Depending on the context**, one approach could be better than the other one.

# Conclusion

- **Polynomial commitment schemes** (PCS) is the cornerstone of all the recent hash-based proof systems, including the MPCitH ones.
- There are **two main approaches** to commit to polynomials using only symmetric cryptography, each of them has its own advantage:
  - Using **GGM trees** (a.k.a seed trees)
    - Smaller internal nodes ( $\lambda$  bits)
    - The committed polynomial is ensured to have the right degree
  - Using **Merkle trees** (a.k.a hash trees)
    - Internal nodes are not sensitive information
    - Sublinear verification
    - Asymptotically-better communication cost
  - We can enhance the soundness/performance by converting the small-domain PCS into a full-domain PCS.
- **Depending on the context**, one approach could be better than the other one.

**Thank you for your attention.**