The Polynomial-IOP Vision of the Latest MPCitH Frameworks for Signature Schemes

Thibauld Feneuil

ACCESS Seminar

October 15, 2024, *online*

CryptoExperts

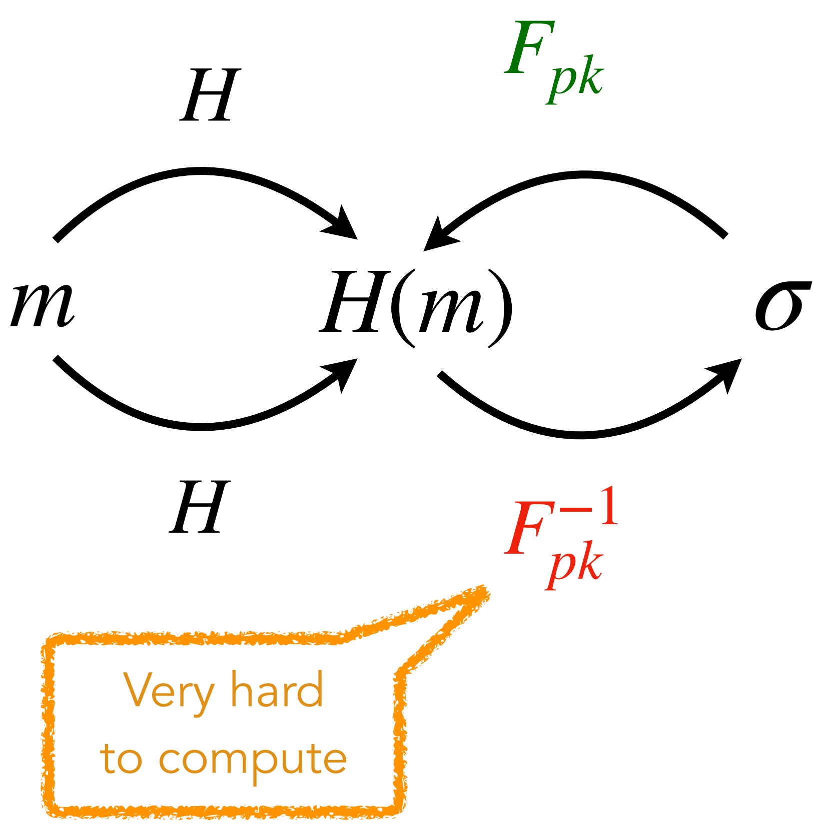
WE INNOVATE TO SECURE YOUR BUSINESS

Table of Contents

Introduction

How to build signature schemes?

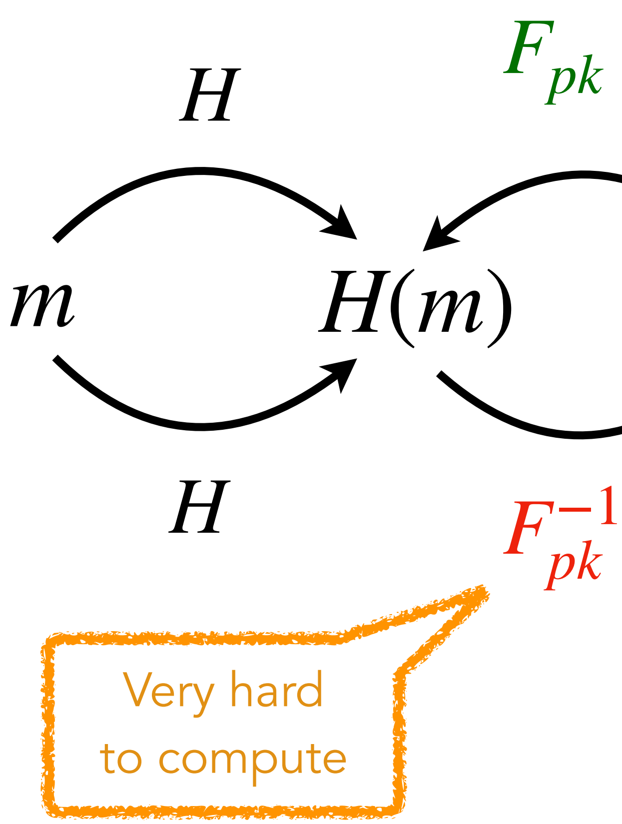
Hash & Sign



- Short signatures
- "Trapdoor" in the public key

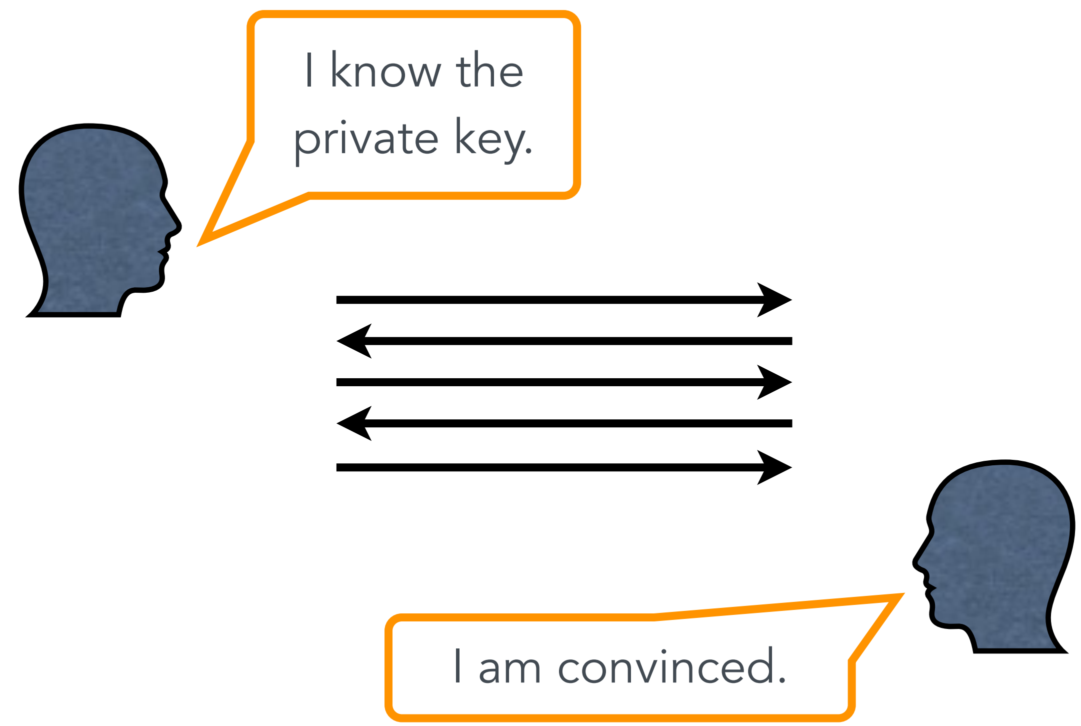
How to build signature schemes?

Hash & Sign



- Short signatures
- “Trapdoor” in the public key

From an identification scheme



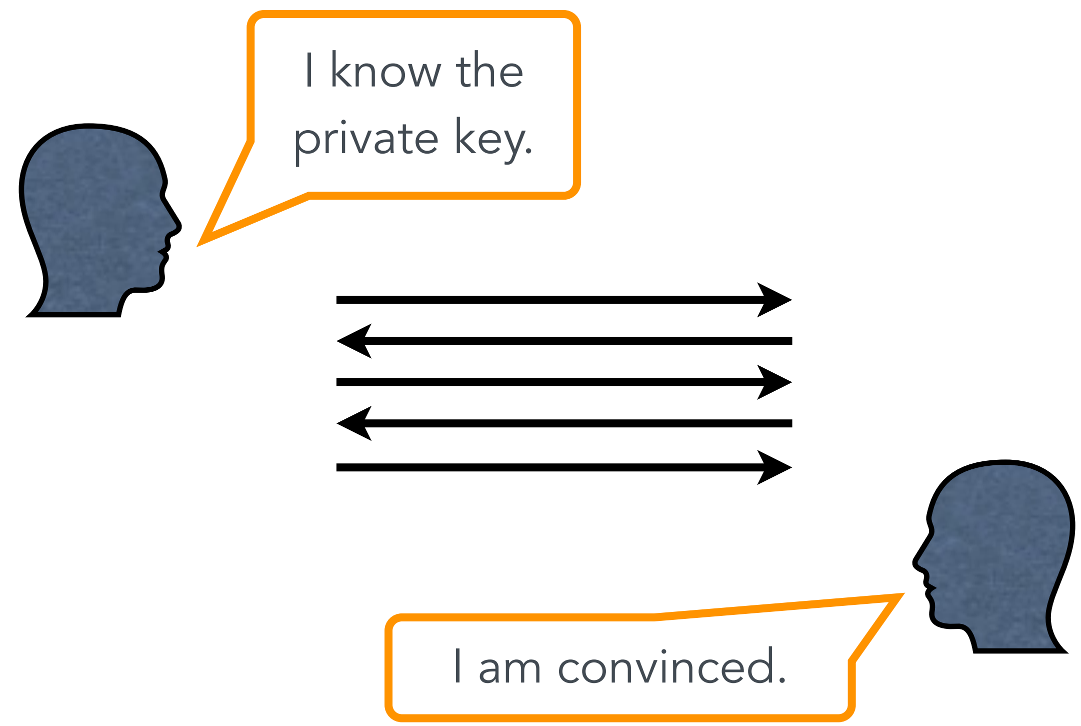
- Large(r) signatures
- Short public key

How to build signature schemes?

Hash & Sign

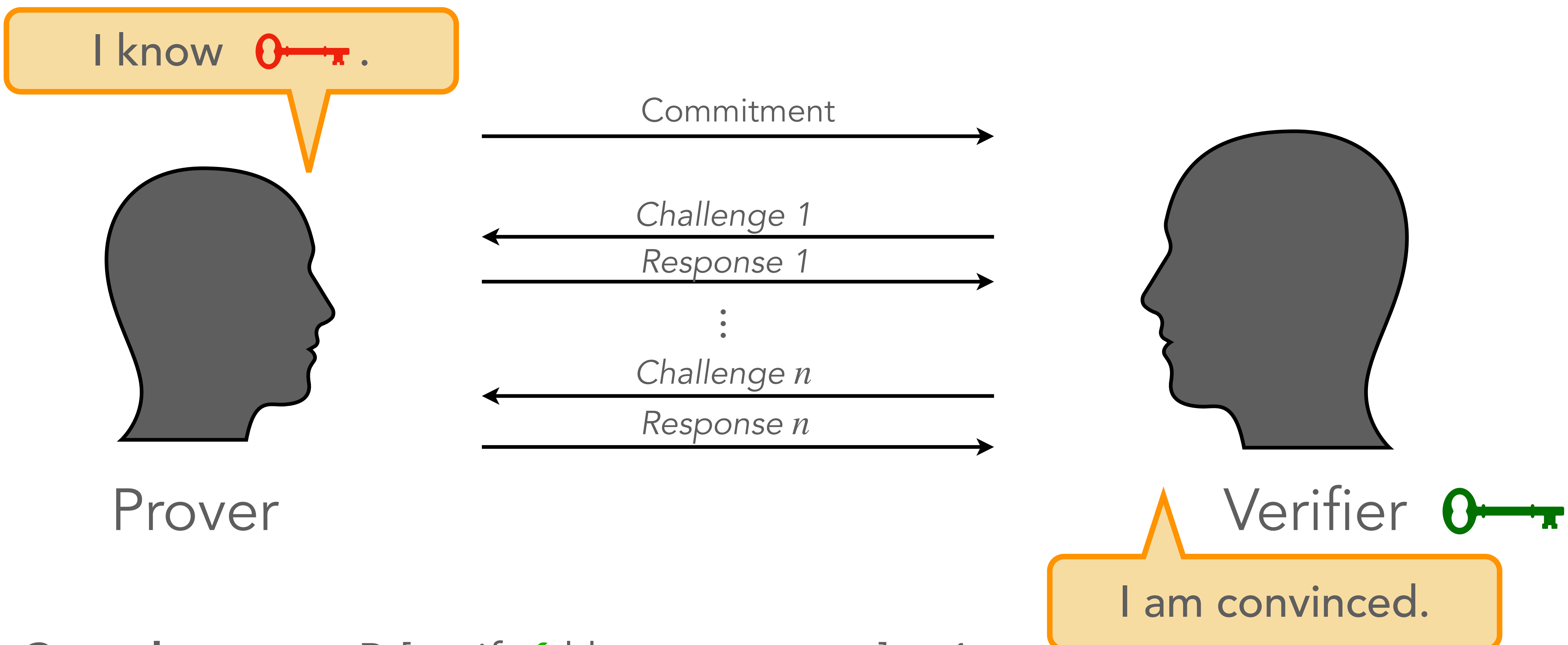
- Short signatures
- “Trapdoor” in the public key

From an identification scheme



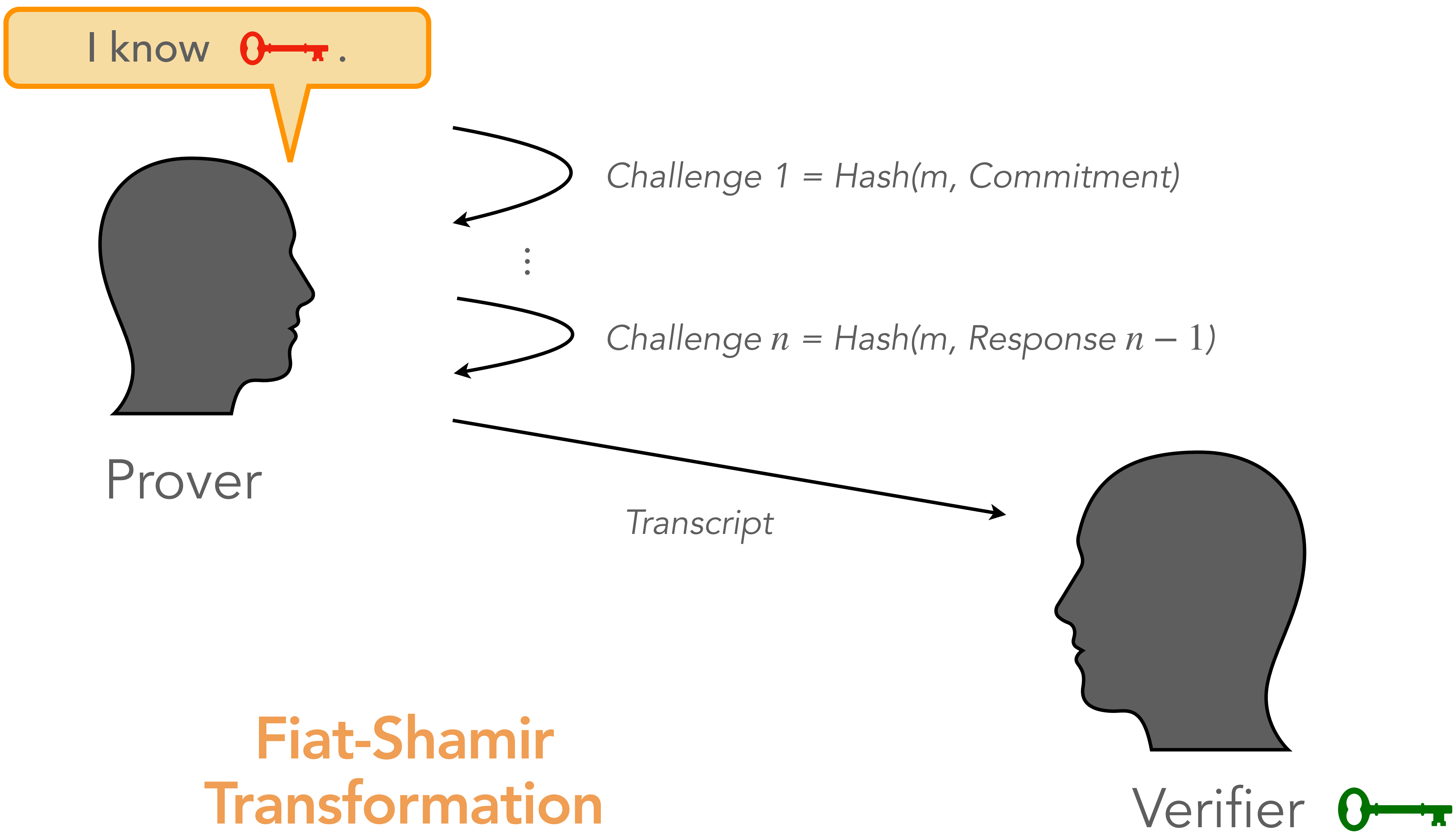
- Large(r) signatures
- Short public key

Identification Scheme



- **Completeness:** Pr[verif ✓ | honest prover] = 1
- **Soundness:** Pr[verif ✓ | malicious prover] $\leq \varepsilon$ (e.g. 2^{-128})
- **Zero-knowledge:** verifier learns nothing on .

Identification Scheme



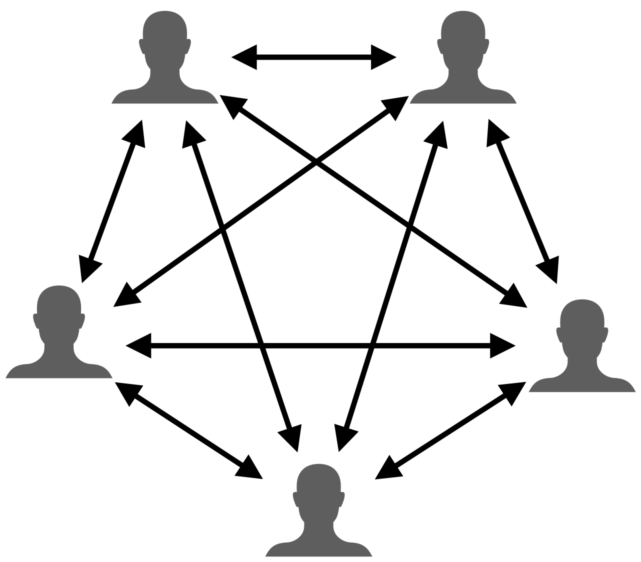
Fiat-Shamir Transformation

m: message to sign

MPC in the Head

- **[IKOS07]** Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Amit Sahai: "Zero-knowledge from secure multiparty computation" (STOC 2007)

- Turn a *multiparty computation* (MPC) into an identification scheme / zero-knowledge proof of knowledge

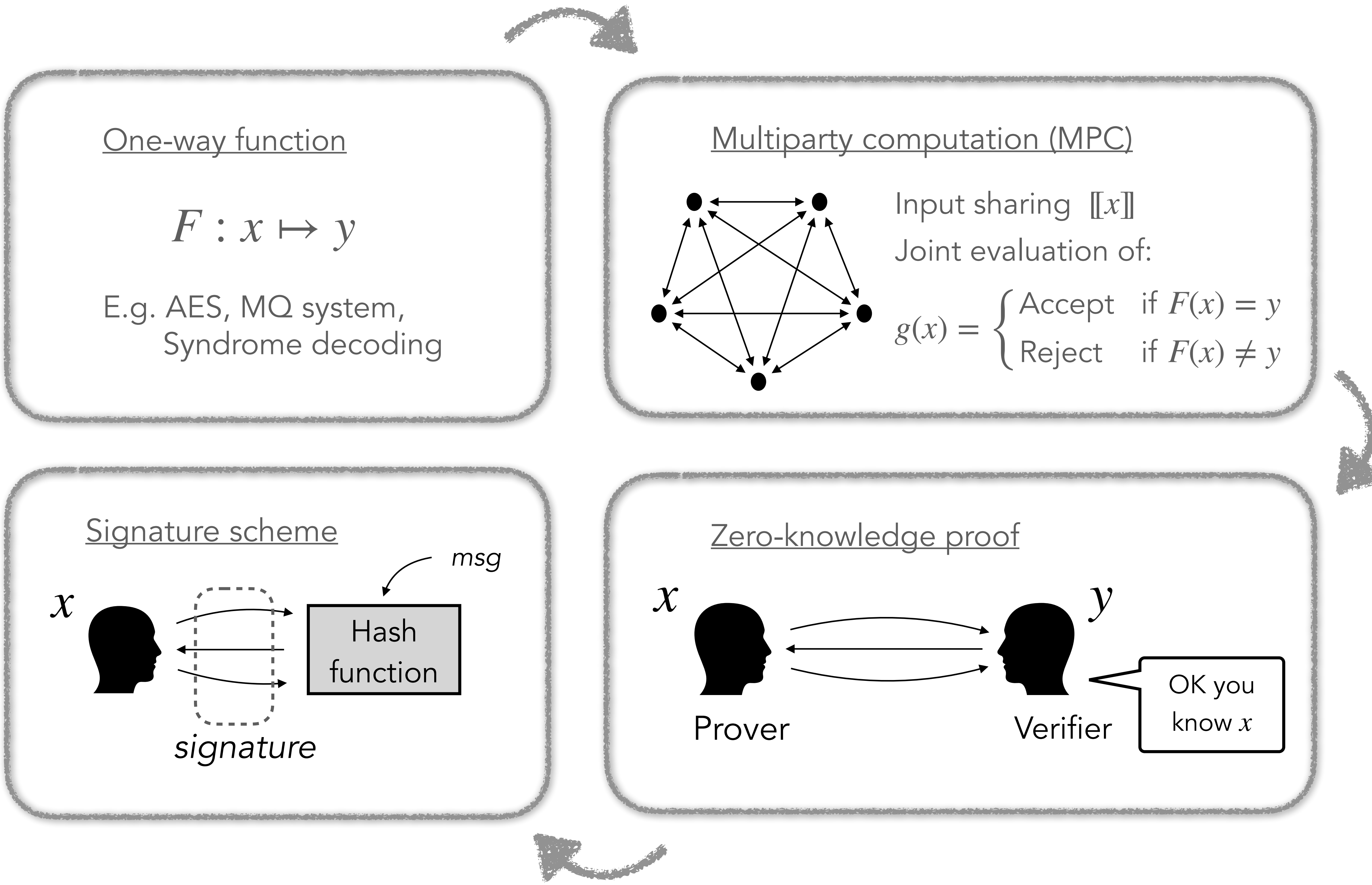


- **Generic**: can be applied to any cryptographic problem

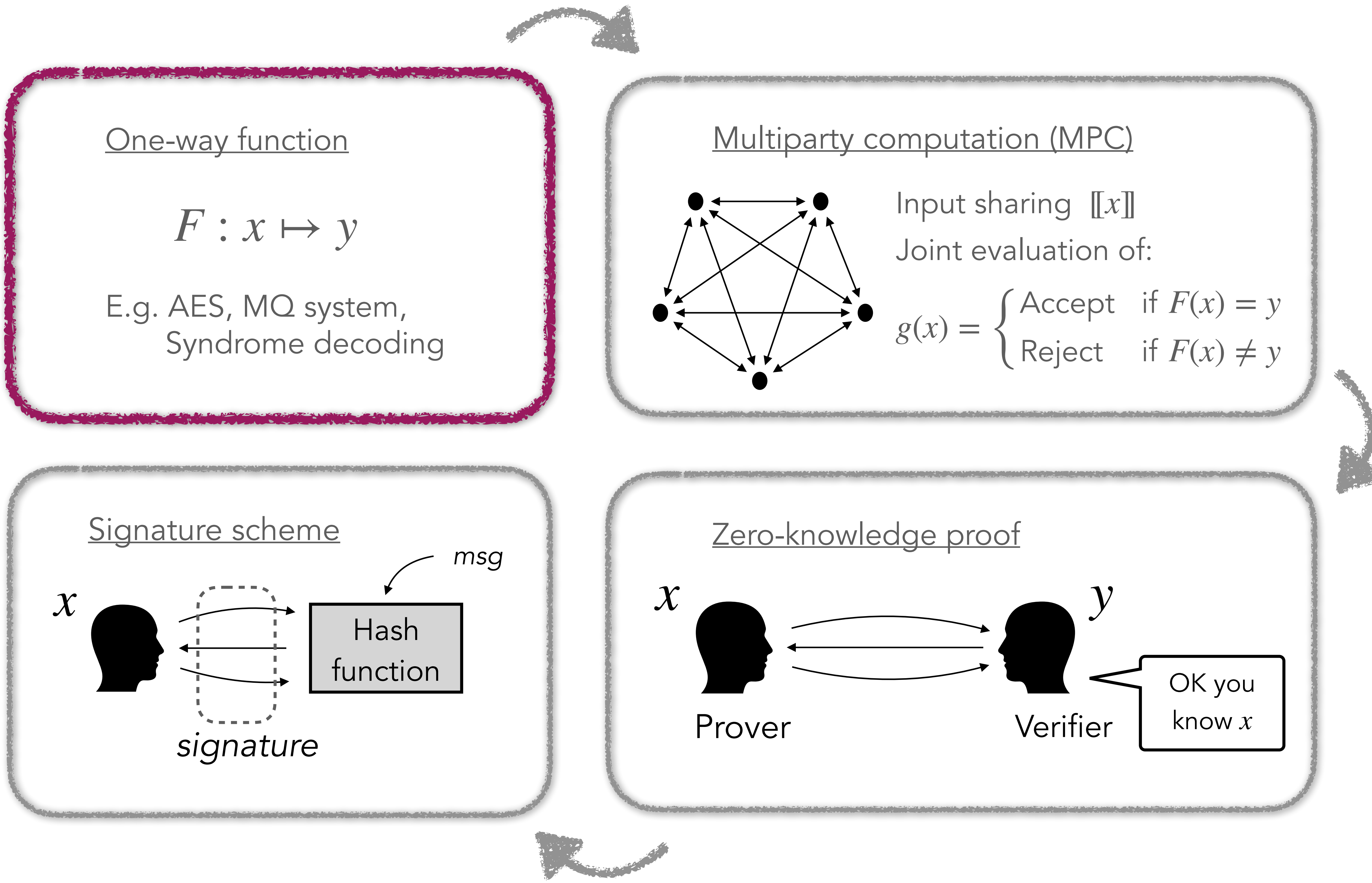
MPC in the Head

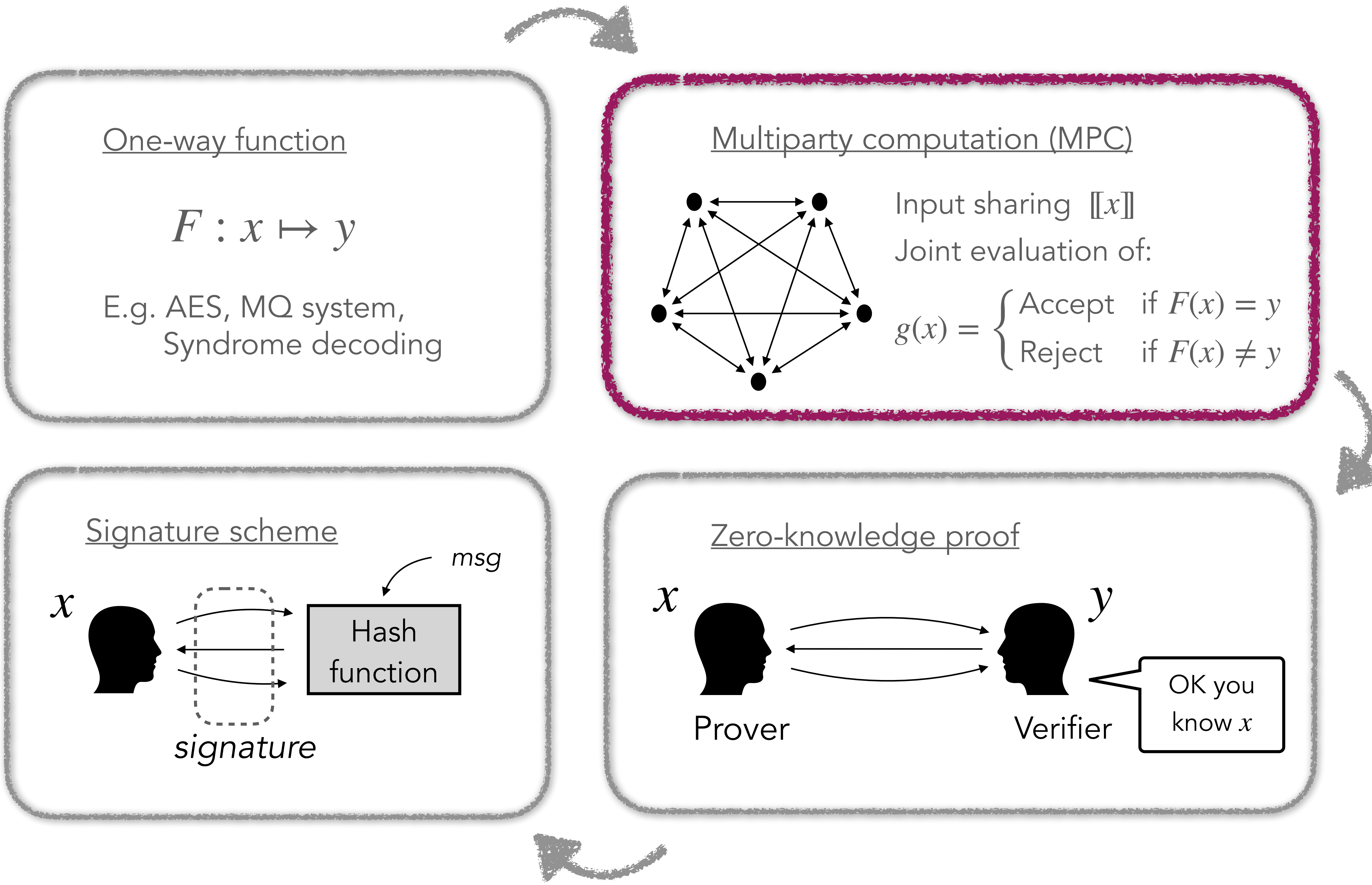
- **[IKOS07]** Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Amit Sahai: "Zero-knowledge from secure multiparty computation" (STOC 2007)
- Convenient to build (candidate) **post-quantum signature** schemes
- **Picnic**: submission to NIST (2017)
- First round of recent NIST call: 7~9 MPCitH schemes / 40 submissions

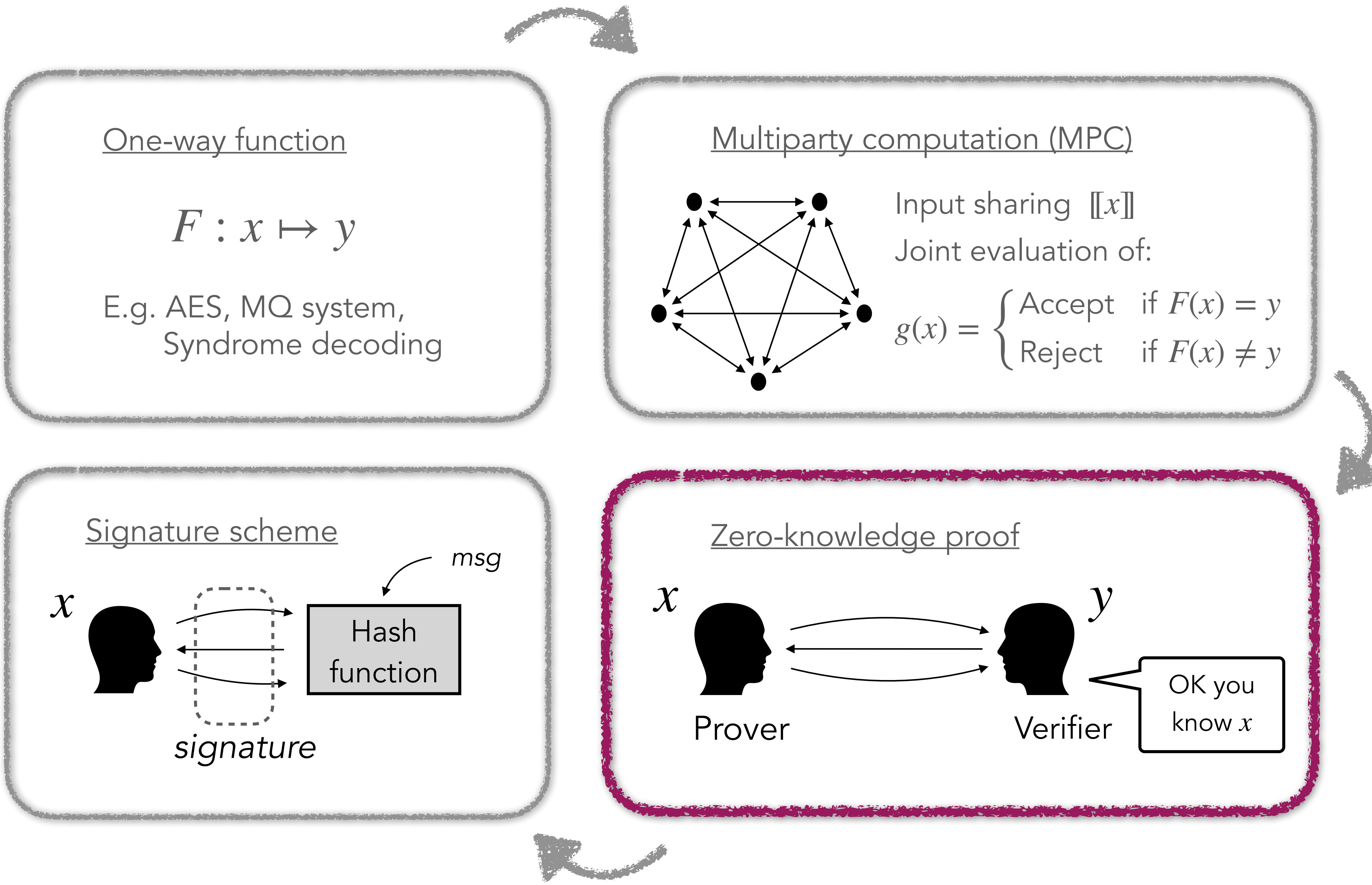
AIMer
Biscuit
FAEST
MIRA
MiRitH
MQOM
PERK
RYDE
SDitH



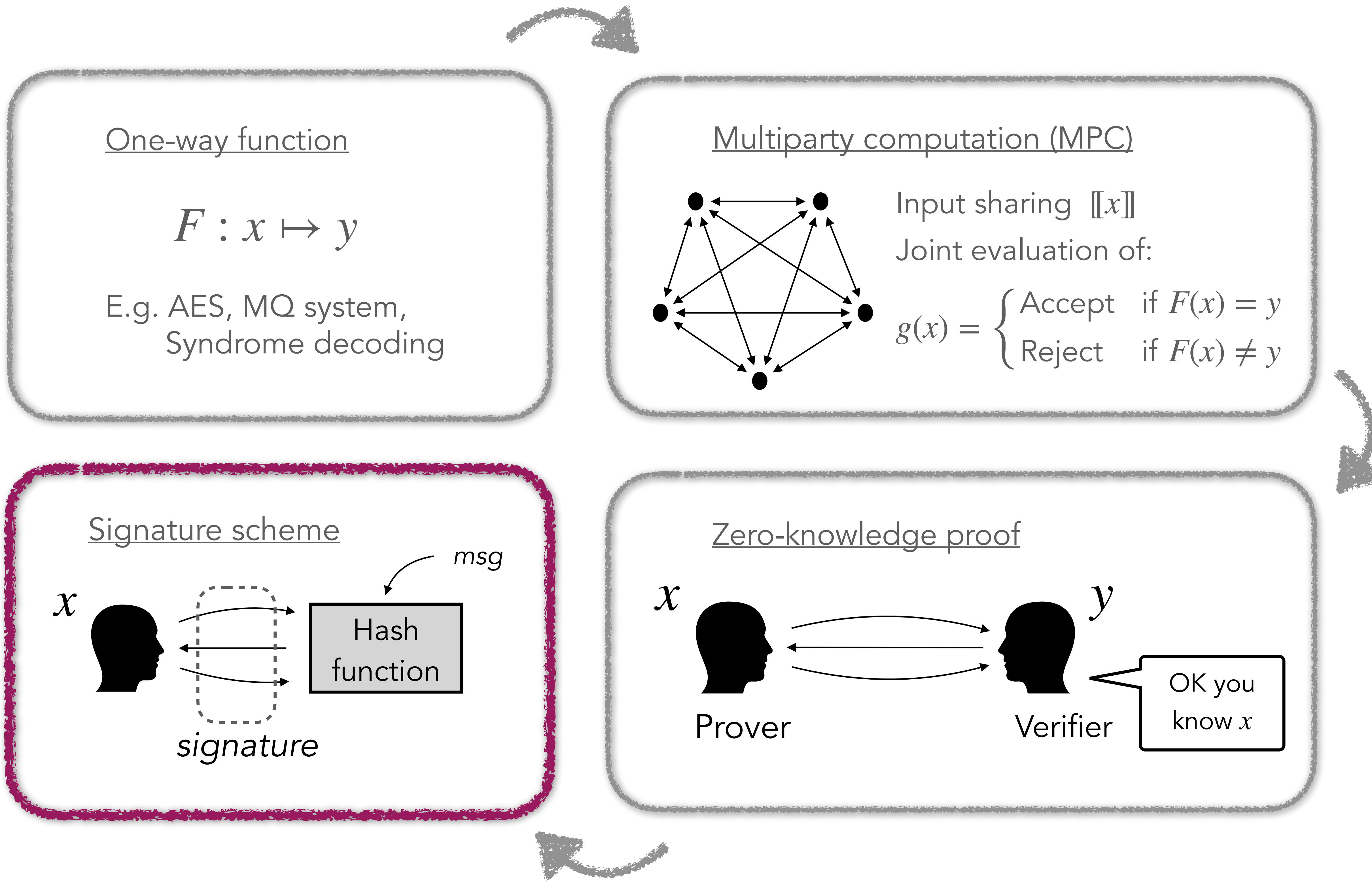
One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
msg
x
Hash
function
signature
Zero-knowledge proof
x
y
Prover
Verifier
OK you
know x

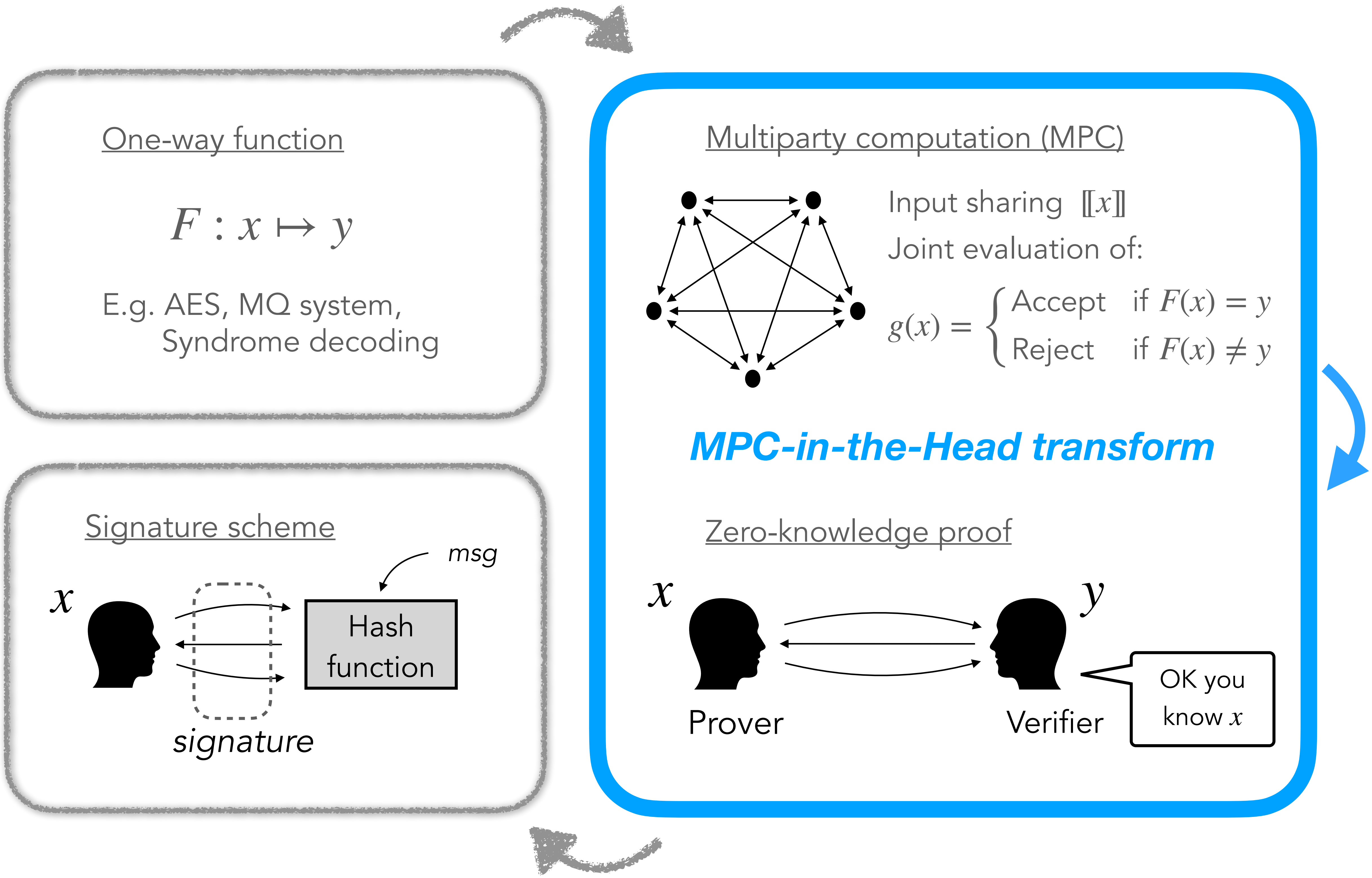

One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Zero-knowledge proof
x
y
Prover
Verifier
OK you know x
Signature scheme
x
msg
Hash function
signature


One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
x
msg
Hash function
signature
Zero-knowledge proof
x
y
Prover
Verifier
OK you know x



One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
x
msg
Hash function
signature
Zero-knowledge proof
x
y
Prover
Verifier
OK you know x

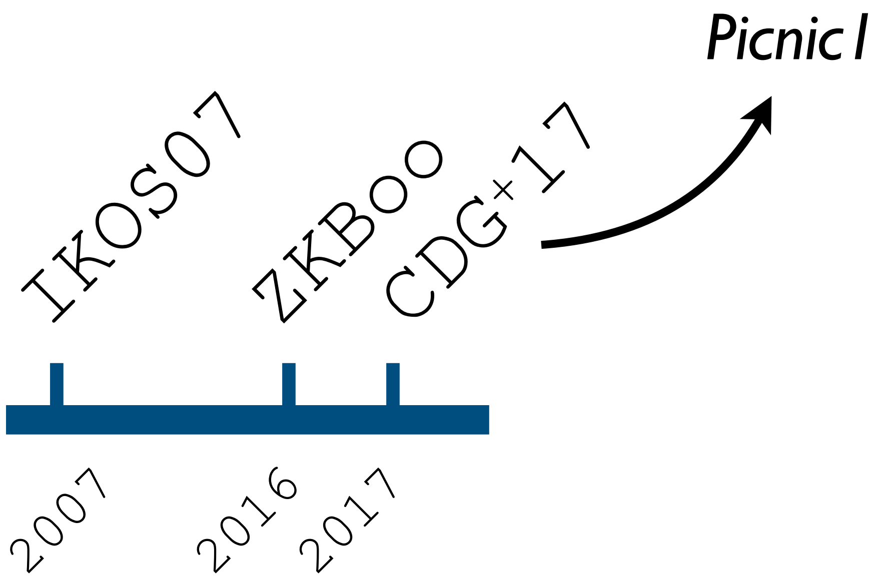

One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
x
msg
Hash function
signature
Zero-knowledge proof
x
y
Prover
Verifier
OK you know x



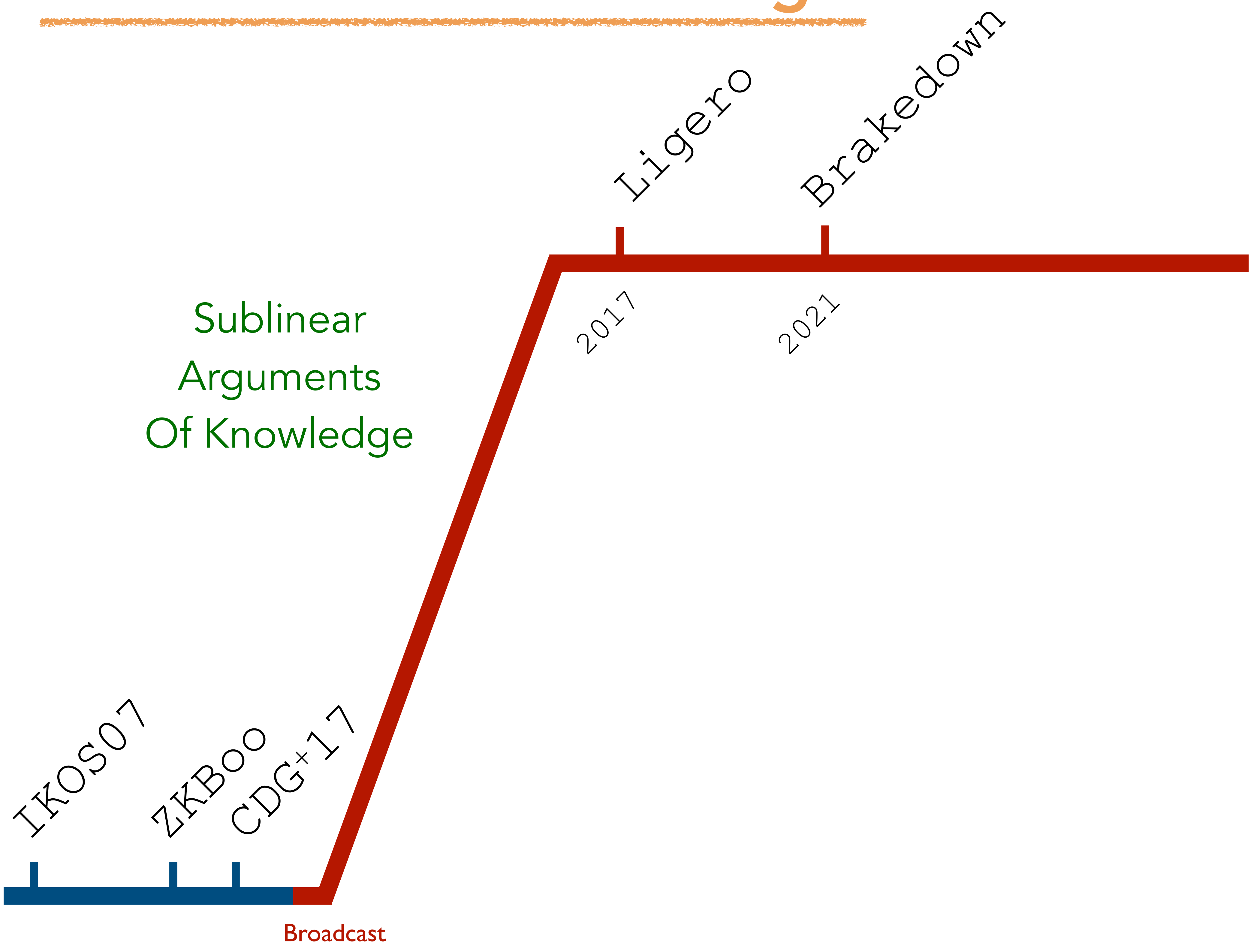
One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
MPC-in-the-Head transform
Zero-knowledge proof
x
y
Prover
Verifier
OK you know x
Signature scheme
x
msg
Hash function
signature

MPCitH for signature schemes

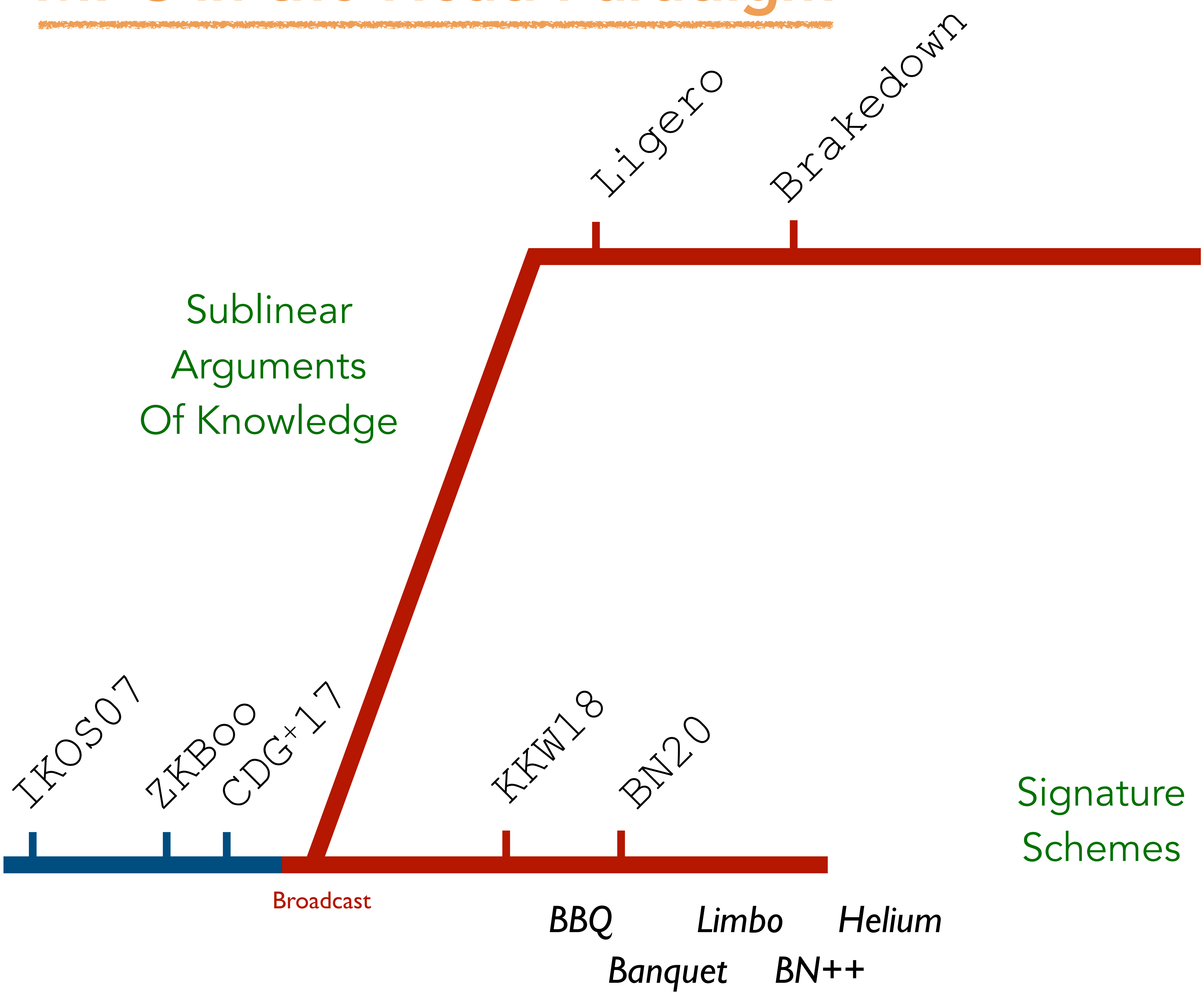
MPC-in-the-Head Paradigm



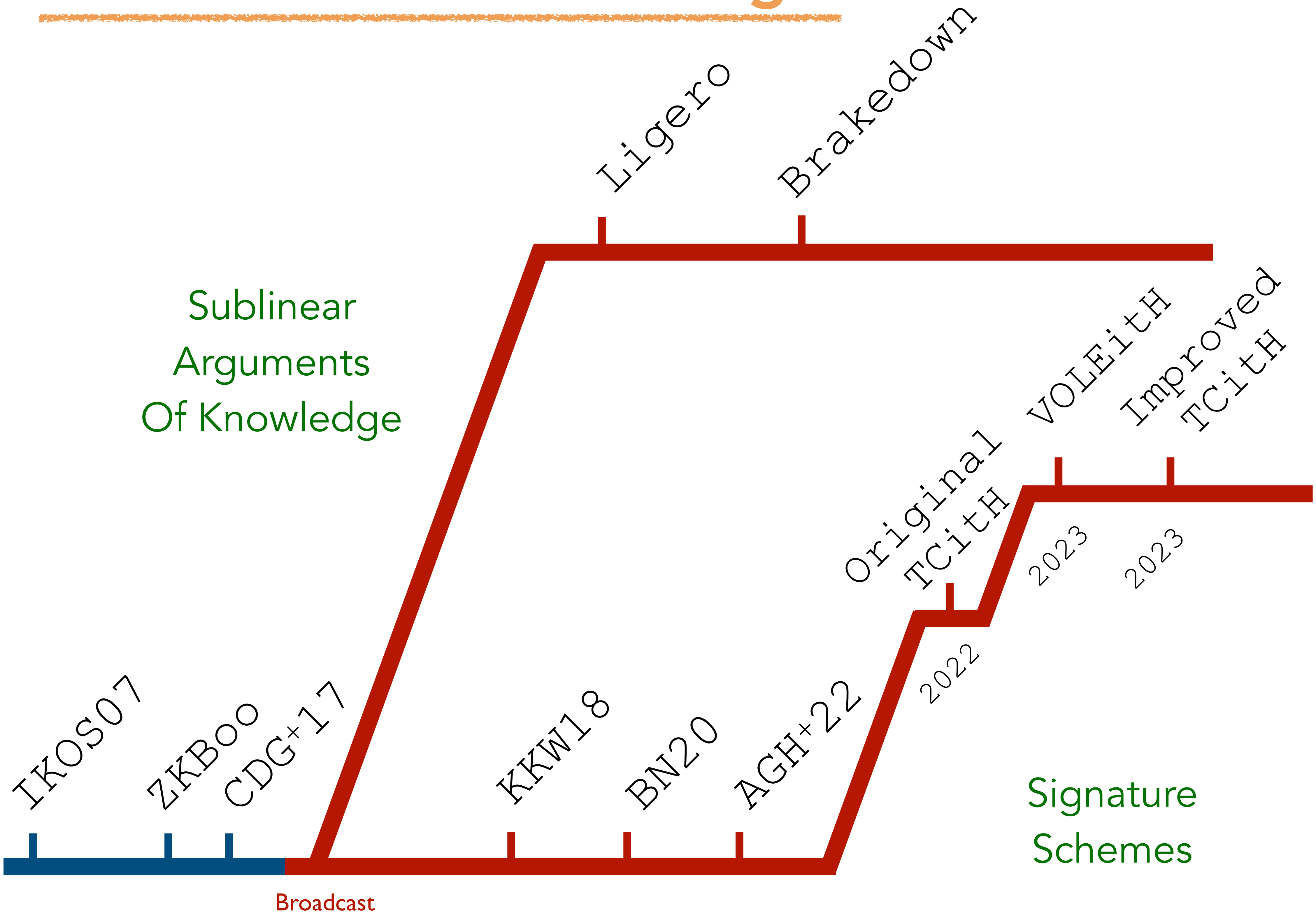
MPC-in-the-Head Paradigm



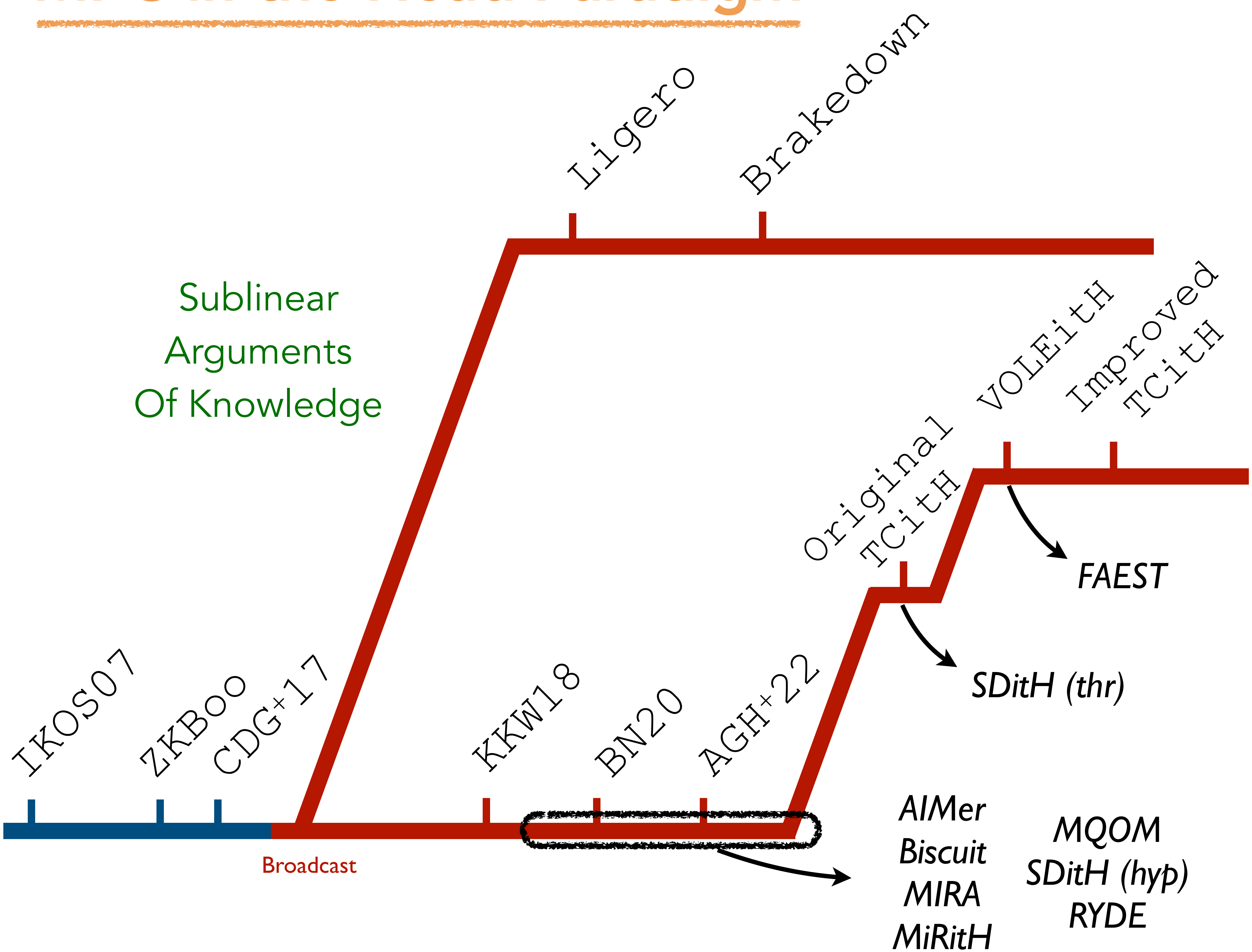
MPC-in-the-Head Paradigm



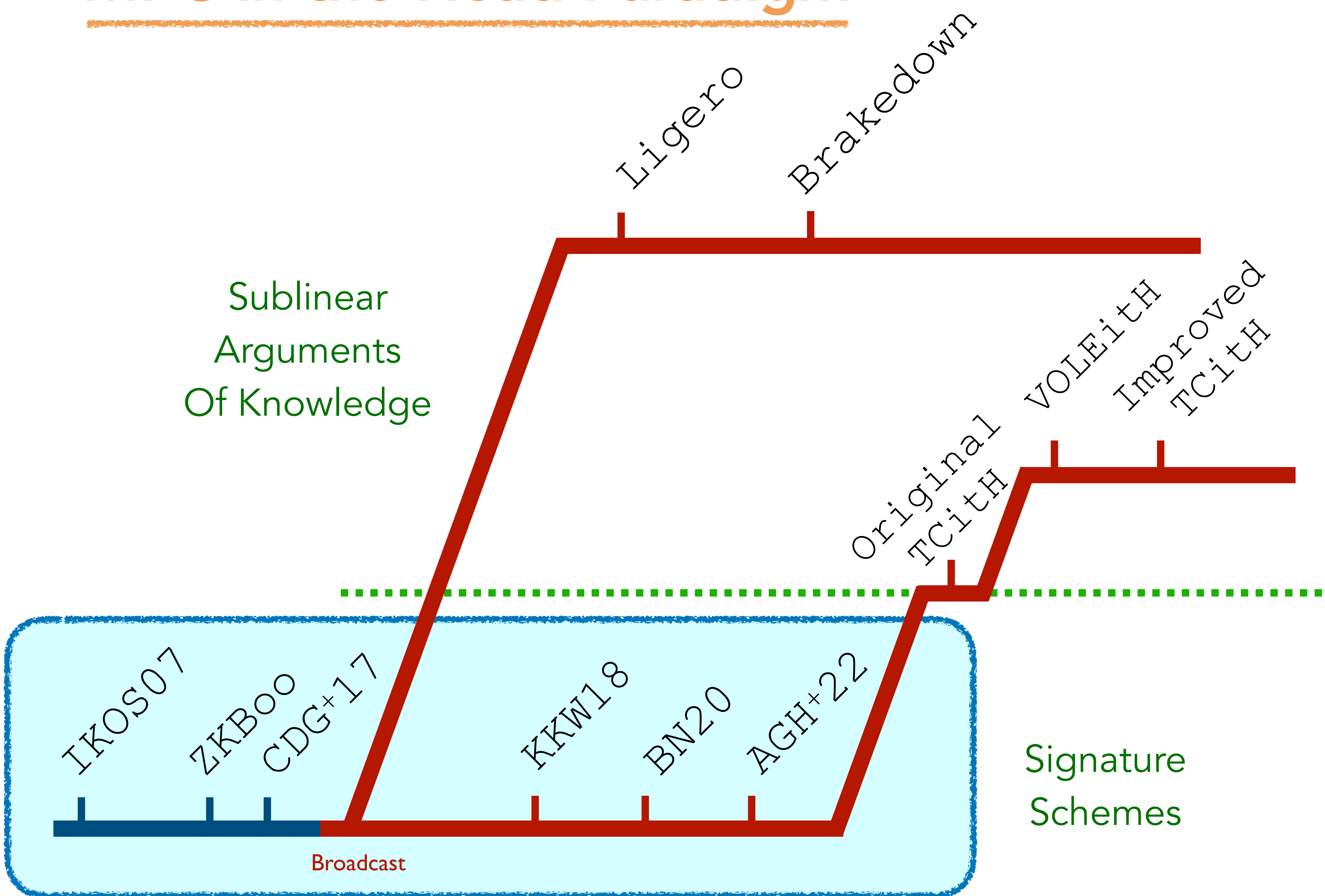
MPC-in-the-Head Paradigm



MPC-in-the-Head Paradigm

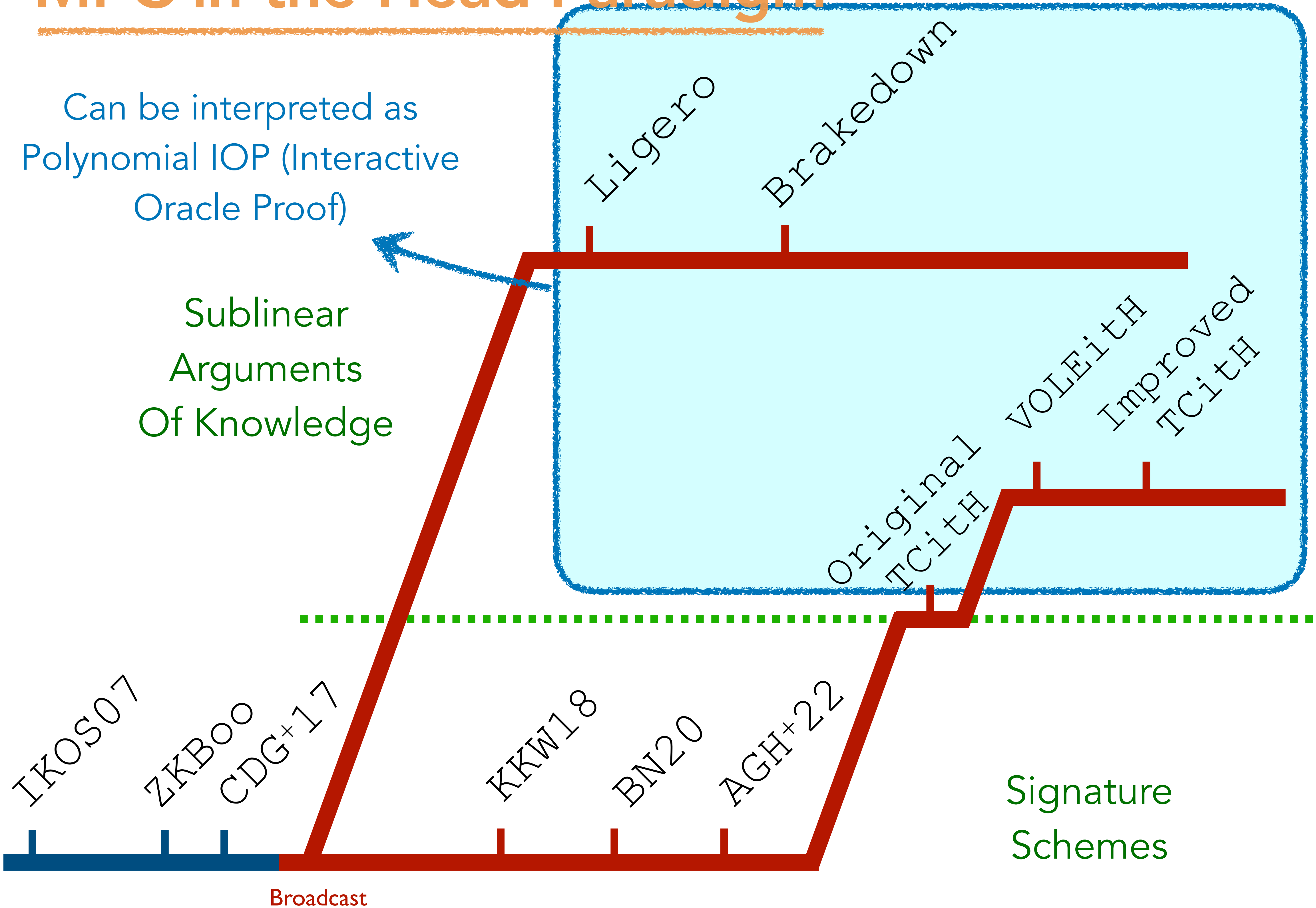


MPC-in-the-Head Paradigm



Rely on MPC techniques : GMW87, Beaver triples, …

MPC-in-the-Head Paradigm



TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

$$f(w_1, \ldots, w_n) = 0$$

where f is a public **degree-d polynomial**.

Prove it!

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

Prover

Verifier

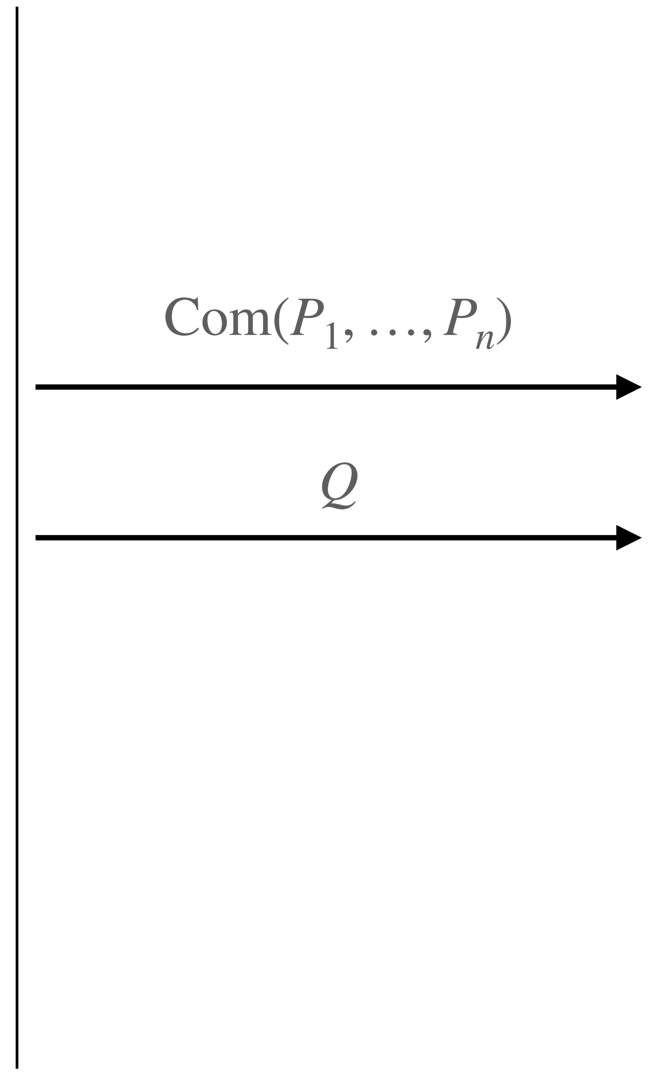
TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

③ Reveal the polynomial $Q(X)$ such that
$X \cdot Q(X) = f(P_1(X), \ldots, P_n(X))$



Prover

Verifier

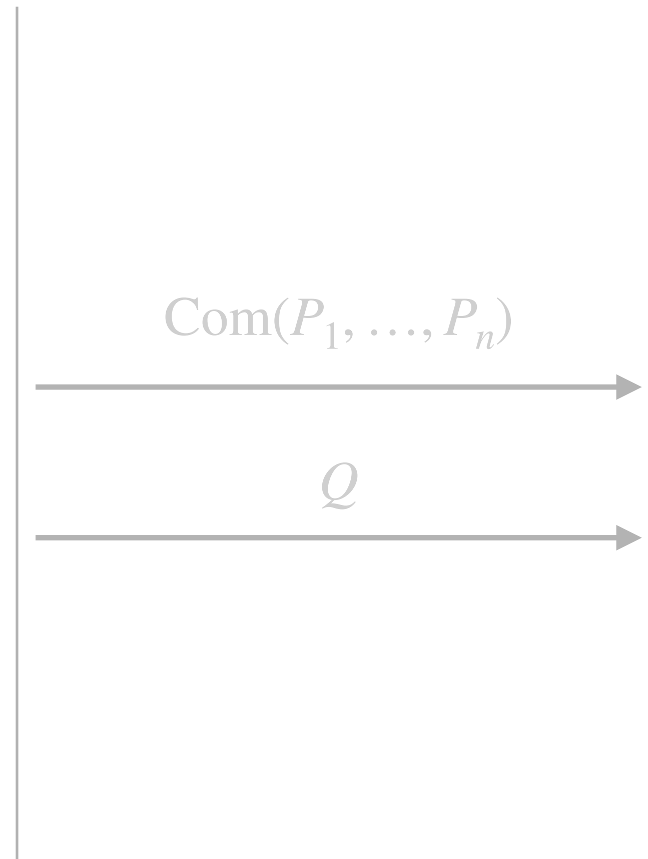
TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

③ Reveal the polynomial $Q(X)$ such that $X \cdot Q(X) = f(P_1(X), \ldots, P_n(X))$



Well-defined!

$$f(P_1(0), \ldots, P_n(0)) = f(w_1, \ldots, w_n) = 0$$

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

③ Reveal the polynomial $Q(X)$ such that $X \cdot Q(X) = f(P_1(X), \ldots, P_n(X))$

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

⑤ Reveal the evaluation $v_i := P_i(r)$ for all i.

Messages (Prover ↔ Verifier):

- Prover → Verifier: $\mathrm{Com}(P_1, \ldots, P_n)$
- Prover → Verifier: Q
- Verifier → Prover: r
- Prover → Verifier: $v_1, \ldots, v_n$

Prover | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

Prover → Verifier: $\mathrm{Com}(P_1, \ldots, P_n)$

③ Reveal the polynomial $Q(X)$ such that
$$X \cdot Q(X) = f(P_1(X), \ldots, P_n(X))$$

Prover → Verifier: Q

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

Verifier → Prover: r

⑤ Reveal the evaluation $v_i := P_i(r)$ for all i.

Prover → Verifier: $v_1, \ldots, v_n$

⑥ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that
$$r \cdot Q(r) = f(v_1, \ldots, v_n)$$

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

Malicious Prover 😈		Verifier
① For all i, choose a degree-ℓ polynomial $P_i(X)$. We have $f(P_1(0), \ldots, P_n(0)) \neq 0$.		
② Commit the polynomials $P_1, \ldots, P_n$	$\mathrm{Com}(P_1, \ldots, P_n)$ →	
③ Reveal the polynomial $Q(X)$. We know that $X \cdot Q(X) \neq f(P_1(X), \ldots, P_n(X))$	Q →	
	← r	④ Choose a random evaluation point $r \in S \subset \mathbb{F}$
⑤ Reveal the evaluation $v_i := P_i(r)$ for all i.	$v_1, \ldots, v_n$ →	
		⑥ Check that $v_1, \ldots, v_n$ are consistent with the commitment. Check that $r \cdot Q(r) = f(v_1, \ldots, v_n)$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

① For all i, choose a degree-ℓ polynomial $P_i(X)$. We have
$$f(P_1(0), \ldots, P_n(0)) \neq 0.$$

② Commit the polynomials $P_1, \ldots, P_n$ → $\mathrm{Com}(P_1, \ldots, P_n)$

③ Reveal the polynomial $Q(X)$. We know that
$$X \cdot Q(X) \neq f(P_1(X), \ldots, P_n(X))$$
→ Q

④ Choose a random evaluation point $r \in S \subset \mathbb{F}$ ← r

⑤ Reveal the evaluation $v_i := P_i(r)$ for all i. → $v_1, \ldots, v_n$

⑥ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that
$$r \cdot Q(r) = f(v_1, \ldots, v_n)$$

Malicious Prover 😈 | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

① For all i, choose a degree-ℓ polynomial $P_i(X)$. We have

$$f(P_1(0), \ldots, P_n(0)) \neq 0.$$

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Reveal the polynomial $Q(X)$. We know that

$$X \cdot Q(X) \neq f(P_1(X), \ldots, P_n(X))$$

Q →

← r

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

⑥ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q(r) = f(v_1, \ldots, v_n)$$

Schwartz-Zippel Lemma: Let P be a *non-zero polynomial of degree* μ. We have

$$\Pr\left[P(r) = 0 \mid r \leftarrow_\$ S\right] \leq \frac{\mu}{|S|}.$$

Since $X \cdot Q(X) - f(P_1(X), \ldots, P_n(X))$ is a degree-$(d \cdot \ell)$ polynomial, we have

$$\Pr[\text{verification passes}] \leq \frac{d \cdot \ell}{|S|}.$$

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

$$f(w_1, \ldots, w_n) = 0$$

where f is a public **degree-d polynomial**.

Prove it!

Prover

Verifier

Soundness Error = $\frac{d \cdot \ell}{|S|}$

Probability that a malicious prover can convince the verifier.

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

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

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

Prove it!

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\text{Com}(P_1, \ldots, P_n)$ →

③ Reveal the polynomials $Q_1(X), \ldots, Q_m(X)$ such that

$$X \cdot Q_1(X) = f_1(P_1(X), \ldots, P_n(X))$$
$$\vdots$$
$$X \cdot Q_m(X) = f_m(P_1(X), \ldots, P_n(X))$$

$Q_1, \ldots, Q_m$ →

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

← r

⑤ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

⑥ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q_1(r) = f_1(v_1, \ldots, v_n)$$
$$\cdots$$
$$r \cdot Q_m(r) = f_m(v_1, \ldots, v_n)$$

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Prover

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

② Commit the polynomials $P_1, \ldots, P_n$

$\text{Com}(P_1, \ldots, P_n)$ →

④ Reveal the polynomials $Q_1(X), \ldots, Q_m(X)$ such that

$$X \cdot Q_1(X) = f_1(P_1(X), \ldots, P_n(X))$$
$$\vdots$$
$$X \cdot Q_m(X) = f_m(P_1(X), \ldots, P_n(X))$$

$Q_1, \ldots, Q_m$ →

Costly! 😰

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q_1(r) = f_1(v_1, \ldots, v_n)$$
$$\cdots$$
$$r \cdot Q_m(r) = f_m(v_1, \ldots, v_n)$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \dots, P_n$

$\mathrm{Com}(P_1, \dots, P_n)$ →

④ Reveal the polynomials $Q_1(X), \dots, Q_m(X)$ such that

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

$Q_1, \dots, Q_m$ →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \dots, v_n$ →

⑦ Check that $v_1, \dots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q_1(r) = f_1(v_1, \dots, v_n)$$
$$\dots$$
$$r \cdot Q_m(r) = f_m(v_1, \dots, v_n)$$

Costly! 😰

Solution: batching

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\text{Com}(P_1, \ldots, P_n)$ →

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

Well-defined!

$$\sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(0), \ldots, P_n(0)) = \sum_{j=1}^{m} \gamma_j \cdot f_j(w_1, \ldots, w_n)$$

$$= \sum_{j=1}^{m} \gamma_j \cdot 0 = 0$$

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

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

② Commit the polynomials $P_1, \ldots, P_n$

$\text{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

Prover | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Prover / **Verifier**

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

② Commit the polynomials $P_1, \ldots, P_n$

Prover → Verifier: $\mathrm{Com}(P_1, \ldots, P_n)$

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

Verifier → Prover: $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Prover → Verifier: Q

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

Verifier → Prover: r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

Prover → Verifier: $v_1, \ldots, v_n$

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment. Check that

$$r \cdot Q(r) = \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

① For all i, choose a degree-ℓ polynomial $P_i(X)$. There exists j^* s.t.

$$f_{j^*}(P_1(0), \ldots, P_n(0)) \neq 0.$$

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$. We know that

$$X \cdot Q(X) \neq \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.
Check that

$$r \cdot Q(r) = \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

Malicious Prover 😈 | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

① For all i, choose a degree-ℓ polynomial $P_i(X)$. There exists j^* s.t.

$$f_{j^*}(P_1(0), \dots, P_n(0)) \neq 0.$$

② Commit the polynomials $P_1, \dots, P_n$

$\text{Com}(P_1, \dots, P_n)$ →

③ Choose random coefficients $\gamma_1, \dots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \dots, \gamma_m$

④ Reveal the polynomial $Q(X)$. We know that

$$X \cdot Q(X) \neq \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \dots, P_n(X))$$

Q →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \dots, v_n$ →

⑦ Check that $v_1, \dots, v_n$ are consistent with the commitment.

It is an inequality with **high probability** over the randomness of $\gamma_1, \dots, \gamma_m$, since we have

$$\sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(0), \dots, P_n(0)) \neq 0$$

Malicious Prover 😈

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Soundness Analysis

① For all i, choose a degree-ℓ polynomial $P_i(X)$. There exists j^* s.t.

$$f_{j^*}(P_1(0), \ldots, P_n(0)) \neq 0.$$

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$. We know that

$$X \cdot Q(X) \neq \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q(r) = \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

Verifier

Schwartz-Zippel Lemma: Since it is a degree-$(d \cdot \ell)$ relation,

$$\Pr[\text{verification passes}] \leq \frac{d \cdot \ell}{|S|}.$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Zero-Knowledge Analysis

Prover

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

② Commit the polynomials $P_1, \ldots, P_n$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

Messages

- Prover → Verifier: $\mathrm{Com}(P_1, \ldots, P_n)$
- Verifier → Prover: $\gamma_1, \ldots, \gamma_m$
- Prover → Verifier: Q
- Verifier → Prover: r
- Prover → Verifier: $v_1, \ldots, v_n$

Verifier

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.
Check that

$$r \cdot Q(r) = \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Zero-Knowledge Analysis

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

② Commit the polynomials $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

Q →

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

$v_1, \ldots, v_n$ →

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.
Check that

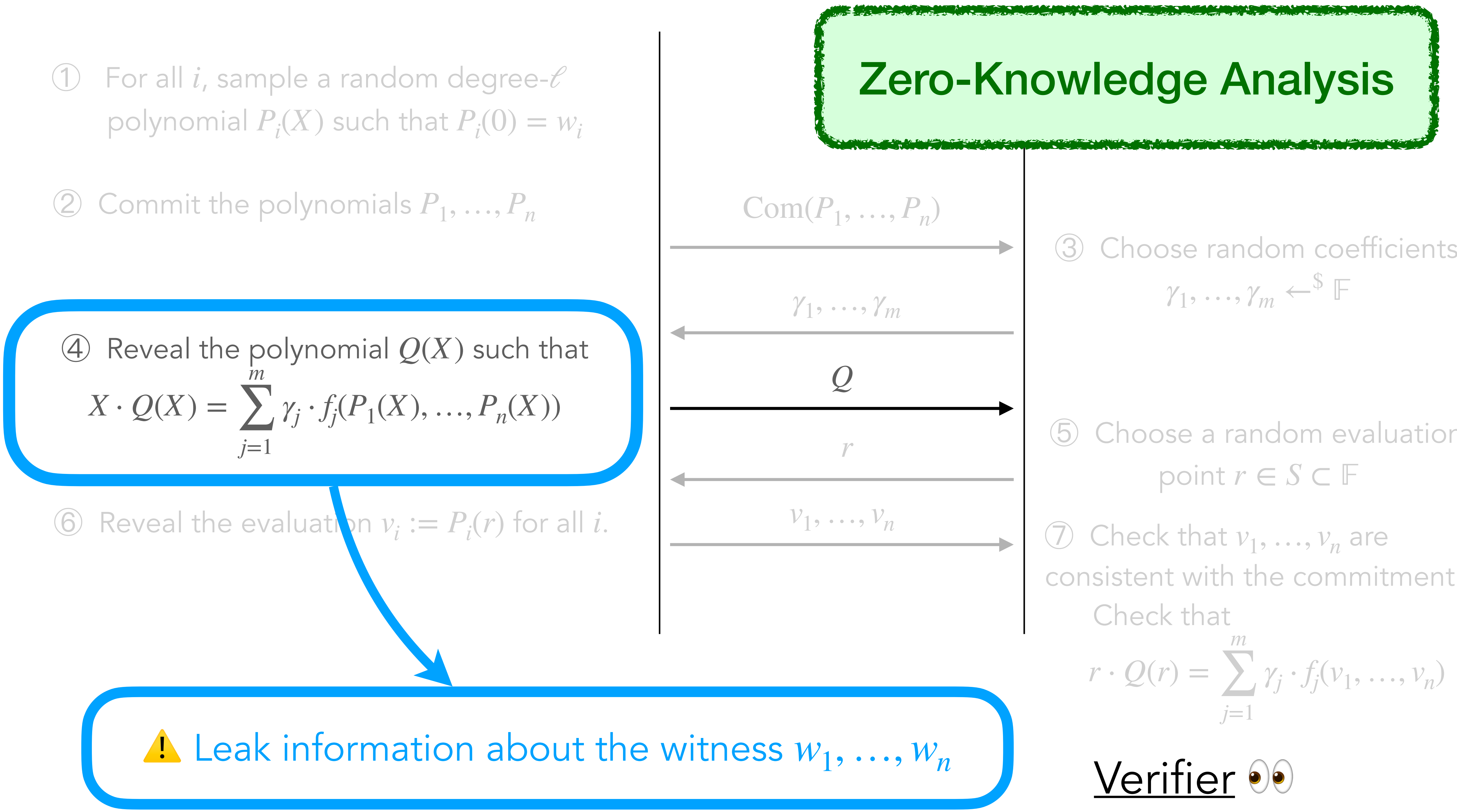
$$r \cdot Q(r) = \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

Verifier 👀

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

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)



TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Zero-Knowledge Analysis

Prover

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

Sample a random degree-$(d\ell - 1)$ polynomial $P_0(X)$

② Commit the polynomials $P_0, P_1, \ldots, P_n$

→ $\mathrm{Com}(P_0, P_1, \ldots, P_n)$

Verifier

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

← $\gamma_1, \ldots, \gamma_m$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

→ Q

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

← r

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

→ $v_0, v_1, \ldots, v_n$

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q(r) = r \cdot v_0 + \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

Prover

Verifier 👀

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Prover		Verifier
① For all i, sample a random degree-ℓ polynomial $P_i(X)$ such that $P_i(0) = w_i$ Sample a random degree-$(d\ell - 1)$ polynomial $P_0(X)$		
② Commit the polynomials $P_0, P_1, \ldots, P_n$	$\mathrm{Com}(P_0, P_1, \ldots, P_n)$ →	③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$
	← $\gamma_1, \ldots, \gamma_m$	
④ Reveal the polynomial $Q(X)$ such that $X \cdot Q(X) = X \cdot P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$	Q →	⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$
	← r	
⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.	$v_0, v_1, \ldots, v_n$ →	⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment. Check that $r \cdot Q(r) = r \cdot v_0 + \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

(for signature schemes)

Prover

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

Sample a random degree-$(d\ell - 1)$ polynomial $P_0(X)$

② Commit the polynomials $P_0, P_1, \ldots, P_n$

④ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j(P_1(X), \ldots, P_n(X))$$

⑥ Reveal the evaluation $v_i := P_i(r)$ for all i.

Messages

Prover → Verifier: $\mathrm{Com}(P_0, P_1, \ldots, P_n)$

Verifier → Prover: $\gamma_1, \ldots, \gamma_m$

Prover → Verifier: Q

Verifier → Prover: r

Prover → Verifier: $v_0, v_1, \ldots, v_n$

Verifier

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$

⑤ Choose a random evaluation point $r \in S \subset \mathbb{F}$

⑦ Check that $v_1, \ldots, v_n$ are consistent with the commitment.

Check that

$$r \cdot Q(r) = r \cdot v_0 + \sum_{j=1}^{m} \gamma_j \cdot f_j(v_1, \ldots, v_n)$$

<u>Prover</u> <u>Verifier</u>

How to commit to polynomials?

degree 10 000

degree 1000

degree 100

① VOLEitH / TCitH-GGM

degree 10

degree 1

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments.* ePrint 2023/1573.

[BBM+24] Baum, Beullens, Mukherjee, Orsini, Ramacher, Rechberger, Roy, Scholl. *One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures.* Asiacrypt 2024.

How to commit to polynomials?

degree 10 000

degree 1000

② Degree-enforcing commitment
(TCitH-MT)

degree 100

① VOLEitH / TCitH-GGM

degree 10

degree 1

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. ePrint 2023/1573.

How to commit to polynomials?

③ Merkle Trees with Ligero-like Proximity Tests

degree 10 000

degree 1000

② Degree-enforcing commitment (TCitH-MT)

degree 100

① VOLEitH / TCitH-GGM

degree 10

degree 1

[AHIV17] Ames, Hazay, Ishai, Venkitasubramaniam. *Ligero: Lightweight Sublinear Arguments Without a Trusted Setup*. CCS 2017.

[GLS+23] Golonew, Lee, Setty, Thaler, Wahby. *Brakedown: Linear-time and field agnotic SNARKs for R1CS*. Crypto 2023.

How to commit to polynomials?

④ FRI-based commitments

degree 10 000

③ Merkle Trees with
Ligero-like Proximity Tests

degree 1000

② Degree-enforcing commitment
(TCitH-MT)

degree 100

① VOLEitH / TCitH-GGM

degree 10

degree 1

[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?

④ FRI-based commitments

degree 10 000

③ Merkle Trees with
Ligero-like Proximity Tests

degree 1000

② Degree-enforcing commitment
(TCitH-MT)

degree 100

① VOLEitH / TCitH-GGM

degree 10

degree 1

For signature schemes, we use
degree-1 polynomials most of the time.

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-1 polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-1 polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1,\ldots,N\}$.

We want to build and commit a **random degree-1 polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Commitment:
We commit to each value r_i ***independently***.

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-**1 **polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Commitment:
We commit to each value r_i ***independently***.

Opening $P(e_{i^*})$:
Reveal all $\{r_i\}_{i \neq i^*}$.

$$P(e_{i^*}) = \sum_{i \neq i^*} r_i \cdot R_i(e_{i^*}) + r_{i^*} \cdot \underbrace{R_{i^*}(e_{i^*})}_{=0}$$

$$= \sum_{i \neq i^*} r_i \cdot R_i(e_{i^*})$$

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-1 polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Commitment:
We commit to each value r_i ***independently***.

Opening $P(e_{i*})$:
Reveal all $\{r_i\}_{i \neq i*}$.

$$P(e_{i*}) = \sum_{i \neq i*} r_i \cdot R_i(e_{i*}) + r_{i*} \cdot \underbrace{R_{i*}(e_{i*})}_{=0}$$

$$= \sum_{i \neq i*} r_i \cdot R_i(e_{i*})$$

The opening leaks *nothing* about P, except $P(e_{i*})$.

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-1 polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Commitment:
We commit to each value r_i ***independently***.

Opening $P(e_{i*})$:
Reveal all $\{r_i\}_{i \neq i*}$.

$$P(e_{i*}) = \sum_{i \neq i*} r_i \cdot R_i(e_{i*}) + r_{i*} \cdot \underbrace{R_{i*}(e_{i*})}_{=0}$$

$$= \sum_{i \neq i*} r_i \cdot R_i(e_{i*})$$

The opening leaks *nothing* about P, except $P(e_{i*})$.

🛠️ Can be adapted to any degree.

Committing to a Polynomial using a Seed Tree

Public data: Let us

- have N distinct values $e_1, \ldots, e_N$, and
- define R_i such that $R_i(0) = 1$ and $R_i(e_i) = 0$, for all i in $\{1, \ldots, N\}$.

We want to build and commit a **random degree-**1 **polynomial** P. We sample N values $r_1, \ldots, r_N$ and define P as

$$P := \sum_i r_i \cdot R_i.$$

Costly! 😰

Correctness:
If $N \geq 2$, P is a random degree-1 polynomial.

Commitment:
We commit to each value r_i ***independently***.

Opening $P(e_{i*})$:
Reveal all $\{r_i\}_{i \neq i*}$.

$$P(e_{i*}) = \sum_{i \neq i*} r_i \cdot R_i(e_{i*}) + r_{i*} \cdot \underbrace{R_{i*}(e_{i*})}_{=0}$$

$$= \sum_{i \neq i*} r_i \cdot R_i(e_{i*})$$

The opening leaks *nothing* about P, except $P(e_{i*})$.

🛠 Can be adapted to any degree.

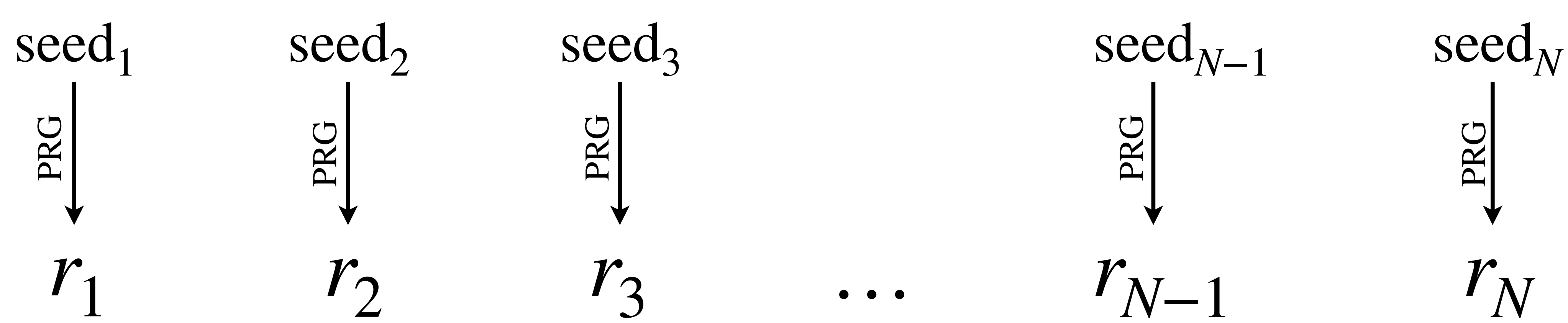
Committing to a Polynomial using a Seed Tree

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

$$r_1 \qquad r_2 \qquad r_3 \qquad \cdots \qquad r_{N-1} \qquad r_N$$

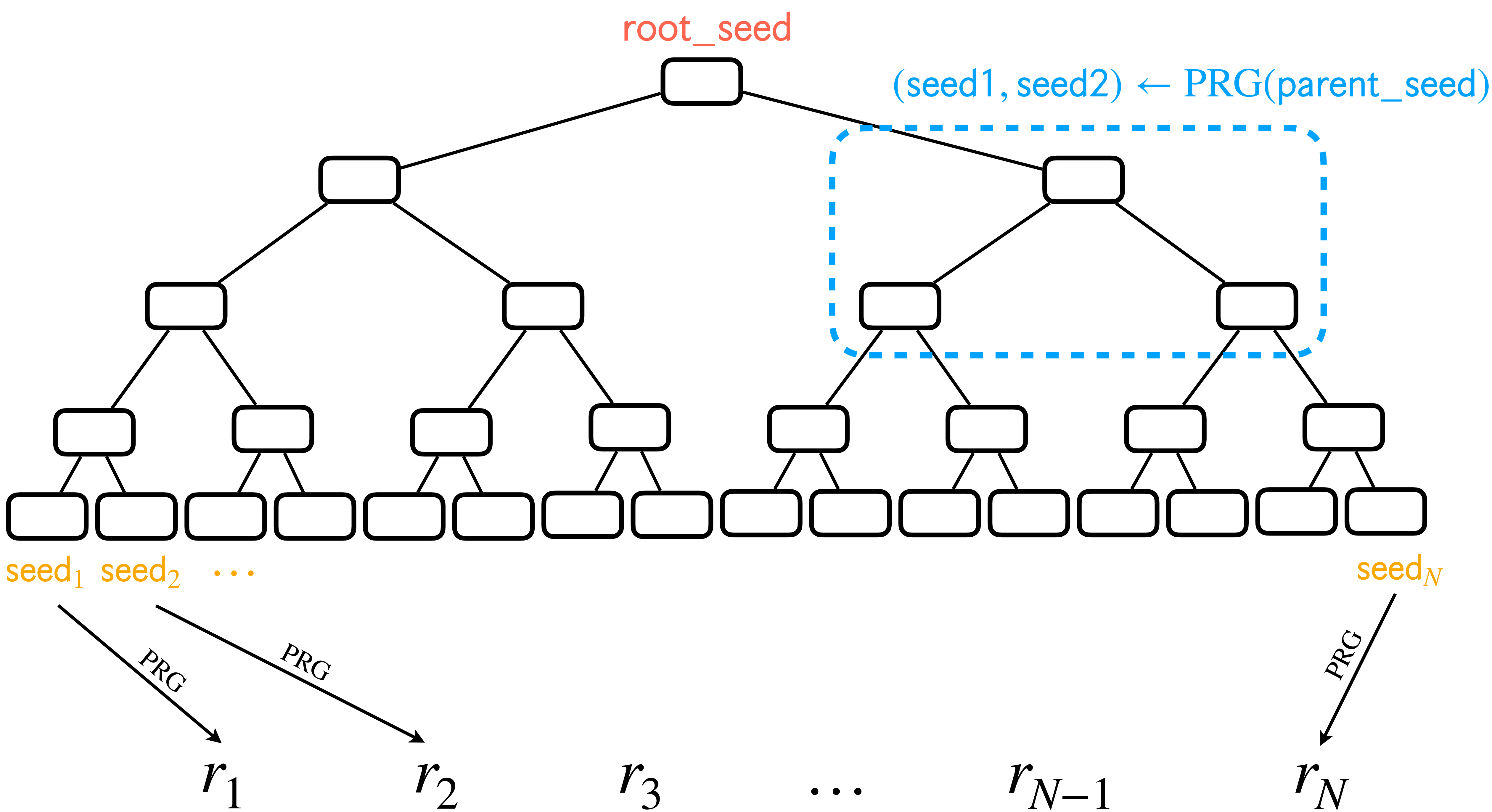
Committing to a Polynomial using a Seed Tree

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



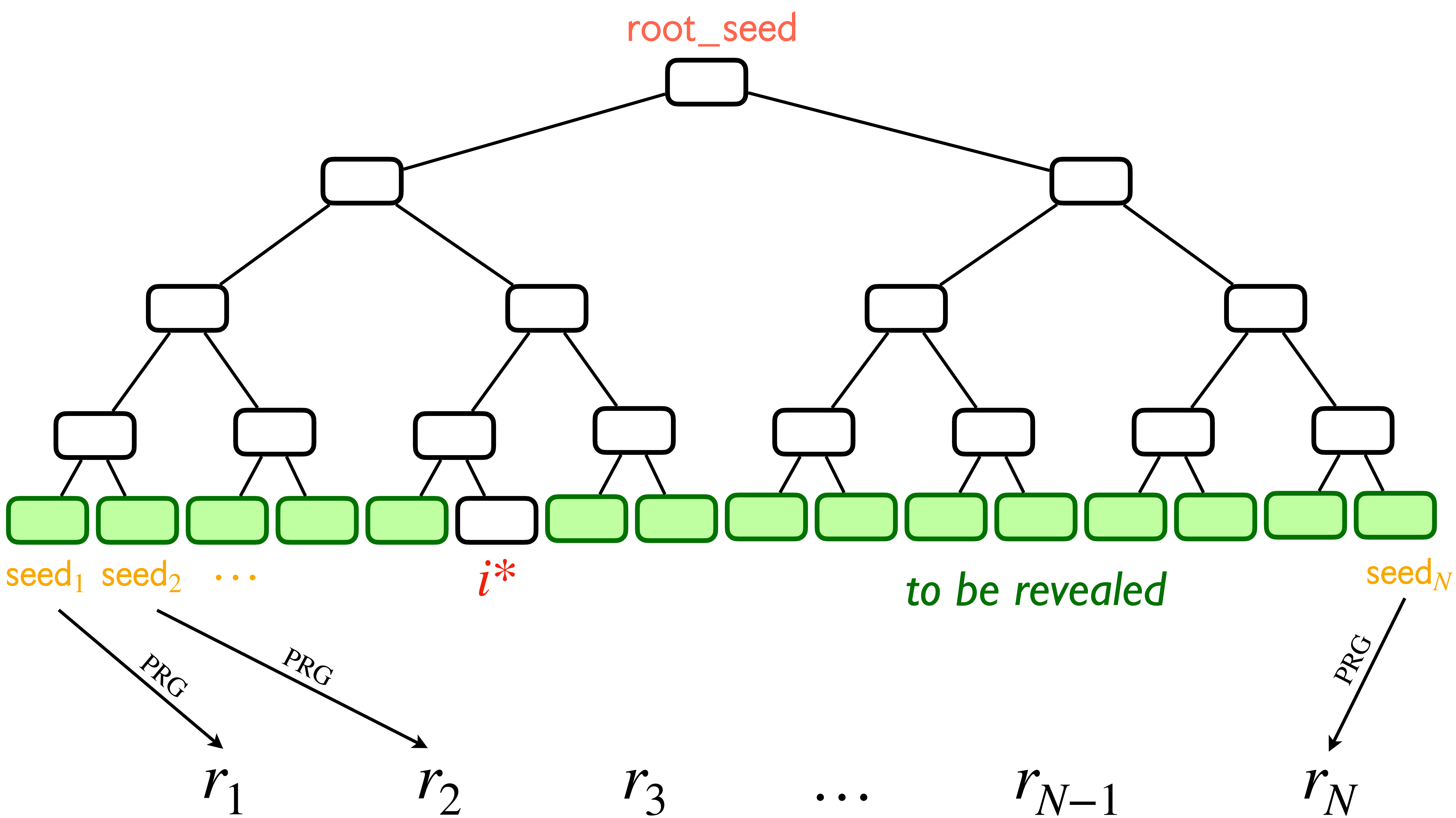
Committing to a Polynomial using a Seed Tree

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



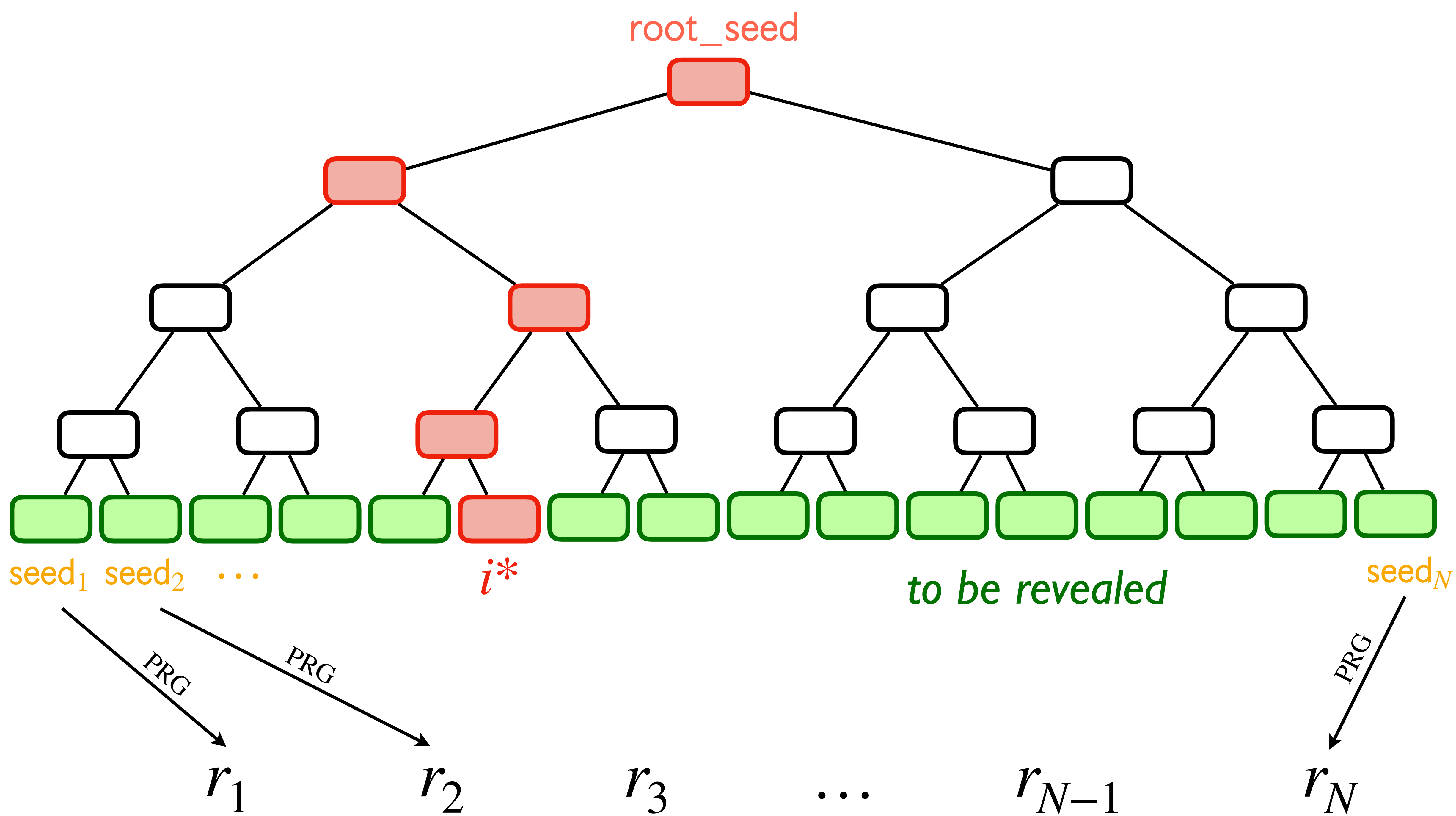
Committing to a Polynomial using a Seed Tree

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



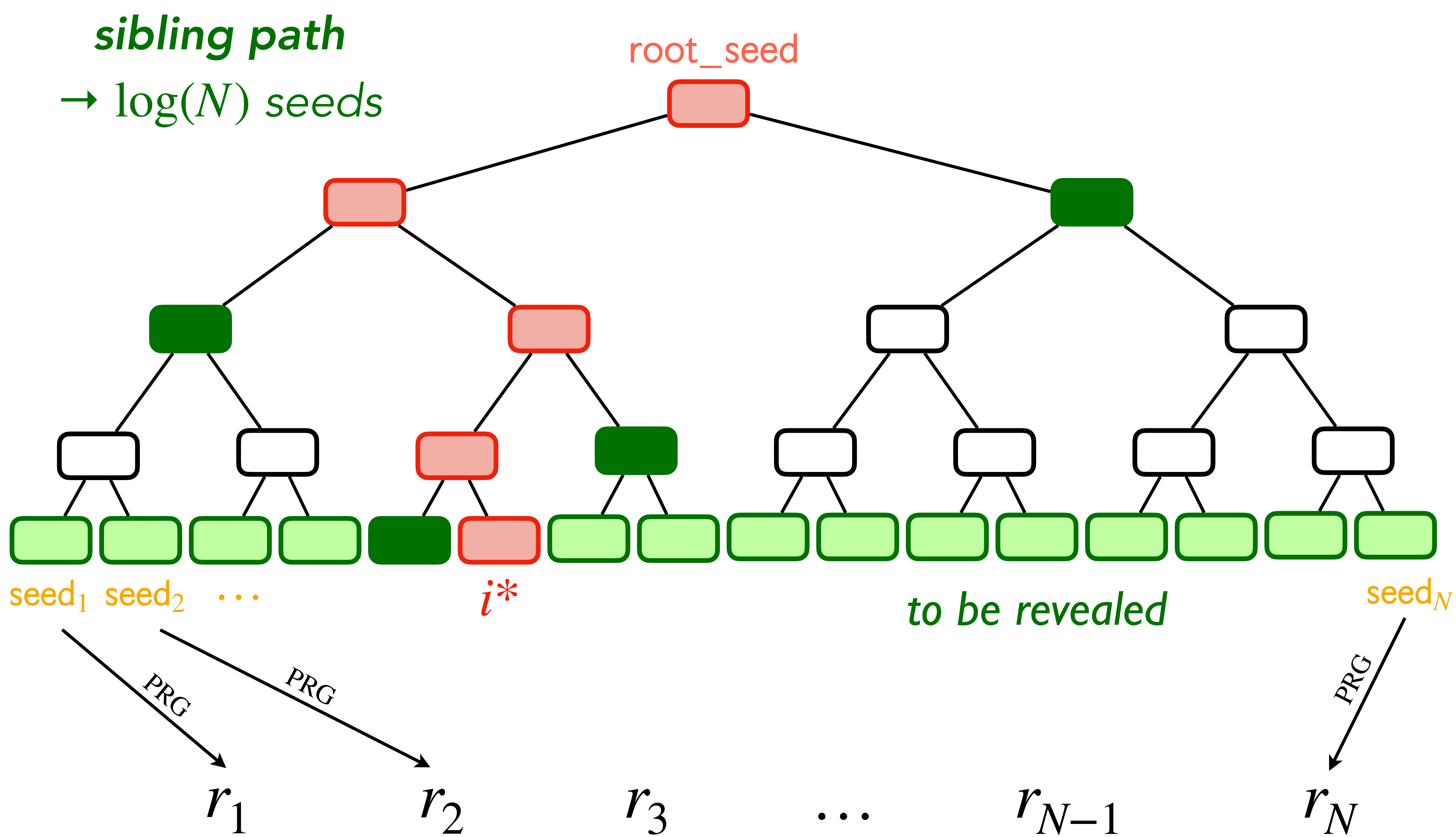
Committing to a Polynomial using a Seed Tree

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



Committing to a Polynomial using a Seed Tree

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



Committing to a Polynomial using a Seed Tree

We want to build and commit a random degree-1 polynomial P **such that** $P(0) = w$.

Committing to a Polynomial using a Seed Tree

We want to build and commit a random degree-1 polynomial P **such that** $P(0) = w$.

1. Sample and commit a random degree-1 polynomial $\tilde{P}$
2. Reveal $\Delta w := w - \tilde{P}(0)$
3. Define P as $P(X) := \tilde{P} + \Delta w$

Committing to a Polynomial using a Seed Tree

We want to build and commit a random degree-1 polynomial P **such that** $P(0) = w$.

1. Sample and commit a random degree-1 polynomial $\tilde{P}$
2. Reveal $\Delta w := w - \tilde{P}(0)$
3. Define P as $P(X) := \tilde{P} + \Delta w$

To open $P(e_{i^*})$ for $i^* \in \{1,\ldots,N\}$, we just need to open $\tilde{P}(e_{i^*})$ and to compute

$$P(e_{i^*}) \leftarrow \tilde{P}(e_{i^*}) + \Delta w.$$

Committing to a Polynomial using a Seed Tree

Complexity in $O(N)$ to have a soundness error of $\frac{d}{N}$ (degree-1 polynomials).

How to have a negligible soundness? 🤔

Committing to a Polynomial using a Seed Tree

Complexity in $O(N)$ to have a soundness error of $\frac{d}{N}$ (degree-1 polynomials).

How to have a negligible soundness? 🤔

1. Taking $N \geq 2^\lambda$. Impossible since the complexity would be in $O(2^\lambda)$.

Committing to a Polynomial using a Seed Tree

Complexity in $O(N)$ to have a soundness error of $\frac{d}{N}$ (degree-1 polynomials).

How to have a negligible soundness?

1. Taking $N \geq 2^\lambda$. Impossible since the complexity would be in $O(2^\lambda)$.

2. *TCitH-GGM Approach*. Taking N small (e.g. $N = 256$) and repeating the protocol τ times. Soundness error of $\left(\frac{d}{N}\right)^\tau$.

Committing to a Polynomial using a Seed Tree

Complexity in $O(N)$ to have a soundness error of $\frac{d}{N}$ (degree-1 polynomials).

How to have a negligible soundness?

1. Taking $N \geq 2^\lambda$. Impossible since the complexity would be in $O(2^\lambda)$.
2. *TCitH-GGM Approach*. Taking N small (e.g. $N = 256$) and repeating the protocol τ times. Soundness error of $\left(\frac{d}{N}\right)^\tau$.
3. *VOLEitH Approach*. Embed τ polynomials over $\mathbb{F}_q$ into a unique polynomial over $\mathbb{F}_{q^\tau}$, for which we will be able to open N^τ evaluations. Soundness error of $\frac{d}{N^\tau}$.

Building Signatures

Building Signatures

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

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

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

Prove it!

Prover

Verifier

Building Signatures

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

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

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

Prover

Prove it!

Verifier

Fiat-Shamir Transformation

Signature Scheme

Building Signature Schemes

The **public key** is composed of the **degree-d polynomials** $f_1, \ldots, f_m$.

The **private key** is the **witness** $w := (w_1, \ldots, w_n)$ that satisfies

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

Building Signature Schemes

The **public key** is composed of the **degree-d polynomials** $f_1, \ldots, f_m$.

The **private key** is the **witness** $w := (w_1, \ldots, w_n)$ that satisfies

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

When $f_1, \ldots, f_n$ are random degree-2 polynomials,

Signature relying on the Multivariate Quadratic (MQ) problem

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. ePrint 2023/1573.

[BBM+24] Baum, Beullens, Mukherjee, Orsini, Ramacher, Rechberger, Roy, Scholl. *One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures*. Asiacrypt 2024.

Building Signature Schemes

Proving that the private key $(x_1, \dots, x_{n'}, q_0, \dots, q_{t-1})$ satisfies

$$\begin{cases} y - Hx & = 0, \\ x_1 \cdot Q(1) & = 0 \\ & \vdots \\ x_n \cdot Q(n) & = 0. \end{cases}$$

Imply that $\mathrm{wt}_H(x) \leq t$.

with $x := (x_1, \dots, x_{n'})$ and $Q(X) := X^t + \sum_{i=0}^{t-1} q_i \cdot X^i$, where (H, y) is the public key.

Signature relying on the Syndrome Decoding (SD) problem

[FJR23] Feneuil, Joux, Rivain. *Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs*. Crypto 2022.

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. ePrint 2023/1573.

Building Signature Schemes

Proving that the private key $(L, R) \in \mathbb{F}^{n \times r} \times \mathbb{F}^{r \times m}$ satisfies

$$y - Hx = 0 \ \text{ with } \ x = \text{vectorialize}(L \cdot R)$$

where (H, y) is the public key.

Signature relying on the MinRank problem

[BFG+24] Bidoux, Feneuil, Gaborit, Neveu, Rivain. *Dual Support Decomposition in the Head: Shorter Signatures from Rank SD and MinRank*. Asiacrypt 2024.

Signature Sizes with the New Frameworks

	NIST Submission		***New frameworks + Opt.****
Security Assumptions	***Candidate Name***	***Sizes***	***Sizes***
AES Block cipher	FAEST	4.6 KB	≈ 4.1-4.5 KB
AIM Block cipher	AIMer	3.8 KB	≈ 3.0 KB
MinRank	MiRitH, MIRA	5.6 KB	≈ 2.9-3.1 KB
Multivariate Quadratic	MQOM	6.3 KB	≈ 2.5-3.0 KB
Permuted Kernel	PERK	5.8 KB	≈ 3.8 KB
Rank Syndrome Decoding	RYDE	6.0 KB	≈ 2.9 KB
Structured MQ	Biscuit	5.7 KB	≈ 3.0 KB
Syndrome Decoding	SDitH	8.3 KB	≈ 5.0 KB

Running times of few ten millions of cycles.

* **[BBM+24]** Baum, Beullens, Mukherjee, Orsini, Ramacher, Rechberger, Roy, Scholl. *One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures.* Asiacrypt 2024.

Conclusion

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any one-way function
  - A practical tool to build *conservative* signature schemes
    - *No structure* in the security assumption

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any one-way function
  - A practical tool to build *conservative* signature schemes
    - *No structure* in the security assumption
  - A lot of improvements since 2016
  - Latest frameworks: VOLEitH and TCitH
    - Can be interpreted as Polynomial IOP (Interactive Oracle Proof)

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any one-way function
  - A practical tool to build *conservative* signature schemes
    - *No structure* in the security assumption
  - A lot of improvements since 2016
  - Latest frameworks: VOLEitH and TCitH
    - Can be interpreted as Polynomial IOP (Interactive Oracle Proof)
- Perspectives: *Signatures with advanced functionalities*

ring signatures, threshold signatures,

blind signatures, multi-signatures, …

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any one-way function
  - A practical tool to build *conservative* signature schemes
    - *No structure* in the security assumption
  - A lot of improvements since 2016
  - Latest frameworks: VOLEitH and TCitH
    - Can be interpreted as Polynomial IOP (Interactive Oracle Proof)

- Perspectives: *Signatures with advanced functionalities*

  ring signatures, threshold signatures,

  blind signatures, multi-signatures, …

Thank you for your attention.