Post-Quantum Signatures from Secure Multiparty Computation

Thibauld Feneuil

ReAdPQC — CIFRIS24

September 27, 2024, Frascati (Rome)

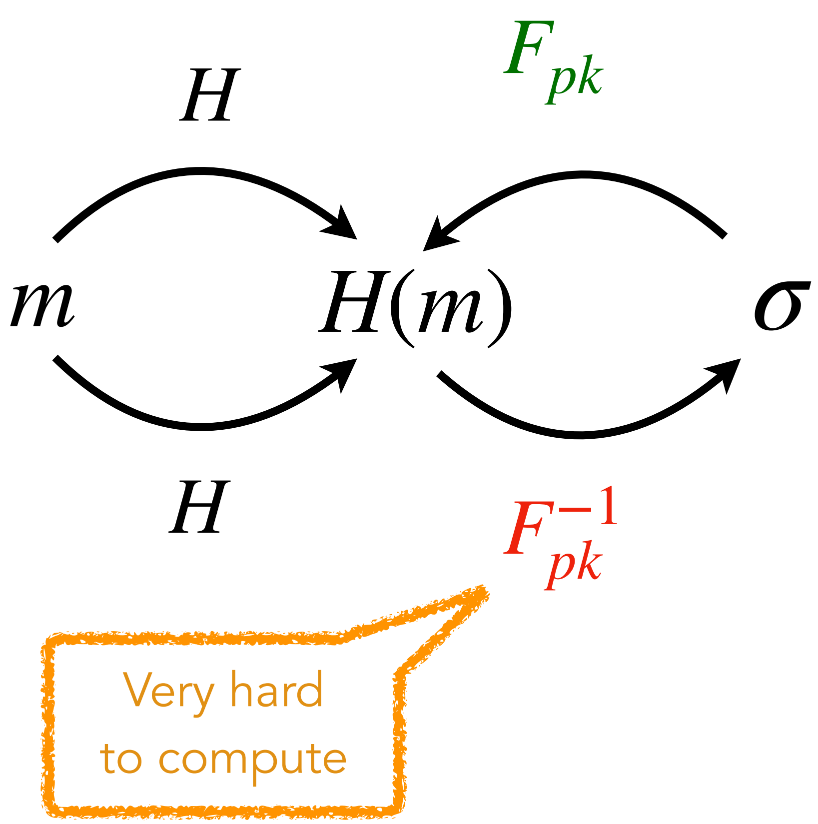
CryptoExperts
WE INNOVATE TO SECURE YOUR BUSINESS

Table of Contents

Introduction

How to build signature schemes?

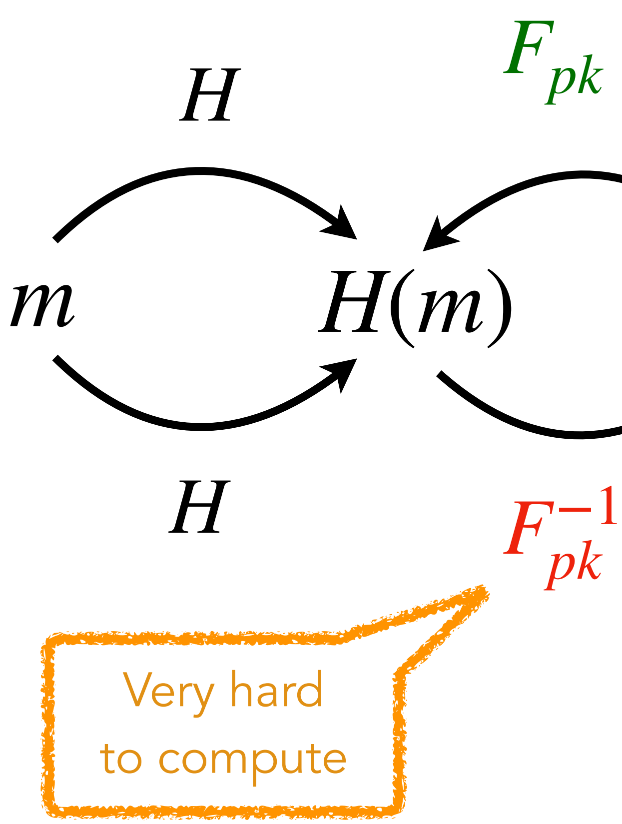
<u>Hash & Sign</u>



- Short signatures
- "Trapdoor" in the public key

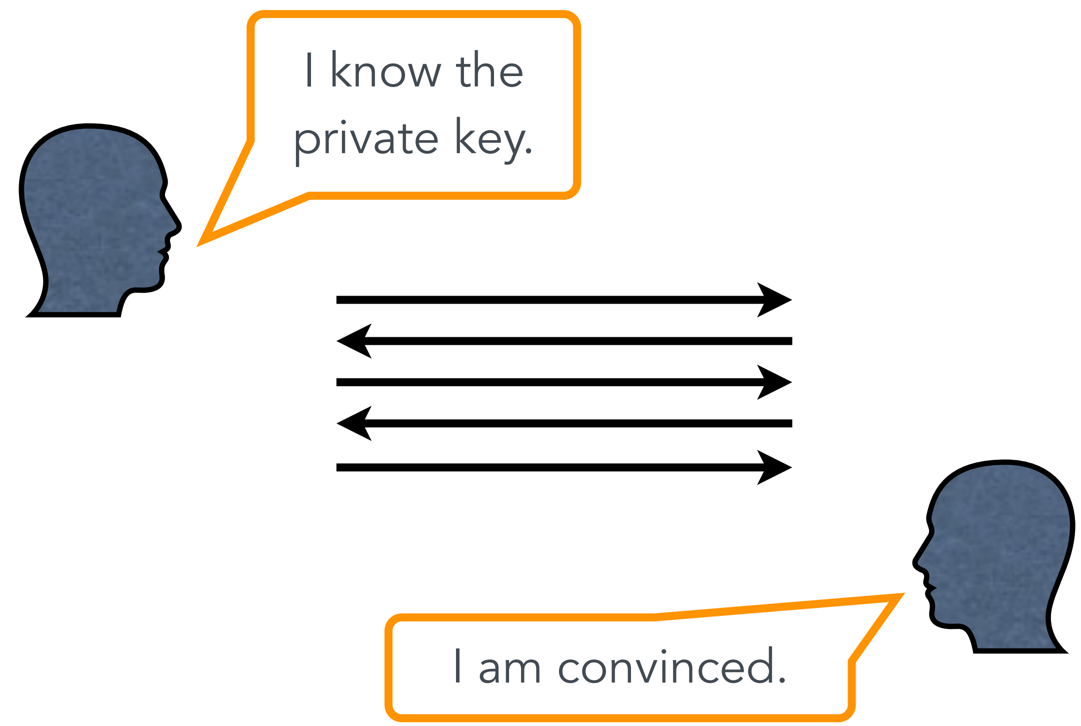
How to build signature schemes?

Hash & Sign



- Short signatures
- "Trapdoor" in the public key

From an identification scheme



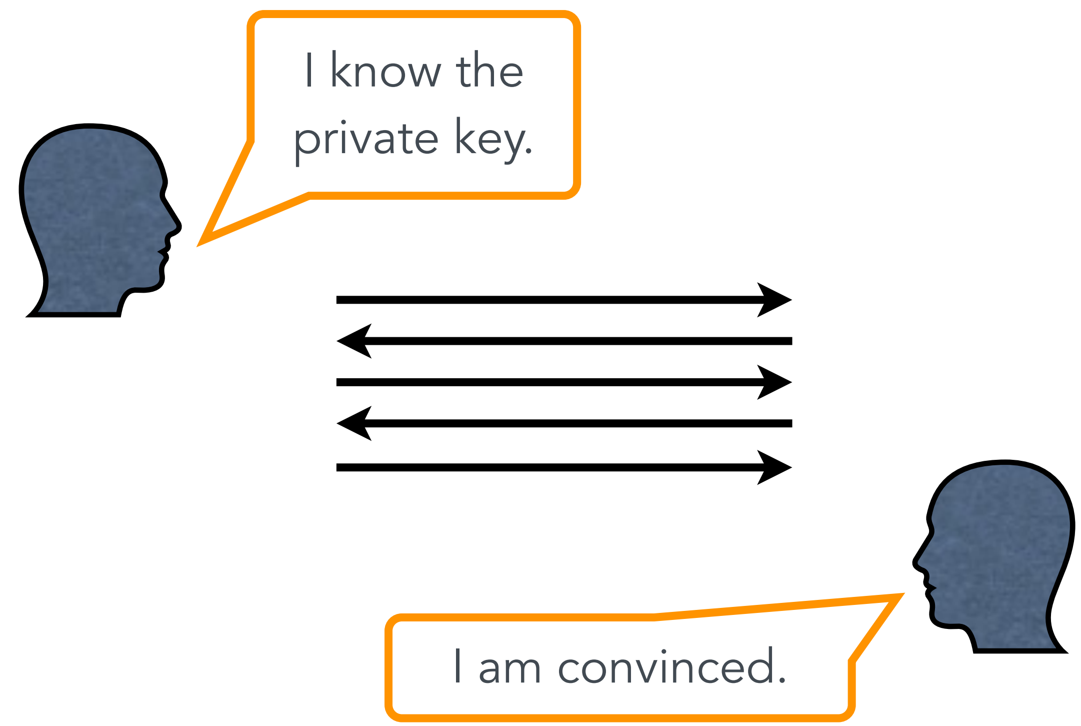
- Large(r) signatures
- Short public key

How to build signature schemes?

Hash & Sign

- Short signatures
- “Trapdoor” in the public key

From an identification scheme

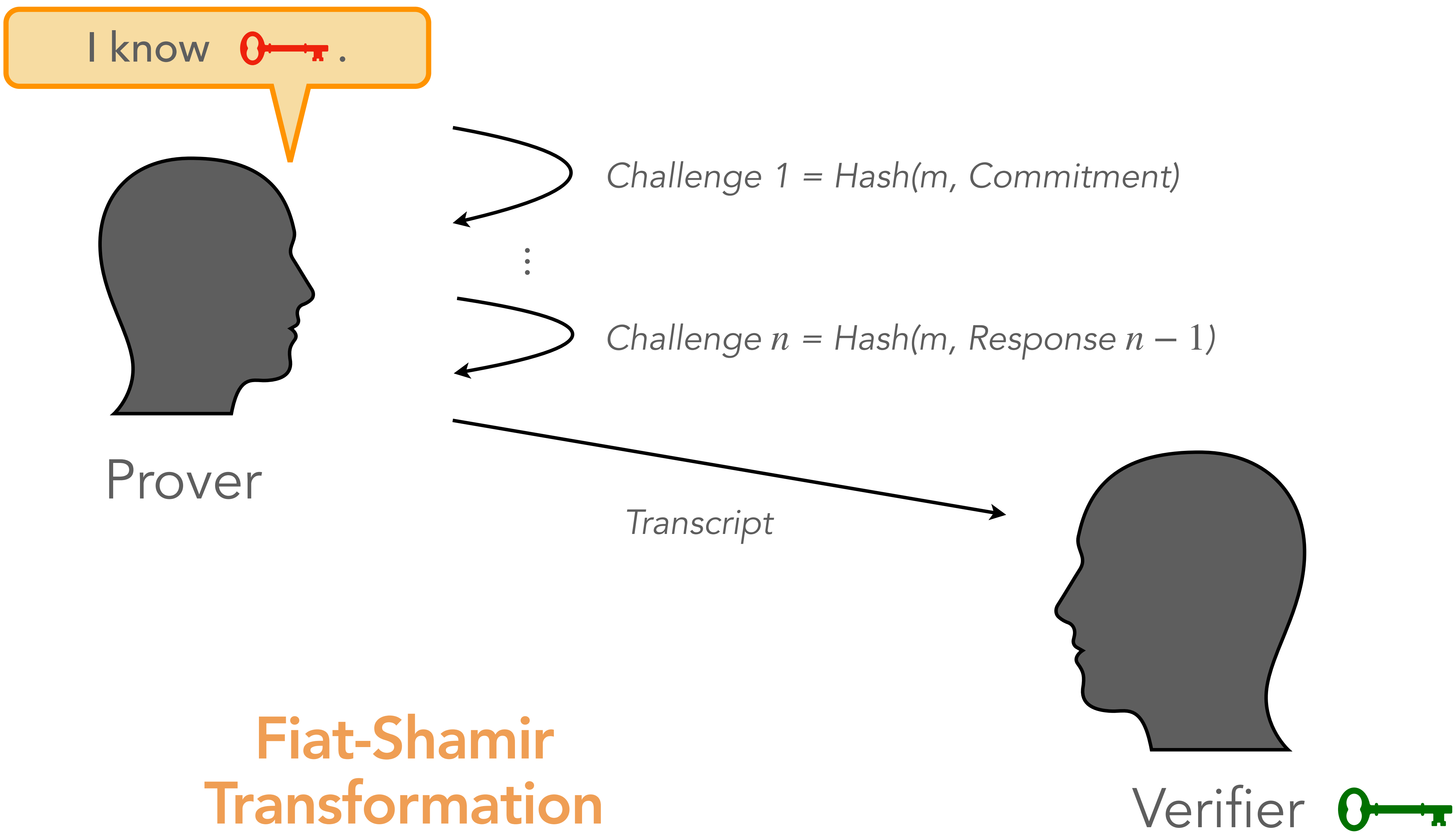


- Large(r) signatures
- Short public key

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



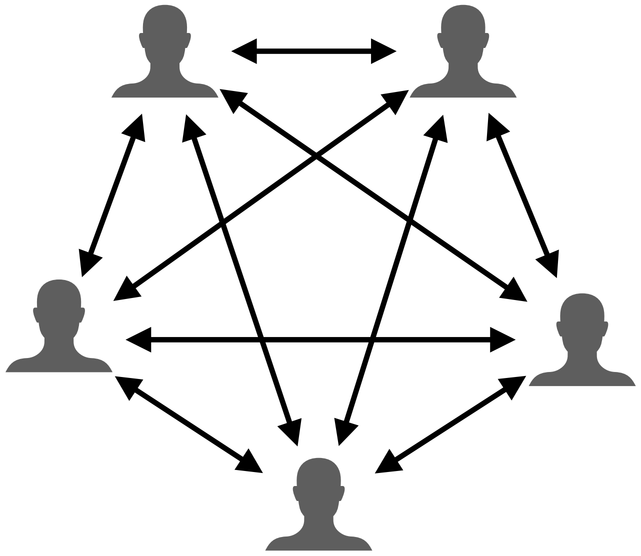
Fiat-Shamir Transformation

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

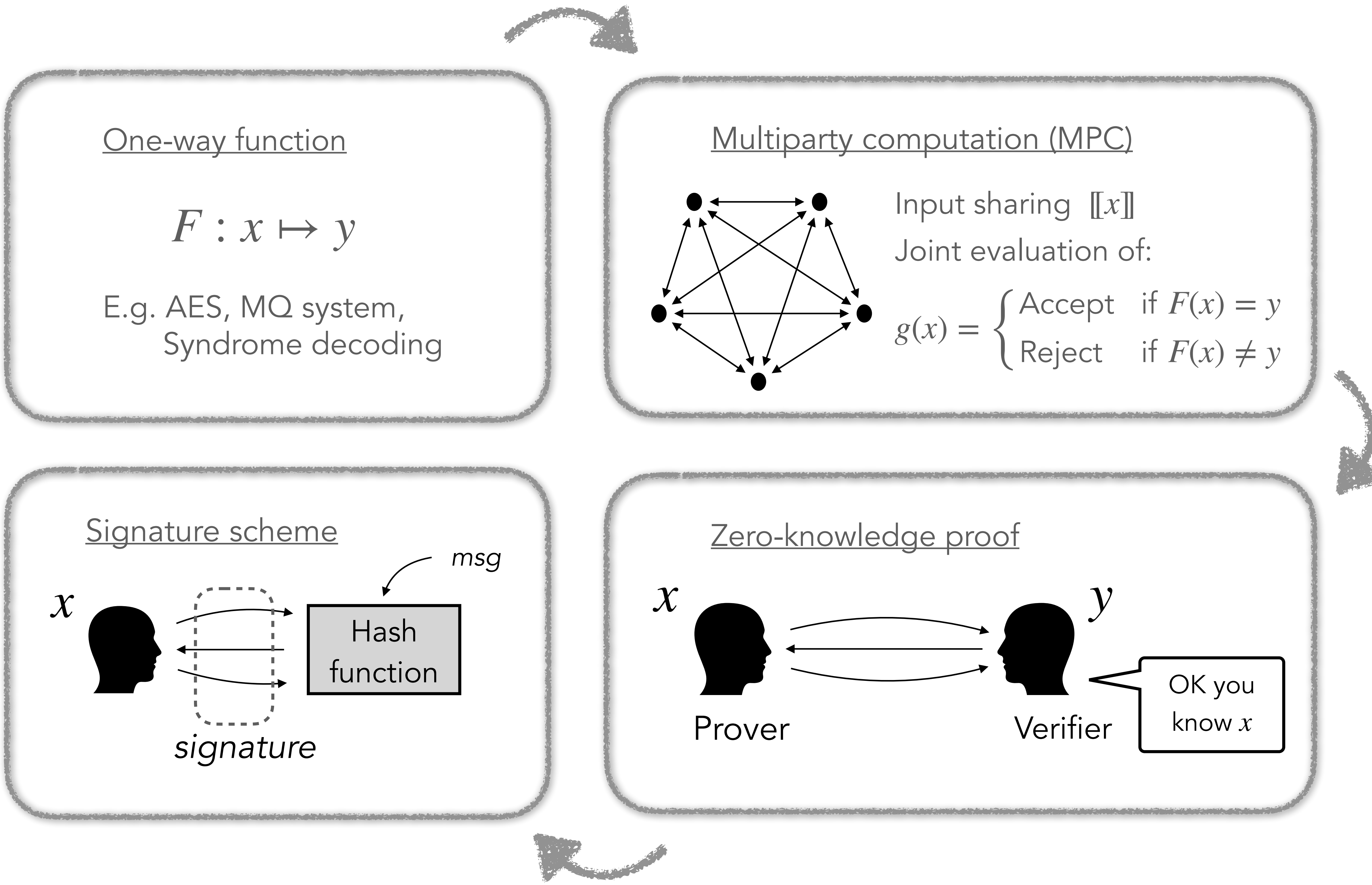


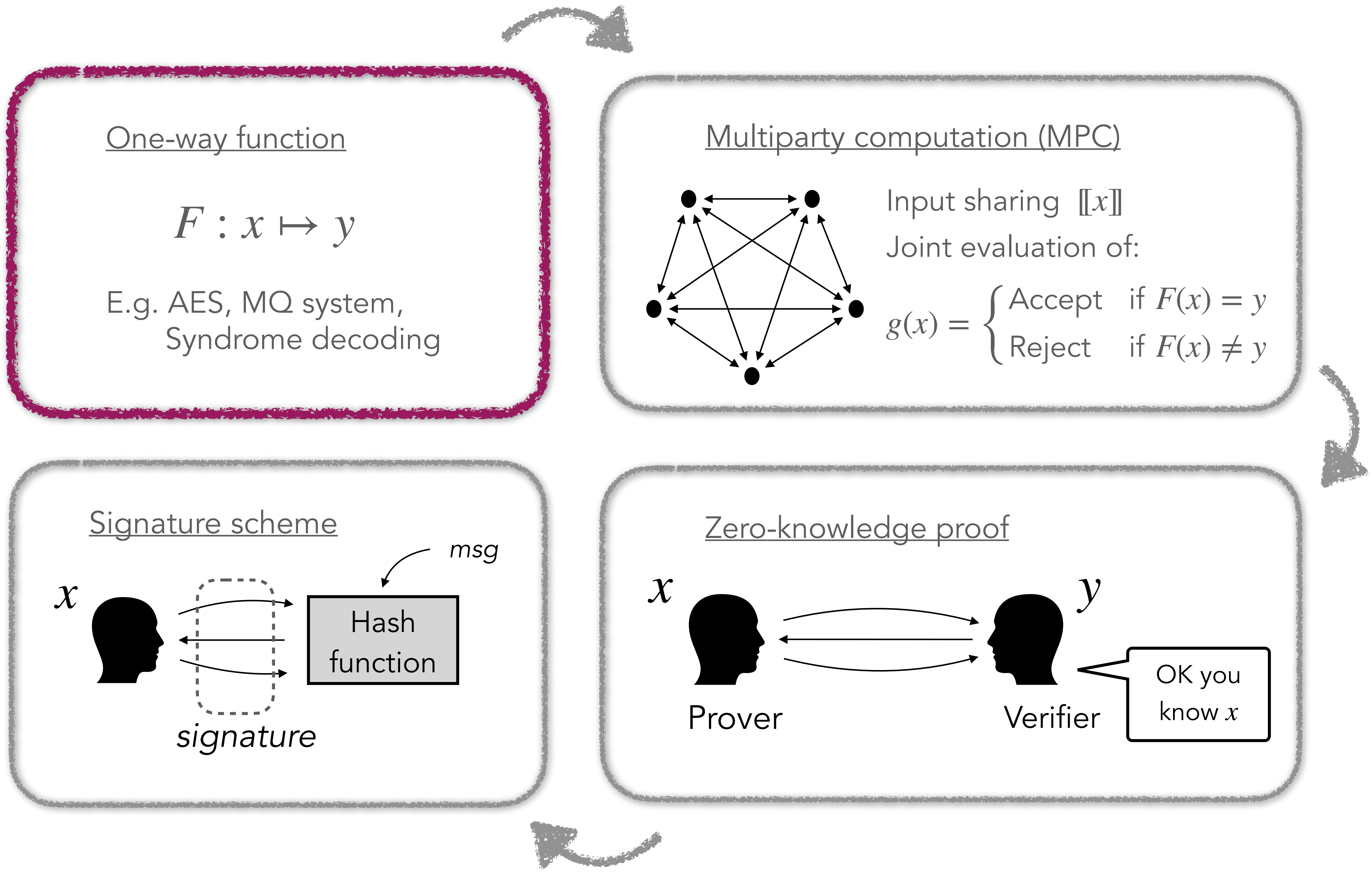
- **Generic**: can be applied to any cryptographic problem

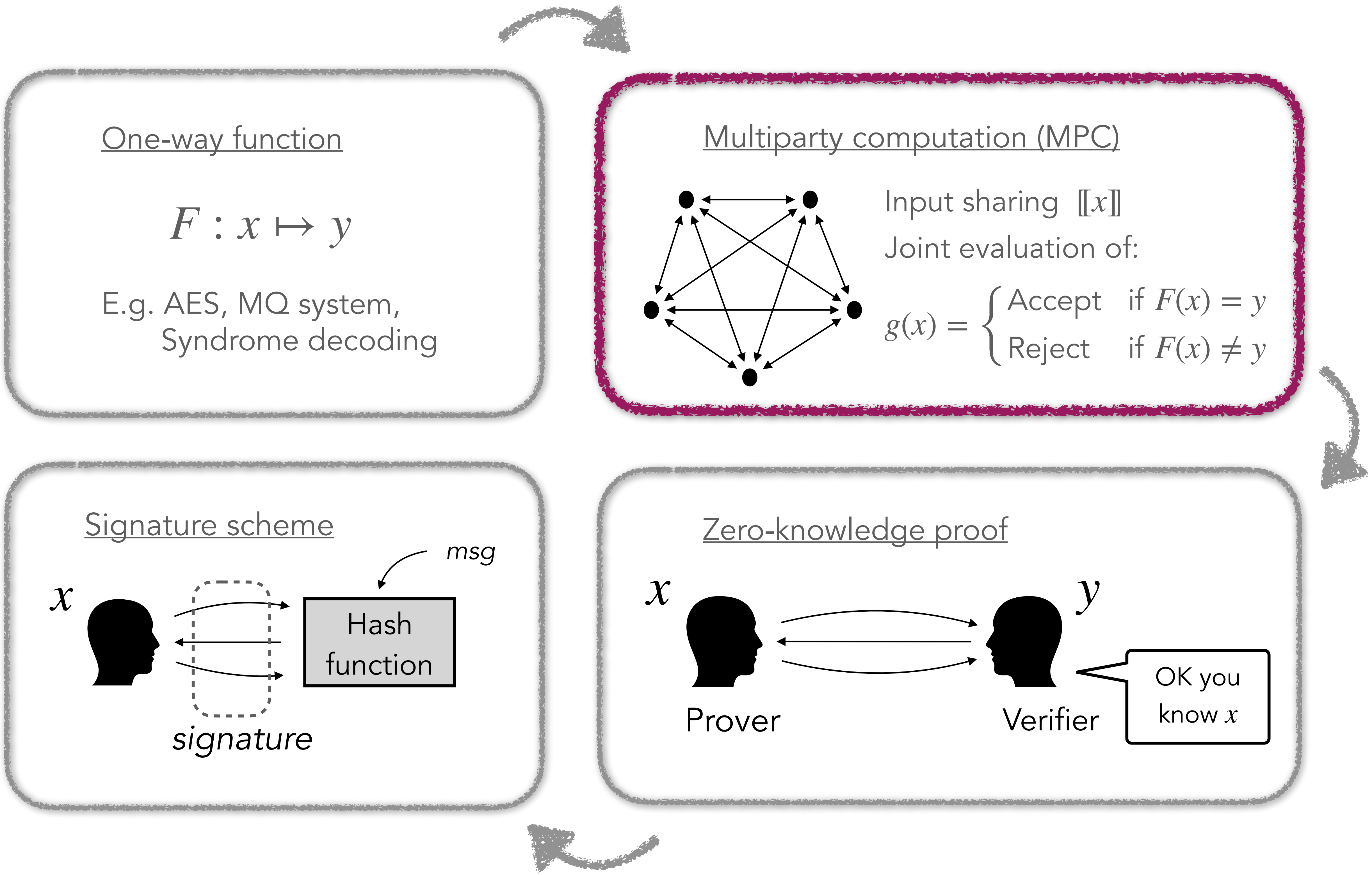
MPC in the Head

- **[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

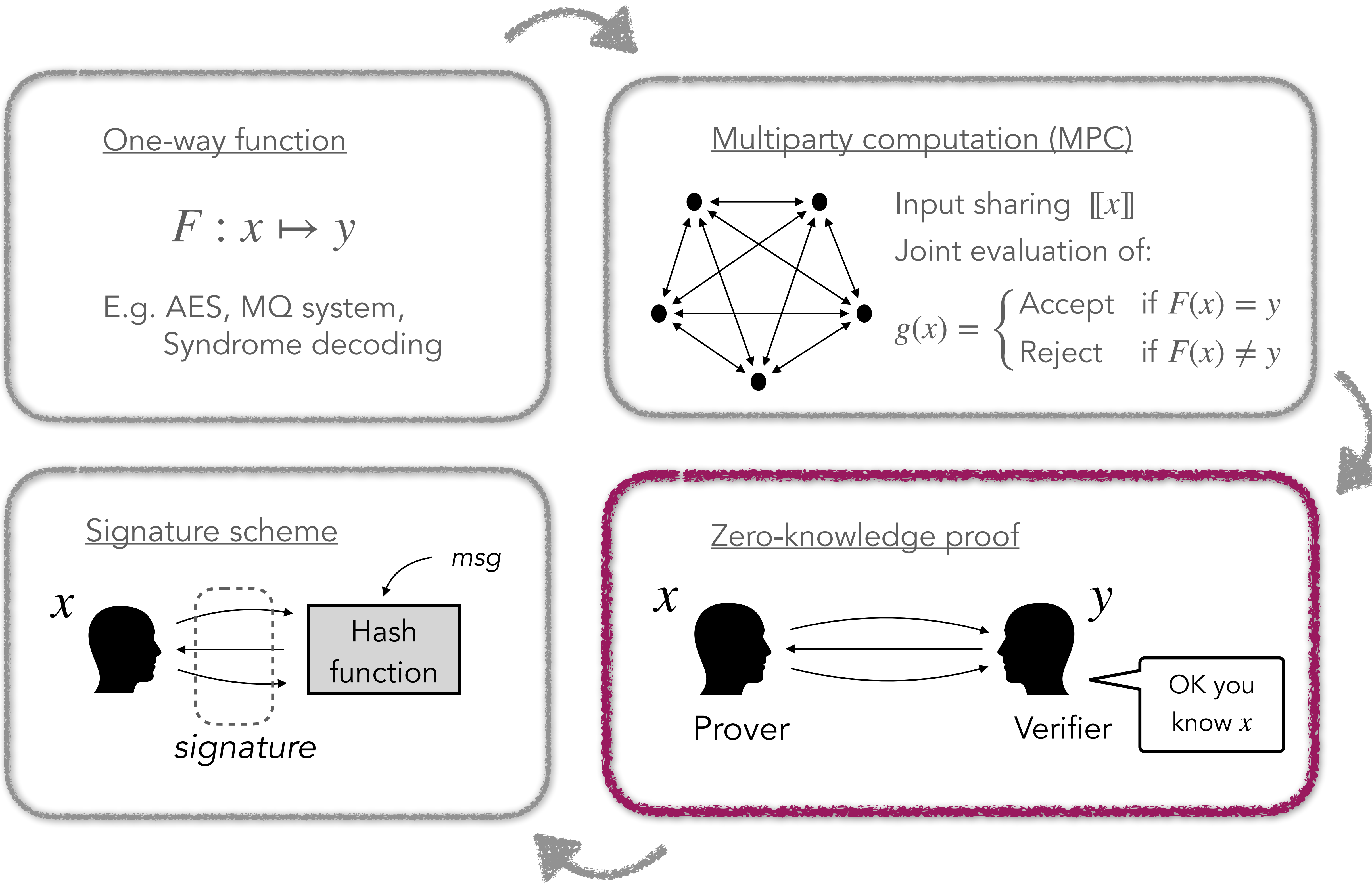
AIMer	*MQOM*
Biscuit	*PERK*
FAEST	*RYDE*
MIRA	*SDitH*
MiRitH	


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

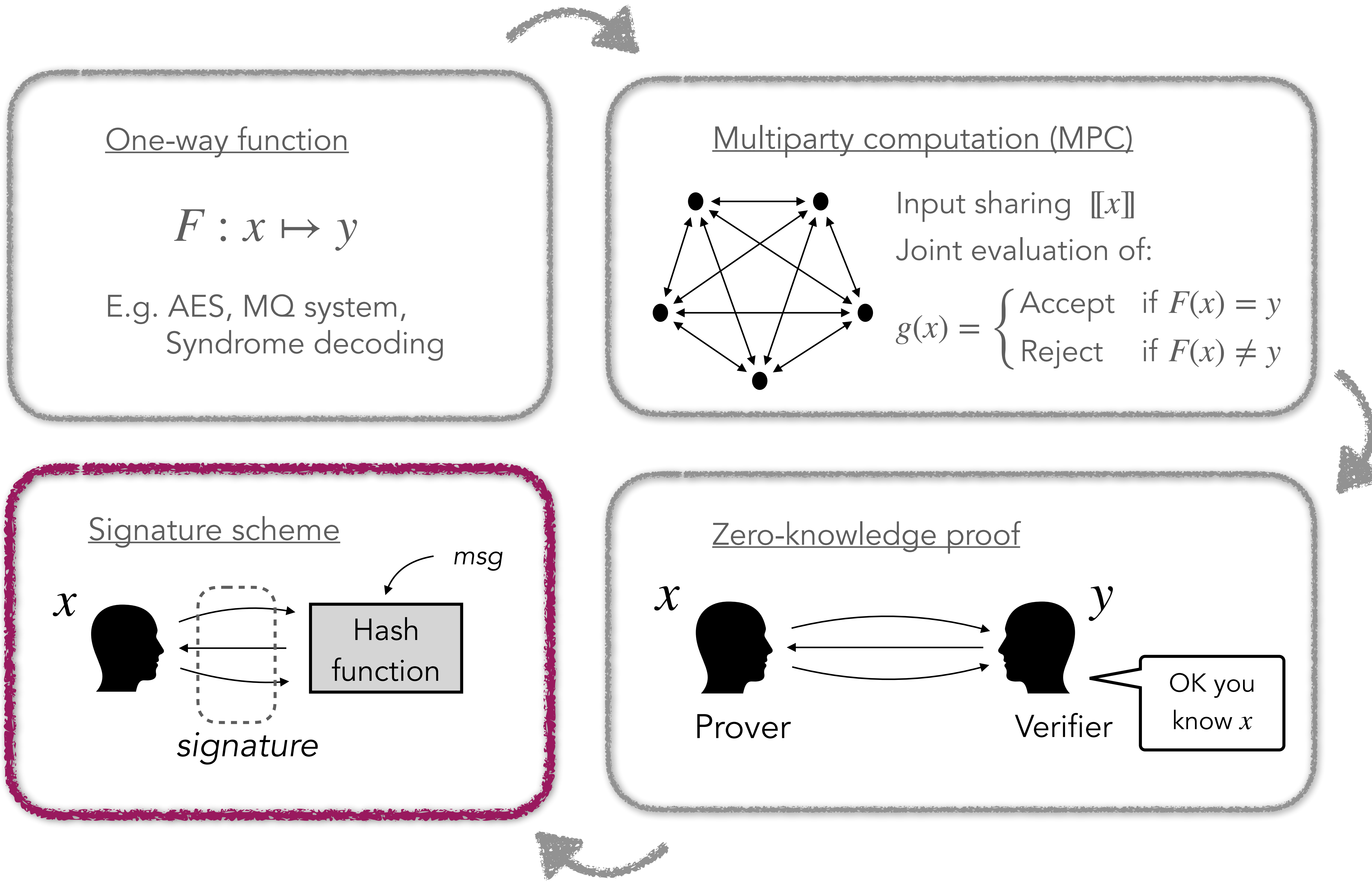

One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
x
msg
Hash
function
signature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

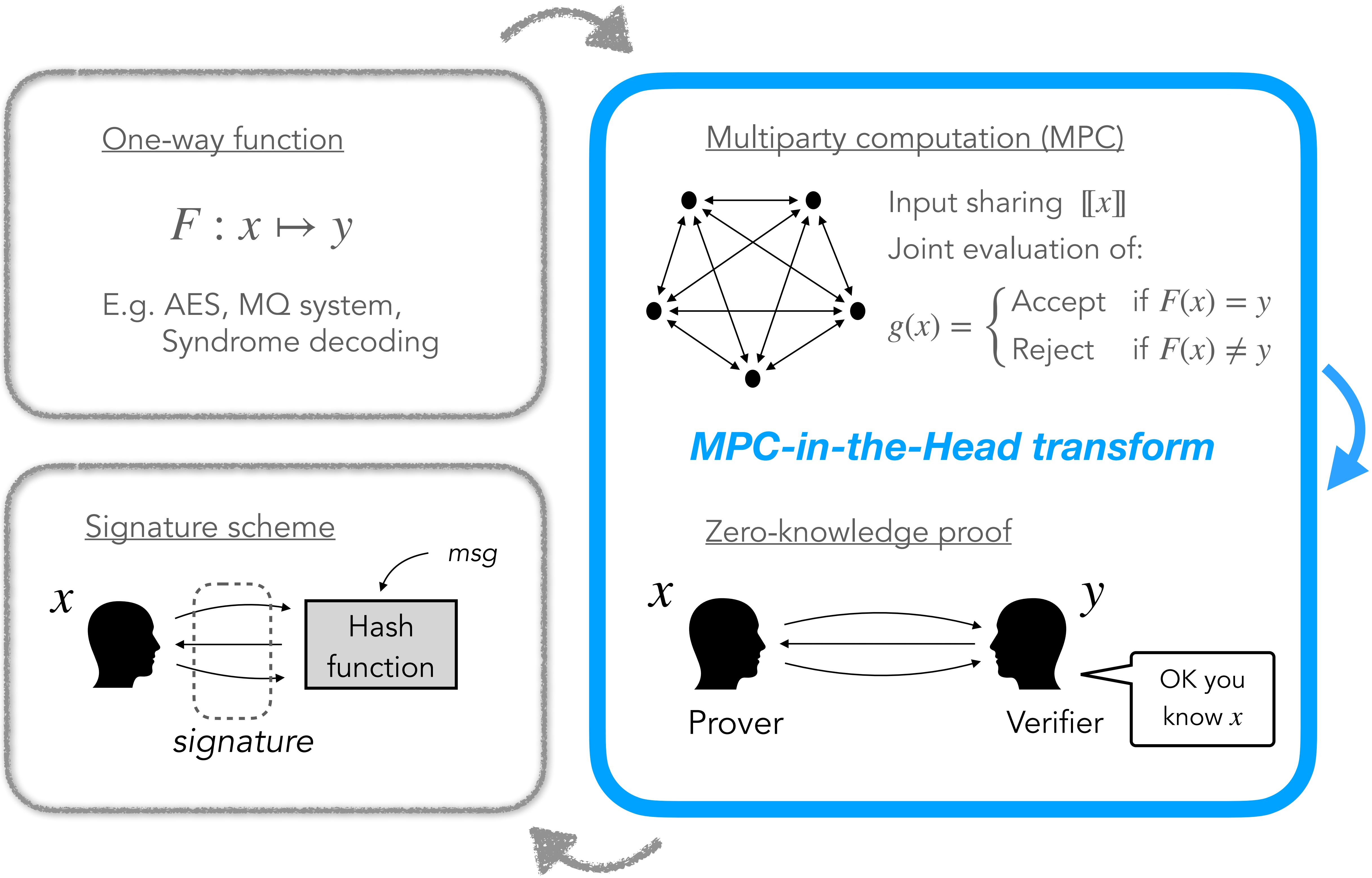


One-way function
F : x ↦ y
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing [[x]]
Joint evaluation of:
g(x) = Accept if F(x) = y
Reject if F(x) ≠ y
Signature scheme
x
msg
Hash
function
signature
Zero-knowledge proof
x
y
Prover
Verifier
OK you
know x



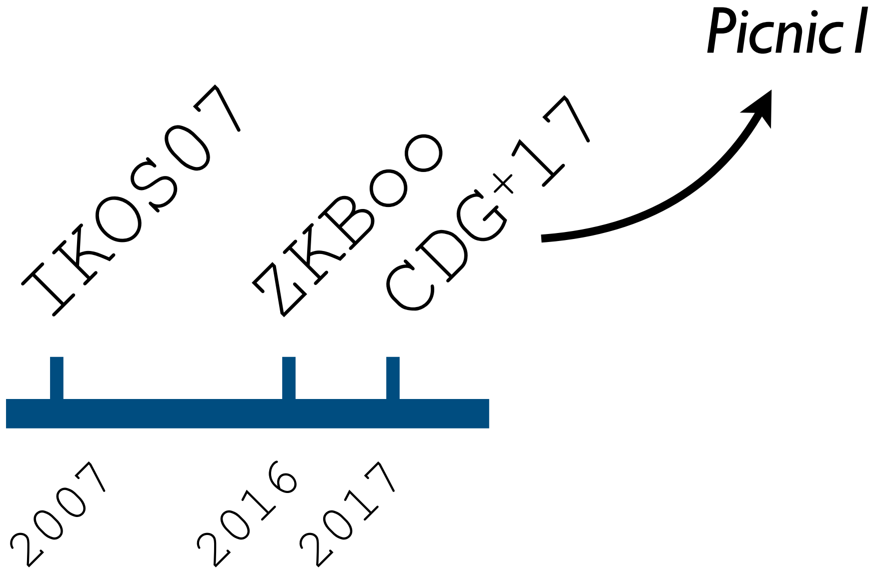
One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
x
msg
Hash function
signature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

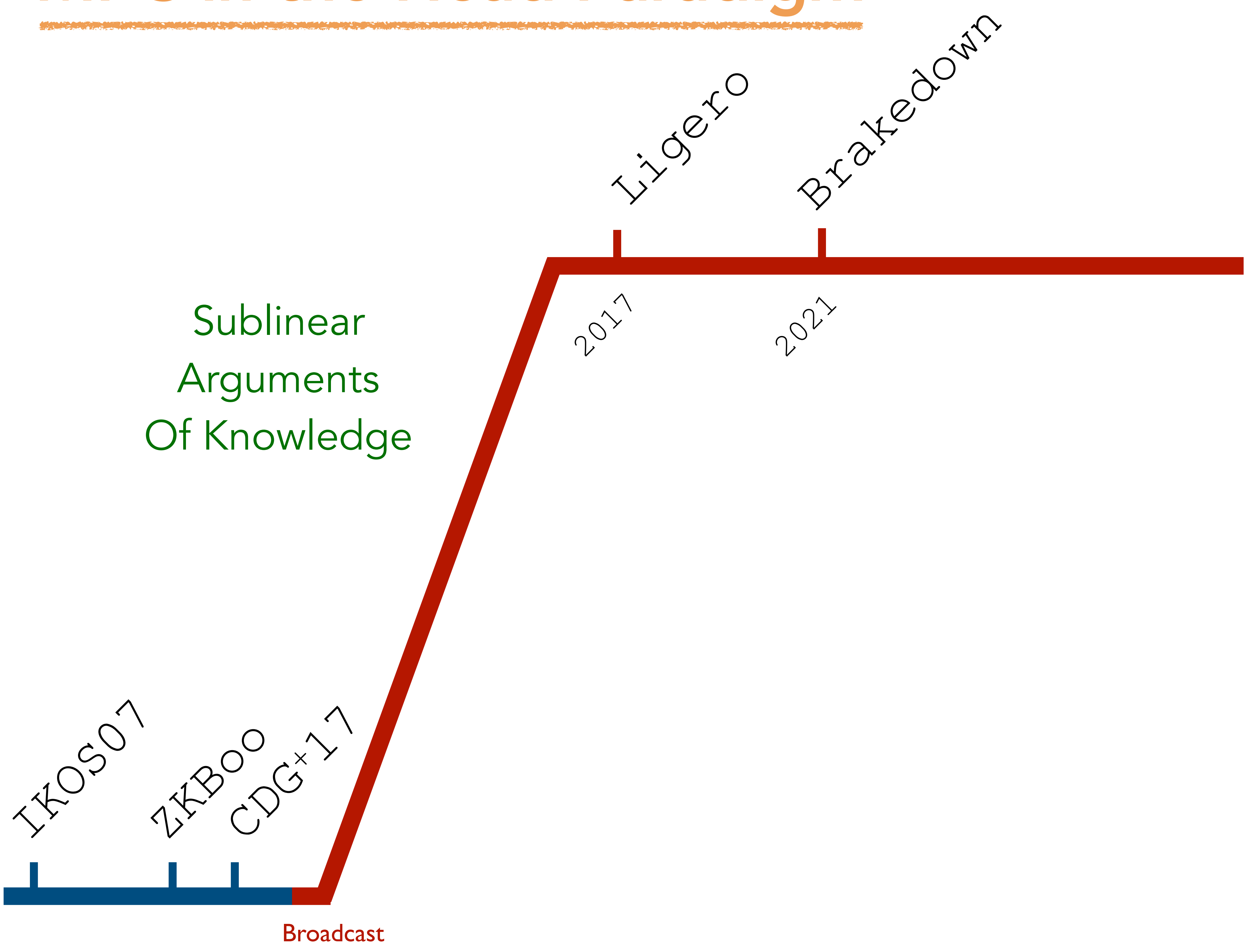

One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
MPC-in-the-Head transform
Zero-knowledge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

MPCitH for signature schemes

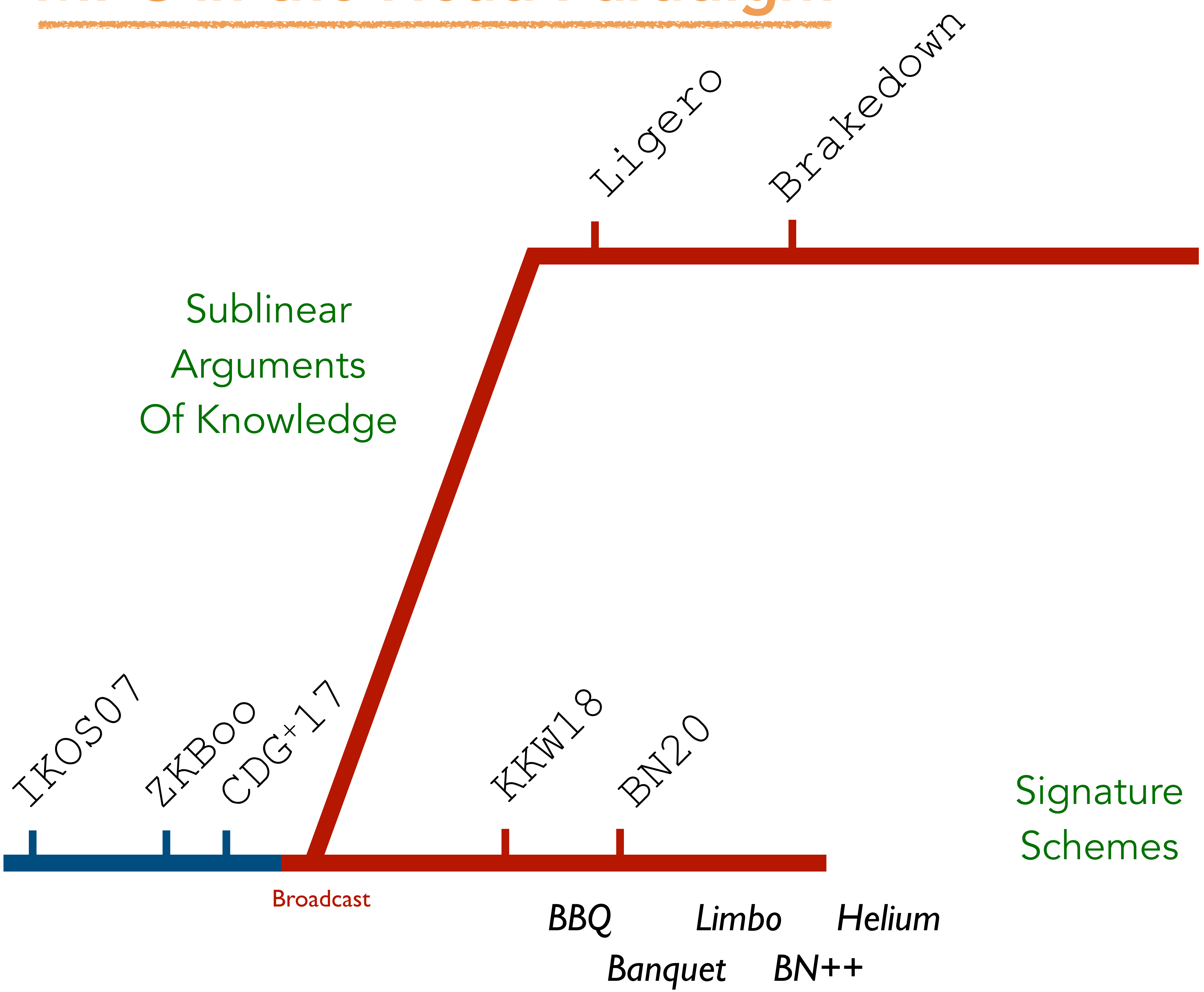
MPC-in-the-Head Paradigm



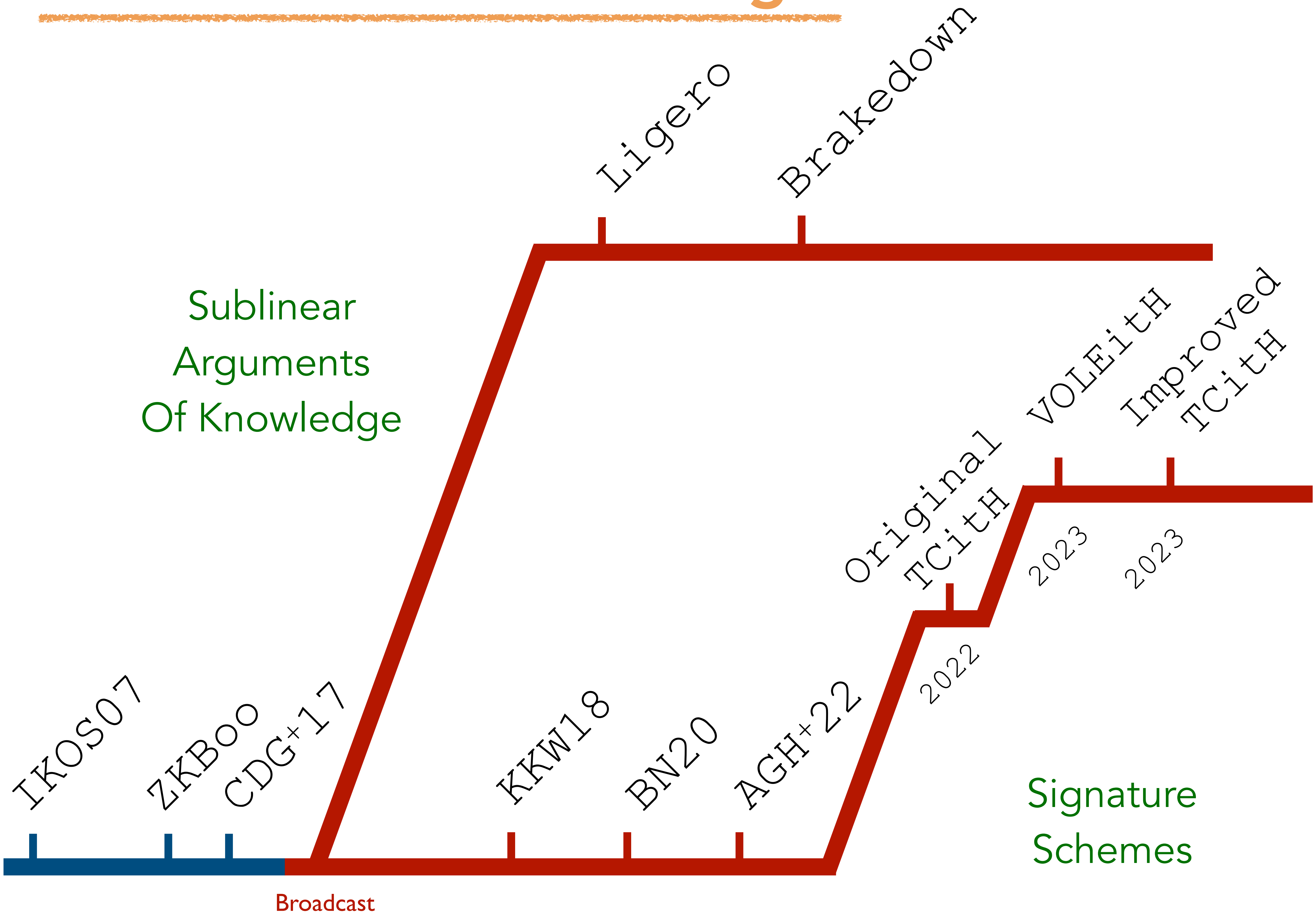
MPC-in-the-Head Paradigm



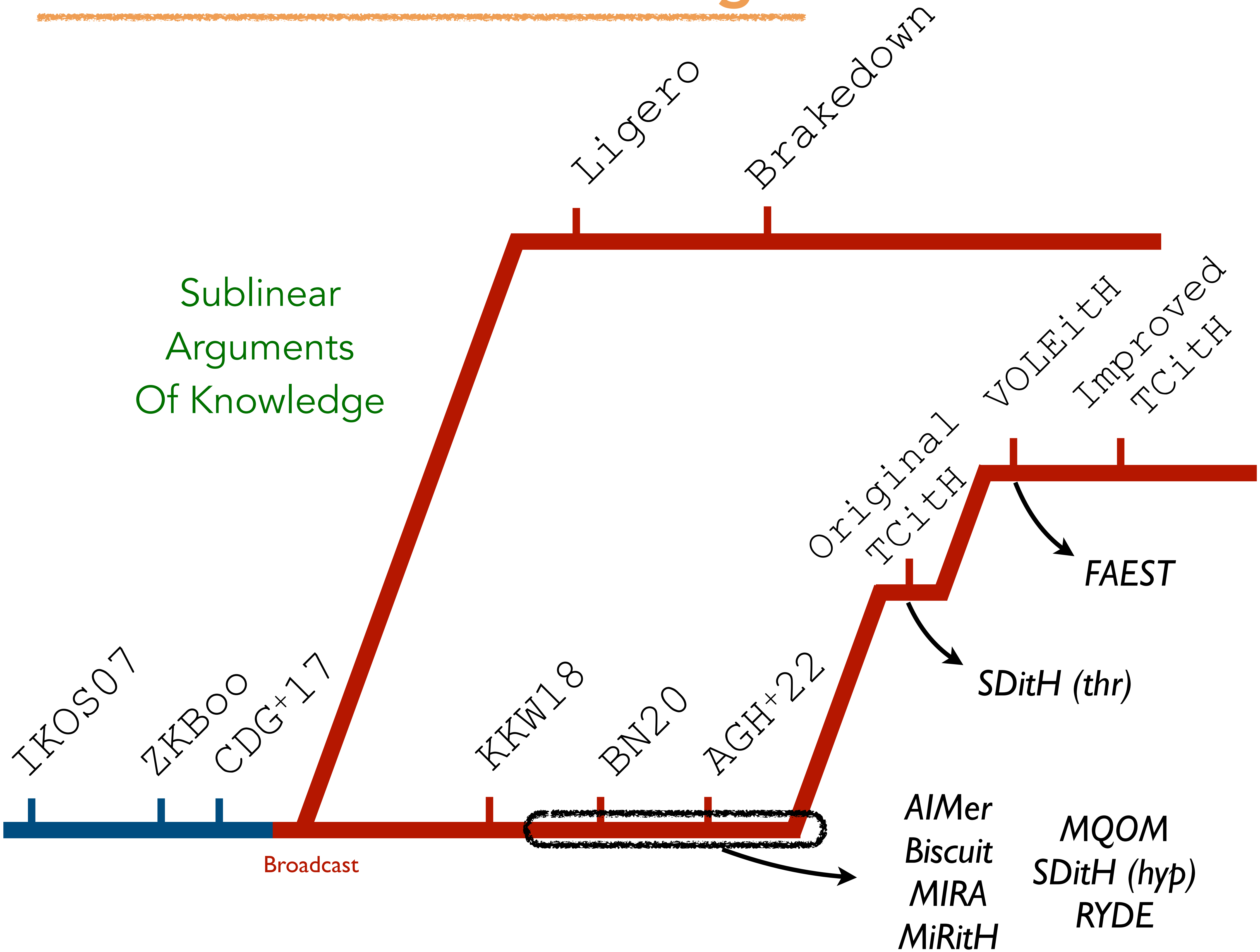
MPC-in-the-Head Paradigm



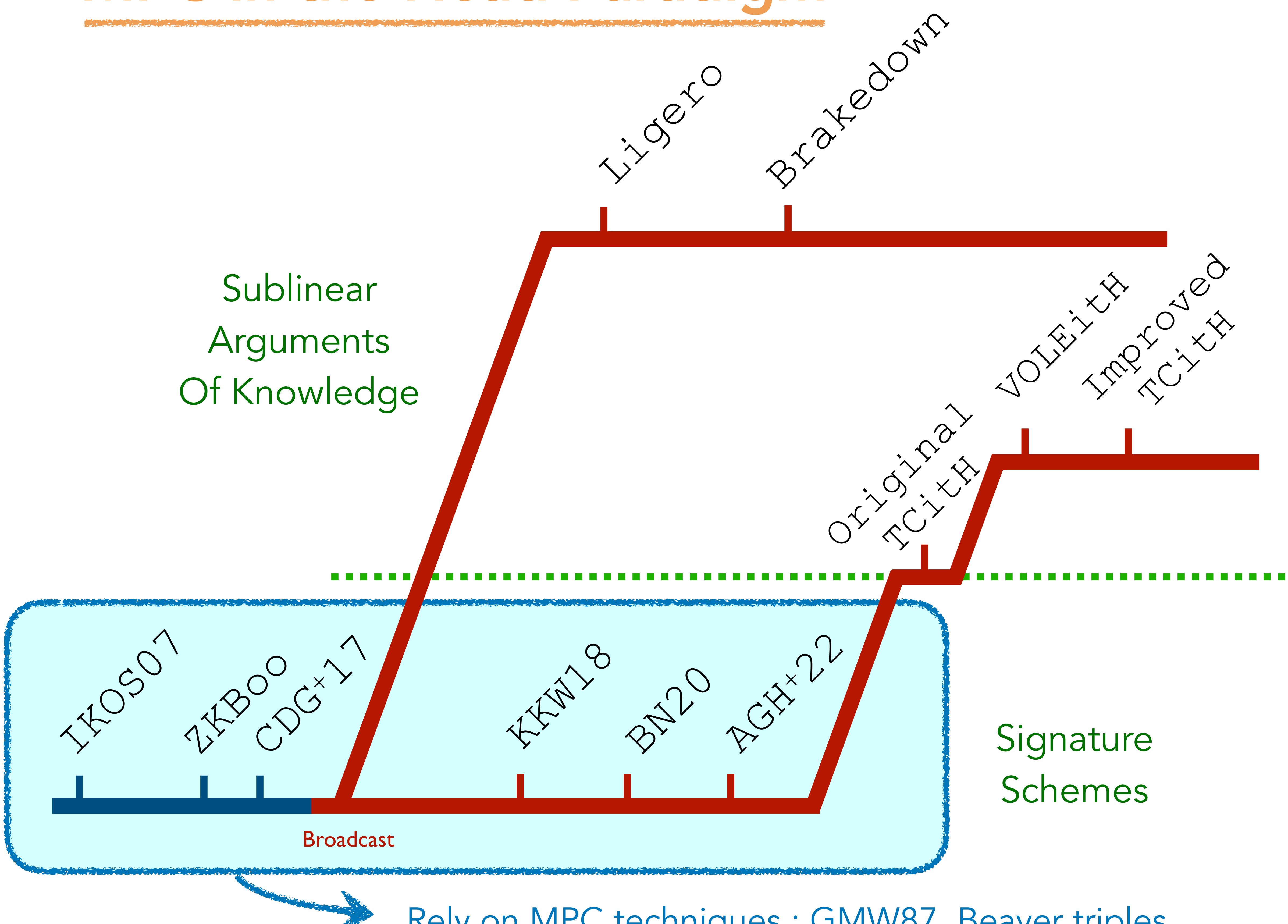
MPC-in-the-Head Paradigm



MPC-in-the-Head Paradigm

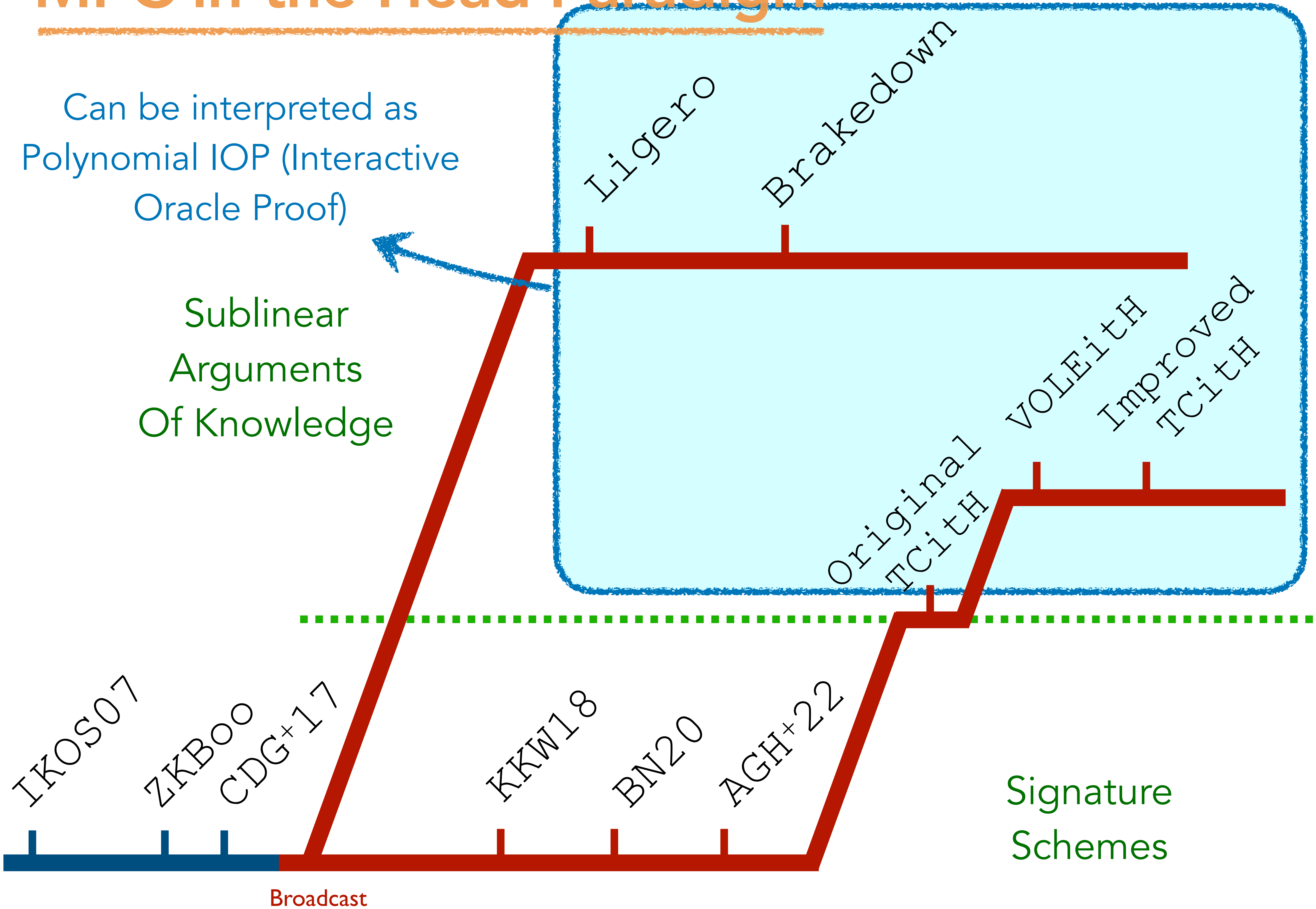


MPC-in-the-Head Paradigm



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

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

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 $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

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 $P_1, \ldots, P_n$

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 Commit 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

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 $P_1, \dots, P_n$

$\mathrm{Com}(P_1, \dots, P_n)$ →

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

← $\gamma_1, \dots, \gamma_m$

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

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

Q →

Well-defined!

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

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

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 $P_1, \ldots, P_n$

$\mathrm{Com}(P_1, \ldots, P_n)$ →

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

← $\gamma_1, \ldots, \gamma_m$

④ Commit 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)

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 Commit 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

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

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 $P_1, \dots, P_n$

$\mathrm{Com}(P_1, \dots, P_n)$ →

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

← $\gamma_1, \dots, \gamma_m$

④ Commit 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. Check that

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

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.

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

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

③ Choose random coefficients $\gamma_1, \ldots, \gamma_m \leftarrow^{\$} \mathbb{F}$ ← $\gamma_1, \ldots, \gamma_m$

④ Commit 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.

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

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

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.

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

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

$\xrightarrow{\mathrm{Com}(P_1, \dots, P_n)}$

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

$\xleftarrow{\gamma_1, \dots, \gamma_m}$

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

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

$\xrightarrow{Q}$

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

$\xleftarrow{r}$

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

$\xrightarrow{v_1, \dots, v_n}$

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

Check that

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

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 $P_1, \ldots, P_n$

→ $\mathrm{Com}(P_1, \ldots, P_n)$

Verifier

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

← $\gamma_1, \ldots, \gamma_m$

Prover

④ Commit 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

Verifier

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

← r

Prover

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

→ $v_1, \ldots, v_n$

Verifier

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

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

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 Commit 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

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

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)

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 $P_1, \ldots, P_n$

$\text{Com}(P_1, \ldots, P_n)$ →

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

← $\gamma_1, \ldots, \gamma_m$

④ Commit 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

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

⚠️ Leak information about the witness $w_1, \ldots, w_n$

Verifier 👀

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 $P_i(0) = w_i$

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

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

$\text{Com}(P_0, P_1, \ldots, P_n)$ →

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

← $\gamma_1, \ldots, \gamma_m$

④ Commit 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

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

<u>Prover</u>

<u>Verifier</u> 👀

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 $P_0, P_1, \dots, P_n$

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

Verifier

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

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

Prover

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

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

Prover → Verifier: Q

Verifier

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

Verifier → Prover: r

Prover

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

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

Verifier

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

Check that

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

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 $P_0, P_1, \ldots, P_n$

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

Verifier

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

← $\gamma_1, \ldots, \gamma_m$

Prover

④ Commit 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

Verifier

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

← r

Prover

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

→ $v_0, v_1, \ldots, v_n$

Verifier

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

Check that

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

Prover | Verifier

How to commit to polynomials?

degree 10 000

degree 1000

degree 100

degree 10

degree 1

① VOLEitH / TCitH-GGM

[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

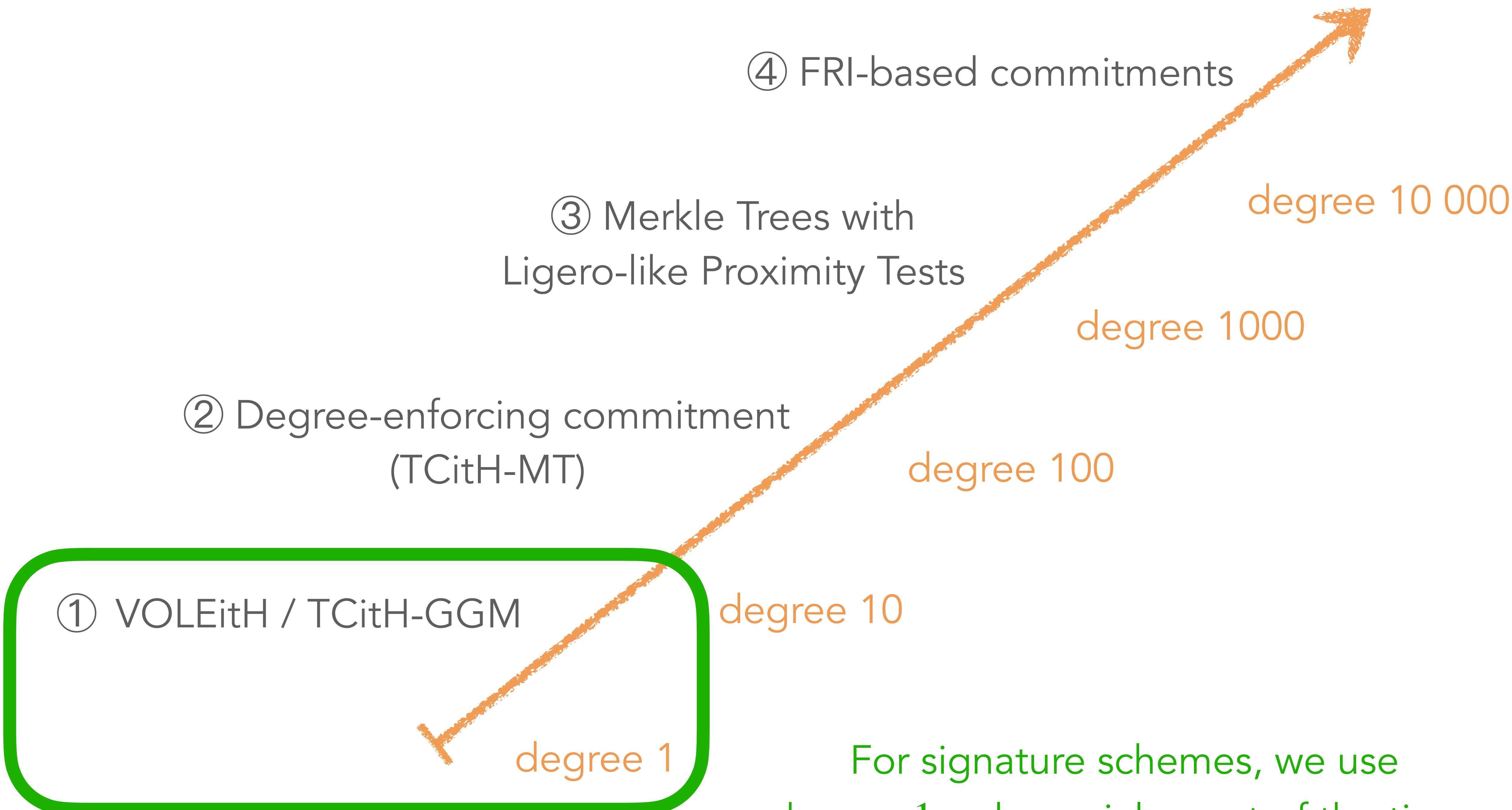
degree 10

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?



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

Committing to a Polynomial using a Seed Tree

Commit: we want to sample and commit **degree-1 polynomials** such that $P_i(0) = w_i$.

1. Sample N values $r_1, \ldots, r_N \in \mathbb{F}^n$.
2. Commit to each value r_i independently.
3. Reveal the value

$$\Delta w \leftarrow w + \sum_{i}^{N} r_i.$$

4. For all i, the committed polynomial $P_i(X)$ is

$$P_i(X) = a_i \cdot X + w_i \qquad \text{with} \qquad a = \sum_{i=1}^{N} \frac{1}{\phi(i)} \cdot r_i \in \mathbb{F}^n.$$

Committing to a Polynomial using a Seed Tree

Commit: we want to sample and commit **degree-1 polynomials** such that $P_i(0) = w_i$.

1. Sample N values $r_1, \ldots, r_N \in \mathbb{F}^n$.
2. Commit to each value r_i independently.
3. Reveal the value

$$\Delta w \leftarrow w + \sum_i^N r_i.$$

4. For all i, the committed polynomial $P_i(X)$ is

$$P_i(X) = a_i \cdot X + w_i \qquad \text{with} \qquad a = \sum_{i=1}^N \frac{1}{\phi(i)} \cdot r_i \in \mathbb{F}^n.$$

Open: to reveal $P(r)$ with $r := \phi(i^*)$, $i^* \in \{1,\ldots,N\}$, just reveal all $\{r_i\}_{i \neq i^*}$.

Committing to a Polynomial using a Seed Tree

Commit: we want to sample and commit **degree-1 polynomials** such that $P_i(0) = w_i$.

1. Sample N values $r_1, \ldots, r_N \in \mathbb{F}^n$.
2. Commit to each value r_i independently.
3. Reveal the value
$$\Delta w \leftarrow w + \sum_i^N r_i.$$
4. For all i, the committed polynomial $P_i(X)$ is
$$P_i(X) = a_i \cdot X + w_i \qquad \text{with} \qquad a = \sum_{i=1}^{N} \frac{1}{\phi(i)} \cdot r_i \in \mathbb{F}^n.$$

Open: to reveal $P(r)$ with $r = \phi(i^*)$, $i^* \in \{1,\ldots,N\}$, just reveal all $\{r_i\}_{i \neq i^*}$.

🔎 An attacker can only restore $w + r_{i^*}$, not w.

Committing to a Polynomial using a Seed Tree

Commit: we want to sample and commit **degree-1 polynomials** such that $P_i(0) = w_i$.

1. Sample N values $r_1, \ldots, r_N \in \mathbb{F}^n$.
2. Commit to each value r_i independently.
3. Reveal the value
$$\Delta w \leftarrow w + \sum_{i}^{N} r_i.$$
4. For all i, the committed polynomial $P_i(X)$ is
$$P_i(X) = a_i \cdot X + w_i \qquad \text{with} \qquad a = \sum_{i=1}^{N} \frac{1}{\phi(i)} \cdot r_i \in \mathbb{F}^n.$$

Open: to reveal $P(r)$ with $r := \phi(i^*)$, $i^* \in \{1, \ldots, N\}$, just reveal all $\{r_i\}_{i \neq i^*}$.

Verify: just check that the commitment of all $\{r_i\}_{i \neq i^*}$ and deduce that
$$P_i(r) = v_i \qquad \text{with} \qquad v = \Delta w + \sum_{i=1, i \neq i^*}^{N} \left(\frac{\phi(i^*)}{\phi(i)} - 1 \right) \cdot r_i \in \mathbb{F}^n,$$

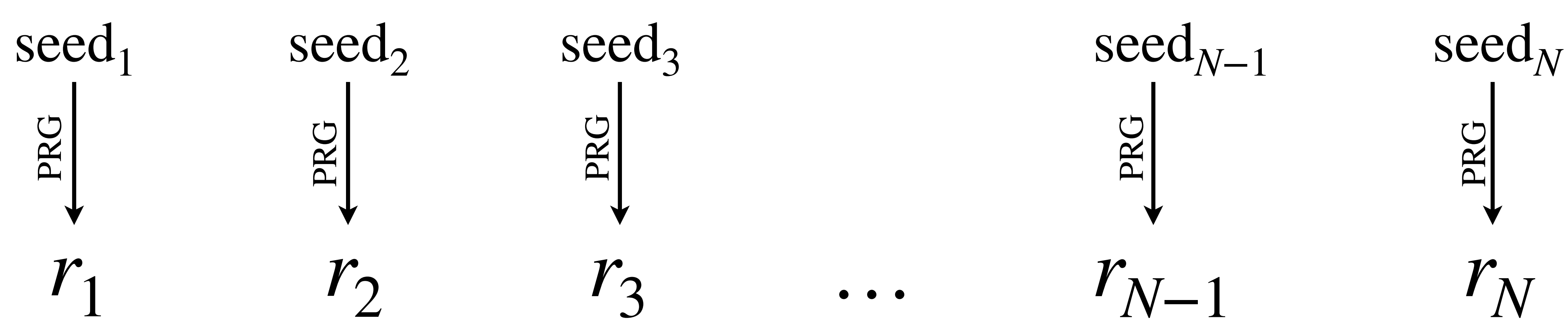
Committing to a Polynomial using a Seed Tree

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

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

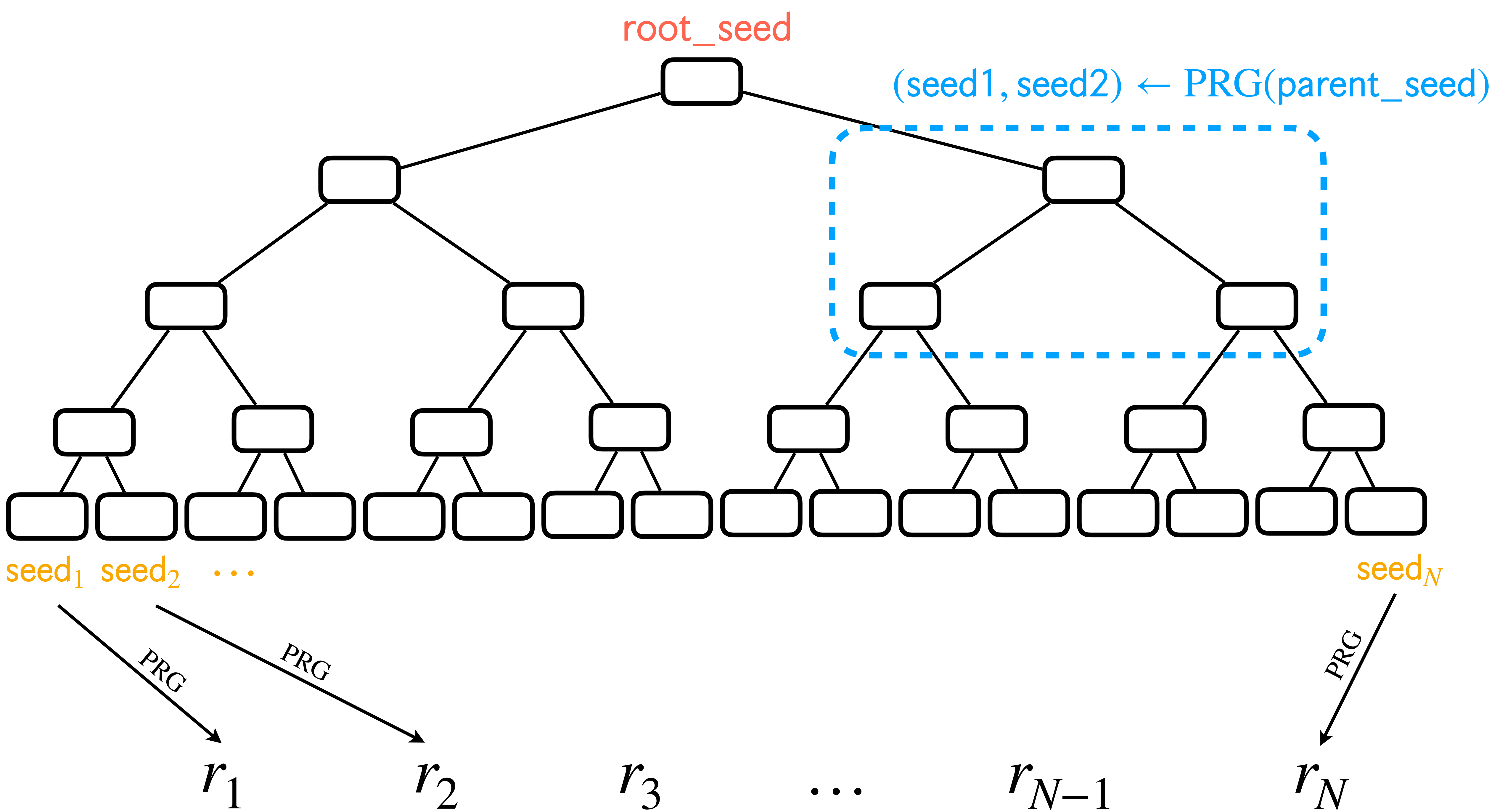
Committing to a Polynomial using a Seed Tree

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



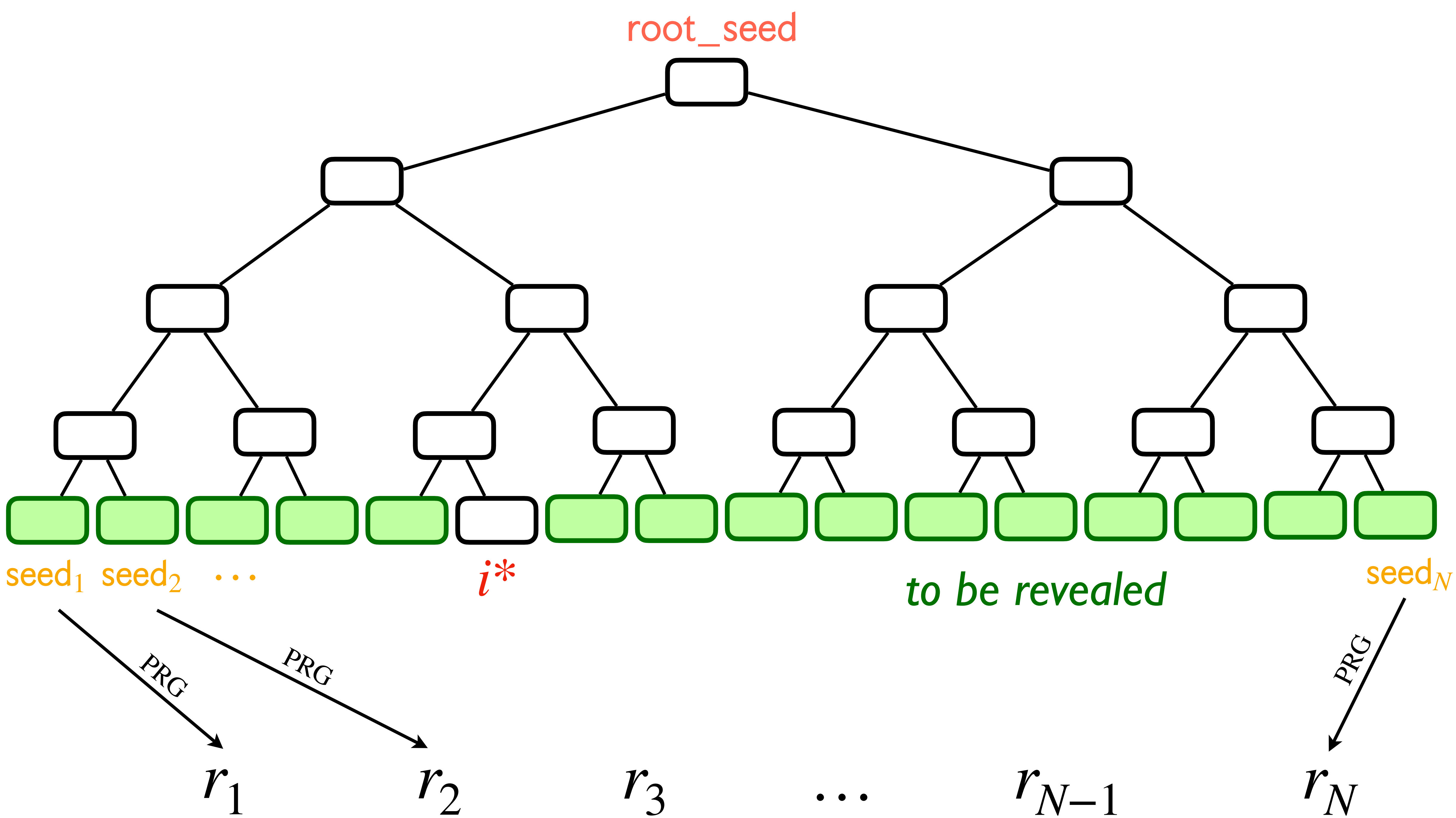
Committing to a Polynomial using a Seed Tree

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



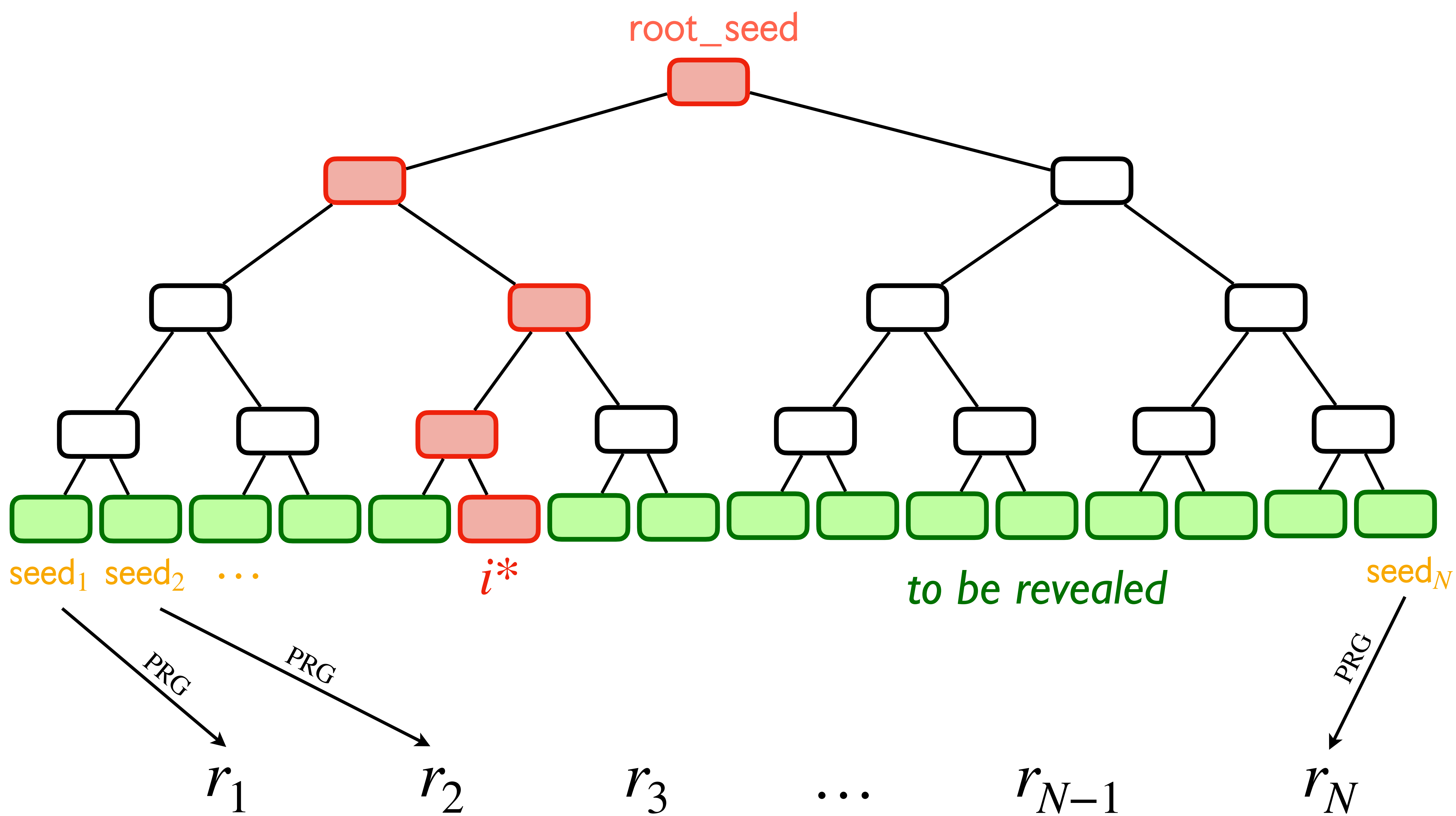
Committing to a Polynomial using a Seed Tree

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



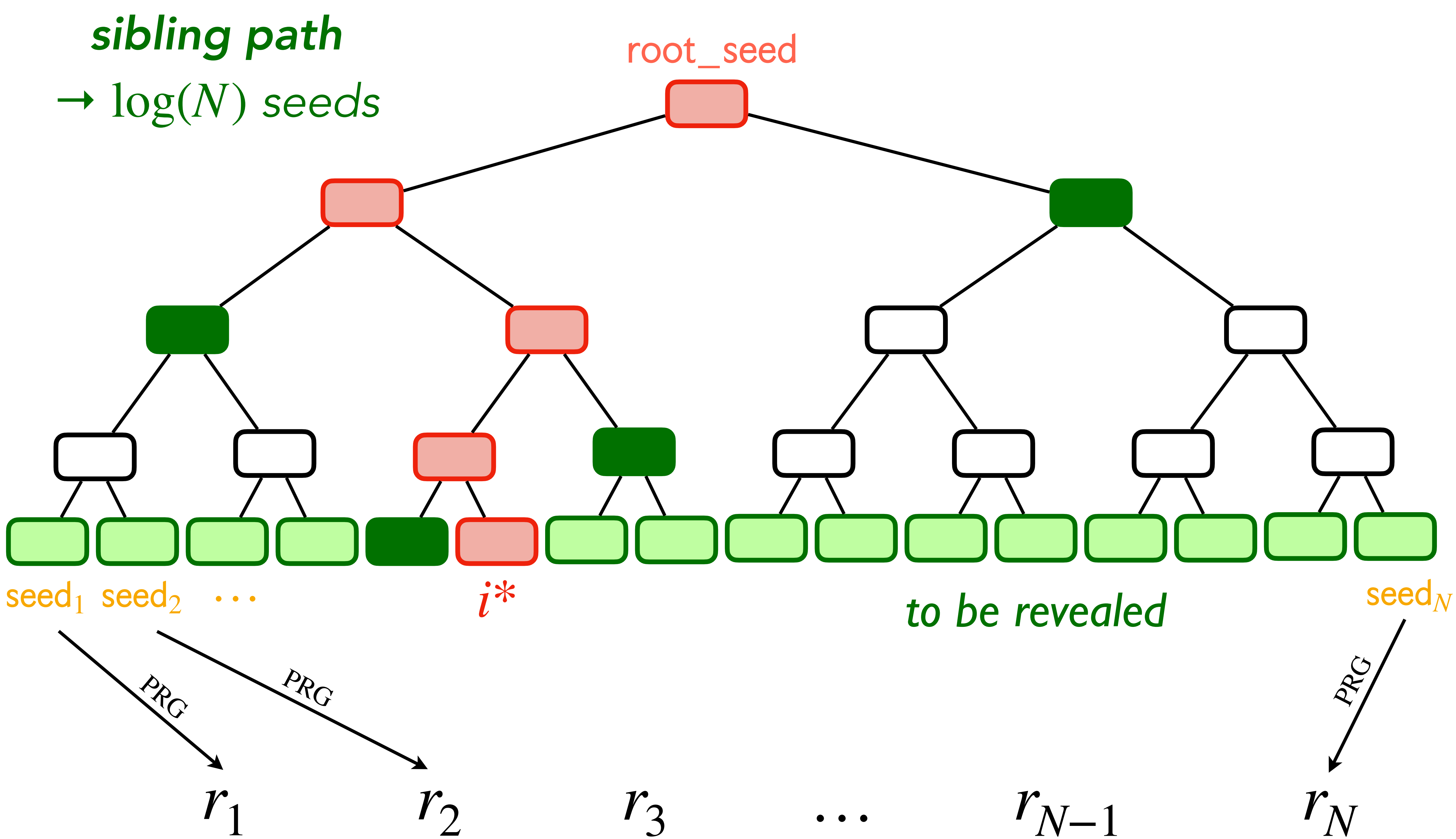
Committing to a Polynomial using a Seed Tree

[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

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 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 $(x_1, \ldots, x_{n'}, q_0, \ldots, q_{t-1})$ satisfies

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

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

with $x := (x_1, \ldots, x_{n'})$ 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 $(L, R) \in \mathbb{F}_q^{n \times r} \times \mathbb{F}^{r \times m}$ 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

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