The MPC-in-the-Head Paradigm for Post-Quantum Signatures: Recent Frameworks and Applications

Thibauld Feneuil

Seminar Crypto UCLouvain

January 27, 2026, *Louvain-la-Neuve*

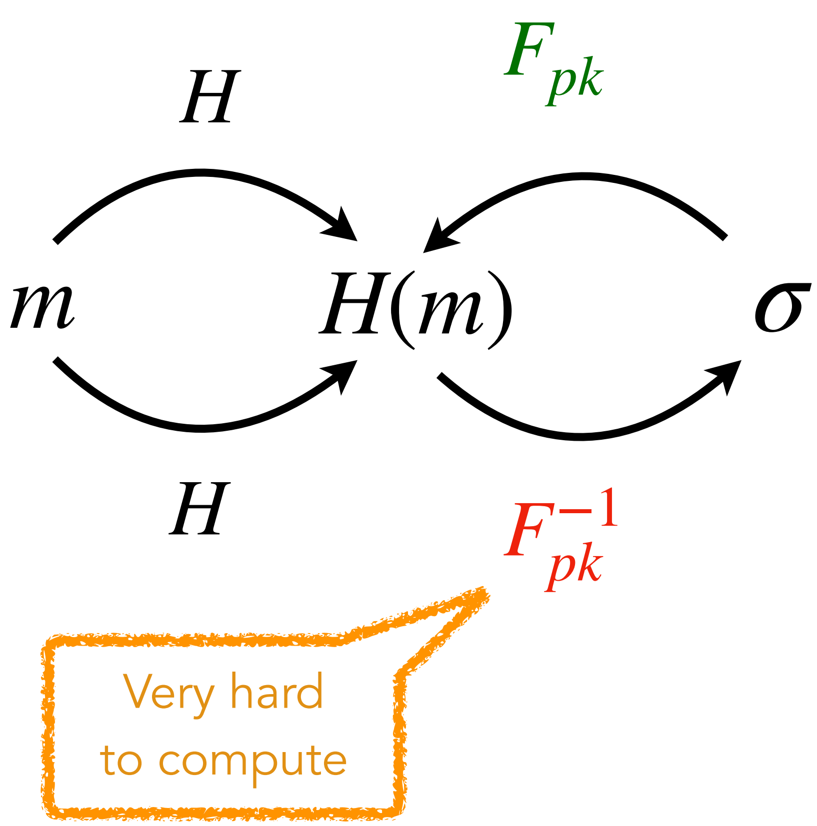
CryptoExperts
WE INNOVATE TO SECURE YOUR BUSINESS

Table of Contents

Introduction

How to build signature schemes?

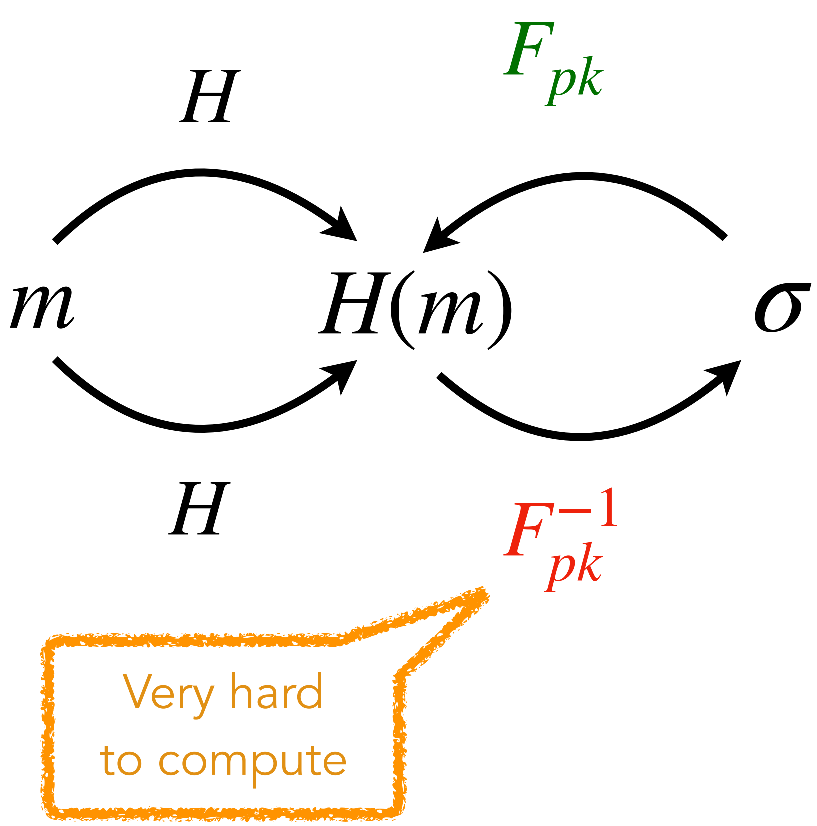
Hash & Sign



- Short signatures
- “Trapdoor” in the public key

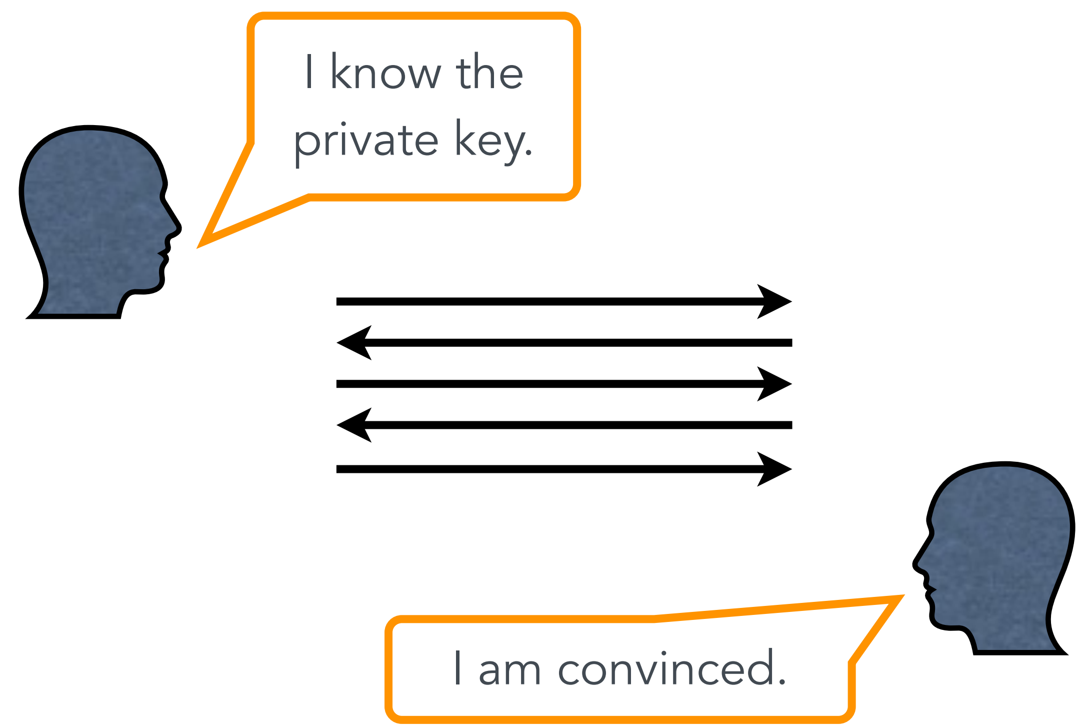
How to build signature schemes?

Hash & Sign



- Short signatures
- "Trapdoor" in the public key

From an identification scheme



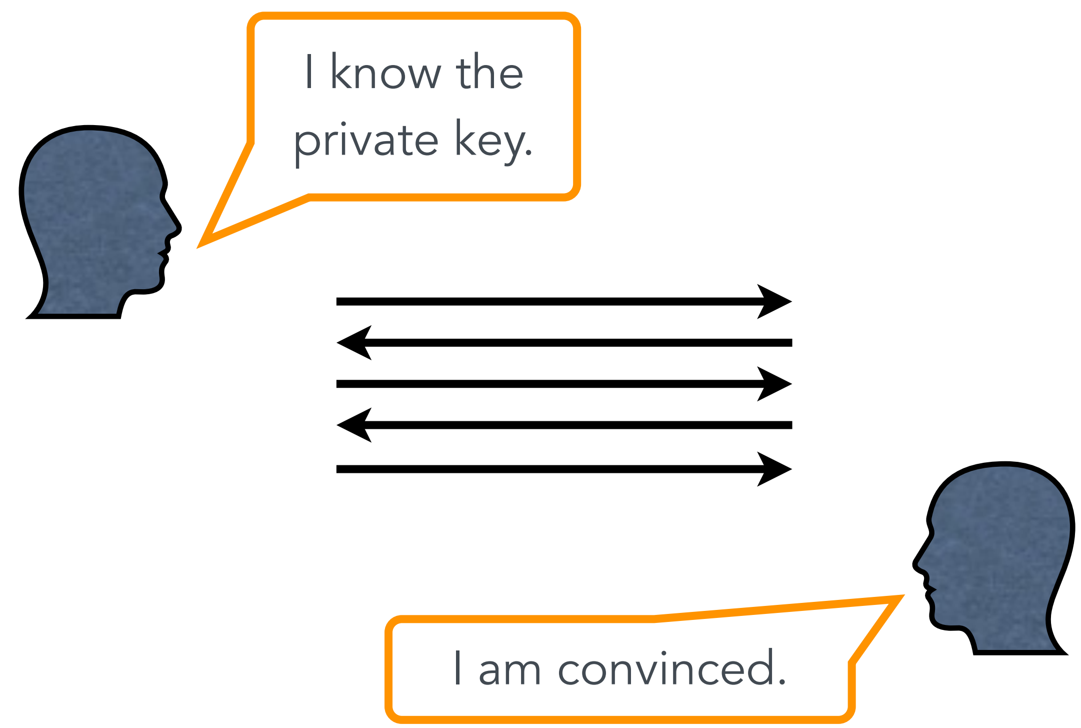
- Large(r) signatures
- Short public key

How to build signature schemes?

Hash & Sign

- Short signatures
- “Trapdoor” in the public key

From an identification scheme

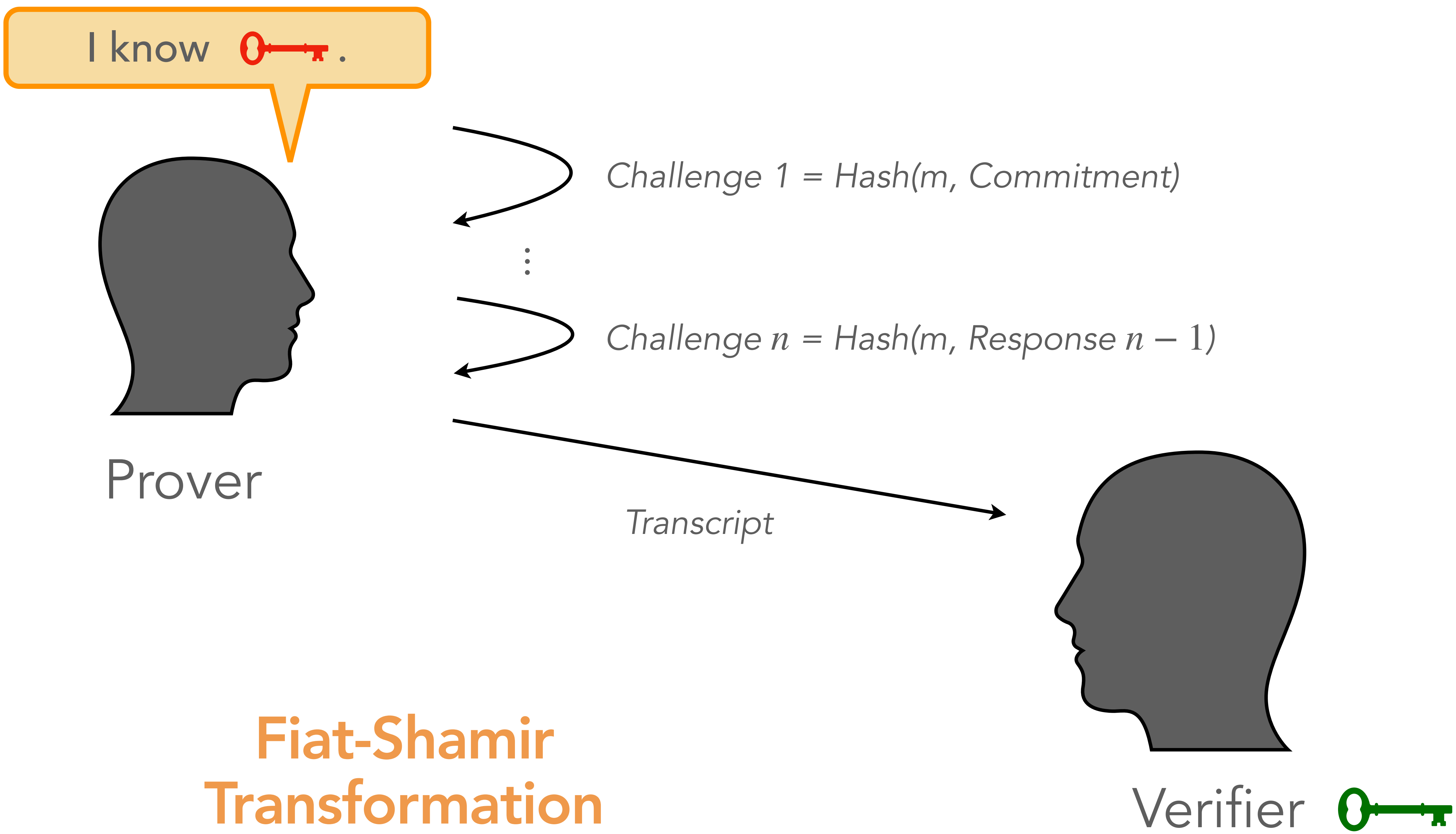


- Large(r) signatures
- Short public key

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

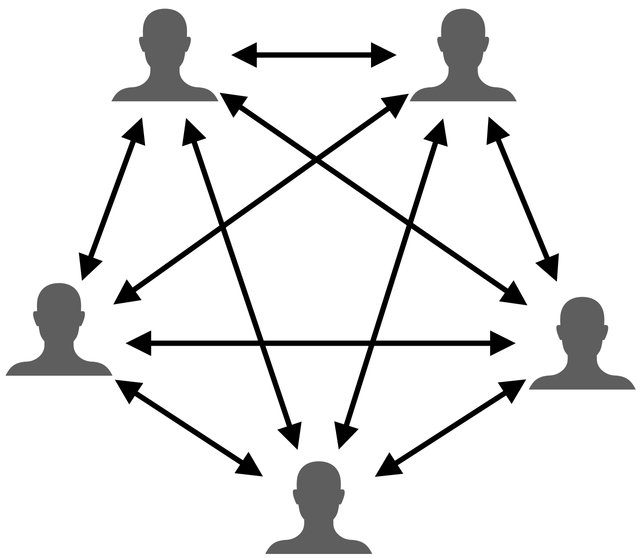
m: message to sign

MPCitH: general principle

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 candidates

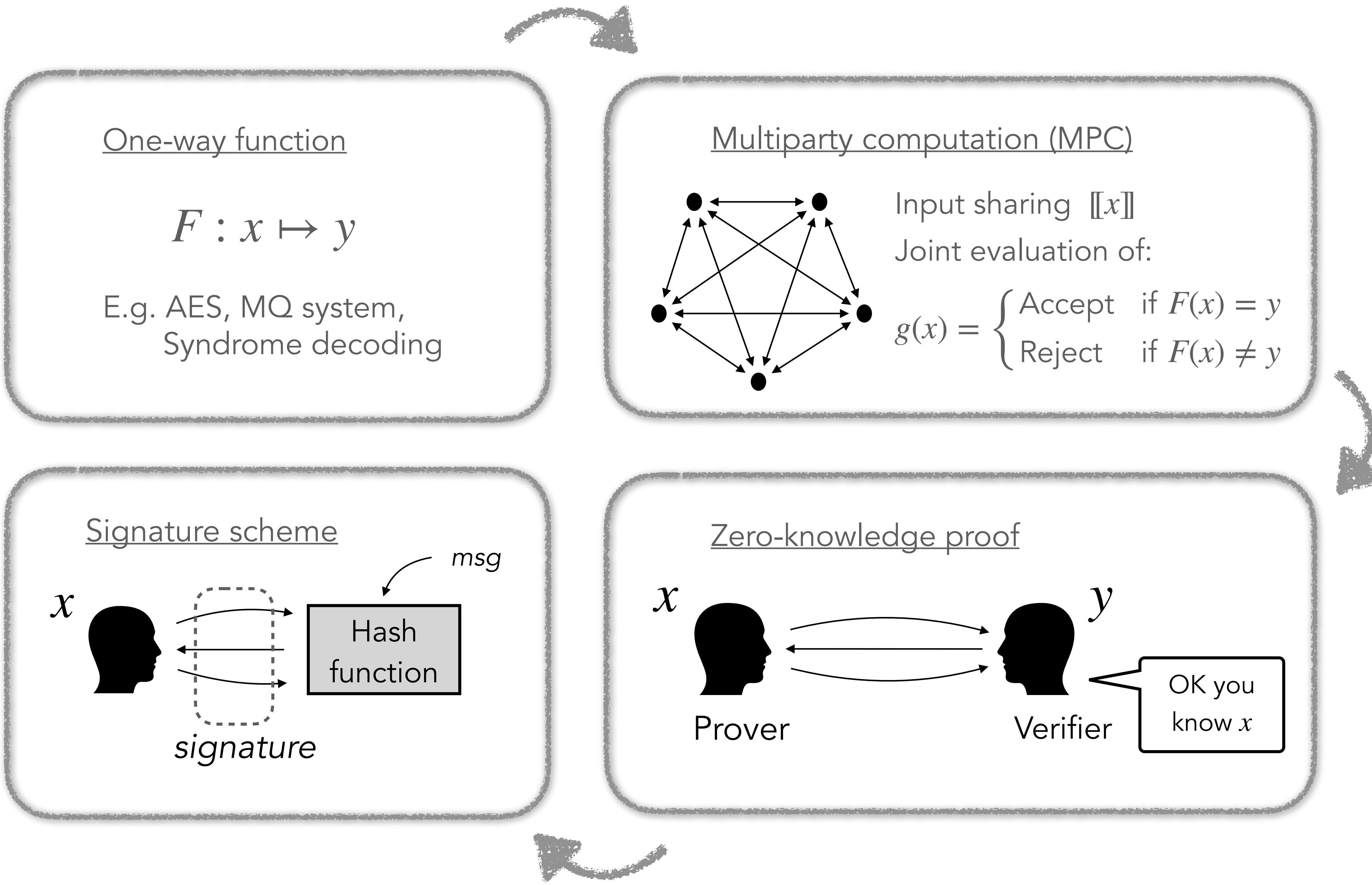
AIMer
Biscuit
FAEST
MIRA
MiRitH

MQOM
PERK
RYDE
SDitH

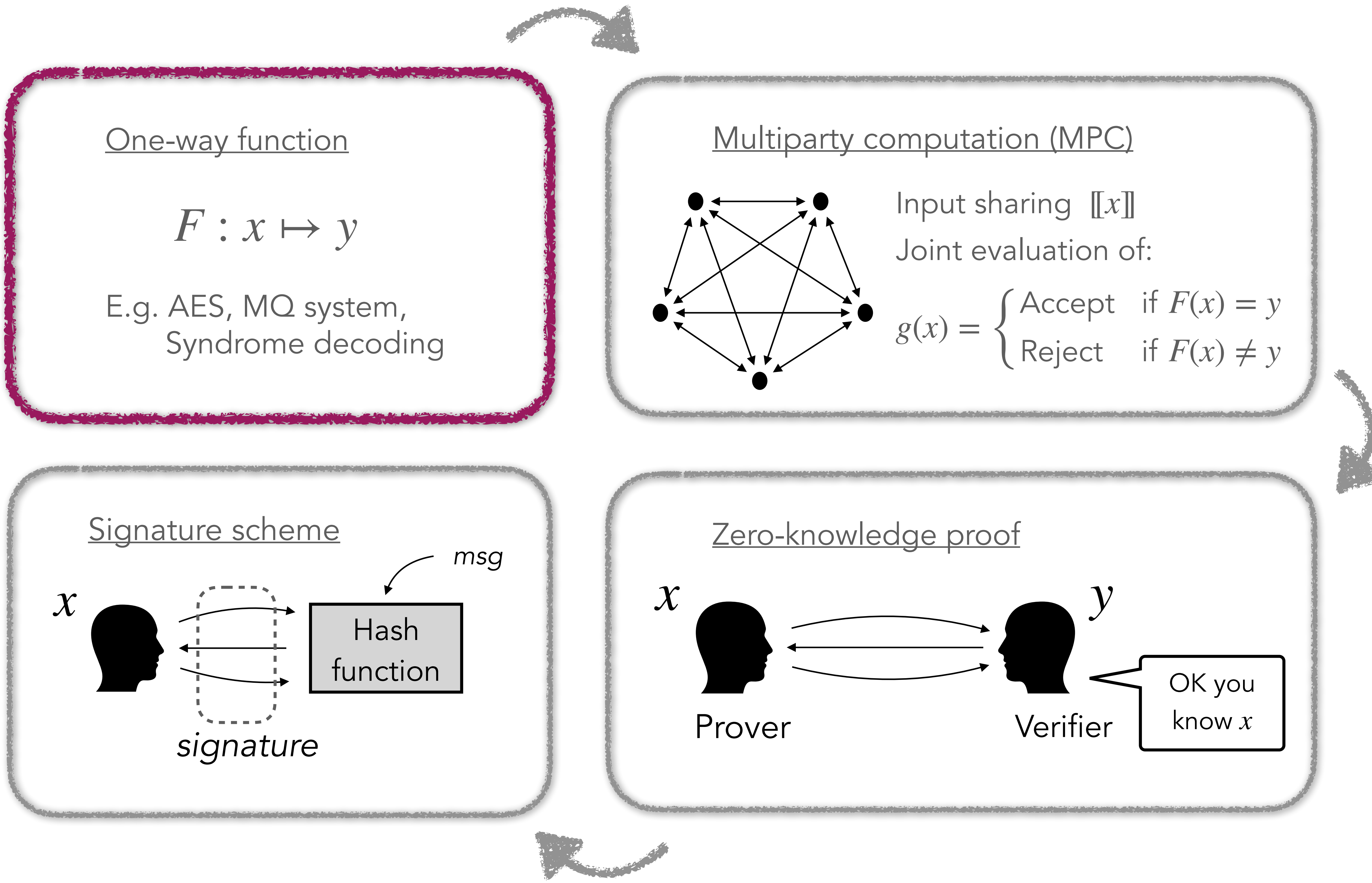
MPC in the Head

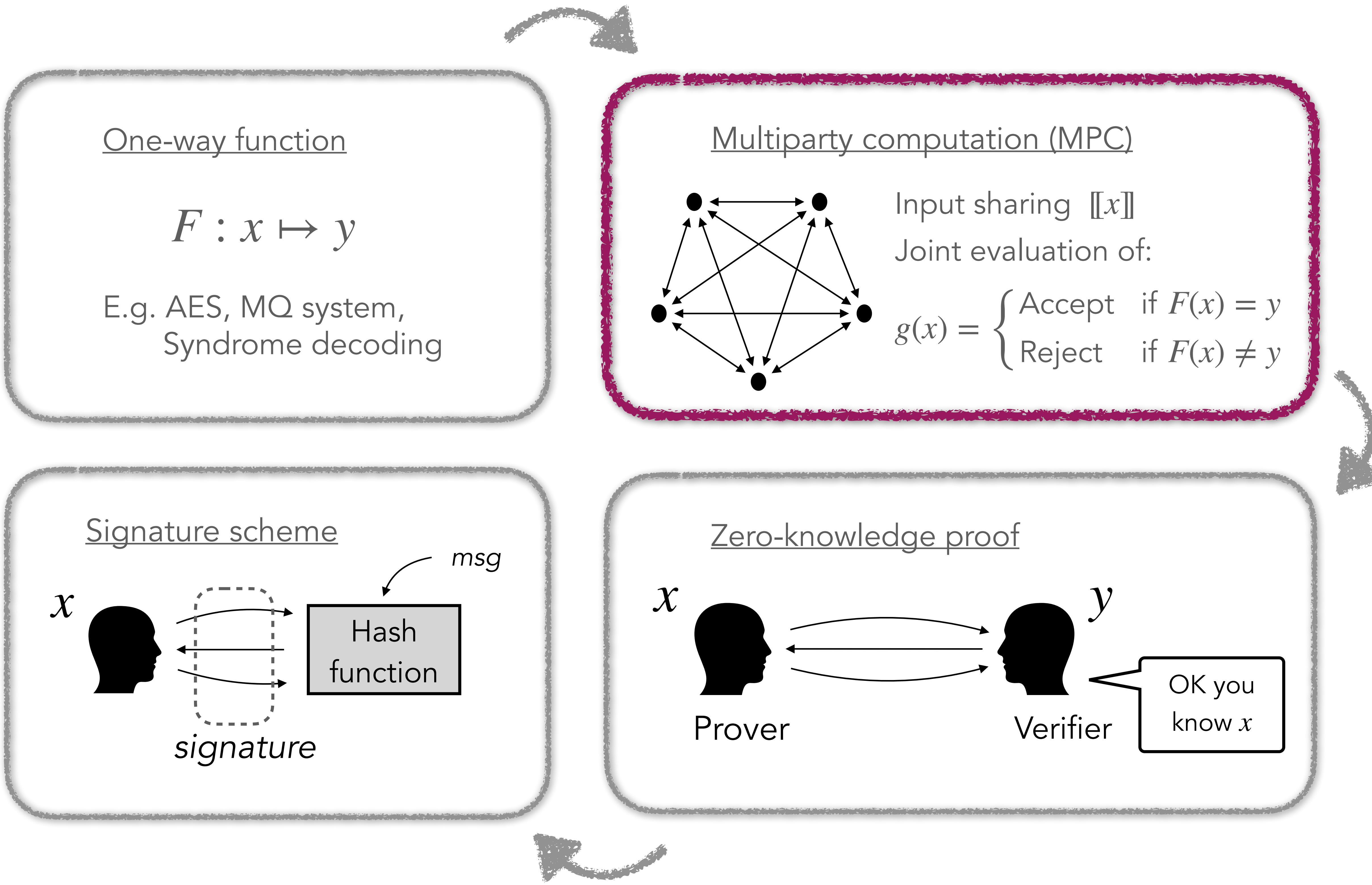
- **[IKOS07]** Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Amit Sahai: "Zero-knowledge from secure multiparty computation" (STOC 2007)
- Convenient to build (candidate) **post-quantum signature** schemes
- **Picnic**: submission to NIST (2017)
- First round of recent NIST call: 7~9 MPCitH schemes / 40 candidates
- Second round of recent NIST call: 5~6 MPCitH schemes / 14 candidates

FAEST	*PERK*
Mirath	*RYDE*
MQOM	*SDitH*

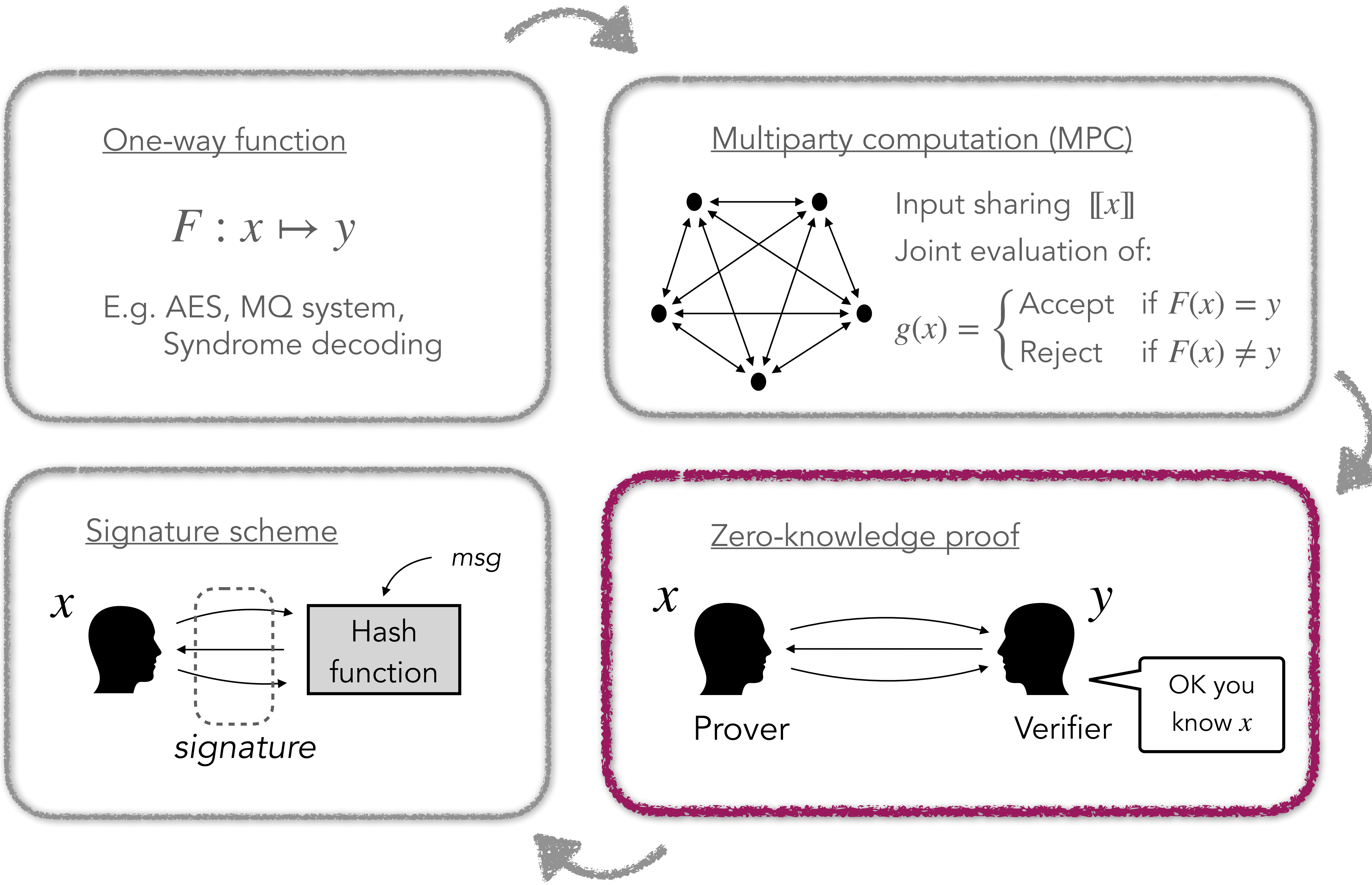


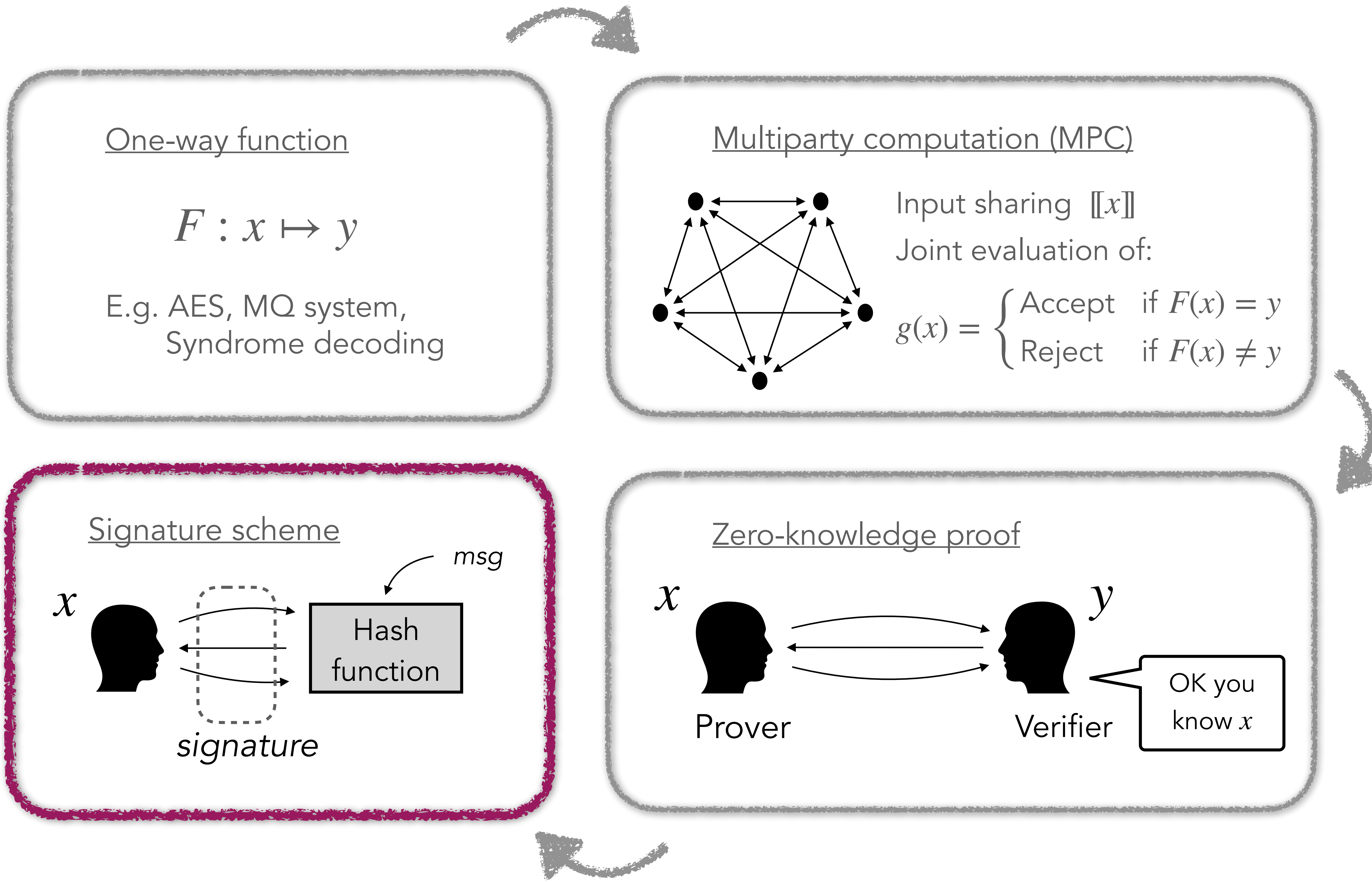
One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Signature scheme
msg
x
Hash
function
signature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

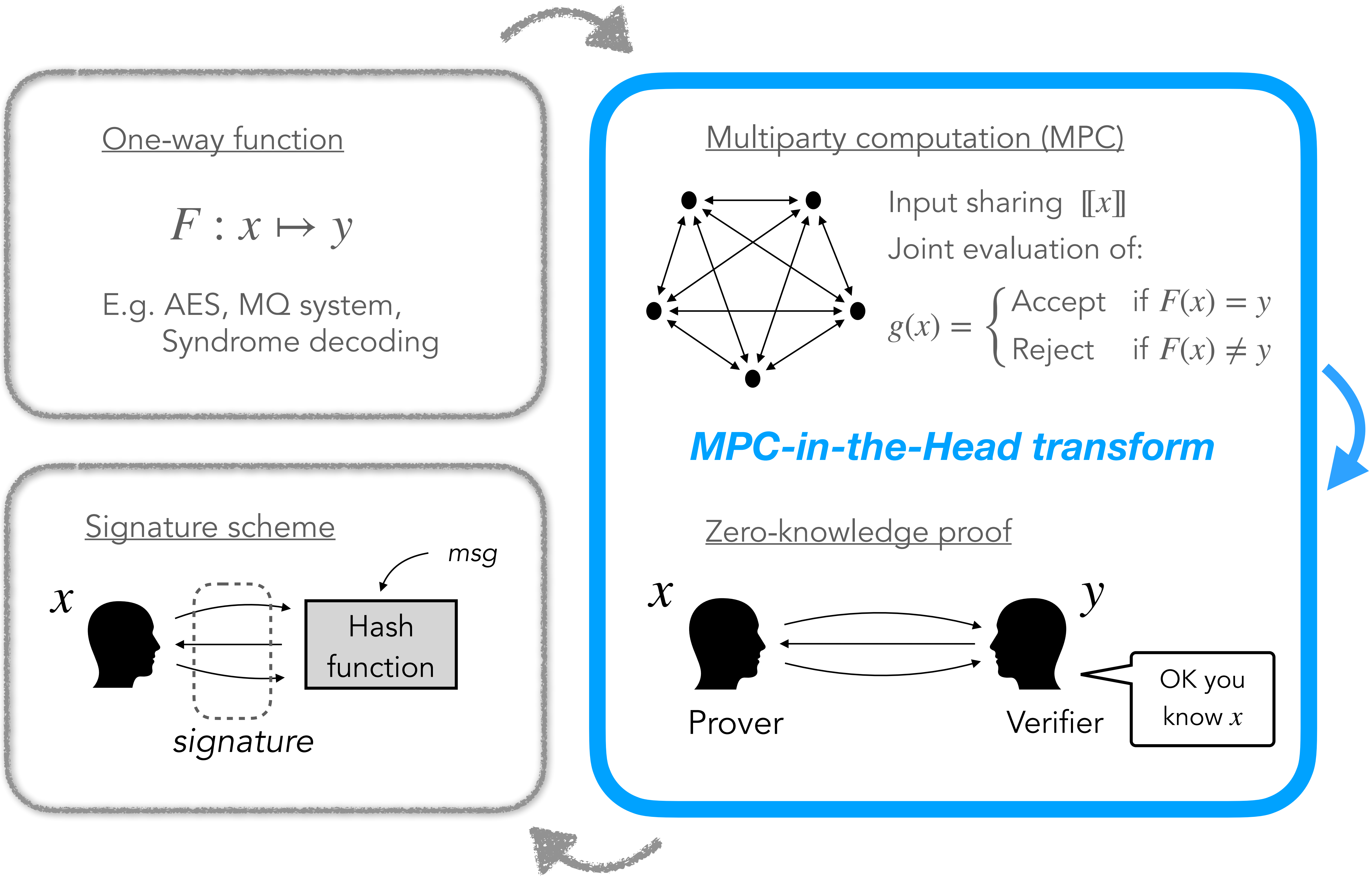

One-way function
$F : x \mapsto y$
E.g. AES, MQ system,
Syndrome decoding
Multiparty computation (MPC)
Input sharing $[\![x]\!]$
Joint evaluation of:
$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$
Zero-knowledge proof
x
y
Prover
Verifier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

MPC-in-the-Head Framework

Secret x which satisfies

some public relation $y = F(x)$

🤔 **How to build a zero-knowledge proof of knowledge for x?**

MPC-in-the-Head Framework

Secret x which satisfies

some public relation $y = F(x)$

↓

Sharing $[\![x]\!]$ of the secret x

Additive secret sharing:

$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_N$

Shamir's secret sharing:

$\forall i, [\![x]\!]_i = P(e_i)$,

where P is a random degree-ℓ polynomial such that $P(0) = x$.

MPC-in-the-Head Framework

Secret x which satisfies
some public relation $y = F(x)$

↓

Sharing $[\![x]\!]$ of the secret x

<u>Additive secret sharing</u>:

$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_N$

<u>Shamir's secret sharing</u>:

$\forall i, [\![x]\!]_i = P(e_i)$,
where P is a random degree-ℓ
polynomial such that $P(0) = x$.

MPC-in-the-Head Framework

Secret x which satisfies
some public relation $y = F(x)$

↓

Sharing $[\![x]\!]$ of the secret x

Additive secret sharing:

$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_N$

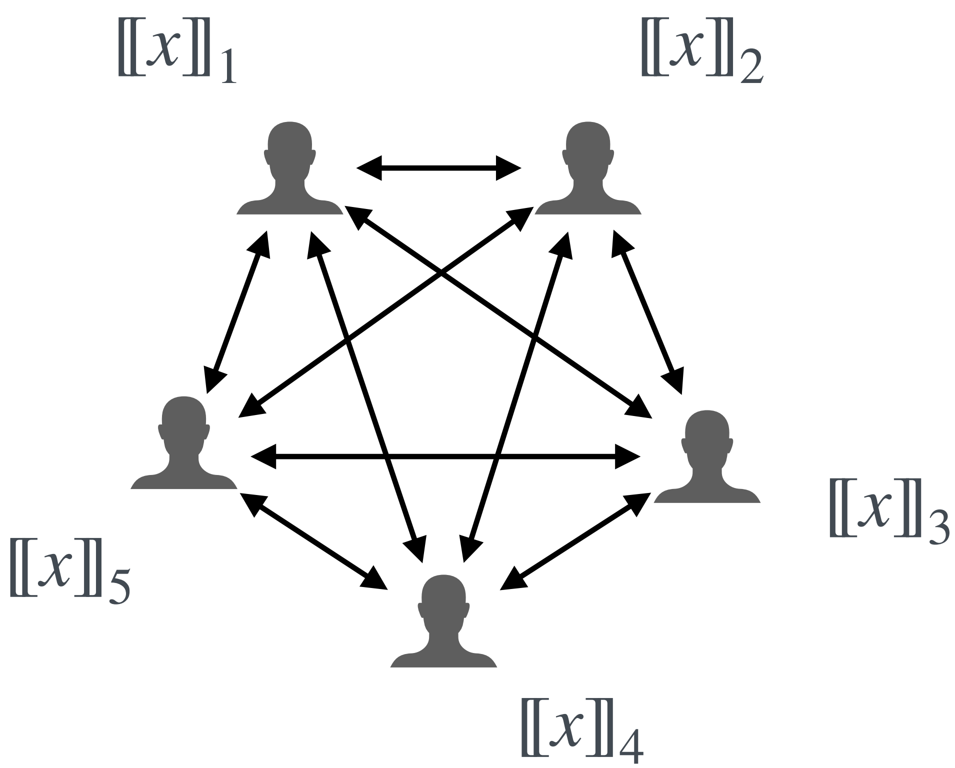
Shamir's secret sharing:

$\forall i, [\![x]\!]_i = P(e_i)$,
where P is a random degree-ℓ
polynomial such that $P(0) = x$.

If $x := 42$ lives in $\mathbb{F}_{1021}$, a possible sharing of x is

$$x = 429 + 19 + 583 + 231 + 822 \text{ over } \mathbb{F}_{1021}$$

MPC-in-the-Head Framework

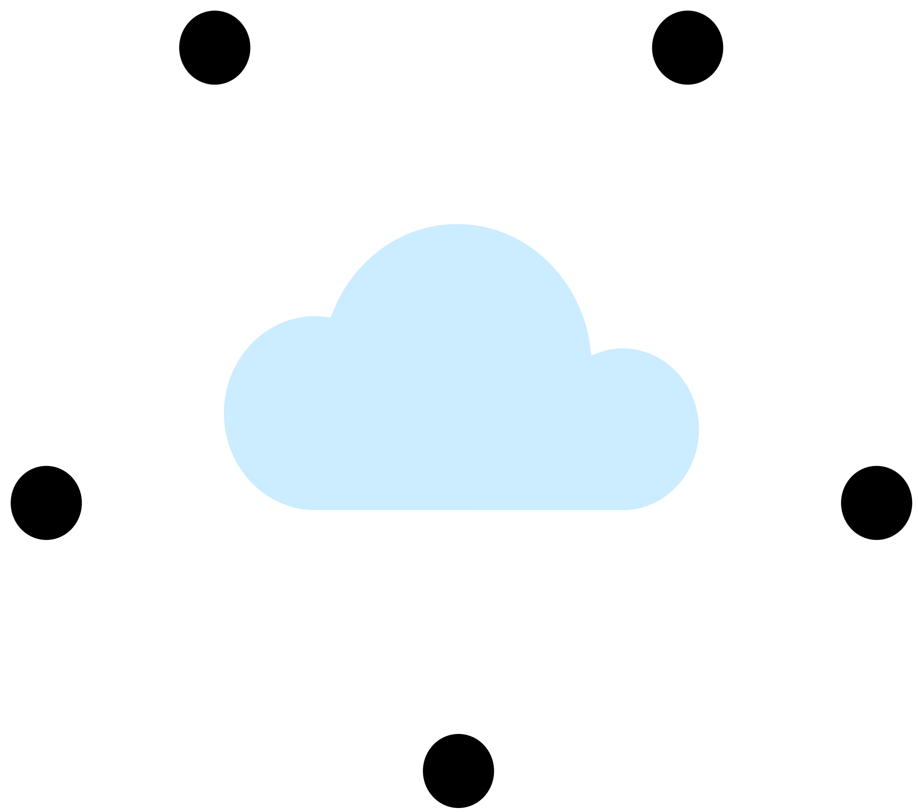


Input sharing $[\![x]\!]$

<u>Joint evaluation of</u>:

$$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$$

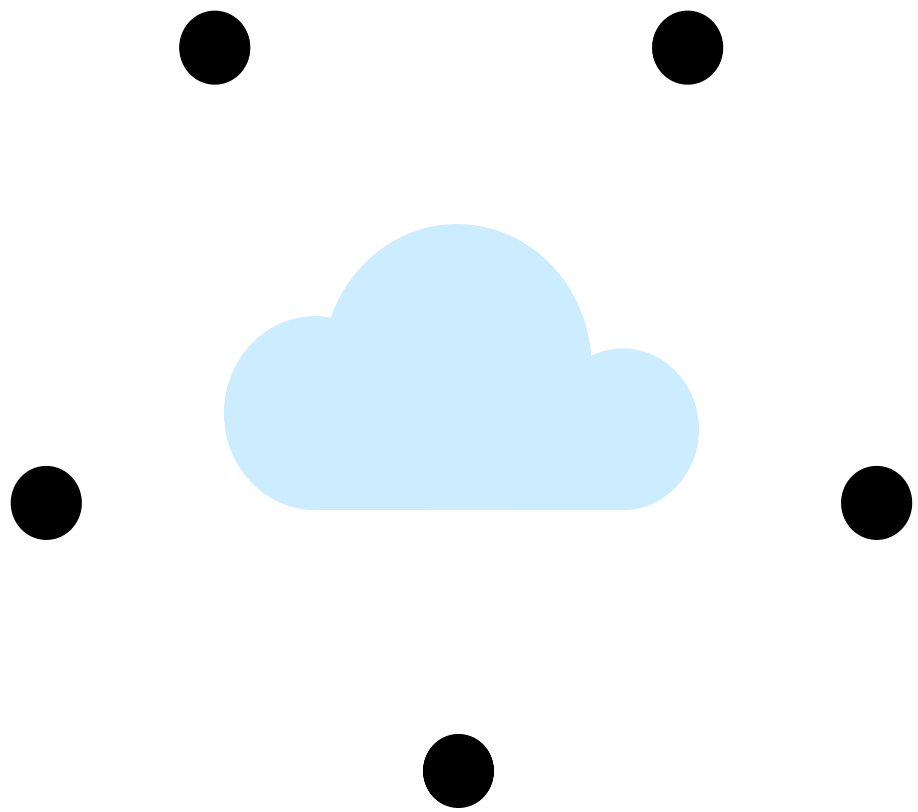
MPC model: discrete logarithm



- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

MPC model: discrete logarithm

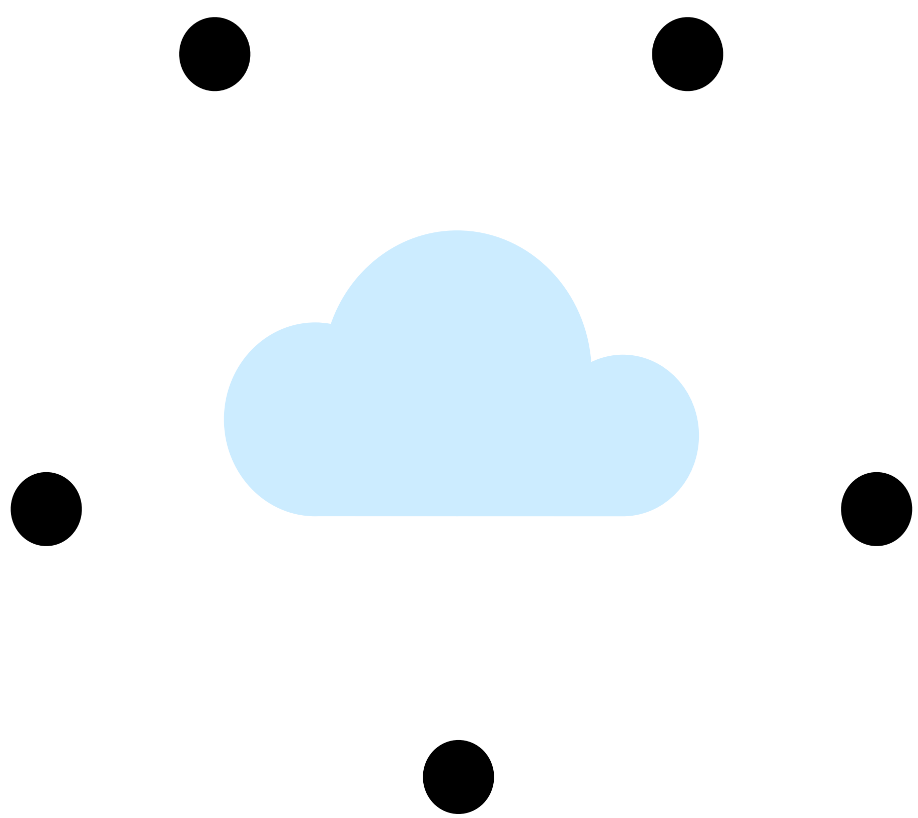


- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



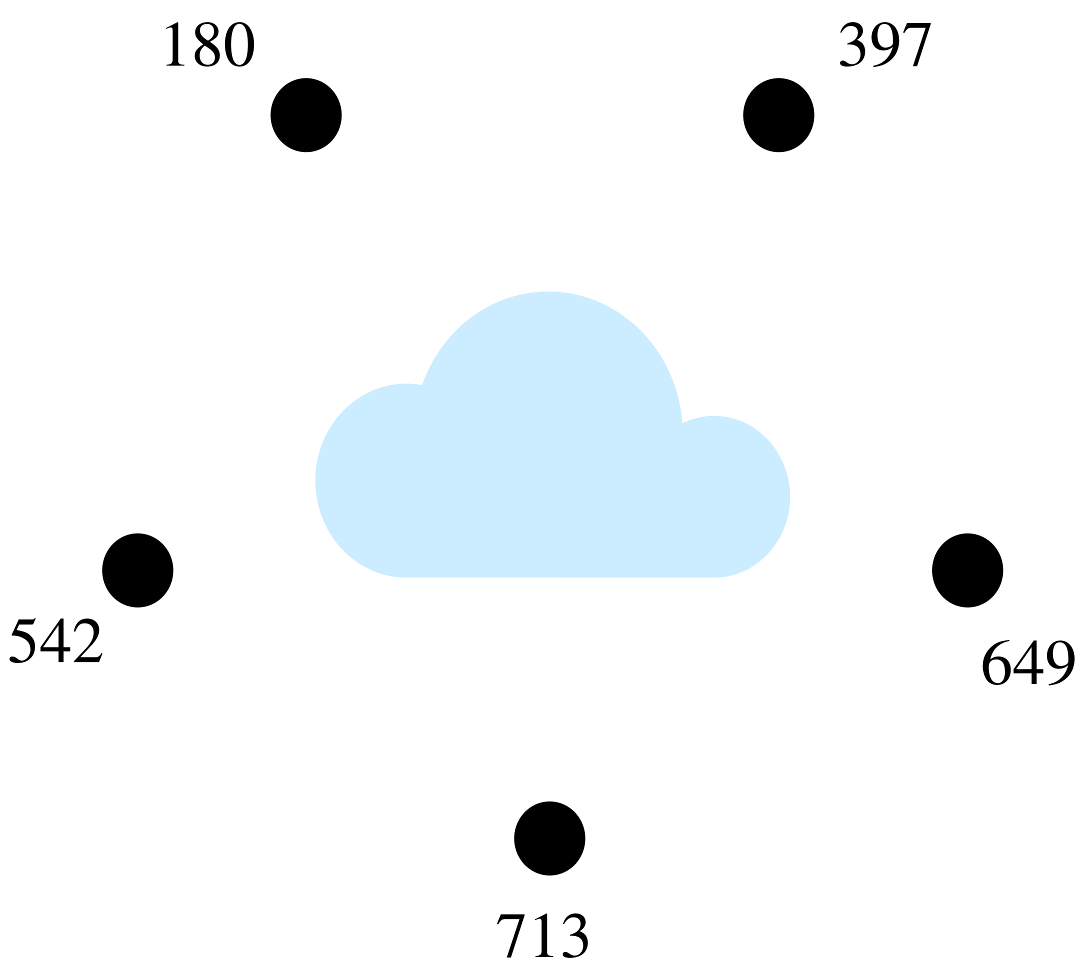
$z = 3 \pmod{1907}$ $x = 575$ $y = 1467 = z^x \pmod{1907}$

- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



$z = 3 \pmod{1907}$ $\quad x = 575 \quad$ $y = 1467 = z^x \pmod{1907}$

$[\![x]\!]_1 = 180, \quad [\![x]\!]_2 = 397, \quad [\![x]\!]_3 = 649, \quad [\![x]\!]_4 = 713, \quad [\![x]\!]_5 = 542$

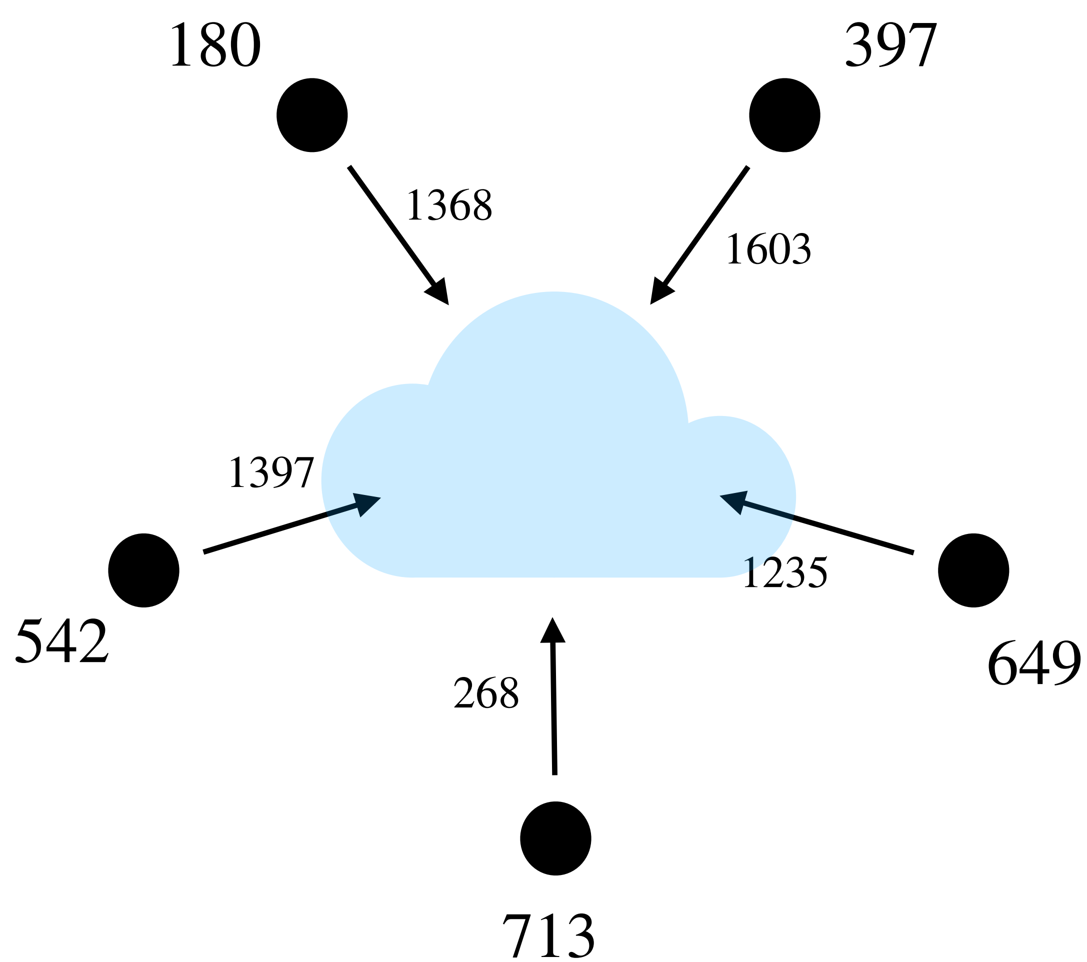
$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 \pmod{953}$$

- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



$z = 3 \pmod{1907}$ $\quad x = 575 \quad$ $y = 1467 = z^x \pmod{1907}$

$[\![x]\!]_1 = 180, \quad [\![x]\!]_2 = 397, \quad [\![x]\!]_3 = 649, \quad [\![x]\!]_4 = 713, \quad [\![x]\!]_5 = 542$

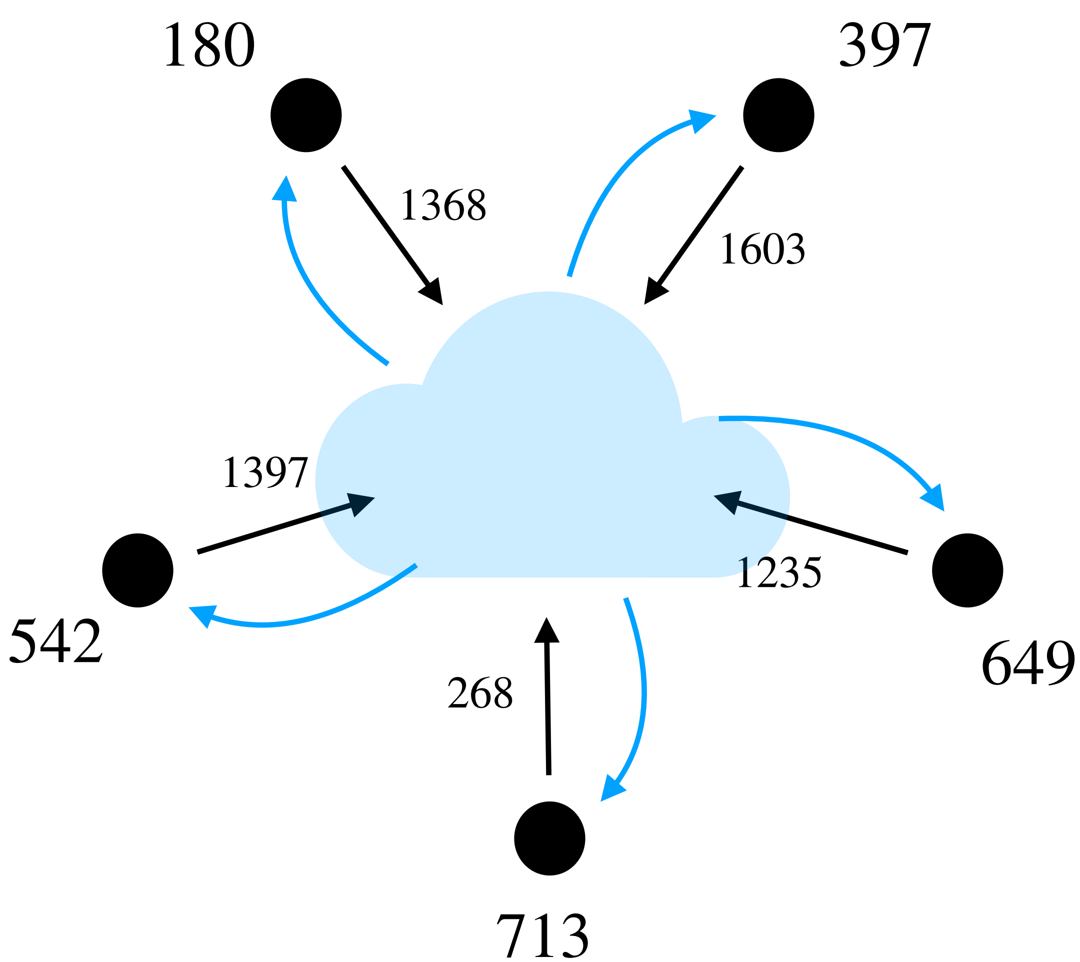
$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 \pmod{953}$$

- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



$z = 3 \pmod{1907}$ $\quad x = 575$ $\quad y = 1467 = z^x \pmod{1907}$

$[\![x]\!]_1 = 180, \quad [\![x]\!]_2 = 397, \quad [\![x]\!]_3 = 649, \quad [\![x]\!]_4 = 713, \quad [\![x]\!]_5 = 542$

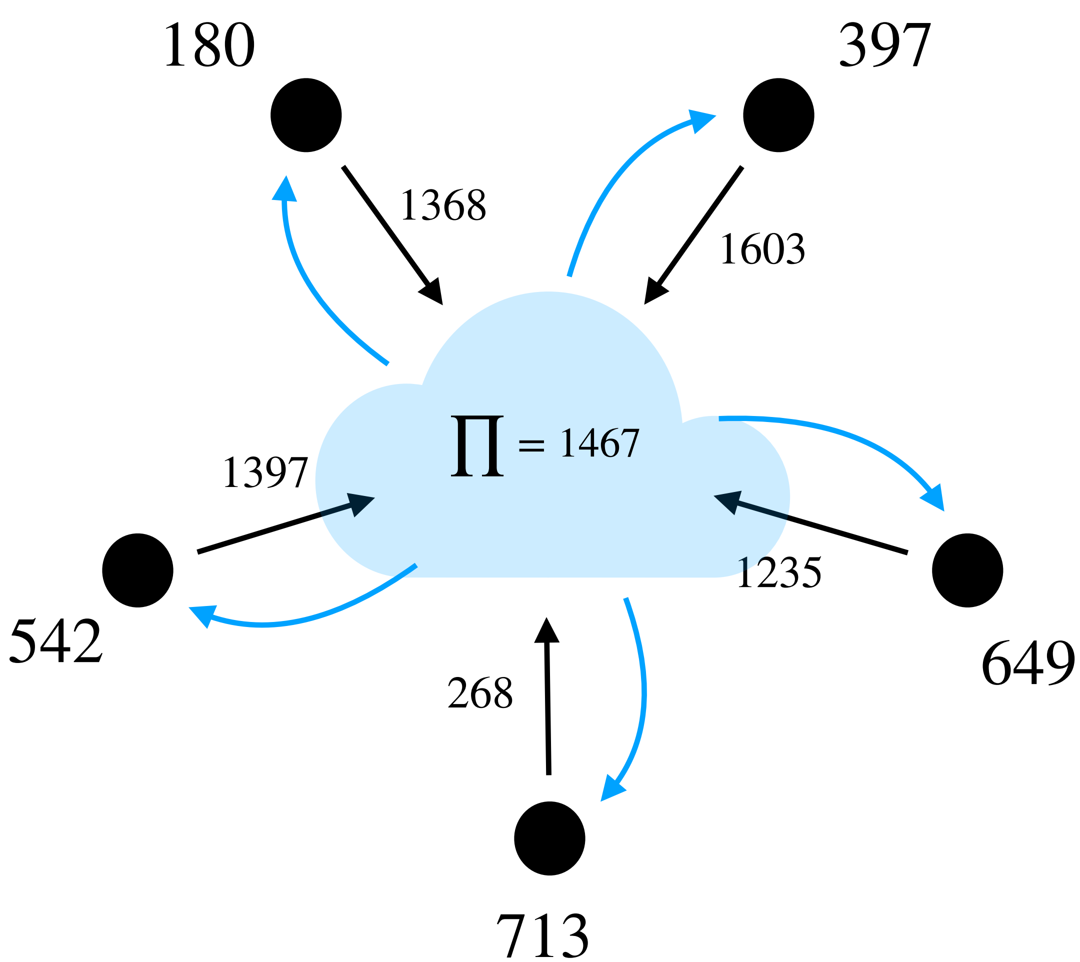
$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 \pmod{953}$$

- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



$z = 3 \pmod{1907}$ $\quad x = 575 \quad$ $y = 1467 = z^x \pmod{1907}$

$[\![x]\!]_1 = 180, \quad [\![x]\!]_2 = 397, \quad [\![x]\!]_3 = 649, \quad [\![x]\!]_4 = 713, \quad [\![x]\!]_5 = 542$

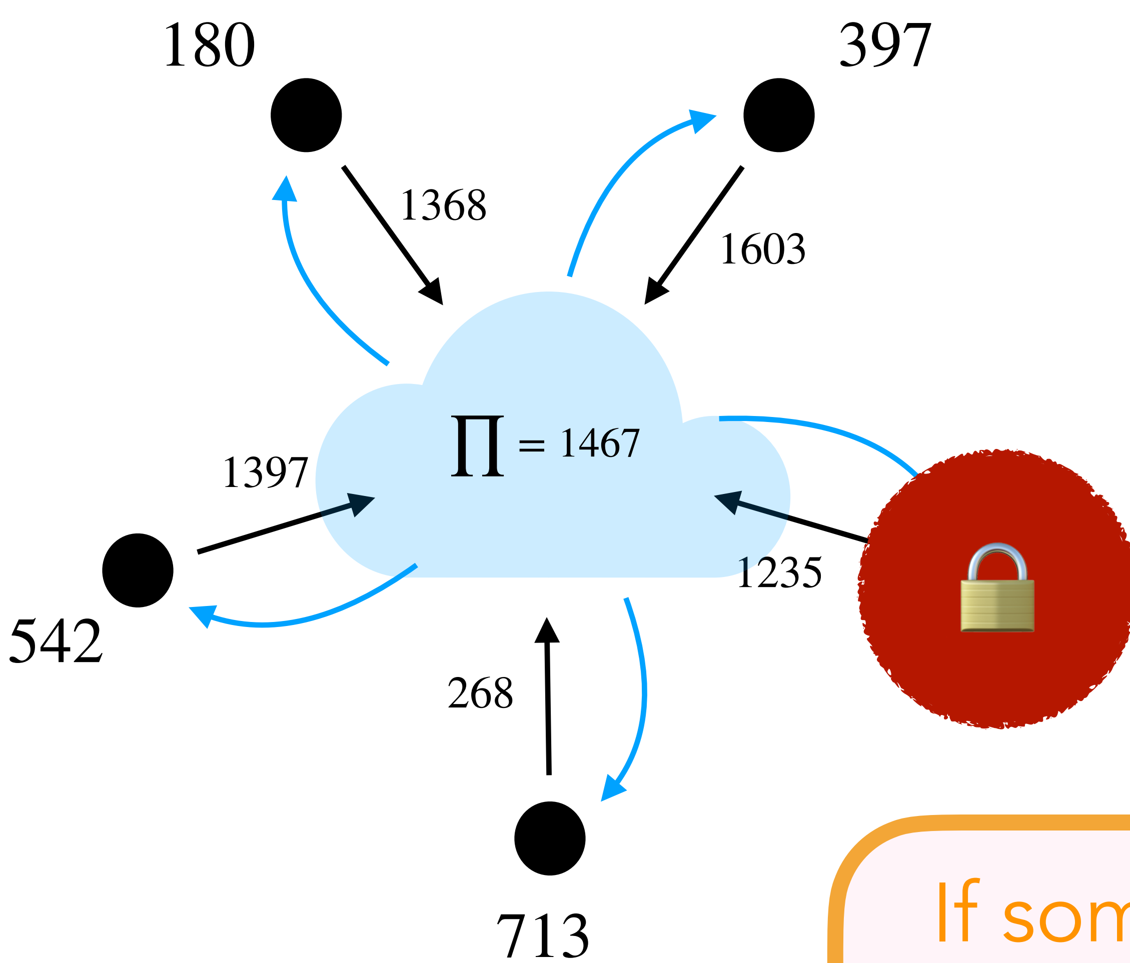
$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 \pmod{953}$$

- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.
  - Broadcast $[\![z^x]\!]_i$.
  - Receive all the broadcasted values $[\![z^x]\!]_1, \ldots, [\![z^x]\!]_N$
  - Recover z^x and check that y.

MPC model: discrete logarithm



- Secret x satisfies $y = z^x$, with z public.
- We want a multiparty computation that computes

$$g(x) = \begin{cases} \text{Accept} & \text{if } z^x = y \\ \text{Reject} & \text{if } z^x \neq y \end{cases}$$

- **Party i:**
  - Receive the i^{th} share $[\![x]\!]_i$
  - Compute $[\![z^x]\!]_i \leftarrow z^{[\![x]\!]_i}$.

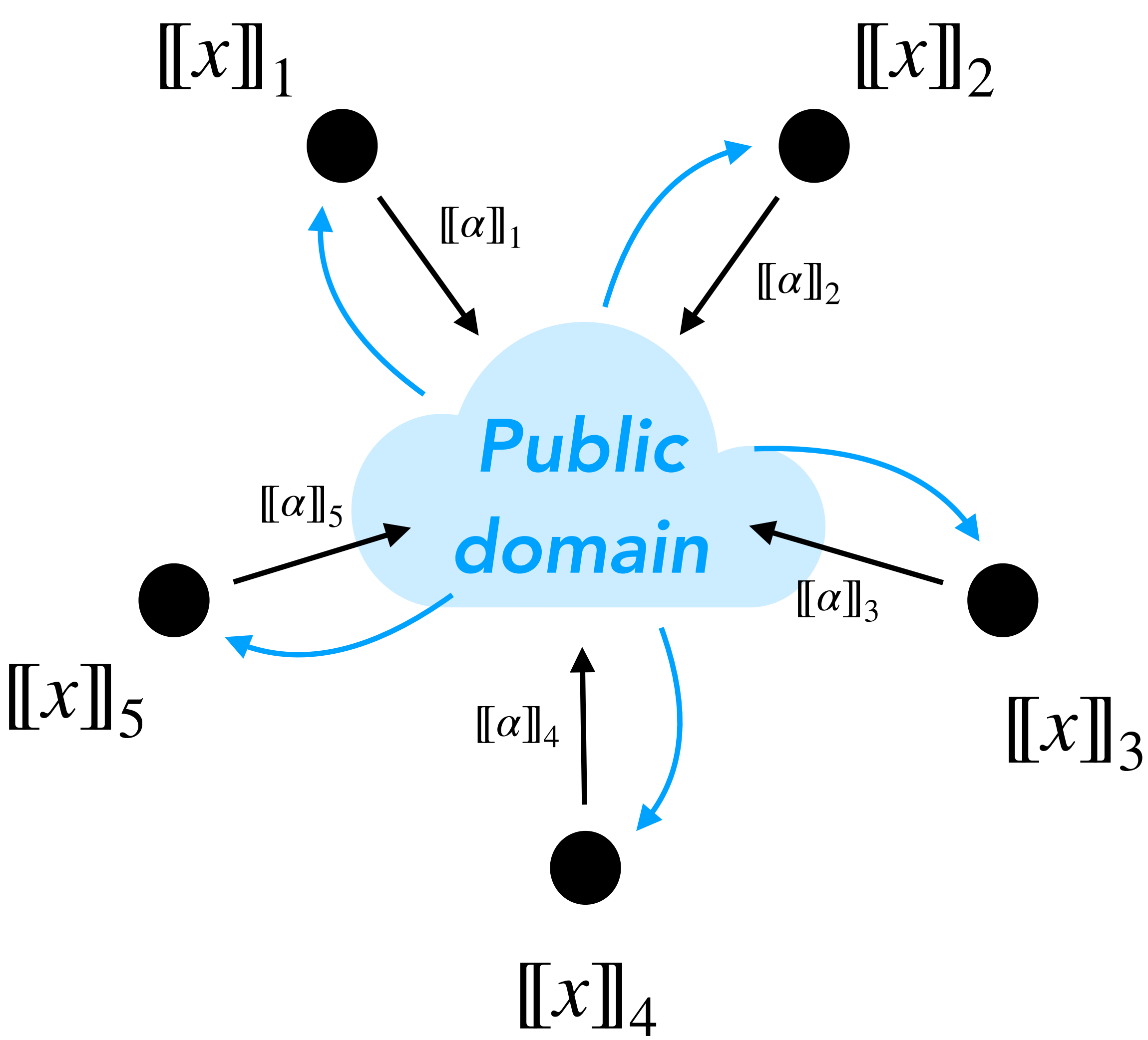
If someone sees the computation of all the parties except one, it leaks ***no information*** on x. 🧐

$z = 3 \pmod{1907}$ $\quad x =$ $\quad y = 1467$

$[\![x]\!]_1 = 180, \quad [\![x]\!]_2 = 397, \quad [\![x]\!]_3 =$, $\quad [\![x]\!]_4 = 7$

$$x = [\![x]\!]_1 + [\![x]\!]_2 + [\![x]\!]_3 + [\![x]\!]_4 + [\![x]\!]_5 \pmod{953}$$

MPC model



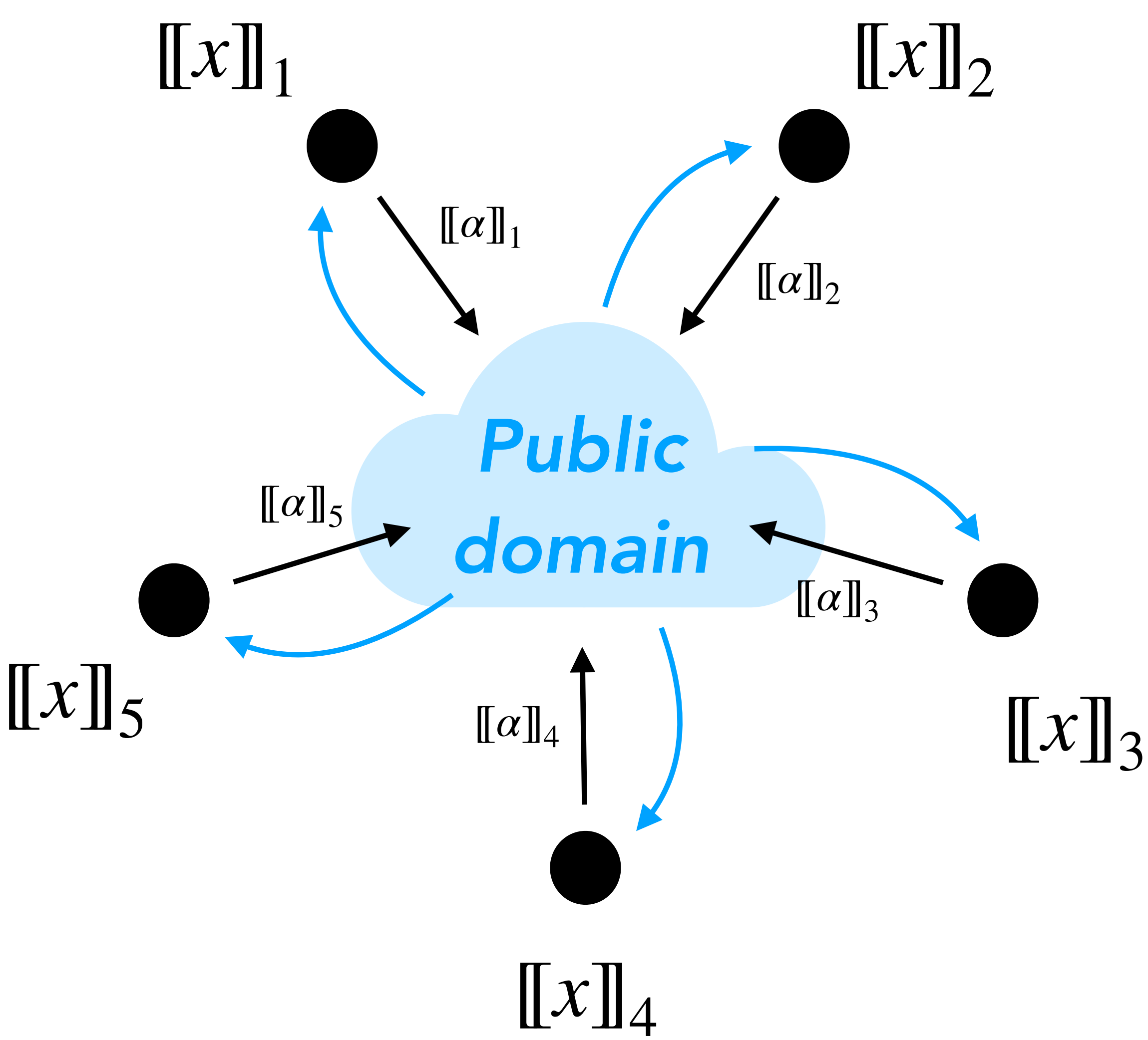
$$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_N$$

- **Jointly compute**

$$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$$

- $(N-1)$ **private:** the views of any $N-1$ parties provide no information on x
- **Semi-honest model:** assuming that the parties follow the steps of the protocol

MPC model



$$x = [\![x]\!]_1 + [\![x]\!]_2 + \ldots + [\![x]\!]_N$$

- **Jointly compute**

$$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$$

- $(N-1)$ **private:** the views of any $N-1$ parties provide no information on x
- **Semi-honest model:** assuming that the parties follow the steps of the protocol
- ***Broadcast model***
  - *Parties locally compute on their shares* $[\![x]\!] \mapsto [\![\alpha]\!]$
  - *Parties broadcast* $[\![\alpha]\!]$ *and recompute* α
  - *Parties start again (now knowing* α*)*

MPCitH transform

Prover

Verifier

MPCitH transform

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\cdots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

Prover

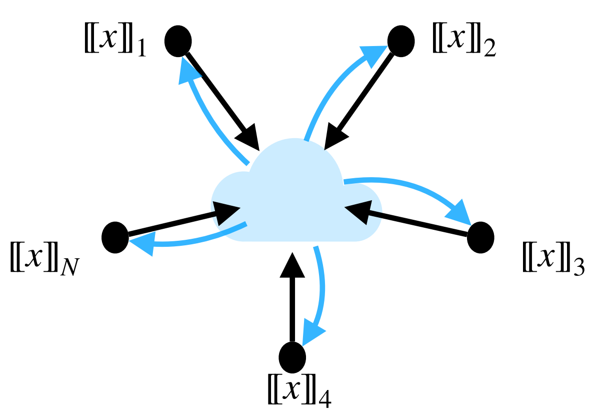
Verifier

MPCitH transform

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\ldots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$

Prover

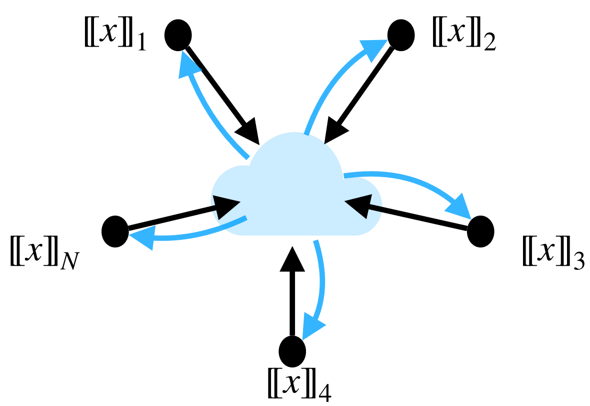
Verifier

MPCitH transform

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

…

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

→

send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$

→

i^*

←

③ Choose a random party

$$i^* \xleftarrow{\$} \{1, \ldots, N\}$$

Prover

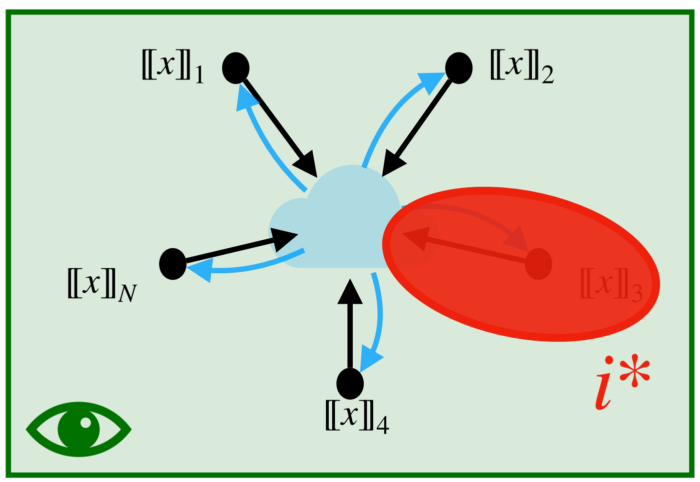
Verifier

MPCitH transform

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\} \backslash \{i^*\}$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$
…
$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$ →

send broadcast
$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$ →

← i^*

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$ →

③ Choose a random party

$$i^* \xleftarrow{\$} \{1,\ldots,N\}$$

Prover

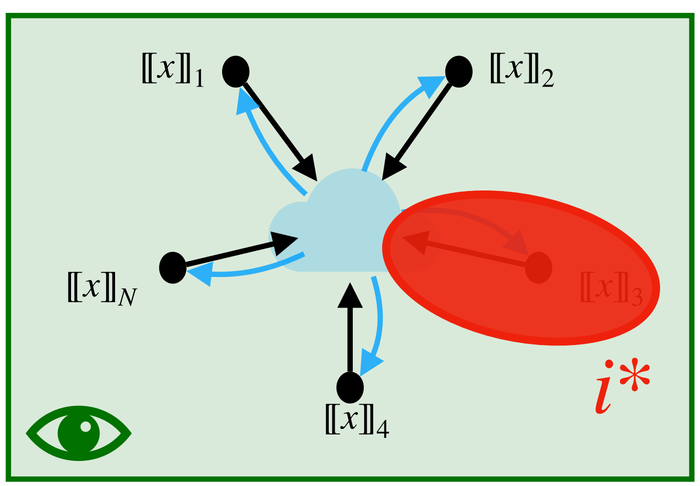
Verifier

MPCitH transform

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\}\backslash\{i^*\}$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$
$\cdots$
$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$ →

send broadcast
$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$ →

← i^*

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$ →

③ Choose a random party

$i^* \xleftarrow{\$} \{1,\ldots,N\}$

⑤ Check $\forall i \neq i^*$

- Commitments $\mathrm{Com}^{\rho_i}([\![x]\!]_i)$
- MPC computation $[\![\alpha]\!]_i = \varphi([\![x]\!]_i)$

Check $\tilde{g}(y, \alpha) =$ Accept

Prover

Verifier

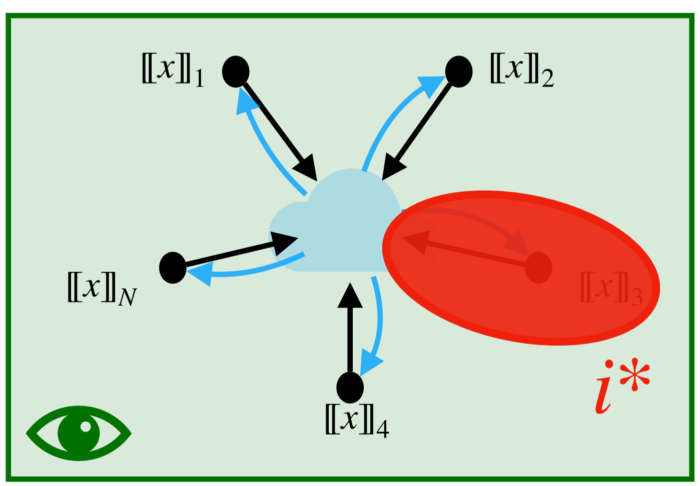
MPCitH transform

✅ Completeness
✅ Zero-Knowledge

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\}\backslash\{i^*\}$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$
…
$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$ →

send broadcast
$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$ →

← i^*

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$ →

③ Choose a random party

$$i^* \xleftarrow{\$} \{1,\ldots,N\}$$

⑤ Check $\forall i \neq i^*$

- Commitments $\mathrm{Com}^{\rho_i}([\![x]\!]_i)$
- MPC computation $[\![\alpha]\!]_i = \varphi([\![x]\!]_i)$

Check $\tilde{g}(y, \alpha) = \text{Accept}$

Prover

Verifier

MPCitH transform

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

We have $F(x) \neq y$ *where*

$x := [\![x]\!]_1 + \ldots + [\![x]\!]_N$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\cdots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

→

Malicious Prover

Verifier

MPCitH transform

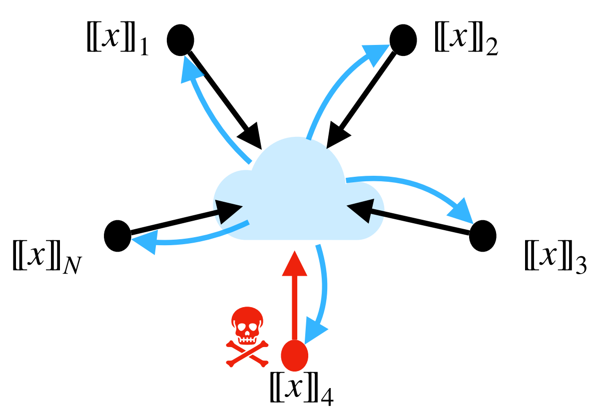
① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

We have $F(x) \neq y$ where

$x := [\![x]\!]_1 + \ldots + [\![x]\!]_N$

② Run MPC in their head



$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\cdots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$

Malicious Prover

Verifier

MPCitH transform

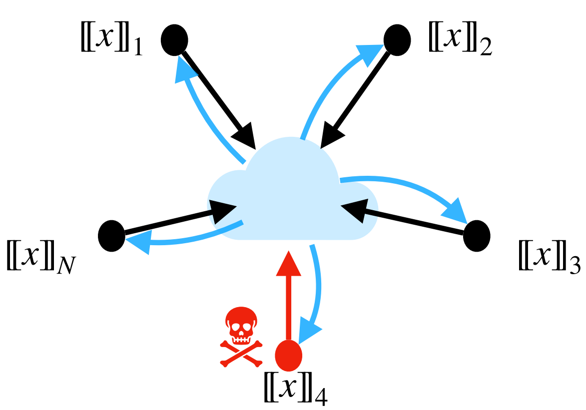
① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

We have $F(x) \neq y$ where

$x := [\![x]\!]_1 + \ldots + [\![x]\!]_N$

② Run MPC in their head



$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\cdots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

→

send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$

→

③ Choose a random party

$i^* \xleftarrow{\$} \{1, \ldots, N\}$

i^*

←

Malicious Prover

Verifier

MPCitH transform

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

We have $F(x) \neq y$ *where*

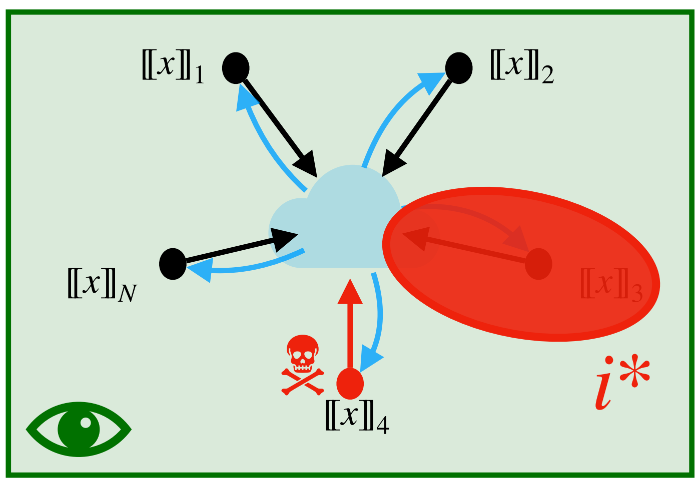
$x := [\![x]\!]_1 + \ldots + [\![x]\!]_N$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

...

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

② Run MPC in their head



send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$

③ Choose a random party

$i^* \xleftarrow{\$} \{1,\ldots,N\}$

i^*

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$

④ Open parties $\{1,\ldots,N\} \backslash \{i^*\}$

Malicious Prover

Verifier

MPCitH transform

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \dots, [\![x]\!]_N)$

We have $F(x) \neq y$ where

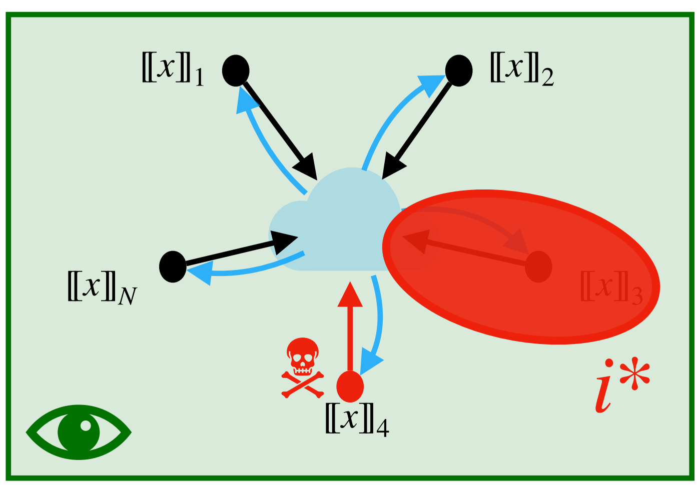
$x := [\![x]\!]_1 + \dots + [\![x]\!]_N$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

…

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$

② Run MPC in their head



send broadcast

$[\![\alpha]\!]_1, \dots, [\![\alpha]\!]_N$

③ Choose a random party

$i^* \xleftarrow{\$} \{1,\dots,N\}$

i^*

④ Open parties $\{1,\dots,N\} \backslash \{i^*\}$

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$

⑤ Check $\forall i \neq i^*$

- Commitments $\mathrm{Com}^{\rho_i}([\![x]\!]_i)$
- MPC computation $[\![\alpha]\!]_i = \varphi([\![x]\!]_i)$

Check $\tilde{g}(y, \alpha) =$ Accept

Malicious Prover

Verifier

Cheating detected!

MPCitH transform

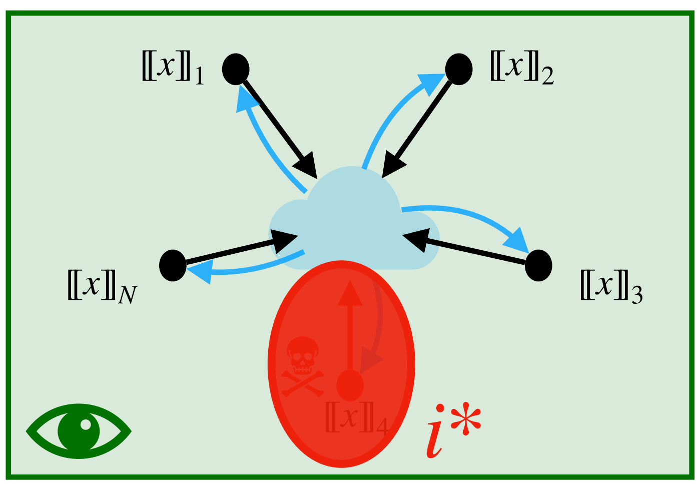
① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

We have $F(x) \neq y$ *where*

$x := [\![x]\!]_1 + \ldots + [\![x]\!]_N$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\}\backslash\{i^*\}$

$\mathrm{Com}^{\rho_1}([\![x]\!]_1)$

$\cdots$

$\mathrm{Com}^{\rho_N}([\![x]\!]_N)$ →

send broadcast

$[\![\alpha]\!]_1, \ldots, [\![\alpha]\!]_N$ →

← i^*

$([\![x]\!]_i, \rho_i)_{i \neq i^*}$ →

③ Choose a random party

$i^* \xleftarrow{\$} \{1,\ldots,N\}$

⑤ Check $\forall i \neq i^*$

- Commitments $\mathrm{Com}^{\rho_i}([\![x]\!]_i)$
- MPC computation $[\![\alpha]\!]_i = \varphi([\![x]\!]_i)$

Check $\tilde{g}(y, \alpha) = \text{Accept}$

Malicious Prover

Verifier

✔ **Seems OK.**

MPCitH transform

- **Zero-knowledge** $\iff$ MPC protocol is $(N-1)$-private

MPCitH transform

- **Zero-knowledge** $\iff$ MPC protocol is $(N-1)$-private
- **Soundness:**

$$
\begin{aligned}
&\mathbb{P}(\text{malicious prover convinces the verifier}) \\
&\qquad = \mathbb{P}(\text{corrupted party remains hidden}) \\
&\qquad = \frac{1}{N}
\end{aligned}
$$

MPCitH transform

- **Zero-knowledge** $\iff$ MPC protocol is $(N-1)$-private
- **Soundness:**

$$
\begin{aligned}
&\mathbb{P}(\text{malicious prover convinces the verifier}) \\
&\qquad = \mathbb{P}(\text{corrupted party remains hidden}) \\
&\qquad = \frac{1}{N}
\end{aligned}
$$

- **Parallel repetition**

Protocol repeated τ times in parallel → soundness error $\left(\frac{1}{N}\right)^{\tau}$

Polynomial Interactive Oracle Proof

PIOP-based MPCitH Frameworks

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

[FR25] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. Journal of Cryptology, 2025.

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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*

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

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

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

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

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

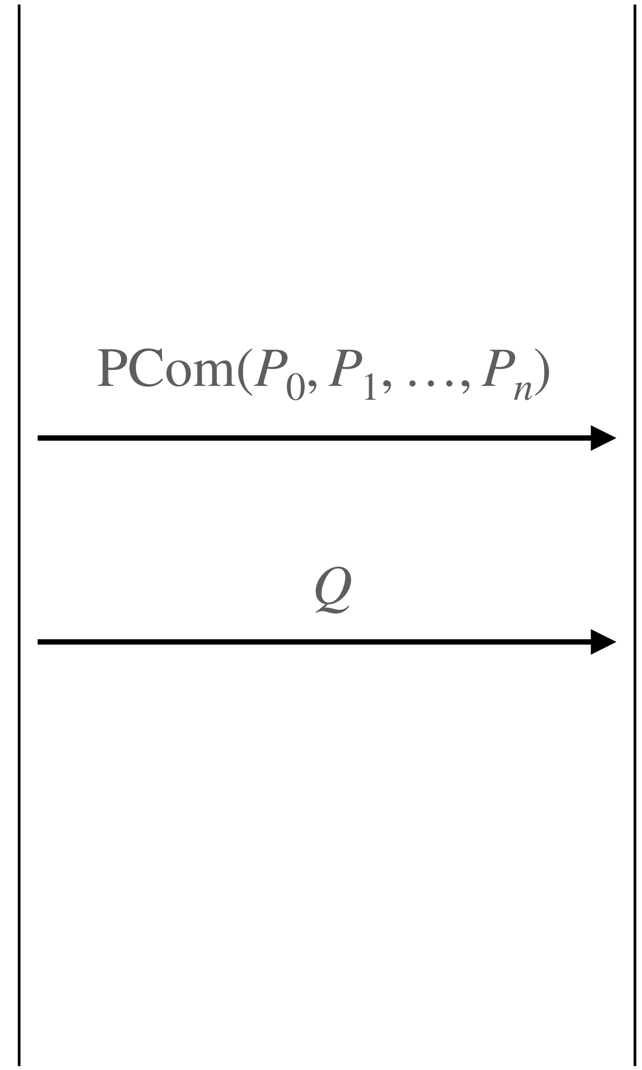
① For all i, sample a random degree-ℓ polynomial $P_i(X)$ such that $P_i(0) = w_i$

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$



Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q →

Well-defined!

$$0 \cdot P_0(0) + f(P_1(0), \ldots, P_n(0)) = 0 + f(w_1, \ldots, w_n) = 0$$

Prover | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q →

← $r_1, \ldots, r_\ell$

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Prover → Verifier: Q

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

Verifier → Prover: $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

Prover → Verifier: $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

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

Choose a degree-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

② Commit to the polynomials $P_0, P_1, \dots, P_n$ — $\text{PCom}(P_0, P_1, \dots, P_n)$ →

③ Reveal a polynomial $Q(X)$. We know that — Q →

$$X \cdot Q(X) \neq X \cdot P_0(X) + f(P_1(X), \dots, P_n(X))$$

④ Choose ℓ random evaluation points $r_1, \dots, r_\ell \in \mathscr{C} \subset \mathbb{F}$ — ← $r_1, \dots, r_\ell$

⑤ For all (i, j), reveal the evaluation — $\{v_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$ →

$$v_{j,i} := P_i(r_j)$$

⑥ Check that $\{v_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \dots, v_{j,n})$$

Malicious Prover 😈 | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

Malicious Prover 😈

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

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

Choose a degree-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

② Commit to the polynomials $P_0, P_1, \dots, P_n$

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + f(P_1(X), \dots, P_n(X))$$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

Evaluation into 0

$= 0$ $\neq 0$

Messages

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

Prover → Verifier: Q

Verifier → Prover: $r_1, \dots, r_\ell$

Prover → Verifier: $\{v_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$

Verifier

④ Choose ℓ random evaluation points $r_1, \dots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑥ Check that $\{v_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \dots, v_{j,n})$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

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

Choose a degree-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

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

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

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q →

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Malicious Prover 😈

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

Choose a degree-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

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

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

③ Reveal a polynomial $Q(X)$. We know that
$$X \cdot Q(X) \neq X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q →

← $r_1, \ldots, r_\ell$

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,
$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Verifier

Schwartz-Zippel Lemma: Let D be the **non-zero** degree-$(d \cdot \ell)$ polynomial defined as
$$D := X \cdot Q(X) - X \cdot P_0(X) - f(P_1(X), \ldots, P_n(X))$$

We have

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

Choose a degree-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

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

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

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

Q →

← $r_1, \ldots, r_\ell$

Soundness Analysis

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Verifier

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

$$D := X \cdot Q(X) - X \cdot P_0(X) - f(P_1(X), \ldots, P_n(X))$$

We have

$$\Pr[\text{verification passes}] = \Pr\left[\forall j, D(r_j) = 0 \mid \{r_j\}_j \subset_\$ \mathscr{C}\right] \leq \frac{\binom{d \cdot \ell}{\ell}}{\binom{|\mathscr{C}|}{\ell}}.$$

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

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

$\text{PCom}(P_0, P_1, \ldots, P_n)$ →

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q →

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Prover

Verifier 👀

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

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

Q →

← $r_1, \ldots, r_\ell$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

Zero-Knowledge Analysis

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Revealing ℓ evaluations of $P_i(X)$ leaks no information about w_i.

Verifier 👀

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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-$(d \cdot \ell - 1)$ polynomial $P_0(X)$

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

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

③ Reveal the polynomial $Q(X)$ such that

$$X \cdot Q(X) = X \cdot P_0(X) + f(P_1(X), \ldots, P_n(X))$$

Q

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

$r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} = P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

⑥ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + f(v_{j,1}, \ldots, v_{j,n})$$

Revealing $Q(X)$ leaks no information about w_i, thanks to $P_0(X)$.

Verifier 👀

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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 = $\frac{\binom{d \cdot \ell}{\ell}}{\binom{|S|}{\ell}}$

Probability that a malicious prover can convince the verifier.

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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*

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

Sample m random degree-$(d \cdot \ell)$ polynomials $\boldsymbol{P}_0(X) = \big(P_{0,1}(X), \ldots, P_{0,m}(X)\big)$

② Commit to the polynomials $\boldsymbol{P}_0, P_1, \ldots, P_n$

$\mathrm{PCom}(\boldsymbol{P}_0, P_1, \ldots, P_n)$ →

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

$$X \cdot Q_1(X) = X \cdot P_{0,1}(X) + f_1(P_1(X), \ldots, P_n(X))$$

$$\vdots$$

$$X \cdot Q_m(X) = X \cdot P_{0,m}(X) + f_m(P_1(X), \ldots, P_n(X))$$

$Q_1, \ldots, Q_m$ →

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑥ Check that $\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q_1(r_j) = r_j \cdot v_{j,0,1} + f_1(v_{j,1}, \ldots, v_{j,n})$$

$$\vdots$$

$$r_j \cdot Q_m(r_j) = r_j \cdot v_{j,0,m} + f_m(v_{j,1}, \ldots, v_{j,n}))$$

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

Sample m random degree-$(d \cdot \ell)$ polynomials $\boldsymbol{P}_0(X) = \big(P_{0,1}(X), \ldots, P_{0,m}(X)\big)$

② Commit to the polynomials $\boldsymbol{P}_0, P_1, \ldots, P_n$

→ $\mathrm{PCom}(\boldsymbol{P}_0, P_1, \ldots, P_n)$

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

$$X \cdot Q_1(X) = X \cdot P_{0,1}(X) + f_1(P_1(X), \ldots, P_n(X))$$
$$\vdots$$
$$X \cdot Q_m(X) = X \cdot P_{0,m}(X) + f_m(P_1(X), \ldots, P_n(X))$$

→ $Q_1, \ldots, Q_m$

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

→ $\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

⑥ Check that $\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q_1(r_j) = r_j \cdot v_{j,0,1} + f_1(v_{j,1}, \ldots, v_{j,n})$$
$$\vdots$$
$$r_j \cdot Q_m(r_j) = r_j \cdot v_{j,0,m} + f_m(v_{j,1}, \ldots, v_{j,n}))$$

Sigma/3-round variant of MQOM v2

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

Sample m random degree-$(d \cdot \ell)$ polynomials $\boldsymbol{P}_0(X) = \big(P_{0,1}(X), \dots, P_{0,m}(X)\big)$

② Commit to the polynomials $\boldsymbol{P}_0, P_1, \dots, P_n$

$\mathrm{PCom}(\boldsymbol{P}_0, P_1, \dots, P_n)$ →

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

$$X \cdot Q_1(X) = X \cdot P_{0,1}(X) + f_1(P_1(X), \dots, P_n(X))$$
$$\vdots$$
$$X \cdot Q_m(X) = X \cdot P_{0,m}(X) + f_m(P_1(X), \dots, P_n(X))$$

$Q_1, \dots, Q_m$ →

④ Choose ℓ random evaluation points $r_1, \dots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \dots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{\boldsymbol{v}_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$ →

⑥ Check that $\{\boldsymbol{v}_{j,0}, v_{j,1}, \dots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q_1(r_j) = r_j \cdot v_{j,0,1} + f_1(v_{j,1}, \dots, v_{j,n})$$
$$\vdots$$
$$r_j \cdot Q_m(r_j) = r_j \cdot v_{j,0,m} + f_m(v_{j,1}, \dots, v_{j,n}))$$

A bit costly!

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

Sample m random degree-$(d \cdot \ell)$ polynomials $\boldsymbol{P}_0(X) = \big(P_{0,1}(X), \ldots, P_{0,m}(X)\big)$

② Commit to the polynomials $\boldsymbol{P}_0, P_1, \ldots, P_n$

$\mathrm{PCom}(\boldsymbol{P}_0, P_1, \ldots, P_n)$ →

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

$$X \cdot Q_1(X) = X \cdot P_{0,1}(X) + f_1(P_1(X), \ldots, P_n(X))$$
$$\vdots$$
$$X \cdot Q_m(X) = X \cdot P_{0,m}(X) + f_m(P_1(X), \ldots, P_n(X))$$

$Q_1, \ldots, Q_m$ →

④ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑤ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑥ Check that $\{\boldsymbol{v}_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q_1(r_j) = r_j \cdot v_{j,0,1} + f_1(v_{j,1}, \ldots, v_{j,n})$$
$$\vdots$$
$$r_j \cdot Q_m(r_j) = r_j \cdot v_{j,0,m} + f_m(v_{j,1}, \ldots, v_{j,n}))$$

A bit costly!

Solution: batching

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

Prover | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

Q →

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

Q →

Well-defined!

$$\sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(0), \ldots, P_n(0)) = \sum_{k=1}^{m} \gamma_k \cdot f_k(w_1, \ldots, w_n)$$

$$= \sum_{k=1}^{m} \gamma_k \cdot 0 = 0$$

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

→ Q

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

→ $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

→ Q

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

→ $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + \sum_{k=1}^{m} \gamma_k \cdot f_k(v_{j,1}, \ldots, v_{j,n})$$

Prover

Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

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

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

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

$\text{PCom}(P_0, P_1, \ldots, P_n)$ →

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

Q →

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + \sum_{k=1}^{m} \gamma_k \cdot f_k(v_{j,1}, \ldots, v_{j,n})$$

Prover | Verifier

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

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

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

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

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

← $\gamma_1, \ldots, \gamma_m$

③ Choose random coefficients

$$\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$$

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

Q →

← $r_1, \ldots, r_\ell$

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ →

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the ... $v_{j,n})$

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

$$\sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(0), \ldots, P_n(0)) \neq 0$$

Prover

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

Soundness Analysis

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

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

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

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

$\xrightarrow{\mathrm{PCom}(P_0, P_1, \ldots, P_n)}$

③ Choose random coefficients

$$\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$$

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

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

$$X \cdot Q(X) \neq X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

$\xrightarrow{Q}$

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

$\xleftarrow{r_1, \ldots, r_\ell}$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\xrightarrow{\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j}$

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + \sum_{k=1}^{m} \gamma_k \cdot f_k(v_{j,1}, \ldots, v_{j,n})$$

Verifier

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

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

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

→ Q

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

← $r_1, \ldots, r_\ell$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

→ $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + \sum_{k=1}^{m} \gamma_k \cdot f_k(v_{j,1}, \ldots, v_{j,n})$$

Verifier

5-round variant used in most of the recent MPCitH-based signature schemes

TCitH and VOLEitH Frameworks, *in the PIOP formalism*

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

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

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

$\xrightarrow{\mathrm{PCom}(P_0, P_1, \ldots, P_n)}$

③ Choose random coefficients

$$\gamma_1, \ldots, \gamma_m \xleftarrow{\$} \mathbb{F}$$

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

$\xrightarrow{Q}$

⑤ Choose ℓ random evaluation points $r_1, \ldots, r_\ell \in \mathscr{C} \subset \mathbb{F}$

$\xleftarrow{r_1, \ldots, r_\ell}$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

$\xrightarrow{\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j}$

⑦ Check that $\{v_{j,0}, v_{j,1}, \ldots, v_{j,n}\}_j$ are consistent with the commitment.

Check that, for all j,

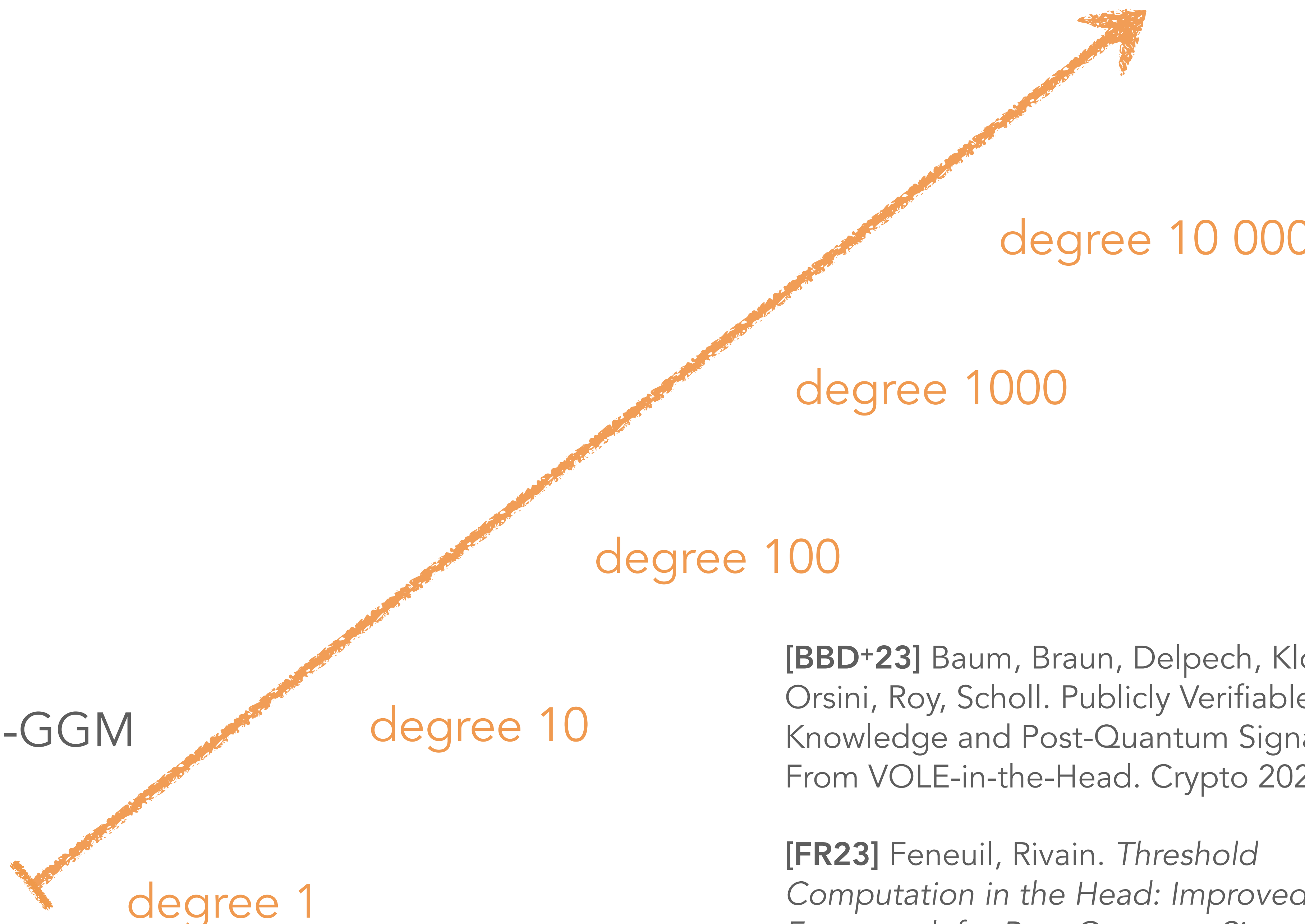
$$r_j \cdot Q(r_j) = r_j \cdot v_{j,0} + \sum_{k=1}^{m} \gamma_k \cdot f_k(v_{j,1}, \ldots, v_{j,n})$$

Prover

Verifier

How to commit to polynomials?

(using symmetric primitives)



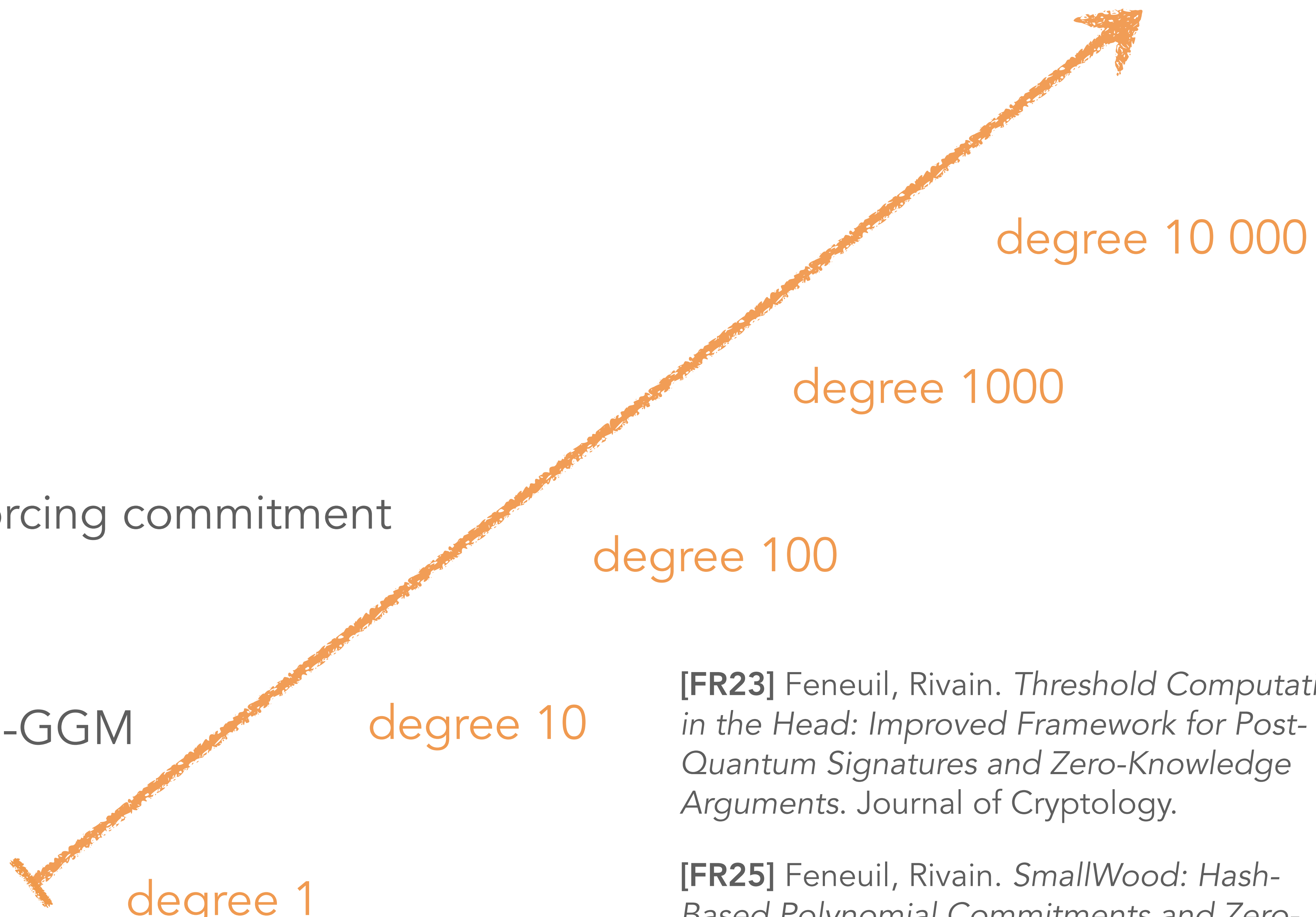
① VOLEitH / TCitH-GGM

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

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. Journal of Cryptology.

How to commit to polynomials?

(using symmetric primitives)



② Degree-enforcing commitment

① VOLEitH / TCitH-GGM

[FR23] Feneuil, Rivain. *Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments*. Journal of Cryptology.

[FR25] Feneuil, Rivain. *SmallWood: Hash-Based Polynomial Commitments and Zero-Knowledge Arguments for Relatively Small Instances*. ePrint 2025/1085.

How to commit to polynomials?

(using symmetric primitives)

③ Merkle Trees with Ligero-like Proximity Tests

degree 10 000

degree 1000

② Degree-enforcing commitment

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?

(using symmetric primitives)

④ FRI-based commitments

degree 10 000

③ Merkle Trees with
Ligero-like Proximity Tests

degree 1000

② Degree-enforcing commitment

degree 100

① VOLEitH / TCitH-GGM

degree 10

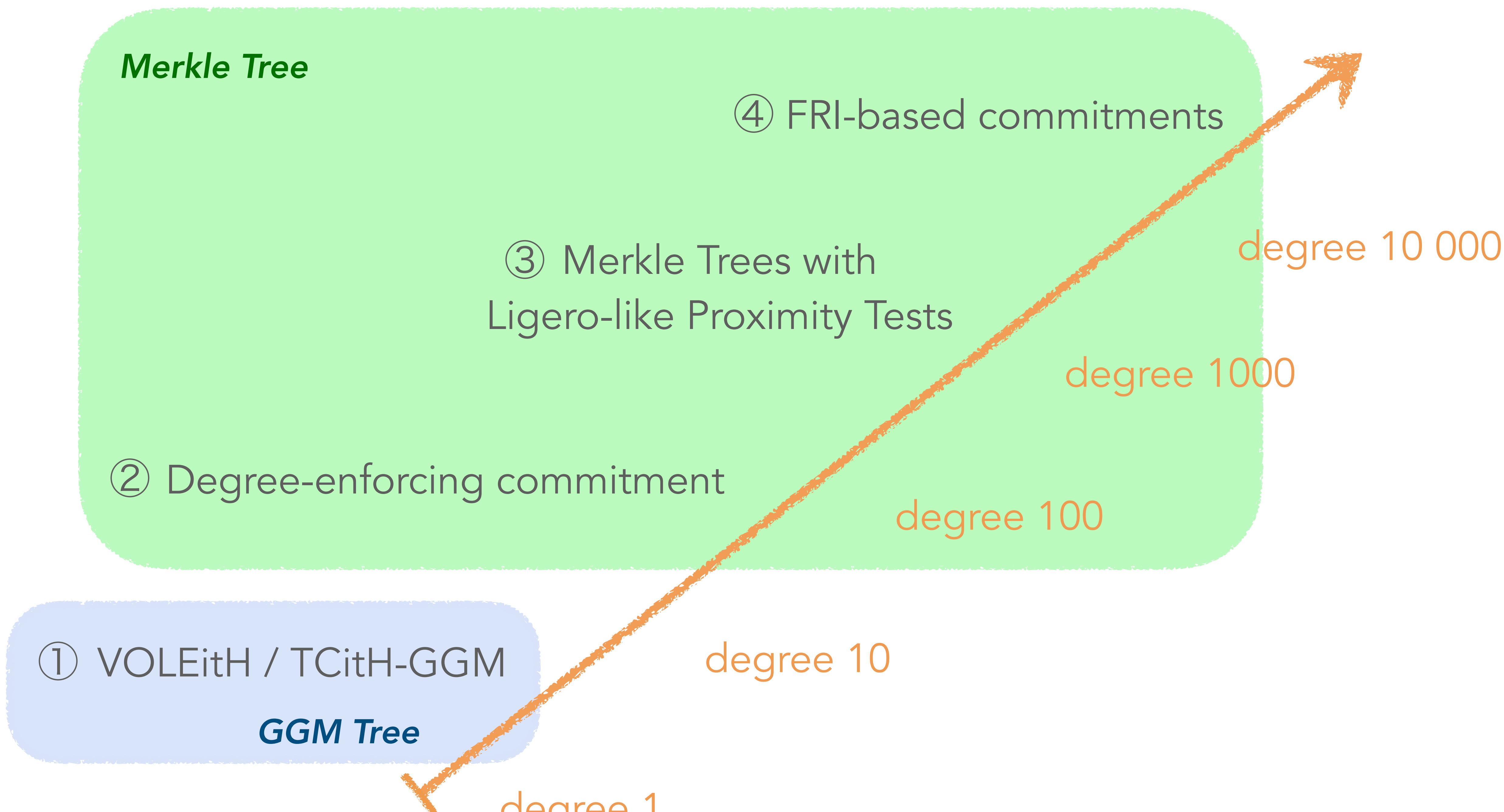
degree 1

[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.

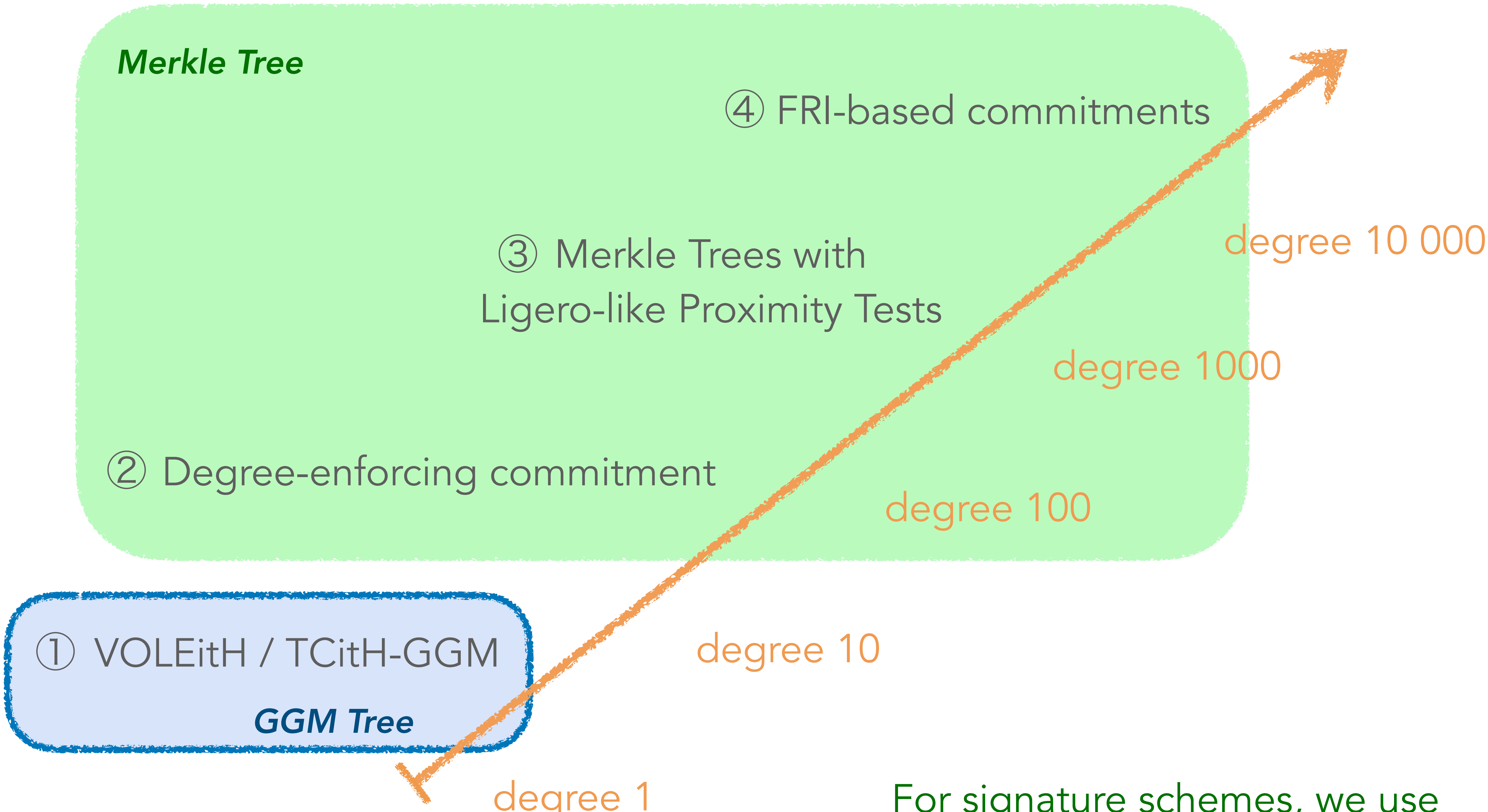
How to commit to polynomials?

(using symmetric primitives)



How to commit to polynomials?

(using symmetric primitives)



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

Committing to a Polynomial using a Seed Tree

A seed tree of N leaves to commit to degree-1 polynomials

- 👉 The prover can provably open N evaluations (i.e. $N = |\mathscr{C}|$)
- 👉 Soundness error of $\dfrac{d}{N}$

$$\frac{\binom{d \cdot \ell}{\ell}}{\binom{|\mathscr{C}|}{\ell}} \text{ with } \ell := 1$$

Committing to a Polynomial using a Seed Tree

A seed tree of N leaves to commit to degree-1 polynomials

- 👉 The prover can provably open N evaluations (i.e. $N = |\mathscr{C}|$)
- 👉 Soundness error of $\dfrac{d}{N}$

How to have a negligible soundness error? 🤔

Committing to a Polynomial using a Seed Tree

A seed tree of N leaves to commit to degree-1 polynomials

👉 The prover can provably open N evaluations (i.e. $N = |\mathscr{C}|$)

👉 Soundness error of $\frac{d}{N}$

How to have a negligible soundness error? 🤔

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

Committing to a Polynomial using a Seed Tree

A seed tree of N leaves to commit to degree-1 polynomials

👉 The prover can provably open N evaluations (i.e. $N = |\mathscr{C}|$)

👉 Soundness error of $\frac{d}{N}$

How to have a negligible soundness error? 🤔

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

A seed tree of N leaves to commit to degree-1 polynomials

👉 The prover can provably open N evaluations (i.e. $N = |\mathscr{C}|$)

👉 Soundness error of $\frac{d}{N}$

How to have a negligible soundness error? 🤔

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 $\left(\frac{d}{N}\right)^\tau$.

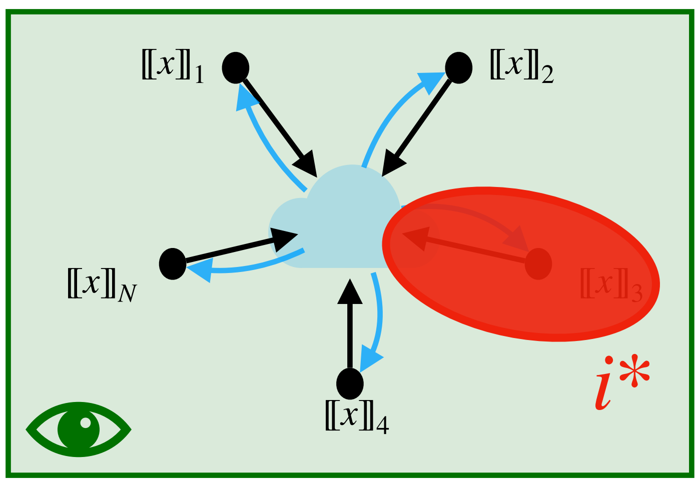
3. *VOLEitH Approach*. Embed τ polynomials over $\mathbb{F}_q$ into a unique polynomial over $\mathbb{F}_{q^\tau}$, for which we will be able to open N^τ evaluations. Soundness error of $\frac{d}{N^\tau}$.

Link between MPCitH and PIOP

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\}\backslash\{i^*\}$

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

⑥ For all (i, j), reveal the evaluation

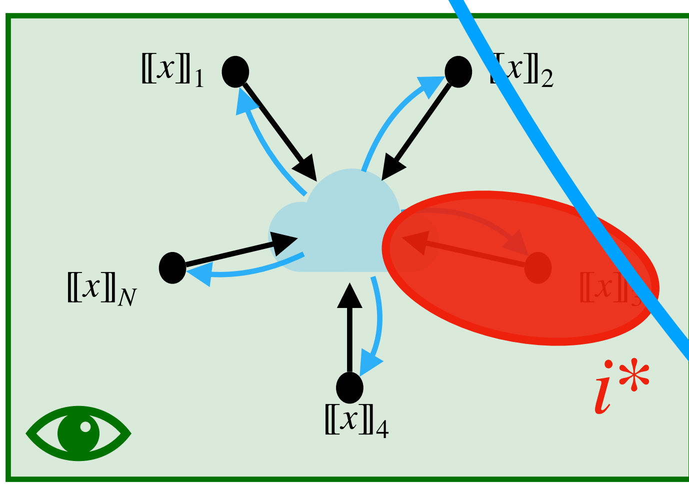
$$v_{j,i} := P_i(r_j)$$

Link between MPCitH and PIOP

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \dots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1,\dots,N\}\backslash\{i^*\}$

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

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

② Commit to the polynomials $P_0, P_1, \dots, P_n$

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \dots, P_n(X))$$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

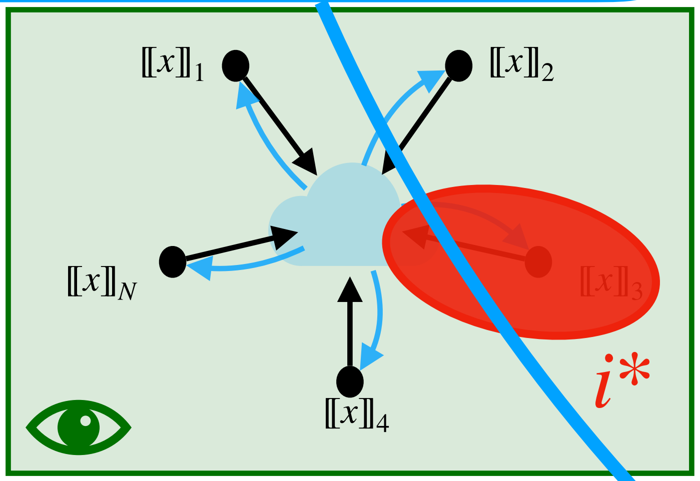
Commit to a (ℓ, N)-Shamir secret sharing

Link between MPCitH and PIOP

① Generate and commit shares

$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\} \backslash \{i^*\}$

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

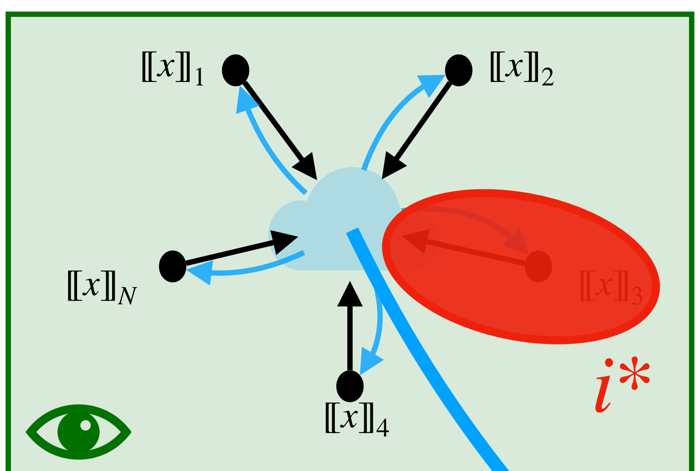
Computation of the MPC protocol, assuming that a multiplication is computed share by share

Link between MPCitH and PIOP

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1, \ldots, N\} \setminus \{i^*\}$

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

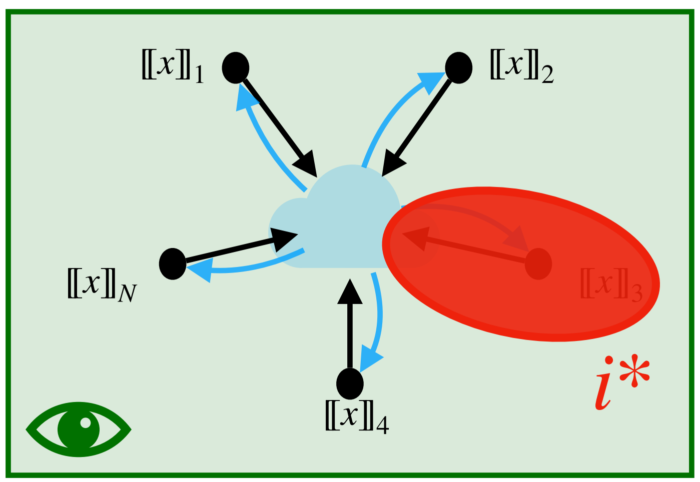
Revealing the polynomial is equivalent to revealing the broadcast Shamir secret sharing.

Link between MPCitH and PIOP

① Generate and commit shares

$$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$$

② Run MPC in their head



④ Open parties $\{1,\ldots,N\}\backslash\{i^*\}$

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

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

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

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

$$X \cdot Q(X) = X \cdot P_0(X) + \sum_{k=1}^{m} \gamma_k \cdot f_k(P_1(X), \ldots, P_n(X))$$

⑥ For all (i, j), reveal the evaluation

$$v_{j,i} := P_i(r_j)$$

It is equivalent to revealing some party computations.

Instantiations

For the sake of diversity…

AES Key Recovery
AIM Key Recovery
LowMC Key Recovery
Rain Key Recovery

Anemoi Hash Preimage
Poseidon Hash Preimage
Griffin Hash Preimage
RescuePrime Hash Preimage

Legendre PRF
BHHG's PRF

LWE
SIS
Subset Sum

Discrete Logarithm
Integer Factorization
Double Discrete Logarithm

Syndrome Decoding
Regular Syndrome Decoding
Permuted Kernel
Linear Code Equivalence
Restricted Syndrome Decoding

MinRank
Rank Syndrome Decoding
Subfield Collision Problem
Matrix Subcode Equivalence

Multivariate Quadratic
PowAff2

For the sake of diversity…

AES Key Recovery
AIM Key Recovery
LowMC Key Recovery
Rain Key Recovery

Anemoi Hash Preimage
Poseidon Hash Preimage
Griffin Hash Preimage
RescuePrime Hash Preimage

Legendre PRF
BHHG's PRF

LWE
SIS
Subset Sum

Discrete Logarithm
Integer Factorization
Double Discrete Logarithm

Syndrome Decoding
Regular Syndrome Decoding
Permuted Kernel
Linear Code Equivalence
Restricted Syndrome Decoding

MinRank
Rank Syndrome Decoding
Subfield Collision Problem
Matrix Subcode Equivalence

Multivariate Quadratic
PowAff2

To use the PIOP-based MPCitH frameworks, one just needs to write those problems using polynomial constraints.

For the sake of diversity…

AES Key Recovery
AIM Key Recovery
LowMC Key Recovery
Rain Key Recovery

Anemoi Hash Preimage
Poseidon Hash Preimage
Griffin Hash Preimage
RescuePrime Hash Preimage

Legendre PRF
BHHG's PRF

LWE
SIS
Subset Sum

Discrete Logarithm
Integer Factorization
Double Discrete Logarithm

Syndrome Decoding
Regular Syndrome Decoding
Permuted Kernel
Linear Code Equivalence
Restricted Syndrome Decoding

MinRank
Rank Syndrome Decoding
Subfield Collision Problem
Matrix Subcode Equivalence

Multivariate Quadratic
PowAff2

We can build highly conservative schemes!

For the sake of diversity…

LWE
SIS
Subset Sum

Syndrome Decoding
Regular Syndrome Decoding
Permuted Kernel
Linear Code Equivalence
Restricted Syndrome Decoding

AES Key Recovery
AIM Key Recovery
LowMC Key Recovery
Rain Key Recovery

MinRank
Rank Syndrome Decoding
Subfield Collision Problem
Matrix Subcode Equivalence

Anemoi Hash Preimage
Poseidon Hash Preimage
Griffin Hash Preimage
RescuePrime Hash Preimage

Multivariate Quadratic
PowAff2

Legendre PRF
BHHG's PRF

Discrete Logarithm
Integer Factorization
Double Discrete Logarithm

We can build highly conservative schemes!

NIST Candidates

	NIST Submission		
Security Assumptions	***Candidate Name***	***Sig. Size***	***PK Size***
AES Block cipher	FAEST v2	3.9-4.5 KB	32 B
MinRank	Mirath	3.0-3.2 KB	57-73 B
Multivariate Quadratic	MQOM v2	2.8-3.2 KB	52-80 B
Permuted Kernel	PERK v2.1	3.5 KB	100 B
Rank Syndrome Decoding	RYDE v2	3.1 KB	69 B
Syndrome Decoding	SDitH v2	3.7 KB	70 B

Using seed trees of around 2048 leaves

NIST Candidates

	NIST Submission		
Security Assumptions	***Candidate Name***	***Sig. Size***	***PK Size***
AES Block cipher	FAEST v2	3.9-4.5 KB	32 B
MinRank	Mirath	3.0-3.2 KB	57-73 B
Multivariate Quadratic	MQOM v2	2.8-3.2 KB	52-80 B
Permuted Kernel	PERK v2.1	3.5 KB	100 B
Rank Syndrome Decoding	RYDE v2	3.1 KB	69 B
Syndrome Decoding	SDitH v2	3.7 KB	70 B

Using seed trees of around 2048 leaves

What about the computational cost?

Implementation

*When using **seed trees** to commit to polynomials, we have*

Signing time ≈ Verification time

Implementation

*When using **seed trees** to commit to polynomials, we have*

Signing time ≈ Verification time

Over laptop-grade CPU,

Signing time ≈ Verification time ≈ few milliseconds (1-10 Mc)

*Using intensively **AES instructions**,*
***vectorialization** instructions, and **large** memory footprint*

Implementation

*When using **seed trees** to commit to polynomials, we have*

Signing time ≈ Verification time

Over laptop-grade CPU,

Signing time ≈ Verification time ≈ few milliseconds (1-10 Mc)

*Using intensively **AES instructions**,*
***vectorialization** instructions, and **large** memory footprint*

Over embedded microcontrollers,

[BBBPP24] Bettaieb, Bidoux, Budroni, Palumbi, Perin. *Enabling PERK and other MPC-in-the-Head Signatures on Resource-Constrained Devices*. TCHES 2024.

[ADENS25] Aranha, Degn, Eilath, Nielsen, Scholl. *FAEST for Memory-Constrained Devices with Side-Channel Protections*. ePrint 2025/1261.

[BF26] Benadjila, Feneuil. *Breaking the Myth of MPCitH Inefficiency: Optimizing MQOM for Embedded Platforms*. ePrint 2026/078.

Implementation (over embedded devices)

Article	*NIST Candidate*	*Memory Footprint*	*Signing time*	*Sig. Sizes*
[BBBPP24]	PERK v1	28 KB	1136 Mc	~ 6 KB
[ADENS25]	FAEST (EM) v1	31 KB	158 Mc	~ 5.6 KB
[BF26]	MQOM v2	10 KB	76 Mc	~ 3.3 KB
		5 KB	183 Mc	

Using seed trees of around 256 leaves

Article	*NIST Candidate*	*Memory Footprint*	*Signing time*	*Sig. Sizes*
[BBBPP24]	PERK v1	-	-	-
[ADENS25]	FAEST (EM) v1	31 KB	1288 Mc	~ 4.6 KB
[BF26]	MQOM v2	14 KB	308 Mc	~ 2.9 KB
		5.5 KB	792 Mc	

Using seed trees of around 2048 leaves

Physical Security

👉 *Side-Channel Leakage*

[GAGLM24] Godard, Aragon, Gaborit, Loiseau, Maillard. *Single Trace Side-Channel Attack on the MPC-in-the-Head Framework*. PQCrypto 2025.

[JD25a] Jendral, Dubrova. *Side-Channel on VOLEitH Signature Schemes Breaking Masked FAEST*. CiC 2025.

👉 *Fault attacks*

[JD25b] Jendral, Dubrova. *Fault Attacks on VOLEitH Signature Schemes.* TCHES 2026.

[SD26] Sarde, Debande. *Differential Fault Attacks on MQOM, Breaking the Heart of Multivariate Evaluation.* CASCADE 2026.

[BBK25] Banda, Brinkmann, Krämer. *Fault Attacks on MPCitH Signature Schemes.* ePrint 2025/1745.

Comparison with the other families

	MPCitH	Dilithium ML-DSA	Falcon FN-DSA	SPHINCS+	UOV	Mayo	SQIsign	LESS	CROSS
		Lattice-based schemes			*UOV-like schemes*			*Alternative code-based schemes*	
Type	FS	FS	H&S	Hash-based	H&S	H&S	FS	FS	FS
|Sig|	2.5-4.5	2.4	0.7	7.8-17	0.1	0.2-0.5	0.1	1.3-2.3	9.0-18
|PK|	< 0.2	1.3	0.9	< 0.1	44-67	1.4-4.9	0.1	14-97	0.1
|Sig|+|PK|	2.5-4.6	3.7	1.6	7.9-17	44-67	1.9-5.1	0.2	17-98	9.0-18
Sign. Time	~	++	++	--	~	~	-	-	+
Verif. Time	~	++	++	~	++	++	~	-	+
Security	AES Unstructured SD Unstructured MQ …	Structured Lattice	Structured Lattice	Hash	UOV Trapdoor	New UOV-like Trapdoor	Isogeny	Code Equivalence	Restricted Syndrome Decoding

Sizes in kilobytes (KB)

FS: Fiat-Shamir transformation

H&S: Hash-and-sign scheme

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any PQ hardness assumption
  - A practical tool to build conservative signature schemes

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any PQ hardness assumption
  - A practical tool to build conservative signature schemes

- PIOP-based MPCitH Frameworks
  - Lead to signatures of 2.5-5 kilobytes
  - Speed up the MPCitH schemes
  - Used in the lastest version of all NIST candidates

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any PQ hardness assumption
  - A practical tool to build conservative signature schemes
- PIOP-based MPCitH Frameworks
  - Lead to signatures of 2.5-5 kilobytes
  - Speed up the MPCitH schemes
  - Used in the lastest version of all NIST candidates
- Next Steps
  - More optimized implementations
  - Side-channel Analysis
  - SCA & Fault countermeasures → Protected implementation

Conclusion

- MPC-in-the-Head
  - Very versatile and tunable
  - Can be applied to any PQ hardness assumption
  - A practical tool to build conservative signature schemes

- PIOP-based MPCitH Frameworks
  - Lead to signatures of 2.5-5 kilobytes
  - Speed up the MPCitH schemes
  - Used in the lastest version of all NIST candidates

- Next Steps
  - More optimized implementations
  - Side-channel Analysis
  - SCA & Fault countermeasures → Protected implementation

Thank you for your attention.