MQOM & SD in the Head Signature Schemes

Matthieu Rivain

Oxford PQC Summit 2023

Sep 4, 2023

CryptoExperts

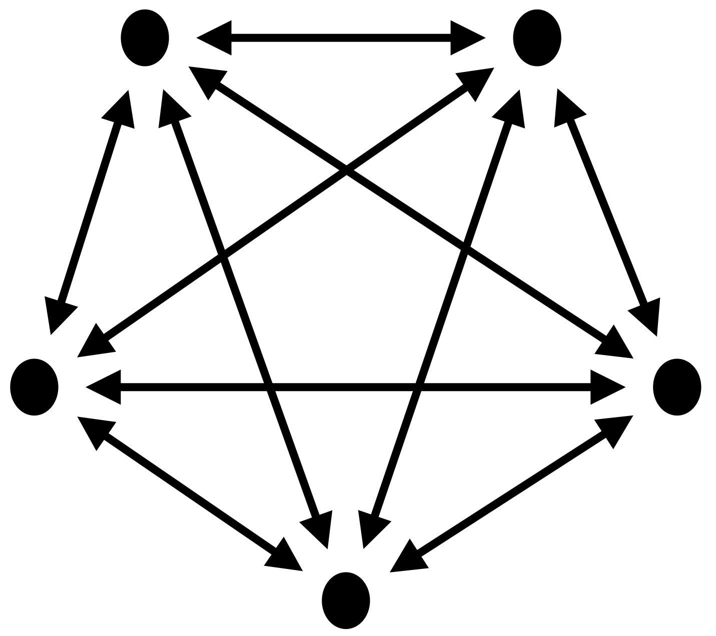
WE INNOVATE TO SECURE YOUR BUSINESS

One-way function

$$F : x \mapsto y$$

E.g. AES, MQ system,
Syndrome decoding

Multiparty computation (MPC)

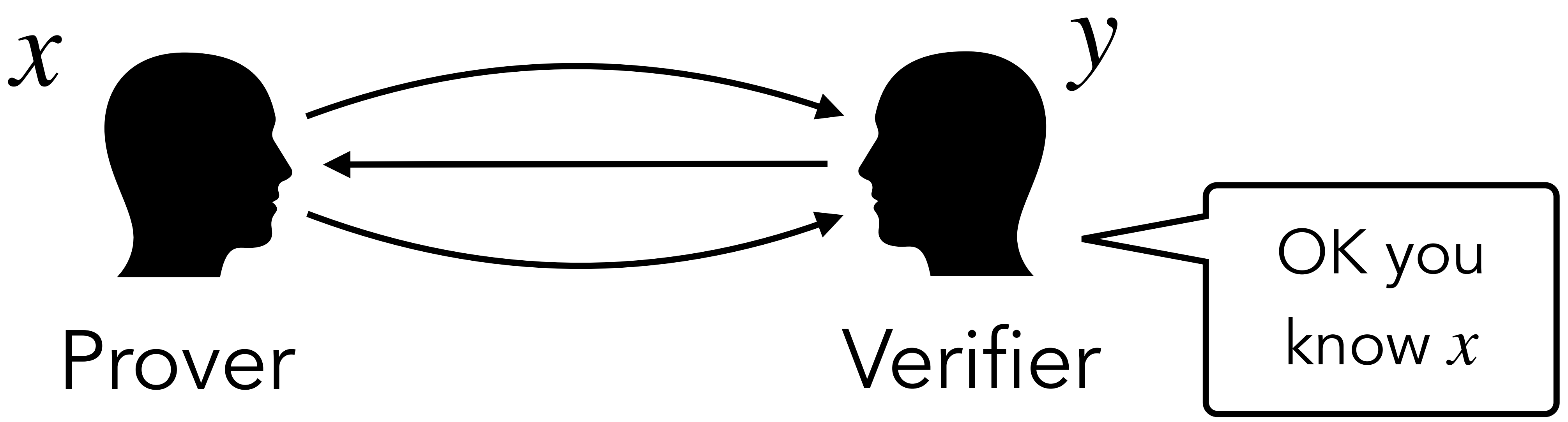


Input sharing $[\![x]\!]$

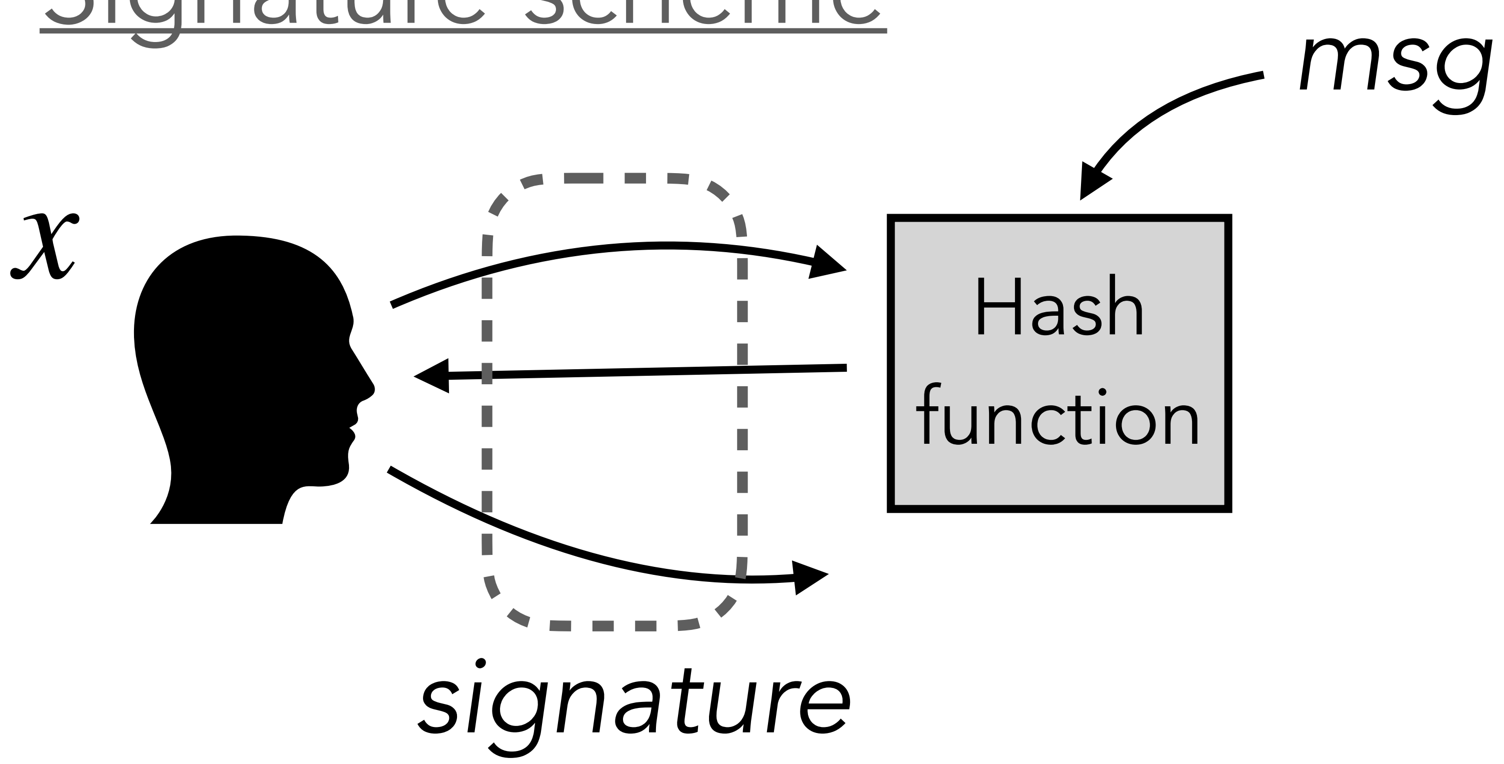
Joint evaluation of:

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

Zero-knowledge proof



Signature scheme

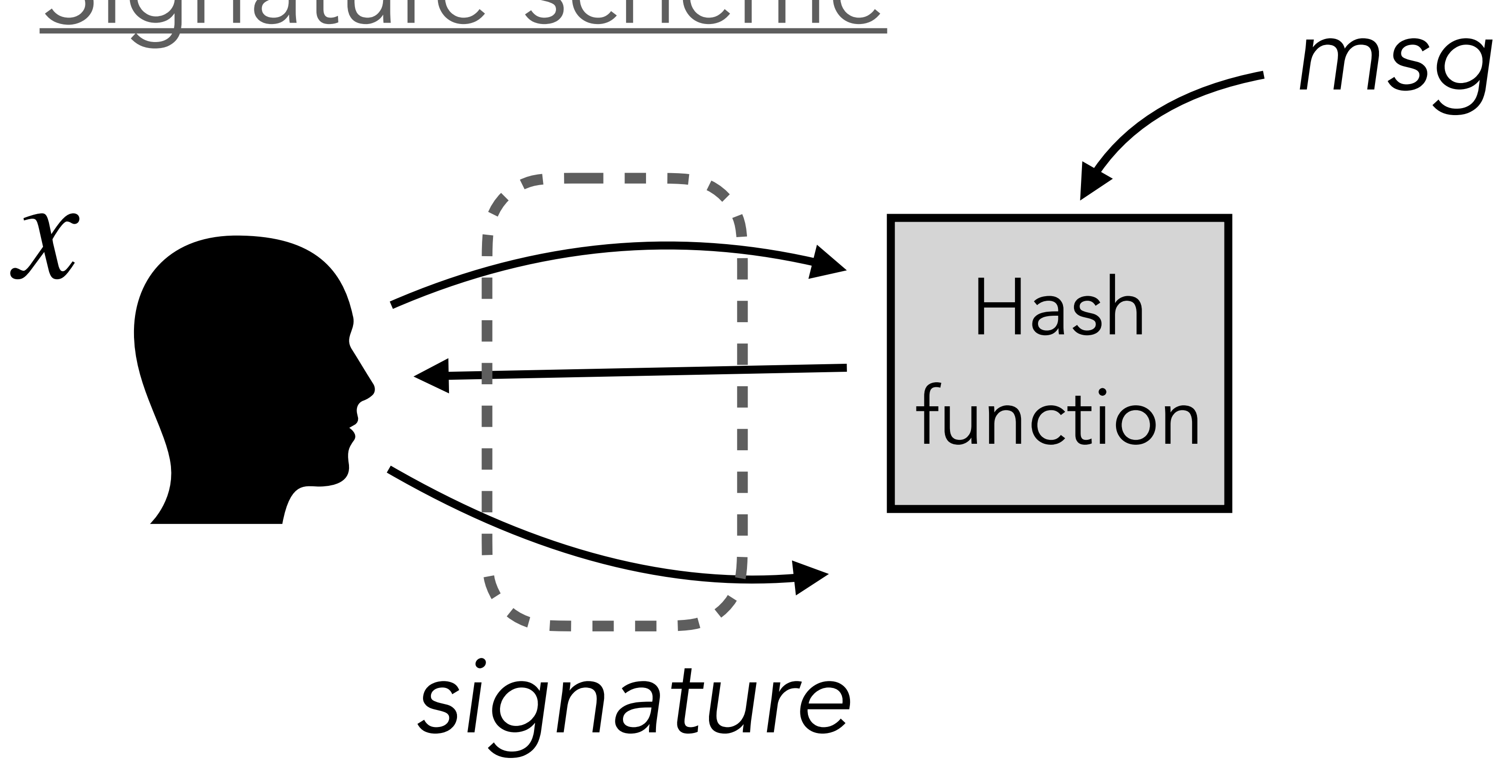


One-way function

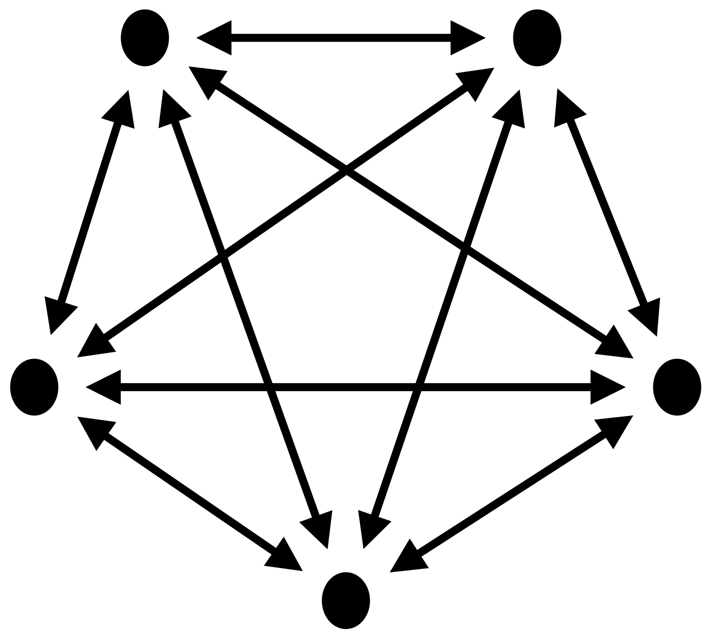
$$F : x \mapsto y$$

E.g. AES, MQ system,
Syndrome decoding

Signature scheme



Multiparty computation (MPC)



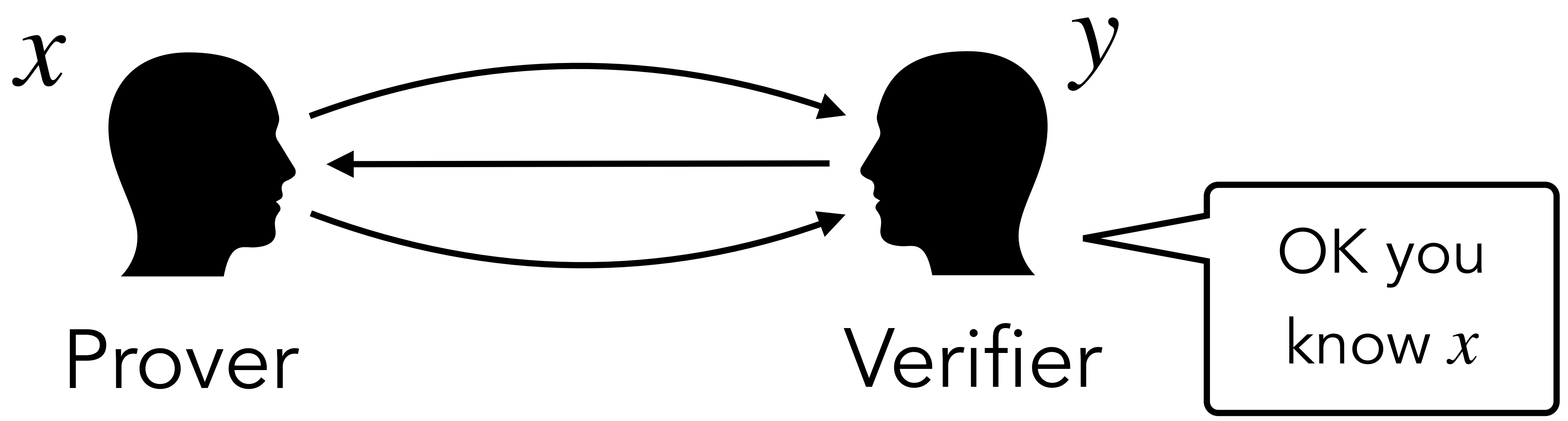
Input sharing $[\![x]\!]$

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

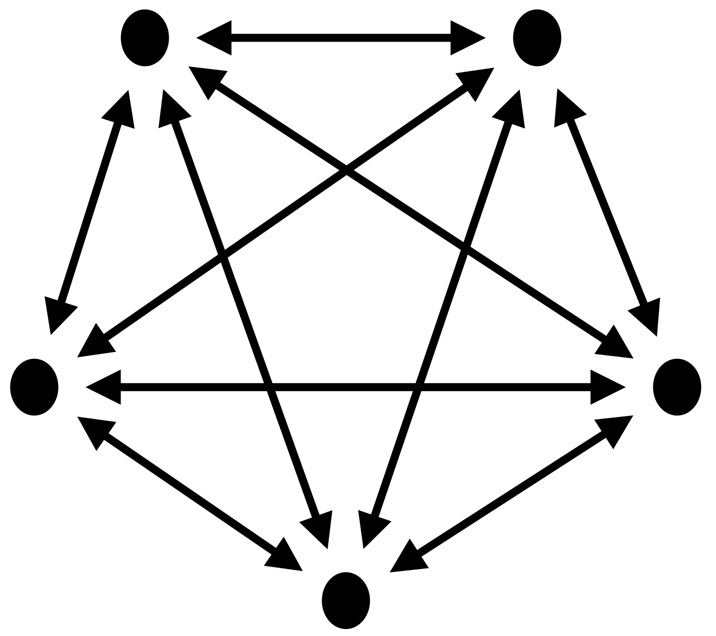


One-way function

$$F : x \mapsto y$$

E.g. AES, MQ system,
Syndrome decoding

Multiparty computation (MPC)



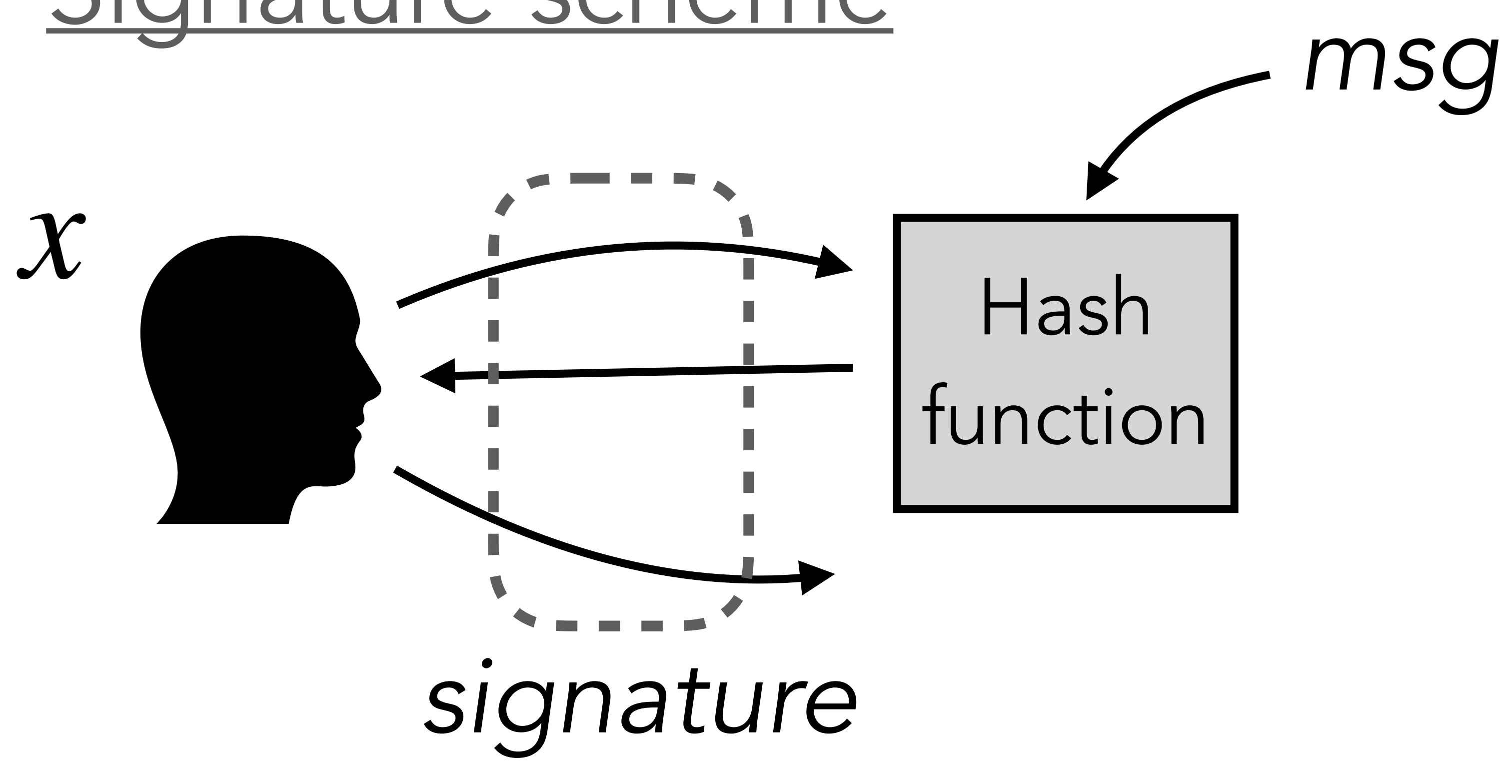
Input sharing $[\![x]\!]$

Joint evaluation of:

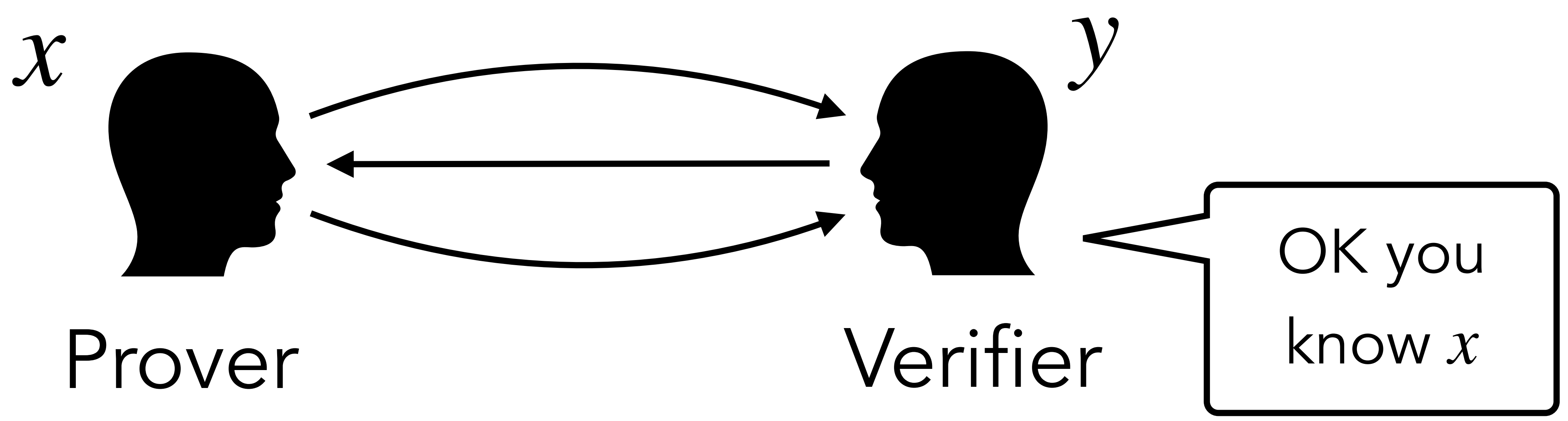
$$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$$

Fiat-Shamir transform

Signature scheme



Zero-knowledge proof

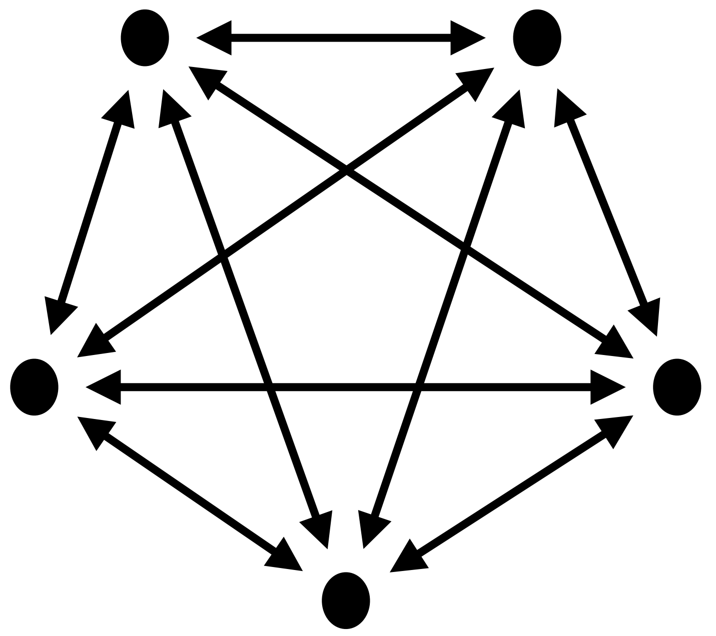


One-way function

$$F : x \mapsto y$$

E.g. AES, MQ system,
Syndrome decoding

Multiparty computation (MPC)

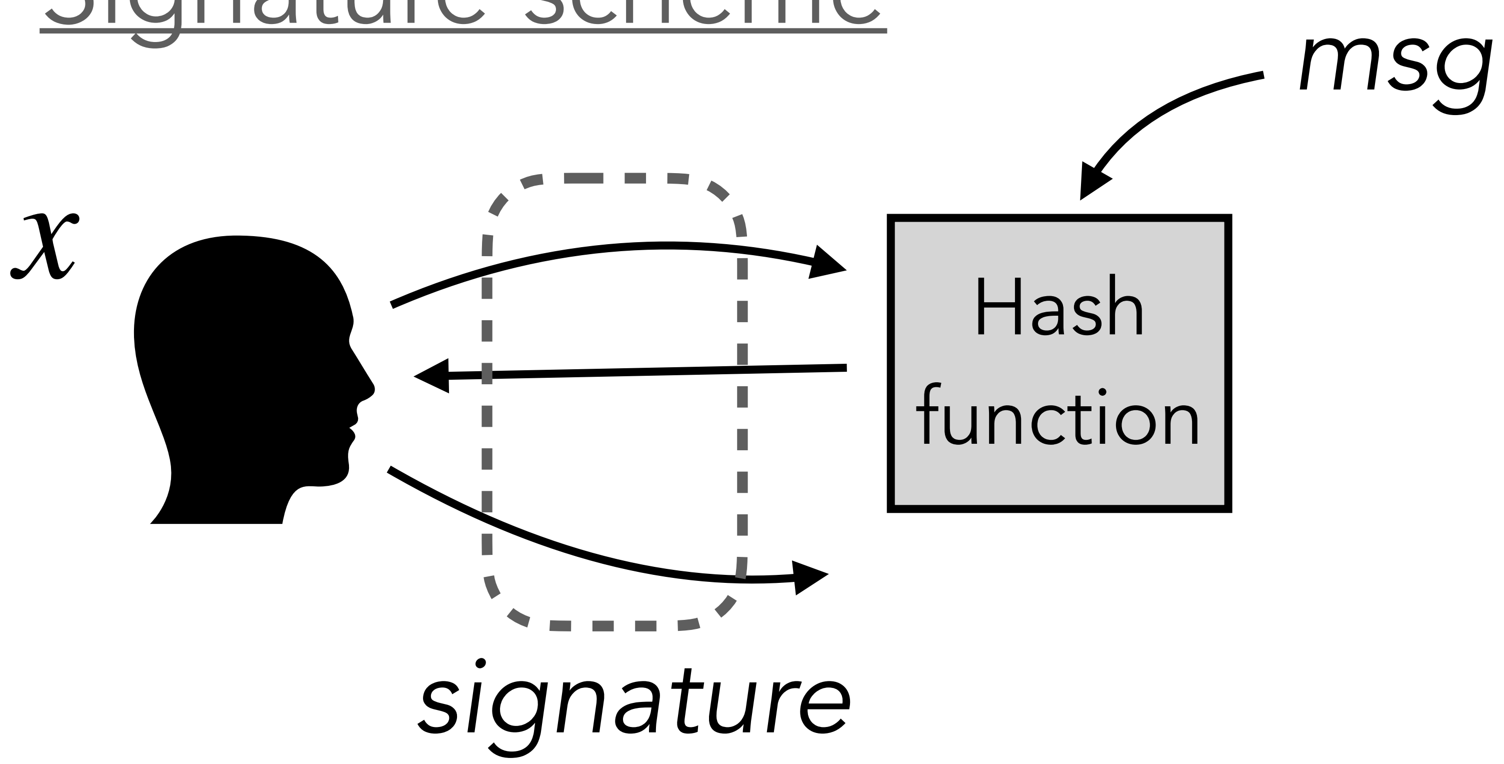


Input sharing $[\![x]\!]$

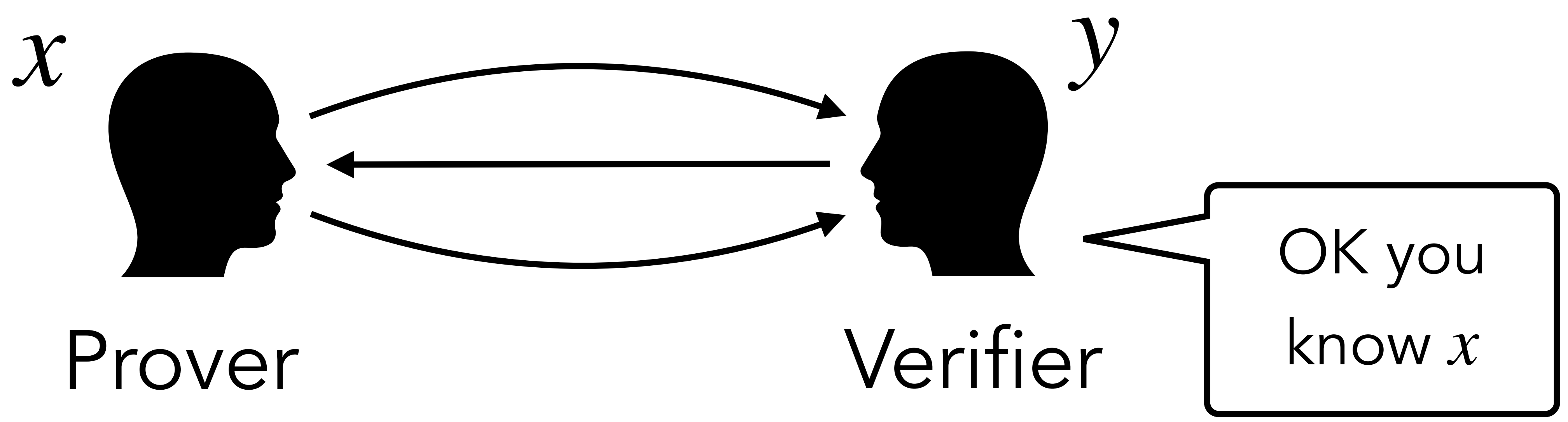
Joint evaluation of:

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

Signature scheme



Zero-knowledge proof

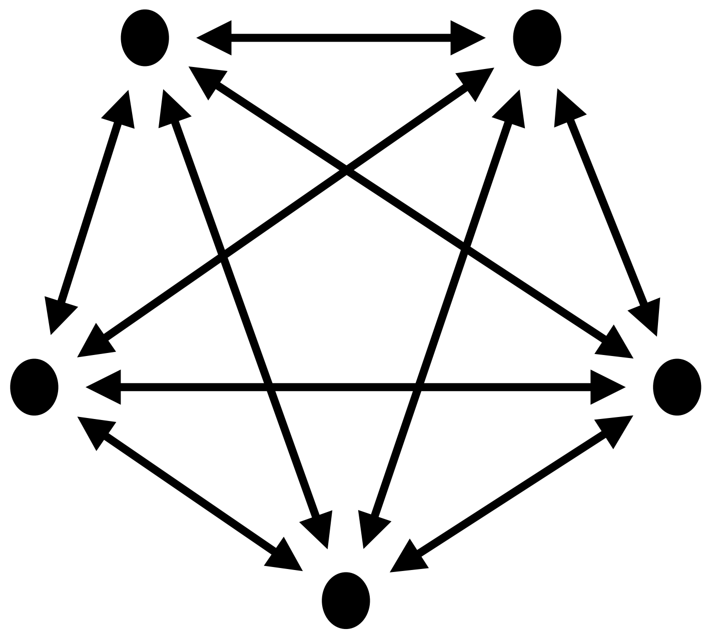


One-way function

$$F : x \mapsto y$$

E.g. AES, **MQ system,**
Syndrome decoding

Multiparty computation (MPC)

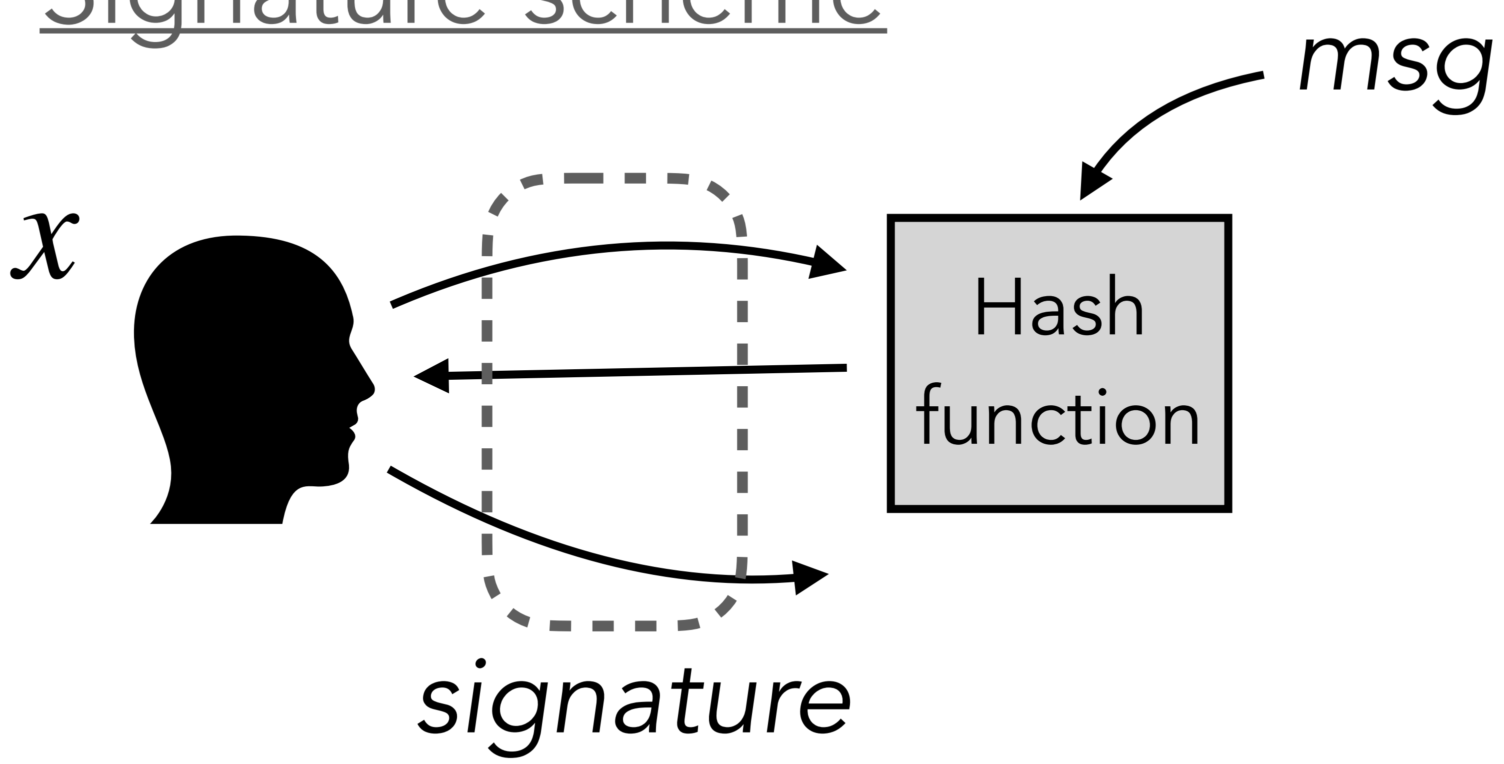


Input sharing $[\![x]\!]$

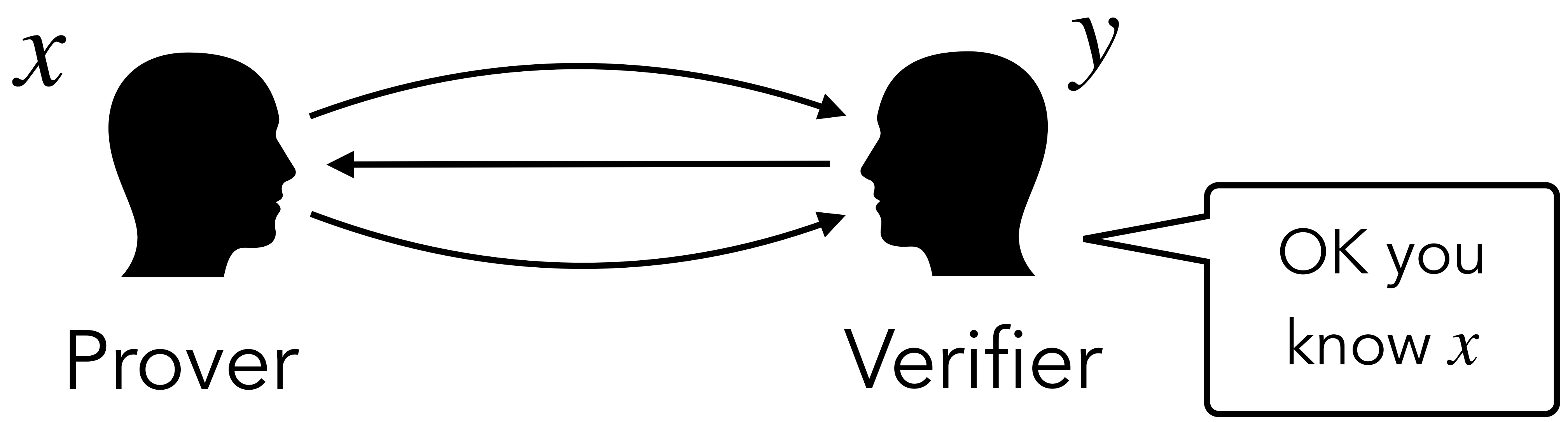
Joint evaluation of:

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

Signature scheme



Zero-knowledge proof



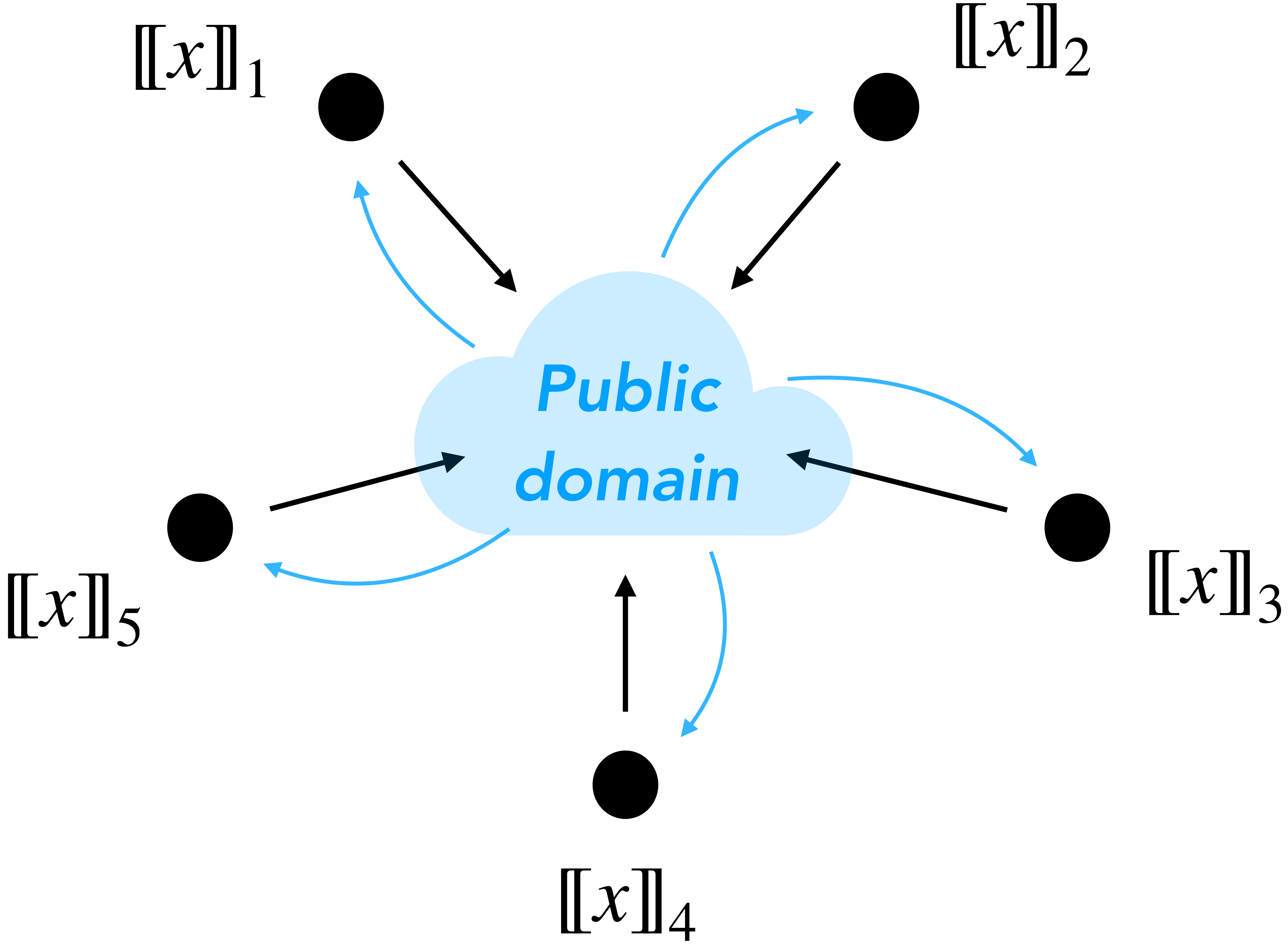
Roadmap

- Technical background
- MQOM MPC protocol
- SDitH MPC protocol
- Threshold MPCitH
- MQOM signature scheme
- SDitH signature scheme

Roadmap

- **Technical background**
- MQOM MPC protocol
- SDitH MPC protocol
- Threshold MPCitH
- MQOM signature scheme
- SDitH signature scheme

MPC model



- **Jointly compute**

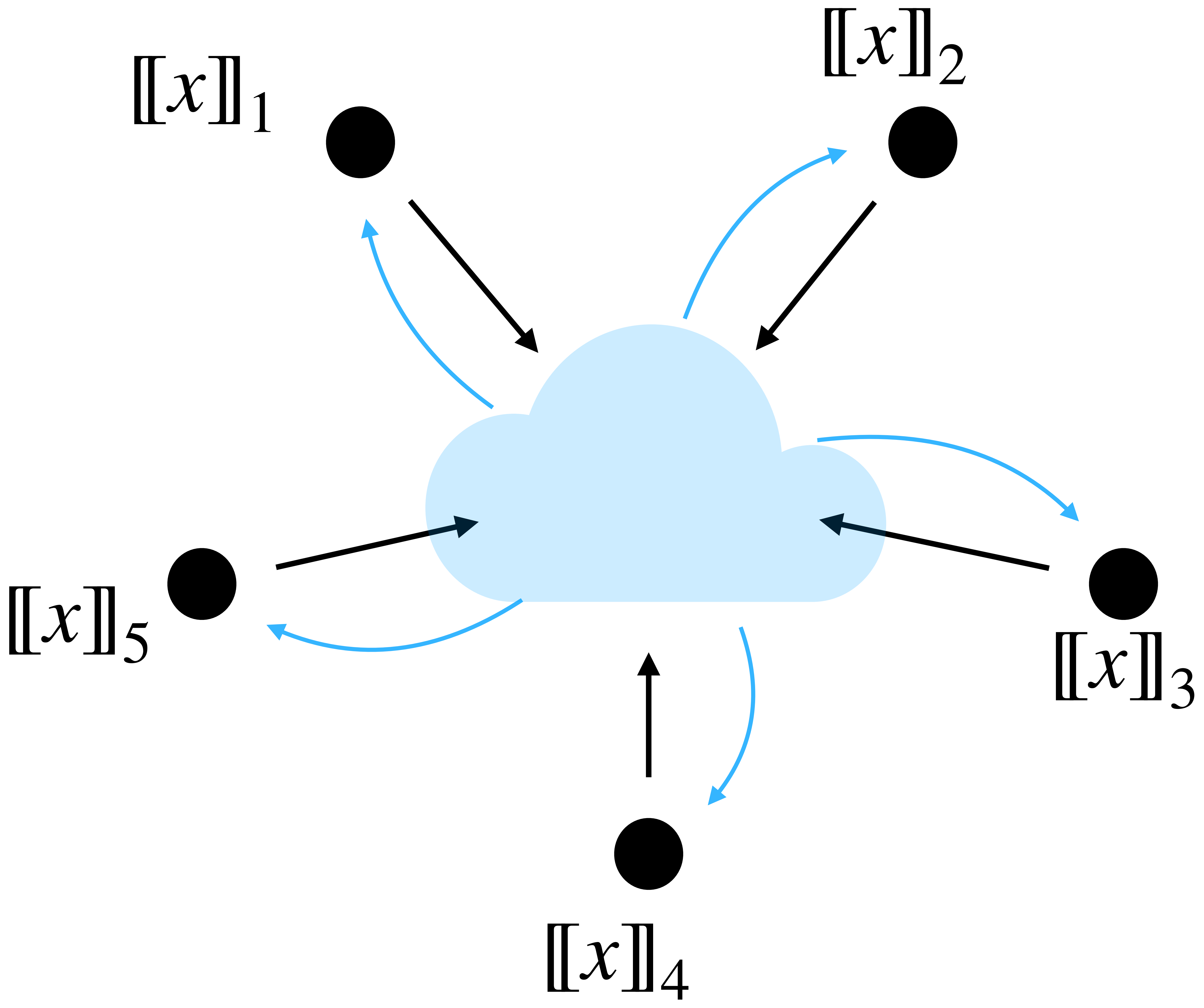
$$g(x) = \begin{cases} \text{Accept} & \text{if } F(x) = y \\ \text{Reject} & \text{if } F(x) \neq y \end{cases}$$

- **Broadcast model**
  - Parties locally compute on their shares $[\![x]\!] \mapsto [\![\alpha]\!]$
  - Parties broadcast $[\![\alpha]\!]$ and recompute α
  - Parties start again (now knowing α)

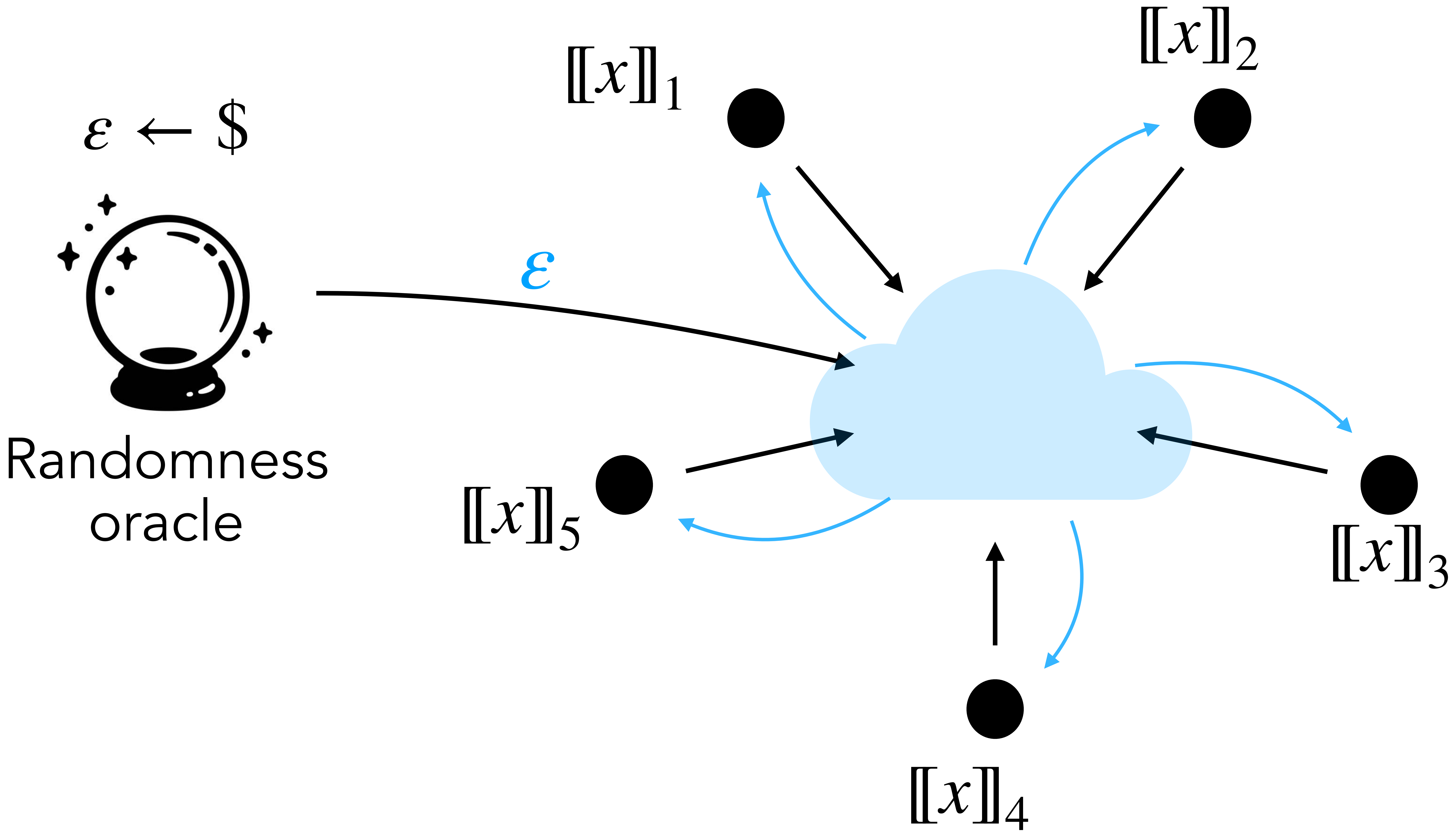
- **False positive probability**

$$\Pr\left[\, g(x) = \text{Accept} \mid F(x) \neq y \,\right]$$

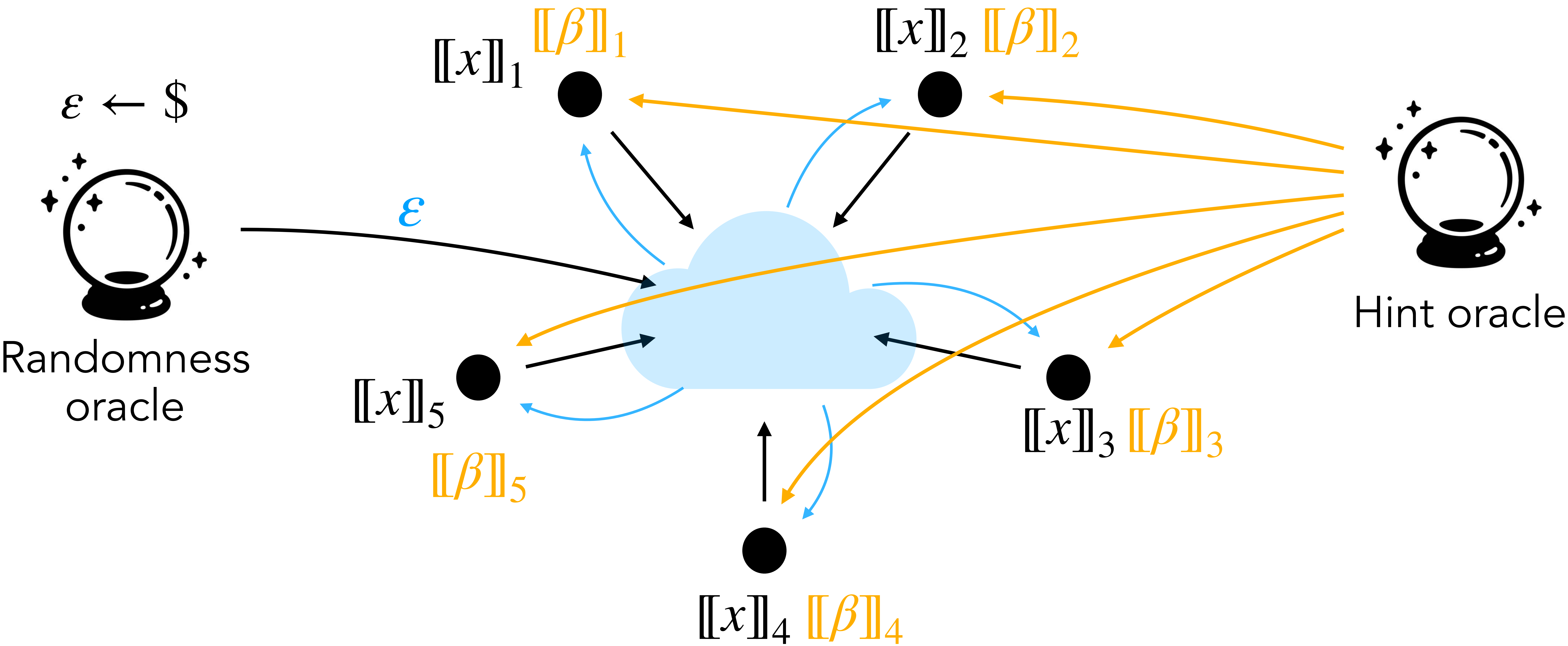
MPC model



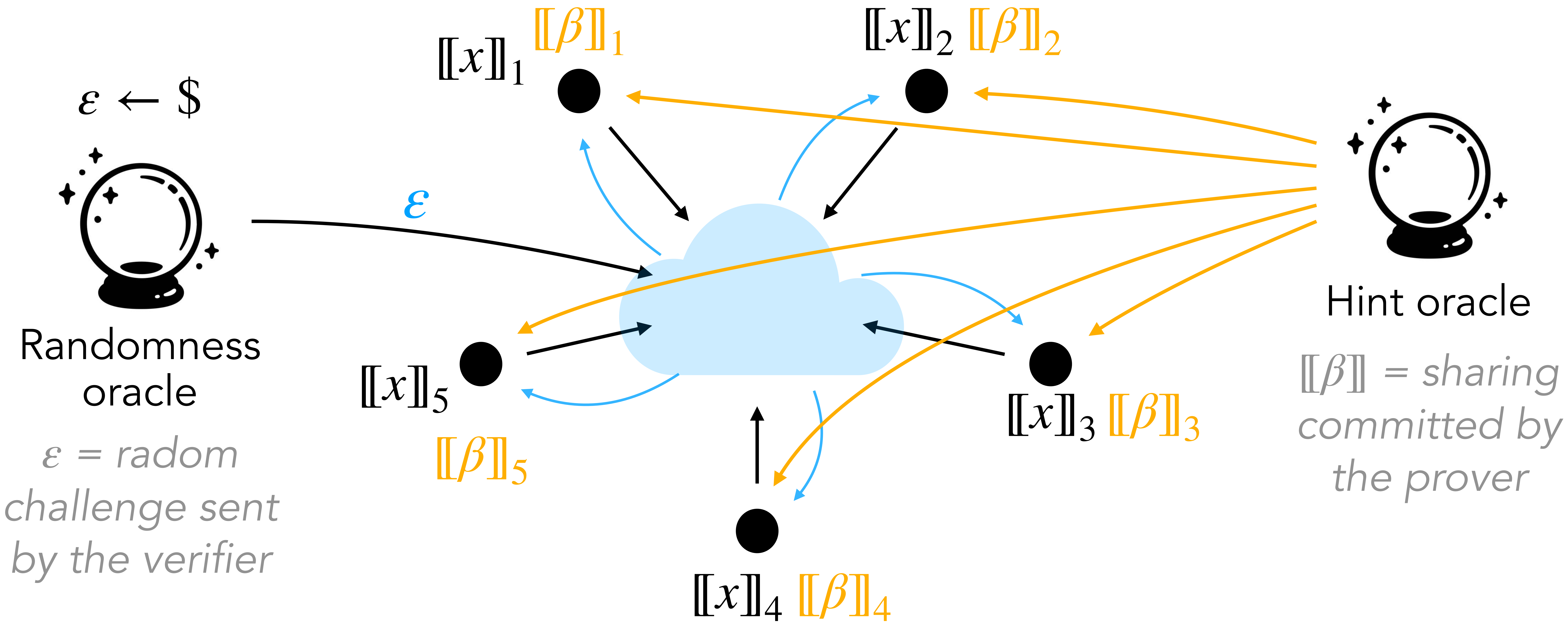
MPC model


$\varepsilon \leftarrow \$$
Randomness oracle
ε
$[\![x]\!]_1$
$[\![x]\!]_2$
$[\![x]\!]_3$
$[\![x]\!]_4$
$[\![x]\!]_5$

MPC model


$\varepsilon \leftarrow \$$
Randomness oracle
ε
$[\![x]\!]_1$ $[\![\beta]\!]_1$
$[\![x]\!]_2$ $[\![\beta]\!]_2$
Hint oracle
$[\![x]\!]_5$
$[\![\beta]\!]_5$
$[\![x]\!]_3$ $[\![\beta]\!]_3$
$[\![x]\!]_4$ $[\![\beta]\!]_4$

MPC model



$\varepsilon \leftarrow \$$

Randomness oracle

ε = radom challenge sent by the verifier

ε

$[\![x]\!]_1$ $[\![\beta]\!]_1$

$[\![x]\!]_2$ $[\![\beta]\!]_2$

$[\![x]\!]_3$ $[\![\beta]\!]_3$

$[\![x]\!]_4$ $[\![\beta]\!]_4$

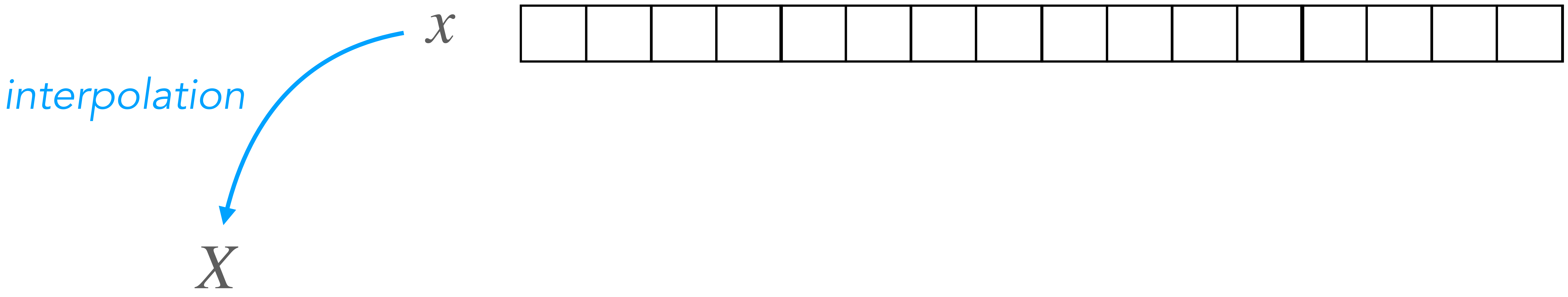
$[\![x]\!]_5$ $[\![\beta]\!]_5$

Hint oracle

$[\![\beta]\!]$ = sharing committed by the prover

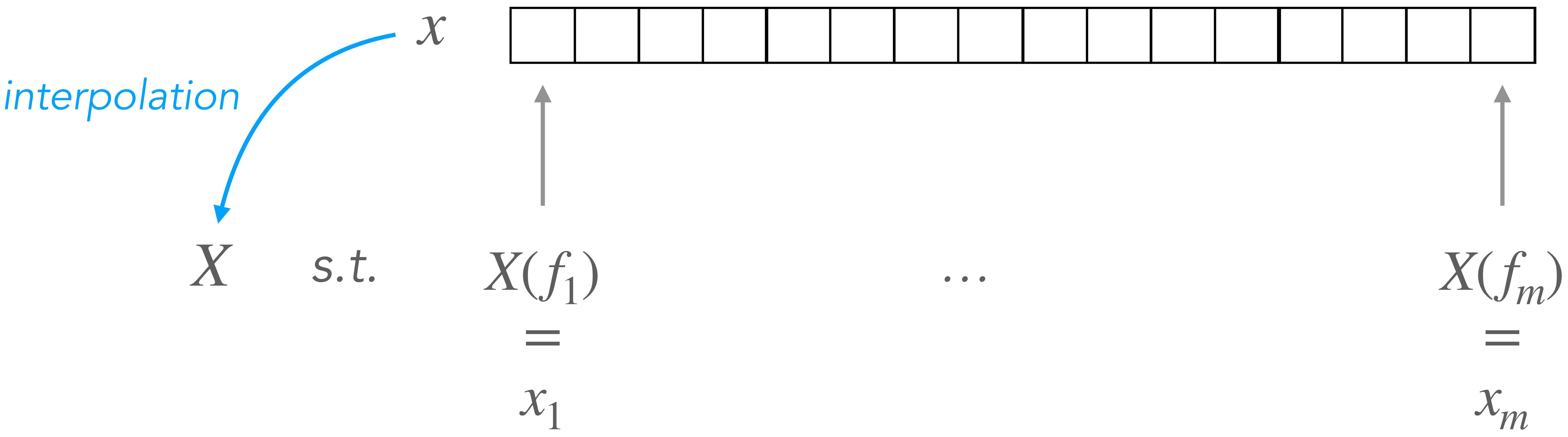
Polynomial interpolation

- Let $f_1, \ldots, f_m$ fixed points of $\mathbb{F}$
- Vector $x \in \mathbb{F}^m$ → polynomial $X \in \mathbb{F}[u]$



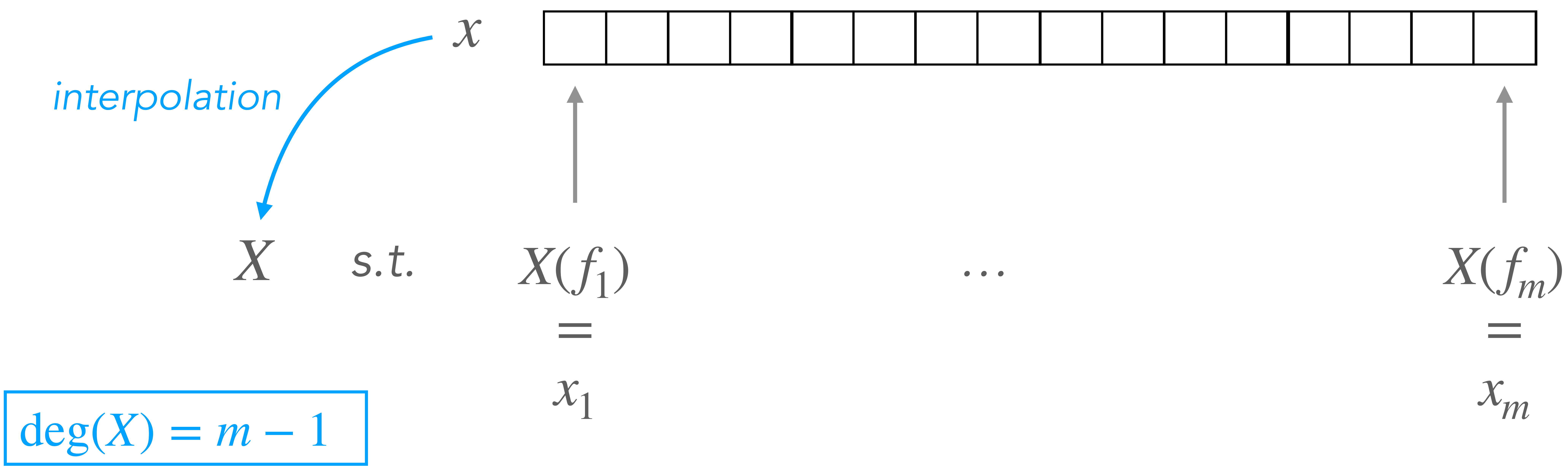
Polynomial interpolation

- Let $f_1, \ldots, f_m$ fixed points of $\mathbb{F}$
- Vector $x \in \mathbb{F}^m$ → polynomial $X \in \mathbb{F}[u]$



Polynomial interpolation

- Let $f_1, \ldots, f_m$ fixed points of $\mathbb{F}$
- Vector $x \in \mathbb{F}^m$ → polynomial $X \in \mathbb{F}[u]$



$$\boxed{\deg(X) = m - 1}$$

Schwartz–Zippel lemma

- Let P and Q two degree-d polynomials of $\mathbb{F}[u]$
- Let r a random point of $\mathbb{F}$

$$\Pr\left[P(r) = Q(r) \mid P \neq Q\right] \leq \frac{d}{|\mathbb{F}|}$$

Schwartz–Zippel lemma

- Let P and Q two degree-d polynomials of $\mathbb{F}[u]$
- Let r a random point of $\mathbb{F}$

$$\Pr\left[P(r) = Q(r) \mid P \neq Q\right] \leq \frac{d}{|\mathbb{F}|}$$

- $P(r) = Q(r) \quad \Leftrightarrow \quad r \in$ roots of $P - Q$

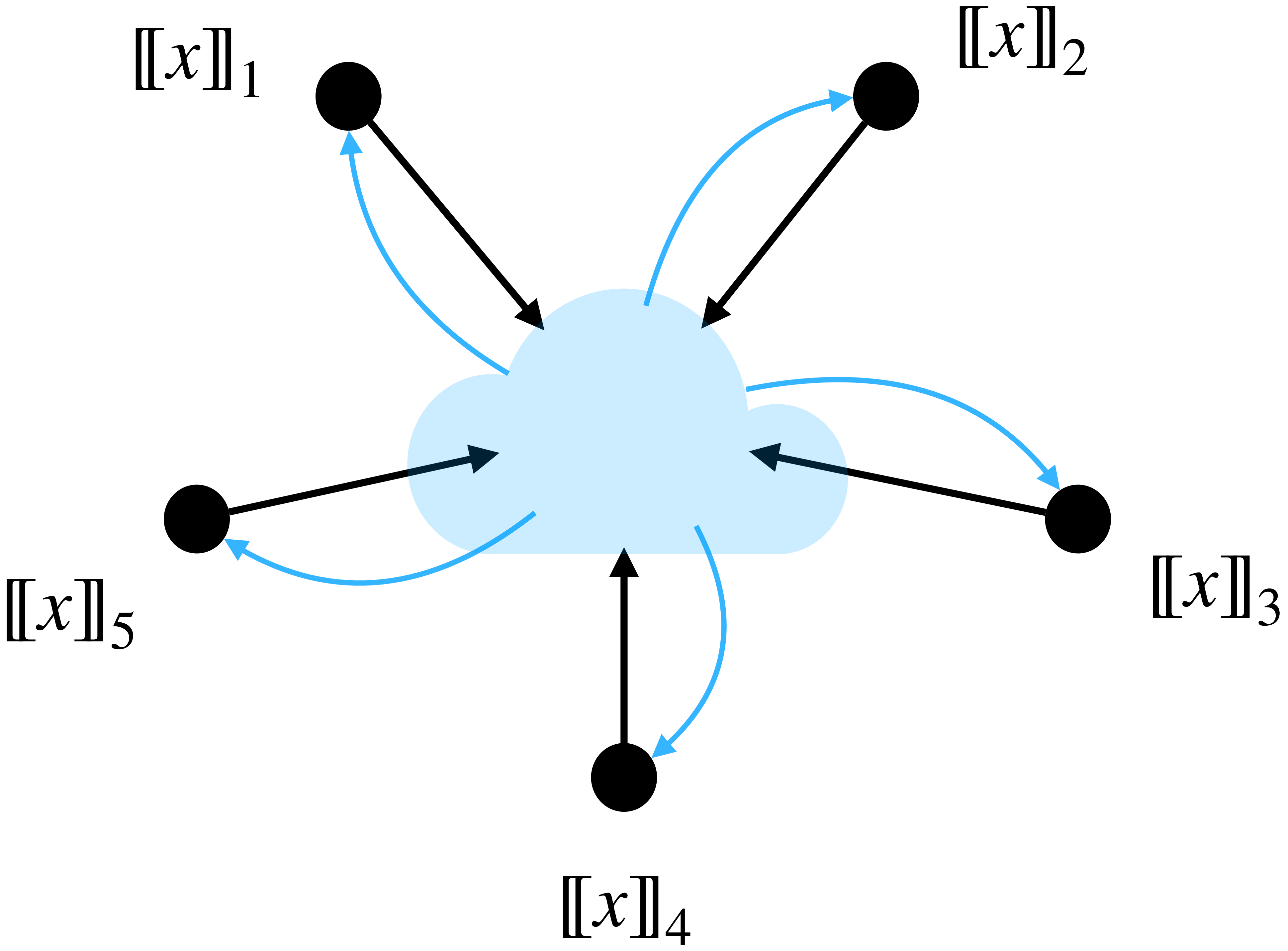
Roadmap

- Technical background
- **MQOM MPC protocol**
- SDitH MPC protocol
- Threshold MPCitH
- MQOM signature scheme
- SDitH signature scheme

MQ problem

- Parameters
    - A field $\mathbb{F}_q$, $n \in \mathbb{N}$ (# variables) , $m \in \mathbb{N}$ (# equations)
- Let
    - $x \leftarrow \mathbb{F}_q^n$ *(MQ solution)*
    - $A_i \leftarrow \mathbb{F}_q^{n\times n} \quad \forall i \in [1:m]$ *(m random matrices)*
    - $b_i \leftarrow \mathbb{F}_q^n \quad \forall i \in [1:m]$ *(m random vectors)*
    - $y = (y_1, \ldots, y_m) \in \mathbb{F}_q^m$ s.t. $\begin{cases} y_1 & = \; x^T A_1 x + b_1^T x \\ & \vdots \\ y_m & = \; x^T A_m x + b_m^T x \end{cases}$
- From $\big(\{A_i\}, \{b_i\}, y\big)$ find x

MQOM MPC protocol



- **Parties receive**
  - $[\![x]\!]$ sharing of the MQ solution
  - $\left(\{A_i\}, \{b_i\}, y\right)$ MQ equations
- **Parties jointly compute**

$$g(x) = \begin{cases} \text{Accept} & \text{if } y_i = x^T A_i x + b_i^T x \quad \forall i \\ \text{Reject} & \text{otherwise} \end{cases}$$

Step 1: batching MQ equations

- Goal: check that $[\![x]\!]$ is s.t. $\underbrace{y_i - x^T A_i\, x - b_i^T x}_{E_i(x)} = 0 \quad \forall i \in [1:m]$
- Randomness oracle $\rightarrow$ $\gamma_1, \ldots, \gamma_m \in \mathbb{F}_q^{\eta}$
- Batched check: $\sum_{i=1}^{m} \gamma_i E_i(x) = 0$

Step 1: batching MQ equations

- Goal: check that $[\![x]\!]$ is s.t. $\underbrace{y_i - x^T A_i x - b_i^T x}_{E_i(x)} = 0 \quad \forall i \in [1:m]$
- Randomness oracle → $\gamma_1, \ldots, \gamma_m \in \mathbb{F}_q^\eta$

  Extension of degree η
- Batched check: $\sum_{i=1}^{m} \gamma_i E_i(x) = 0$

Step 1: batching MQ equations

- Goal: check that $[\![x]\!]$ is s.t. $\underbrace{y_i - x^T A_i x - b_i^T x}_{E_i(x)} = 0 \;\; \forall i \in [1:m]$
- Randomness oracle $\rightarrow$ $\gamma_1, \ldots, \gamma_m \in \mathbb{F}_q^{\eta}$
- Batched check: $\sum_{i=1}^{m} \gamma_i E_i(x) = 0$

Extension of degree η

$\Rightarrow$ False positive probability : $p_1 = \dfrac{1}{q^{\eta}}$

Step 1: batching MQ equations

- Goal: check that $[\![x]\!]$ is s.t. $\underbrace{y_i - x^T A_i x - b_i^T x}_{E_i(x)} = 0 \quad \forall i \in [1 : m]$
- Randomness oracle $\rightarrow \gamma_1, \ldots, \gamma_m \in \boxed{\mathbb{F}_q^\eta}$

Extension of degree η

- Batched check: $\sum_{i=1}^{m} \gamma_i E_i(x) = 0$

$\Rightarrow$ *False positive probability :* $p_1 = \frac{1}{q^\eta}$

- Rewrite as $\langle x, w \rangle = z$

$$z := \sum_{i=1}^{m} \gamma_i (y_i - b_i^T x)$$

$$w := \left(\sum_{i=1}^{m} \gamma_i A_i \right) x$$

Linear (affine) functions of x
$\Rightarrow$ sharings $[\![w]\!]$ and $[\![z]\!]$
locally computed

Step 2: inner product check

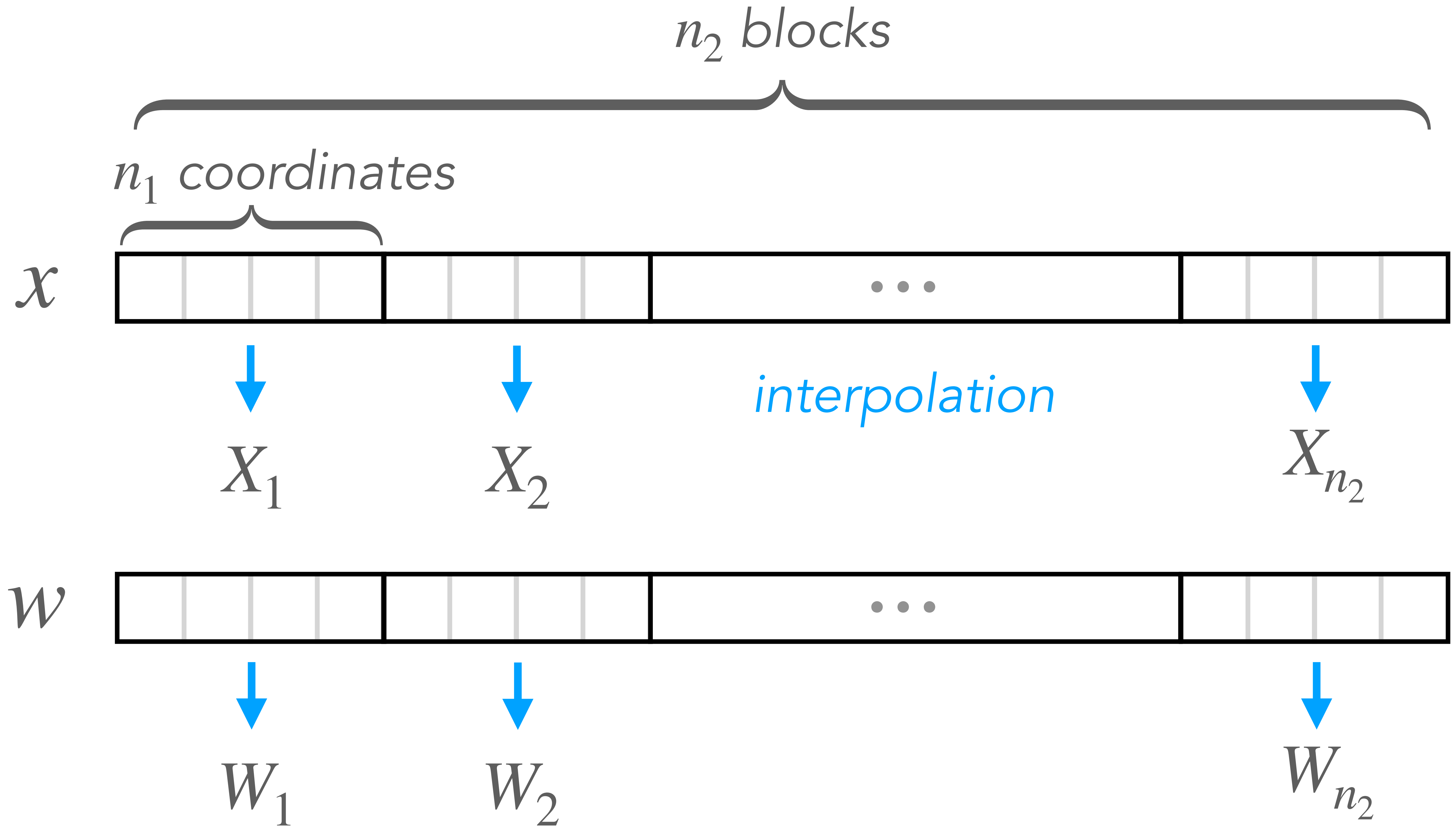
- Goal: check that $[\![x]\!]$, $[\![w]\!]$, $[\![z]\!]$ are s.t. $\langle x, w \rangle = z$

Step 2: inner product check

- Goal: check that $[\![x]\!]$, $[\![w]\!]$, $[\![z]\!]$ are s.t. $\langle x, w\rangle = z$
- Locally interpolate

  $[\![X_1]\!], \ldots, [\![X_{n_2}]\!]$

  $[\![W_1]\!], \ldots, [\![W_{n_2}]\!]$

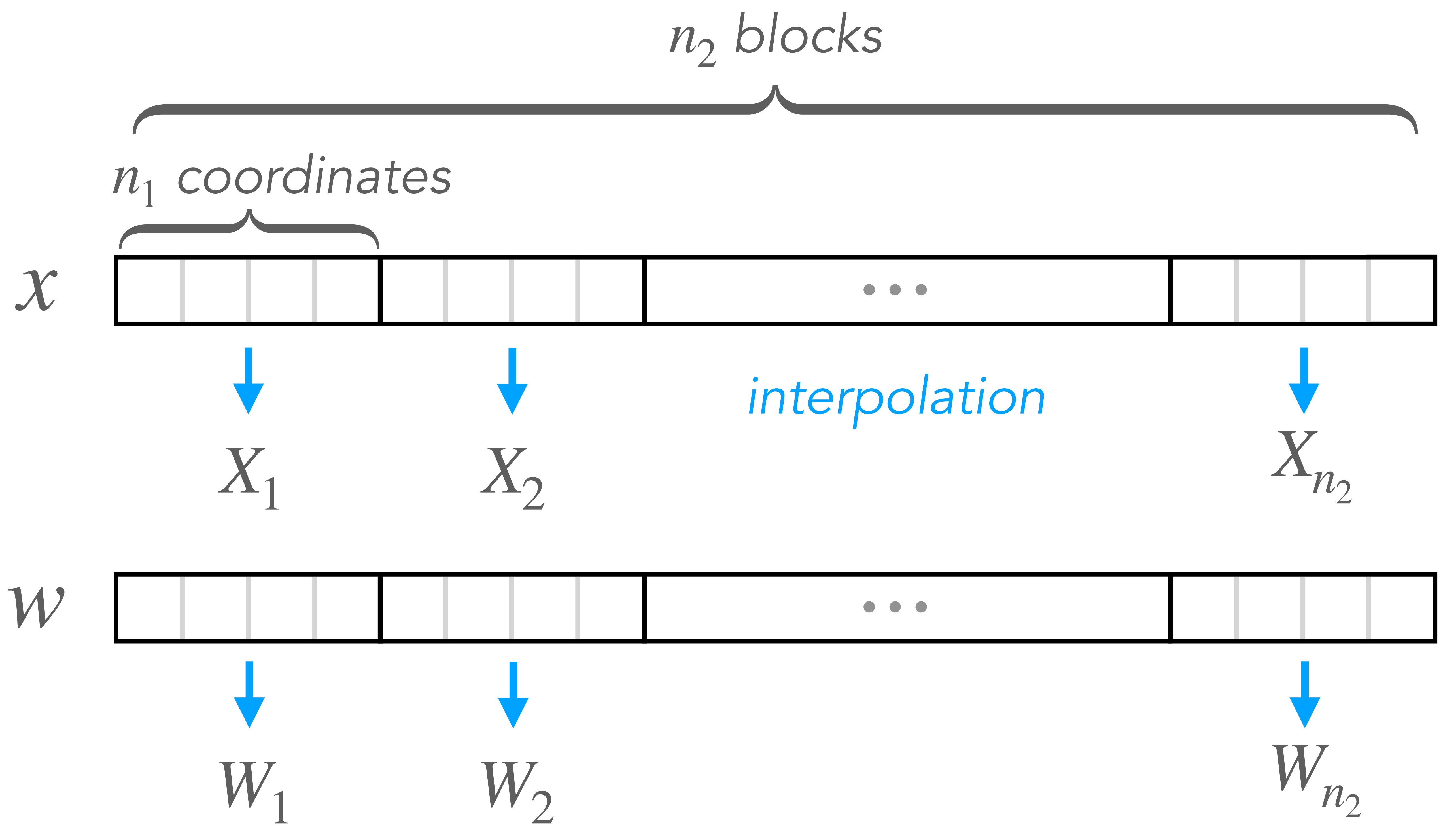


Step 2: inner product check

- Goal: check that $[\![x]\!]$, $[\![w]\!]$, $[\![z]\!]$ are s.t. $\langle x, w \rangle = z$
- Locally interpolate

  $[\![X_1]\!], \ \ldots, [\![X_{n_2}]\!]$

  $[\![W_1]\!], \ldots, [\![W_{n_2}]\!]$

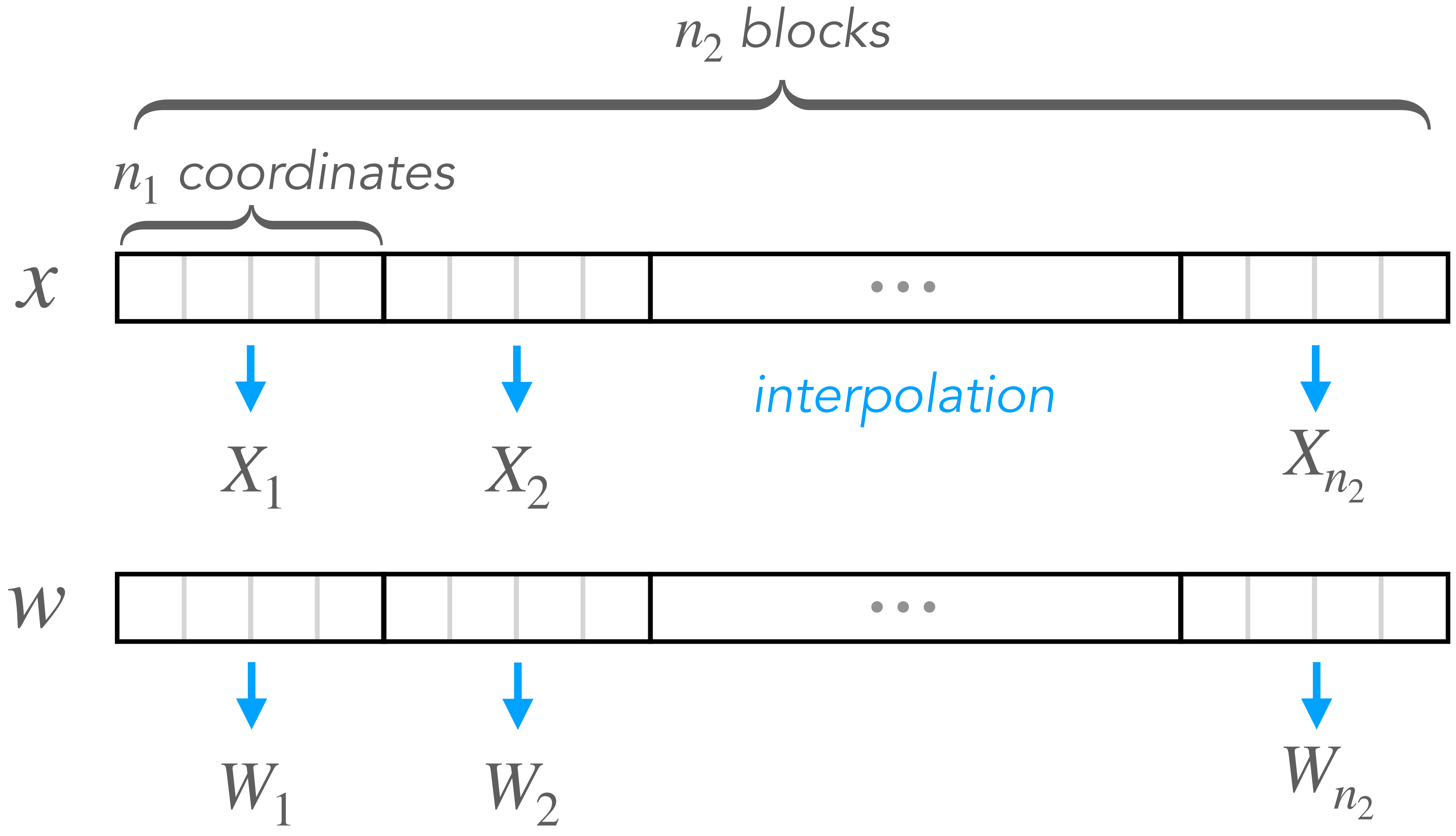


$$\langle x, w \rangle = z \iff \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} X_j(f_i) W_j(f_i) = z$$

Step 2: inner product check

- Goal: check that $[\![x]\!]$, $[\![w]\!]$, $[\![z]\!]$ are s.t. $\langle x, w\rangle = z$
- Locally interpolate
  $[\![X_1]\!], \ \dots, [\![X_{n_2}]\!]$
  $[\![W_1]\!], \dots, [\![W_{n_2}]\!]$
- Hint oracle $\rightarrow$ $[\![Q_0]\!]$ s.t.

  (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$



$$\langle x, w\rangle = z \iff \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} X_j(f_i) W_j(f_i) = z$$

Step 2: inner product check

- Goal: check that $\llbracket x \rrbracket$, $\llbracket w \rrbracket$, $\llbracket z \rrbracket$ are s.t. $\langle x, w \rangle = z$

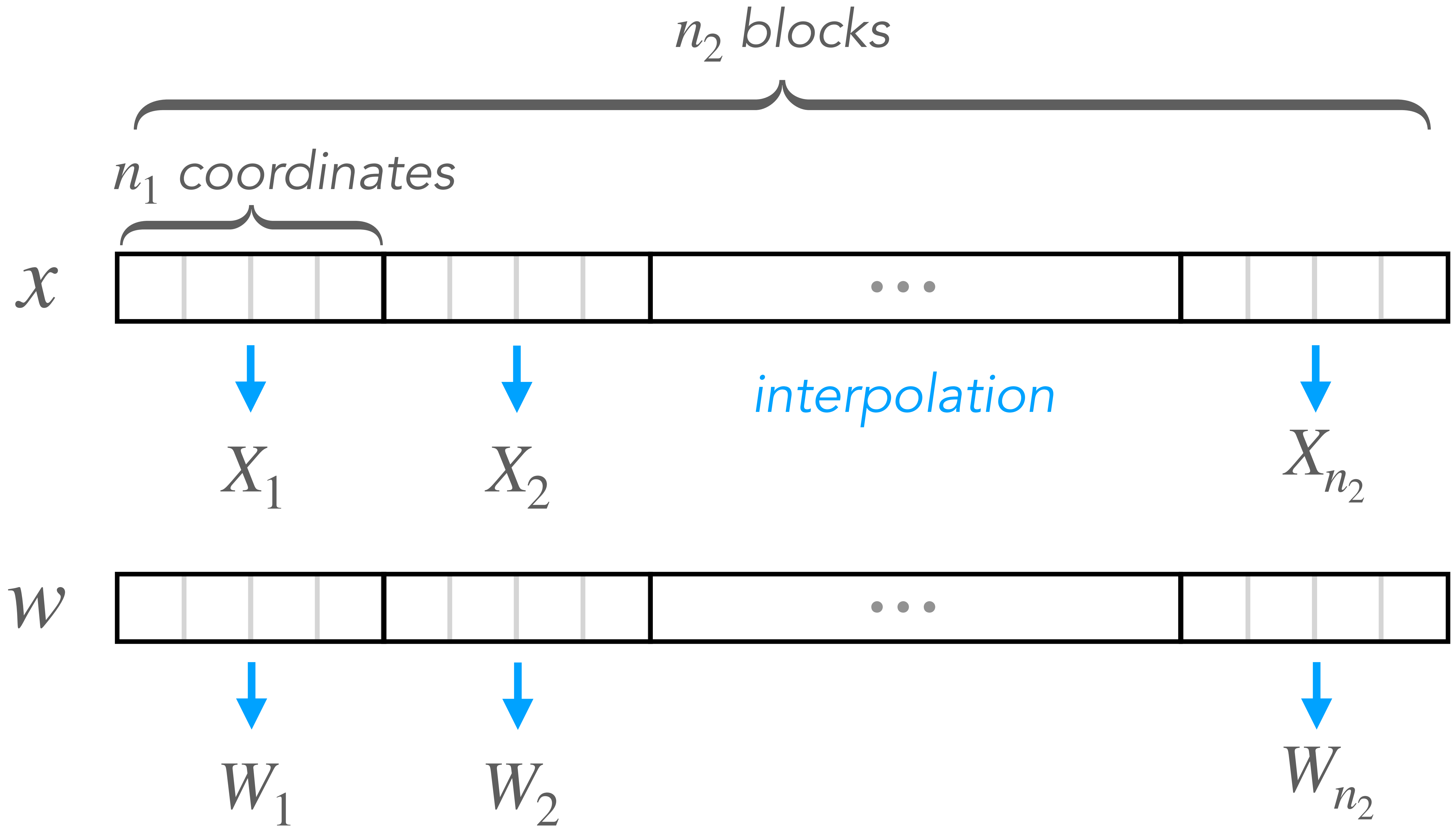
- Locally interpolate
  $\llbracket X_1 \rrbracket, \ \ldots, \llbracket X_{n_2} \rrbracket$
  $\llbracket W_1 \rrbracket, \ldots, \llbracket W_{n_2} \rrbracket$

- Hint oracle → $\llbracket Q_0 \rrbracket$ s.t.

  (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Check that $\llbracket Q_0 \rrbracket$ sat. (1) and

  (2) $\sum_{i=1}^{n_1} Q_0(f_i) = z$



n_2 blocks

n_1 coordinates

x

interpolation

X_1 X_2 X_{n_2}

w

W_1 W_2 W_{n_2}

$$\langle x, w \rangle = z \iff \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} X_j(f_i) W_j(f_i) = z$$

Step 2: inner product check

Checking (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Randomness oracle → r
- Locally compute $[\![\alpha_j]\!] = [\![W_j]\!](r)$
- Broadcast $[\![\alpha_j]\!]$ → publicly recompute $\alpha_j = W_j(r)$
- Locally compute $[\![v_1]\!] = [\![Q_0]\!](r) - \sum_{j=1}^{n_2} \alpha_j \, [\![X_j]\!](r)$
- Broadcast $[\![v_1]\!]$ → publicly recompute v_1
- Check that $v_1 = 0$

Step 2: inner product check

Checking (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Randomness oracle → r
- Locally compute $[\![\alpha_j]\!] = [\![W_j]\!](r)$
- Broadcast $[\![\alpha_j]\!]$ → publicly recompute $\alpha_j = W_j(r)$
- Locally compute $[\![v_1]\!] = [\![Q_0]\!](r) - \sum_{j=1}^{n_2} \alpha_j\, [\![X_j]\!](r)$
- Broadcast $[\![v_1]\!]$ → publicly recompute v_1
- Check that $v_1 = 0$

False positive proba:

$p_2 \approx 2n_1 / q^\eta$

Step 2: inner product check

Checking (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Randomness oracle $\rightarrow$ r
- Locally compute $[\![\alpha_j]\!] = [\![W_j]\!](r)$
- Broadcast $[\![\alpha_j]\!]$ $\rightarrow$ publicly recompute $\alpha_j = W_j(r)$
- Locally compute $[\![v_1]\!] = [\![Q_0]\!](r) - \sum_{j=1}^{n_2} \alpha_j\, [\![X_j]\!](r)$
- Broadcast $[\![v_1]\!]$ $\rightarrow$ publicly recompute v_1
- Check that $v_1 = 0$

False positive proba:
$p_2 \approx 2n_1/q^\eta$

⚠ *α_j leaks information on w*
Must be masked for ZK to hold

Step 2: inner product check

Checking (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Randomness oracle → r
- Locally compute $[\![\alpha_j]\!] = [\![W_j]\!](r)$
- Broadcast $[\![\alpha_j]\!]$ → publicly recompute $\alpha_j = W_j(r)$
- Locally compute $[\![v_1]\!] = [\![Q_0]\!](r) - \sum_{j=1}^{n_2} \alpha_j [\![X_j]\!](r)$
- Broadcast $[\![v_1]\!]$ → publicly recompute v_1
- Check that $v_1 = 0$

False positive proba:
$p_2 \approx 2n_1/q^\eta$

⚠ *α_j leaks information on w*
Must be masked for ZK to hold

Checking (2) $\sum_{i=1}^{n_1} Q_0(f_i) = z$

- Locally compute

$$[\![v_2]\!] = [\![z]\!] - \sum_{i=1}^{n_1} [\![Q_0]\!](f_i)$$

- Broadcast $[\![v_2]\!]$
  → publicly recompute v_2
- Check that $v_2 = 0$

Step 2: inner product check

Checking (1) $Q_0 = \sum_{j=1}^{n_2} X_j W_j$

- Randomness oracle → r
- Locally compute $[\![\alpha_j]\!] = [\![W_j]\!](r)$
- Broadcast $[\![\alpha_j]\!]$ → publicly recompute $\alpha_j = W_j(r)$
- Locally compute $[\![v_1]\!] = [\![Q_0]\!](r) - \sum_{j=1}^{n_2} \alpha_j\, [\![X_j]\!](r)$
- Broadcast $[\![v_1]\!]$ → publicly recompute v_1
- Check that $v_1 = 0$

False positive proba:
$p_2 \approx 2n_1/q^\eta$

⚠️ *α_j leaks information on w*
Must be masked for ZK to hold

Checking (2) $\sum_{i=1}^{n_1} Q_0(f_i) = z$

- Locally compute

$$[\![v_2]\!] = [\![z]\!] - \sum_{i=1}^{n_1} [\![Q_0]\!](f_i)$$

- Broadcast $[\![v_2]\!]$
  → publicly recompute v_2
- Check that $v_2 = 0$

💡 *We save communication by removing constant term of Q_0 from the hint*

Roadmap

- Technical background
- MQOM MPC protocol
- **SDitH MPC protocol**
- Threshold MPCitH
- MQOM signature scheme
- SDitH signature scheme

Syndrome decoding problem

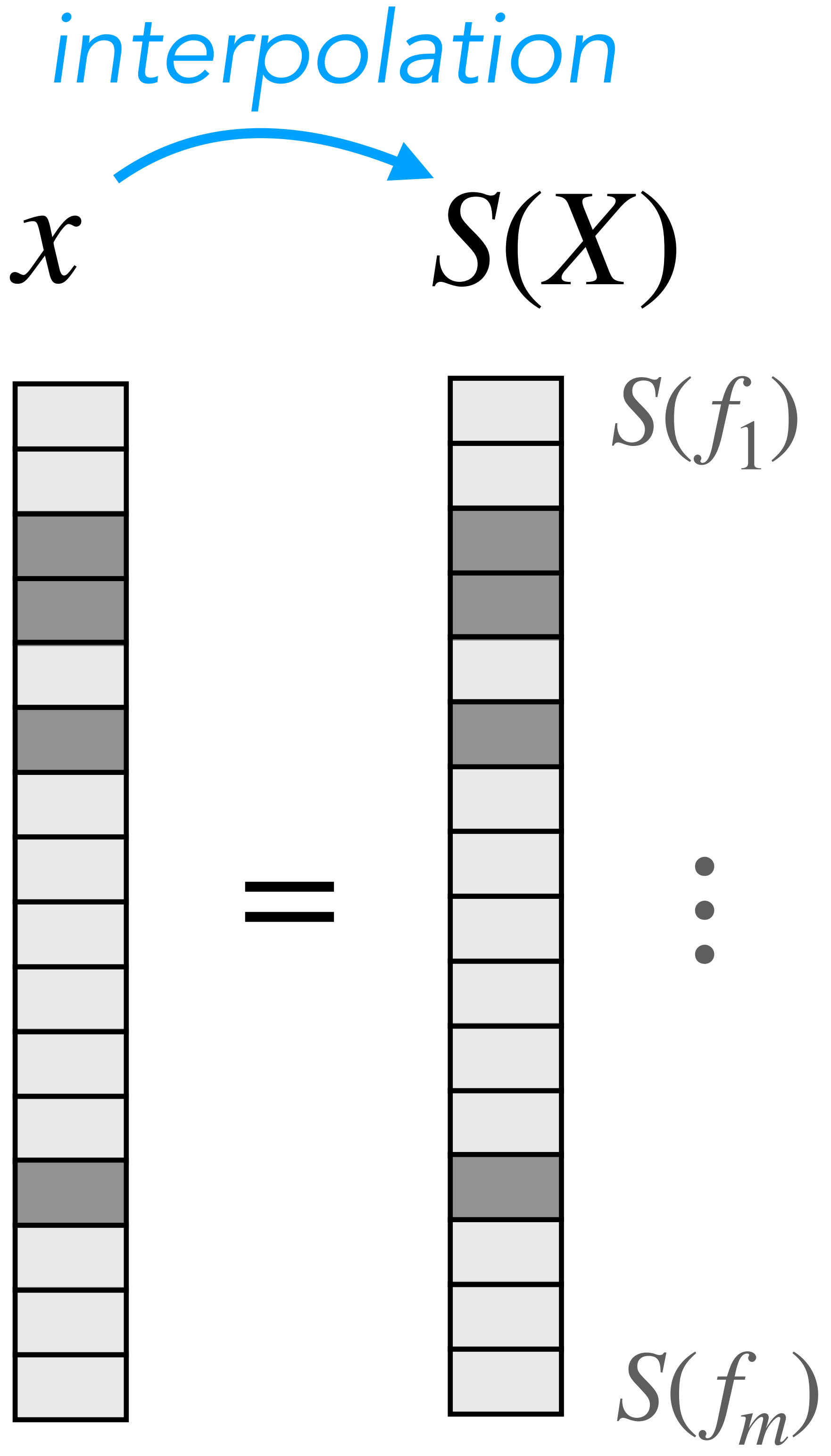
- Parameters
  - A field $\mathbb{F}_q$, $m \in \mathbb{N}$ *(code length)* , $k < m$ *(code dimension)* , $w < m$ *(weight)*
- Let
  - $H \leftarrow \mathbb{F}_q^{(m-k)\times m}$ *(random parity-check matrix)*
  - $x \leftarrow \mathbb{F}_q^m$ s.t. $\mathrm{wt}(x) \leq w$ *(SD solution)*
  - $y = Hx$ *(syndrome)*
- From (H, y) find x

Syndrome decoding problem

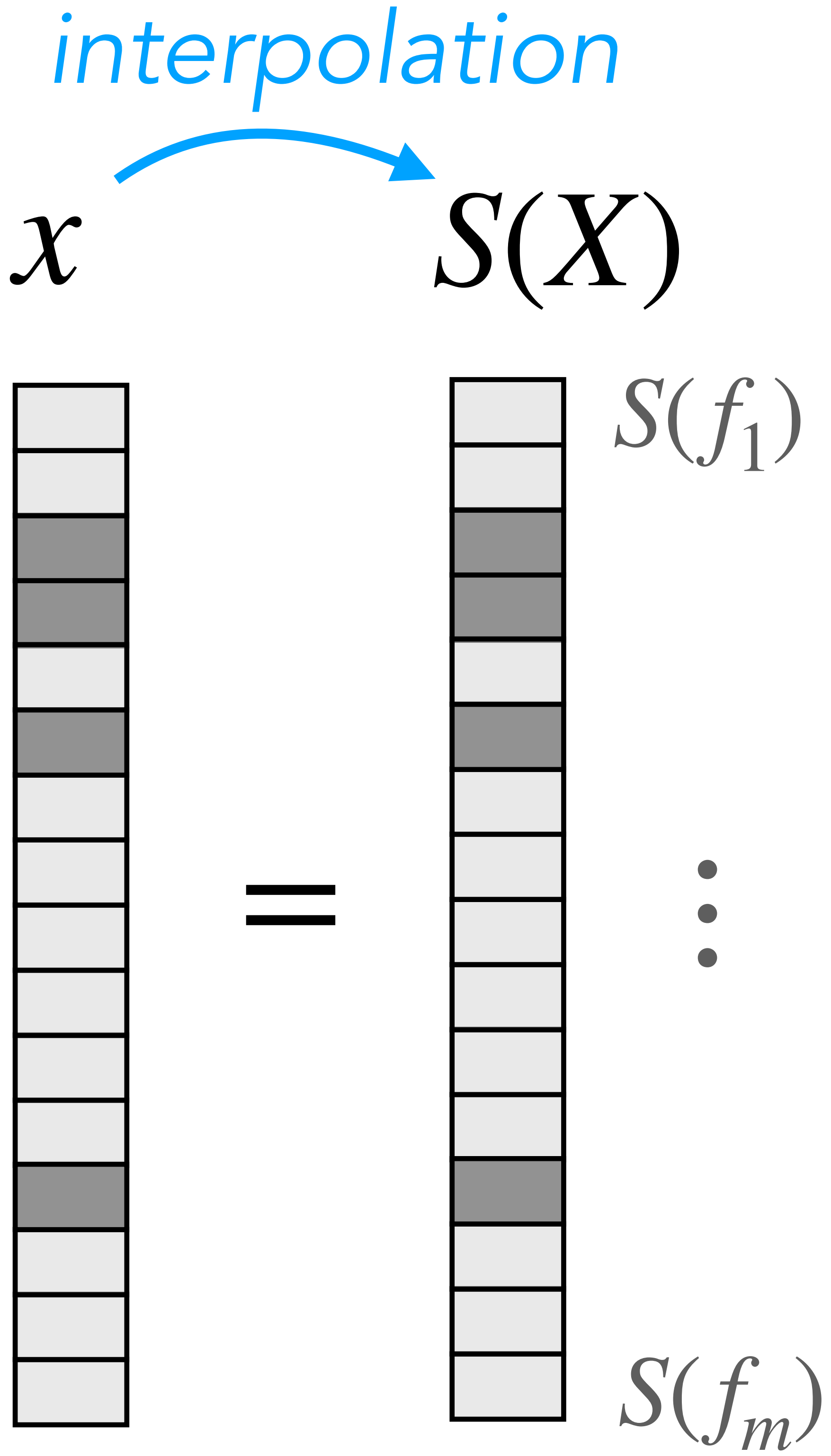
- Parameters
  - A field $\mathbb{F}_q$, $m \in \mathbb{N}$ *(code length)* , $k < m$ *(code dimension)* , $w < m$ *(weight)*
- Let
  - $H \leftarrow \mathbb{F}_q^{(m-k)\times m}$ *(random parity-check matrix)*
  - $x \leftarrow \mathbb{F}_q^{m}$ s.t. $\mathrm{wt}(x) \leq w$ *(SD solution)*
  - $y = Hx$ *(syndrome)*
- From (H, y) find x
- Standard form (wlog): $H = (H' \,|\, I_{m-k})$ $\Rightarrow$ $y = H'x_A + x_B$ where $x = (\overset{|x_A| = k}{x_A} \,|\, \overset{|x_B| = m-k}{x_B})$

$$\Rightarrow \quad x_B = y - H'x_A$$

Polynomial constraints

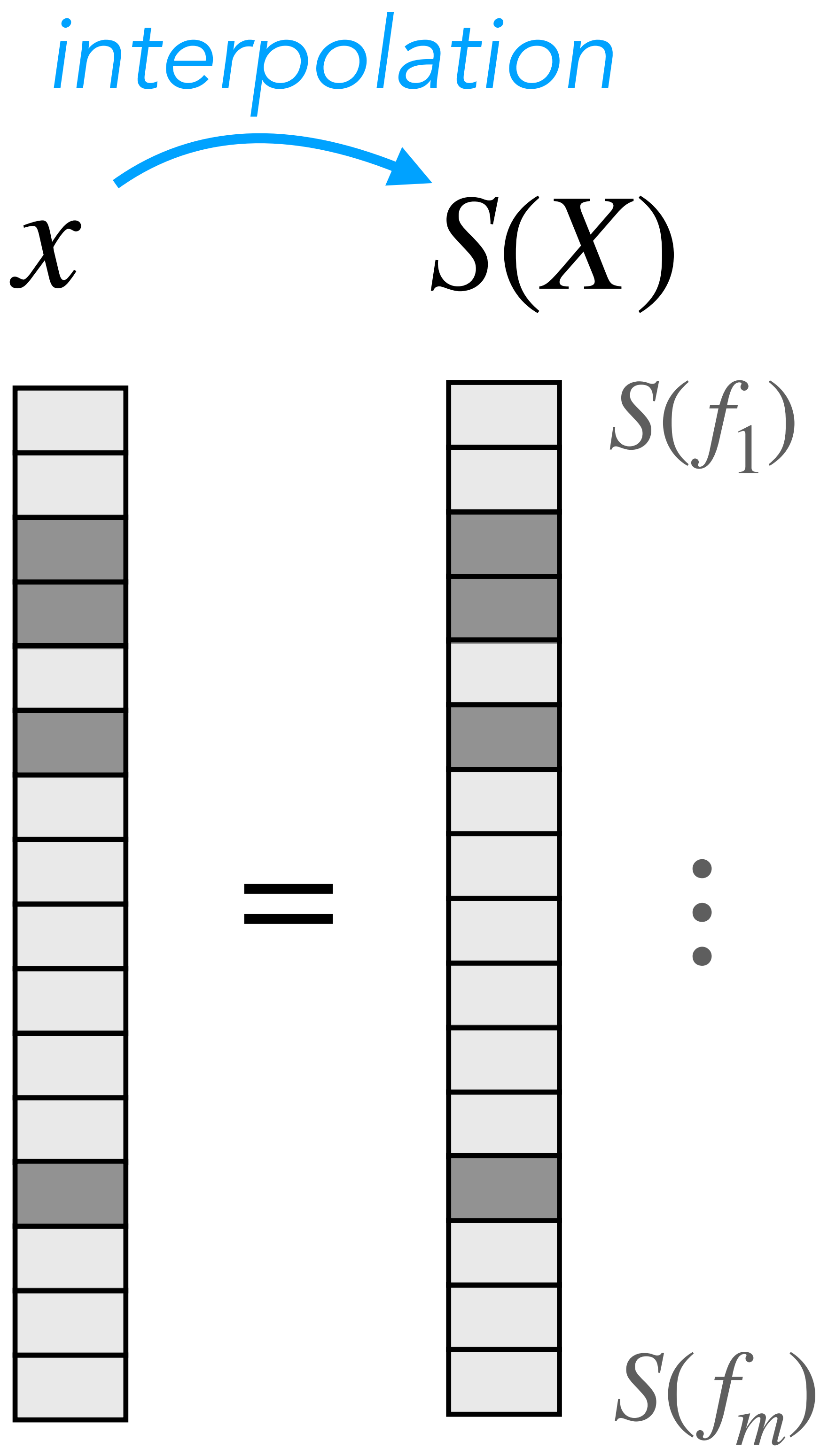


Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

Polynomial constraints

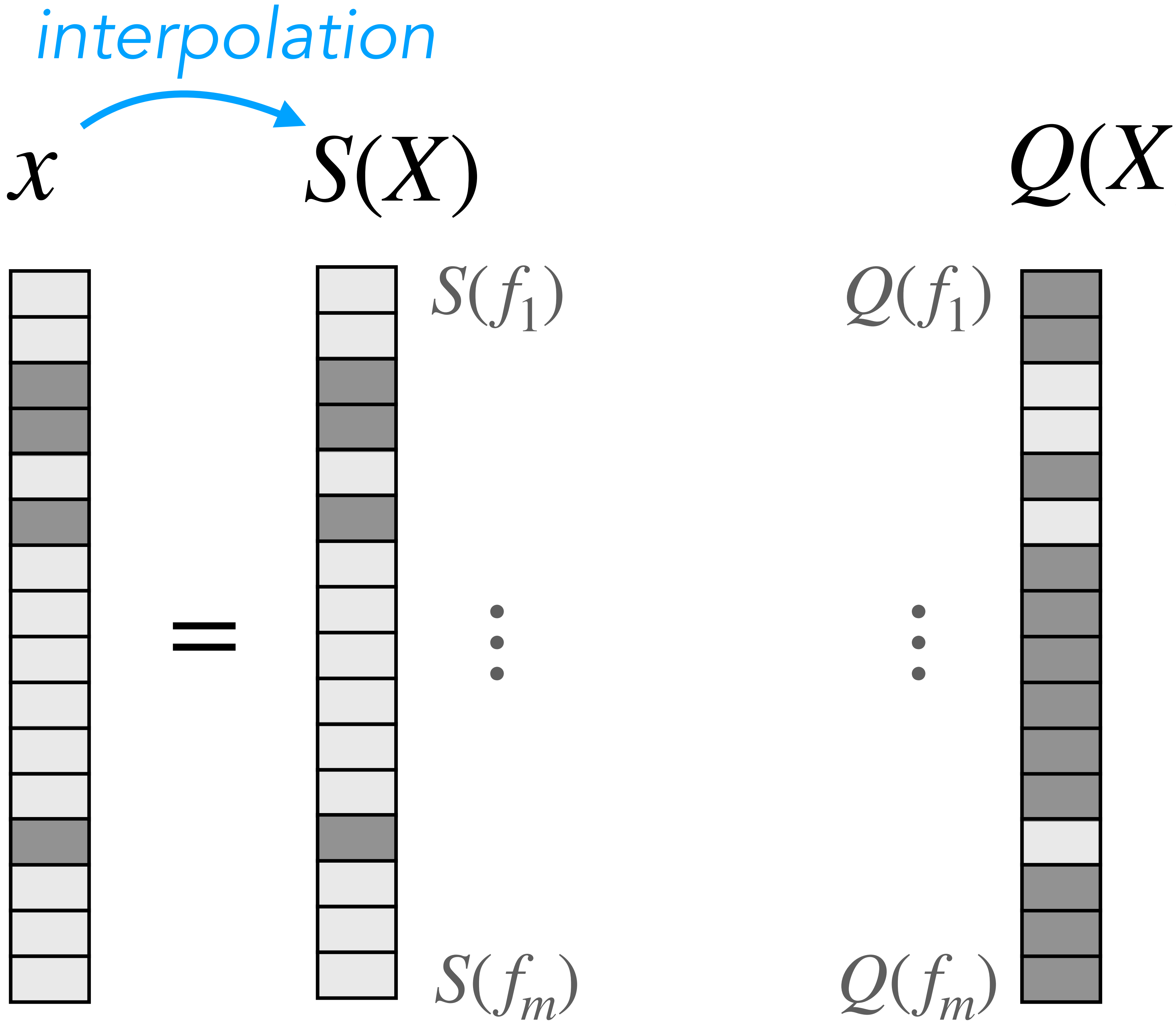


$$Q(X) = \prod_{i \in E} (X - f_i)$$

indices i s.t. $x_i \neq 0$

$|E| \leq w \;\Rightarrow\; \deg(Q) \leq w$

Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

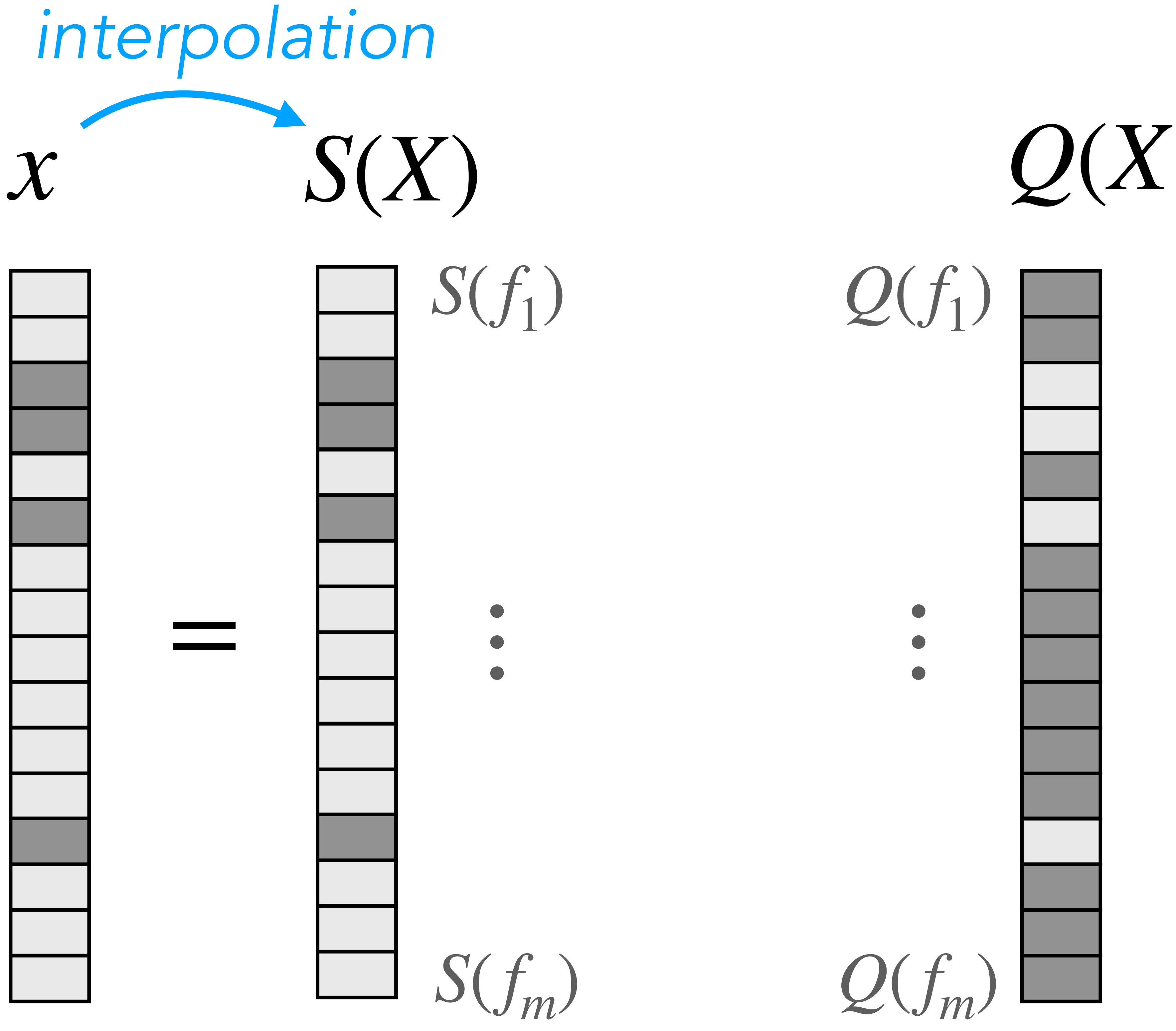
indices i s.t. $x_i \neq 0$

$|E| \leq w \Rightarrow \deg(Q) \leq w$

▢ = zero coordinate

▨ = non-zero coordinate

Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

indices i s.t. $x_i \neq 0$

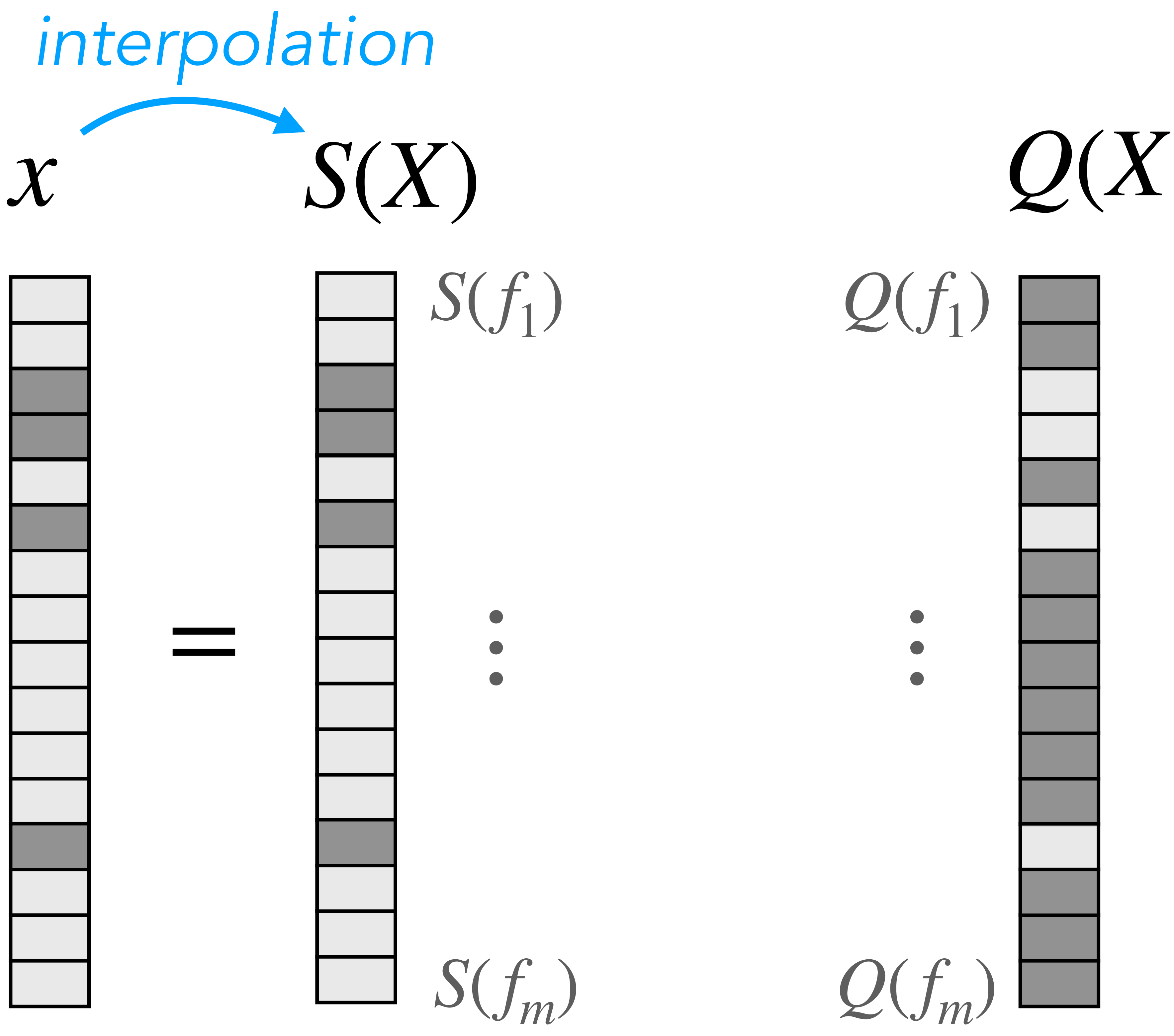
$|E| \leq w \;\Rightarrow\; \deg(Q) \leq w$

□ = zero coordinate

■ = non-zero coordinate

$\Rightarrow$ $S(X) \cdot Q(X)$ evaluates to 0 in $f_1, \ldots, f_m$

Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

indices i s.t. $x_i \neq 0$

$|E| \leq w \;\Rightarrow\; \deg(Q) \leq w$

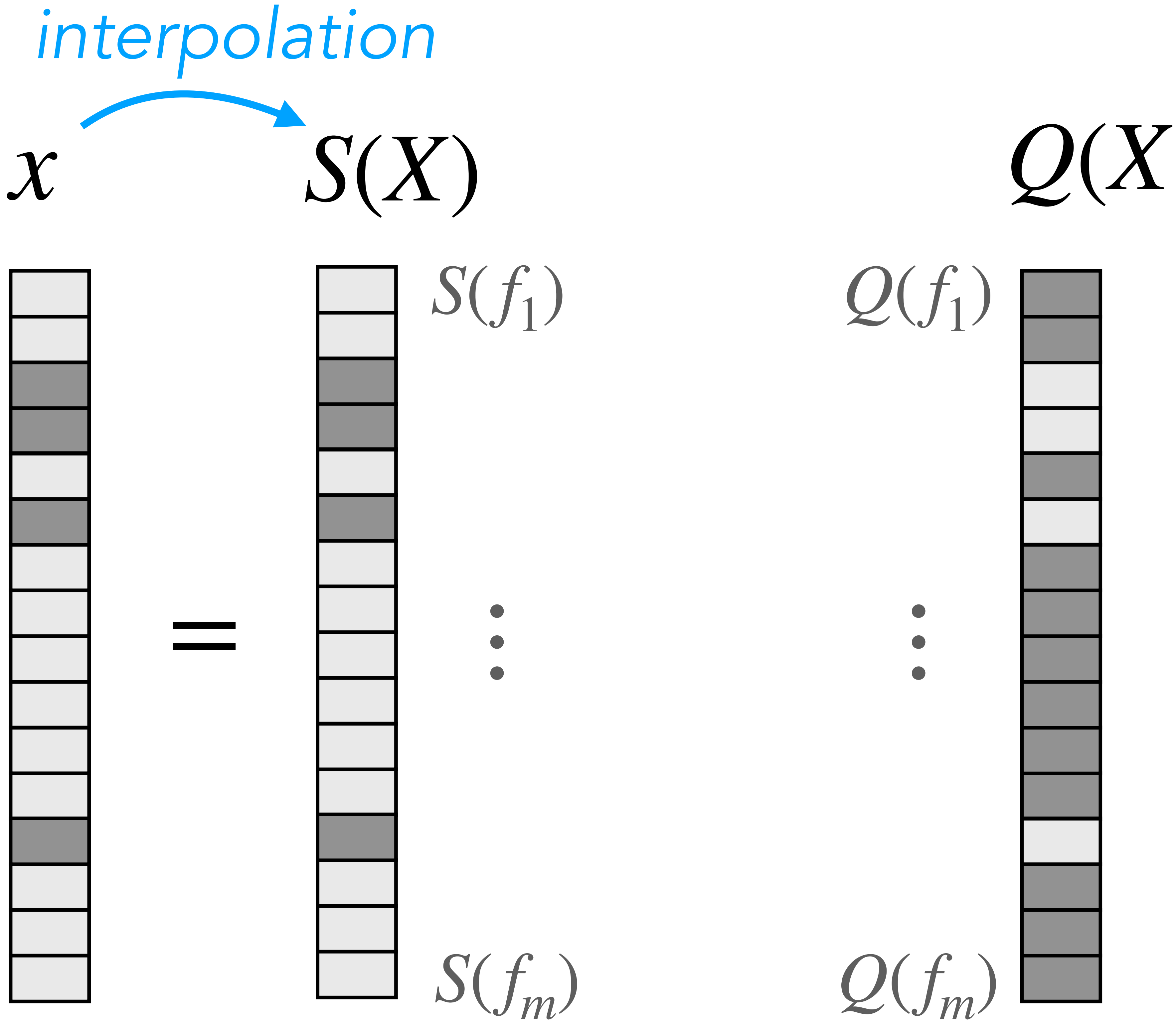
☐ = zero coordinate

■ = non-zero coordinate

$\Rightarrow$ $S(X) \cdot Q(X)$ evaluates to 0 in $f_1, \ldots, f_m$

$\Rightarrow$ $S(X) \cdot Q(X) = F(X) \cdot P(X)$

Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

indices i s.t. $x_i \neq 0$

$|E| \leq w \;\Rightarrow\; \deg(Q) \leq w$

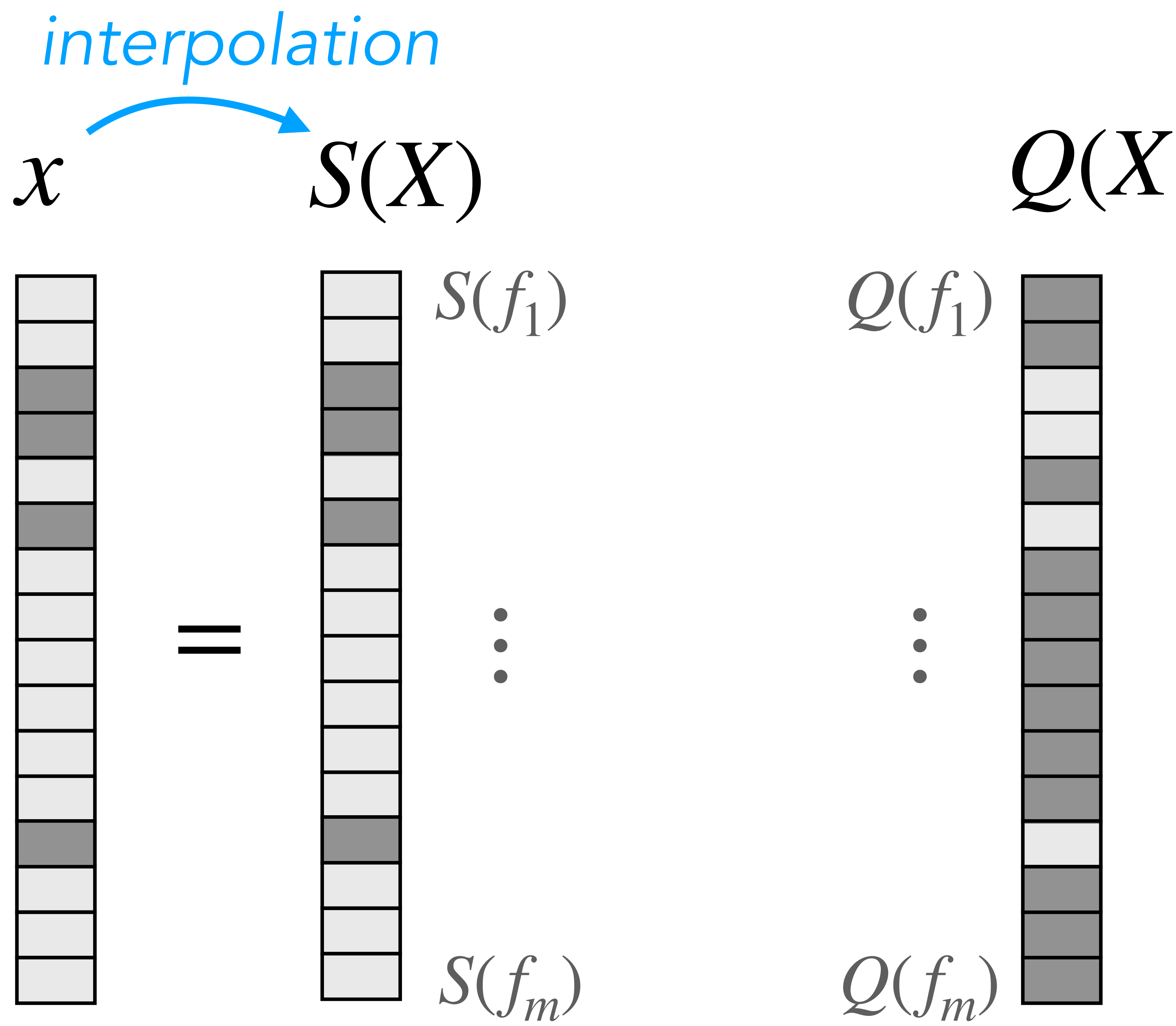
☐ = zero coordinate

▨ = non-zero coordinate

$\Rightarrow$ $S(X) \cdot Q(X)$ evaluates to 0 in $f_1, \ldots, f_m$

$$\Rightarrow \; S(X) \cdot Q(X) = \underbrace{F(X)}_{\prod_{i \in [1:m]} (X - f_i)} \cdot P(X)$$

Polynomial constraints



$$Q(X) = \prod_{i \in E} (X - f_i)$$

indices i s.t. $x_i \neq 0$

$|E| \leq w \Rightarrow \deg(Q) \leq w$

□ = zero coordinate

■ = non-zero coordinate

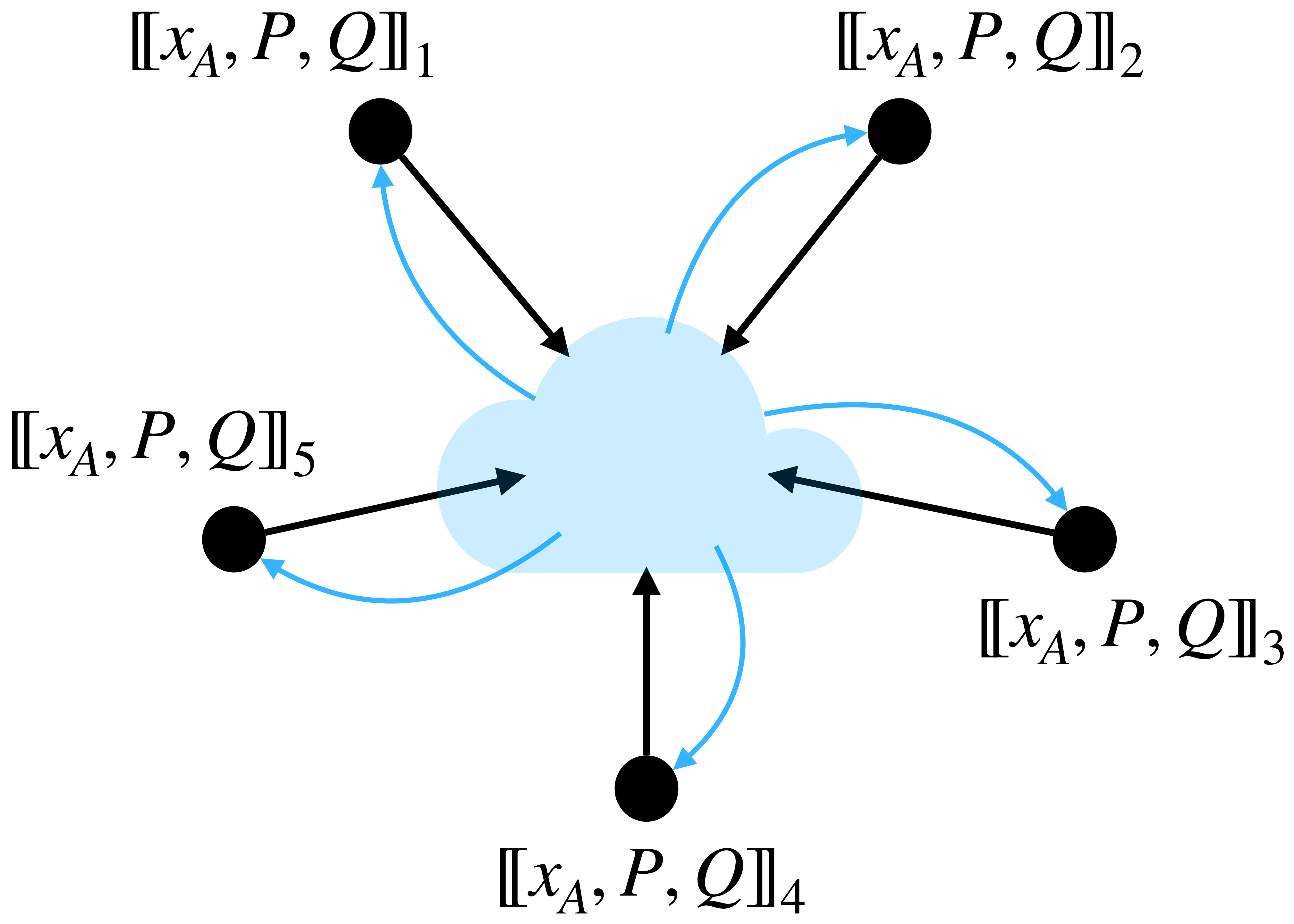
$\Rightarrow S(X) \cdot Q(X)$ evaluates to 0 in $f_1, \ldots, f_m$

$\Rightarrow S(X) \cdot Q(X) = F(X) \cdot P(X)$

$\prod_{i \in [1:m]} (X - f_i)$

some degree $\leqslant w - 1$ polynomial

SDitH MPC protocol



- **Parties receive**
  - $[\![x_A]\!], [\![P]\!], [\![Q]\!]$ sharings of x_A, P, Q
  - (H', y) SD instance
- **Parties jointly compute**

$$g(x_A, P, Q) = \begin{cases} \text{Accept} & \text{if } SQ = FP \\ \text{Reject} & \text{otherwise} \end{cases}$$

where $x_B = y - H'x_A$ and $S = \mathrm{Interp}(x_A \,|\, x_B)$

SDitH MPC protocol

- Principle: check $SQ = FP$ on t random points (SZ lemma)

1. Locally compute $[\![x_B]\!] = y - H' [\![x_A]\!]$
2. Locally compute $[\![S]\!]$ by Lagrange interpolation of $[\![x]\!] = ([\![x_A]\!] \mid [\![x_B]\!])$
3. Randomness oracle $\rightarrow$ $r_1, \ldots, r_t \in \mathbb{F}_q^\eta$
4. Locally compute $[\![S(r_i)]\!], [\![Q(r_i)]\!], F(r_i) \cdot [\![P(r_i)]\!] \quad \forall i \in [1:t]$
5. Check the product $S(r_i) \cdot Q(r_i) = F(r_i) \cdot P(r_i)$ from the shares

SDitH MPC protocol

- Principle: check $SQ = FP$ on t random points (SZ lemma)

1. Locally compute $[\![x_B]\!] = y - H' [\![x_A]\!]$
2. Locally compute $[\![S]\!]$ by Lagrange interpolation of $[\![x]\!] = ([\![x_A]\!] \mid [\![x_B]\!])$
3. Randomness oracle $\rightarrow$ $r_1, \ldots, r_t \in \mathbb{F}_q^\eta$, $\varepsilon_1, \ldots, \varepsilon_t \in \mathbb{F}_q^\eta$
4. Locally compute $[\![S(r_i)]\!], [\![Q(r_i)]\!], F(r_i) \cdot [\![P(r_i)]\!] \quad \forall i \in [1:t]$
5. Check the product $S(r_i) \cdot Q(r_i) = F(r_i) \cdot P(r_i)$ from the shares
    - using [BN20] product-check protocol

SDitH MPC protocol

- Principle: check $SQ = FP$ on t random points (SZ lemma)

1. Locally compute $[\![x_B]\!] = y - H'[\![x_A]\!]$
2. Locally compute $[\![S]\!]$ by Lagrange interpolation of $[\![x]\!] = ([\![x_A]\!] \mid [\![x_B]\!])$
3. Randomness oracle $\rightarrow$ $r_1, \ldots, r_t \in \mathbb{F}_q^\eta$, $\varepsilon_1, \ldots, \varepsilon_t \in \mathbb{F}_q^\eta$
4. Locally compute $[\![S(r_i)]\!], [\![Q(r_i)]\!], F(r_i) \cdot [\![P(r_i)]\!] \quad \forall i \in [1:t]$
5. Check the product $S(r_i) \cdot Q(r_i) = F(r_i) \cdot P(r_i)$ from the shares
    - using [BN20] product-check protocol

- False positive probability: $p = \sum_{i=0}^{t} \binom{t}{i} \left(\frac{m+w-1}{q^\eta}\right)^i \left(1 - \frac{m+w-1}{q^\eta}\right)^{t-i} \left(\frac{1}{q^\eta}\right)^{t-i}$

Roadmap

- Technical background
- MQOM MPC protocol
- SDitH MPC protocol
- **Threshold MPCitH**
- MQOM signature scheme
- SDitH signature scheme

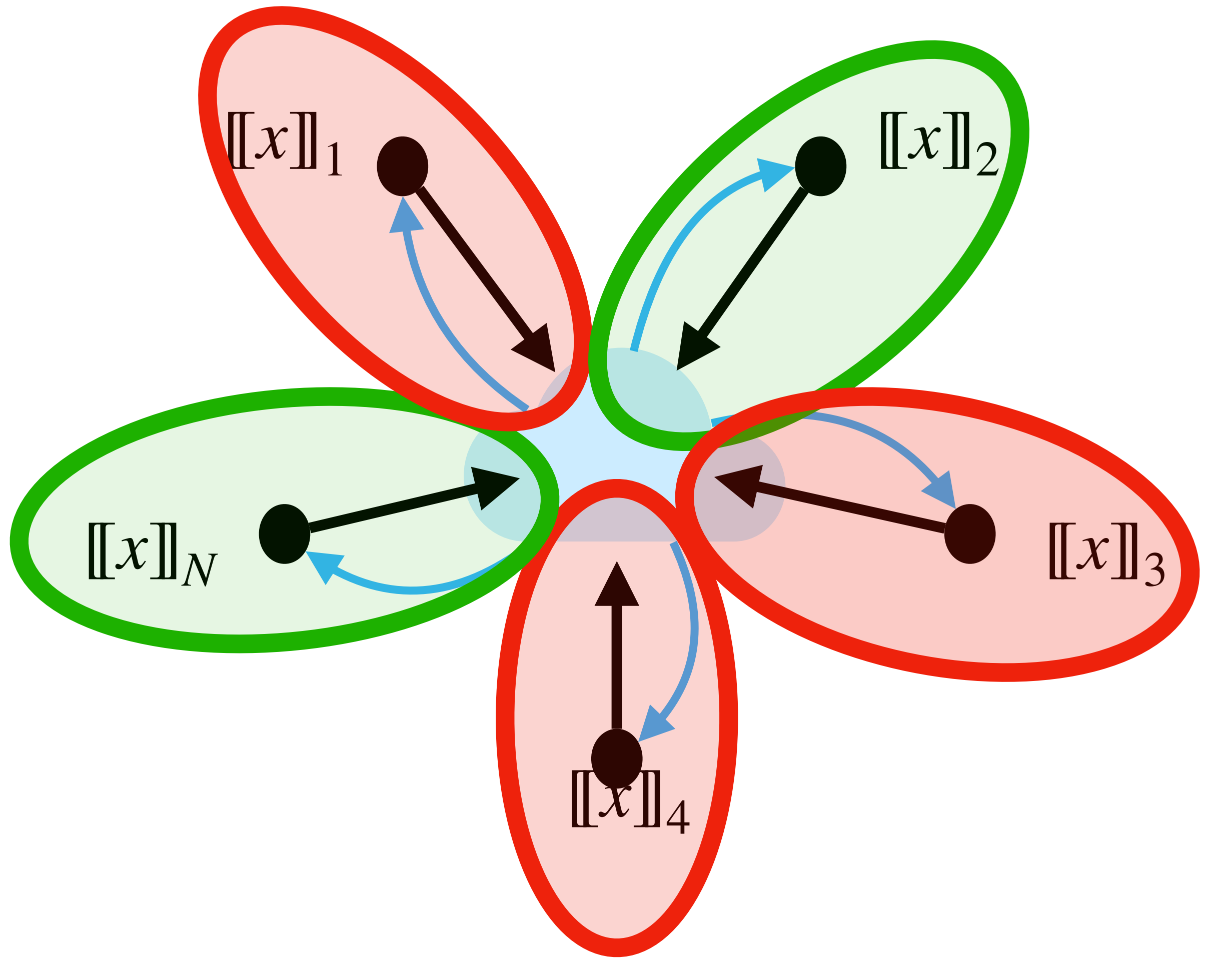
Threshold MPCitH

- **[FR22]** MPCitH using $(\ell + 1, N)$-threshold LSSS (linear secret sharing)
    - Linearity: $[\![x]\!] + [\![y]\!] = [\![x + y]\!]$
    - Any set of ℓ shares is random and independent of x
    - Any set of $\ell + 1$ shares → all the shares
    - Example: Shamir's secret sharing

Threshold MPCitH

- **[FR22]** MPCitH using $(\ell + 1, N)$-threshold LSSS (linear secret sharing)
  - Linearity: $[\![x]\!] + [\![y]\!] = [\![x + y]\!]$
  - Any set of ℓ shares is random and independent of x
  - Any set of $\ell + 1$ shares → all the shares
  - Example: Shamir's secret sharing
- ZK property $\Rightarrow$ only open ℓ parties

E.g. $\ell = 2$

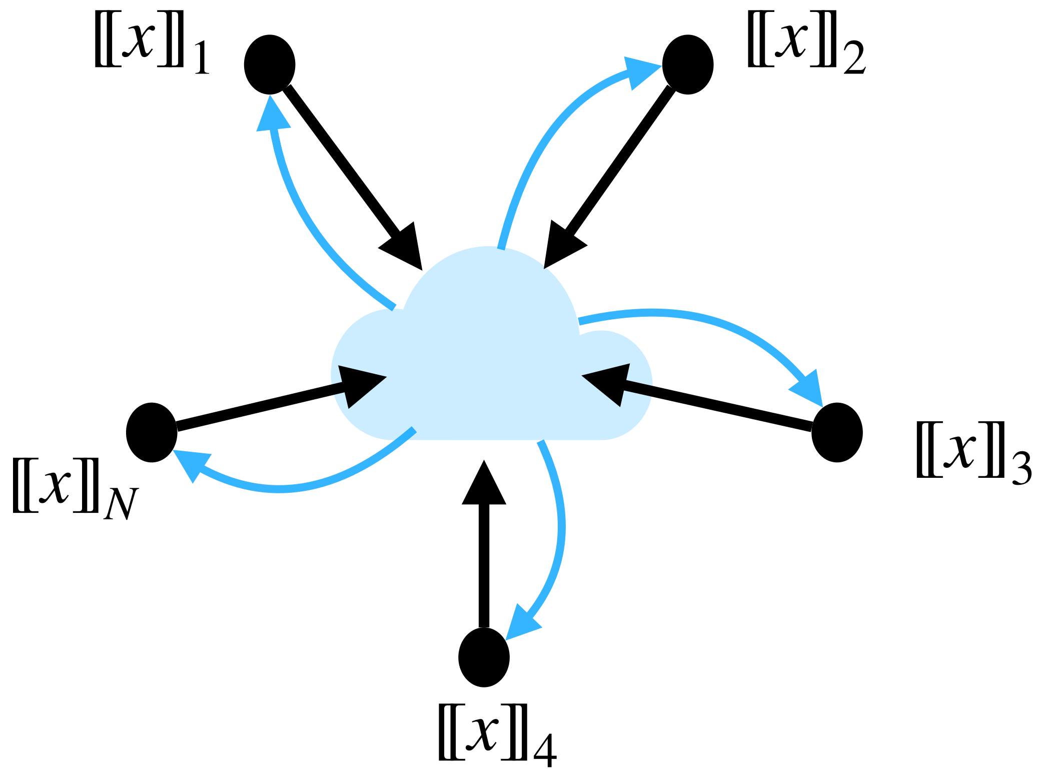


MPCitH transform with threshold LSSS

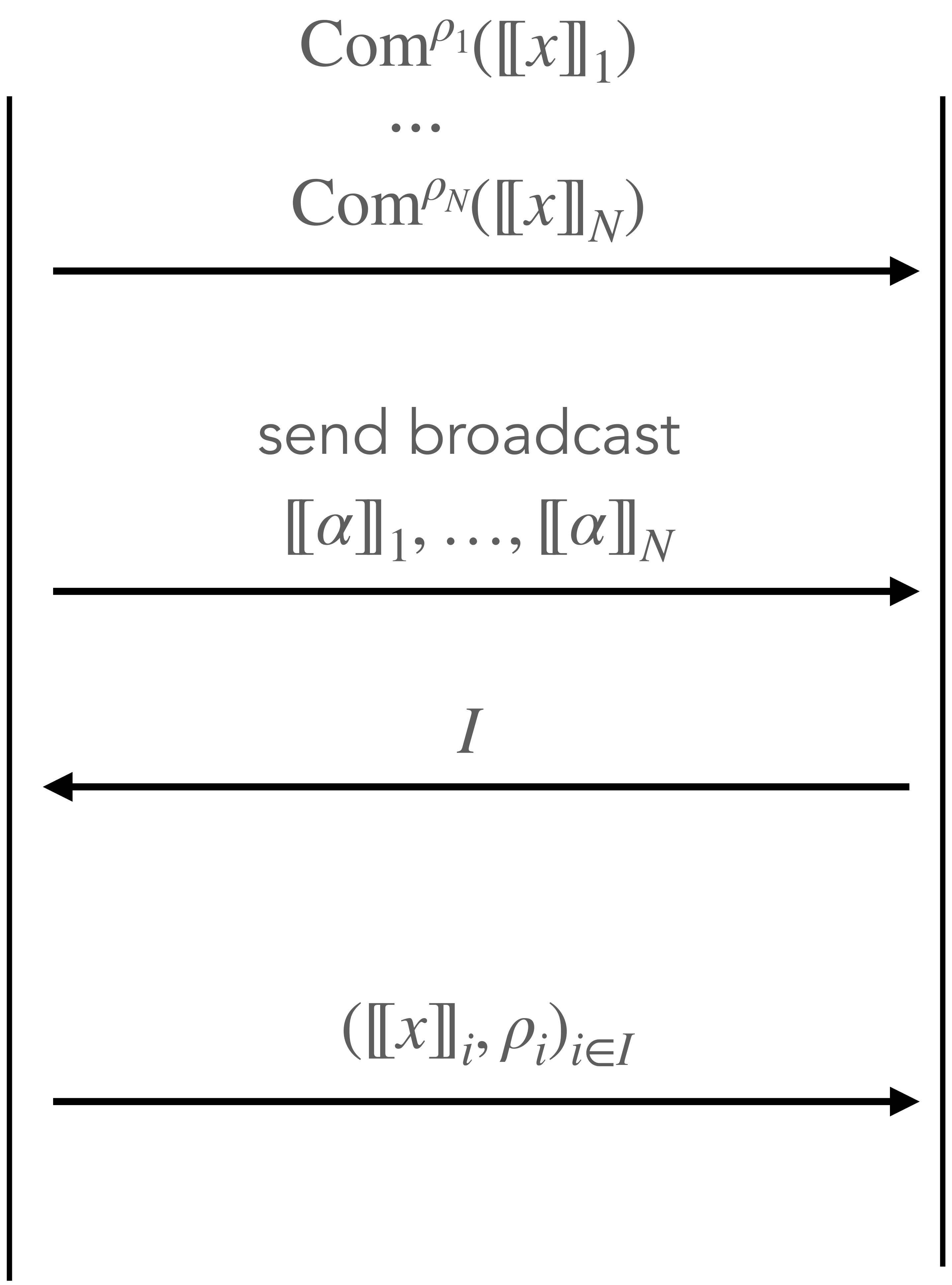
① Generate and commit shares

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

② Run MPC in their head



④ Open parties in I



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

$\cdots$

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

$\longrightarrow$

send broadcast

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

$\longrightarrow$

I

$\longleftarrow$

$([\![x]\!]_i, \rho_i)_{i \in I}$

$\longrightarrow$

③ Chose random set of parties

$I \subseteq \{1,\ldots,N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

Check $g(y, \alpha) =$ Accept

Prover

Verifier

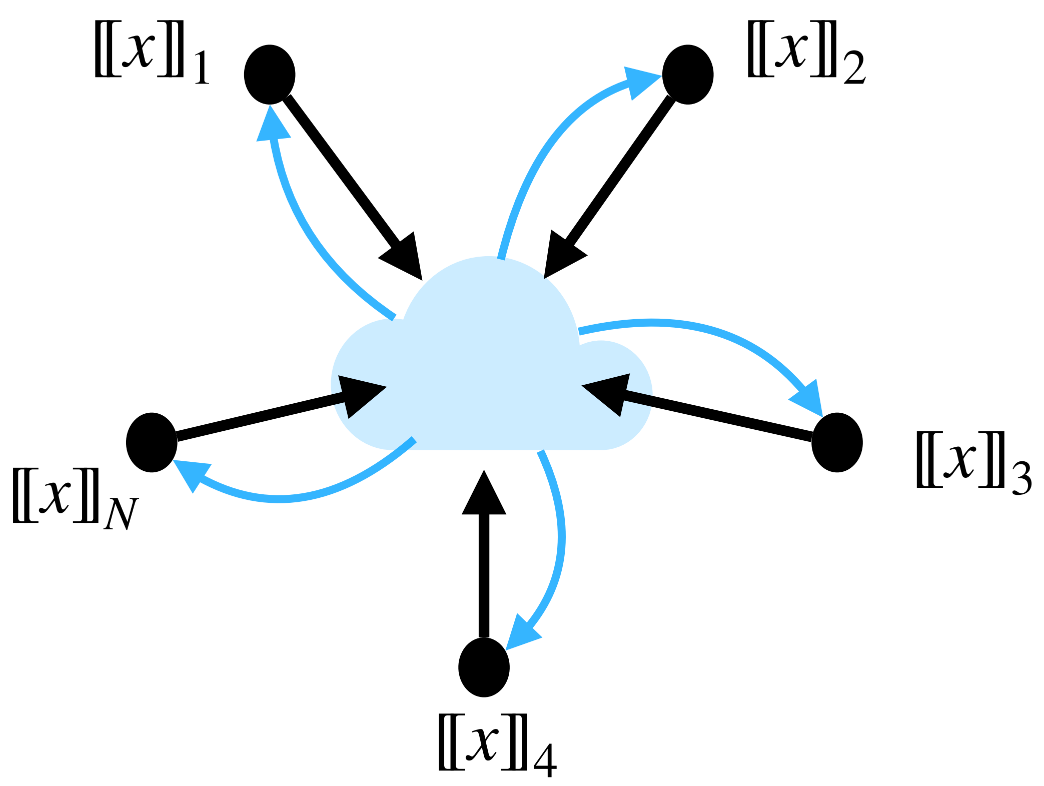
MPCitH transform with threshold LSSS

① Generate and commit shares

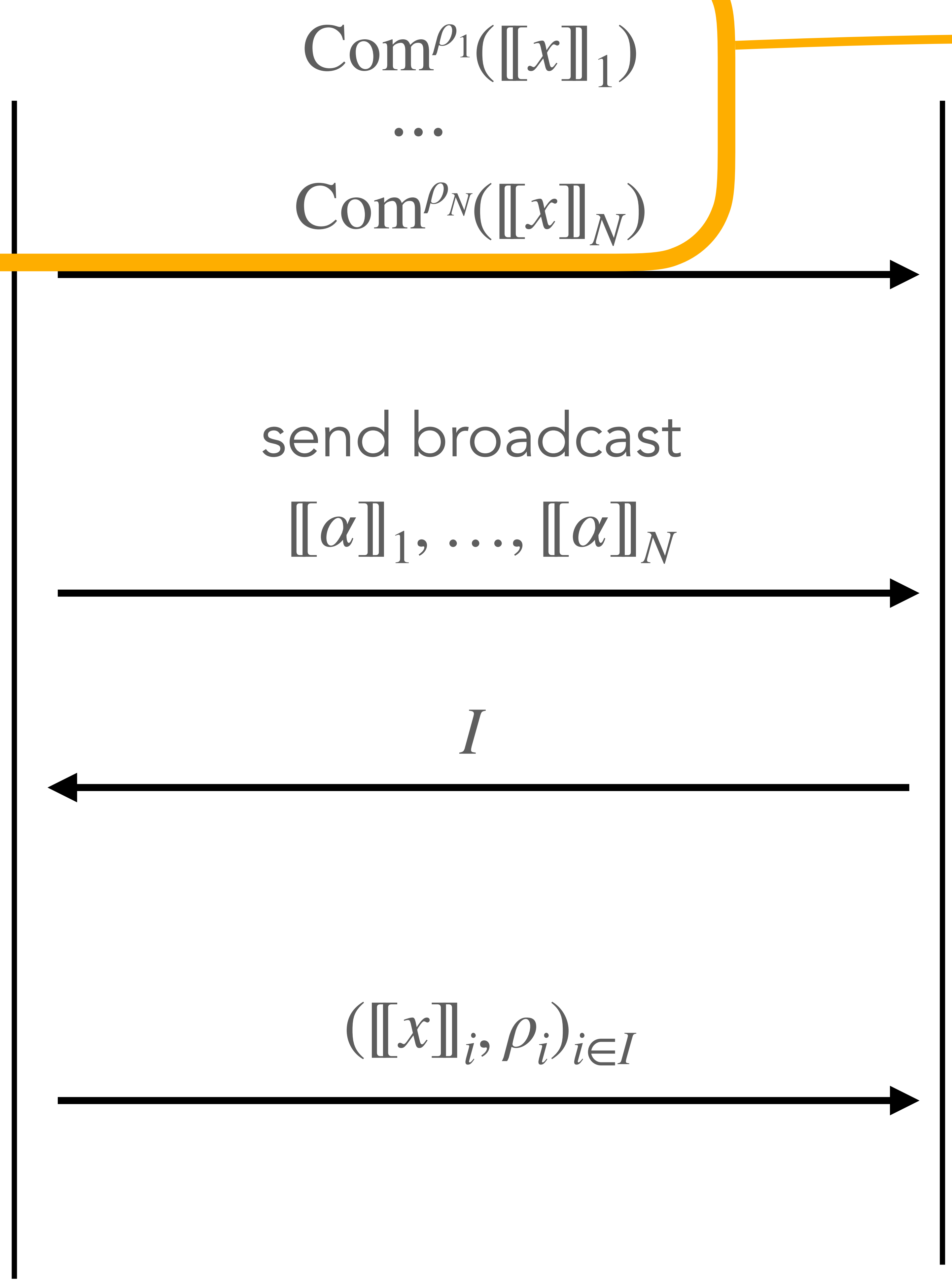
$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

Threshold LSSS ⇒ cannot generate shares from seeds

② Run MPC in their head



④ Open parties in I



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

$\cdots$

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

send broadcast

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

I

$([\![x]\!]_i, \rho_i)_{i \in I}$

③ Chose random set of parties

$I \subseteq \{1, \ldots, N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

Check $g(y, \alpha) =$ Accept

Prover

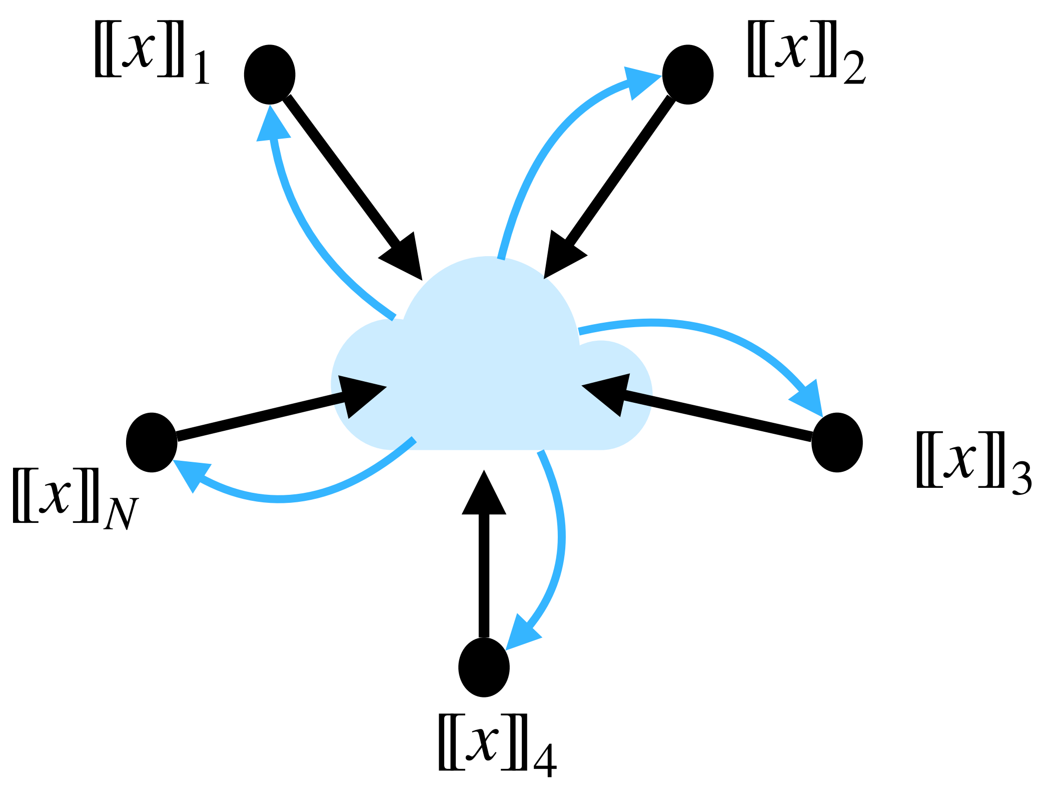
Verifier

MPCitH transform with threshold LSSS

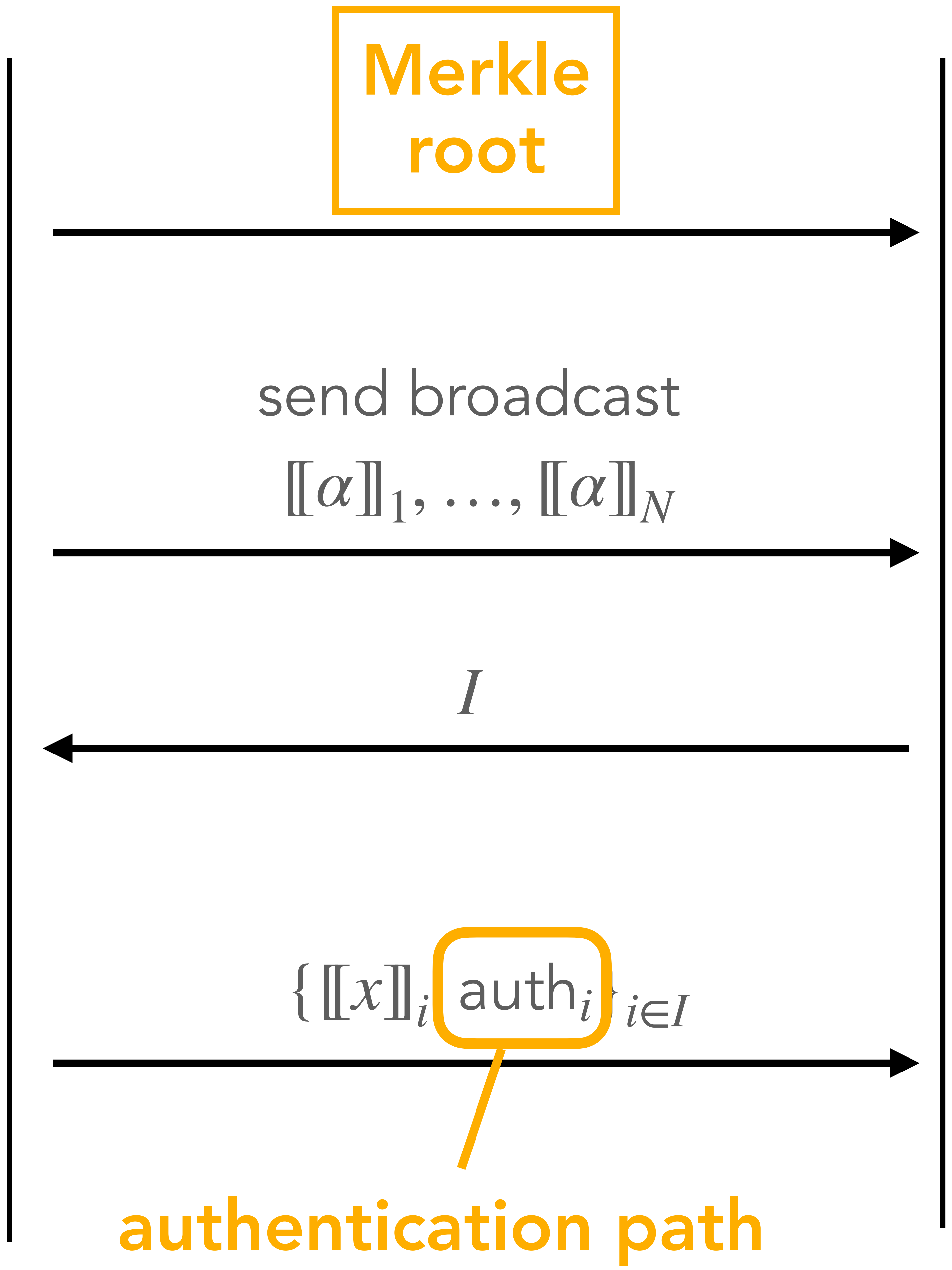
① Generate and commit shares

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

② Run MPC in their head



④ Open parties in I



Merkle root

send broadcast

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

I

$\{[\![x]\!]_i, \boxed{\text{auth}_i}\}_{i \in I}$

authentication path

Prover

③ Chose random set of parties

$I \subseteq \{1,\dots,N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

Check $g(y, \alpha) =$ Accept

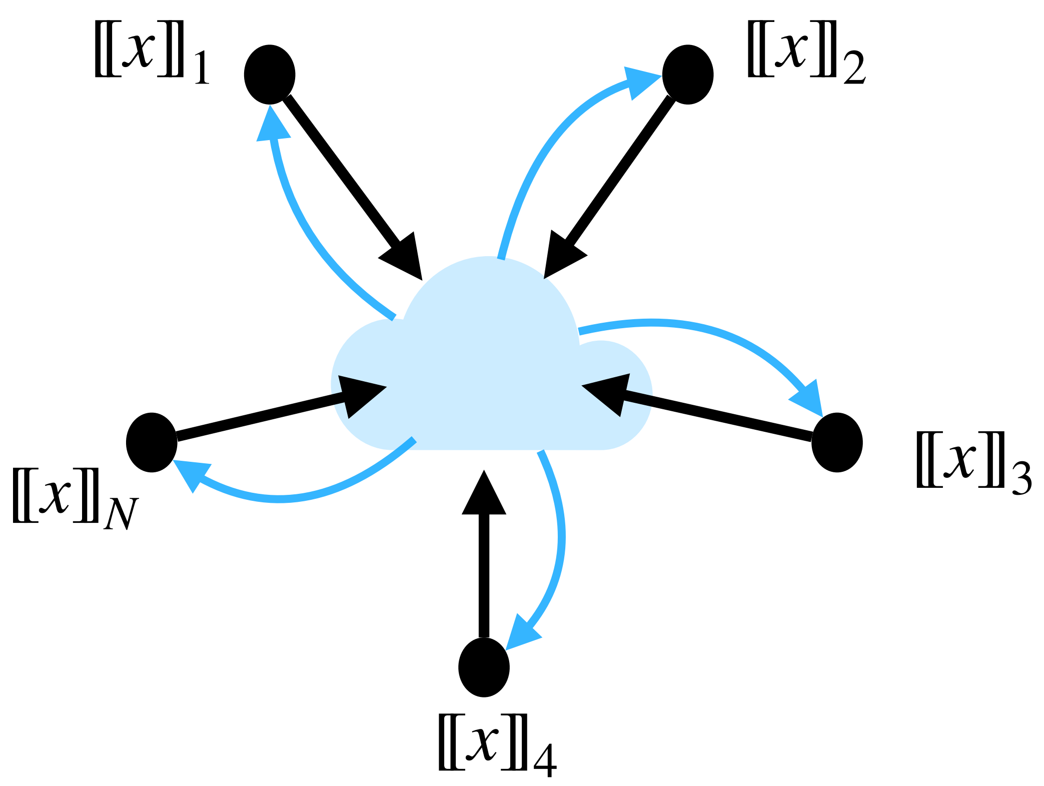
Verifier

MPCitH transform with threshold LSSS

① Generate and commit shares

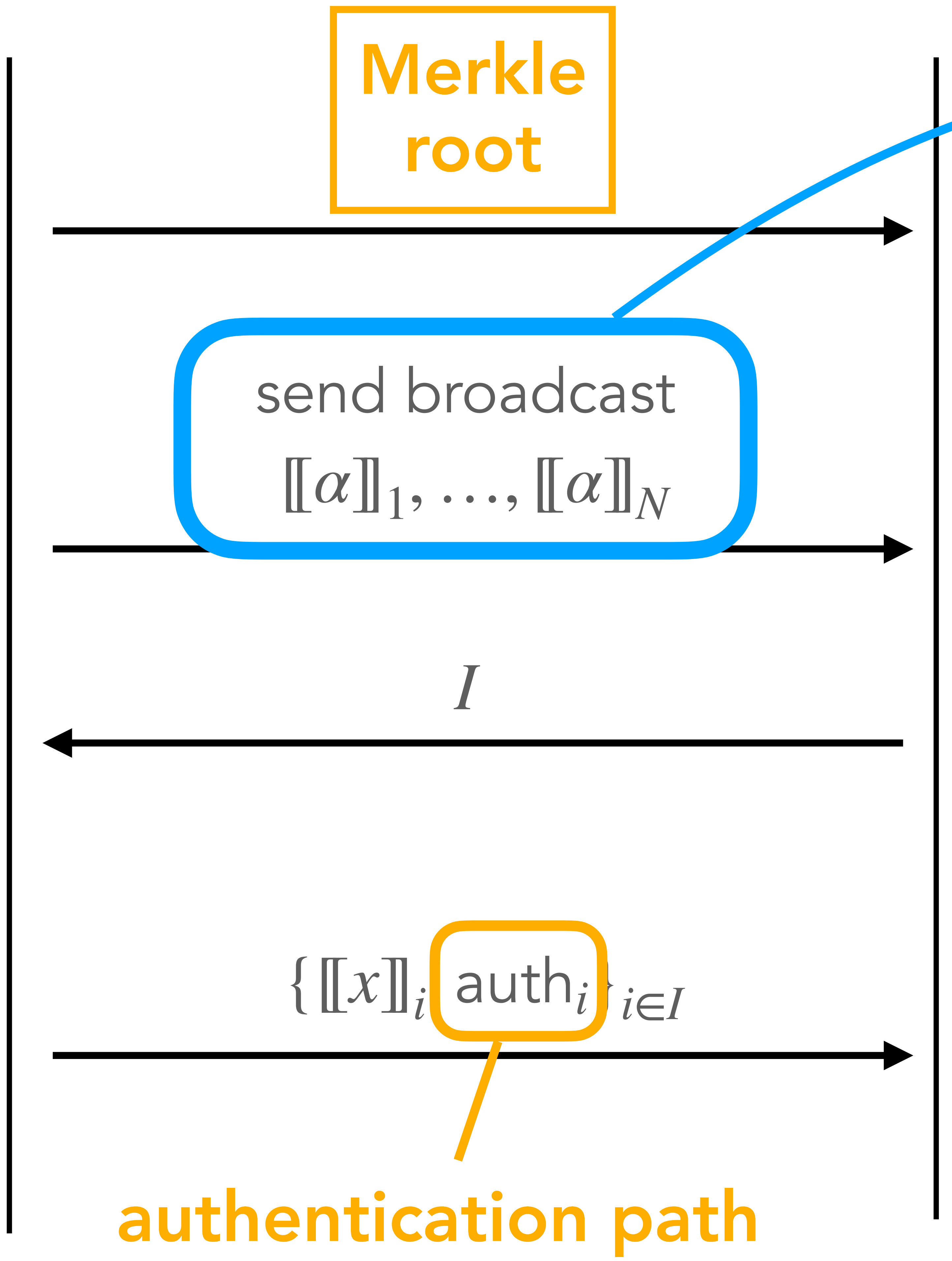
$[\![x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_N)$

② Run MPC in their head



④ Open parties in I

Prover



Merkle root

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

I

$\{[\![x]\!]_i, \text{auth}_i\}_{i\in I}$

authentication path

$[\![\alpha]\!]$ is redundant

⇒ $\ell + 1$ shares fully determine the sharing

③ Chose random set of parties

$I \subseteq \{1,\ldots,N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

Check $g(y, \alpha)$ = Accept

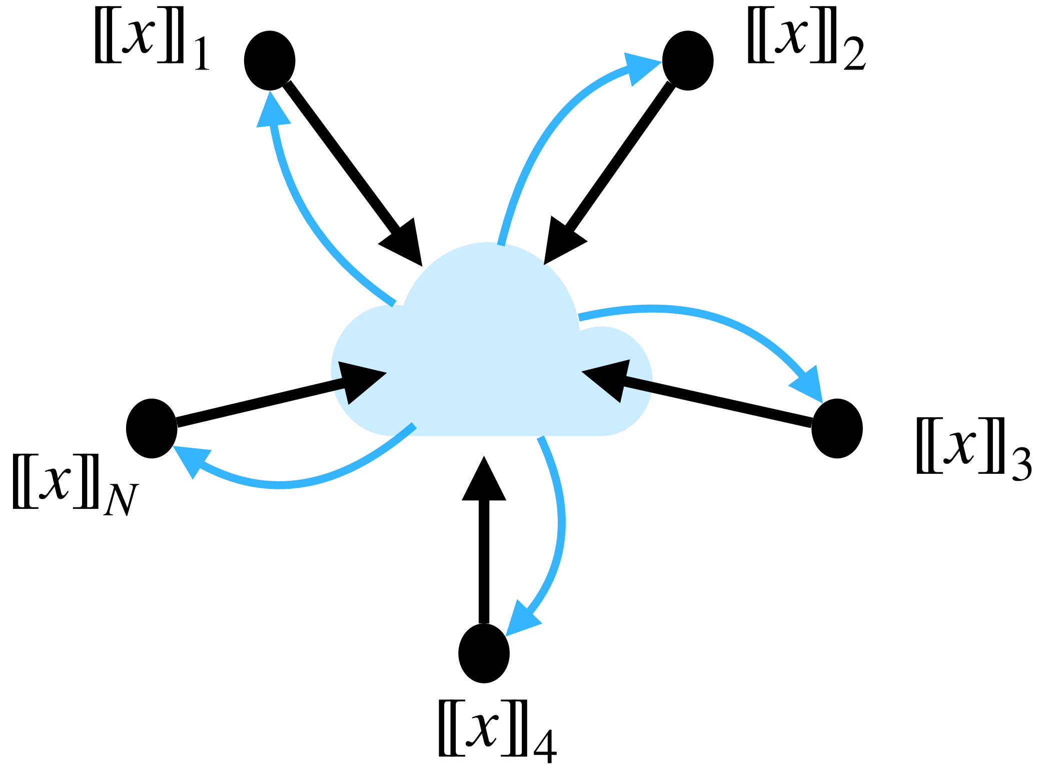
Verifier

MPCitH transform with threshold LSSS

① Generate and commit shares

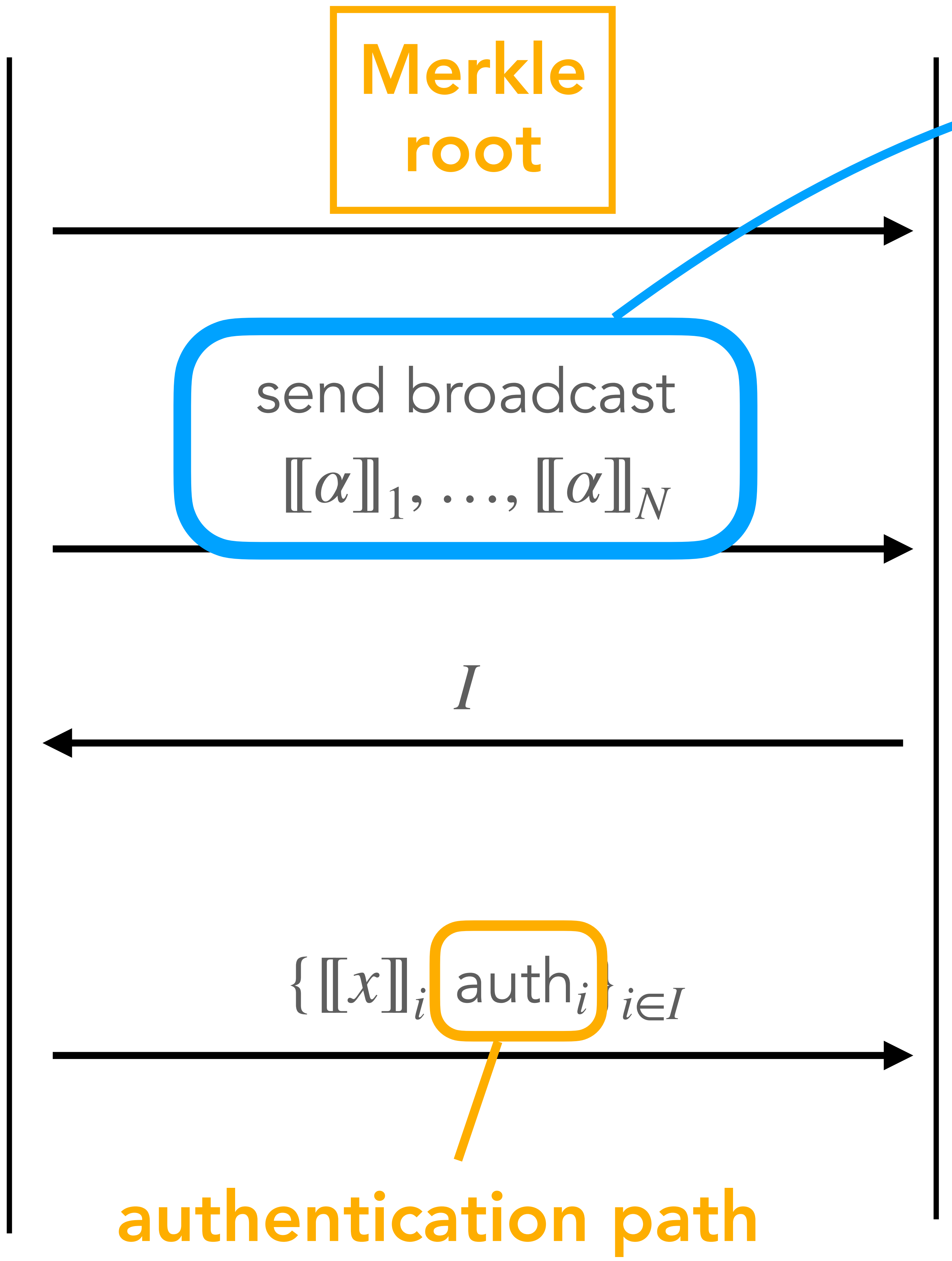
$[\![x]\!] = ([\![x]\!]_1, \dots, [\![x]\!]_N)$

② Run MPC in their head



④ Open parties in I

<u>Prover</u>



Merkle root

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

I

$\{[\![x]\!]_i, \text{auth}_i\}_{i\in I}$

authentication path

$[\![\alpha]\!]$ is redundant
$\Rightarrow \ell + 1$ shares fully determine the sharing
*$\Rightarrow$ **only $\ell + 1$ party computations required***

③ Chose random set of parties $I \subseteq \{1,\dots,N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

Check $g(y, \alpha) =$ Accept

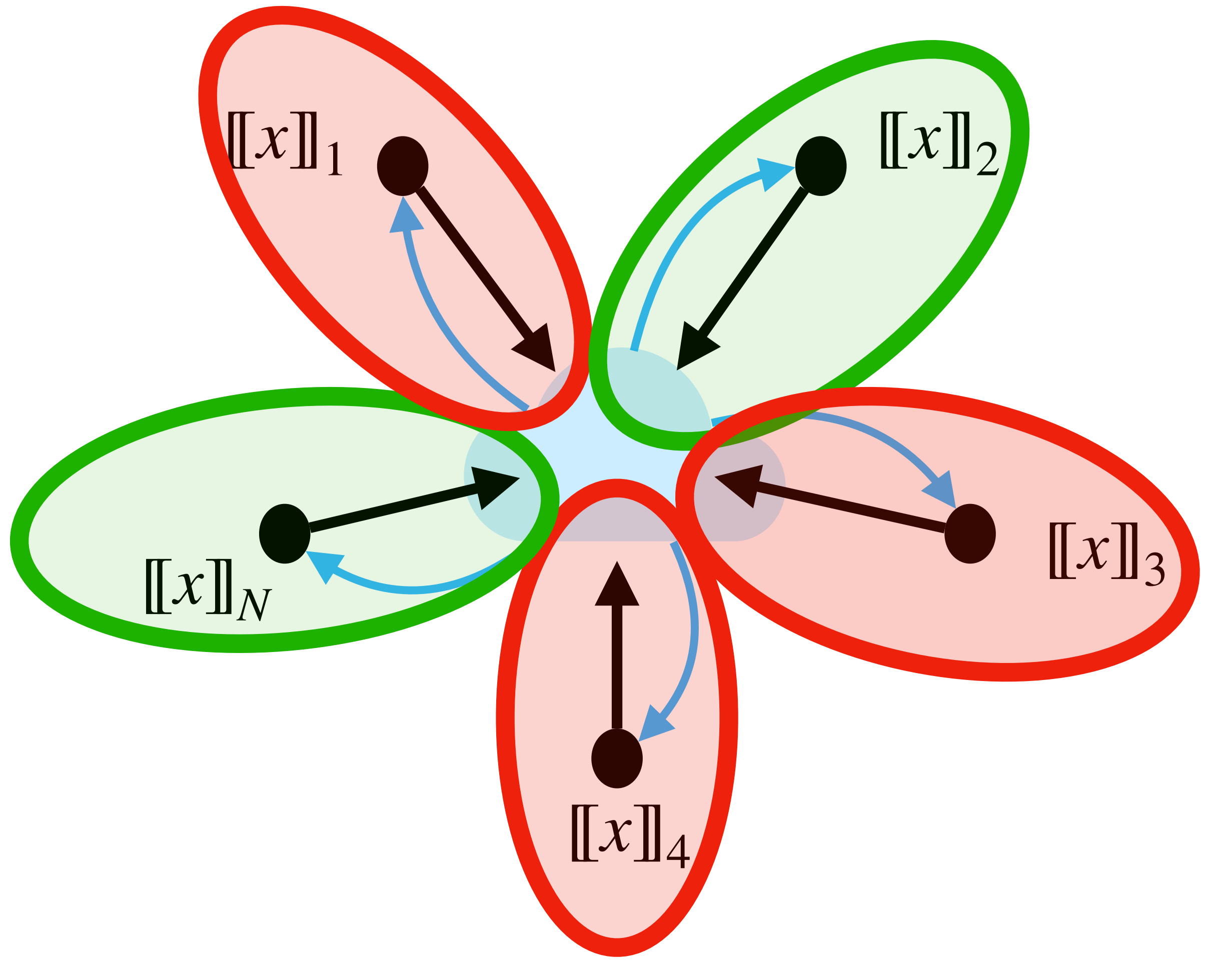
<u>Verifier</u>

MPCitH transform with threshold LSSS

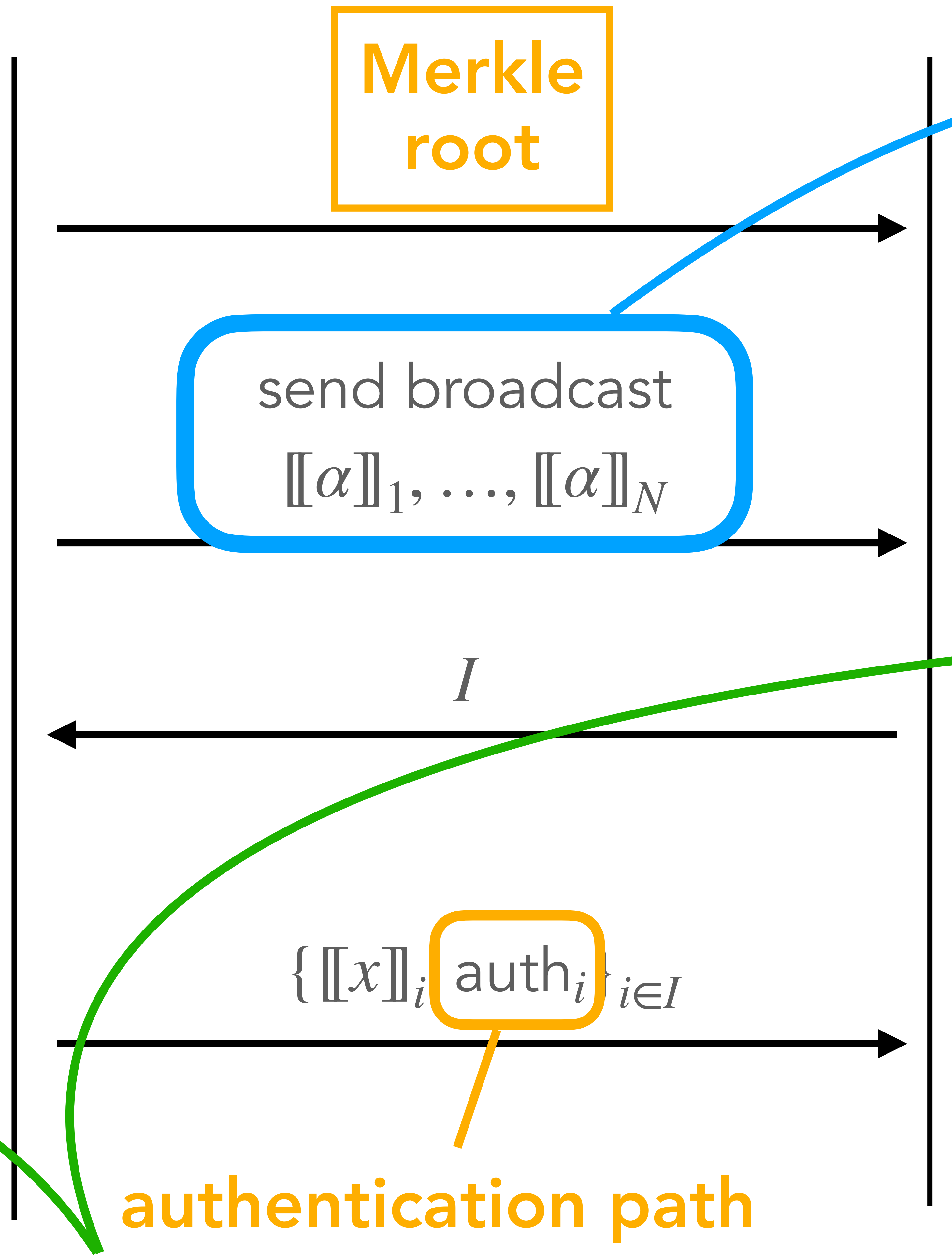
① Generate and commit shares

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

② Run MPC in their head



④ Open parties in I



Merkle root

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

I

$\{[\![x]\!]_i, \text{auth}_i\}_{i \in I}$

authentication path

$[\![\alpha]\!]$ is redundant

$\Rightarrow \ell + 1$ shares fully determine the sharing

$\Rightarrow$ only $\ell + 1$ party computations required

③ Chose random set of parties $I \subseteq \{1,\ldots,N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

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

Prover

ℓ parties opened instead of $N - 1$

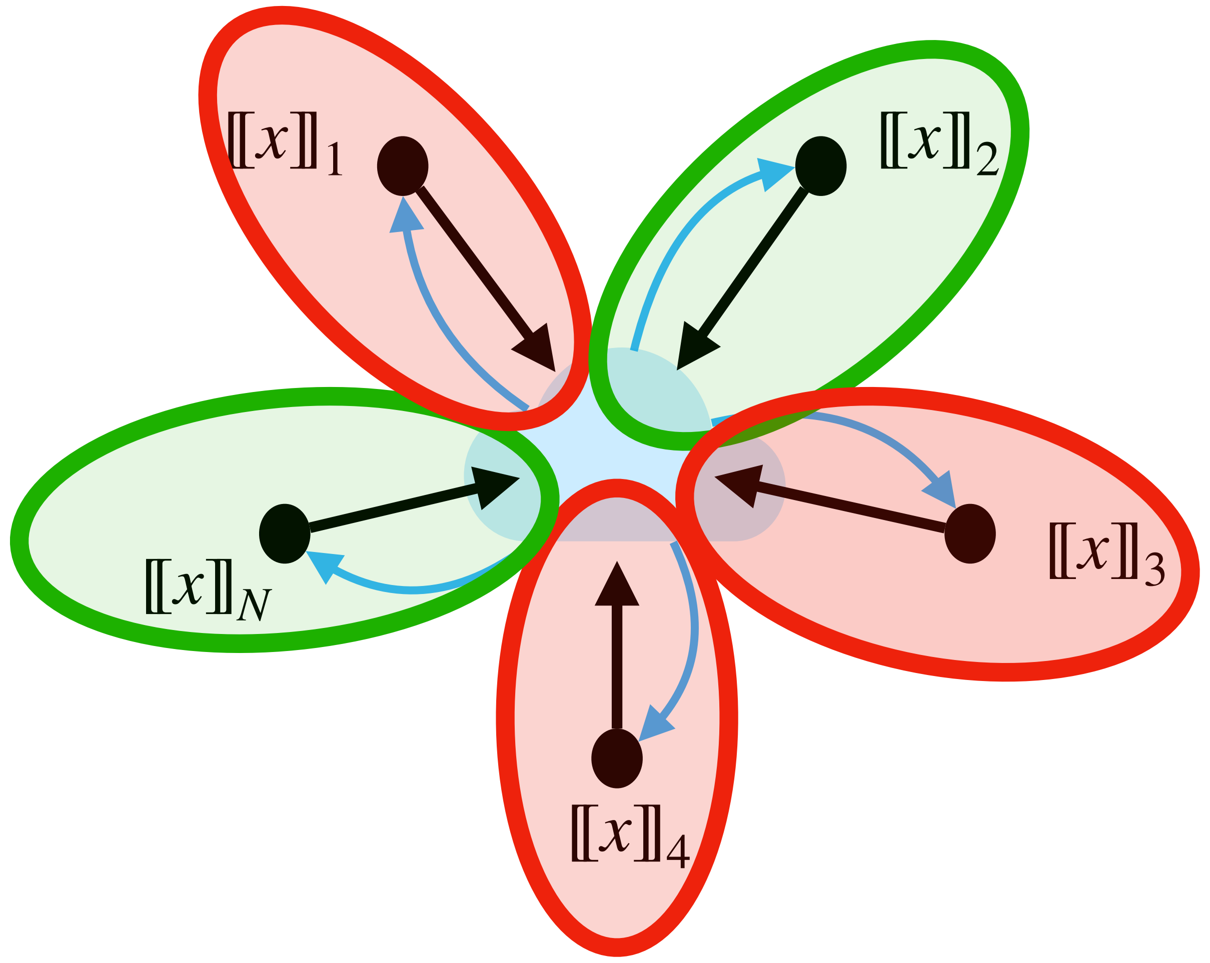
Verifier

MPCitH transform with threshold LSSS

① Generate and commit shares

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

② Run MPC in their head



④ Open parties in I

Prover

Merkle root

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

I

$\{[\![x]\!]_i \text{ auth}_i\}_{i \in I}$

authentication path

ℓ parties opened instead of $N-1$

$[\![\alpha]\!]$ is redundant

$\Rightarrow \ell + 1$ shares fully determine the sharing

$\Rightarrow$ only $\ell + 1$ party computations required

③ Chose random set of parties $I \subseteq \{1, \ldots, N\}$, s.t. $|I| = \ell$

⑤ Check $\forall i \in I$

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

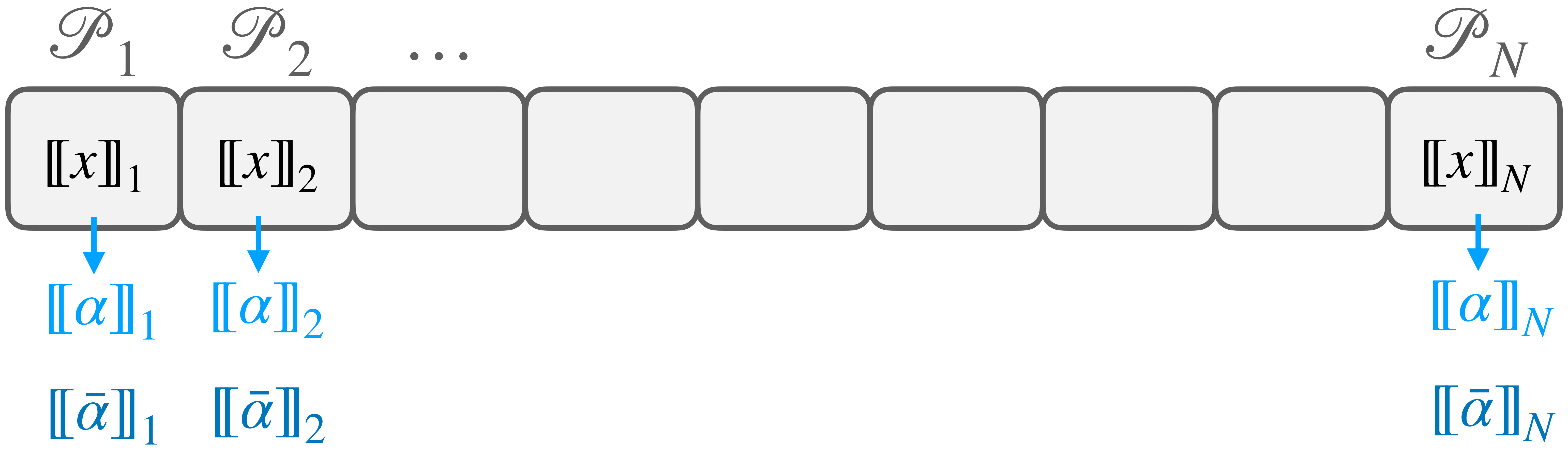
Check $g(y, \alpha) =$ Accept

only ℓ party computations required

Comparison

	Additive sharing + seed trees + hypercube	**Threshold LSSS** with $\ell = 1$
Soundness error	$\frac{1}{N} + p\left(1 - \frac{1}{N}\right)$	$\frac{1}{N} + p\left(\frac{N-1}{2}\right)$
Prover # party computations	$\log N + 1$	2
Verifier # party computations	$\log N$	1
Size (in bits) of seed tree / Merkle tree	$\lambda(\log N)$	$2\lambda(\log N)$

Soundness



$\mathcal{P}_1$ $\mathcal{P}_2$ $\dots$ $\mathcal{P}_N$

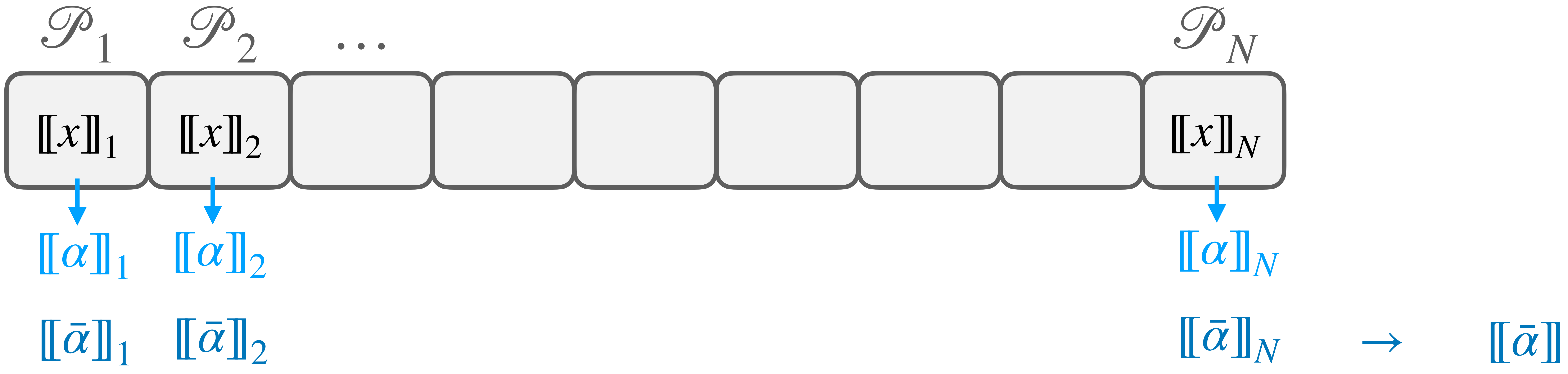
$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_N$

$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t.
$g(y, \bar{\alpha}) = \text{Accept}$

Soundness



$\mathscr{P}_1$ $\mathscr{P}_2$ $\dots$ $\mathscr{P}_N$

$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_N$

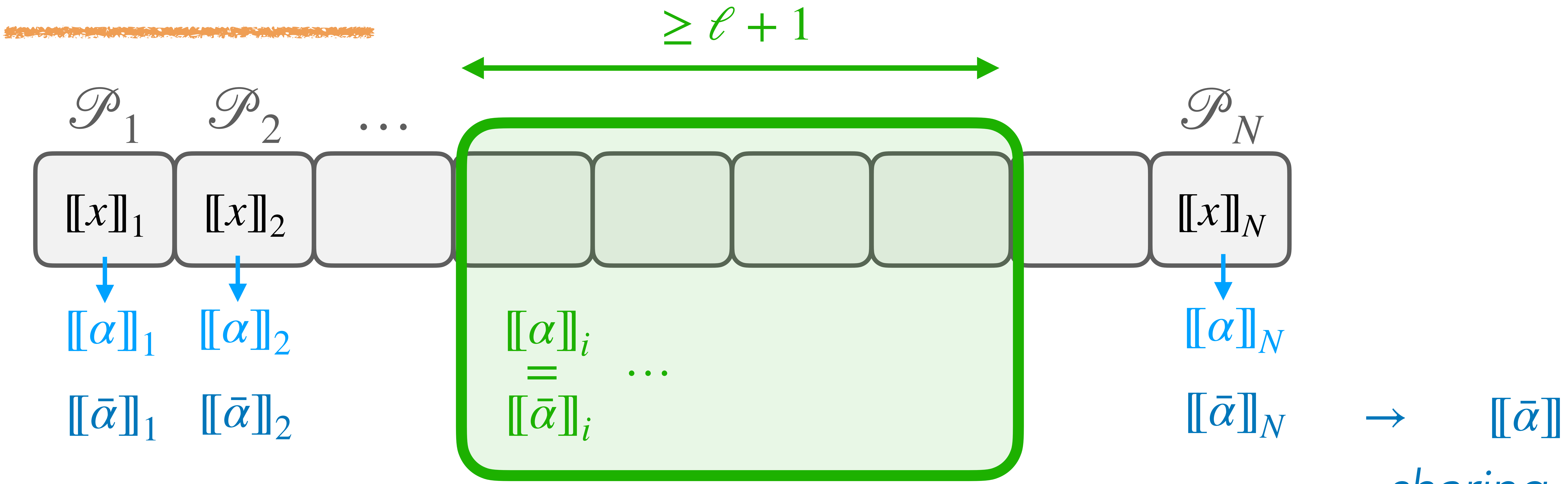
$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t.

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

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$

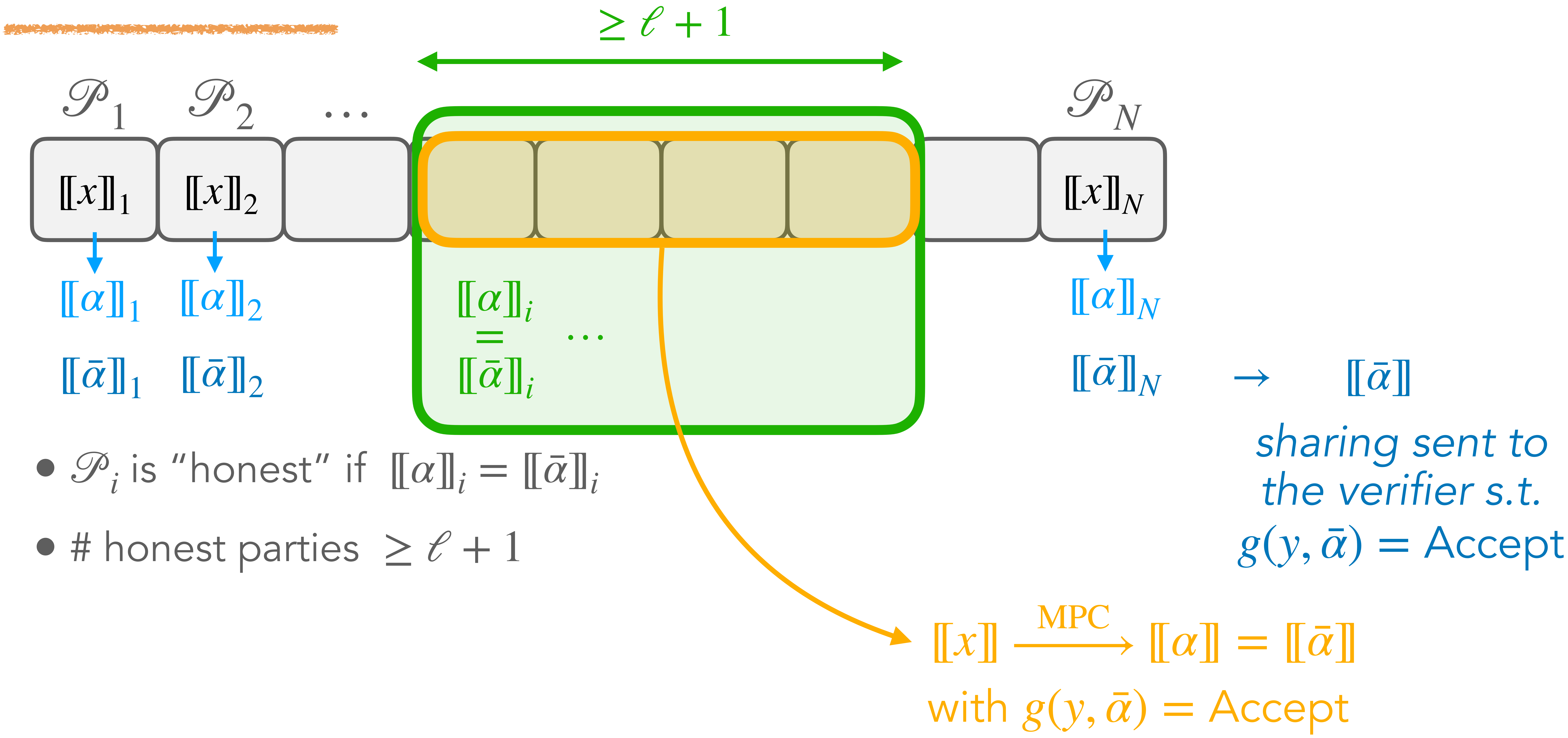
Soundness



sharing sent to
the verifier s.t.
$g(y, \bar{\alpha})$ = Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$

Soundness



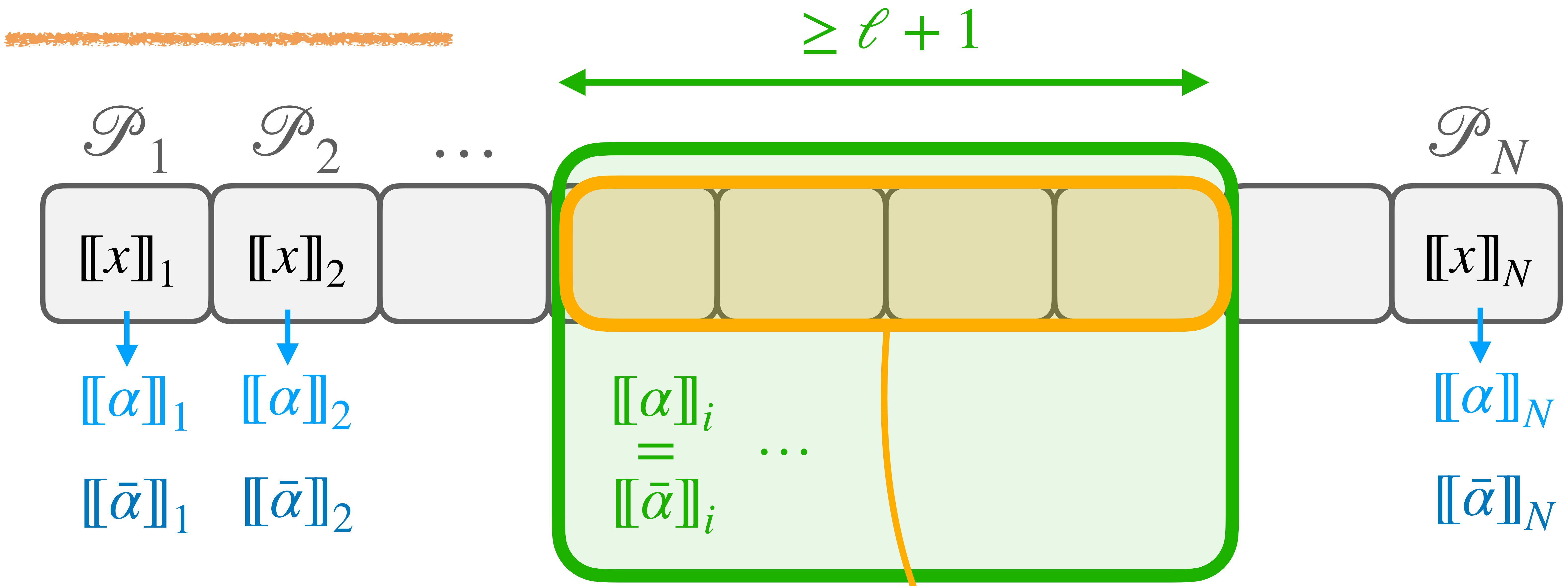
sharing sent to the verifier s.t. $g(y, \bar{\alpha})$ = Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$

$[\![x]\!] \xrightarrow{\text{MPC}} [\![\alpha]\!] = [\![\bar{\alpha}]\!]$

with $g(y, \bar{\alpha})$ = Accept

Soundness



$\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t. $g(y, \bar{\alpha})$ = Accept

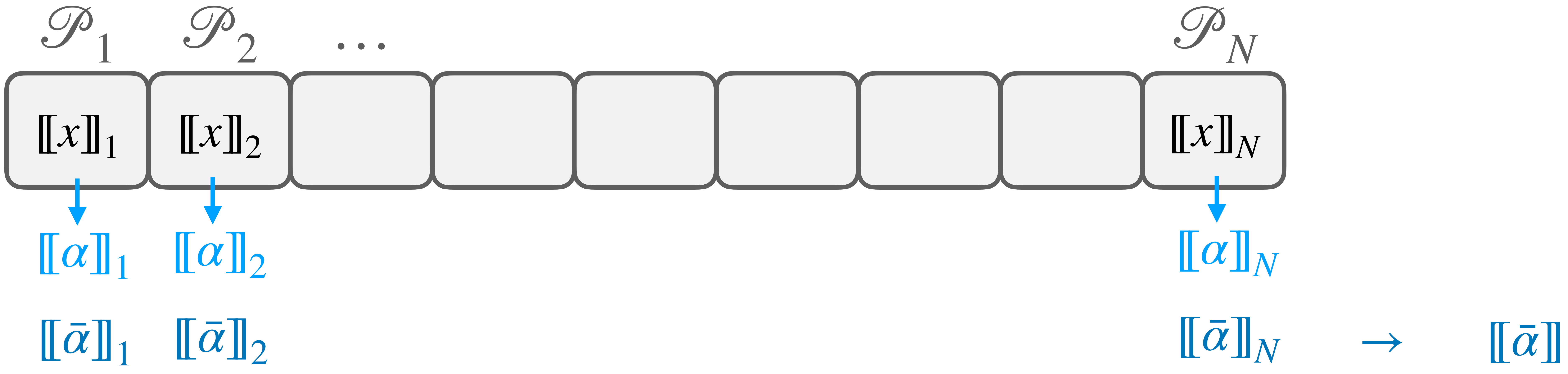
- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$

$[\![x]\!] \xrightarrow{\text{MPC}} [\![\alpha]\!] = [\![\bar{\alpha}]\!]$

with $g(y, \bar{\alpha})$ = Accept

$\Leftrightarrow$ $[\![x]\!]$ encodes a genuine x

Soundness



$\mathscr{P}_1 \quad \mathscr{P}_2 \quad \cdots \quad \mathscr{P}_N$

$[\![x]\!]_1 \quad [\![x]\!]_2 \quad [\![x]\!]_N$

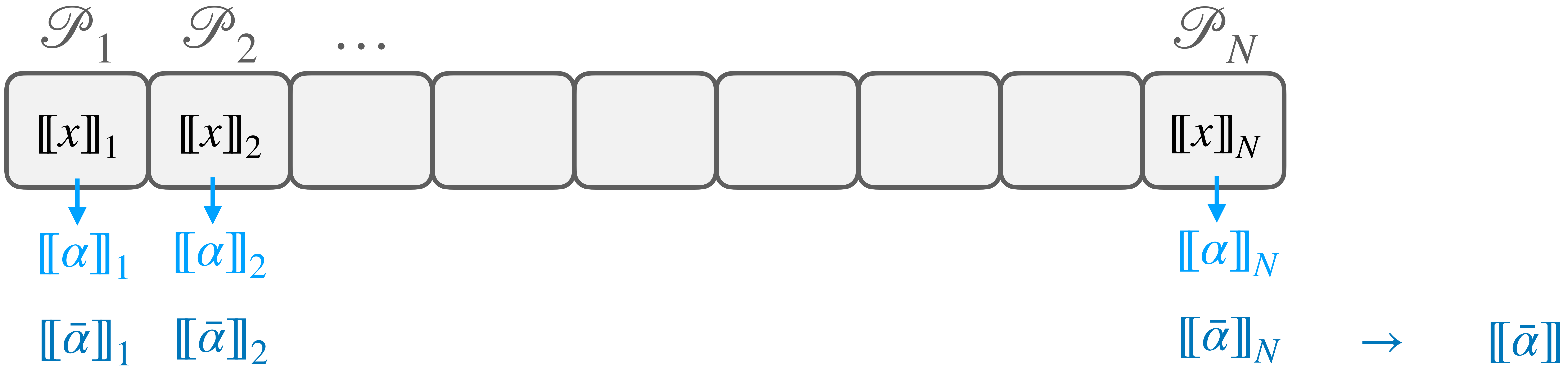
$[\![\alpha]\!]_1 \quad [\![\alpha]\!]_2 \quad [\![\alpha]\!]_N$

$[\![\bar{\alpha}]\!]_1 \quad [\![\bar{\alpha}]\!]_2 \quad [\![\bar{\alpha}]\!]_N \quad \rightarrow \quad [\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t. $g(y, \bar{\alpha}) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover

Soundness



$\mathscr{P}_1$ $\mathscr{P}_2$ $\dots$ $\mathscr{P}_N$

$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

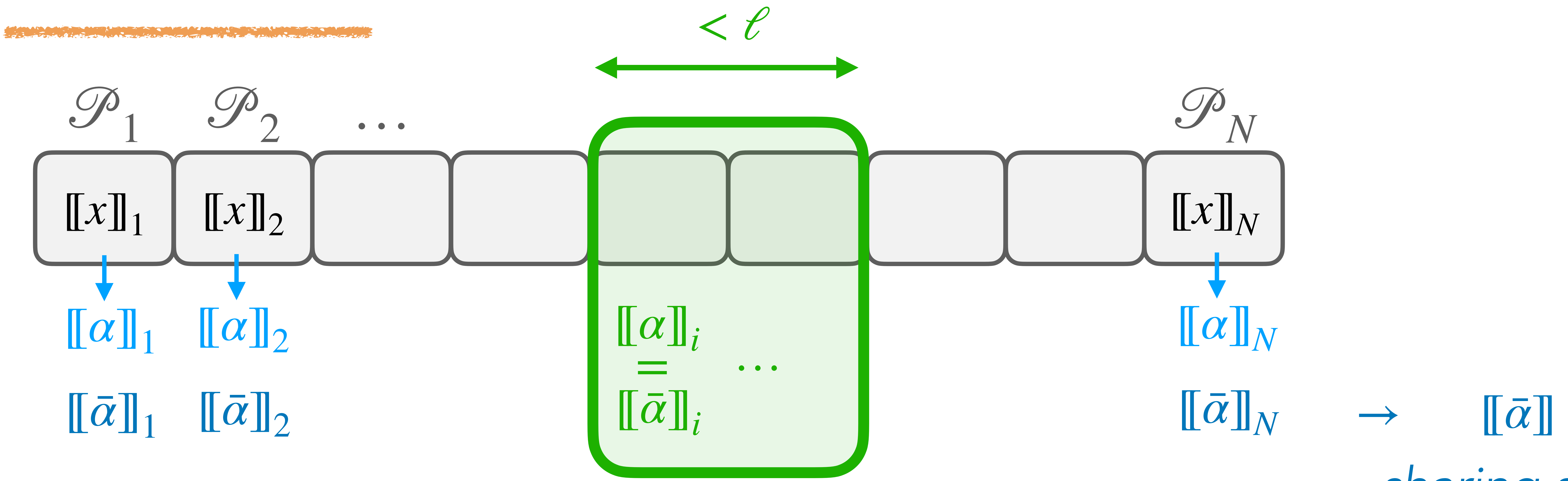
$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_N$

$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t. $g(y, \bar{\alpha}) = \text{Accept}$

- $\mathscr{P}_i$ is “honest” if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$

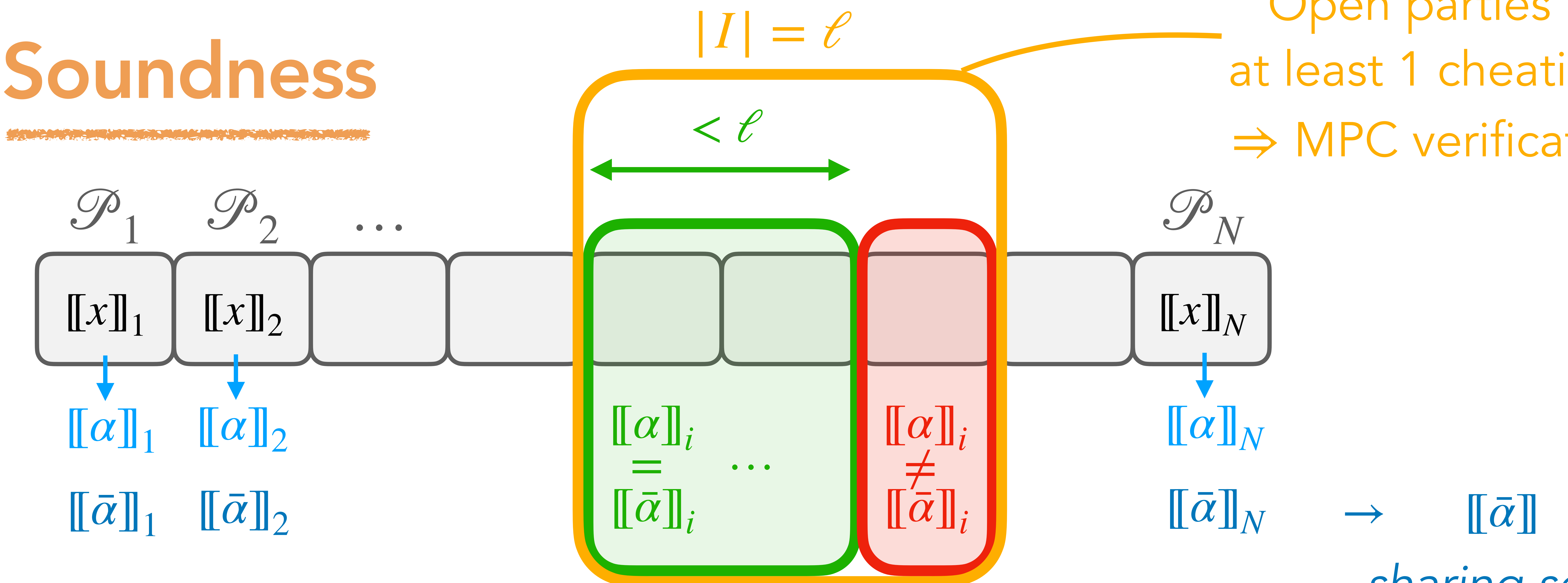
Soundness



sharing sent to the verifier s.t.
$g(y, \bar{\alpha})$ = Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - # honest parties $< \ell$

Soundness

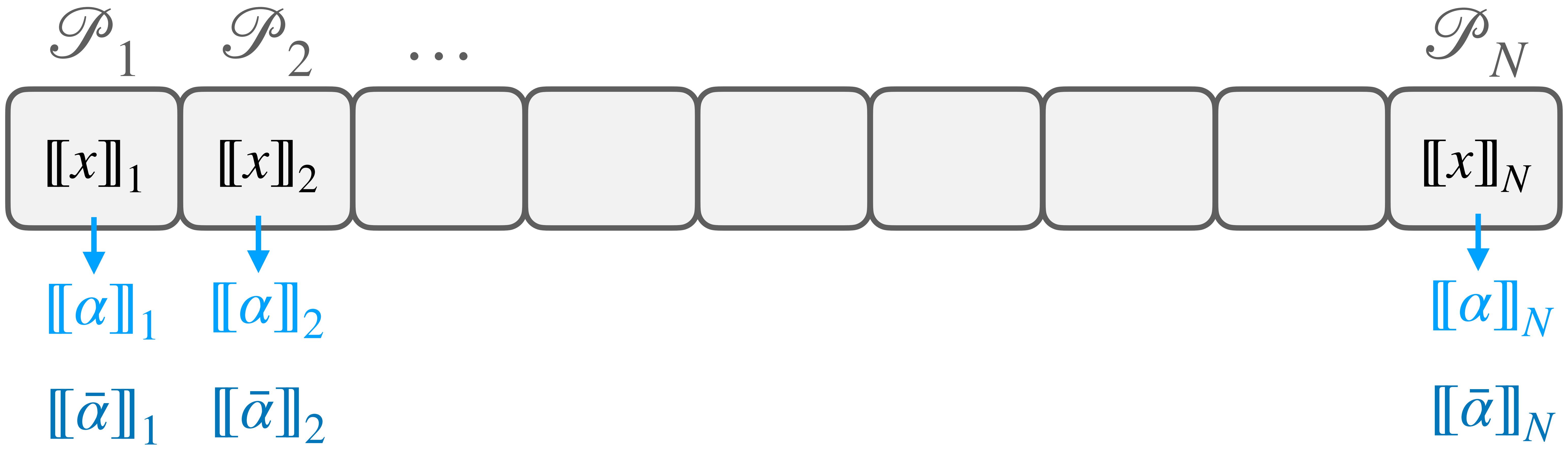


Open parties include
at least 1 cheating party
⇒ MPC verification fails

sharing sent to
the verifier s.t.
$g(y, \bar\alpha) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar\alpha]\!]_i$
- # honest parties $\geq \ell + 1$ ⇒ honest prover
- Malicious prover ⇒ # honest parties $\leq \ell$
  - # honest parties $< \ell$

Soundness



$\mathscr{P}_1$ $\mathscr{P}_2$ $\ldots$ $\mathscr{P}_N$

$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

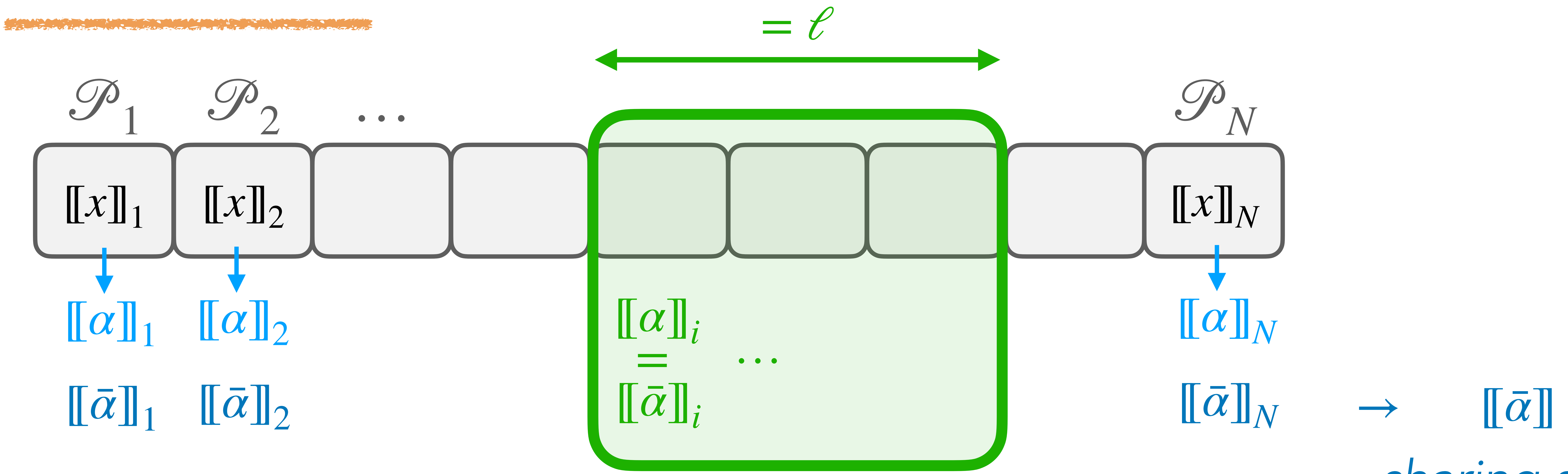
$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_N$

$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t. $g(y, \bar{\alpha}) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - # honest parties $< \ell$ $\Rightarrow$ cheat always detected

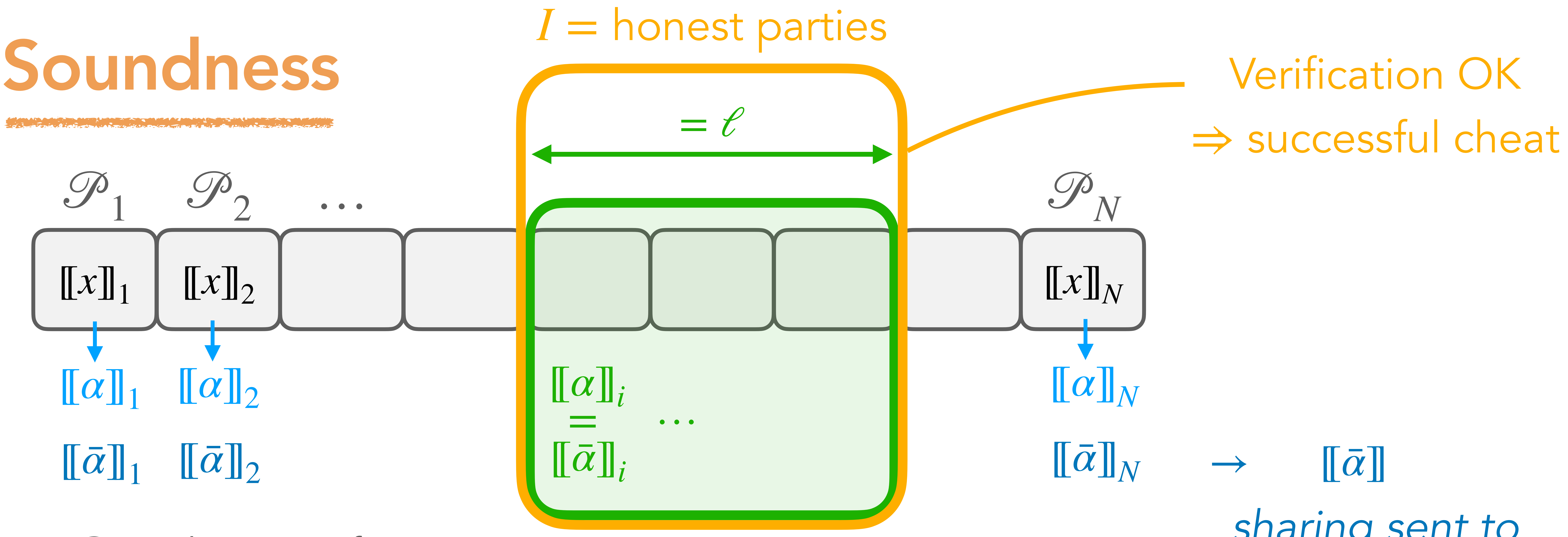
Soundness



sharing sent to the verifier s.t. $g(y, \bar{\alpha}) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - # honest parties $< \ell$ $\Rightarrow$ cheat always detected
  - # honest parties $= \ell$

Soundness



I = honest parties

$= \ell$

Verification OK

$\Rightarrow$ successful cheat

$\mathscr{P}_1$ $\mathscr{P}_2$ $\cdots$ $\mathscr{P}_N$

$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_i$ $\cdots$ $[\![\alpha]\!]_N$

$=$

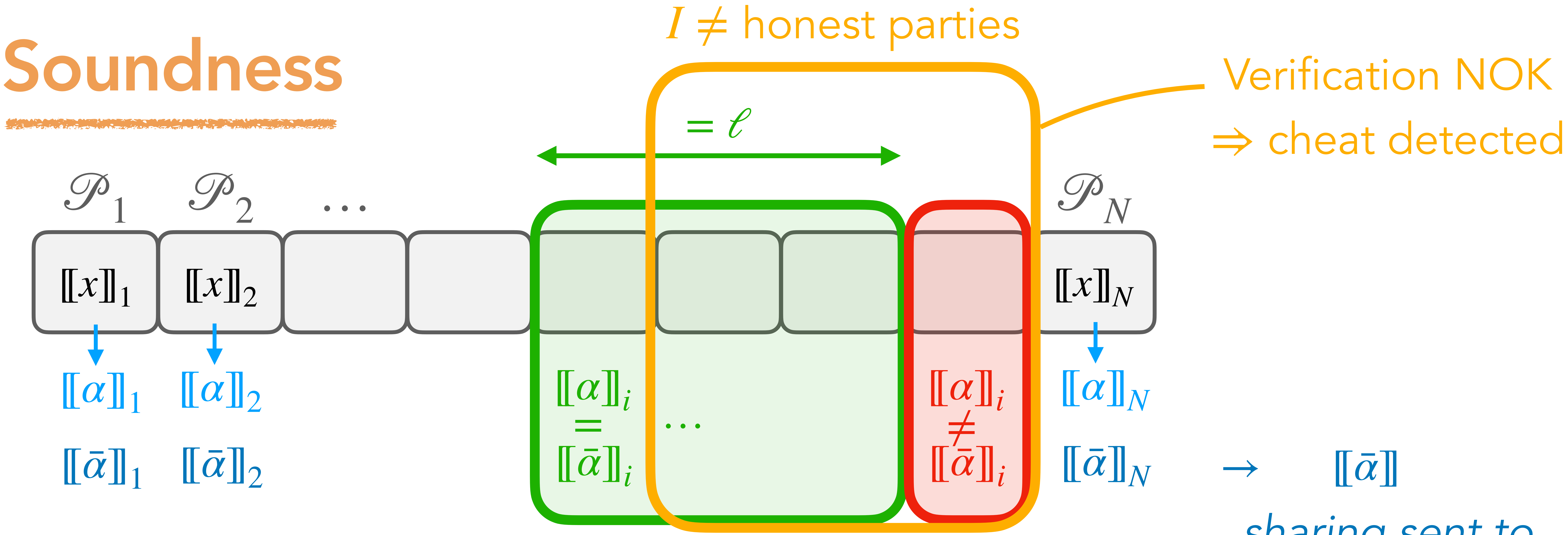
$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_i$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t.

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

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - # honest parties $< \ell$ $\Rightarrow$ cheat always detected
  - # honest parties $= \ell$

Soundness

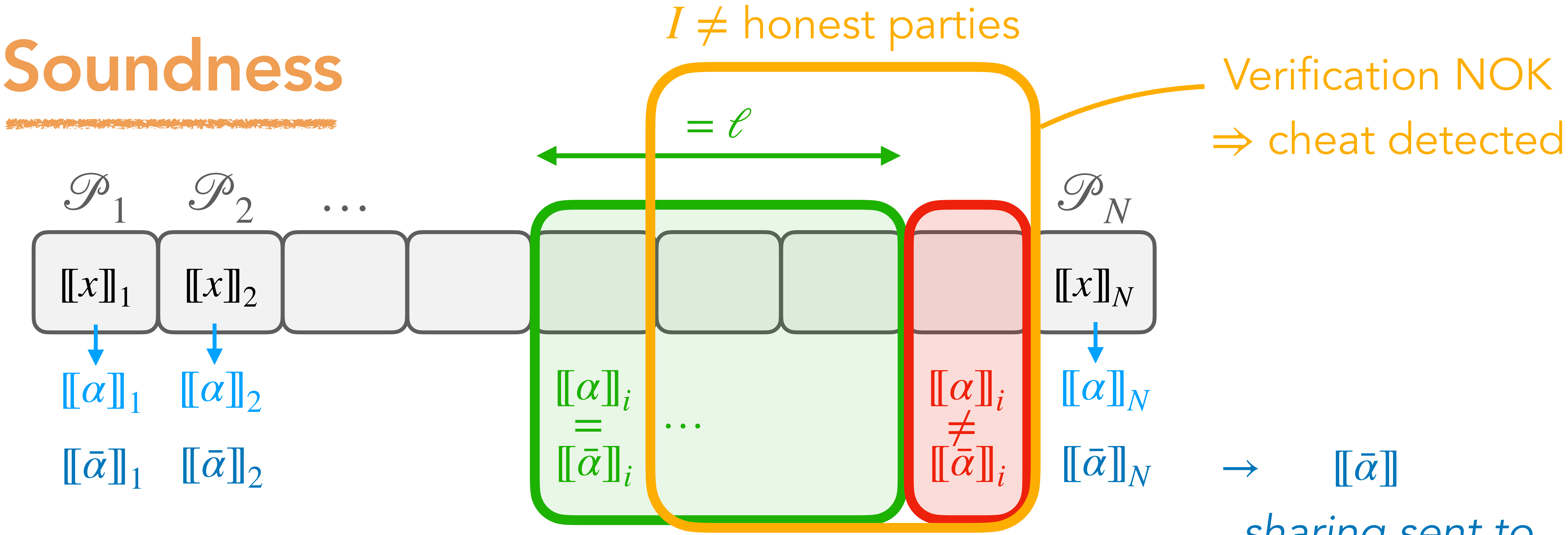


Verification NOK
$\Rightarrow$ cheat detected

sharing sent to the verifier s.t.
$g(y, \bar{\alpha}) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - # honest parties $< \ell$ $\Rightarrow$ cheat always detected
  - # honest parties $= \ell$

Soundness



$I \neq$ honest parties

Verification NOK

$\Rightarrow$ cheat detected

$= \ell$

$\mathscr{P}_1$ $\mathscr{P}_2$ $\dots$ $\mathscr{P}_N$

$[\![x]\!]_1$ $[\![x]\!]_2$ $[\![x]\!]_N$

$[\![\alpha]\!]_1$ $[\![\alpha]\!]_2$ $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$ $\dots$ $[\![\alpha]\!]_i \neq [\![\bar{\alpha}]\!]_i$ $[\![\alpha]\!]_N$

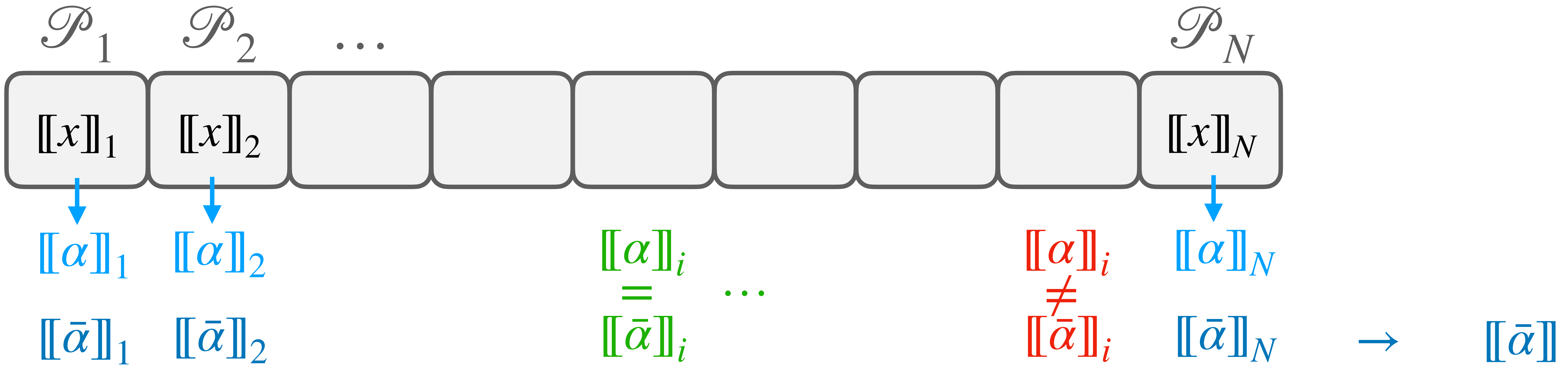
$[\![\bar{\alpha}]\!]_1$ $[\![\bar{\alpha}]\!]_2$ $[\![\bar{\alpha}]\!]_N$ $\rightarrow$ $[\![\bar{\alpha}]\!]$

sharing sent to the verifier s.t. $g(y, \bar{\alpha}) =$ Accept

- $\mathscr{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- \# honest parties $\geq \ell + 1$ $\Rightarrow$ honest prover
- Malicious prover $\Rightarrow$ # honest parties $\leq \ell$
  - \# honest parties $< \ell$ $\Rightarrow$ cheat always detected
  - \# honest parties $= \ell$

💡 Cheat successful iff $I =$ honest parties

Soundness



sharing sent to the verifier s.t. $g(y, \bar{\alpha}) =$ Accept

- $\mathcal{P}_i$ is "honest" if $[\![\alpha]\!]_i = [\![\bar{\alpha}]\!]_i$
- # honest parties $\geq \ell + 1 \Rightarrow$ honest prover
- Malicious prover ⇒ # honest parties $\leq \ell$
  - # honest parties $< \ell \Rightarrow$ cheat always detected
  - # honest parties $= \ell \Rightarrow$ soundness error $\dfrac{1}{\binom{N}{\ell}}$

💡 Cheat successful iff I = honest parties

Soundness

- False positive probability $p \neq 0$ → more complex analysis **[FR22]**
- Soundness error

$$\frac{1}{\binom{N}{\ell}} + p\,\frac{\ell(N-\ell)}{\ell+1}$$

- Fiat-Shamir transform: p should be small for efficient application

Roadmap

- Technical background
- MQOM MPC protocol
- SDitH MPC protocol
- Threshold MPCitH
- **MQOM signature scheme**
- SDitH signature scheme

MQOM: MQ on my Mind

Feneuil and Rivain

- New MPCitH-friendly MPC protocol for MQ
  - Batching of MQ equations **[Fen22]**
  - New inner product checking protocol inspired from Banquet and Limbo [**BDKOSZ21, DOT21]**
- Standard additive sharing MPCitH techniques
  - Seed trees **[KKW18]**
  - Hypercube technique **[AMGHHJY23]**

Choice of parameters

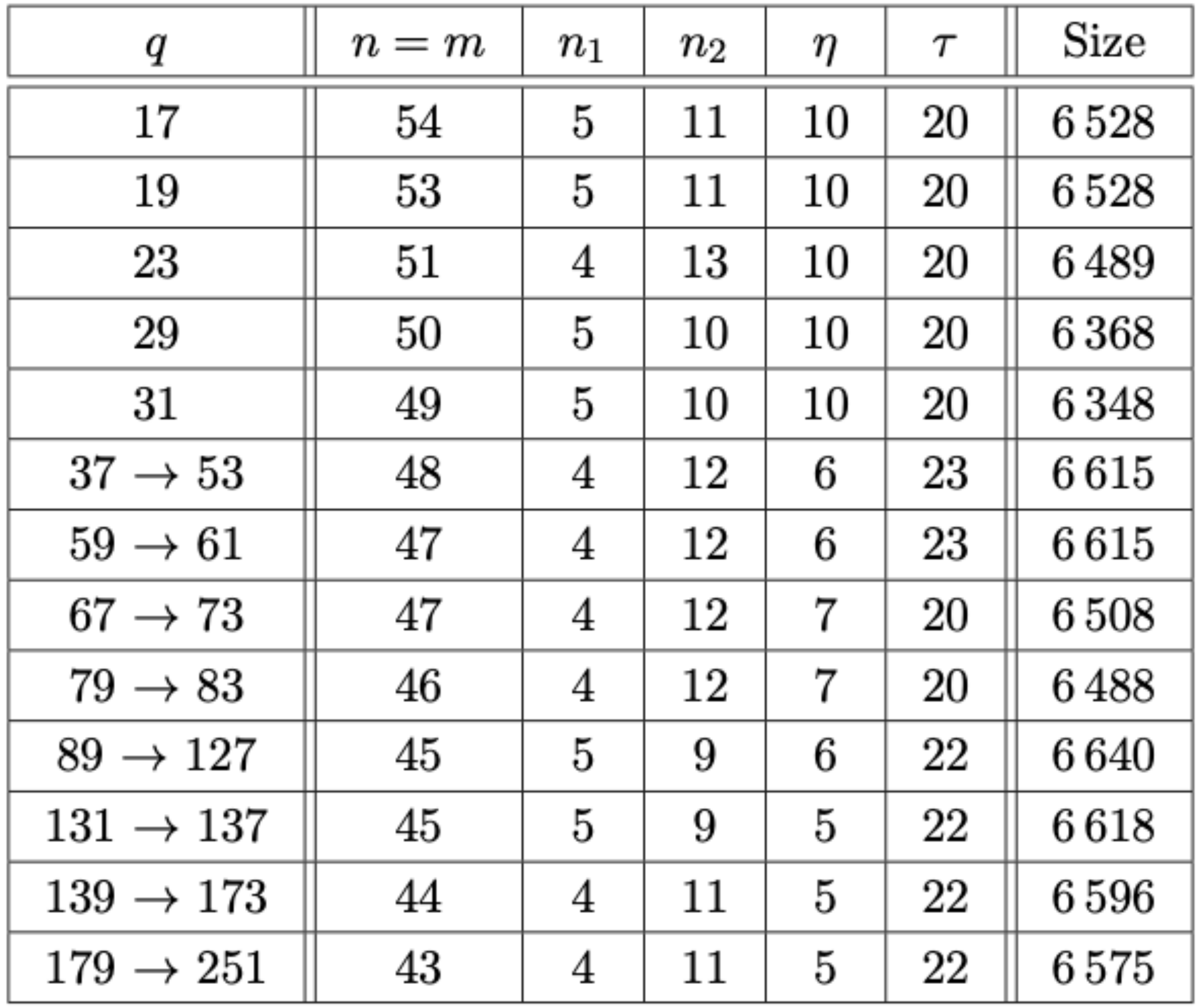
- MQ parameters:
  - Take $m = n$
  - Test several $q \rightarrow n$ for 3 security levels using MQ estimator [BMSV22]

Choice of parameters

- MQ parameters:
  - Take $m = n$
  - Test several $q \rightarrow n$ for 3 security levels using MQ estimator [BMSV22]
- MPC parameters:
  - Take $N = 256$ (good tradeoff)
  - Test several $(\eta, n_1, n_2) \rightarrow \tau$
    - τ = number of // executions to thwart generic forgery attack (à la [KZ20])

Choice of parameters

- MQ parameters:
  - Take $m = n$
  - Test several $q \rightarrow n$ for 3 security levels using MQ estimator [BMSV22]
- MPC parameters:
  - Take $N = 256$ (good tradeoff)
  - Test several $(\eta, n_1, n_2) \rightarrow \tau$
    - τ = number of // executions to thwart generic forgery attack (à la [KZ20])

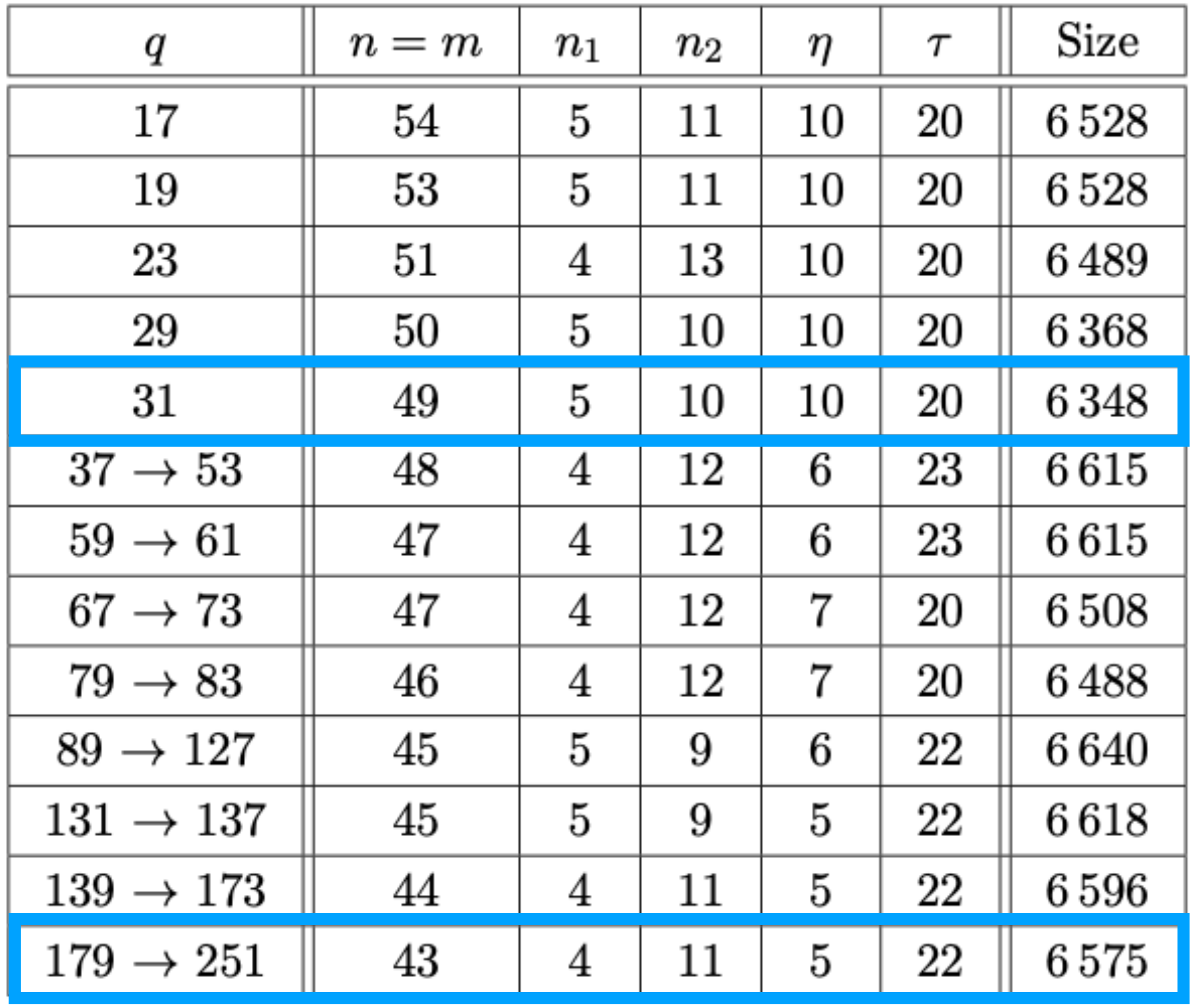


q	$n = m$	n_1	n_2	η	τ	Size
17	54	5	11	10	20	6 528
19	53	5	11	10	20	6 528
23	51	4	13	10	20	6 489
29	50	5	10	10	20	6 368
31	49	5	10	10	20	6 348
$37 \rightarrow 53$	48	4	12	6	23	6 615
$59 \rightarrow 61$	47	4	12	6	23	6 615
$67 \rightarrow 73$	47	4	12	7	20	6 508
$79 \rightarrow 83$	46	4	12	7	20	6 488
$89 \rightarrow 127$	45	5	9	6	22	6 640
$131 \rightarrow 137$	45	5	9	5	22	6 618
$139 \rightarrow 173$	44	4	11	5	22	6 596
$179 \rightarrow 251$	43	4	11	5	22	6 575

Best parameters (re. signature size) for different q for security level 1 (128-bit)

Choice of parameters

- MQ parameters:
  - Take $m = n$
  - Test several $q \rightarrow n$ for 3 security levels using MQ estimator [BMSV22]
- MPC parameters:
  - Take $N = 256$ (good tradeoff)
  - Test several $(\eta, n_1, n_2) \rightarrow \tau$
    - τ = number of // executions to thwart generic forgery attack (à la [KZ20])
- Two MQ instances:
  - $q = 31$: shortest signature / already considered in MQ-DSS
  - $q = 251$: larger field whose elements hold in bytes / more amenable to threshold MPCitH

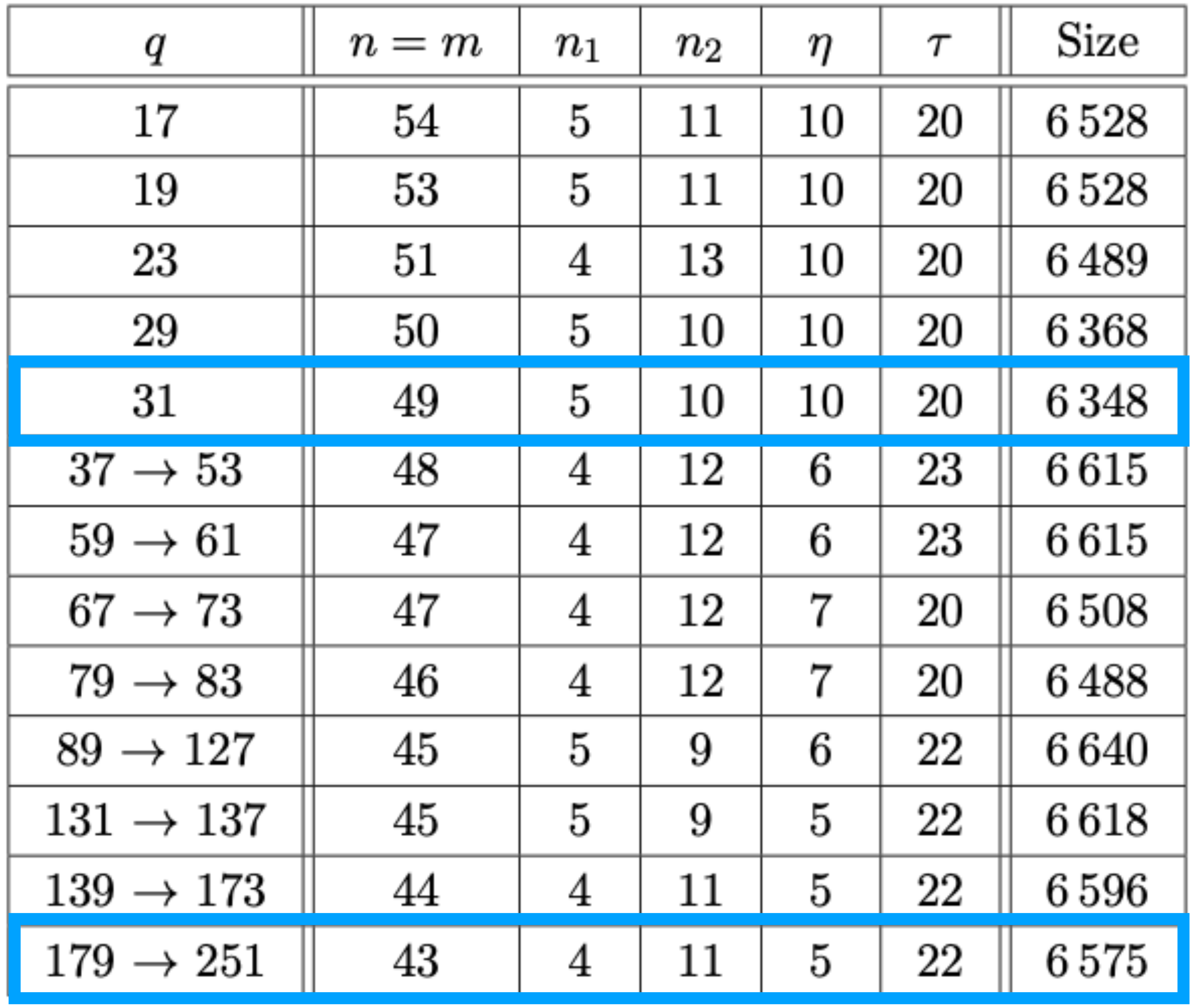


q	$n = m$	n_1	n_2	η	τ	Size
17	54	5	11	10	20	6 528
19	53	5	11	10	20	6 528
23	51	4	13	10	20	6 489
29	50	5	10	10	20	6 368
31	49	5	10	10	20	6 348
$37 \rightarrow 53$	48	4	12	6	23	6 615
$59 \rightarrow 61$	47	4	12	6	23	6 615
$67 \rightarrow 73$	47	4	12	7	20	6 508
$79 \rightarrow 83$	46	4	12	7	20	6 488
$89 \rightarrow 127$	45	5	9	6	22	6 640
$131 \rightarrow 137$	45	5	9	5	22	6 618
$139 \rightarrow 173$	44	4	11	5	22	6 596
$179 \rightarrow 251$	43	4	11	5	22	6 575

Best parameters (re. signature size) for different q for security level 1 (128-bit)

Choice of parameters

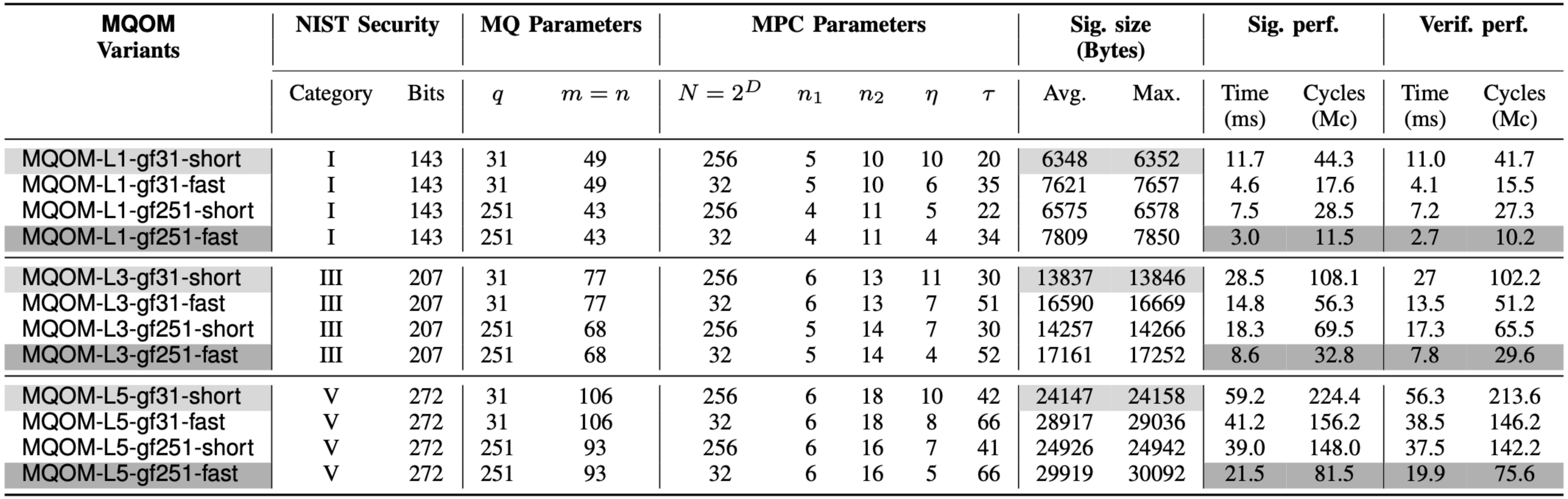
- MQ parameters:
  - Take $m = n$
  - Test several $q \rightarrow n$ for 3 security levels using MQ estimator [BMSV22]
- MPC parameters:
  - Take $N = 256$ (good tradeoff)
  - Test several $(\eta, n_1, n_2) \rightarrow \tau$
    - τ = number of // executions to thwart generic forgery attack (à la [KZ20])
- Two MQ instances:
  - $q = 31$: shortest signature / already considered in MQ-DSS
  - $q = 251$: larger field whose elements hold in bytes / more amenable to threshold MPCitH
- Fast variants with $N = 32$



q	$n = m$	n_1	n_2	η	τ	Size
17	54	5	11	10	20	6 528
19	53	5	11	10	20	6 528
23	51	4	13	10	20	6 489
29	50	5	10	10	20	6 368
31	49	5	10	10	20	6 348
$37 \rightarrow 53$	48	4	12	6	23	6 615
$59 \rightarrow 61$	47	4	12	6	23	6 615
$67 \rightarrow 73$	47	4	12	7	20	6 508
$79 \rightarrow 83$	46	4	12	7	20	6 488
$89 \rightarrow 127$	45	5	9	6	22	6 640
$131 \rightarrow 137$	45	5	9	5	22	6 618
$139 \rightarrow 173$	44	4	11	5	22	6 596
$179 \rightarrow 251$	43	4	11	5	22	6 575

Best parameters (re. signature size) for different q for security level 1 (128-bit)

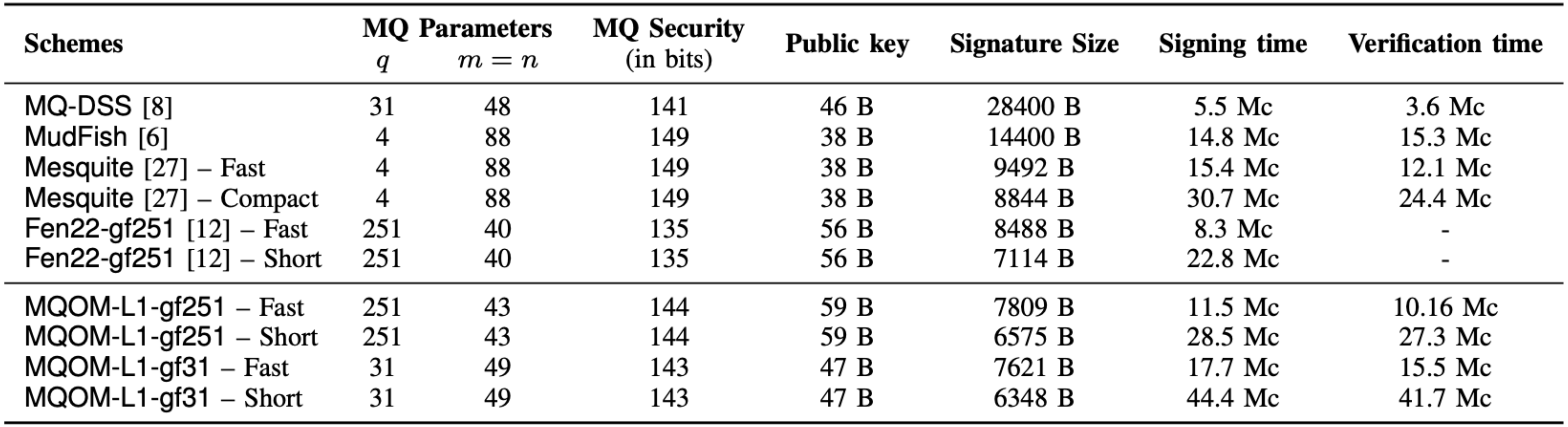
Performances



MQOM Variants	**NIST Security**		**MQ Parameters**		**MPC Parameters**					**Sig. size (Bytes)**		**Sig. perf.**		**Verif. perf.**	
	Category	Bits	q	$m = n$	$N = 2^D$	n_1	n_2	η	τ	Avg.	Max.	Time (ms)	Cycles (Mc)	Time (ms)	Cycles (Mc)
MQOM-L1-gf31-short	I	143	31	49	256	5	10	10	20	6348	6352	11.7	44.3	11.0	41.7
MQOM-L1-gf31-fast	I	143	31	49	32	5	10	6	35	7621	7657	4.6	17.6	4.1	15.5
MQOM-L1-gf251-short	I	143	251	43	256	4	11	5	22	6575	6578	7.5	28.5	7.2	27.3
MQOM-L1-gf251-fast	I	143	251	43	32	4	11	4	34	7809	7850	3.0	11.5	2.7	10.2
MQOM-L3-gf31-short	III	207	31	77	256	6	13	11	30	13837	13846	28.5	108.1	27	102.2
MQOM-L3-gf31-fast	III	207	31	77	32	6	13	7	51	16590	16669	14.8	56.3	13.5	51.2
MQOM-L3-gf251-short	III	207	251	68	256	5	14	7	30	14257	14266	18.3	69.5	17.3	65.5
MQOM-L3-gf251-fast	III	207	251	68	32	5	14	4	52	17161	17252	8.6	32.8	7.8	29.6
MQOM-L5-gf31-short	V	272	31	106	256	6	18	10	42	24147	24158	59.2	224.4	56.3	213.6
MQOM-L5-gf31-fast	V	272	31	106	32	6	18	8	66	28917	29036	41.2	156.2	38.5	146.2
MQOM-L5-gf251-short	V	272	251	93	256	6	16	7	41	24926	24942	39.0	148.0	37.5	142.2
MQOM-L5-gf251-fast	V	272	251	93	32	6	16	5	66	29919	30092	21.5	81.5	19.9	75.6

- Sig sizes:
  - Cat I (128-bit): 6.3 – 7.8 KB
  - Cat III (192-bit): 14 – 17 KB
  - Cat V (256-bit): 24 – 30 KB
- Key sizes:
  - Cat I (128-bit): |pk| , |sk| $\lesssim$ 100 B
  - Cat III (192-bit): |pk| , |sk| $\lesssim$ 160 B
  - Cat V (256-bit): |pk| , |sk| $\lesssim$ 220 B
- Timings: one to few dozen Mc (megacycles)

Comparison



Schemes	MQ Parameters		MQ Security	Public key	Signature Size	Signing time	Verification time
	q	$m = n$	(in bits)				
MQ-DSS [8]	31	48	141	46 B	28400 B	5.5 Mc	3.6 Mc
MudFish [6]	4	88	149	38 B	14400 B	14.8 Mc	15.3 Mc
Mesquite [27] – Fast	4	88	149	38 B	9492 B	15.4 Mc	12.1 Mc
Mesquite [27] – Compact	4	88	149	38 B	8844 B	30.7 Mc	24.4 Mc
Fen22-gf251 [12] – Fast	251	40	135	56 B	8488 B	8.3 Mc	-
Fen22-gf251 [12] – Short	251	40	135	56 B	7114 B	22.8 Mc	-
MQOM-L1-gf251 – Fast	251	43	144	59 B	7809 B	11.5 Mc	10.16 Mc
MQOM-L1-gf251 – Short	251	43	144	59 B	6575 B	28.5 Mc	27.3 Mc
MQOM-L1-gf31 – Fast	31	49	143	47 B	7621 B	17.7 Mc	15.5 Mc
MQOM-L1-gf31 – Short	31	49	143	47 B	6348 B	44.4 Mc	41.7 Mc

- Shortest signatures for non-structured MQ
- Other MQ signature schemes submitted to NIST
    - either have large public keys (e.g. UOV)
    - or are based on recent structured assumptions (e.g. MAYO)
- Other MPCitH schemes have 5—10 KB signature sizes (based on different assumptions)

Roadmap

- Technical background
- MQOM MPC protocol
- SDitH MPC protocol
- Threshold MPCitH
- MQOM signature scheme
- **SDitH signature scheme**

Syndrome Decoding in the Head (SDitH)

Aguilar Melchor, Feneuil, Gama, Gueron,
Howe, Joseph, Joux, Persichetti,
Randrianarisoa, Rivain, Yue

- Originally proposed in **[FJR22]**
- Two variants:
  - "Hypercube": additive sharing w. seed trees **[KKW18]** & hypercube technique **[AMGHHJY23]**
  - "Threshold": threshold MPCitH **[FR22]**

Choice of parameters

- Two fields $\mathbb{F}_{251}$ and $\mathbb{F}_{256}$
    - Good size for SDitH / elements hold in bytes
    - Binary vs. prime (latter might be more conservative?)
    - $\mathbb{F}_{251}$ better for arithmetic (in particular in absence of carry-less multiplier)
    - $\mathbb{F}_{256}$ better for pseudo-random sampling

Choice of parameters

- Two fields $\mathbb{F}_{251}$ and $\mathbb{F}_{256}$
  - Good size for SDitH / elements hold in bytes
  - Binary vs. prime (latter might be more conservative?)
  - $\mathbb{F}_{251}$ better for arithmetic (in particular in absence of carry-less multiplier)
  - $\mathbb{F}_{256}$ better for pseudo-random sampling
- Other SD parameters (m, k, w, d) chosen to resist
  - Information Set Decoding (ISD)
  - Generalised Birthday Algorithms (GBA)

while minimising the signature size

Choice of parameters

- Two fields $\mathbb{F}_{251}$ and $\mathbb{F}_{256}$
  - Good size for SDitH / elements hold in bytes
  - Binary vs. prime (latter might be more conservative?)
  - $\mathbb{F}_{251}$ better for arithmetic (in particular in absence of carry-less multiplier)
  - $\mathbb{F}_{256}$ better for pseudo-random sampling
- Other SD parameters (m, k, w, d) chosen to resist
  - Information Set Decoding (ISD)
  - Generalised Birthday Algorithms (GBA)

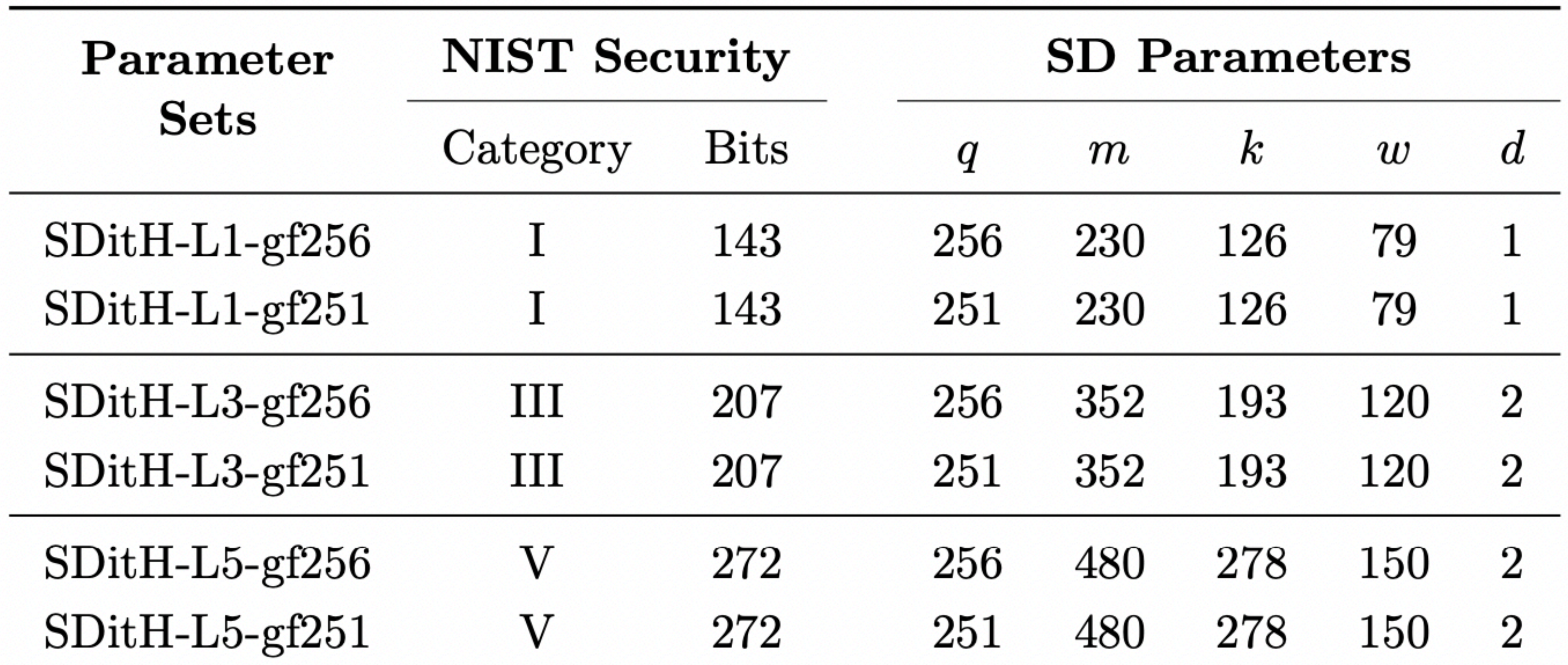
while minimising the signature size

- MPC parameters
  - $N = 256$ for hypercube variant
  - $N = q$ and $\ell = 3$ for threshold variant

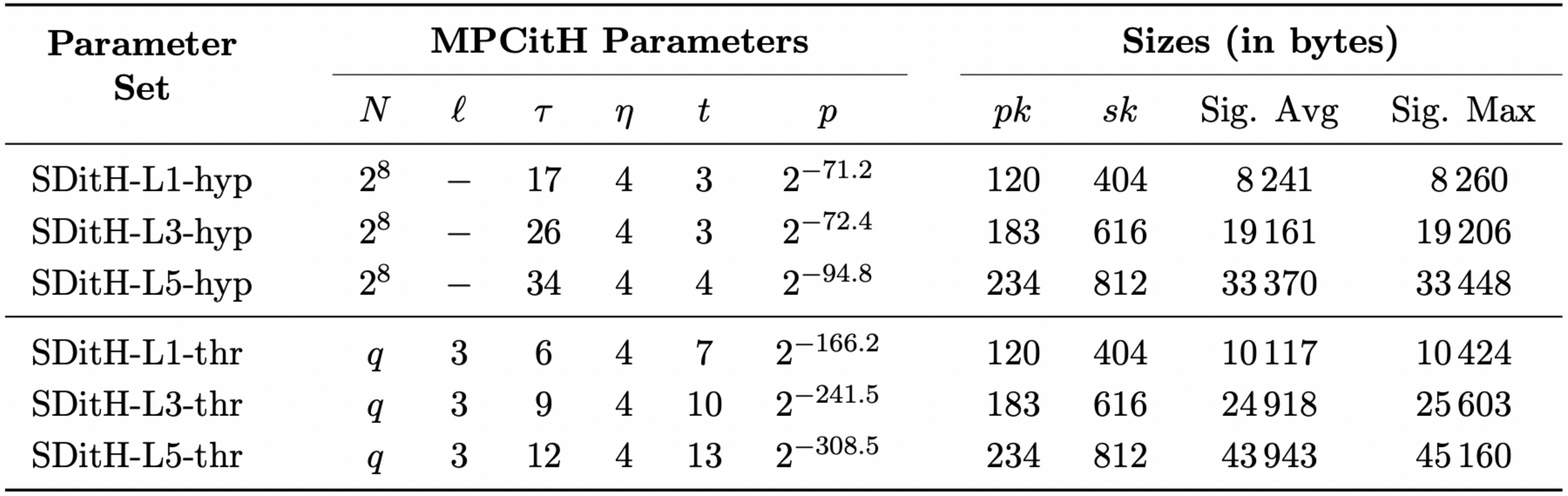
    → good tradeoff between signature size and timings
  - $\eta = 4$: common field extensions to all variants and security categories

    → easy implementation / good tradeoff between the different settings
  - t and τ chosen to minimise the signature size for target security

Parameters and sizes



Parameter Sets	NIST Security		SD Parameters				
	Category	Bits	q	m	k	w	d
SDitH-L1-gf256	I	143	256	230	126	79	1
SDitH-L1-gf251	I	143	251	230	126	79	1
SDitH-L3-gf256	III	207	256	352	193	120	2
SDitH-L3-gf251	III	207	251	352	193	120	2
SDitH-L5-gf256	V	272	256	480	278	150	2
SDitH-L5-gf251	V	272	251	480	278	150	2



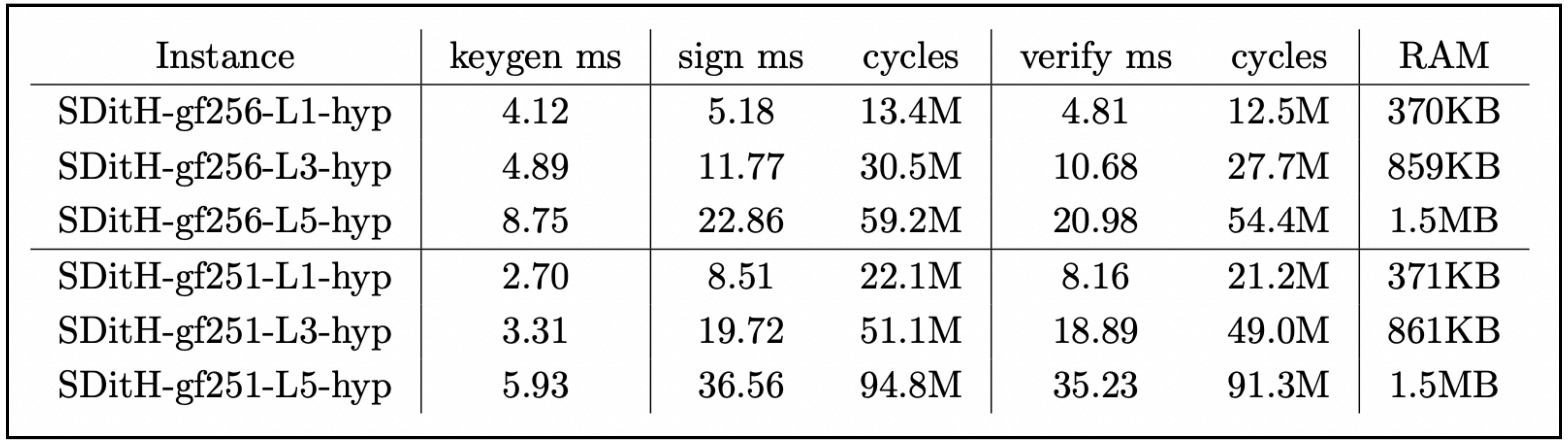
Parameter Set	MPCitH Parameters						Sizes (in bytes)			
	N	ℓ	τ	η	t	p	pk	sk	Sig. Avg	Sig. Max
SDitH-L1-hyp	2^8	–	17	4	3	$2^{-71.2}$	120	404	8 241	8 260
SDitH-L3-hyp	2^8	–	26	4	3	$2^{-72.4}$	183	616	19 161	19 206
SDitH-L5-hyp	2^8	–	34	4	4	$2^{-94.8}$	234	812	33 370	33 448
SDitH-L1-thr	q	3	6	4	7	$2^{-166.2}$	120	404	10 117	10 424
SDitH-L3-thr	q	3	9	4	10	$2^{-241.5}$	183	616	24 918	25 603
SDitH-L5-thr	q	3	12	4	13	$2^{-308.5}$	234	812	43 943	45 160

Sig sizes:

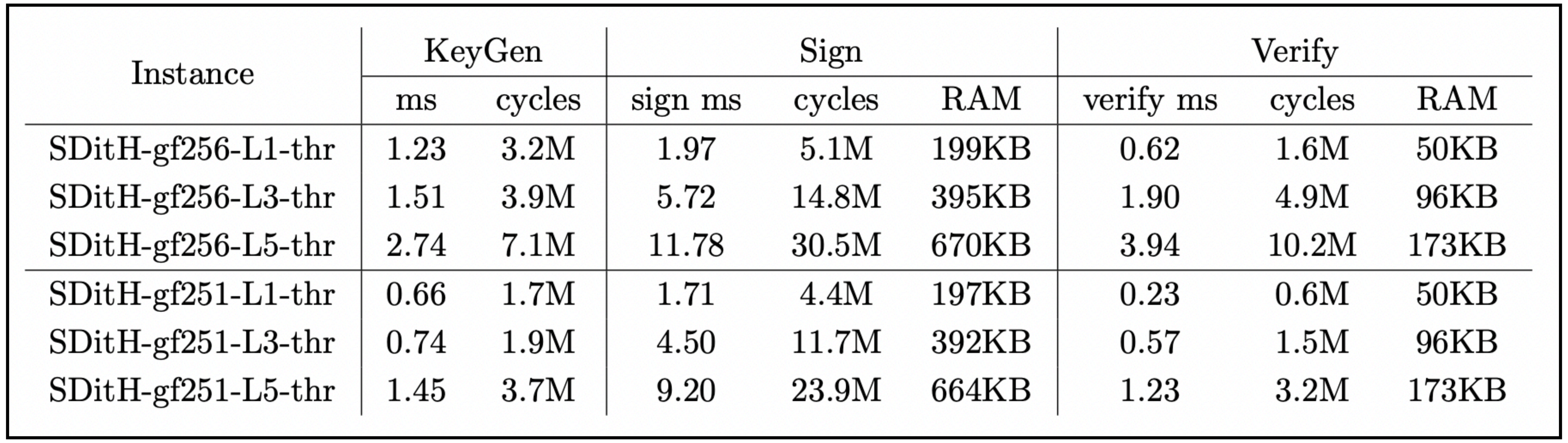
- Hypercube: 8.2 KB (19 KB, 33 KB)
- Threshold: 10.4 KB (25 KB, 45 KB)
- Hypercube 2KB shorter

Small keys

Performances



Instance	keygen ms	sign ms	cycles	verify ms	cycles	RAM
SDitH-gf256-L1-hyp	4.12	5.18	13.4M	4.81	12.5M	370KB
SDitH-gf256-L3-hyp	4.89	11.77	30.5M	10.68	27.7M	859KB
SDitH-gf256-L5-hyp	8.75	22.86	59.2M	20.98	54.4M	1.5MB
SDitH-gf251-L1-hyp	2.70	8.51	22.1M	8.16	21.2M	371KB
SDitH-gf251-L3-hyp	3.31	19.72	51.1M	18.89	49.0M	861KB
SDitH-gf251-L5-hyp	5.93	36.56	94.8M	35.23	91.3M	1.5MB



Instance	KeyGen		Sign			Verify		
	ms	cycles	sign ms	cycles	RAM	verify ms	cycles	RAM
SDitH-gf256-L1-thr	1.23	3.2M	1.97	5.1M	199KB	0.62	1.6M	50KB
SDitH-gf256-L3-thr	1.51	3.9M	5.72	14.8M	395KB	1.90	4.9M	96KB
SDitH-gf256-L5-thr	2.74	7.1M	11.78	30.5M	670KB	3.94	10.2M	173KB
SDitH-gf251-L1-thr	0.66	1.7M	1.71	4.4M	197KB	0.23	0.6M	50KB
SDitH-gf251-L3-thr	0.74	1.9M	4.50	11.7M	392KB	0.57	1.5M	96KB
SDitH-gf251-L5-thr	1.45	3.7M	9.20	23.9M	664KB	1.23	3.2M	173KB

Signing timings:

- Hypercube: 5.2 ms (12 ms, 23 ms)
- Threshold: 1.7 ms (5 ms, 9 ms)
- Threshold 2-3x faster

Verification timings:

- Hypercube: 4.8 ms (11 ms, 21 ms)
- Threshold: 0.2 ms (0.6 ms, 1.2 ms)
- Threshold ~20x faster

Comparison

- Shortest signatures for SD on random linear codes
- Only submission to NIST using Threshold MPCitH

  → fast variant (especially for verification)
- Other MPCitH schemes have 5–10 KB signature sizes (based on different assumptions)

Questions ?

References

- **[KKW18]** Katz, Kolesnikov, Wang: "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures" (CCS 2018)
- **[KZ20]** Kales, Zaverucha: "An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes" (CANS 2020)
- **[BN20]** Baum, Nof: "Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography" (PKC 2020)
- **[BDKOSZ21]** Baum, Delpech de Saint Guilhem, Kales, Orsini, Scholl, Zaverucha. "Banquet: Short and Fast Signatures from AES" (PKC 2021)
- **[DOT21]** Delpech de Saint Guilhem, Orsini, Tanguy. "Limbo: Efficient Zero-knowledge MPCitH-based Arguments" (CCS 2021)
- **[FJR22]** Feneuil, Joux, Rivain: "Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs" (CRYPTO 2022)
- **[BMSV22]** Bellini, Makarim, Sanna, Verbel. "An Estimator for the Hardness of the MQ Problem" (AFRICACRYPT 2022)
- **[FR22]** Feneuil, Rivain. "Threshold Linear Secret Sharing to the Rescue of MPC-in-the-Head" (ePrint 2022, to appear in ASIACRYPT 2023)
- **[Fen22]** Feneuil: "Building MPCitH-based Signatures from MQ, MinRank, Rank SD and PKP" (ePrint 2022)
- **[AGHHJY23]** Aguilar Melchor, Gama, Howe, Hülsing, Joseph, Yue: "The Return of the SDitH" (EUROCRYPT 2023)
- **[FR23]** Feneuil, Rivain. "MQOM: MQ on my Mind" (Submission to NIST call for additional post-quantum signature schemes, 2023)
- **[AGGHJJPRRY23]** Aguilar Melchor, Feneuil, Gama, Gueron, Howe, Joseph, Joux, Persichetti, Randrianarisoa, Rivain, Yue. "The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme" (Submission to NIST call for additional post-quantum signature schemes, 2023)