

SmallWood: Hash-Based Zero-Knowledge Arguments for Relatively Small Instances

Matthieu Rivain

Join work with Thibauld Feneuil

NIST Workshop on Multi-Party Threshold Schemes 2026

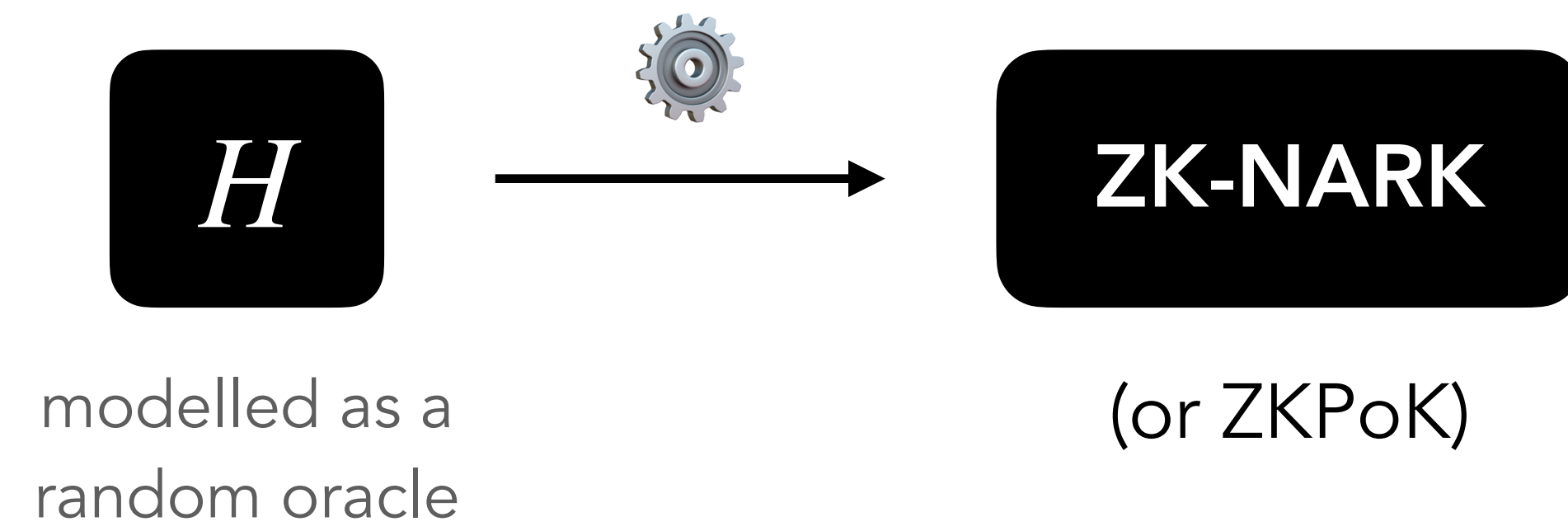
January 29, 2026



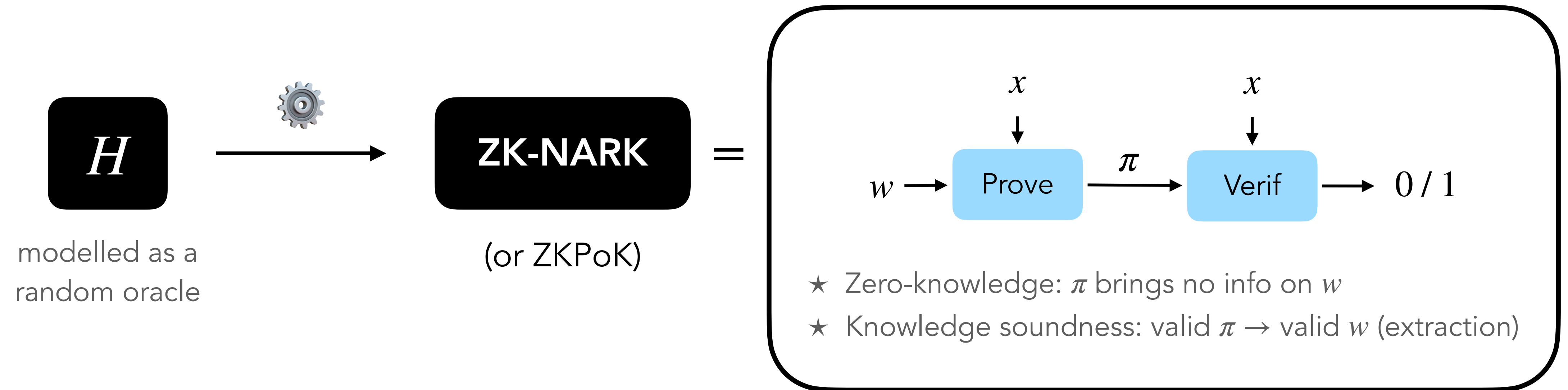
Roadmap

1. Intro
2. Design paradigm: PIOP + PCS
3. SmallWood PIOP / PACS constraint system
4. SmallWood PCS
5. Applications

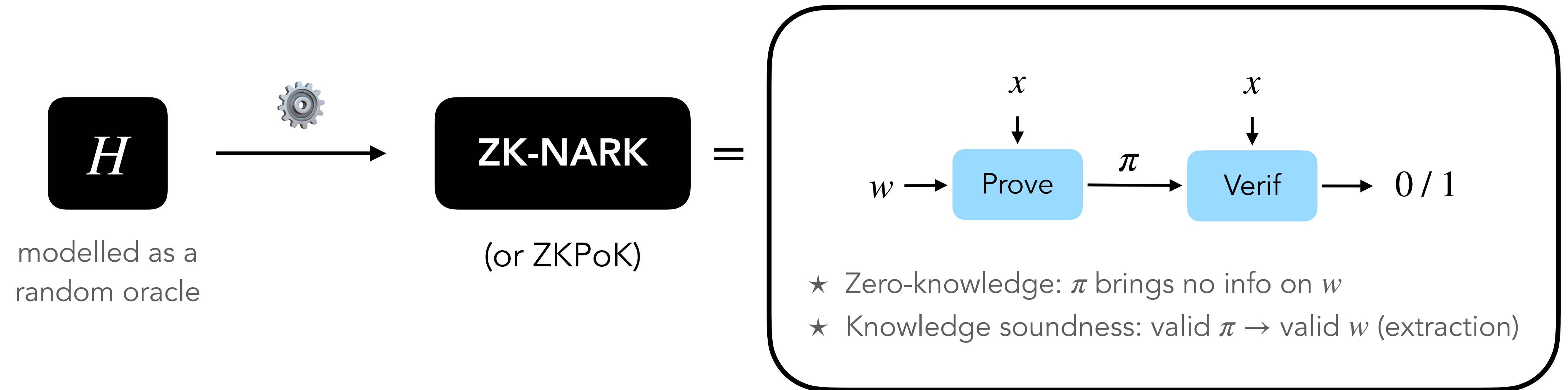
SmallWood: Hash-based ZK-NARK



SmallWood: Hash-based ZK-NARK



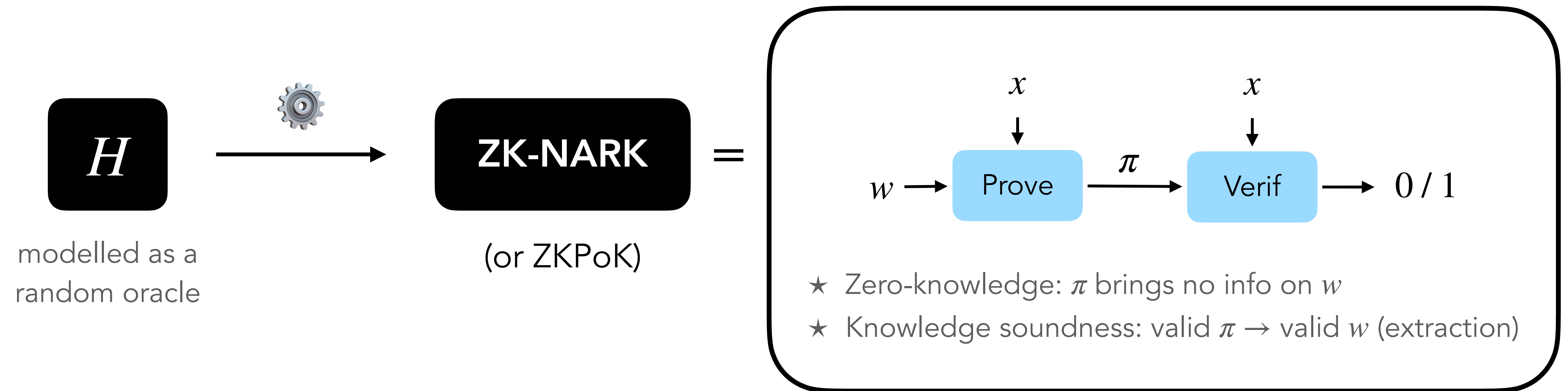
SmallWood: Hash-based ZK-NARK



Main features

- Conservative security, plausibly post-quantum
- Fast prover, faster verifier
- Compact proofs for "relatively small instances"
 - Extended witness from ≈ 500 B to 500 KB

SmallWood: Hash-based ZK-NARK



Main features

- Conservative security, plausibly post-quantum
- Fast prover, faster verifier
- Compact proofs for “relatively small instances”
 - Extended witness from ≈ 500 B to 500 KB

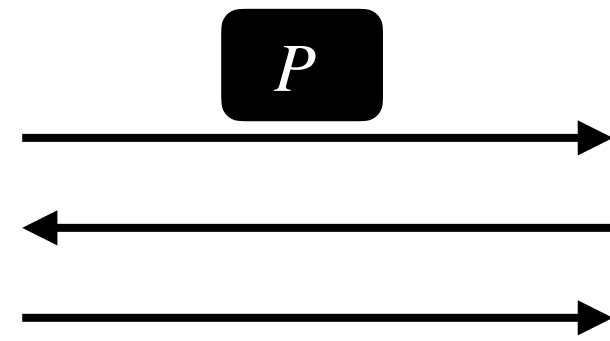
Built upon

- Ligerio [AHIV17, AHIV23]
- Threshold-Computation-in-the-Head [FR23, FR25]
- Brakedown [GLS+23]

PIOP + PCS

PIOP + PCS

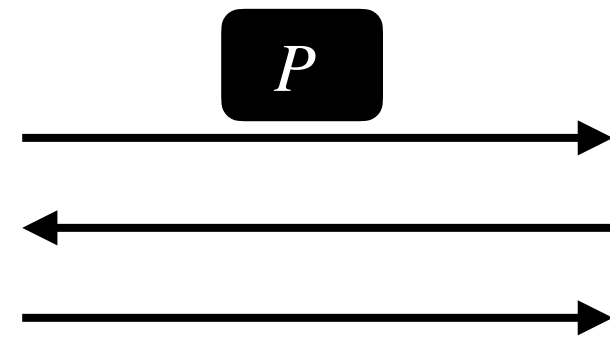
Polynomial Interactive
Oracle Proof



PIOP + PCS

Polynomial Interactive
Oracle Proof

Polynomial Commitment
Scheme



Commit (P) \rightarrow com
Prove (P, e) $\rightarrow P(e), \pi_e$
Verif (com, p_e, π_e) \rightarrow 0/1

PIOP + PCS



Statement x , secret witness w



Statement x

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

polynomial oracle

P



Statement x

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

polynomial oracle

P

c



Statement x

$c \leftarrow \$$

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

polynomial oracle

P

c

Q

$c \leftarrow \$$



Statement x

PIOP + PCS

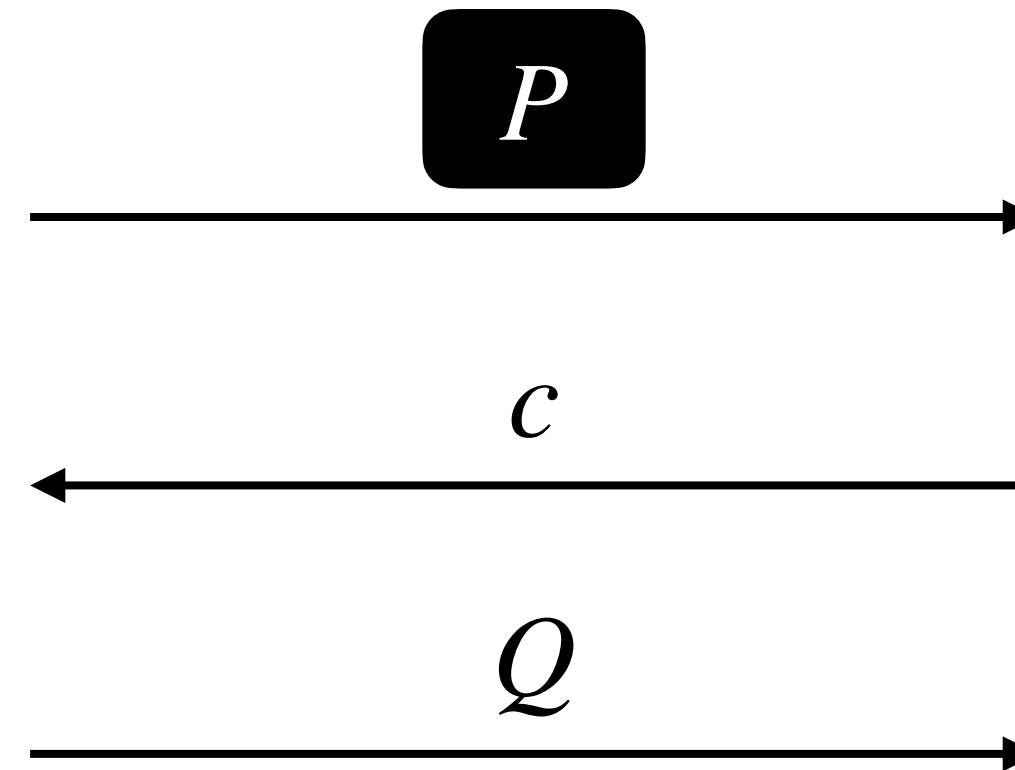


Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

polynomial oracle



Statement x

$c \leftarrow \$$

(1) Check Q (e.g. $Q(0) = 0$?)

(2) Oracle queries:

$E \subseteq \mathbb{F} \leftarrow \$$

$\forall e \in E : p_e = \mathbf{P}(e)$

check $Q(e) = \Psi(p_e, x, c)$

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

polynomial oracle

P

c

Q



Statement x

$c \leftarrow \$$

(1) Check Q (e.g. $Q(0) = 0$?)

(2) Oracle queries:

$E \subseteq \mathbb{F} \leftarrow \$$

$\forall e \in E : p_e = P(e)$

check $Q(e) = \Psi(p_e, x, c)$

If Q well defined: (1) satisfied
 $\iff w$ valid (w.o.p.)

PIOP + PCS

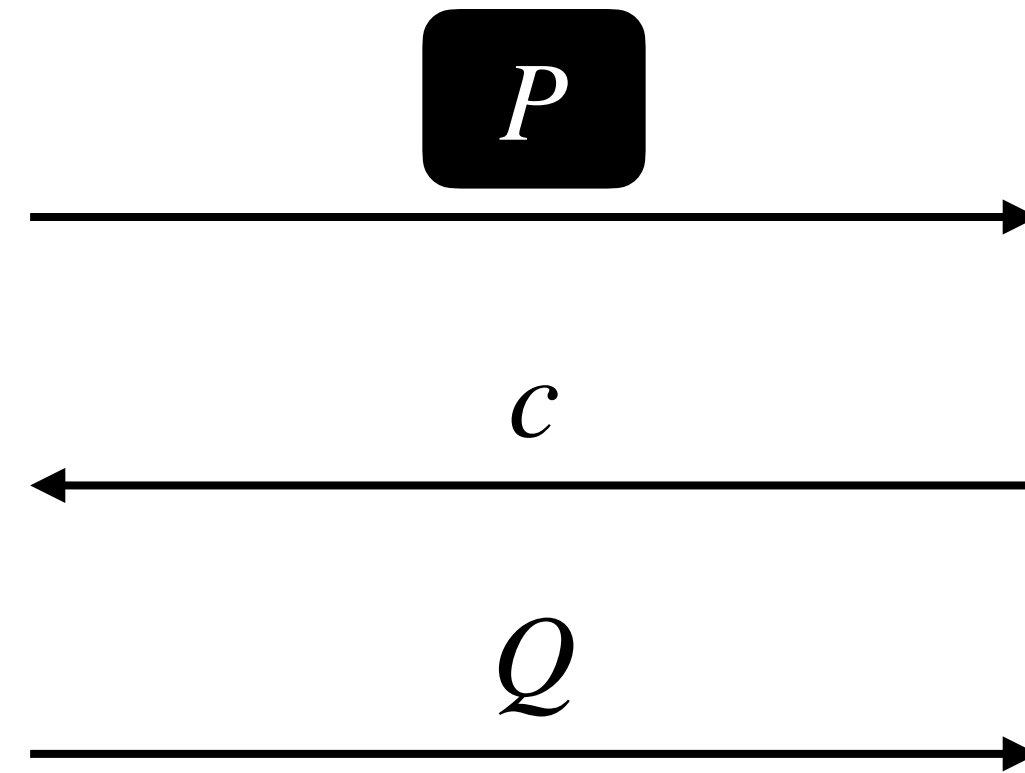


Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

polynomial oracle



Statement x

$c \leftarrow \$$

(1) Check Q (e.g. $Q(0) = 0$?)

(2) Oracle queries:

$E \subseteq \mathbb{F} \leftarrow \$$

$\forall e \in E : p_e = \mathbf{P}(e)$

check $Q(e) = \Psi(p_e, x, c)$

If Q well defined: (1) satisfied
 $\iff w$ valid (w.o.p.)

If Q **not** well defined, by Schwartz-Zippel lemma:
 $\Pr[(2) \text{ OK}] \leq (d_Q / |\mathbb{F}|)^{|E|}$

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$



Statement x

$c \leftarrow \$$

(1) Check Q (e.g. $Q(0) = 0$?)

$\text{com} = \text{PCS.Commit}(P)$

c

Q

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$



Statement x

$c \leftarrow \$$

(1) Check Q (e.g. $Q(0) = 0$?)

(2) Evaluation queries:

$E \subseteq \mathbb{F} \leftarrow \$$

$\text{com} = \text{PCS.Commit}(P)$

c

Q

E

PIOP + PCS



Statement x , secret witness w

$r \leftarrow \$$
 $(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

$\forall e \in E : p_e = P(e)$
 $\pi_e \leftarrow \text{PCS.Prove}(\text{com}, P, e)$



Statement x

$c \leftarrow \$$

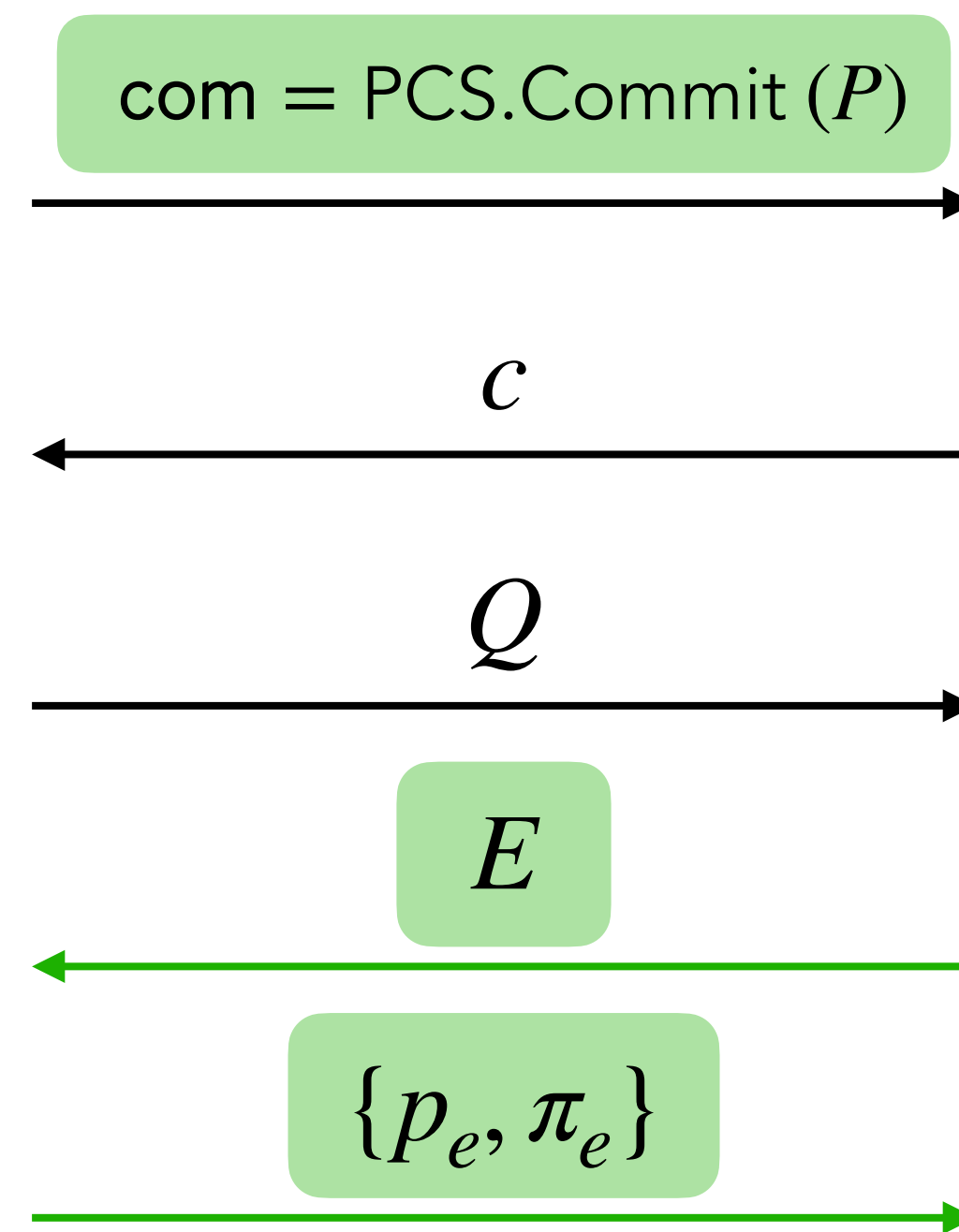
(1) Check Q (e.g. $Q(0) = 0$?)

(2) Evaluation queries:

$E \subseteq \mathbb{F} \leftarrow \$$

$\forall e \in E : \text{PCS.Verif}(\text{com}, p_e, \pi_e)$

check $Q(e) = \Psi(p_e, x, c)$



PIOP + PCS



Statement x , secret witness w

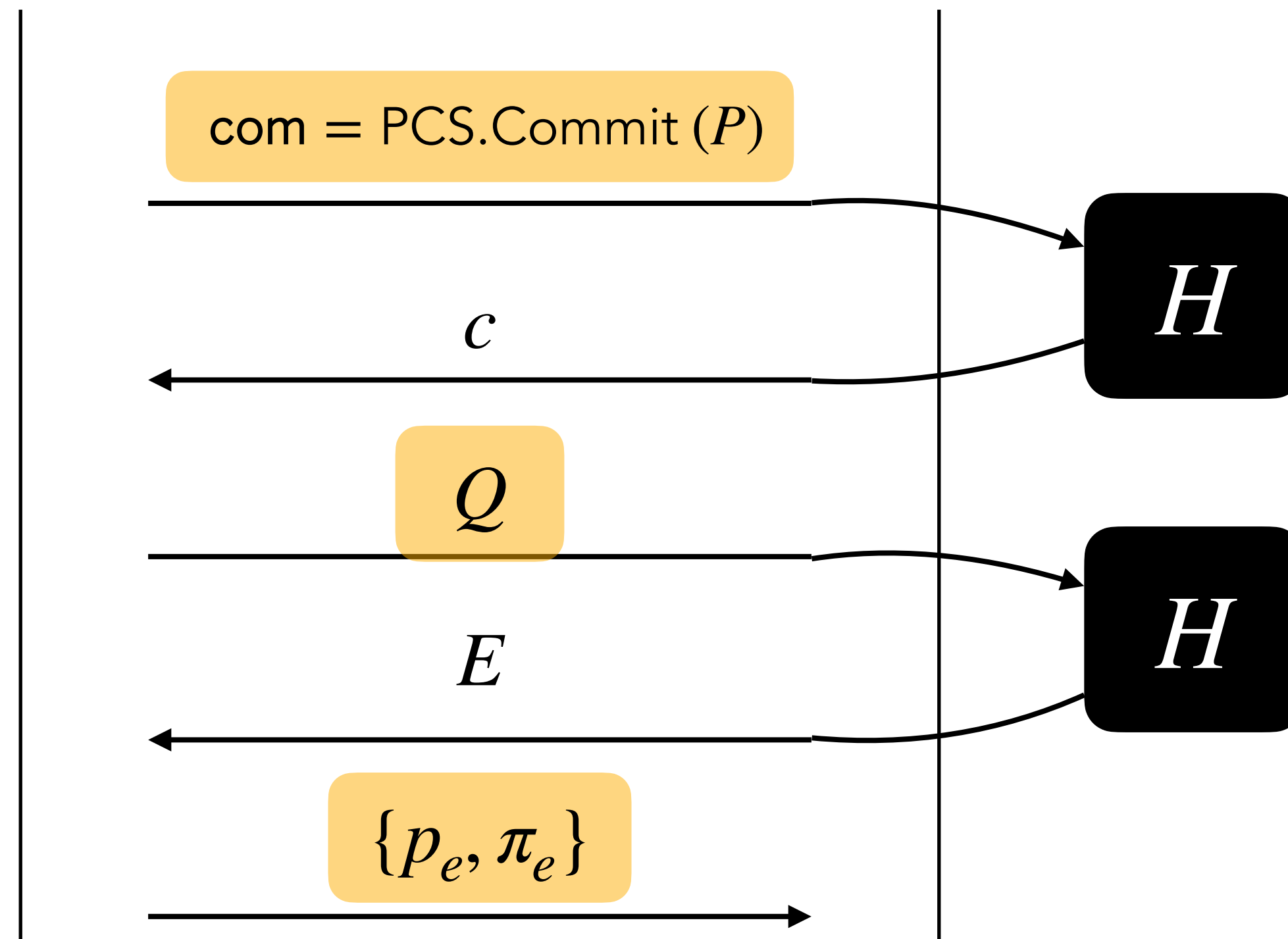
$r \leftarrow \$$

$(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

$\forall e \in E : p_e = P(e)$

$\pi_e \leftarrow \text{PCS.Prove}(\text{com}, P, e)$



Fiat-Shamir :
PIOP + PCS \mapsto NARK

PIOP + PCS



Statement x , secret witness w

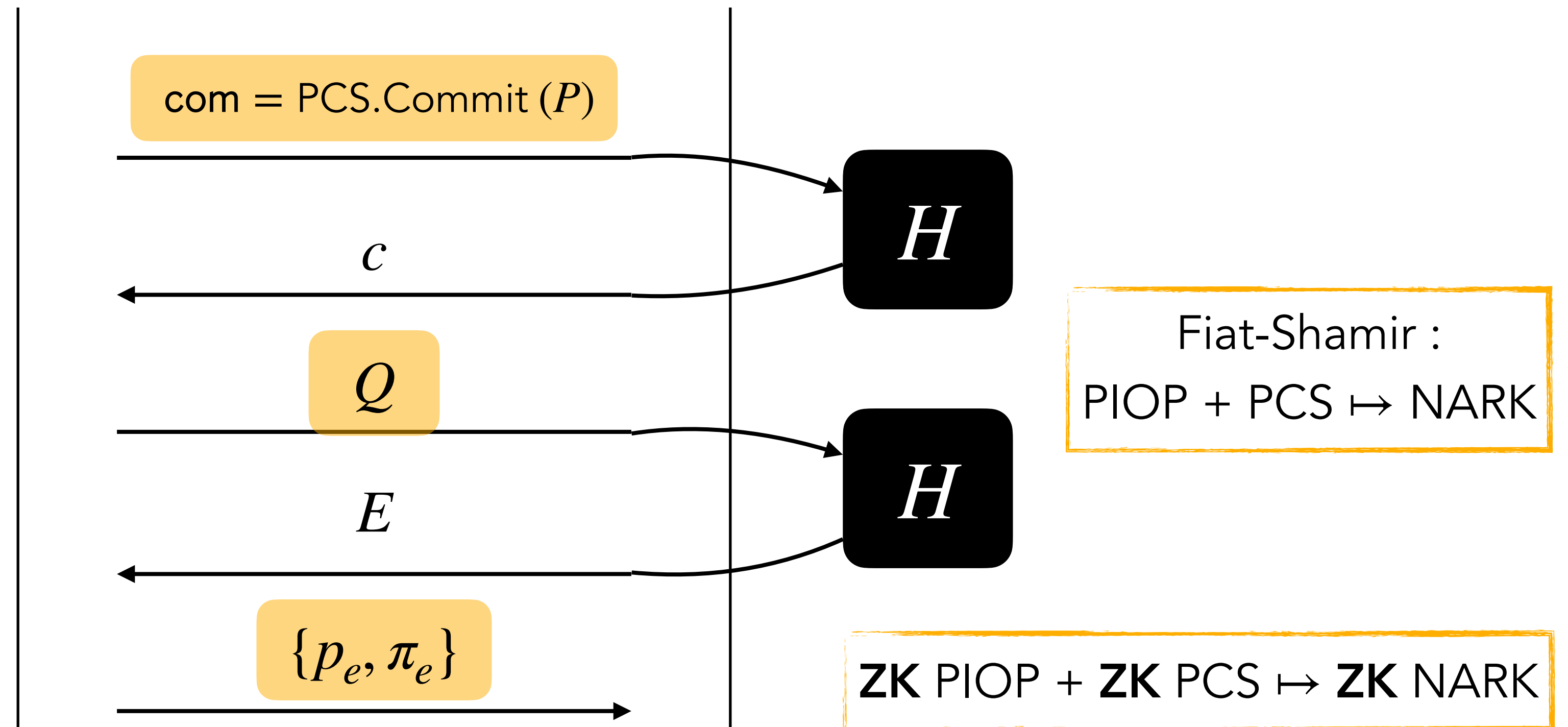
$r \leftarrow \$$

$(w, r) \mapsto P$

$Q = \Psi(P, x, c)$

$\forall e \in E : p_e = P(e)$

$\pi_e \leftarrow \text{PCS.Prove}(\text{com}, P, e)$



PACS



Parallel and Aggregated Constraint System
(generalisation of Ligerio)

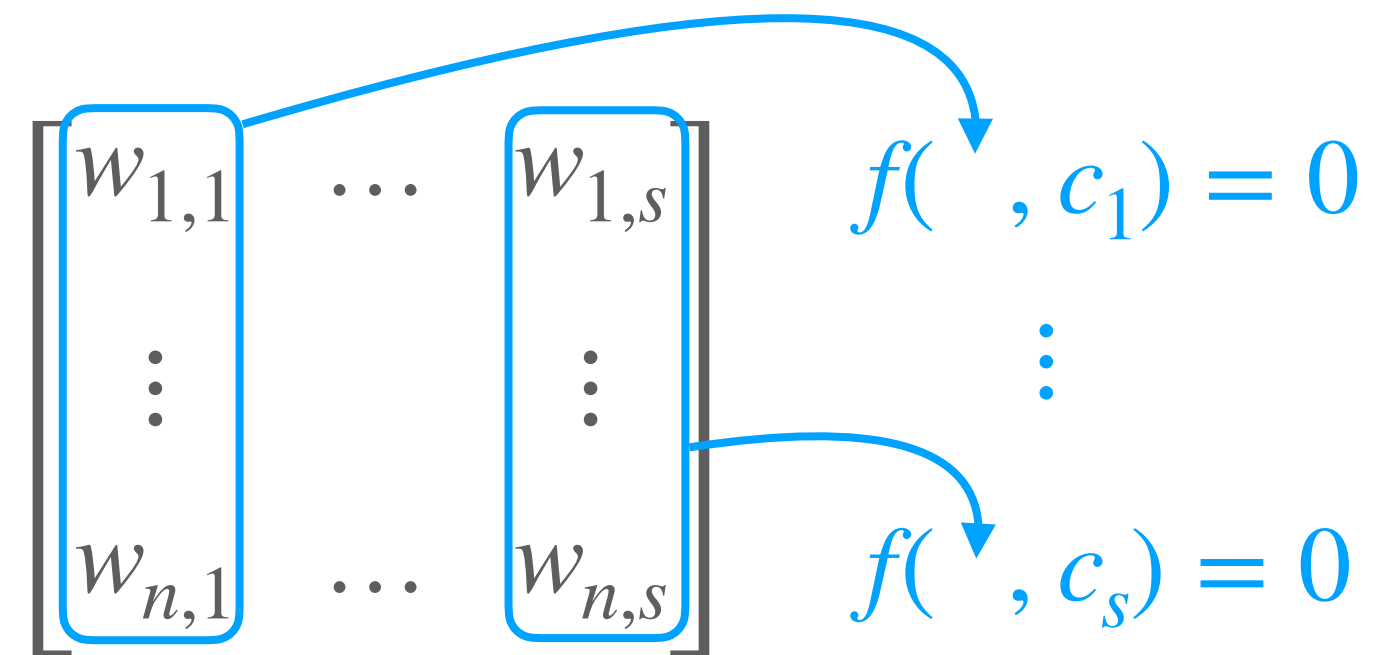
PACS

Extended witness

$$\begin{bmatrix} w_{1,1} & \cdots & w_{1,s} \\ \vdots & & \vdots \\ w_{n,1} & \cdots & w_{n,s} \end{bmatrix}$$

PACS

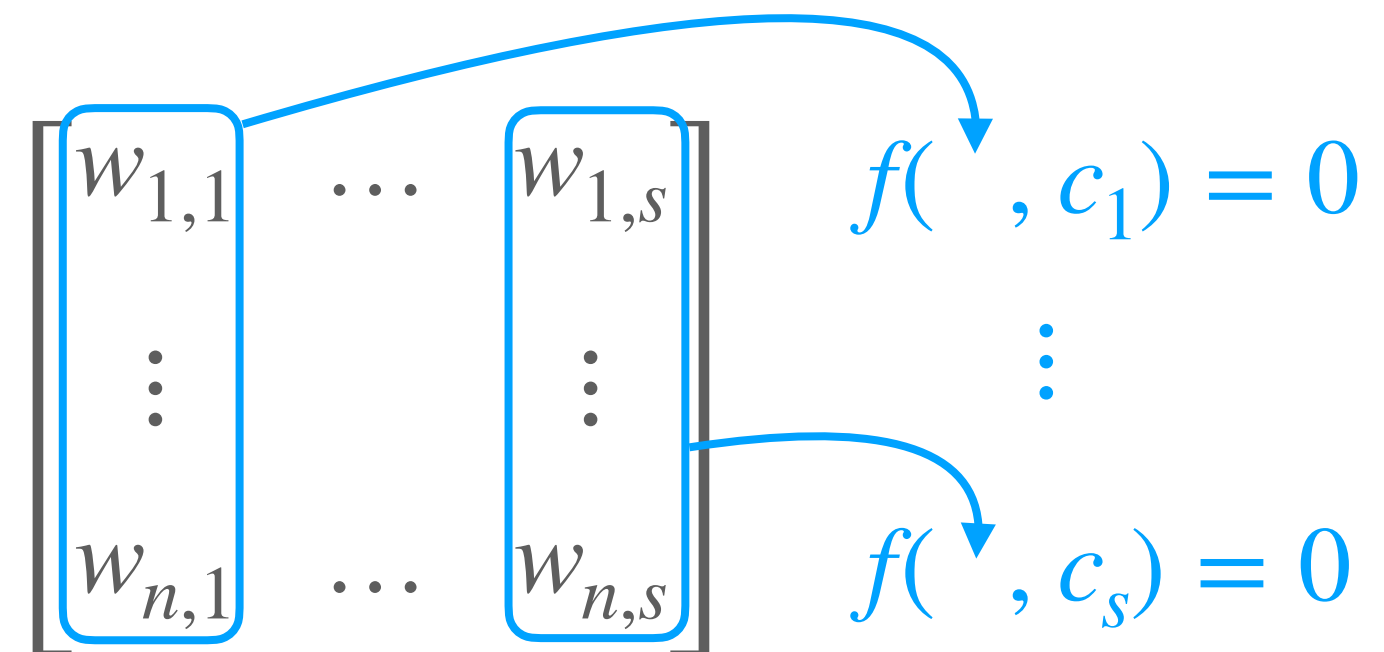
Extended witness



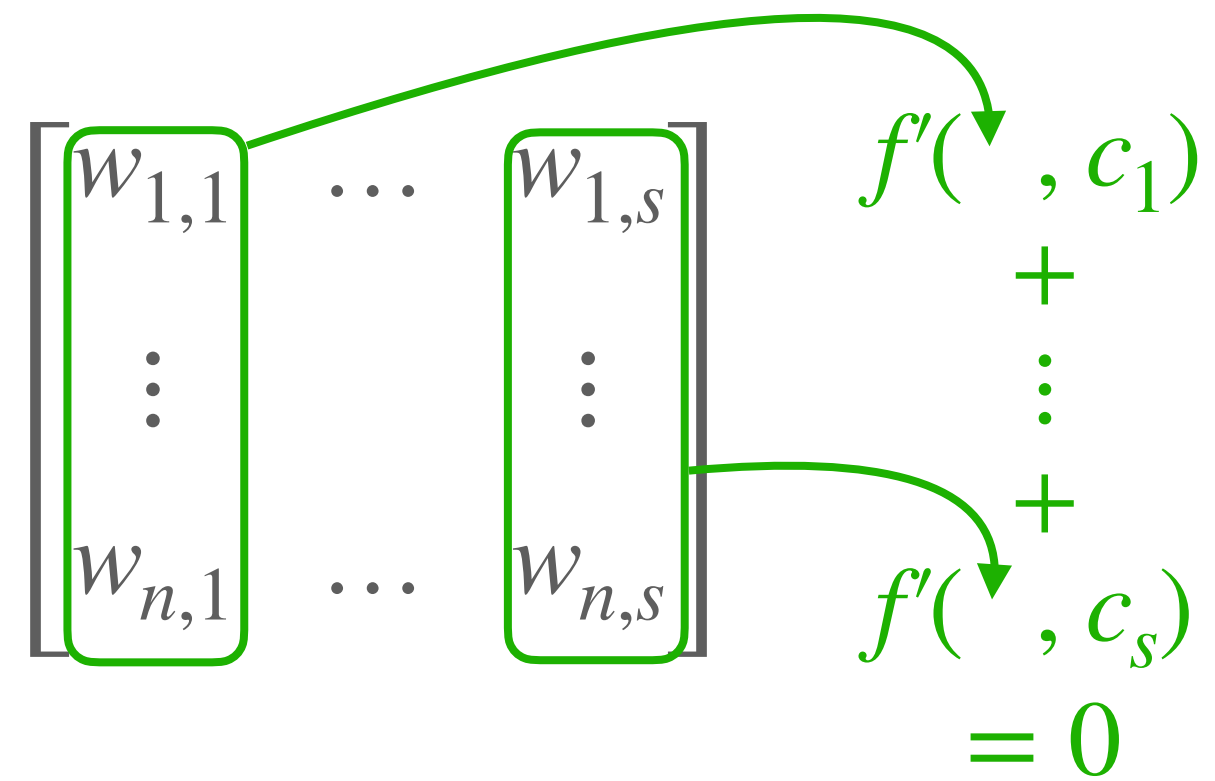
Parallel polynomial constraint

PACS

Extended witness



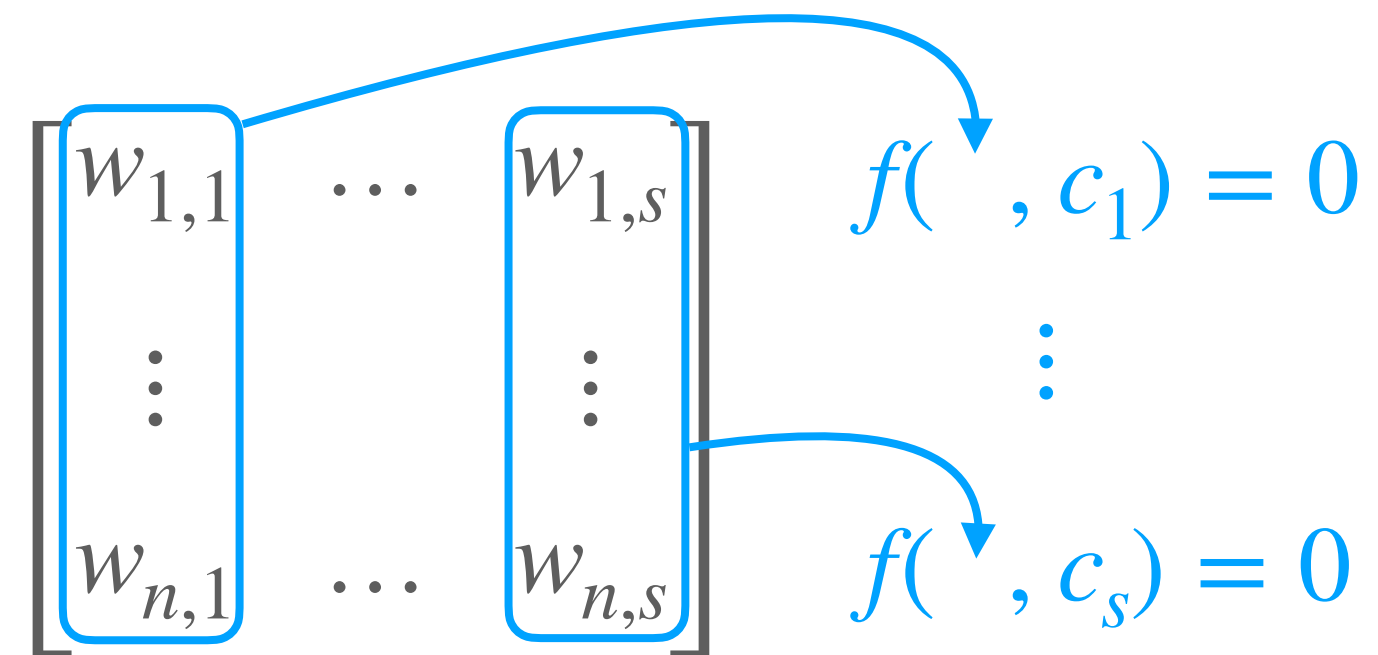
Parallel polynomial constraint



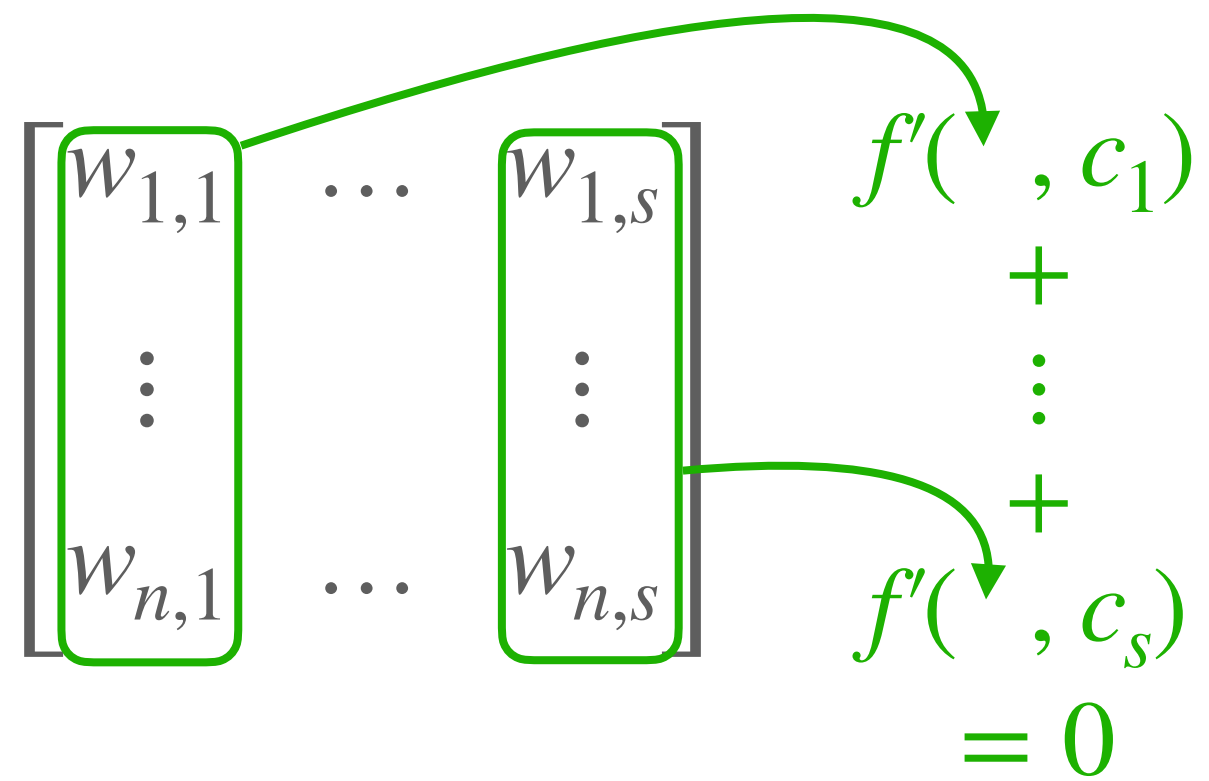
Aggregated parallel polynomial constraint

PACS

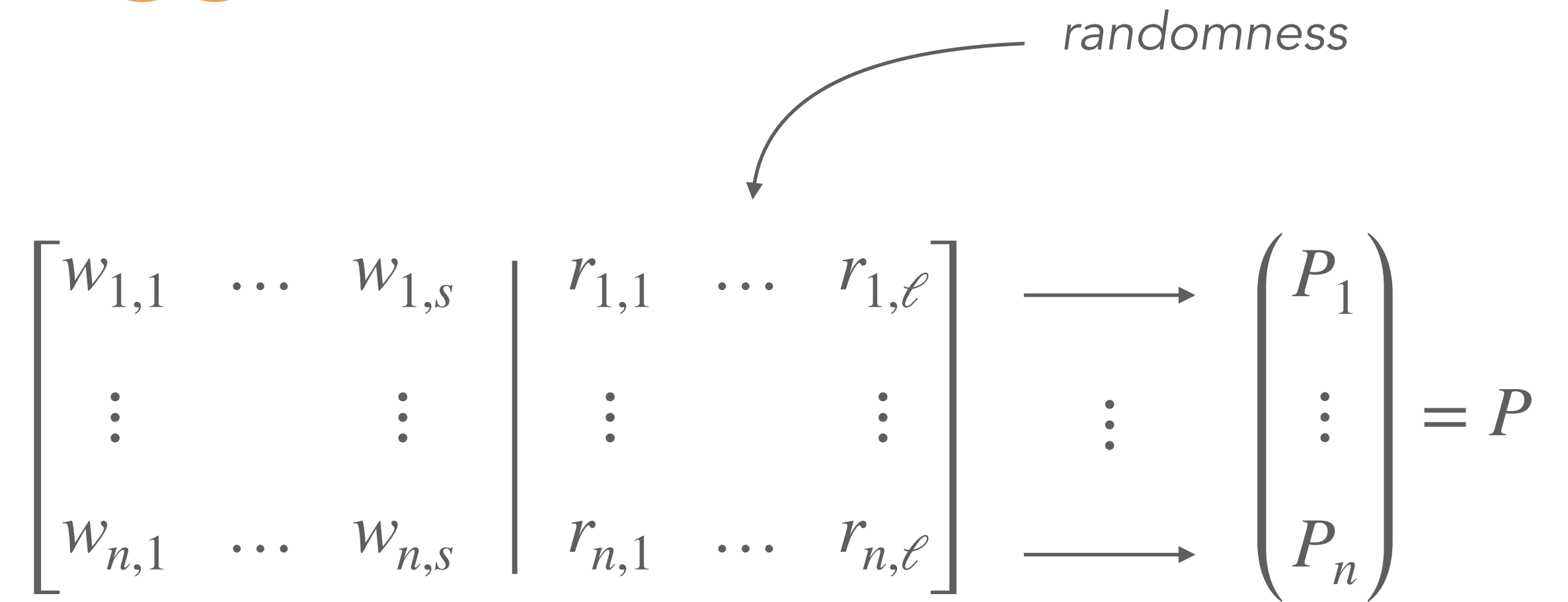
Extended witness



Parallel polynomial constraint



Aggregated parallel polynomial constraint

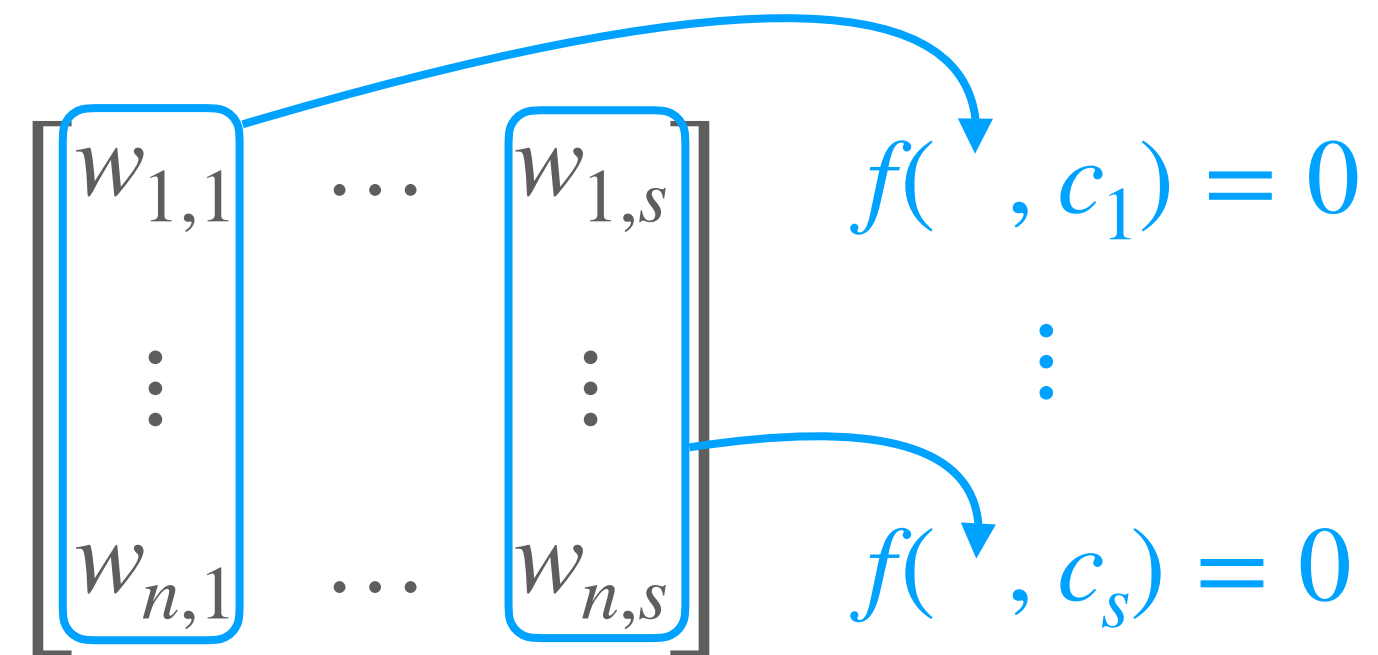


Ω : witness support
 witness = $\{P(\omega)\}_{\omega \in \Omega}$

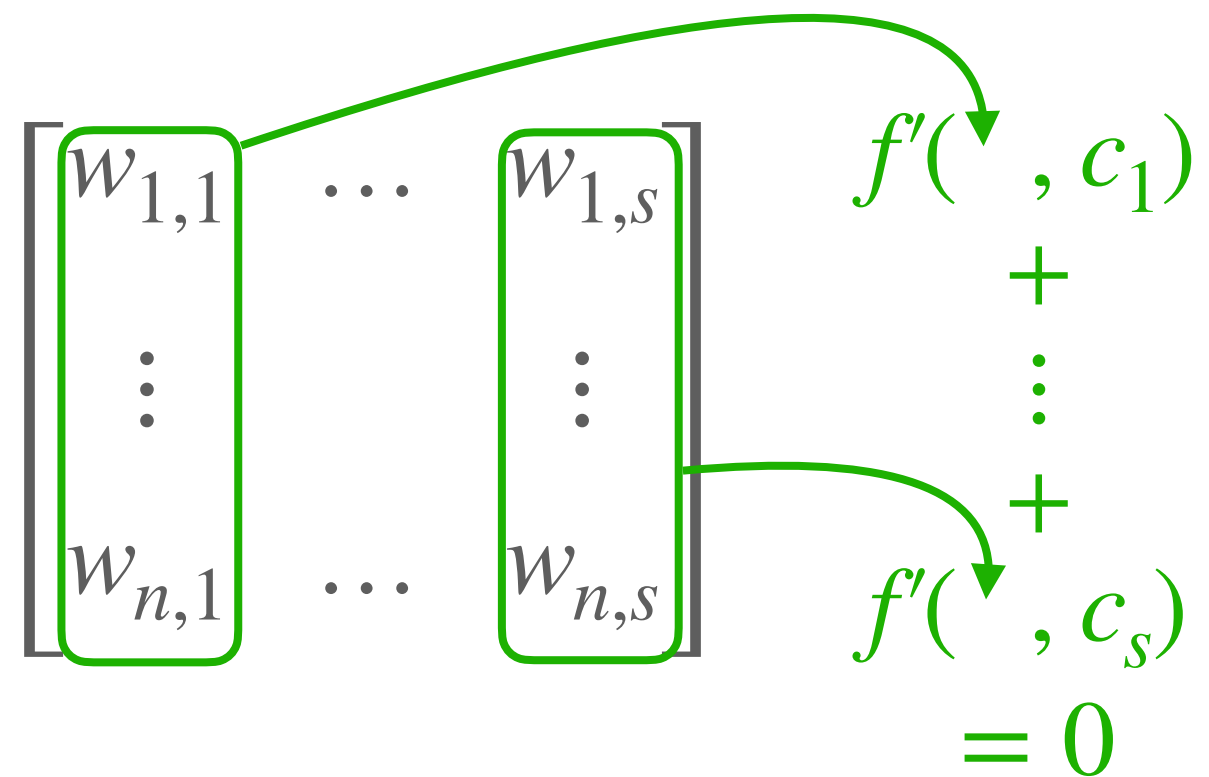
Lagrange interpolation

PACS

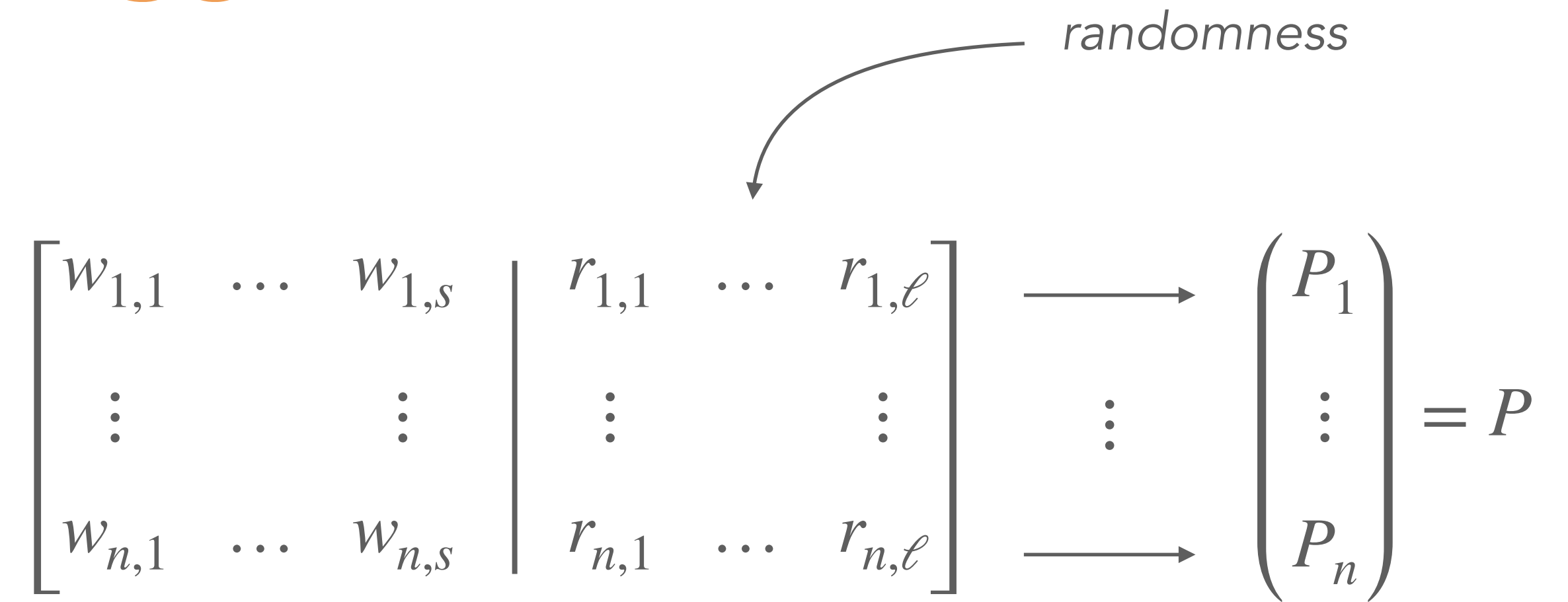
Extended witness



Parallel polynomial constraint



Aggregated parallel polynomial constraint



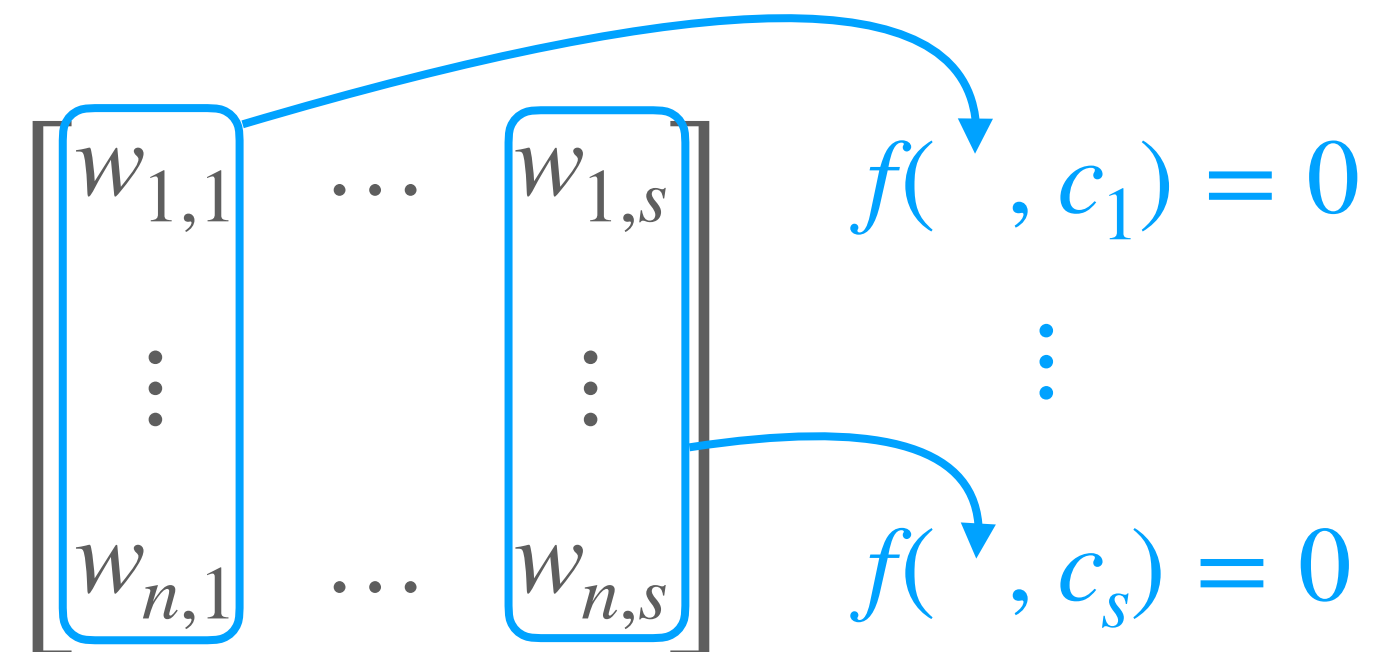
Ω : witness support
 witness = $\{P(\omega)\}_{\omega \in \Omega}$

Lagrange
 interpolation

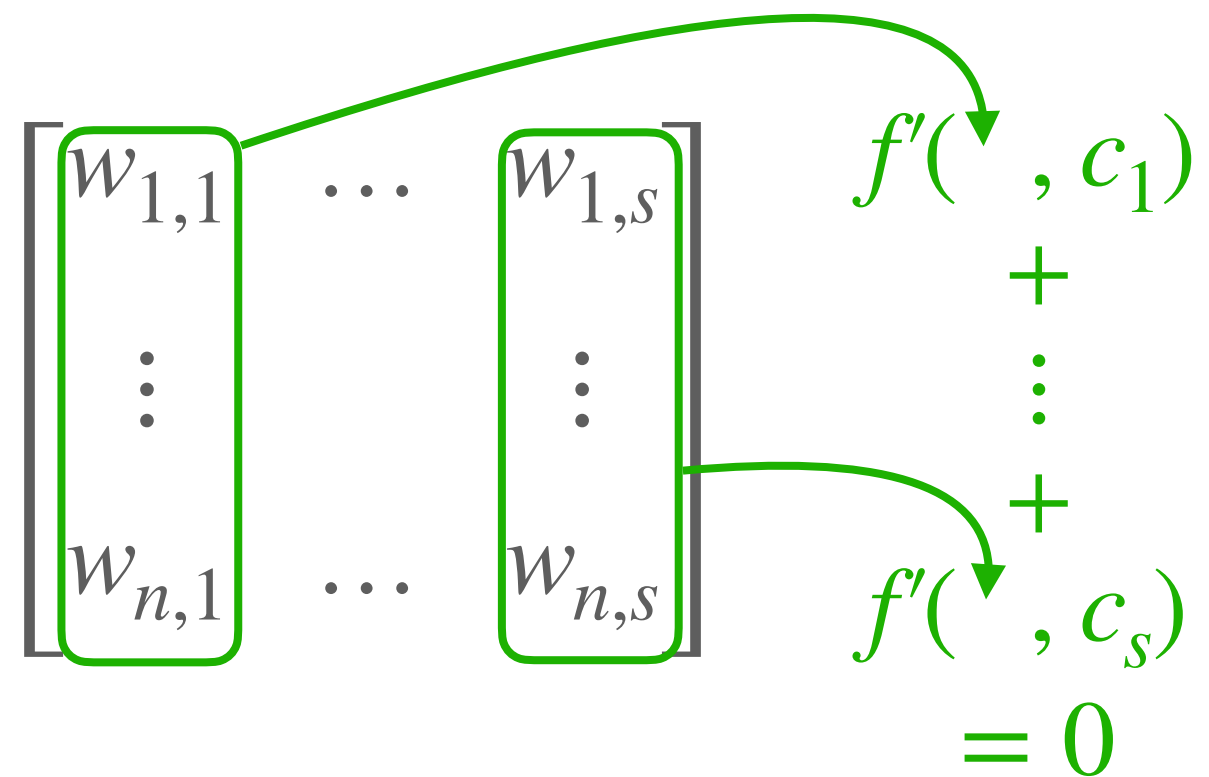
$$Q(X) = f(P(X), c(X)) \implies Q(\omega) = 0 \quad \forall \omega \in \Omega$$

PACS

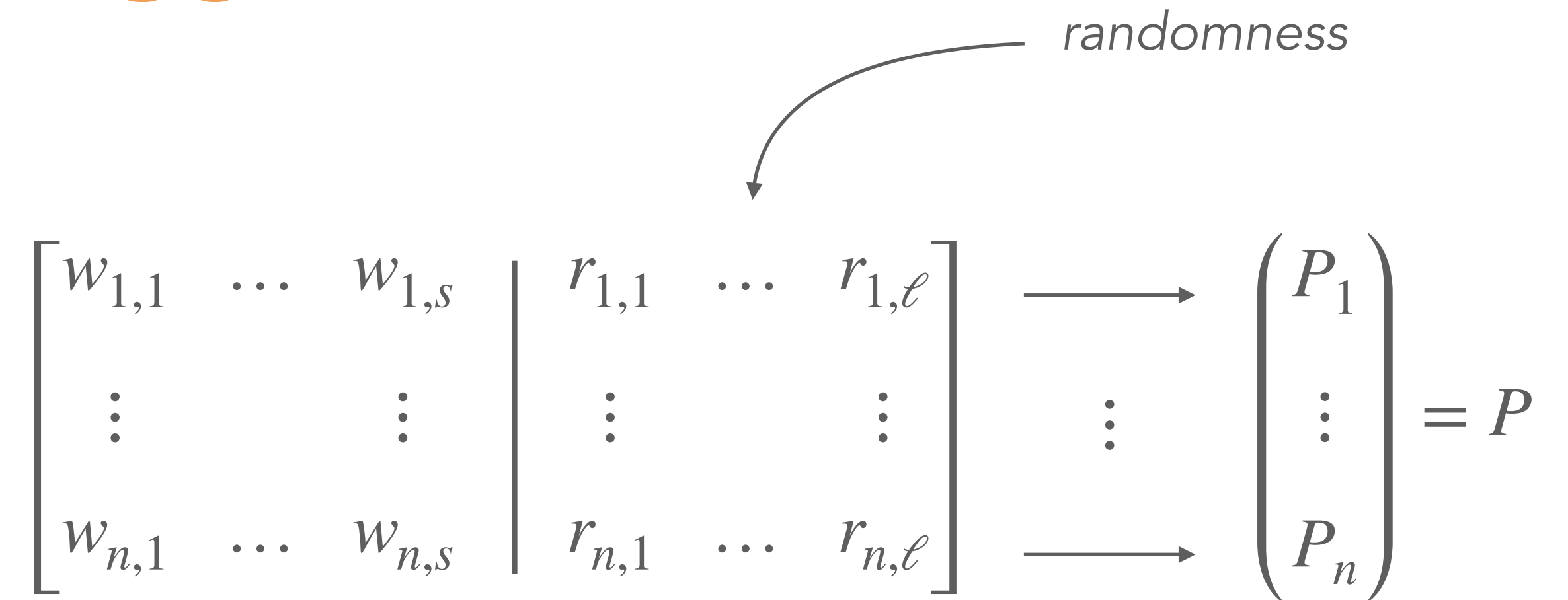
Extended witness



Parallel polynomial constraint



Aggregated parallel polynomial constraint



Ω : witness support
 witness = $\{P(\omega)\}_{\omega \in \Omega}$

Lagrange
 interpolation

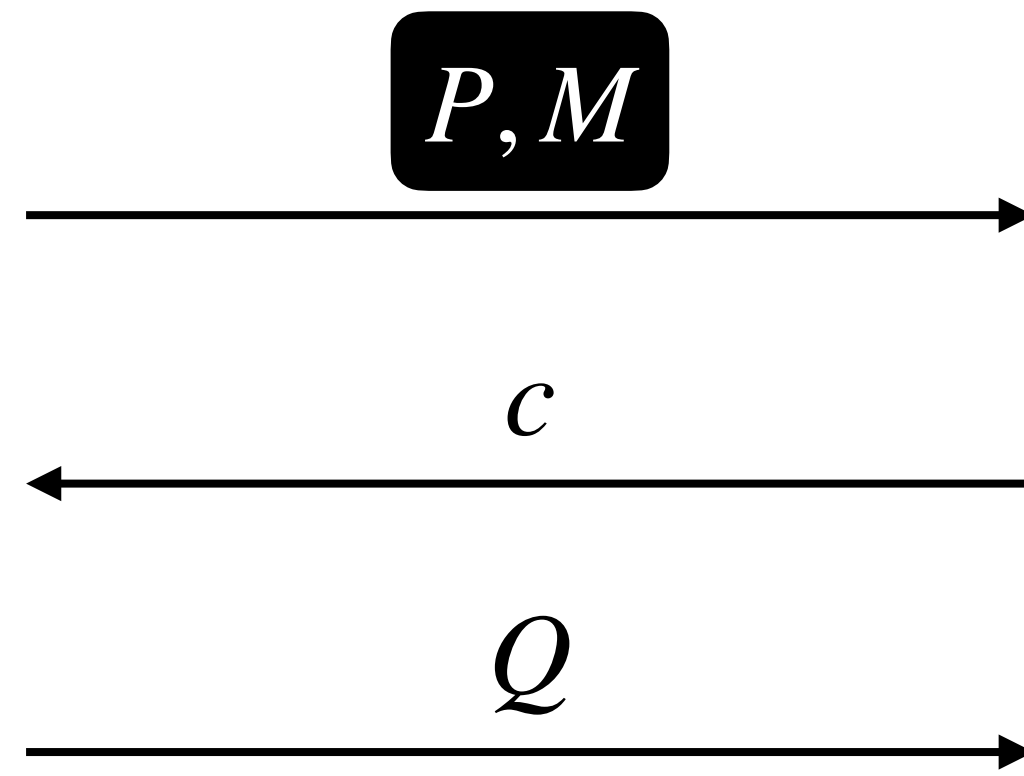
$$Q(X) = f(P(X), c(X)) \implies Q(\omega) = 0 \quad \forall \omega \in \Omega$$

$$Q'(X) = f'(P(X), c'(X)) \implies \sum_{\Omega} Q'(\omega) = 0$$

PACS PIOP



f_1, \dots, f_{m_1} p.p. constraints
 f'_1, \dots, f'_{m_2} a.p.p. constraints



PACS PIOP

Random masking
polynomial M

$$\text{s.t. } \sum_{\omega \in \Omega} M(\omega) = 0$$



f_1, \dots, f_{m_1} p.p. constraints

f'_1, \dots, f'_{m_2} a.p.p. constraints

P, M



c



Q



PACS PIOP

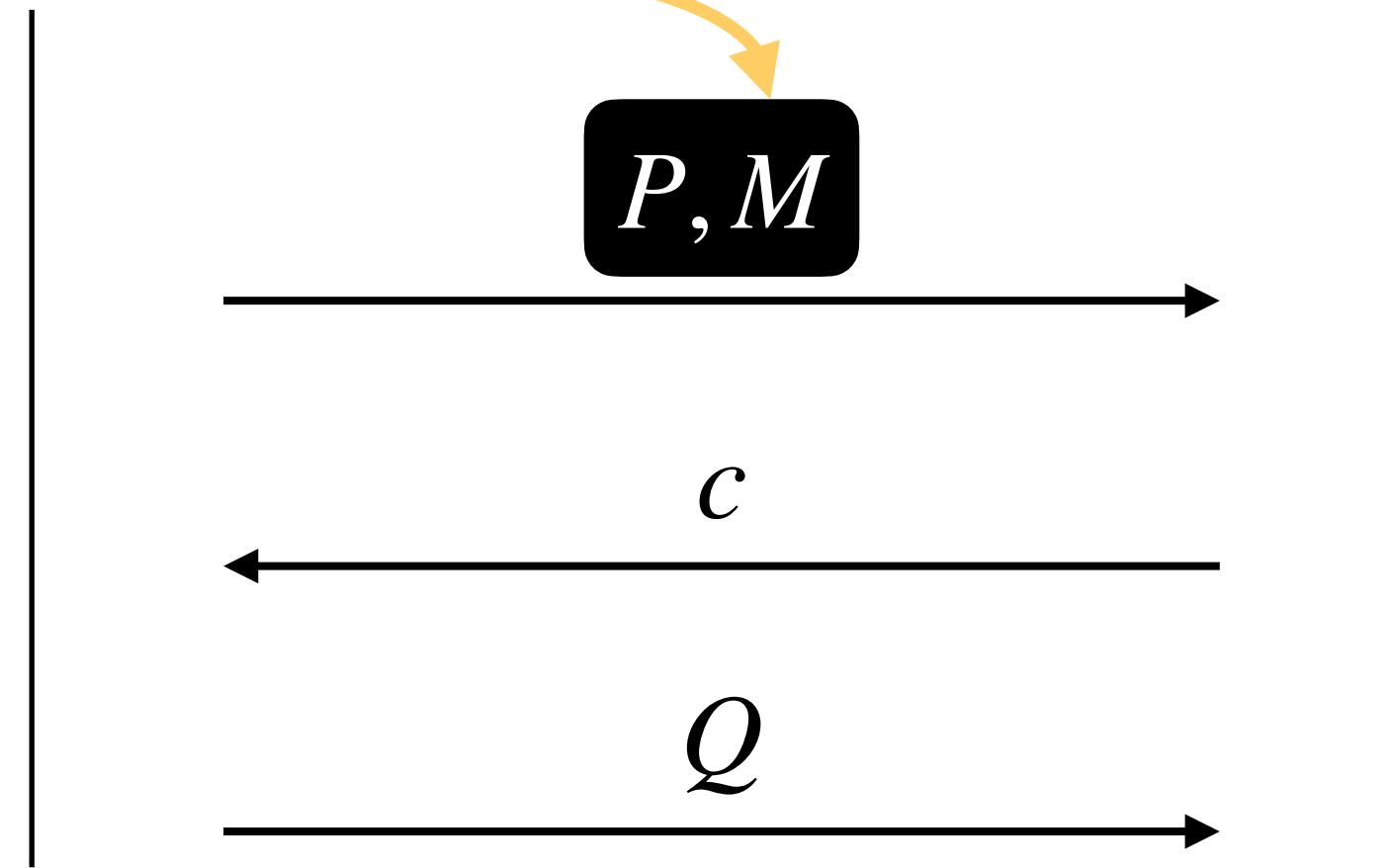
Random masking
polynomial M

$$\text{s.t. } \sum_{\omega \in \Omega} M(\omega) = 0$$



f_1, \dots, f_{m_1} p.p. constraints

f'_1, \dots, f'_{m_2} a.p.p. constraints



$$Q(X) = M(X) + \sum_i \Gamma_i(X) \cdot f_i(P(X), c_i(X)) + \sum_i \gamma_i \cdot f'_i(P(X), c'_i(X))$$

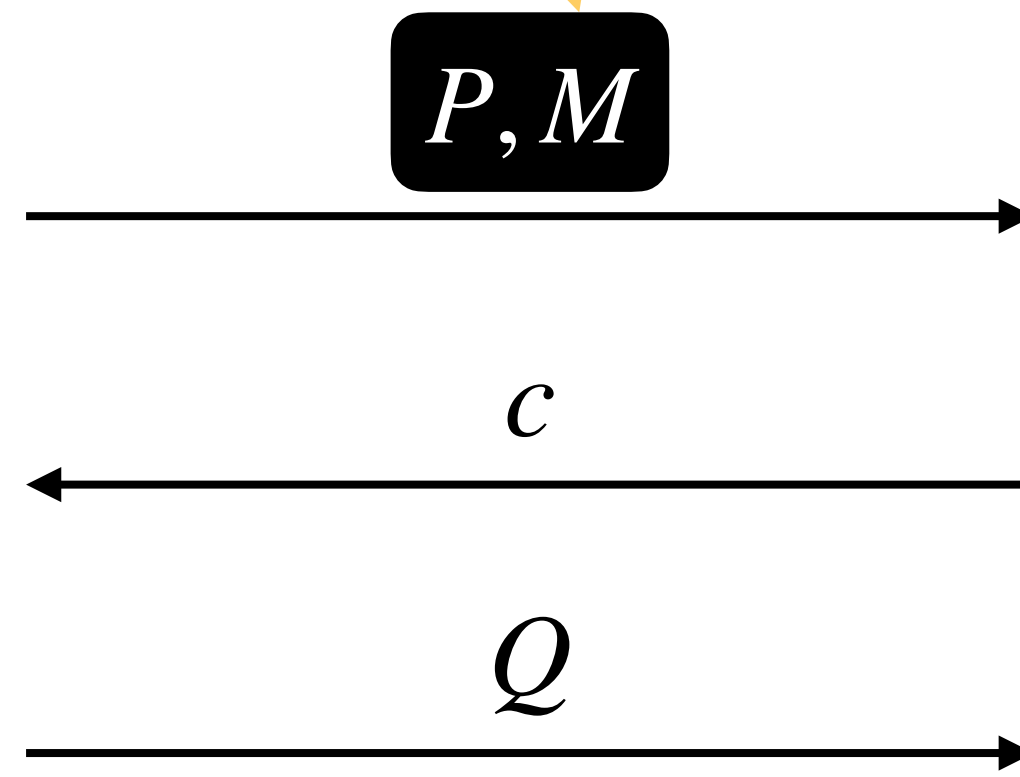
PACS PIOP

Random masking
polynomial M
s.t. $\sum_{\omega \in \Omega} M(\omega) = 0$



f_1, \dots, f_{m_1} p.p. constraints

f'_1, \dots, f'_{m_2} a.p.p. constraints

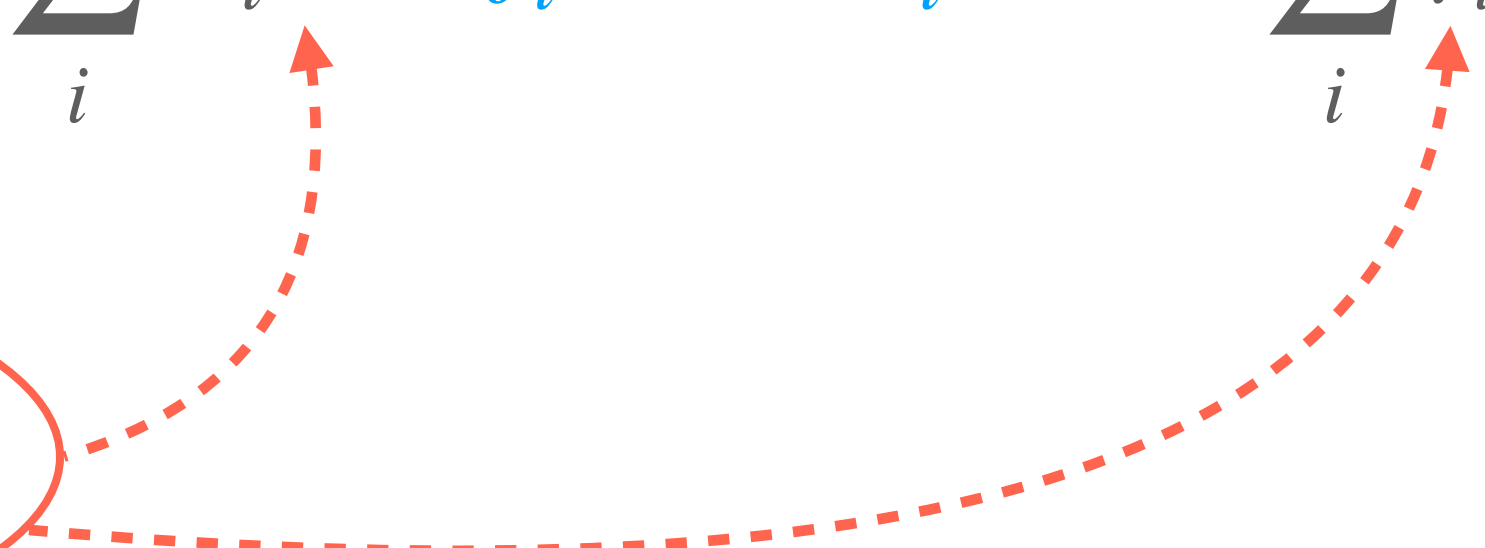


challenge

$c = \{\Gamma_i(X)\}, \{\gamma_i\}$

$$Q(X) = M(X) + \sum_i \Gamma_i(X) \cdot f_i(P(X), c_i(X)) + \sum_i \gamma_i \cdot f'_i(P(X), c'_i(X))$$

Batching (aggregated)
p.p. constraints



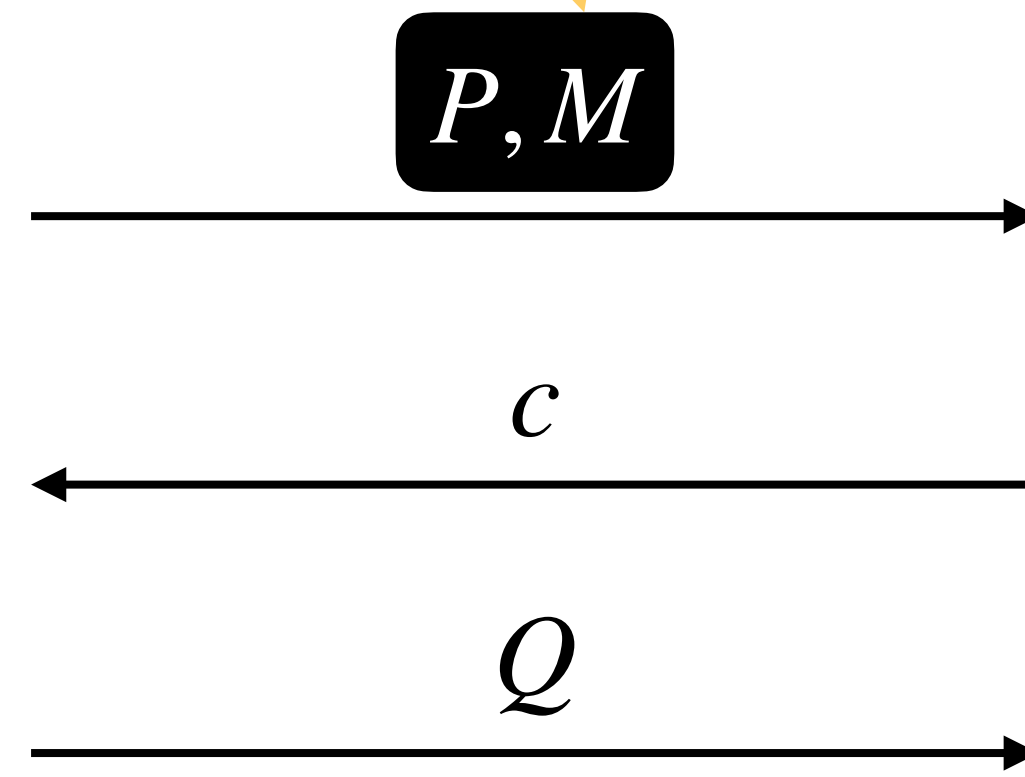
PACS PIOP

Random masking
polynomial M
s.t. $\sum_{\omega \in \Omega} M(\omega) = 0$



f_1, \dots, f_{m_1} p.p. constraints

f'_1, \dots, f'_{m_2} a.p.p. constraints



challenge

$c = \{\Gamma_i(X)\}, \{\gamma_i\}$

$$Q(X) = M(X) + \sum_i \Gamma_i(X) \cdot \underbrace{f_i(P(X), c_i(X))}_{= 0 \ \forall x \in \Omega} + \sum_i \gamma_i \cdot \underbrace{f'_i(P(X), c'_i(X))}_{\sum_{\Omega} = 0}$$

Batching (aggregated)
p.p. constraints

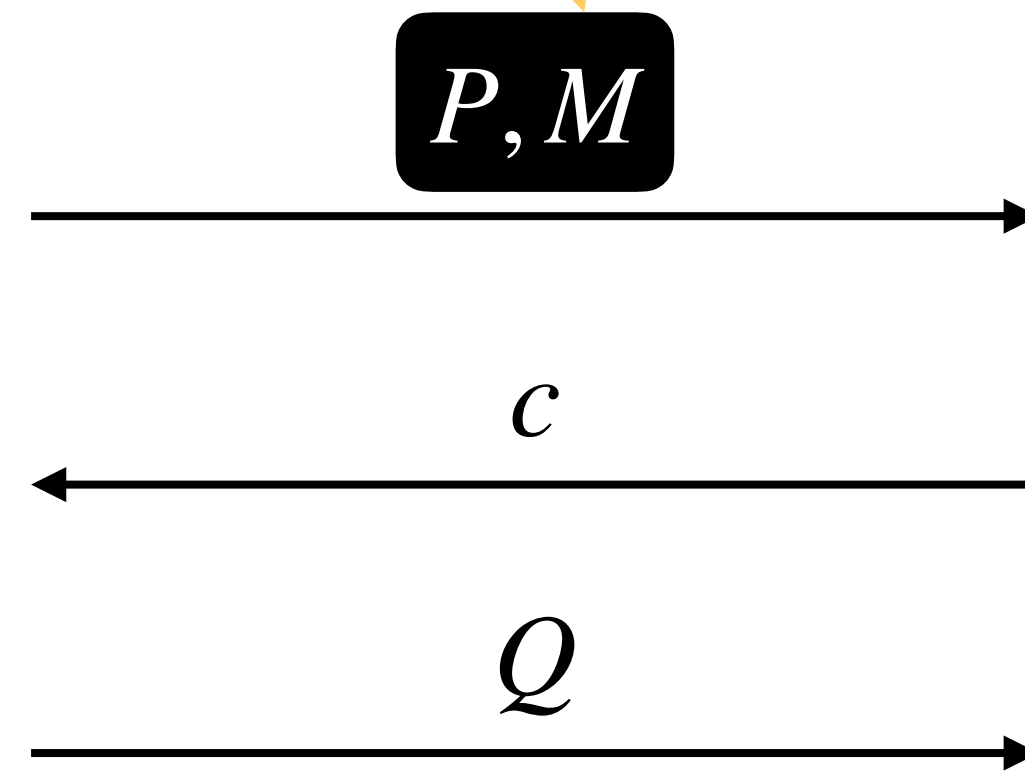
PACS PIOP

Random masking
polynomial M
s.t. $\sum_{\omega \in \Omega} M(\omega) = 0$



f_1, \dots, f_{m_1} p.p. constraints

f'_1, \dots, f'_{m_2} a.p.p. constraints



challenge

$c = \{\Gamma_i(X)\}, \{\gamma_i\}$

$$Q(X) = M(X) + \sum_i \Gamma_i(X) \cdot \underbrace{f_i(P(X), c_i(X))}_{= 0 \forall x \in \Omega} + \sum_i \gamma_i \cdot \underbrace{f'_i(P(X), c'_i(X))}_{\sum_{\Omega} = 0}$$

$$\implies \sum_{\omega \in \Omega} Q(\omega) = 0$$

Batching (aggregated)
p.p. constraints

Example: Ligeró's Arithmetization

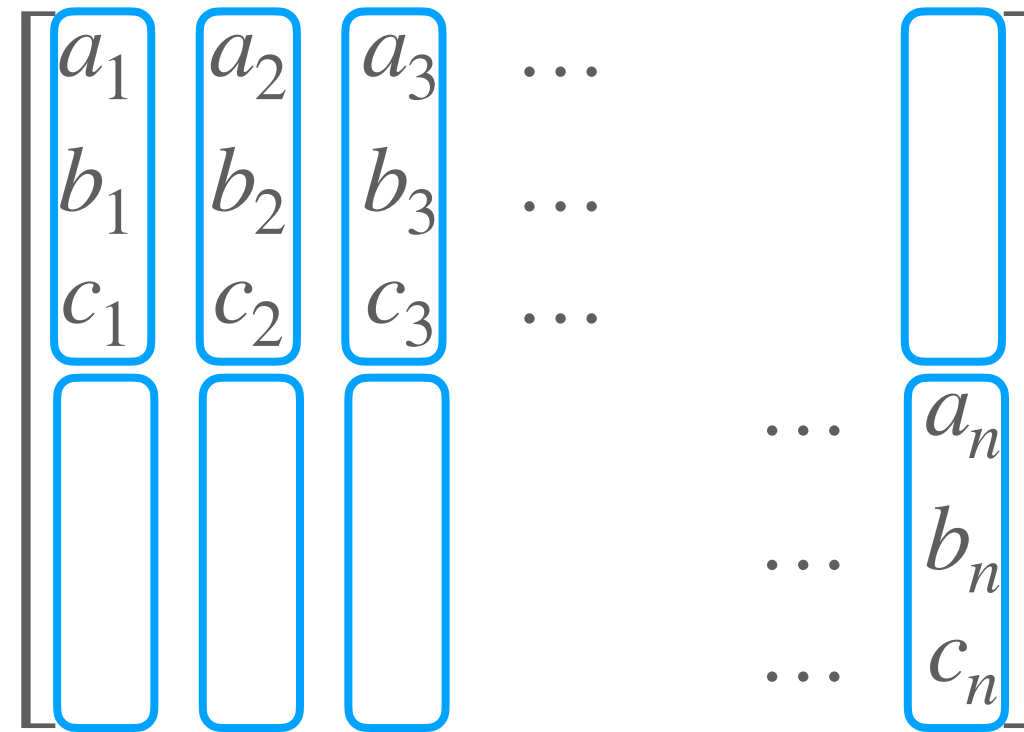
Checking a circuit $C(x, w) = 1$

Let $\{(a_i, b_i, c_i)\}$ all the multiplications
in the computation s.t. $a_i \cdot b_i = c_i$

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & & & \\ b_1 & b_2 & b_3 & \dots & & & \\ c_1 & c_2 & c_3 & \dots & & & \\ & & & & \dots & a_n & \\ & & & & \dots & b_n & \\ & & & & \dots & c_n & \end{bmatrix}$$

Example: Ligeró's Arithmetization

Checking a circuit $C(x, w) = 1$
Let $\{(a_i, b_i, c_i)\}$ all the multiplications
in the computation s.t. $a_i \cdot b_i = c_i$



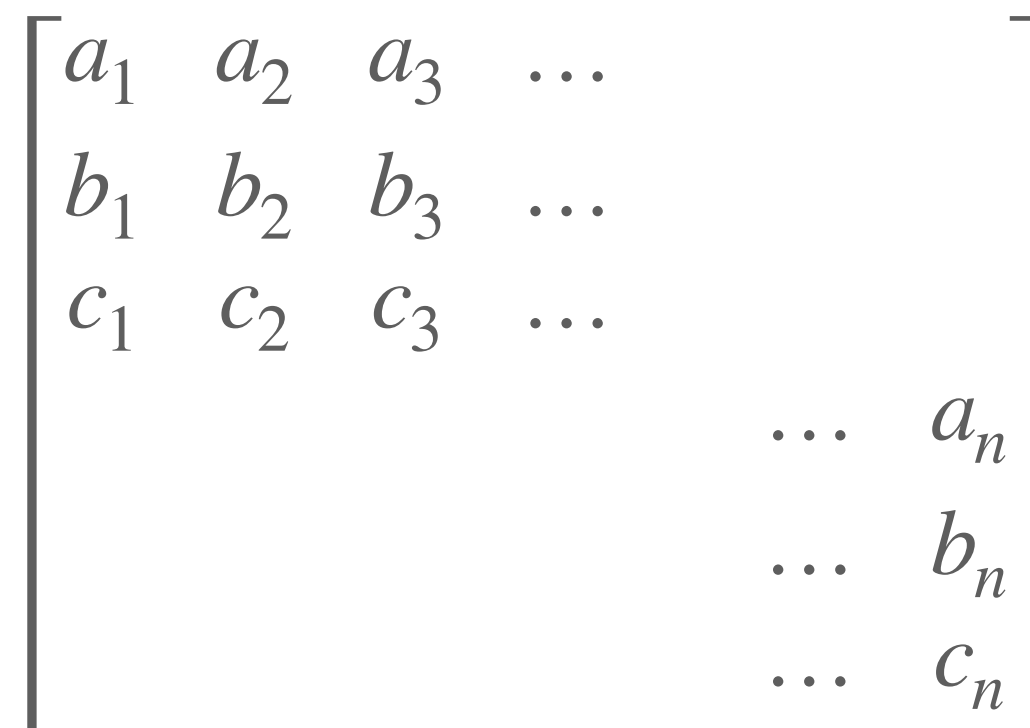
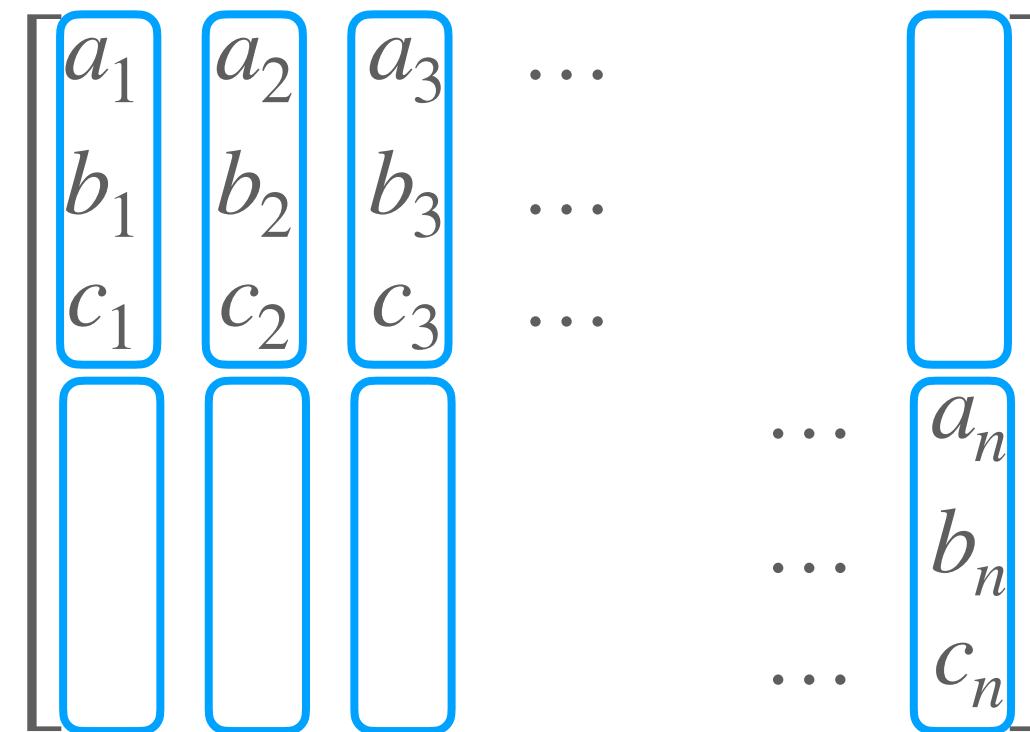
Checking multiplications

$$f: (a, b, c) \mapsto a \cdot b - c$$

Parallel polynomial constraints

Example: Ligeró's Arithmetization

Checking a circuit $C(x, w) = 1$
 Let $\{(a_i, b_i, c_i)\}$ all the multiplications
 in the computation s.t. $a_i \cdot b_i = c_i$



Checking multiplications

$$f : (a, b, c) \mapsto a \cdot b - c$$

Parallel polynomial constraints

Checking wiring

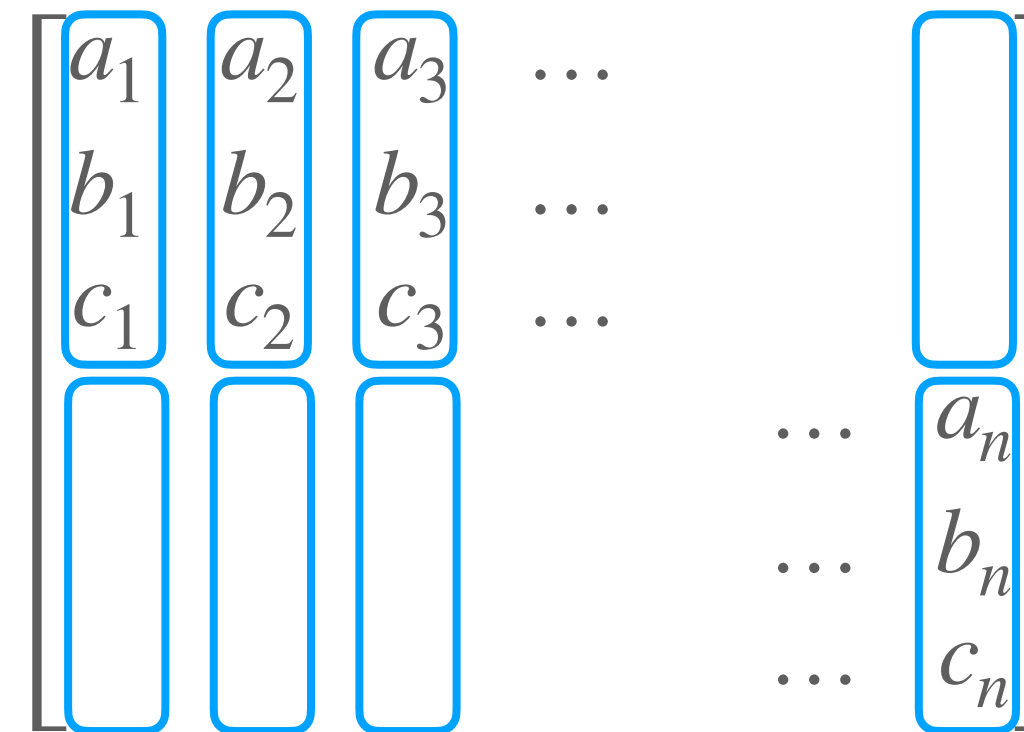
$$a_i = \langle \alpha_i, in \parallel (c_1, \dots, c_n) \rangle$$

$$b_i = \langle \beta_i, in \parallel (c_1, \dots, c_n) \rangle$$

α_i, β_i constants

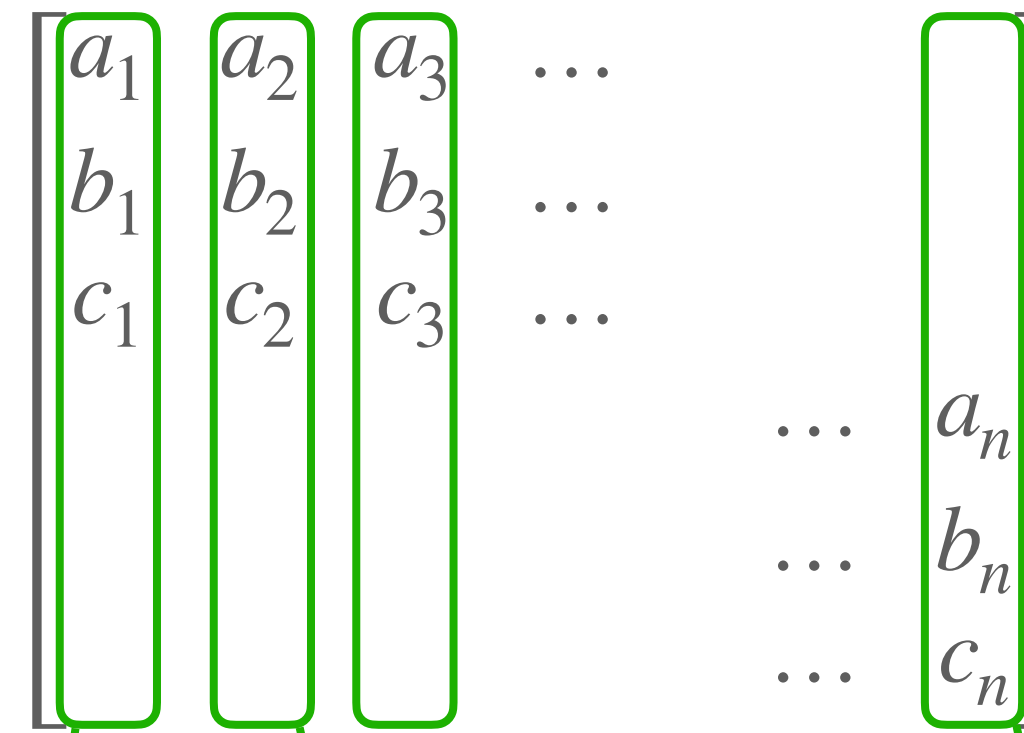
Example: Ligeró's Arithmetization

Checking a circuit $C(x, w) = 1$
 Let $\{(a_i, b_i, c_i)\}$ all the multiplications
 in the computation s.t. $a_i \cdot b_i = c_i$



Checking multiplications
 $f: (a, b, c) \mapsto a \cdot b - c$

Parallel polynomial constraints




Checking wiring
 $a_i = \langle \alpha_i, in \parallel (c_1, \dots, c_n) \rangle$
 $b_i = \langle \beta_i, in \parallel (c_1, \dots, c_n) \rangle$
 α_i, β_i constants

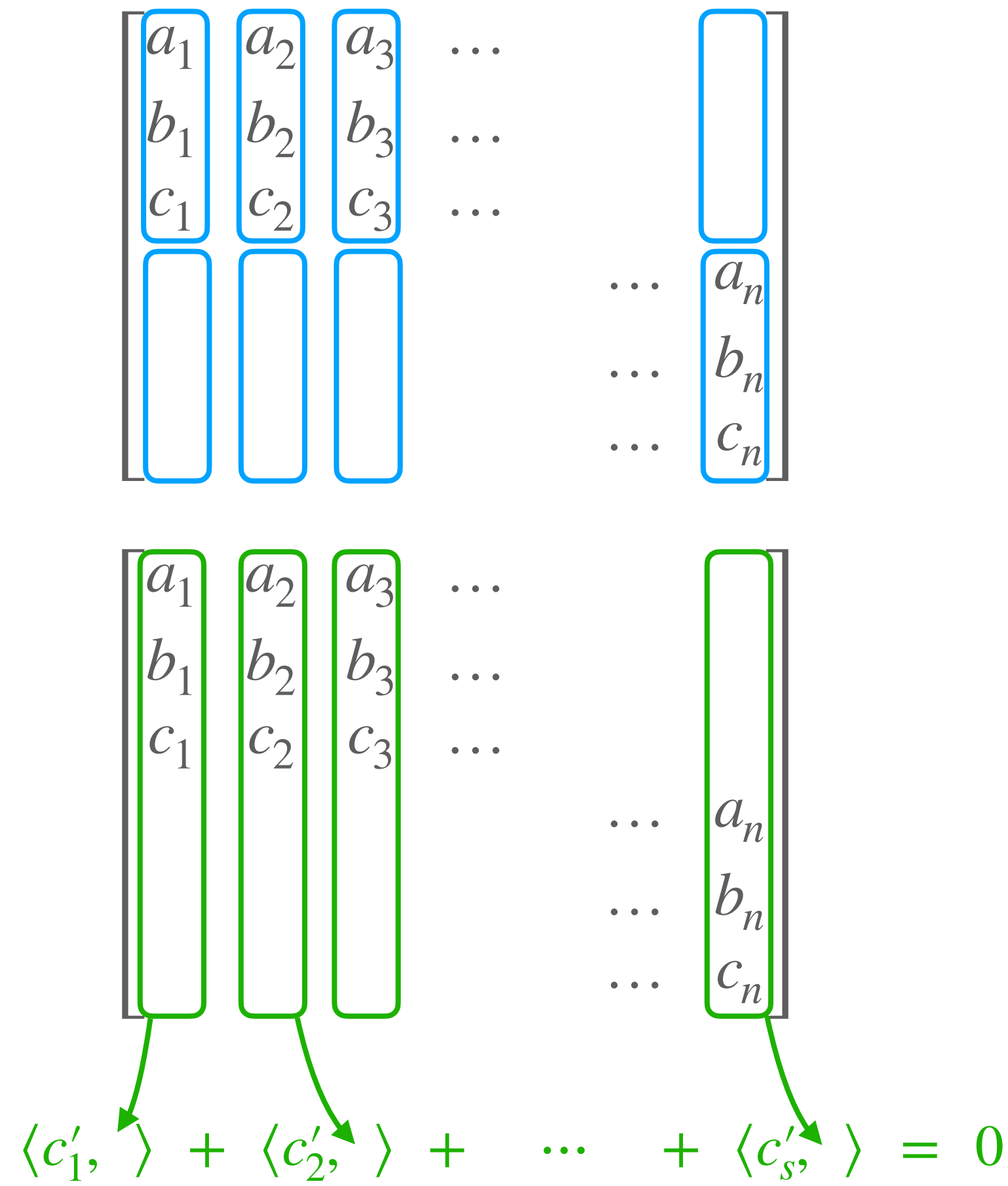
Aggregated p.p. constraints
 (of degree 1)

$$\langle c'_1, \rangle + \langle c'_2, \rangle + \dots + \langle c'_s, \rangle = 0$$

Example: Ligeró's Arithmetization

Checking a circuit $C(x, w) = 1$
 Let $\{(a_i, b_i, c_i)\}$ all the multiplications
 in the computation s.t. $a_i \cdot b_i = c_i$

 PACS enables a better
 arithmetization, saving
 33% of the extended witness



Checking multiplications

$$f: (a, b, c) \mapsto a \cdot b - c$$

Parallel polynomial constraints

Checking wiring

$$a_i = \langle \alpha_i, in \parallel (c_1, \dots, c_n) \rangle$$


$$b_i = \langle \beta_i, in \parallel (c_1, \dots, c_n) \rangle$$


α_i, β_i constants

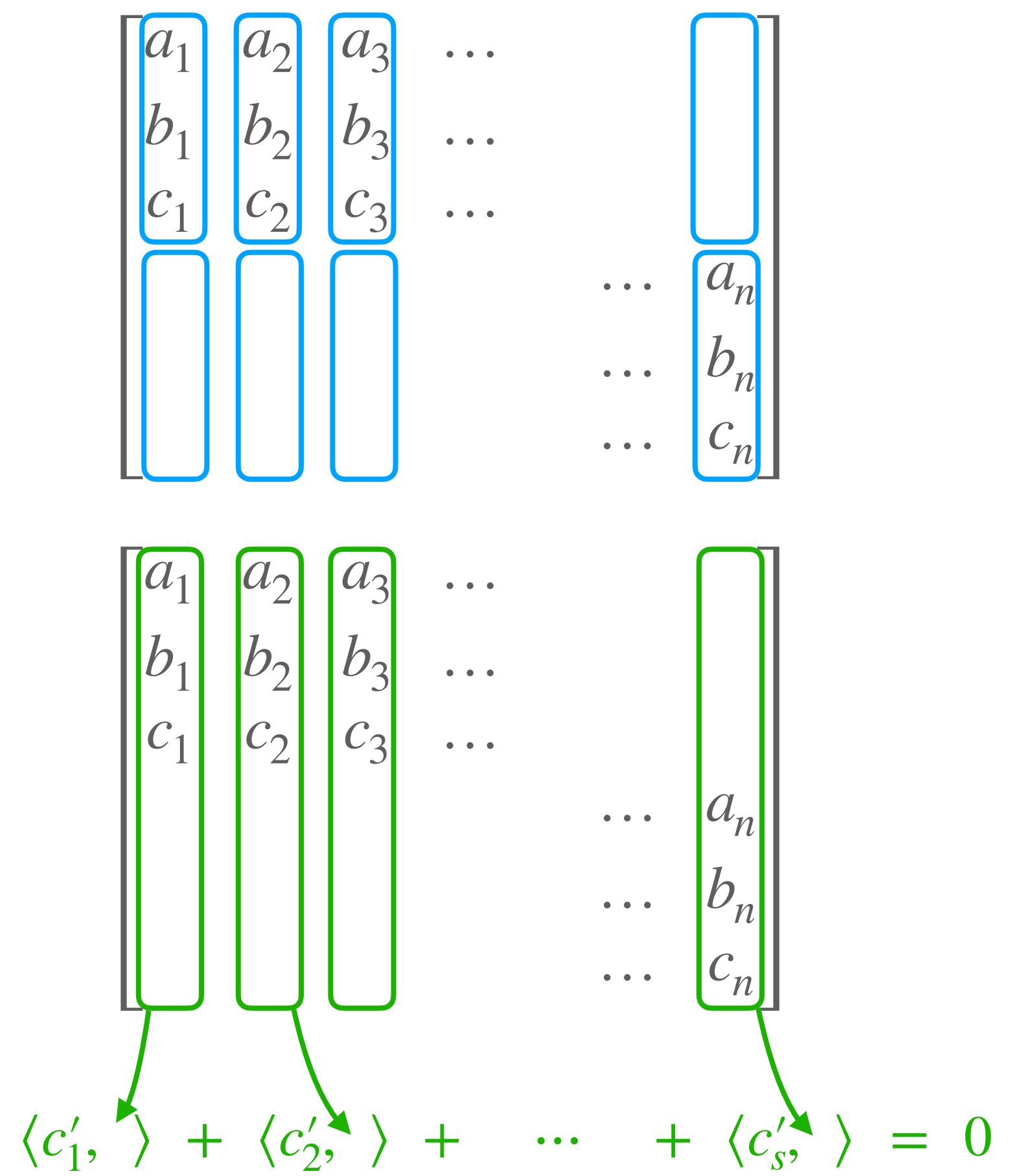
*Aggregated p.p. constraints
 (of degree 1)*

Example: Ligeró's Arithmetization

Checking a circuit $C(x, w) = 1$
 Let $\{(a_i, b_i, c_i)\}$ all the multiplications
 in the computation s.t. $a_i \cdot b_i = c_i$

 PACS enables a better arithmetization, saving 33% of the extended witness

 PACS enables gates of higher degree & of various fan-in / fan-out



Checking multiplications
 $f: (a, b, c) \mapsto a \cdot b - c$
 Parallel polynomial constraints

Checking wiring
 $a_i = \langle \alpha_i, in \parallel (c_1, \dots, c_n) \rangle$
 $b_i = \langle \beta_i, in \parallel (c_1, \dots, c_n) \rangle$
 α_i, β_i constants
 Aggregated p.p. constraints (of degree 1)

SmallWood PCS

Small-domain PCS

- Commit $P(X) \in (\mathbb{F}[X])^n$
- Prove $P(e) = p_e$ for $e \in \mathbb{E} \subseteq \mathbb{F}$

Linear-map Vector
Commitment Scheme (LVCS)

- Commit $\vec{r}_1, \dots, \vec{r}_n \in \mathbb{F}^m$
- Prove $\sum_i c_i \cdot \vec{r}_i = \vec{v}$ for $c_1, \dots, c_n \in \mathbb{F}$

(Full-domain) PCS

- Commit $P(X) \in (\mathbb{F}[X])^n$
- Prove $P(e) = p_e$ for $e \in \mathbb{F}$

SmallWood PCS

Small-domain PCS

“Degree-enforcing CS” from TCitH [FR25],
variant of Ligerio [AHIV17,AHIV23]
Commit polynomial evaluations in a Merkle tree
+ degree enforced in the commitment

SmallWood PCS

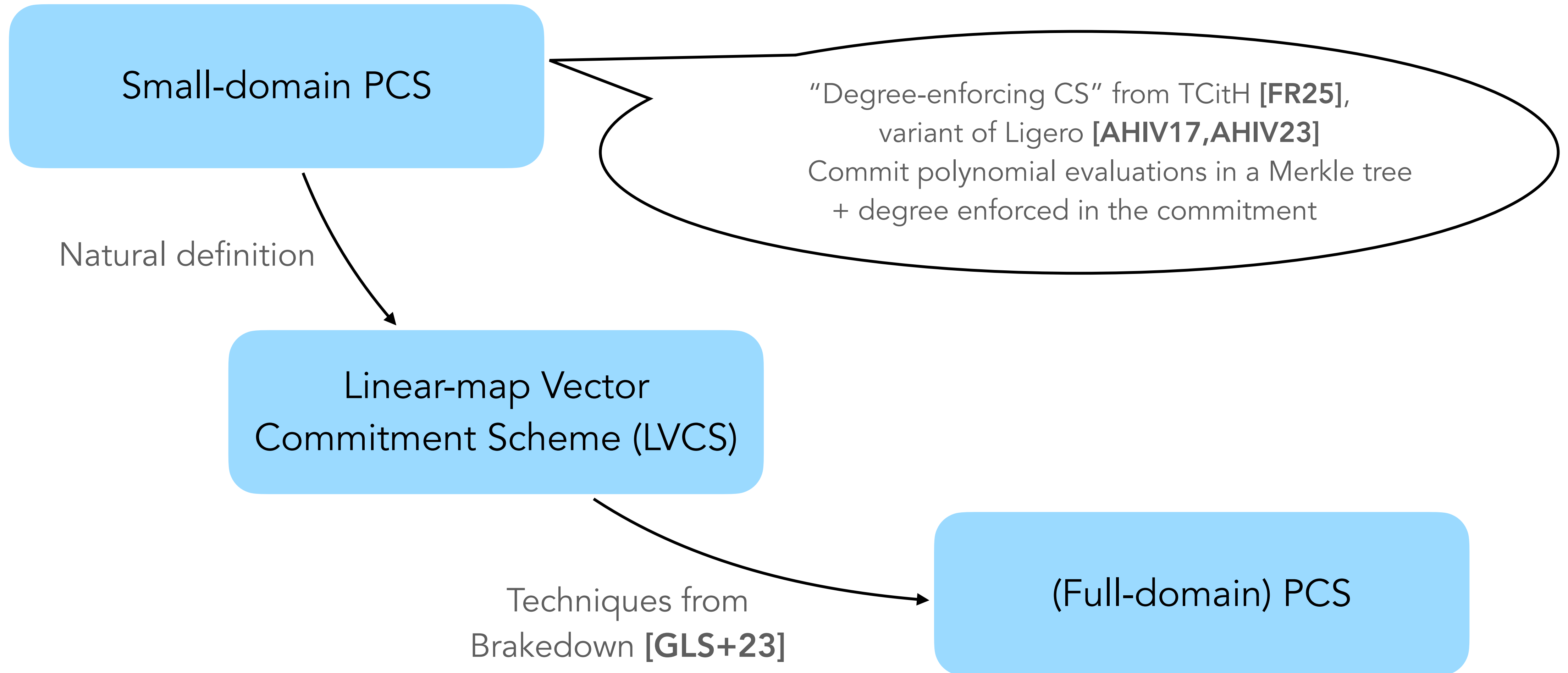
Small-domain PCS

“Degree-enforcing CS” from TCitH [FR25],
variant of Ligerio [AHIV17,AHIV23]
Commit polynomial evaluations in a Merkle tree
+ degree enforced in the commitment

Natural definition

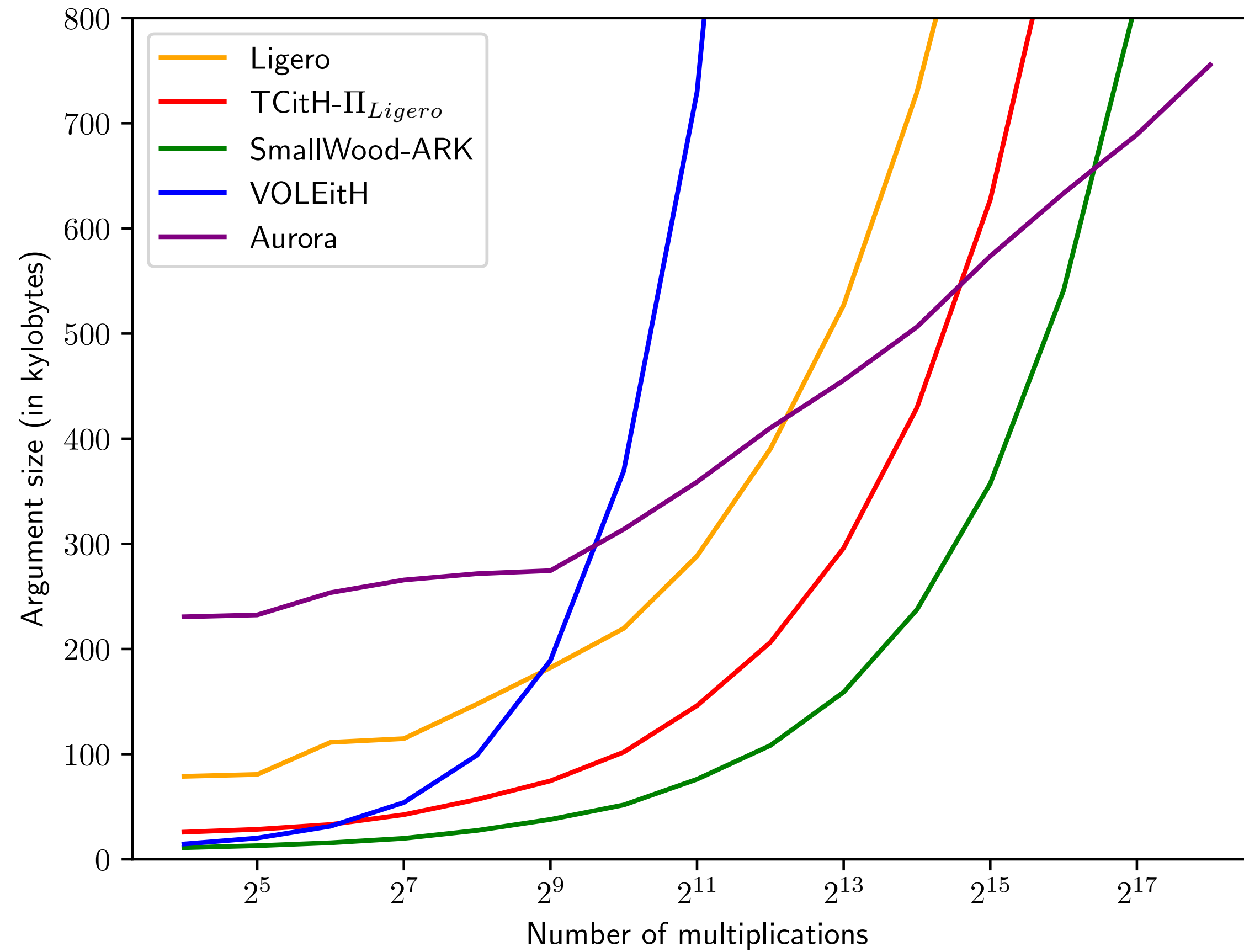
Linear-map Vector
Commitment Scheme (LVCS)

SmallWood PCS



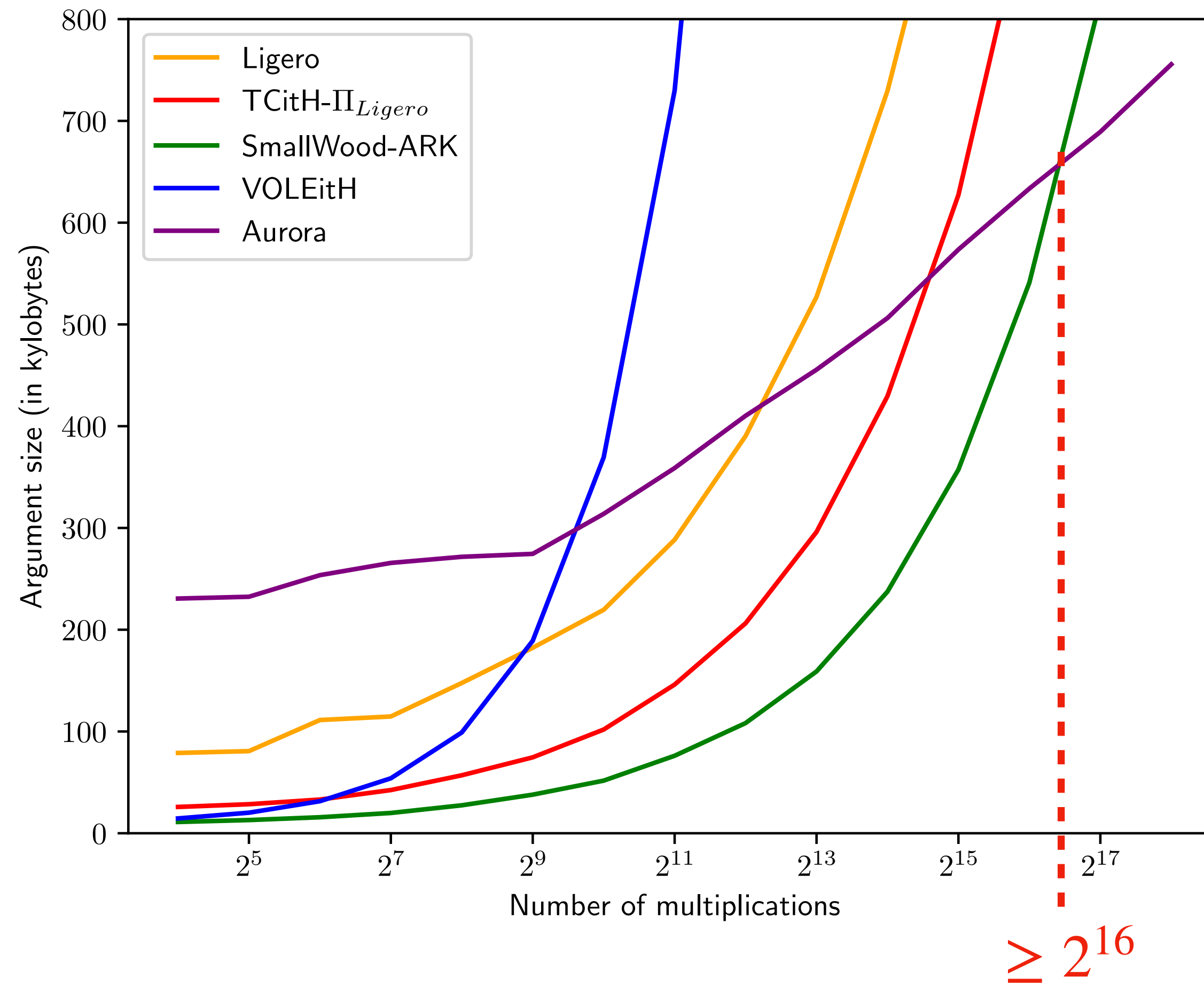
Application to Arithmetic Circuits

256-bit field



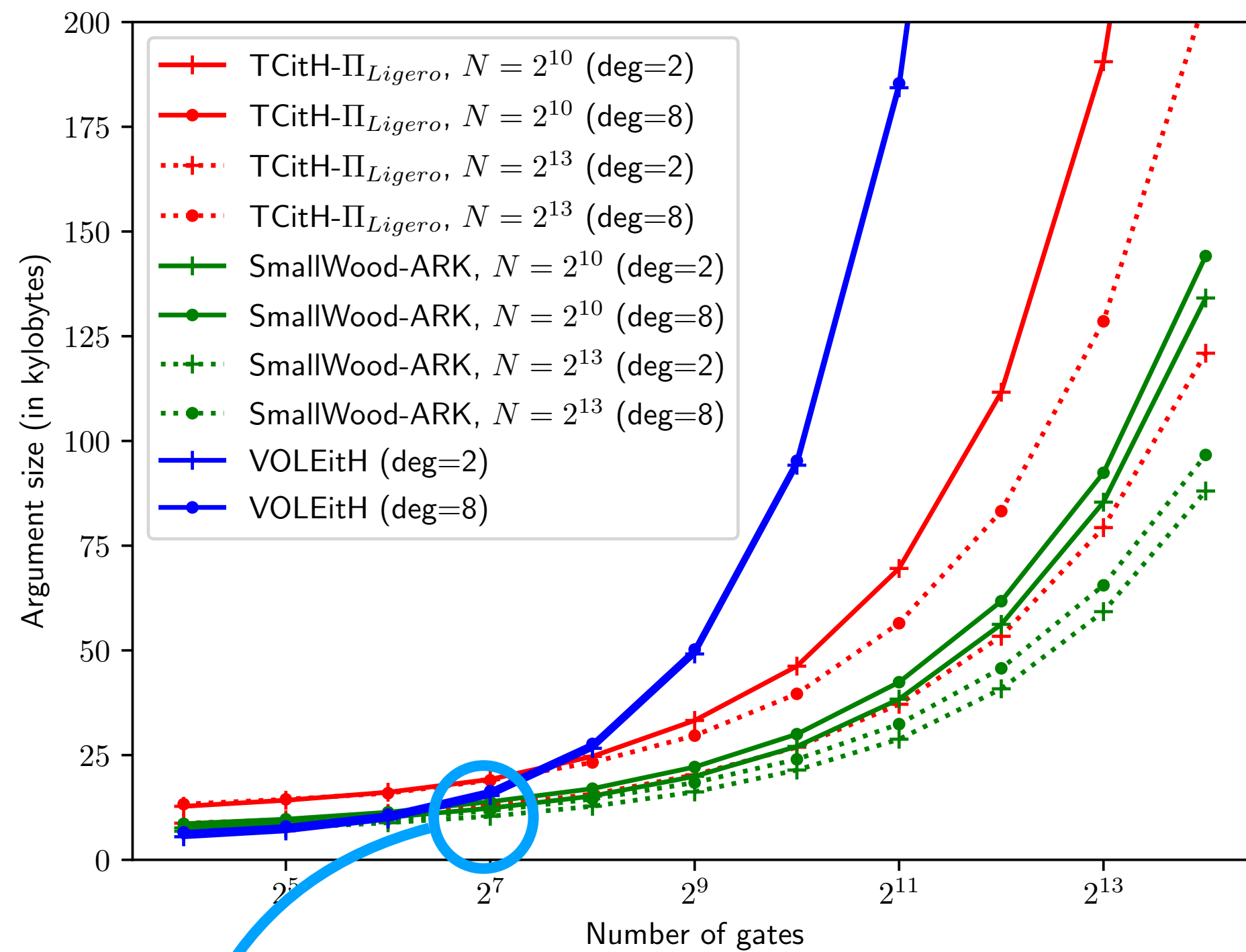
Application to Arithmetic Circuits

256-bit field



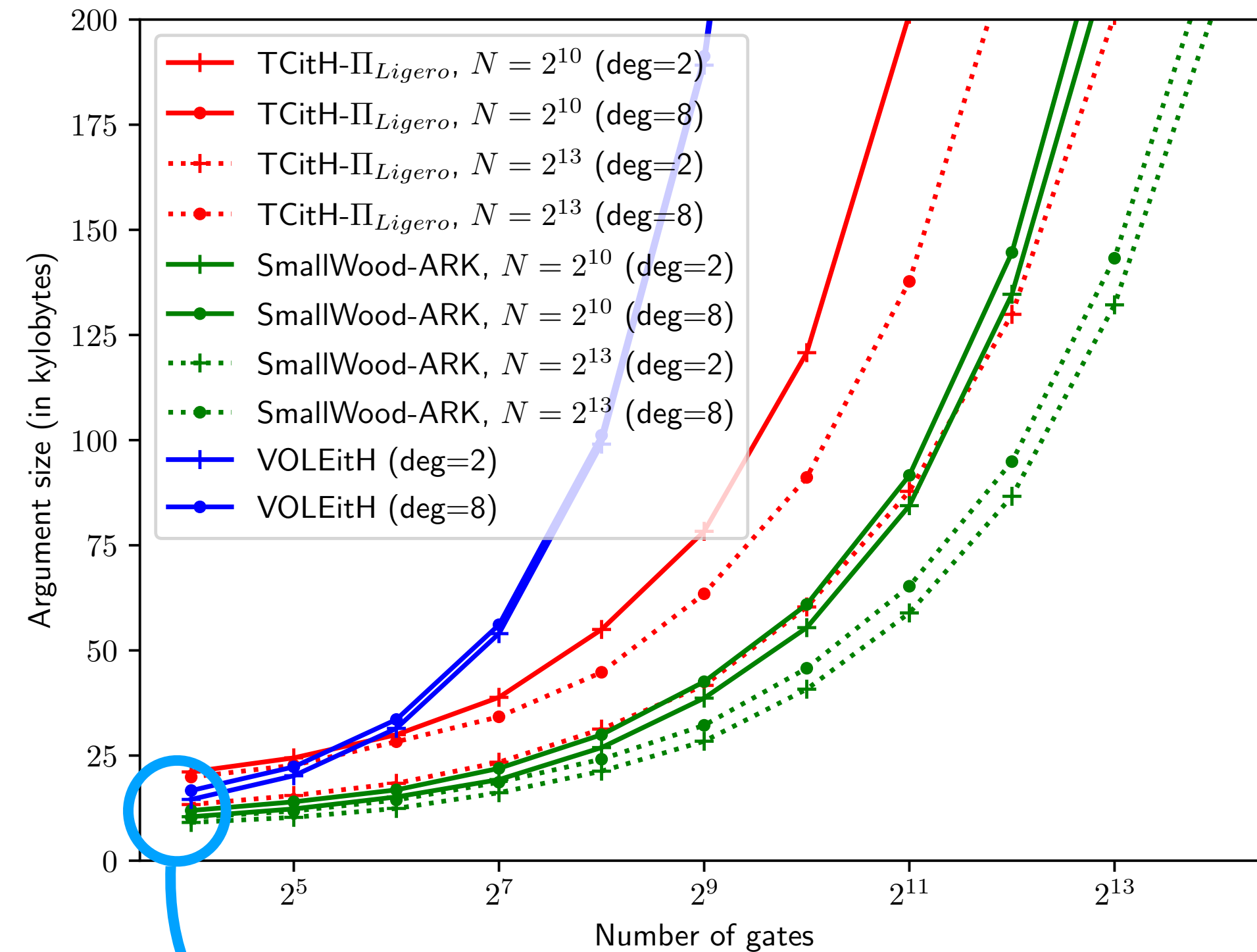
Application to Arithmetic Circuits

64-bit field



Smaller than VOLEitH
for $\#gates \geq 2^7$

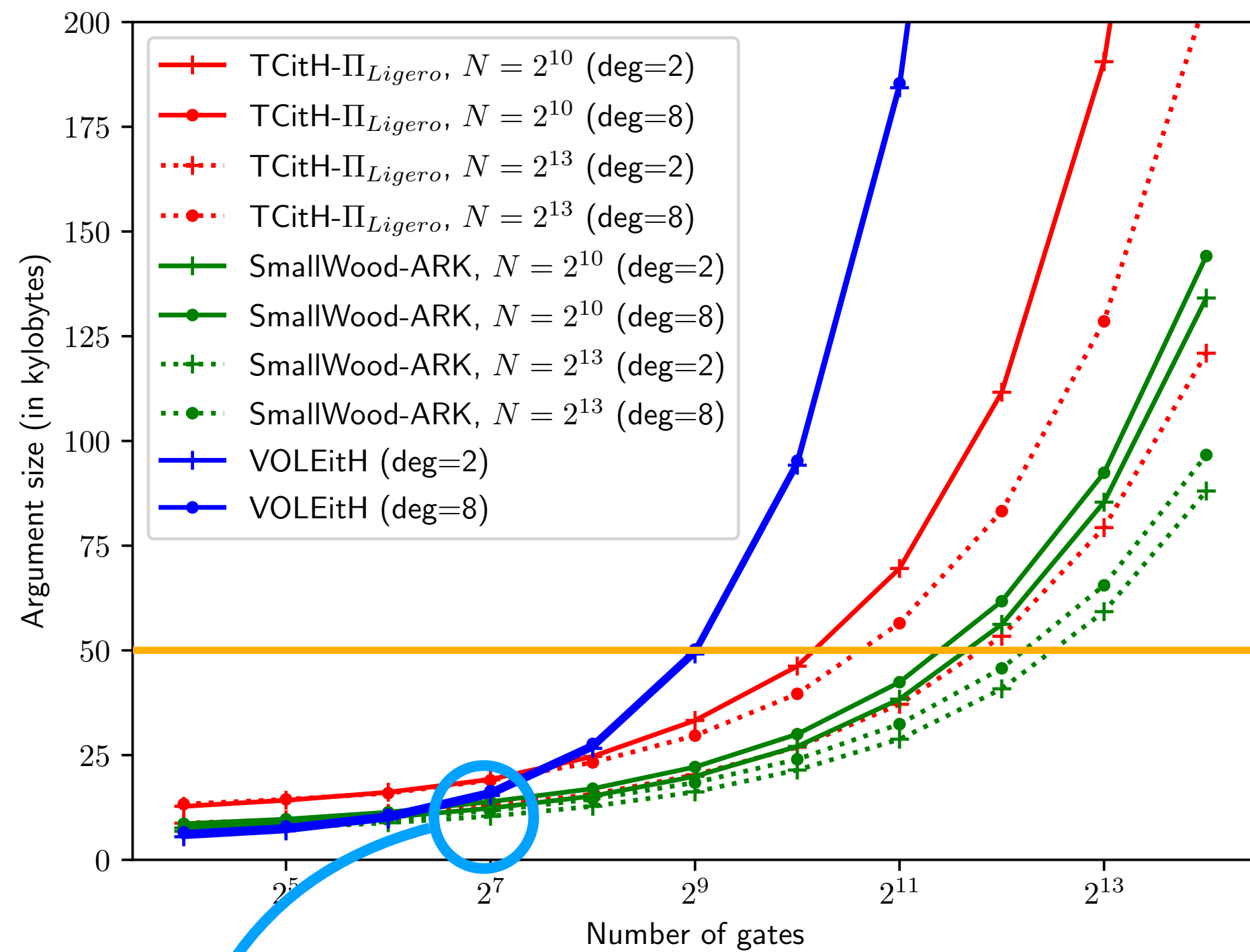
256-bit field



Smaller than VOLEitH
for $\#gates \geq 2^4$

Application to Arithmetic Circuits

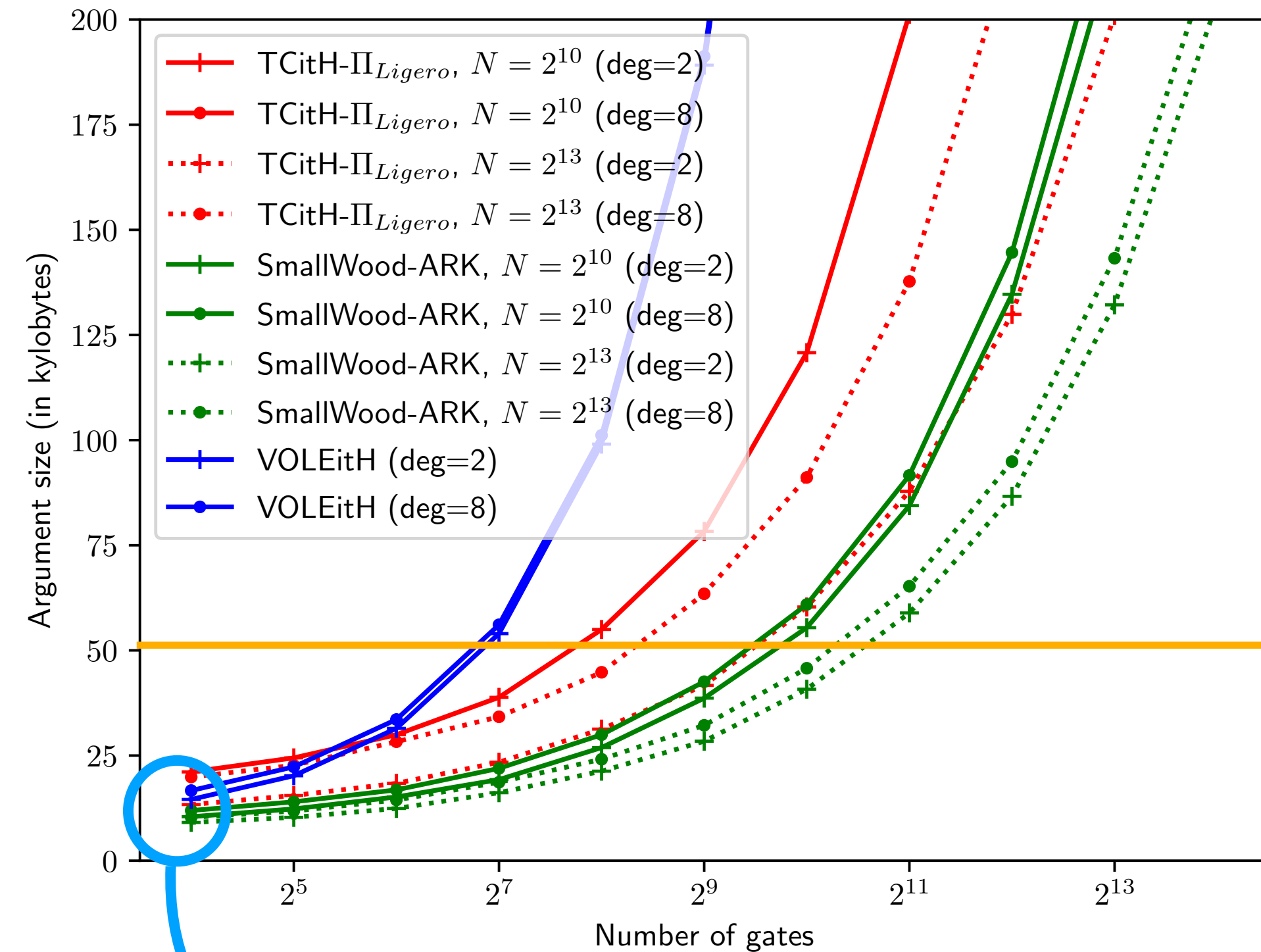
64-bit field



Smaller than VOLEitH
for #gates $\geq 2^7$

4000 gates $\rightarrow \leq 50$ KB

256-bit field

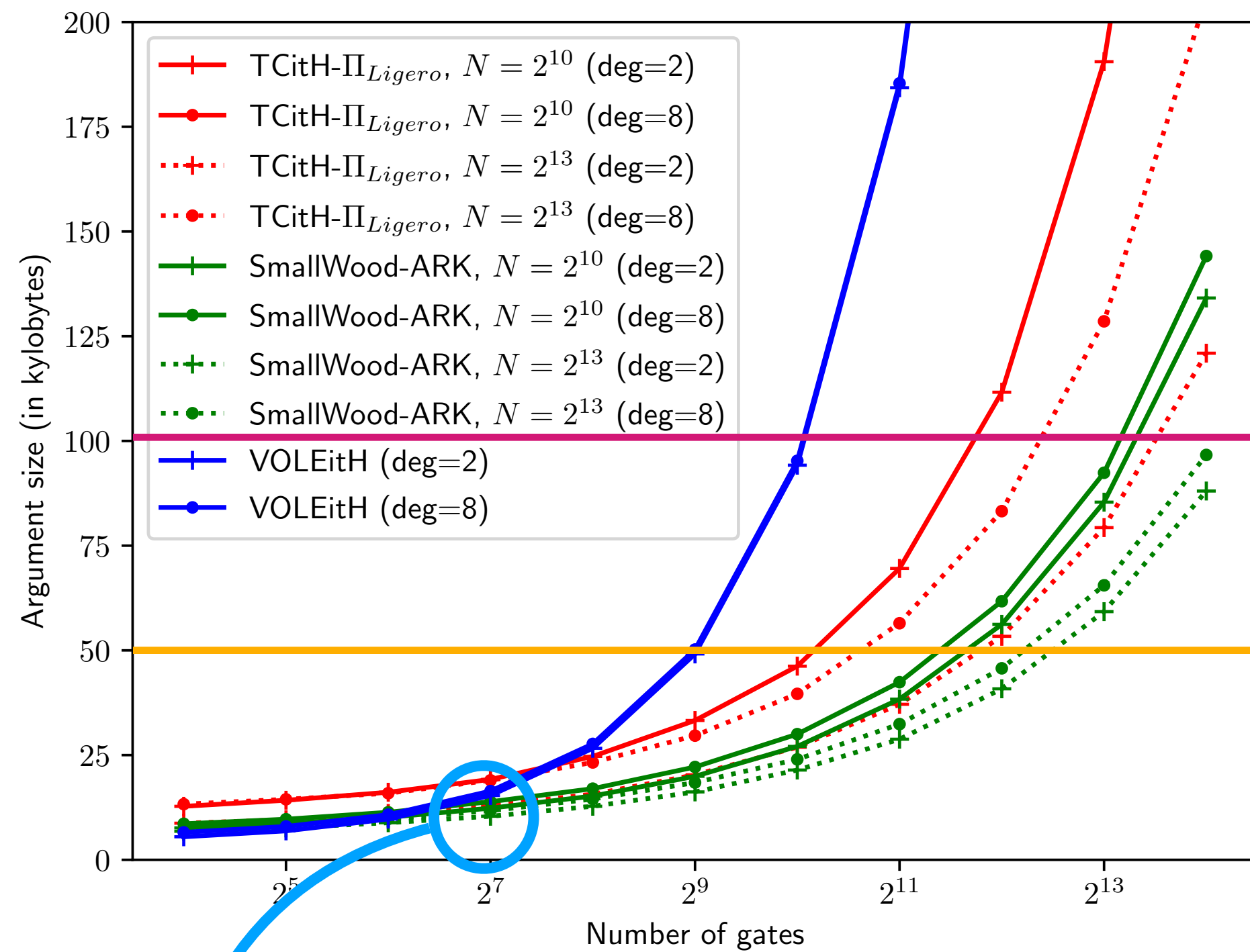


Smaller than VOLEitH
for #gates $\geq 2^4$

1000 gates $\rightarrow \leq 50$ KB

Application to Arithmetic Circuits

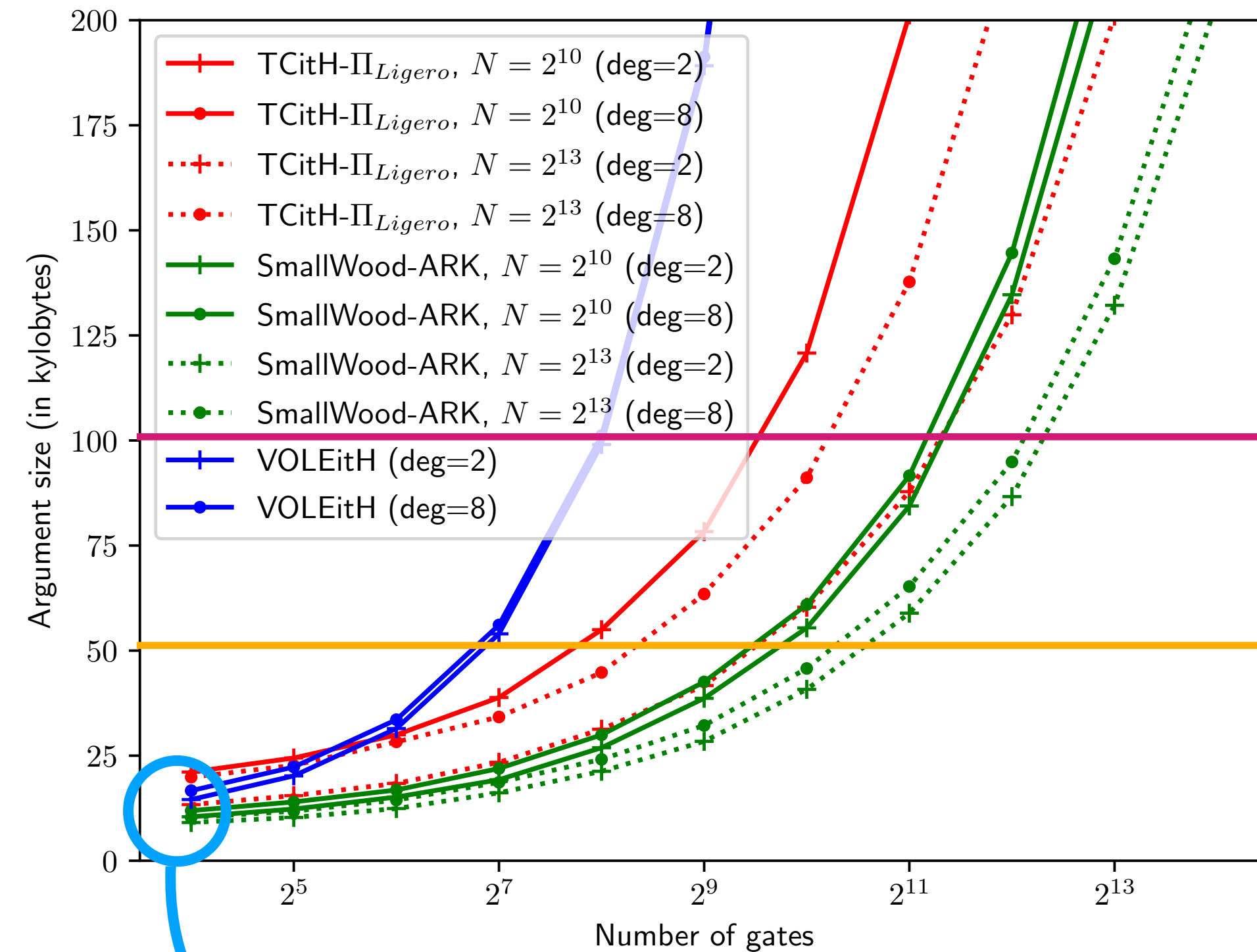
64-bit field



Smaller than VOLEitH
for #gates $\geq 2^7$

4000 gates $\rightarrow \leq 50$ KB
10 000 gates $\rightarrow \leq 100$ KB

256-bit field



Smaller than VOLEitH
for #gates $\geq 2^4$

1000 gates $\rightarrow \leq 50$ KB
4000 gates $\rightarrow \leq 100$ KB

Application to Lattices

LWE Instance	LWE Parameters				SmallWood-ARK	Comparison	
	q	n	m	β		TCitH-MT	VOLEitH
Toy Example 1	$\approx 2^{32}$	2048	512	1	16 977 B	38 081 B	39 072 B [*]
Toy Example 2	$\approx 2^{61}$	4096	1024	1/2	21 356 B	57 409 B	51 470 B [*]
Kyber512	3329	2×256	2×256	3	14 115 B	21 185 B	11 754 B
Kyber768	3329	3×256	3×256	2	15 004 B	24 938 B	15 519 B
Kyber1024	3329	4×256	4×256	2	16 455 B	28 241 B	19 637 B
Dilithium2	8380417	4×256	4×256	2	17 514 B	40 100 B	36 241 B
Dilithium3	8380417	5×256	6×256	4	22 076 B	57 526 B	45 727 B
Dilithium5	8380417	7×256	8×256	2	22 700 B	58 088 B	60 527 B
Subset-Sum	$\approx 2^{256}$	256	1	1/2	12 557 B	-	13 484 B [*]

Application to Lattices

LWE Instance	LWE Parameters				SmallWood-ARK	Comparison	
	q	n	m	β		TCitH-MT	VOLEitH
Toy Example 1	$\approx 2^{32}$	2048	512	1	16 977 B	38 081 B	39 072 B [*]
Toy Example 2	$\approx 2^{61}$	4096	1024	1/2	21 356 B	57 409 B	51 470 B [*]
Kyber512	3329	2×256	2×256	3	14 115 B	21 185 B	11 754 B
Kyber768	3329	3×256	3×256	2	15 004 B	24 938 B	15 519 B
Kyber1024	3329	4×256	4×256	2	16 455 B	28 241 B	19 637 B
Dilithium2	8380417	4×256	4×256	2	17 514 B	40 100 B	36 241 B
Dilithium3	8380417	5×256	6×256	4	22 076 B	57 526 B	45 727 B
Dilithium5	8380417	7×256	8×256	2	22 700 B	58 088 B	60 527 B
Subset-Sum	$\approx 2^{256}$	256	1	1/2	12 557 B	-	13 484 B [*]

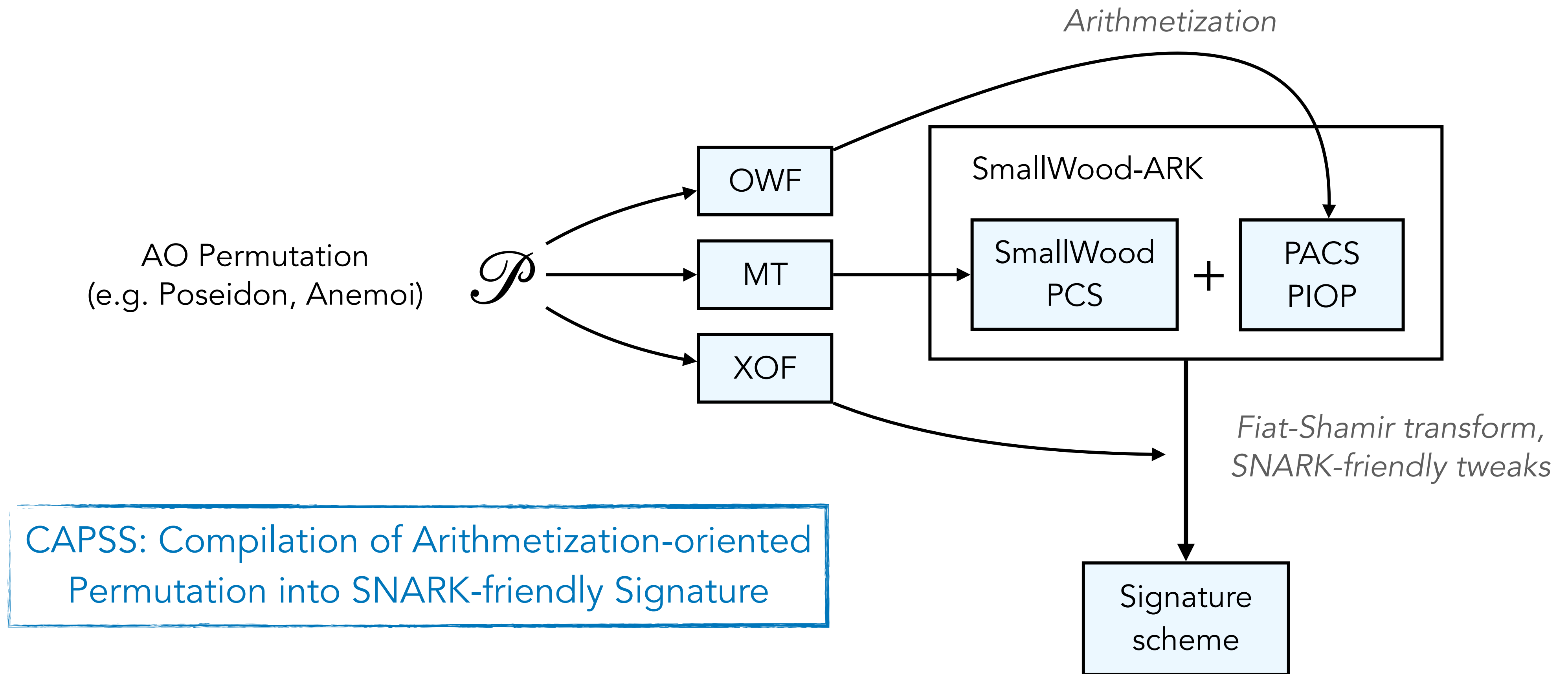
Application to Lattices

LWE Instance	LWE Parameters				SmallWood-ARK	Comparison	
	q	n	m	β		TCitH-MT	VOLEitH
Toy Example 1	$\approx 2^{32}$	2048	512	1	16 977 B	38 081 B	39 072 B [*]
Toy Example 2	$\approx 2^{61}$	4096	1024	1/2	21 356 B	57 409 B	51 470 B [*]
Kyber512	3329	2×256	2×256	3	14 115 B	21 185 B	11 754 B
Kyber768	3329	3×256	3×256	2	15 004 B	24 938 B	15 519 B
Kyber1024	3329	4×256	4×256	2	16 455 B	28 241 B	19 637 B
Dilithium2	8380417	4×256	4×256	2	17 514 B	40 100 B	36 241 B
Dilithium3	8380417	5×256	6×256	4	22 076 B	57 526 B	45 727 B
Dilithium5	8380417	7×256	8×256	2	22 700 B	58 088 B	60 527 B
Subset-Sum	$\approx 2^{256}$	256	1	1/2	12 557 B	-	13 484 B [*]

Application to Lattices

LWE Instance	LWE Parameters				SmallWood-ARK	Comparison	
	q	n	m	β		TCitH-MT	VOLEitH
Toy Example 1	$\approx 2^{32}$	2048	512	1	16 977 B	38 081 B	39 072 B [*]
Toy Example 2	$\approx 2^{61}$	4096	1024	1/2	21 356 B	57 409 B	51 470 B [*]
Kyber512	3329	2×256	2×256	3	14 115 B	21 185 B	11 754 B
Kyber768	3329	3×256	3×256	2	15 004 B	24 938 B	15 519 B
Kyber1024	3329	4×256	4×256	2	16 455 B	28 241 B	19 637 B
Dilithium2	8380417	4×256	4×256	2	17 514 B	40 100 B	36 241 B
Dilithium3	8380417	5×256	6×256	4	22 076 B	57 526 B	45 727 B
Dilithium5	8380417	7×256	8×256	2	22 700 B	58 088 B	60 527 B
Subset-Sum	$\approx 2^{256}$	256	1	1/2	12 557 B	-	13 484 B [*]

CAPSS Framework



CAPSS Framework

Signature Scheme	Sig. Size	#R1CS	Signing Time	Verif. Time	Assumptions
SPHINCS ^s	8 KB	≈ 460 K	≈ 200 ms	≈ 1 ms	Hash
SPHINCS ^f	16 KB	≈ 1 400 K	≈ 14	≈ 2 ms	Hash
Picnic1	32 KB	≈ 3 500 K	≈ 1 – 2 ms	≈ 1 – 2 ms	LowMC + Hash
Picnic3	12 KB	≈ 21 600 K	≈ 5 ms	≈ 4 ms	LowMC + Hash
LegRoast	16 KB	≈ 1 100 K	≈ 3 ms	≈ 3 ms	Leg. PRF + Hash
Banquet	12 KB	≈ 11 800 K	≈ 40 ms	≈ 40 ms	AES + Hash
Rainer	8 KB	≈ 26 100 K	≈ 1 ms	≈ 1 ms	Rain + Hash
FAEST	≈ 4 – 5 KB	–	≈ 0.5 – 4 ms	≈ 0.5 – 4 ms	AES + Hash
Loquat-128 (Keccak) [ZSE ⁺ 24]	57 KB	–	5.0 s	0.2 s	Legendre PRF + Keccak
Loquat-128 (Griffin) [ZSE ⁺ 24]	57 KB	≈ 150 K	105 s	11 s	Legendre PRF + Griffin
Loquat [*] -128 (Keccak) [ZSE ⁺ 24]	114 KB	–	5.0 s	0.2 s	Legendre PRF + Keccak
Loquat [*] -128 (Griffin) [ZSE ⁺ 24]	114 KB	≈ 300 K	214 s	25 s	Legendre PRF + Griffin
RescuePrime + STARKs [AdSGK24]	80–100 KB	–	9–23 ms	1 ms	RescuePrime (+ Blake3)
RescuePrime + STARKs [AdSGK24]	80–100 KB	–	94–370 ms	21–27 ms	RescuePrime
CAPSS-Anemoi	9–16 KB	24 K – 36 K	400 ms – 6 s [*]	22–28 ms [*]	Anemoi
CAPSS-Griffin	10–16 KB	20 K – 32 K	200 ms – 6 s [*]	12–17 ms [*]	Griffin
CAPSS-Poseidon	16–26 KB	40 K – 71 K	100 ms – 3 s [*]	4–7 ms [*]	Poseidon
CAPSS-RescuePrime	11–17 KB	40 K – 64 K	500 ms – 14 s [*]	31–47 ms [*]	RescuePrime

👉 <https://www.cryptoexperts.com/capss/>

Conclusion

- SmallWood is a generic hash-based ZK-NARK (or ZKPoK)
- SmallWood provides “compact” proofs for “relatively small” instances
 - Few (dozen) KBs for extended witness of size up to $\sim 2^{16}$
- Plan for our submission
 - Improve the design / proof size
 - Provide efficient implementations
 - Make it easy to use
 - Investigate further applications (e.g. PoK of signature, PoK of FHE plaintext, ...)

References

- **[ADK24]** S. Atapoor, C. Delpéch de Saint Guilhem, A. Kindi. STARK-based signatures from the RPO permutation. 2024. <https://eprint.iacr.org/2024/1553.pdf>
- **[AHIV17]** S. Ames, C. Hazay, Y. Ishai, M. Venkatasubramanian. Ligerio: Lightweight sublinear arguments without a trusted setup. CCS 2017.
- **[AHIV23]** S. Ames, C. Hazay, Y. Ishai, M. Venkatasubramanian. Ligerio: Lightweight sublinear arguments without a trusted setup. Extended version. DCC 2023. <https://eprint.iacr.org/2022/1608.pdf>
- **[BCR+19]** E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, N. P. Ward. Aurora: Transparent Succinct Arguments for R1CS. EUROCRYPT 2019. <https://eprint.iacr.org/2018/828.pdf>
- **[FR23]** T. Feneuil, M. Rivain. Threshold Linear Secret Sharing to the Rescue of MPC-in-the-Head. <https://eprint.iacr.org/2022/1407.pdf>
- **[FR25]** T. Feneuil, M. Rivain. Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments. Journal of Cryptology 2025. <https://eprint.iacr.org/2023/1573.pdf>
- **[FR25b]** T. Feneuil, M. Rivain. SmallWood: Hash-Based Polynomial Commitments and Zero-Knowledge Arguments for Relatively Small Instances. <https://eprint.iacr.org/2025/1085.pdf>
- **[FR25c]** T. Feneuil, M. Rivain. CAPSS: A Framework for SNARK-Friendly Post-Quantum Signatures. 2025. <https://eprint.iacr.org/2025/061>
- **[GLS+23]** A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, R. S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. CRYPTO 2023. <https://eprint.iacr.org/2021/1043>
- **[ZSE+24]** X. Zhang, R. Steinfeld, M. F. Esgin, J. K. Liu, D. Liu, S. Ruj. Loquat: A SNARK-Friendly Post-Quantum Signature based on the Legendre PRF with Applications in Ring and Aggregate Signatures. CRYPTO 2024. <https://eprint.iacr.org/2024/868.pdf>