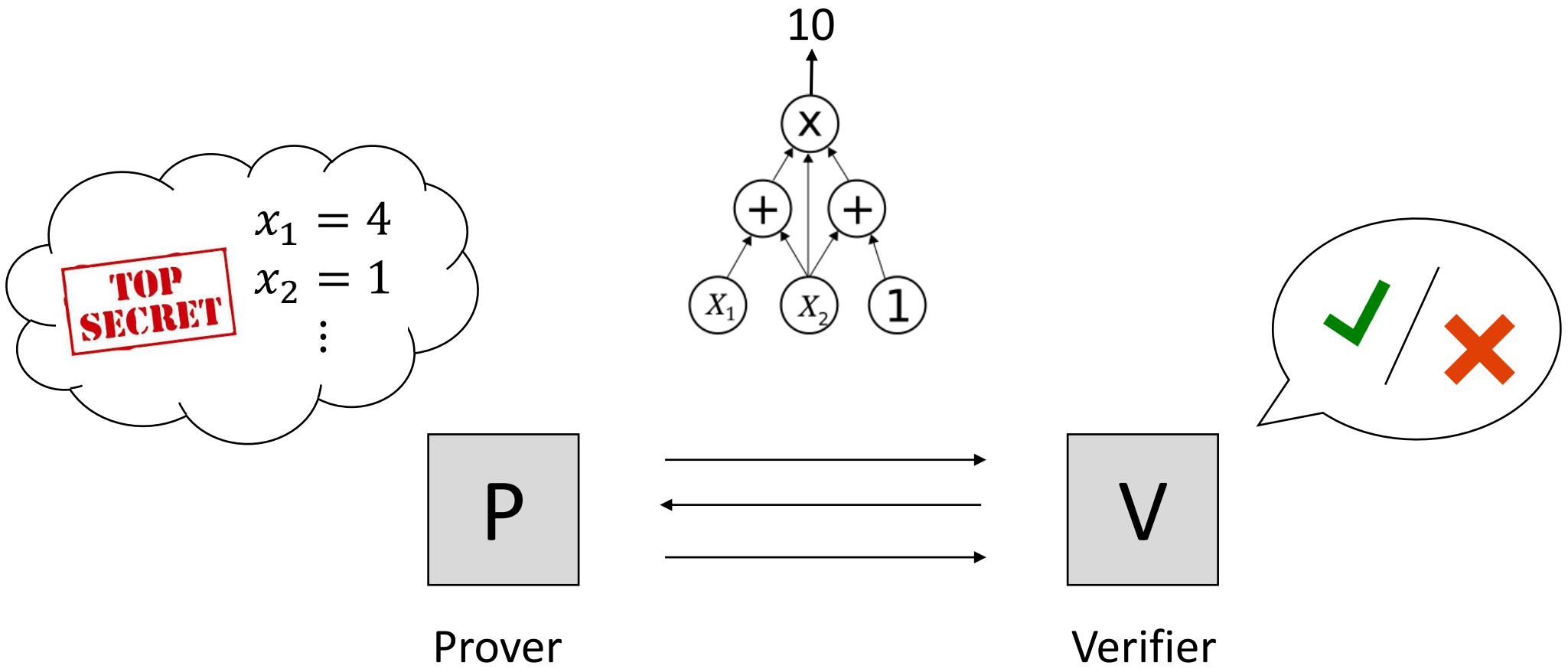
Linear-Time Arguments with Sublinear Verification from Tensor Codes

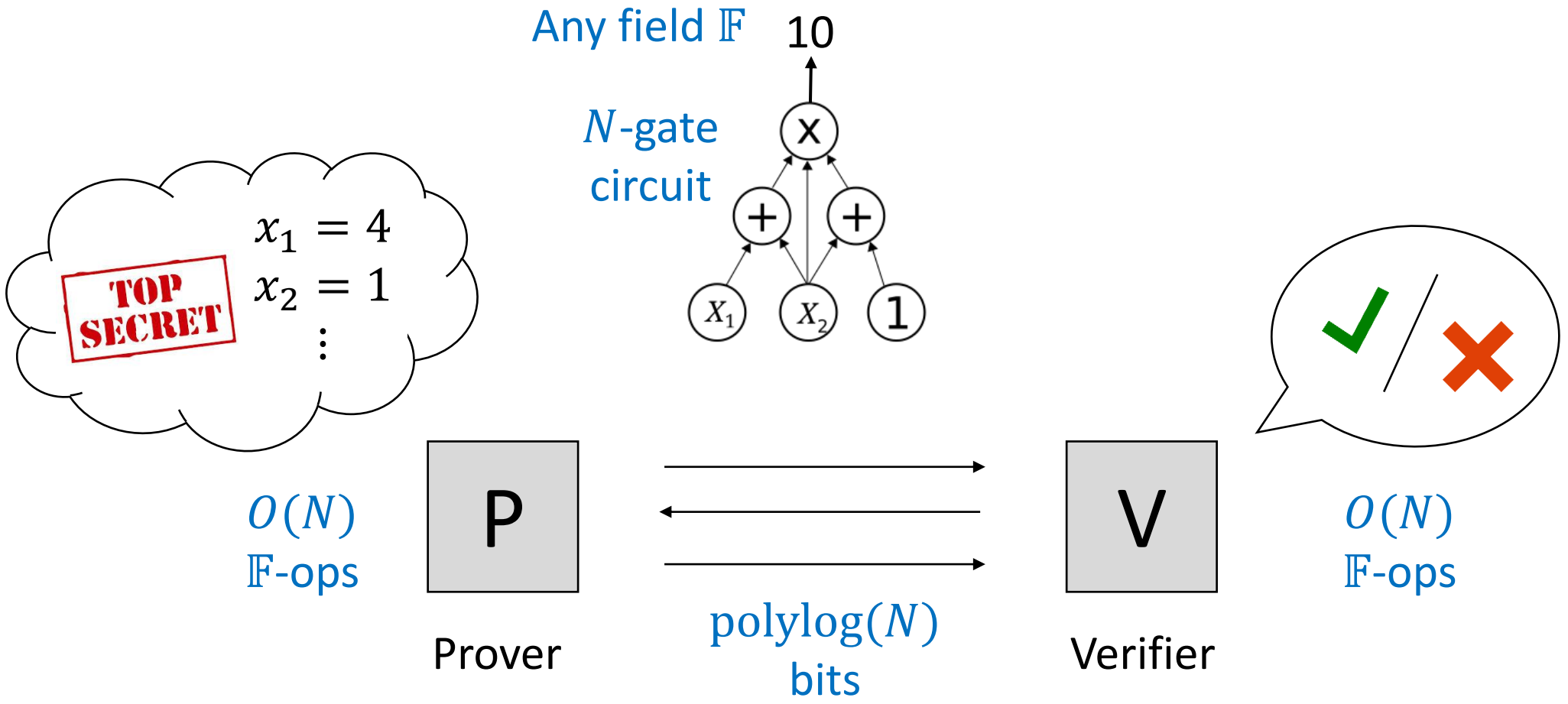
Jonathan Bootle (IBM Research – Zurich)

Joint work with Alessandro Chiesa (UC Berkeley) and Jens Groth (Dfinity)

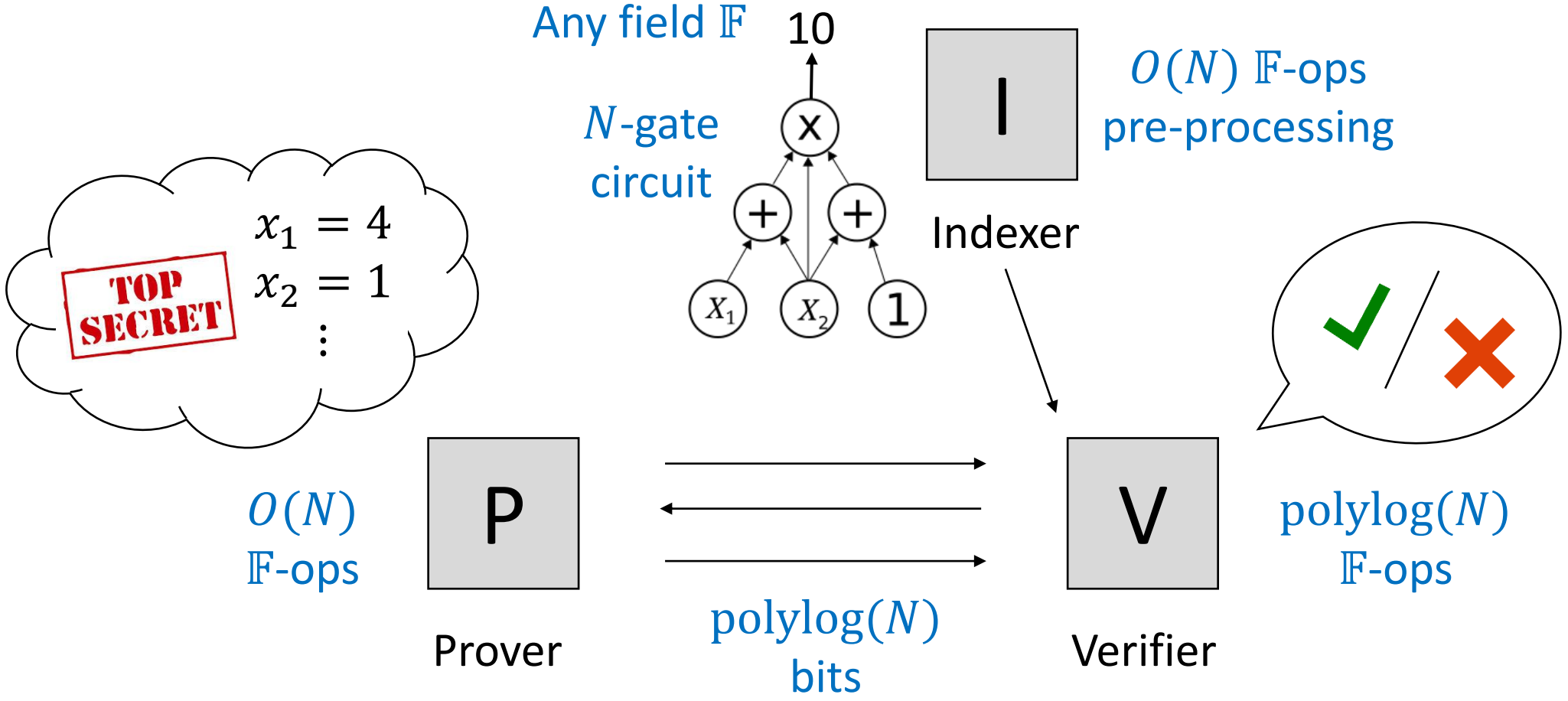
The holy grail for efficient arguments for NP



The holy grail for efficient arguments for NP



The holy grail for efficient arguments for NP



Obstacles

Fast Fourier transforms

$(w_1, w_2 \ldots, w_M)$
$O(N)$ wire values

$p(X), q(X)$
degree $O(N)$ polynomials

$O(N \log N)$
$\mathbb{F}$ -ops

$(p(\omega_1), \ldots, p(\omega_k))$
RS encodings

$p(X) \cdot q(X) = r(X)$
multiplication

Algebraic commitments

$(w_1, w_2, \ldots, w_M)$
$O(N)$ wire values

$(g_1, g_2 \ldots, g_M)$
$O(N)$ group elements

$O(N)$ group exponentiations
$=O(\lambda N)$ $\mathbb{F}$-ops

$c = g_1^{w_1} g_2^{w_2} \cdots g_M^{w_M}$
commitment

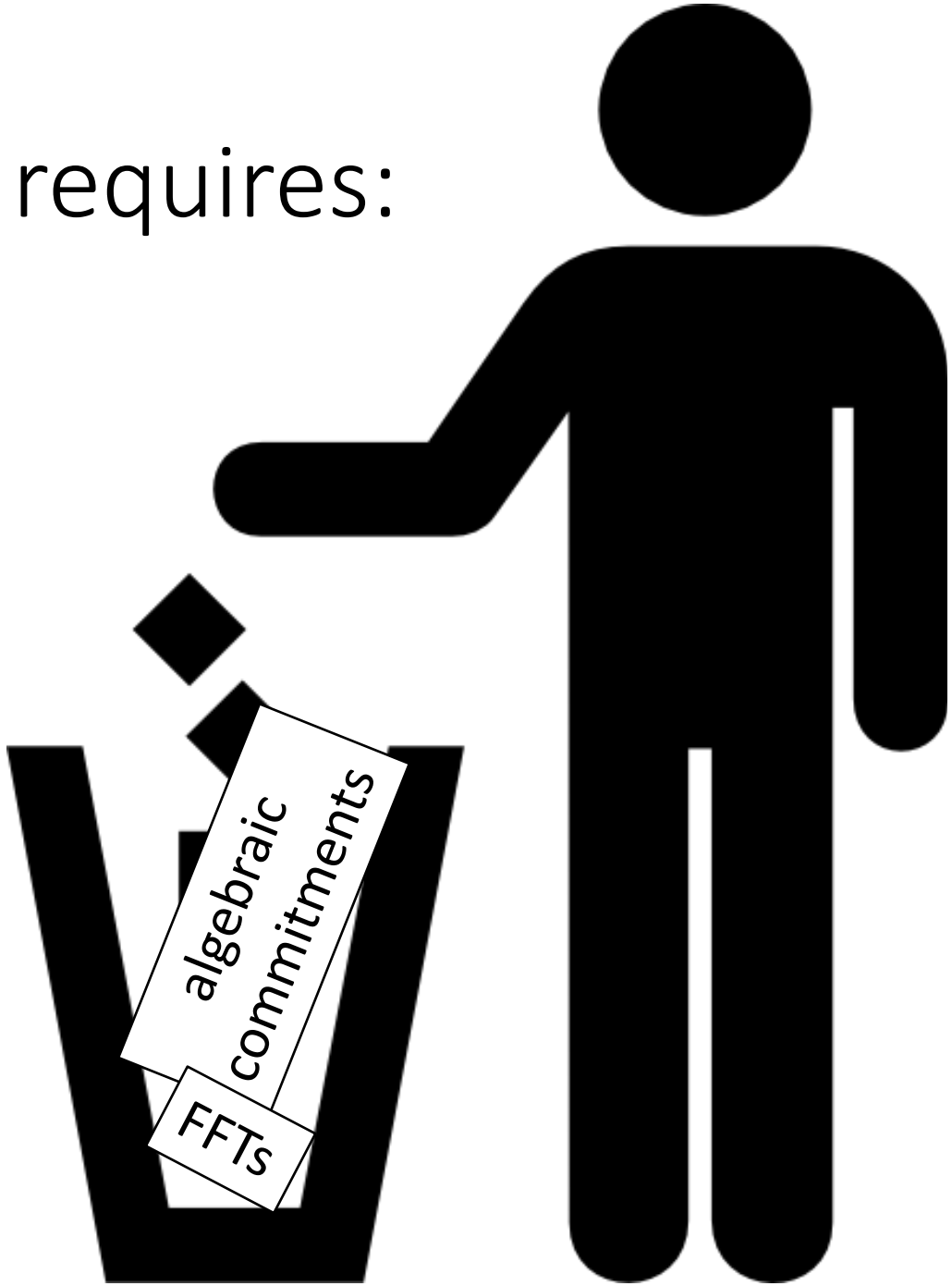
Exciting progress on provers without FFTs

Proof System	$\mathbb{F}$-ops	Other ops	Proof Size
[G16,…]	$O(N \log N)$	$O(N)$ group expo	$O(1)$
[BCCGP16], [BBBPWM18]	$O(N)$	$O(N)$ group expo	$O(\log N)$
[XZPPS19]	$O(N)$	$O(N)$ group expo	$O(D \log N)$
[S20]	$O(N)$	$O(N)$ group expo	$O(\log^2 N)$

N-gate arithmetic circuits over $\mathbb{F}$

- Close to the holy grail, but not quite linear-time…
- Excellent concrete efficiency!

The holy grail requires:



A ray of hope

- [BCGGHJ17] cryptographic argument:

Indexer complexity	Prover complexity	Verifier complexity	Proof size
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{1/2})$ $\mathbb{F}$-ops	$O(N^{1/2})$

[AHIKV17] hashes $O(N)$ hashing $= O(N)$ $\mathbb{F}$-ops

- [BCGGHJ17] interactive oracle proof:

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{1/2})$ $\mathbb{F}$-ops	$O(N^{1/2})$

Information theoretic

Challenge: can we construct linear-time IOPs with better query complexity?

Results

Our results

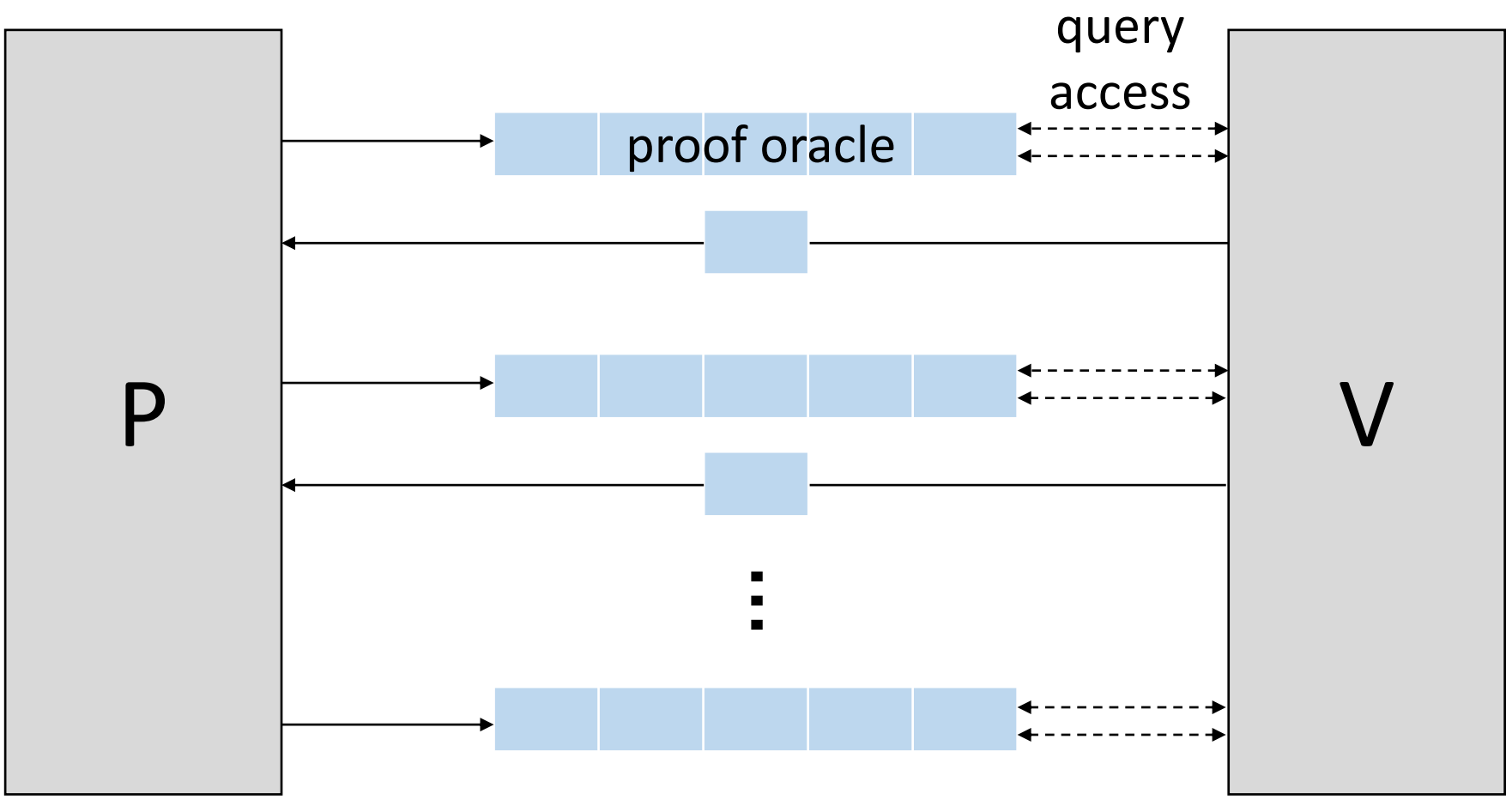
- **Corollary:** for any $\epsilon \in (0,1)$, given any linear-time CRH as a black-box, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a public-coin argument system with

Indexer complexity	Prover complexity	Verifier complexity	Proof size
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$

- **Main theorem:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a point-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$

Interactive oracle proofs

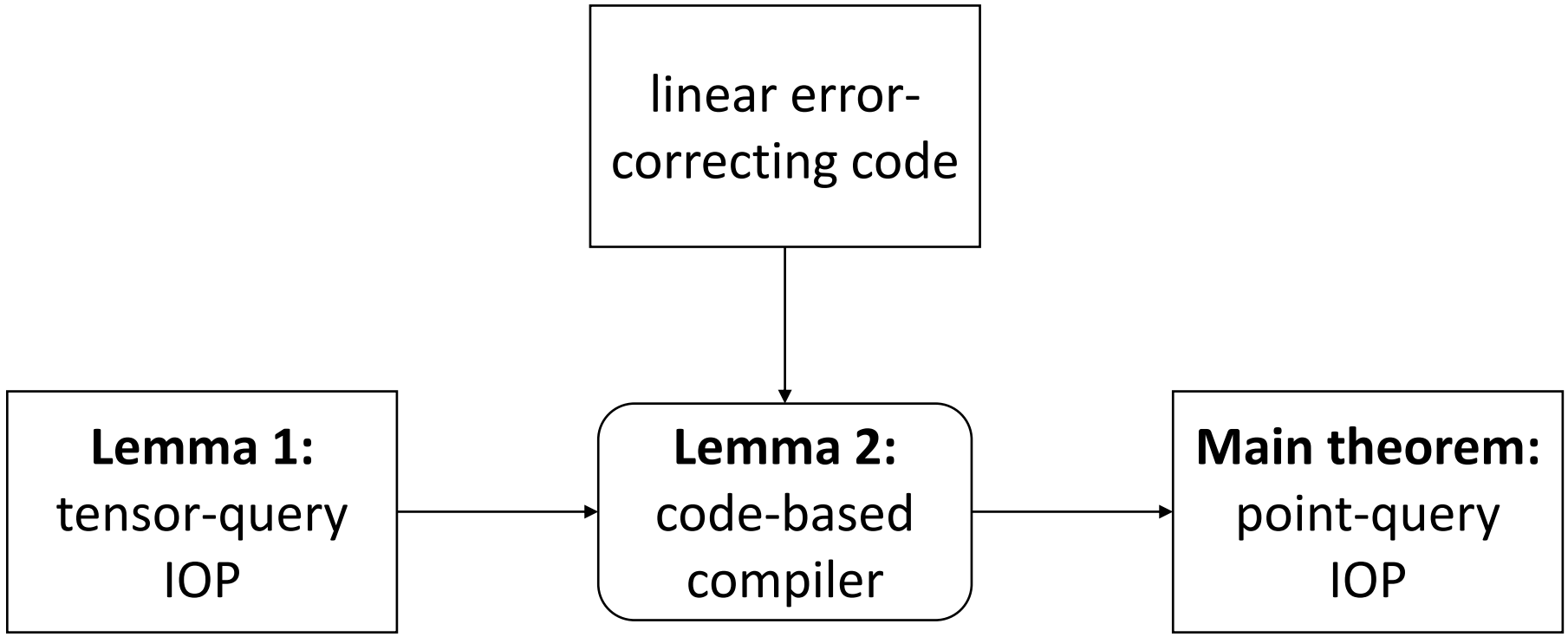


Point queries:
$query(\pi, i) = \pi(i)$
(main result)

Tensor queries:
$query(\pi, q_1, q_2)$
$= \langle \pi, q_1 \otimes q_2 \rangle$

Linear queries:
$query(\pi, q) = \langle \pi, q \rangle$

Our approach to the main theorem



Our approach in more detail

- **Lemma 1:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a tensor-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(1)$

↓ **Lemma 2:** code-based compiler

- **Main theorem:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a point-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$

Our approach in more detail

- **Lemma 1:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a tensor-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(1)$

↓ **Lemma 2:** code-based compiler

- **Main theorem:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a point-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$

Lemma 2: the code-based compiler

- **Input:** tensor-query IOP

Indexer complexity	Prover complexity	Verifier complexity	Proof length	# Queries
T_I	T_P	T_V	l	q

- **Input:** linear code C

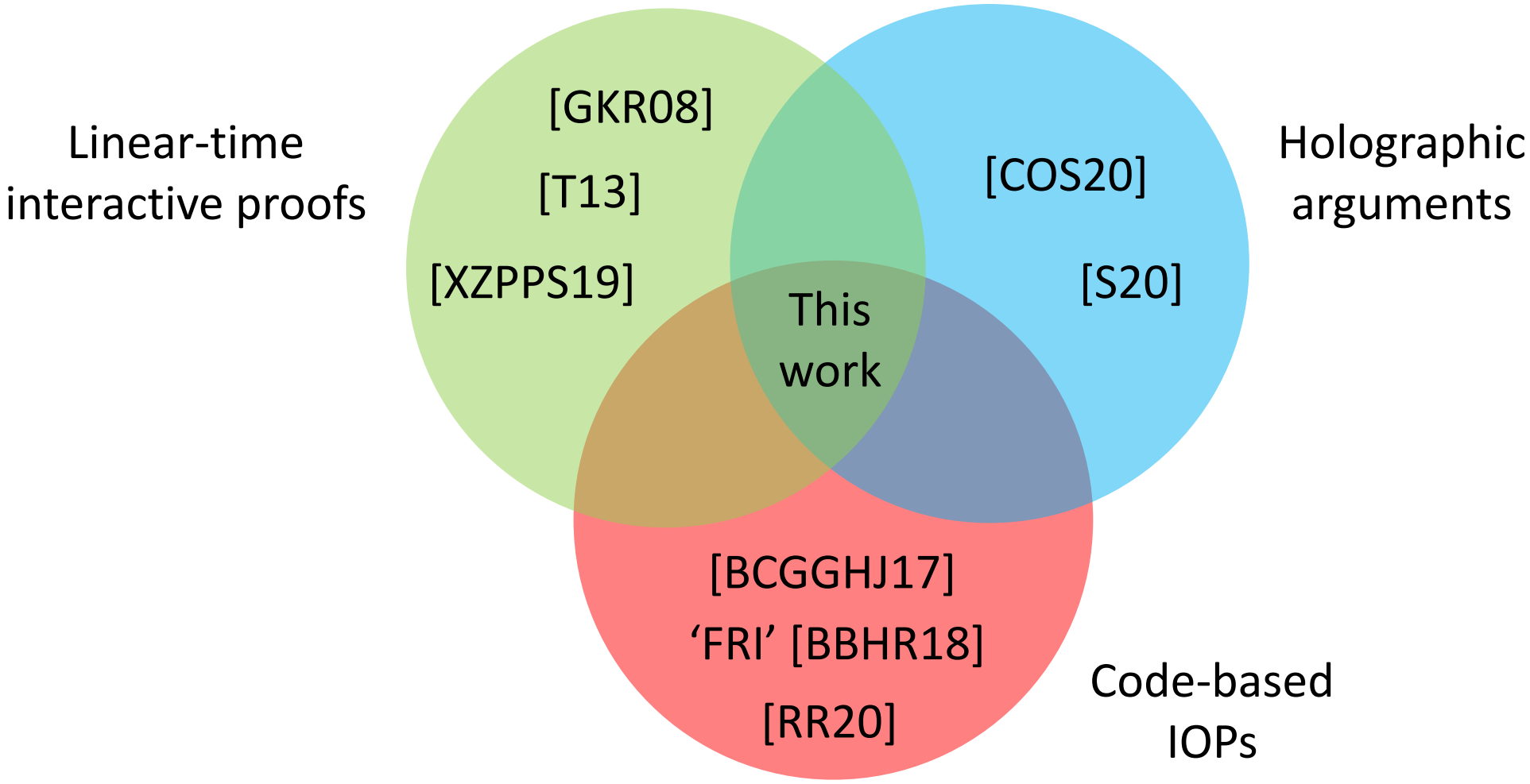
Message length	Rate	Encoding time
$k = O(l^{\epsilon})$	ρ	$k \cdot \theta(k)$

If C is linear-time encodable then the compiler preserves prover and indexer complexity

- **Output:** point-query IOP

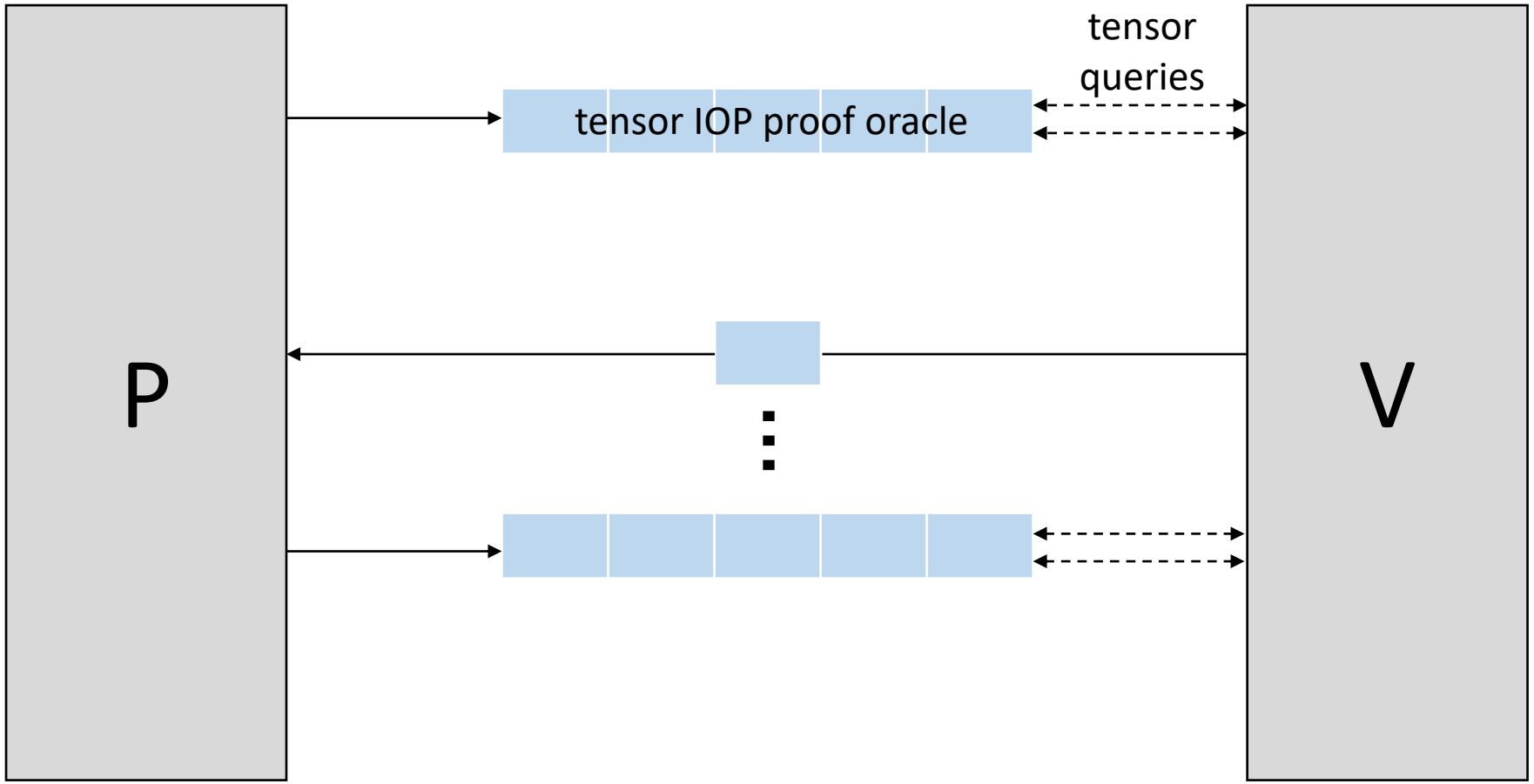
Indexer complexity	Prover complexity	Verifier complexity	Proof length	# Queries
$T_I + O_\rho(l) \cdot \theta(k)$	$T_I + O_\rho(l) \cdot \theta(k)$	$T_V + O(q \cdot k) \cdot \theta(k)$	$O_\rho(q \cdot l)$	$O(q \cdot k)$

Related techniques

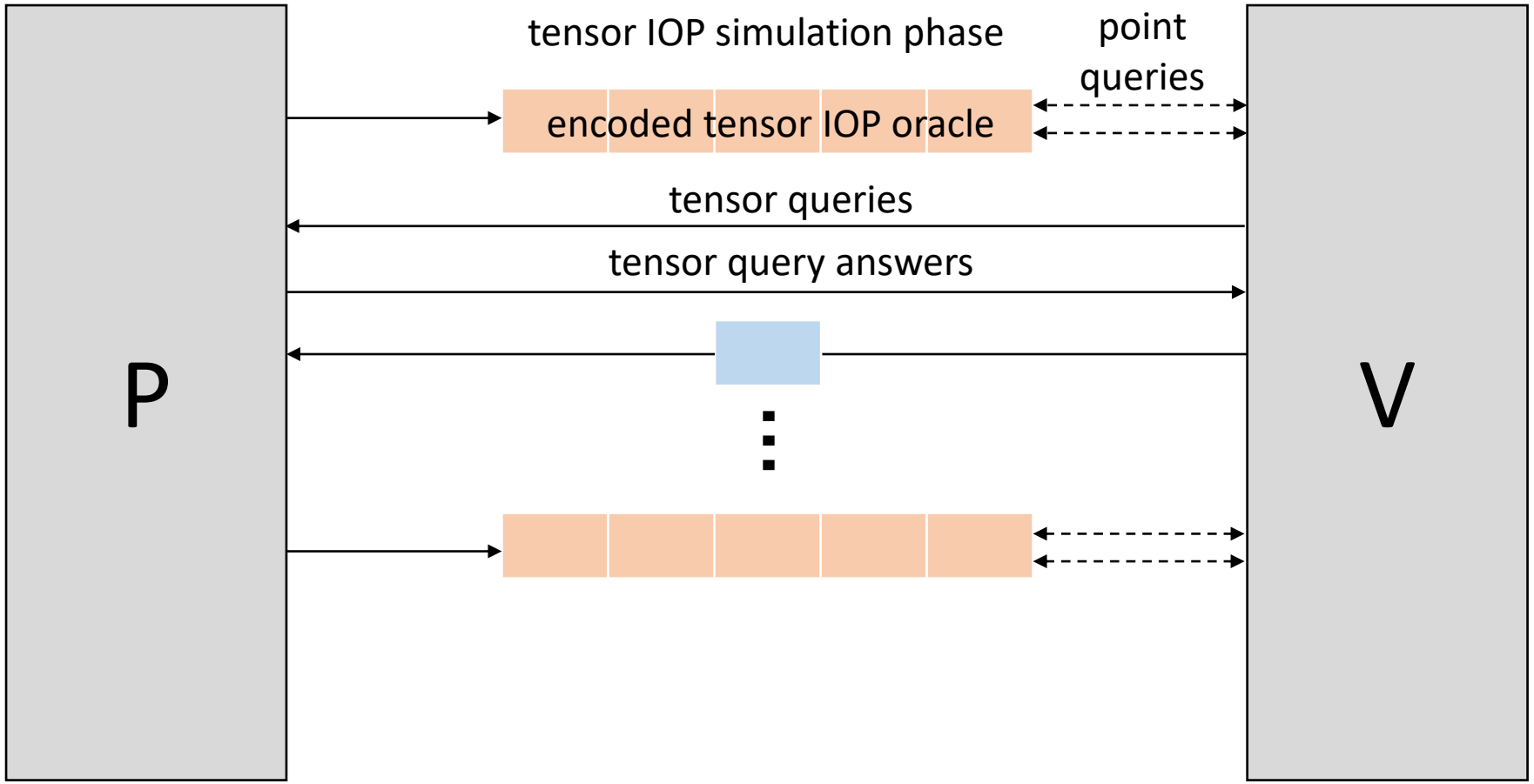


Code-based compiler techniques

The input tensor-query IOP



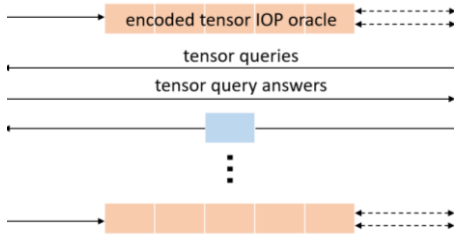
The compiled point-query IOP



The compiled point-query IOP

P

tensor IOP simulation phase



tensor query consistency phase

1) Proximity test: did the prover really send encoded oracles (or close enough)?
2) Consistency test: are query answers consistent with the encoded oracles?

IOPP for 1) and 2)

Explicit: query answer v, implicit: encoding e

Show $e = enc(\pi)$ and $v = \langle \pi , \bigotimes_i q_i \rangle$

V

Which encodings preserve linear time and admit proximity and consistency IOPPs?

Encodings using the tensor code $C^{\otimes t}$

$O(N)$
tensor IOP
proof data

$t = 3$

- No special properties needed from C
- Linear-time-encodable if C is linear-time encodable [S96] or [DI14]

How does the consistency test work?

tensor IOP
proof oracle
π

tensor codeword
$enc(\pi)$

How are these
connected?

tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

How does the consistency test work?

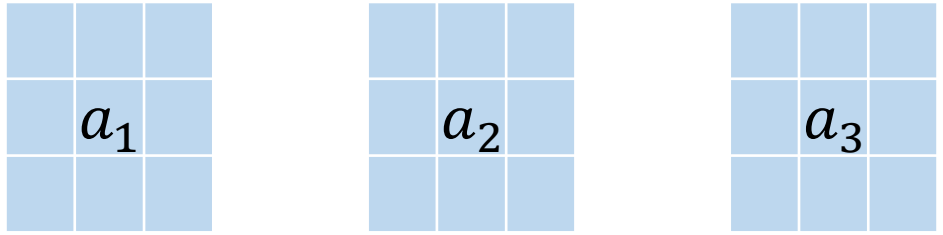
tensor IOP
proof oracle
π

tensor codeword
$enc(\pi)$

How are tensor queries computed?

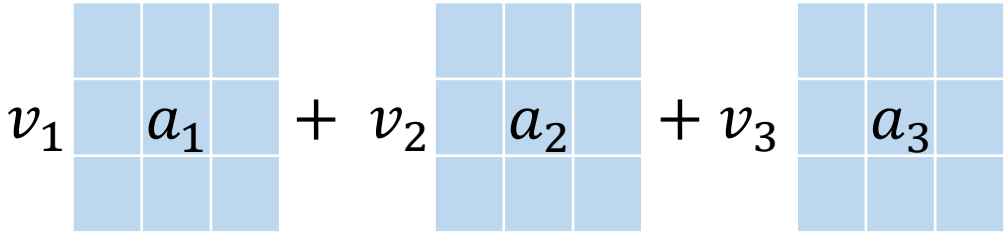
tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

A folding operation

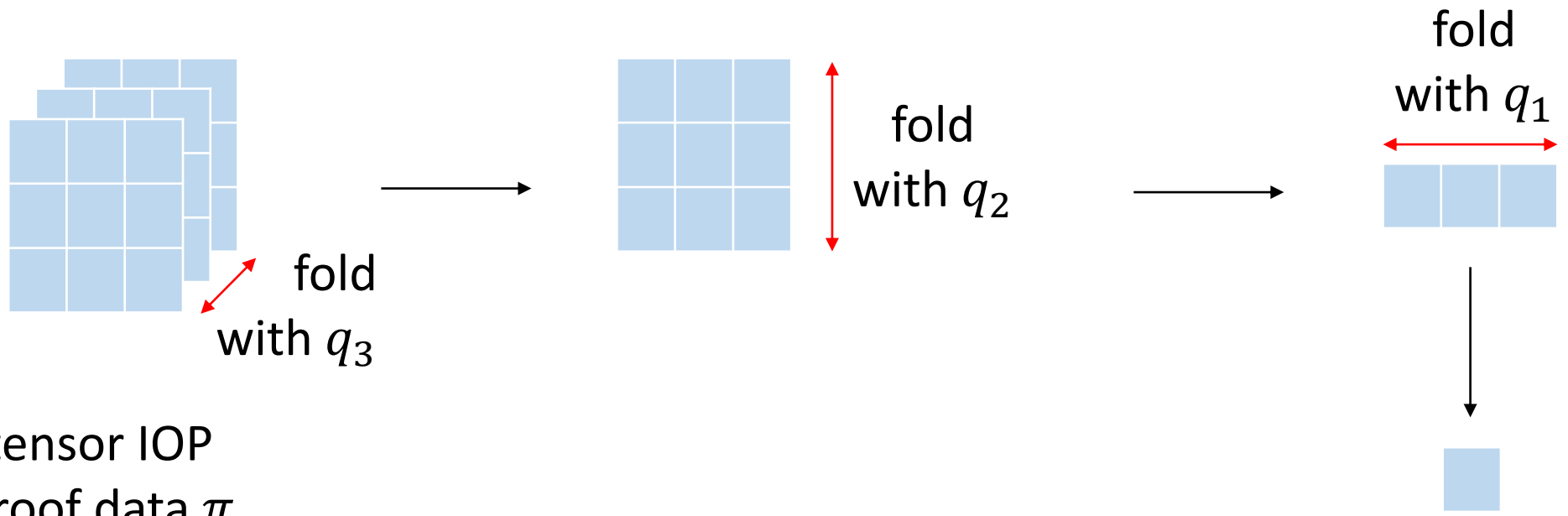


$$v = (v_1, v_2, v_3)$$

'fold with v'



Computing query answers by folding



Computing query answers by folding

tensor IOP
proof oracle
π

fold

fold

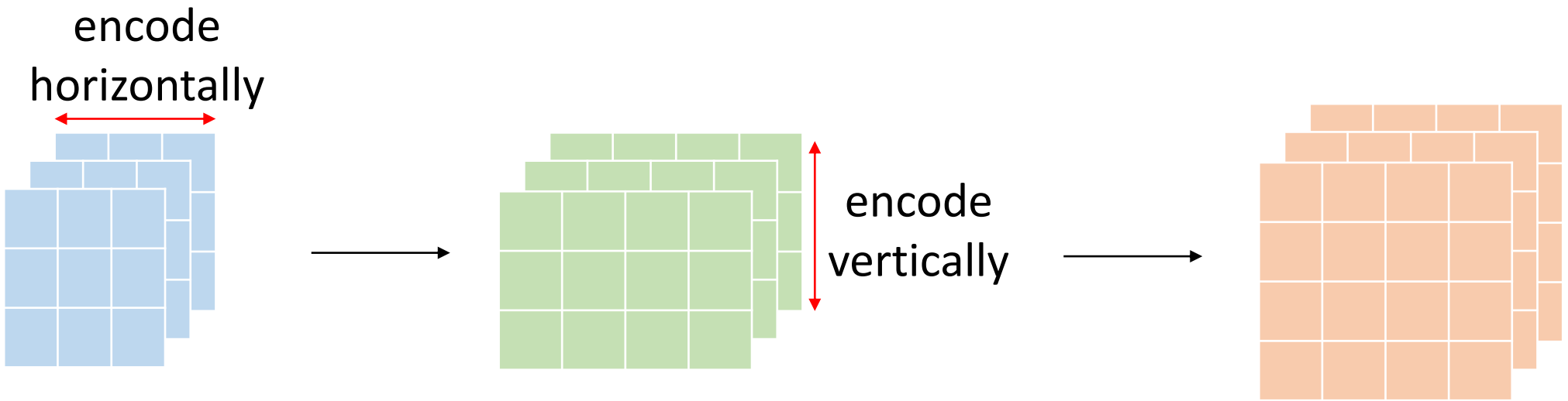
fold

tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

How are tensor
encodings computed?

tensor codeword
$enc(\pi)$

Partial tensor encodings



tensor IOP
proof oracle
π

$t = 3$

tensor codeword
$enc(\pi)$

Partially encoding tensor IOP proofs

tensor IOP proof oracle π

encode

encode

tensor codeword $enc(\pi)$

tensor query answer $\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

How do we check consistency?

tensor IOP
proof oracle
π

fold

encode

tensor codeword
$enc(\pi)$

tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

How do we check consistency?

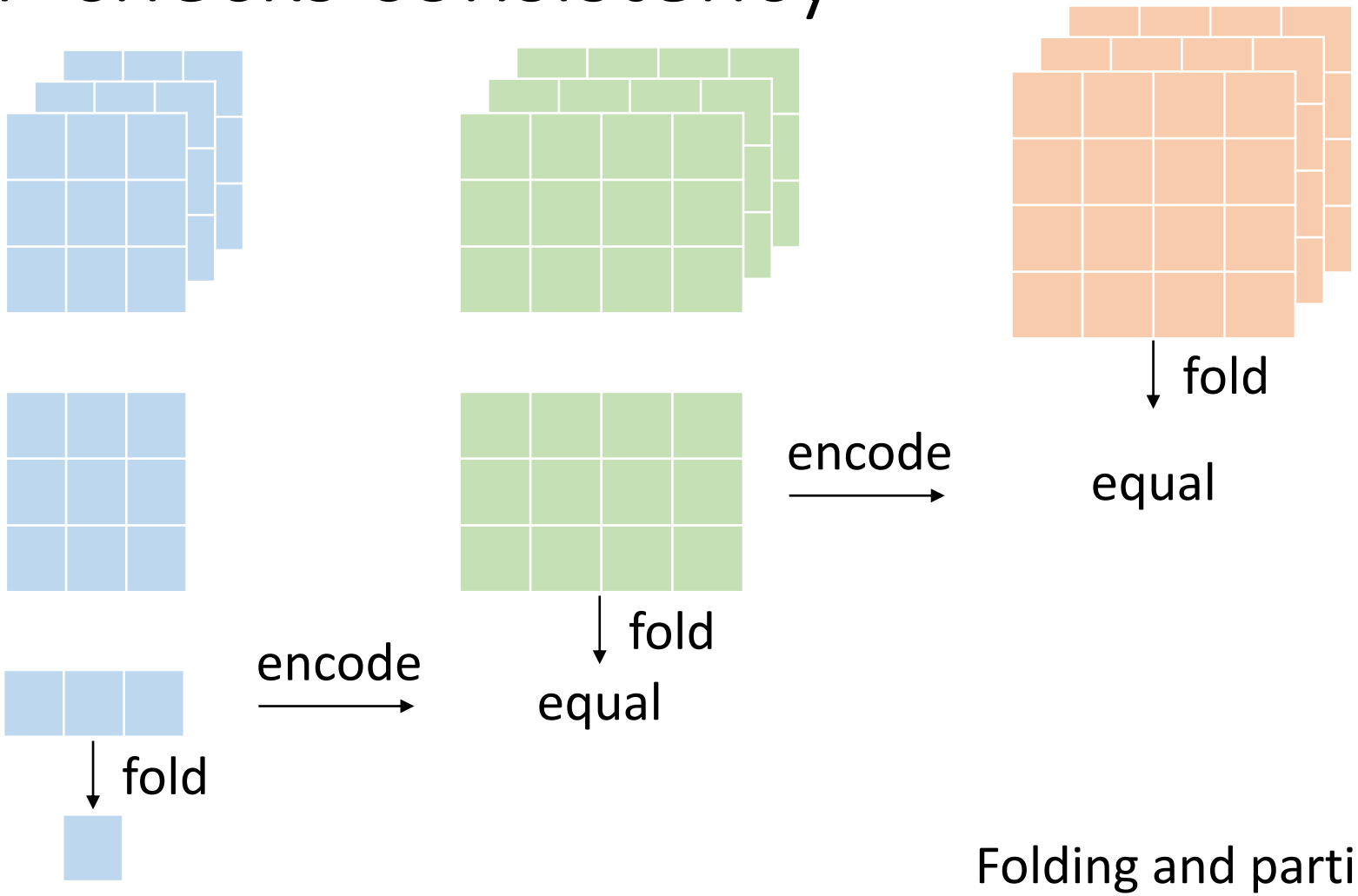
tensor IOP
proof oracle
π

tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

tensor codeword
$enc(\pi)$

Check consistency at
each step?

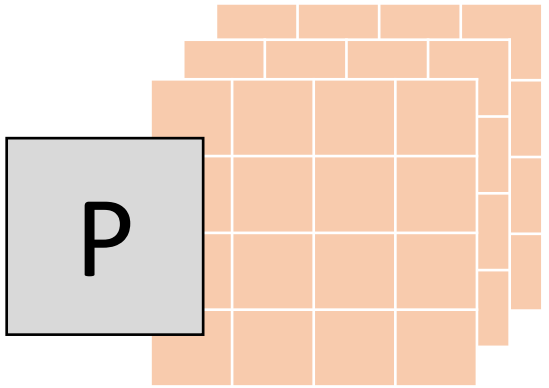
How the IOPP checks consistency



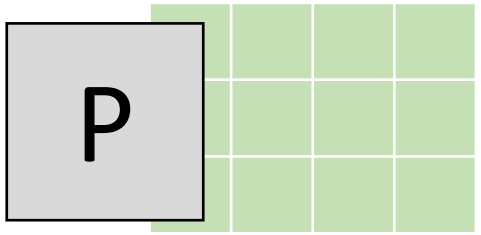
Folding and partial encodings commute

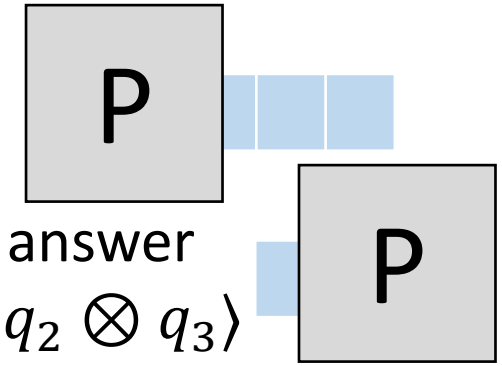
The consistency check IOPP

tensor IOP
proof oracle
π



tensor codeword
$enc(\pi)$

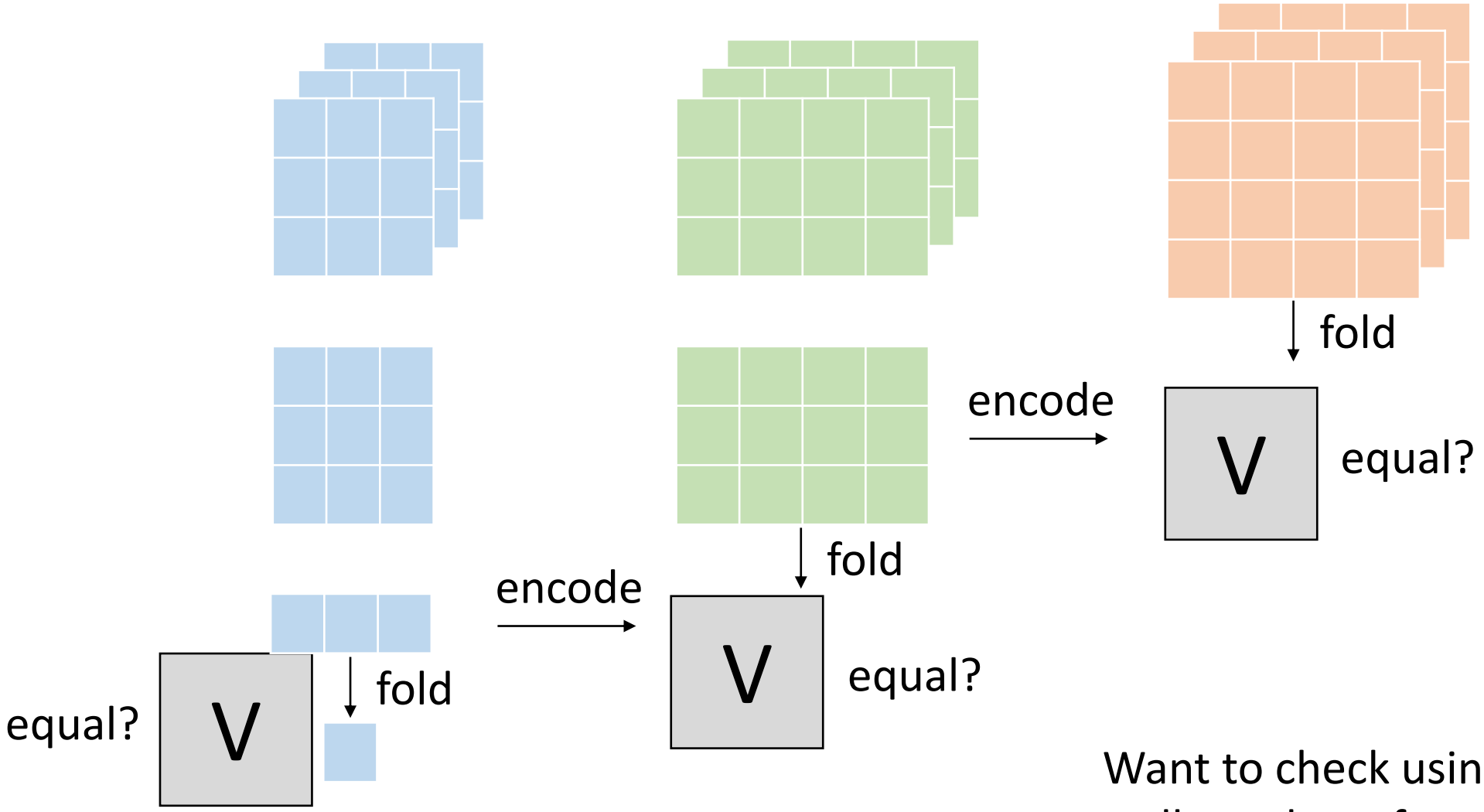




tensor query answer
$\langle vec(\pi), q_1 \otimes q_2 \otimes q_3 \rangle$

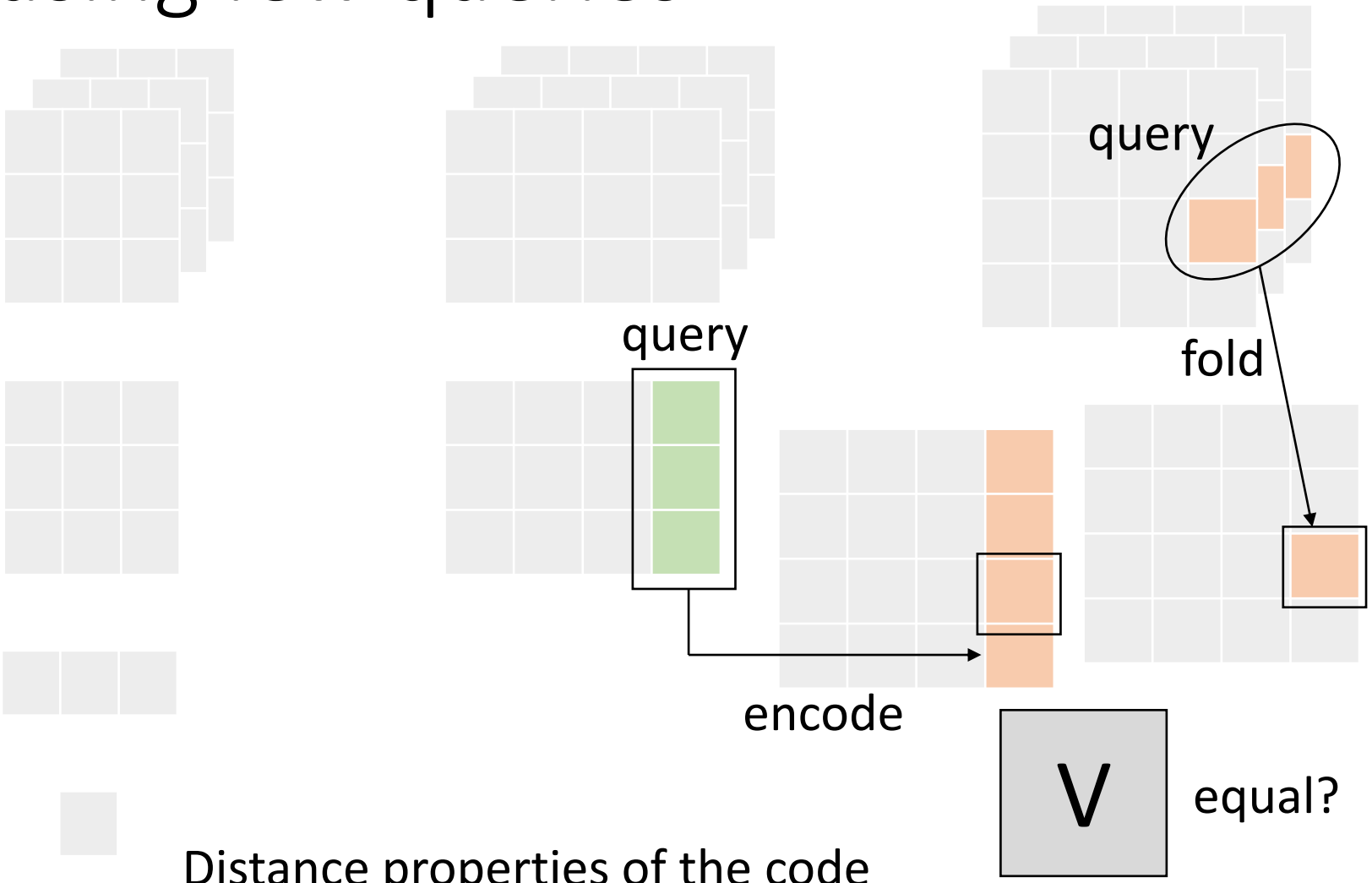
prover messages

What the verifier wants to check



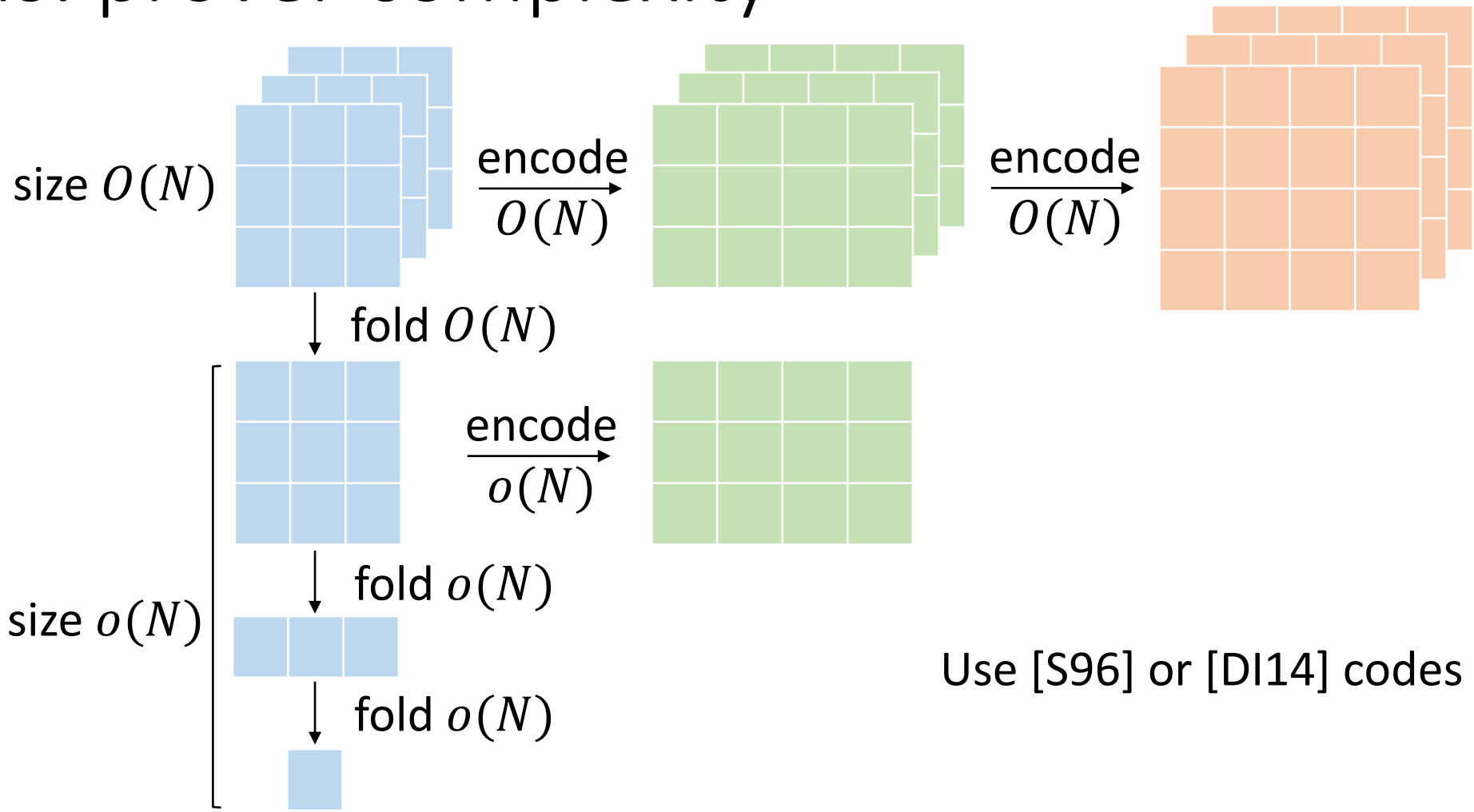
Want to check using a small number of queries

Spot-checks using few queries



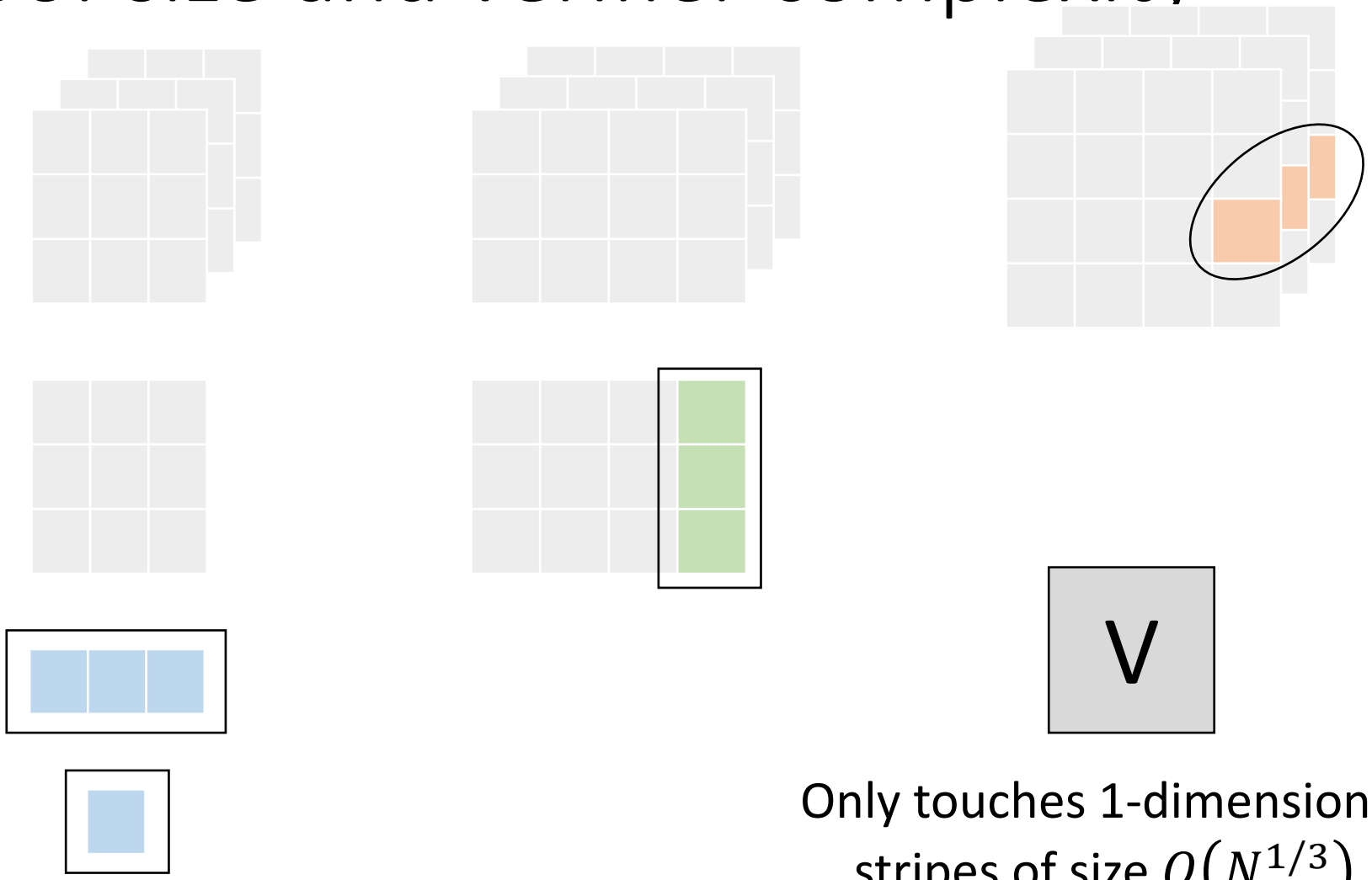
Distance properties of the code guarantee inconsistencies are caught

Analysis: prover complexity



Use [S96] or [DI14] codes

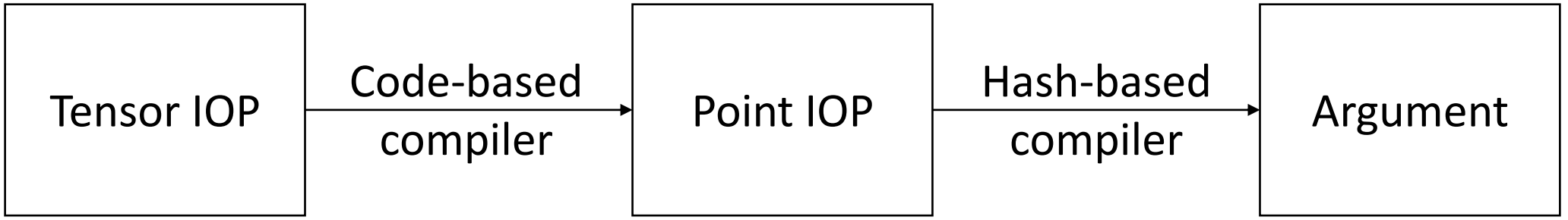
Analysis: proof size and verifier complexity



Only touches 1-dimensional stripes of size $O\left(N^{1/3}\right)$

Conclusion

Our approach



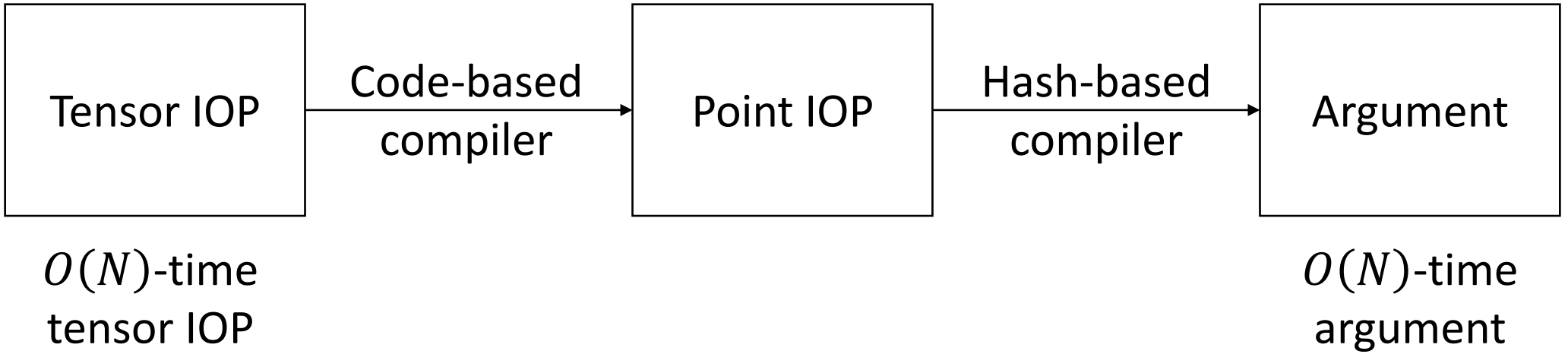
Any tensor IOP

Sub-linear proof size

Any linear code

Any hash function

Our approach



Thanks!

- **Corollary:** for any $\epsilon \in (0,1)$, given any linear-time CRH as a black-box, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a public-coin argument system with

Indexer complexity	Prover complexity	Verifier complexity	Proof size
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$

- **Main theorem:** for any $\epsilon \in (0,1)$, CSAT over any field $\mathbb{F}$ of size $\Omega(N)$ has a point-query IOP with

Indexer complexity	Prover complexity	Verifier complexity	# Queries
$O(N)$ $\mathbb{F}$-ops	$O(N)$ $\mathbb{F}$-ops	$O(N^{\epsilon})$ $\mathbb{F}$-ops	$O(N^{\epsilon})$