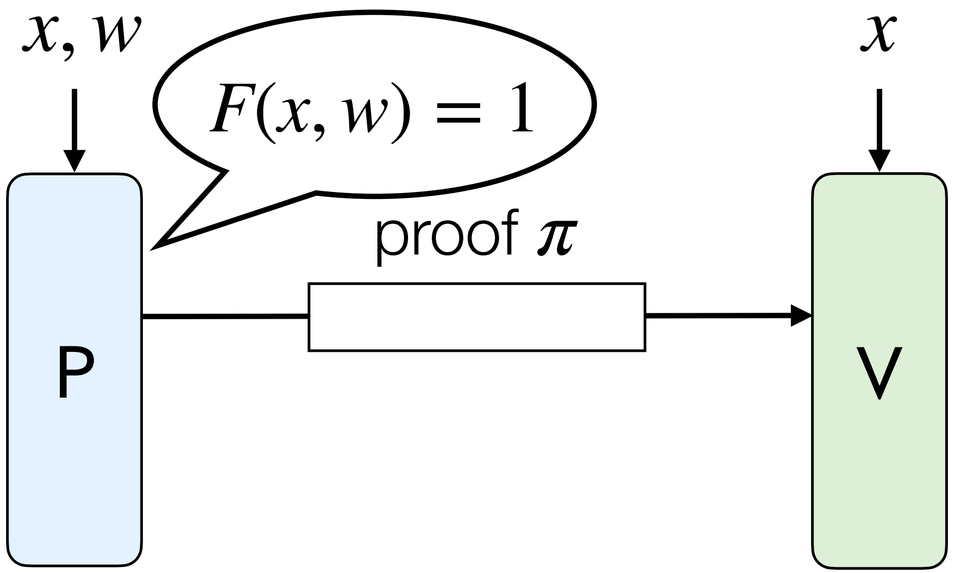
The Quest for Optimal IOPs

Pratyush Mishra

University of Pennsylvania

SNARKs

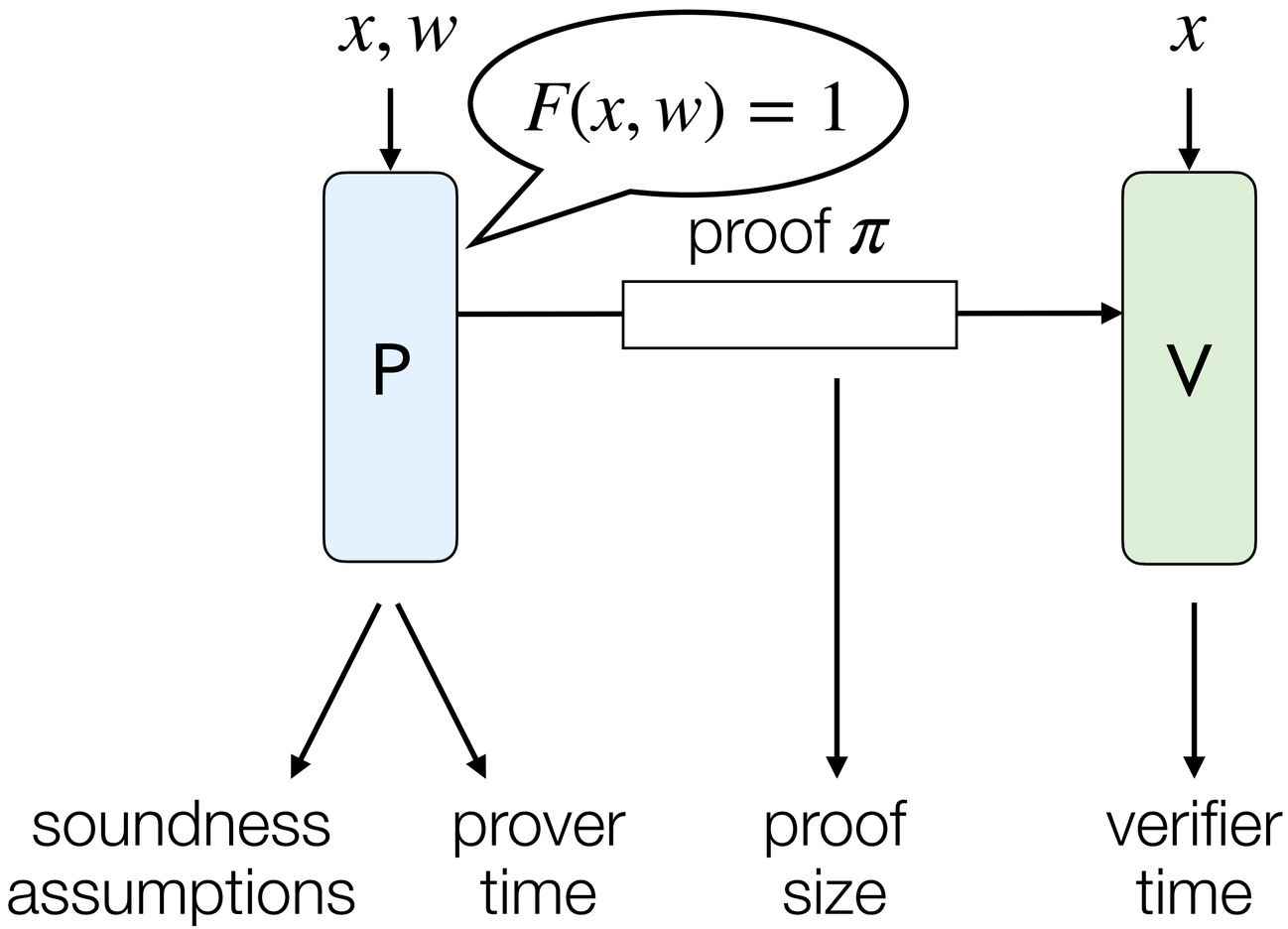


scalable ledgers

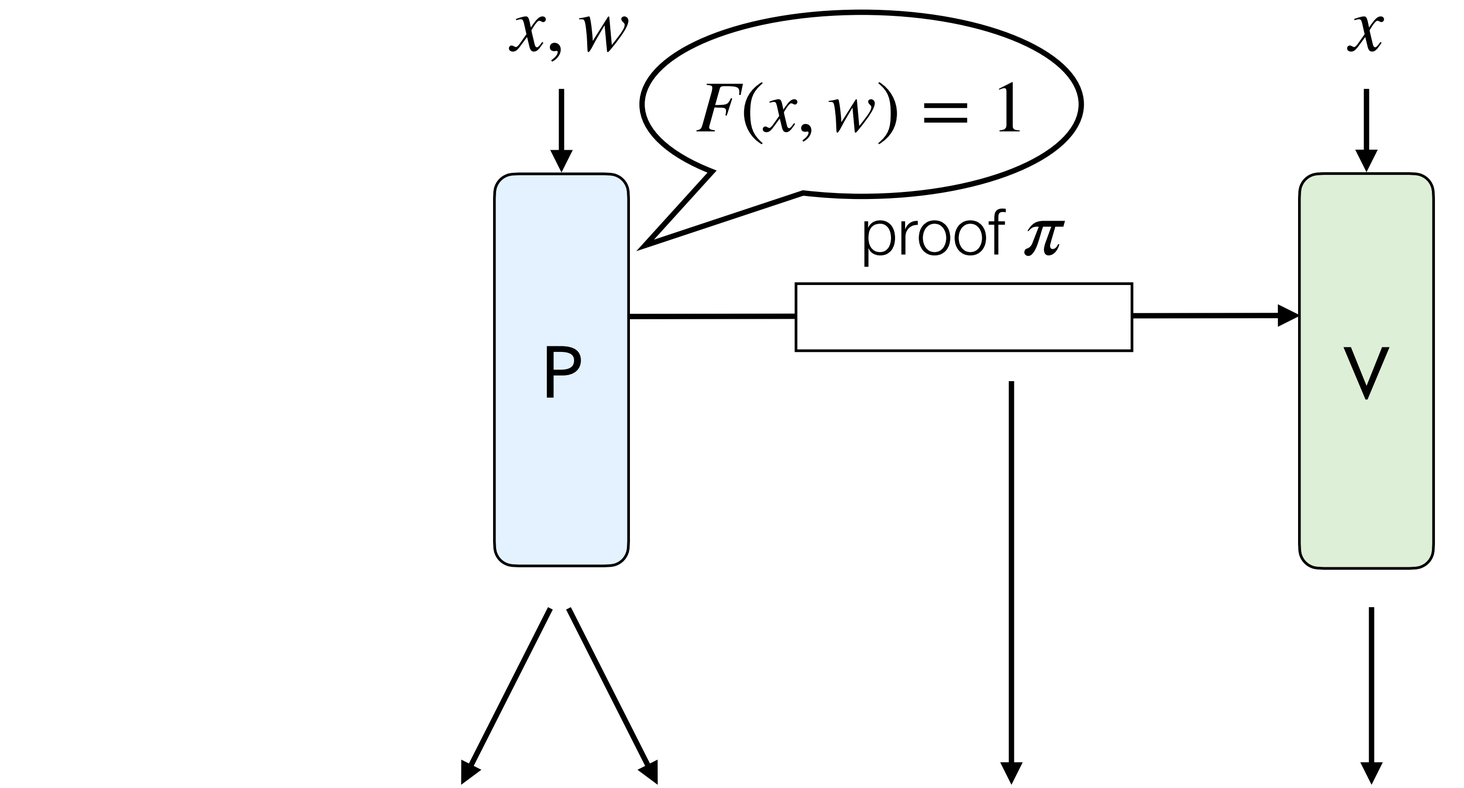
private digital IDs

private cryptocurrencies

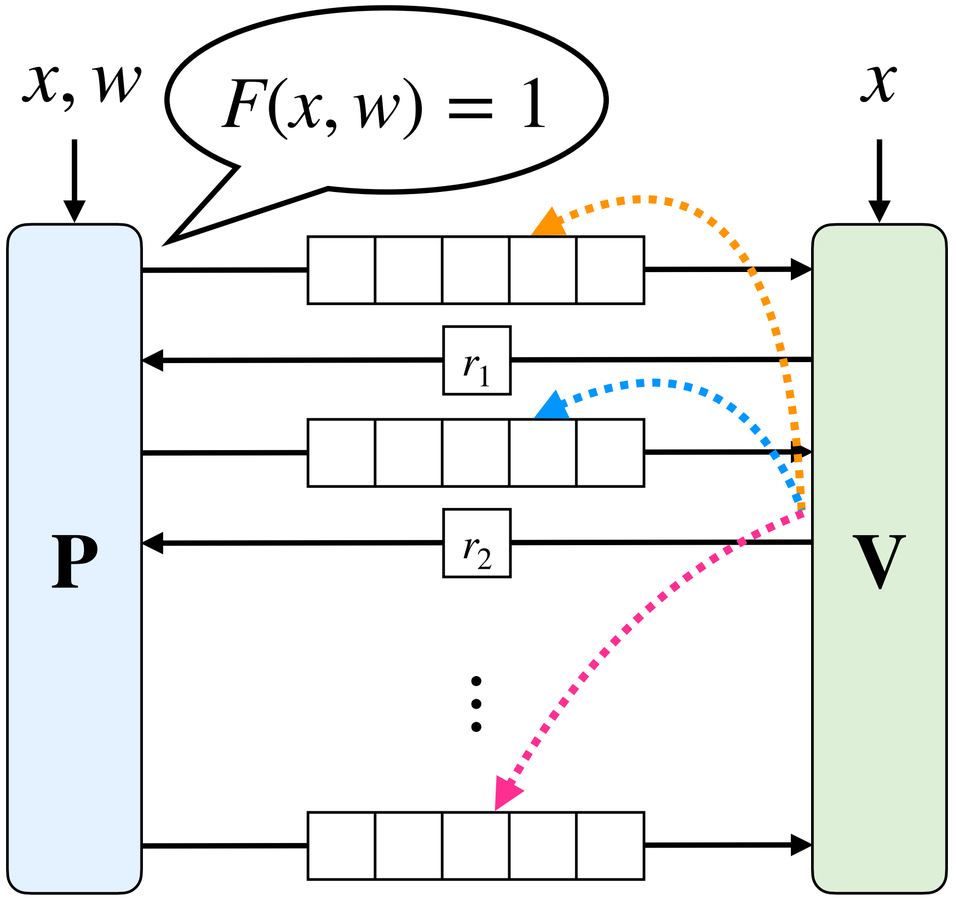
What Properties Do We Care About?



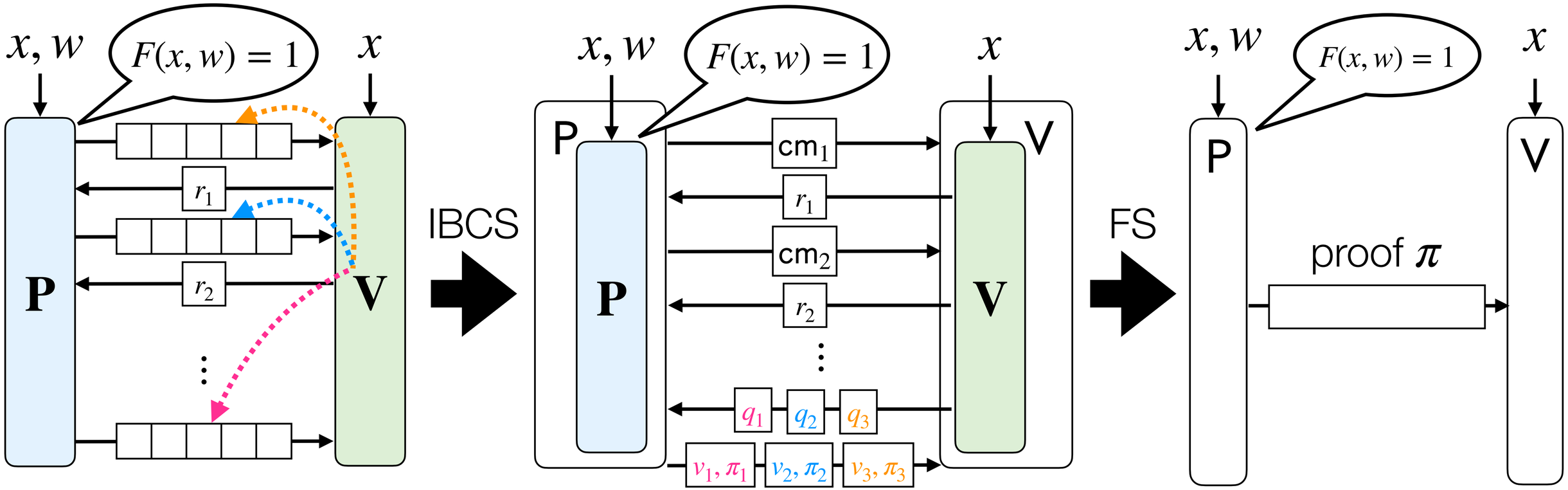
SNARKs from IOPs [K92, Mic94, BCS16]



Interactive Oracle Proofs (IOPs) [BCS16, RRR16]

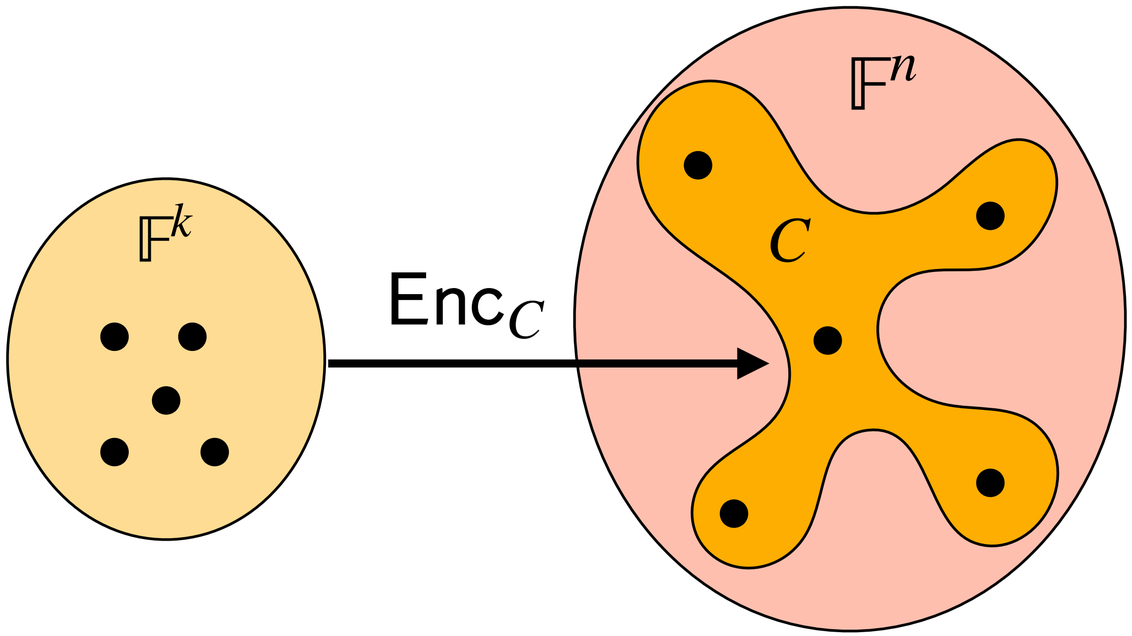


IOPs → SNARKs [Mic94, BCS16]



prover time	proof size	verifier time
$\lvert \mathbf{P} \rvert$	$Q \log n$	$Q \log n$ hashes + $\lvert \mathbf{V} \rvert$

Constructing IOPs from Codes



Distance δ : how far apart are points in C ?

Rate $\rho = k/n$: how redundant is C ?

Encoding time T_C : how long does Enc_C take?

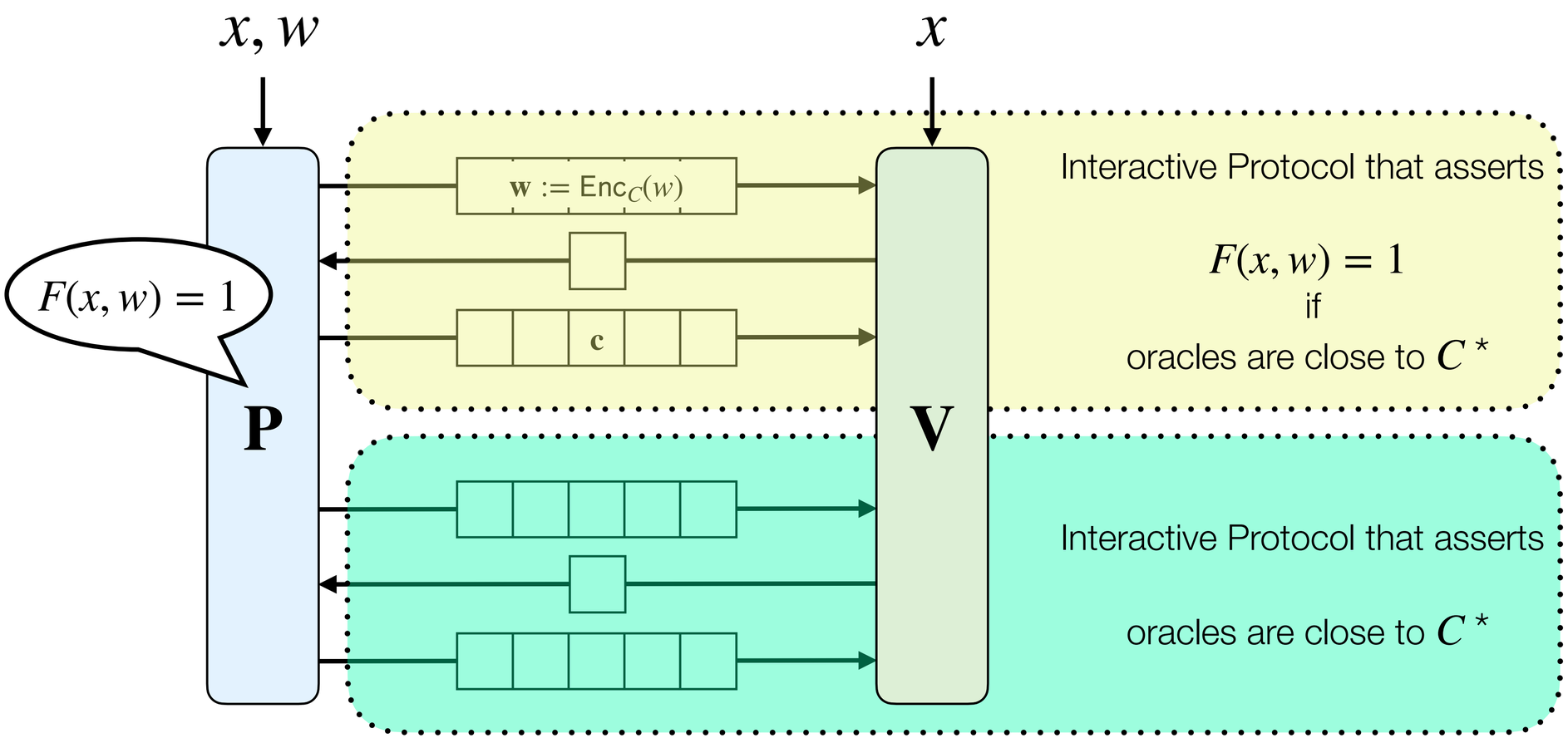
Constructing IOPs from Codes

Code C with

Distance δ

Rate $\rho = k/n$

Encoding time T_C



* and (simple) claims about underlying messages are true

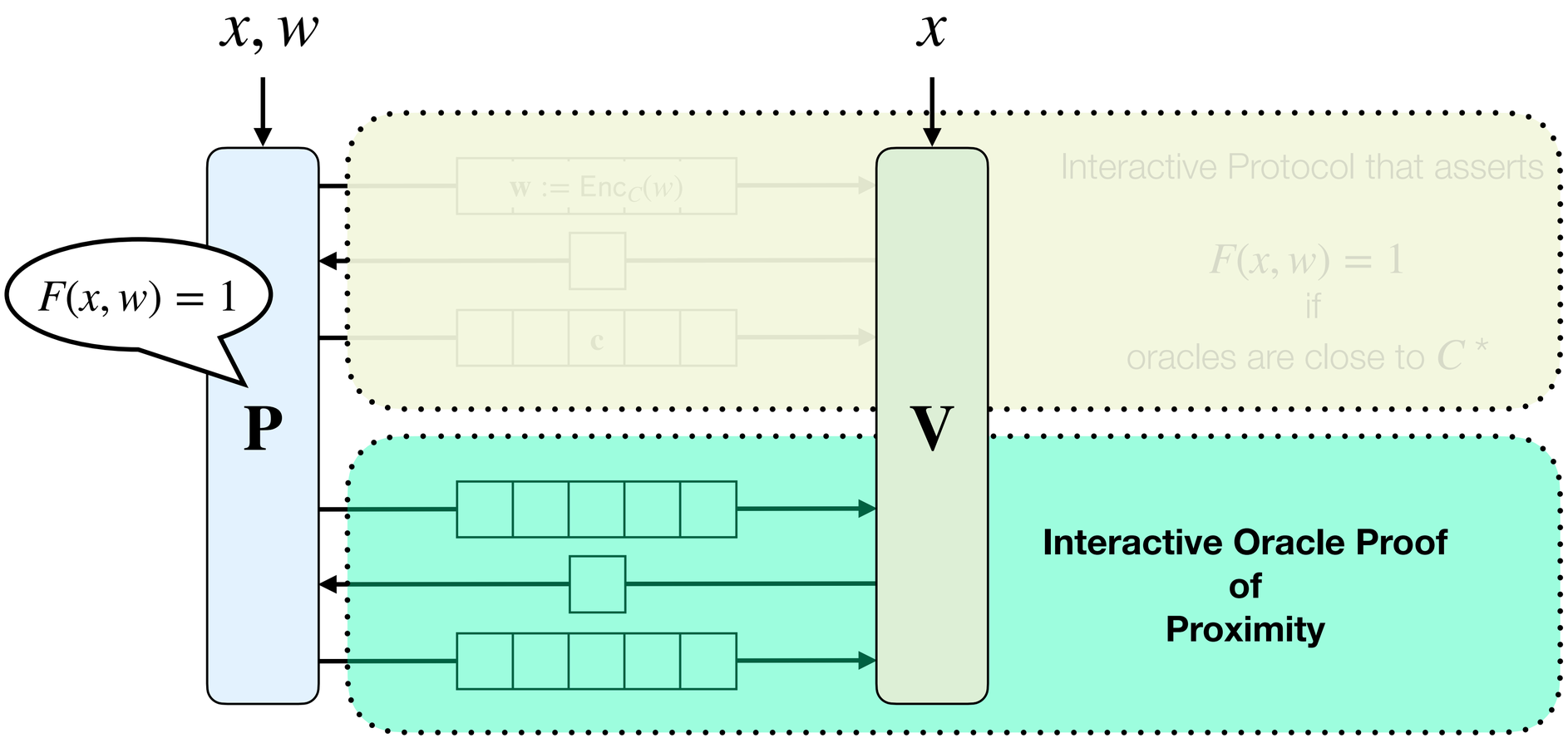
Constructing IOPs from Codes

Code C with

Distance δ

Rate $\rho = k/n$

Encoding time T_C



Near-optimal constructions
[Set20, CBBZ23]
Prover time: $O(T_F + T_C)$
Query complexity: 0 (!)

Prover time: ???
Query complexity: ???

IOP prover time $= O(T_F + T_C + T_{\mathsf{IOPP}})$

IOP query complexity $= Q_{\mathsf{IOPP}}$

* and (simple) claims about underlying messages are true

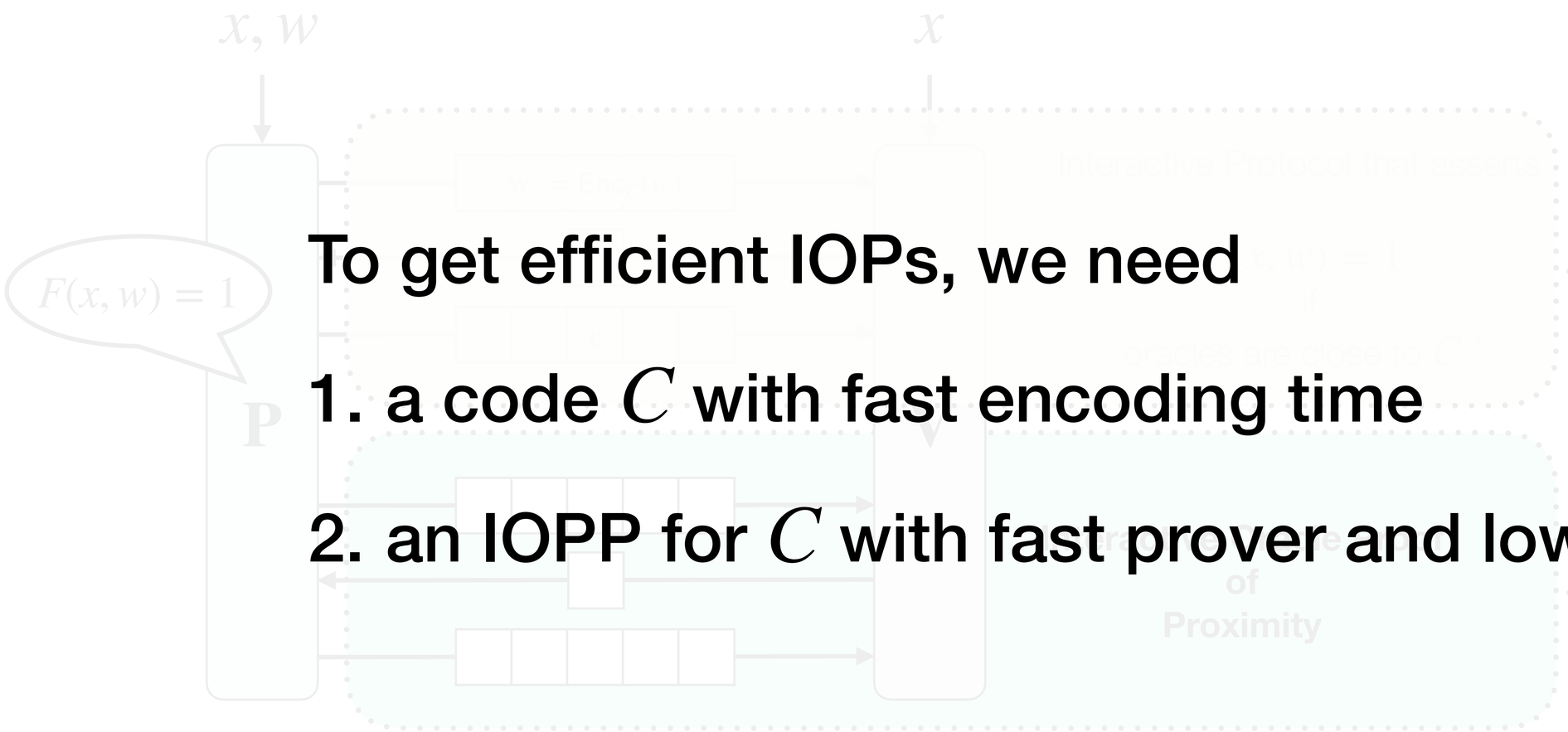
Constructing IOPs from Codes

Key tool: Code C with

Distance δ

Rate $\rho = k/n$

Encoding time T_C



Near-optimal constructions
[Set20, CBBZ23]

Prover time: $O(T_F + T_C)$

Query complexity: 0 (!)

Prover time: ???

Query complexity: ???

To get efficient IOPs, we need

1. a code C with fast encoding time

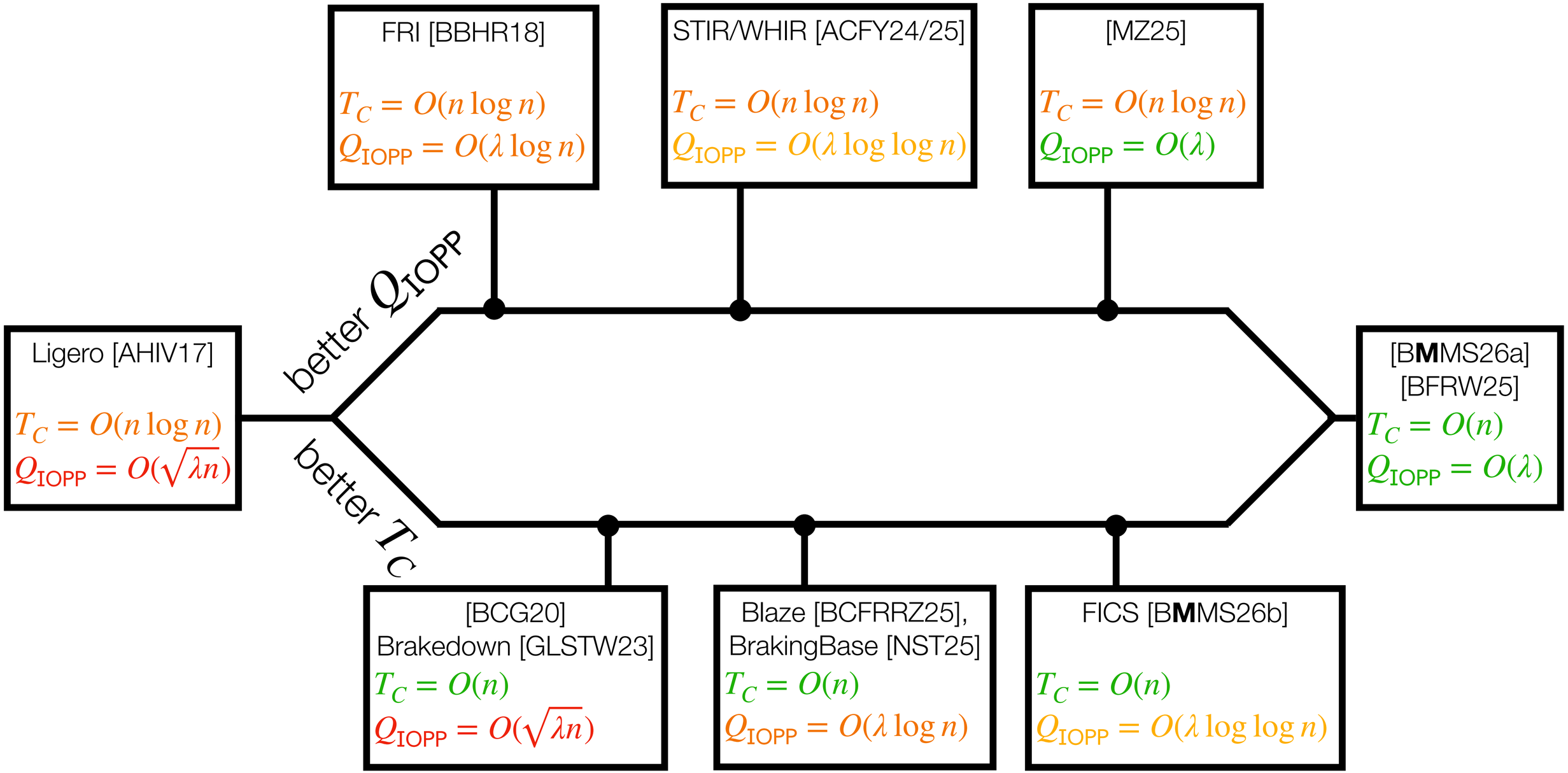
2. an IOPP for C with fast prover and low # queries

Prover time $= O(T_F + T_C + T_{\text{IOPP}})$

Query complexity $= Q_{\text{IOPP}}$

* and (simple) claims about underlying messages are true

Efficient Codes and IOPPs for them



Chapter 1
IOPPs by Interleaving Codes

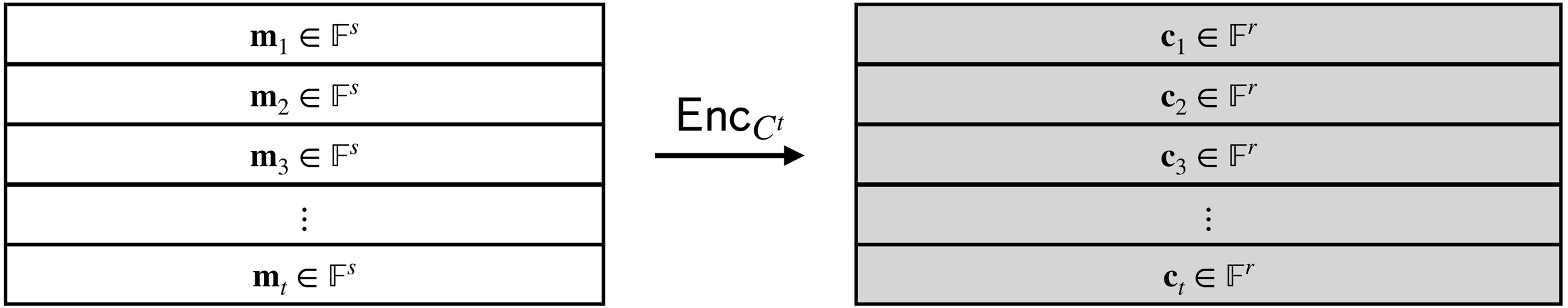
Goal:

IOPP with $O(\sqrt{\lambda k})$ queries

Interleaved Codes

Let $C \subseteq \mathbb{F}^r$ be a code with message space $\mathbb{F}^s$.

The t-wise **interleaving** of C is denoted by C^t

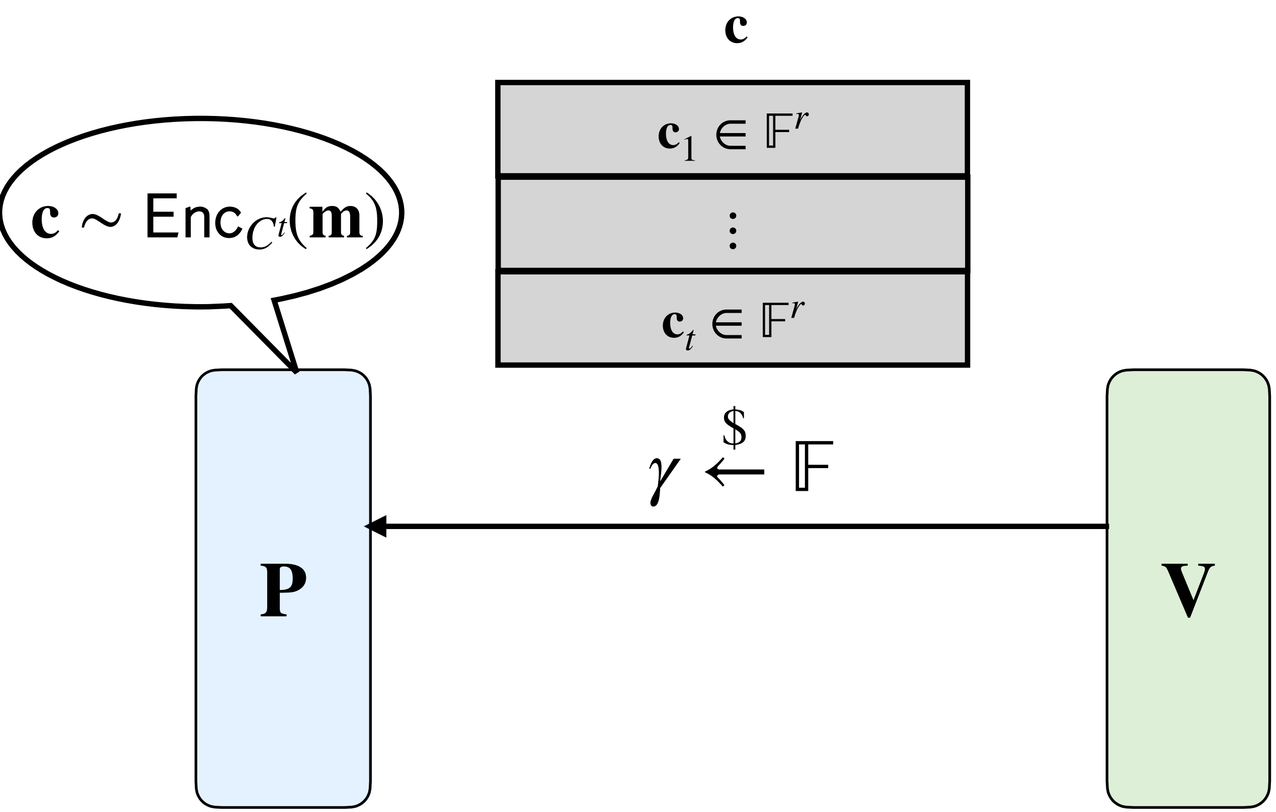


Rate $\rho_{C^t} = s/r = \rho_C$

Encoding time $T_{C^t} = t \cdot T_C$

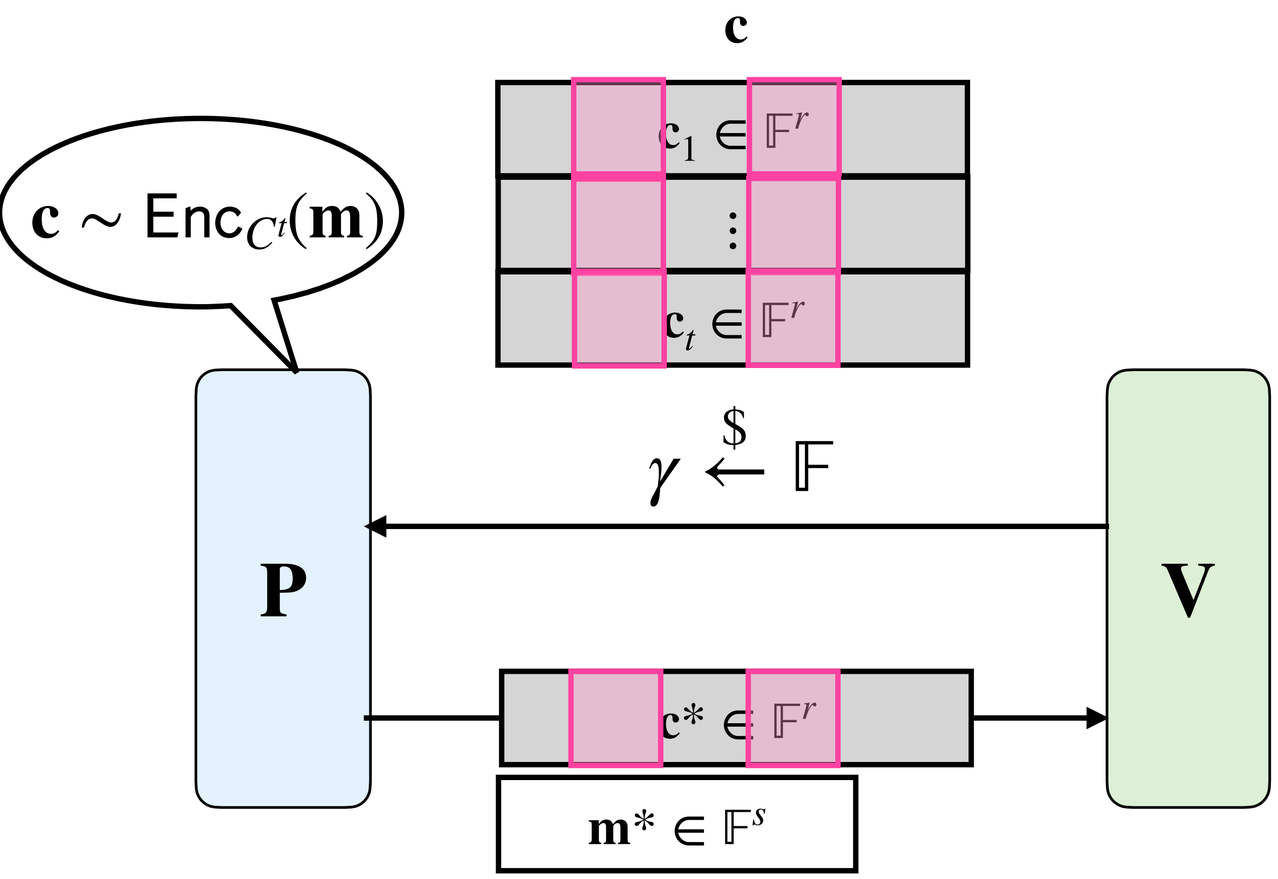
Distance $\delta_{C^t} = \delta_C/t$

IOPP for Interleaved Codes [AHIV17]



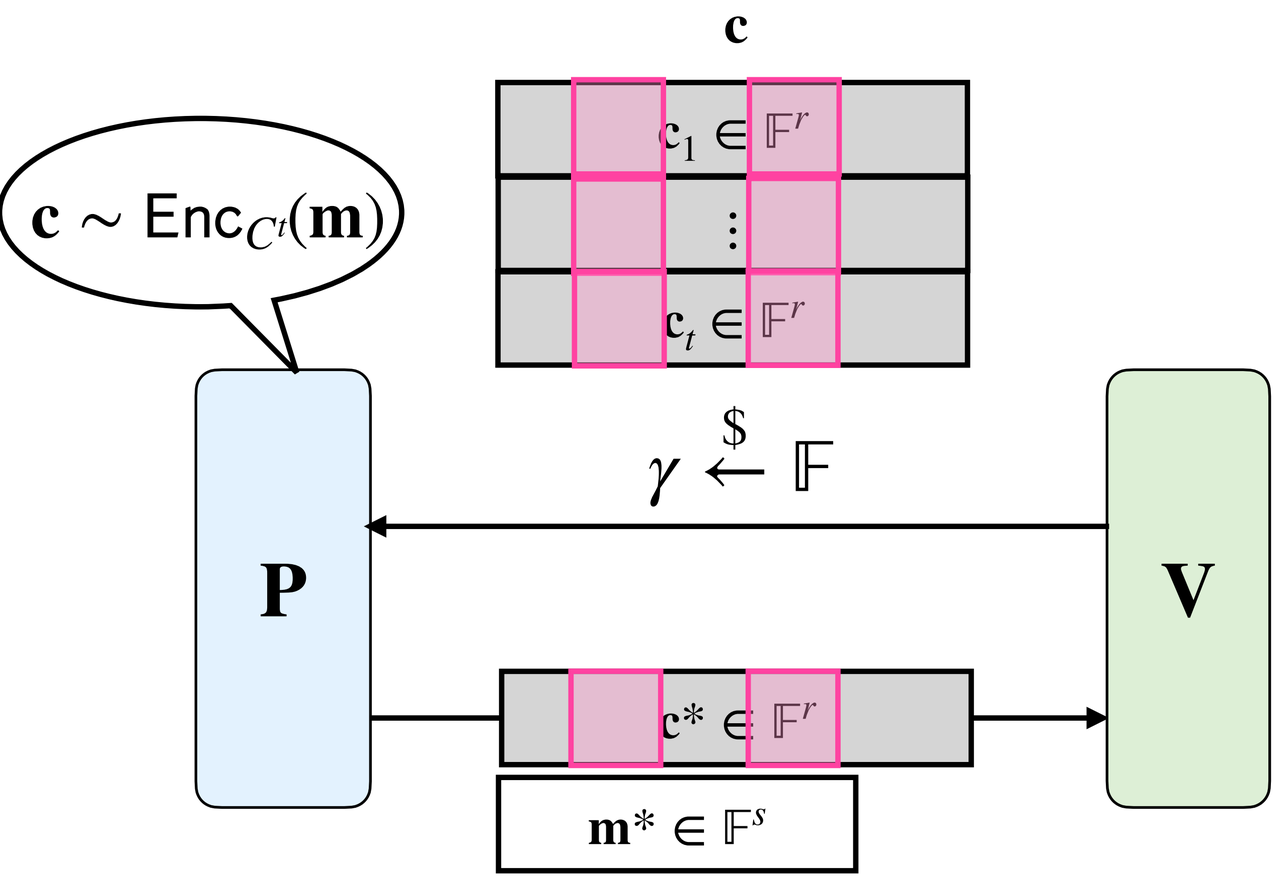
1. **V** sends $\gamma \xleftarrow{\$} \mathbb{F}$

IOPP for Interleaved Codes [AHIV17]



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$ **spot checks:**
   1. $\mathbf{V}$ samples ν indices $j_1, \ldots, j_\nu$.
   2. For each index j:
      1. $\mathbf{V}$ queries $\mathbf{c}_1[j], \ldots, \mathbf{c}_t[j]$ and $\mathbf{c}^*[j]$
      2. $\mathbf{V}$ checks that $\mathbf{c}^*[j] = \sum_i \gamma^i \cdot \mathbf{c}_i[j]$
4. $\mathbf{V}$ checks that $\mathbf{c}^*$ is close to a codeword in C by checking that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$

IOPP for Interleaved Codes [AHIV17]



Completeness:

$\mathbf{P}$ is honest, so $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$, so spot checks pass

Honest $\mathbf{P}$ can set $\mathbf{m}^* = \sum_i \gamma^i \mathbf{m}_i$

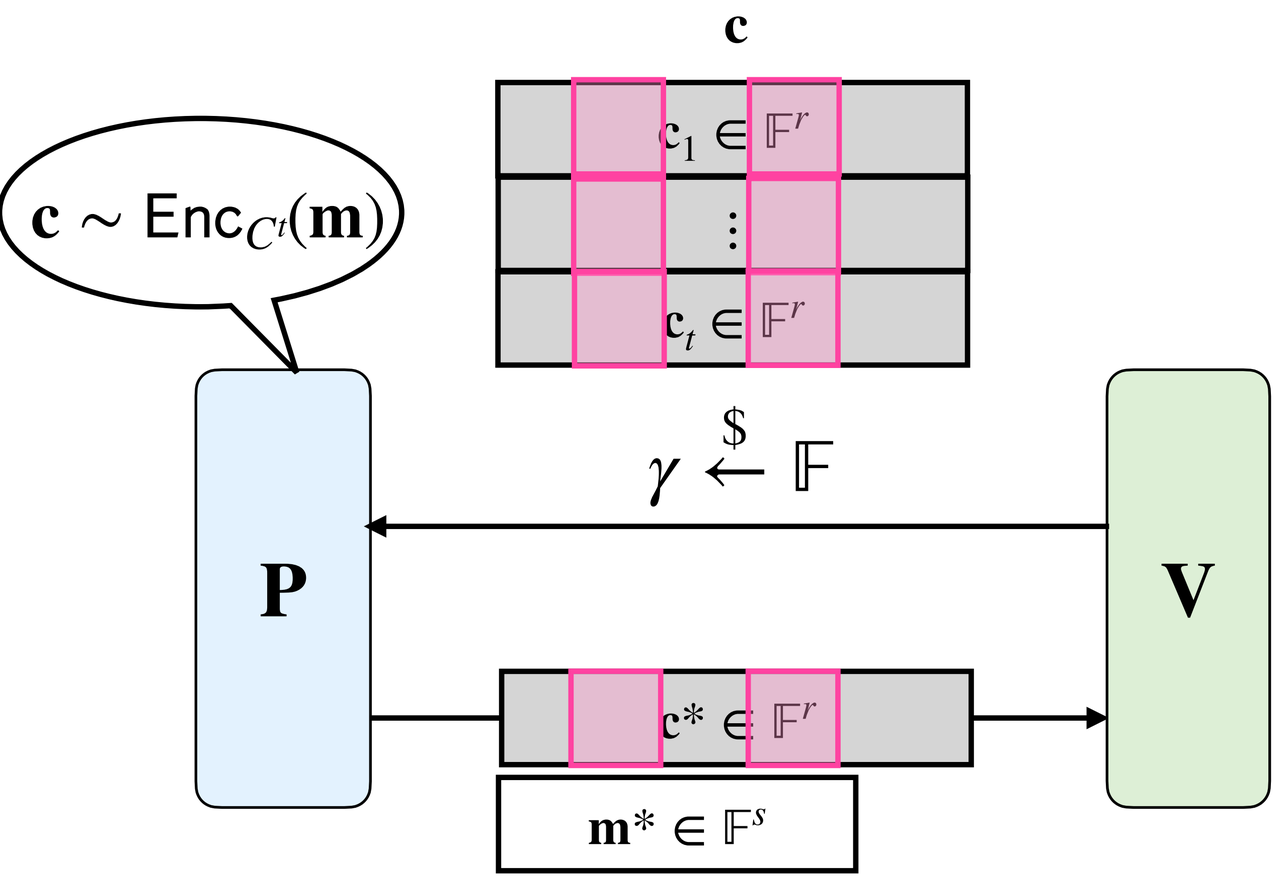
Since C is linear, $\mathsf{Enc}_C(\mathbf{m}^*) = \mathsf{Enc}_C(\sum_i \gamma^i \mathbf{m}_i)$

$$= \sum_i \gamma^i \mathsf{Enc}_C(\mathbf{m}_i)$$

$$= \sum_i \gamma^i \mathbf{c}_i$$

$$= \mathbf{c}^*$$

IOPP for Interleaved Codes [AHIV17]



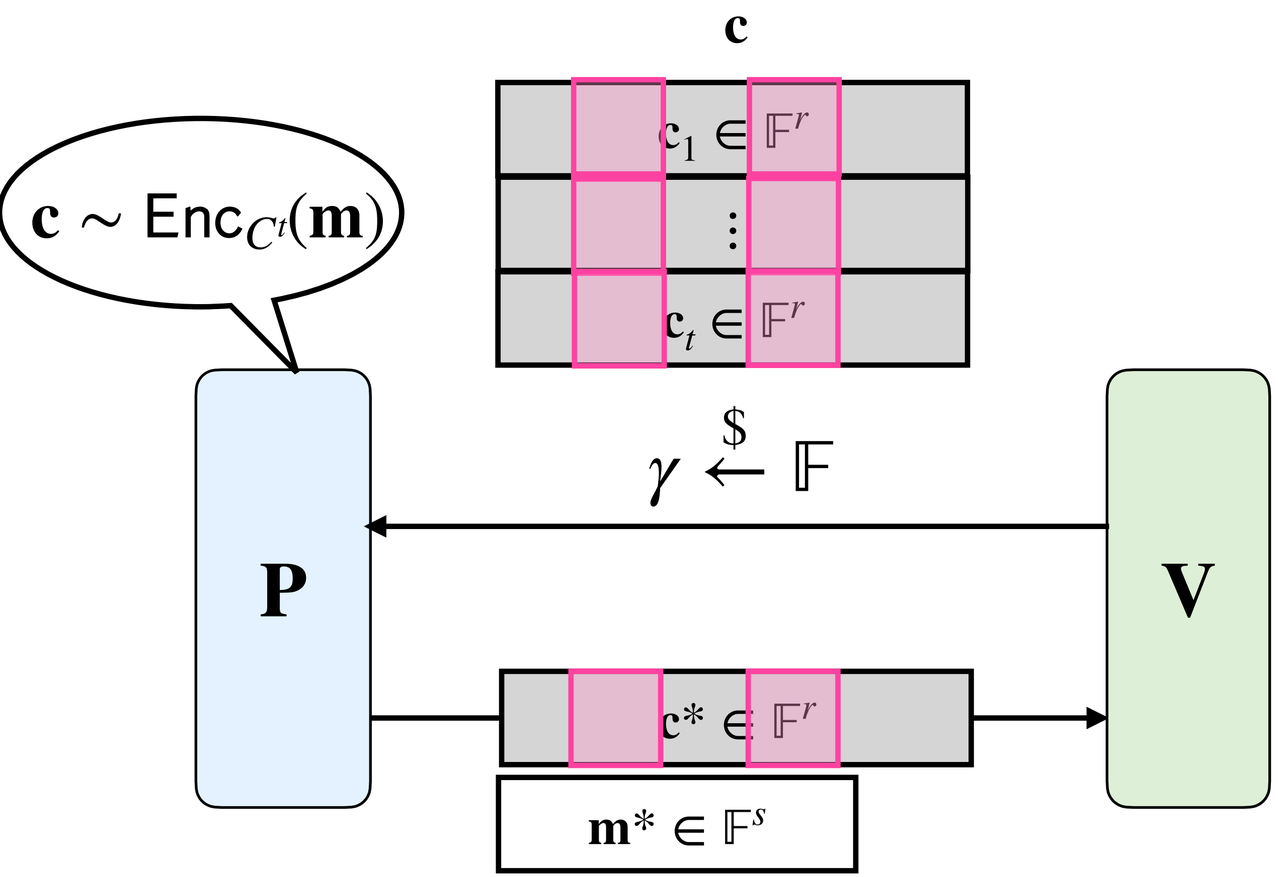
Soundness: $\mathbf{V}$ accepts but at least one $\mathbf{c}_i$ is far from C

Case 0: $\mathbf{c}^*$ is far from C
$\mathbf{V}$ rejects in re-encoding check

Case 1.1: $\mathbf{c}^*$ is close to C and $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$
Happens with negligible probability due to proximity gaps [Dan's talk!].

Case 1.2: $\mathbf{c}^*$ is close to C and far from $\sum_i \gamma^i \mathbf{c}_i$
Spot checks all pass with probability $(1-\delta_C)^\nu$;
set $\nu = -\lambda/\log_2(1-\delta_C)$ to make this negligible.

IOPP for Interleaved Codes [AHIV17]



Efficiency:

prover time: $T_C + O(k)$

queries: $\nu \cdot t + s$ $\mathbb{F}$.

when encoding k-length messages, this is minimized when $t = \sqrt{k/\nu}$ and $s = \sqrt{k\nu}$, resulting in query complexity of

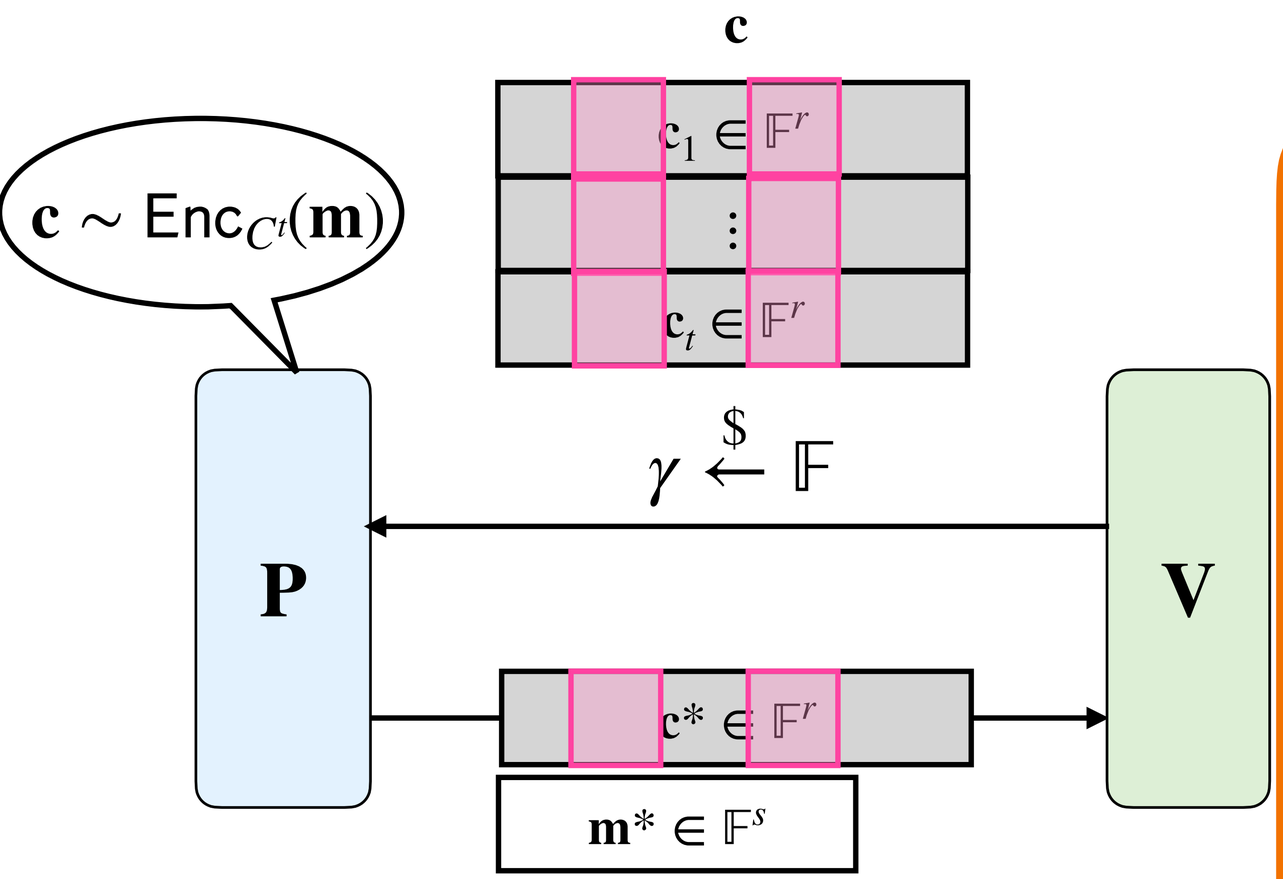
$$O(\sqrt{k\lambda})$$

Chapter 2
Query Reduction by Interleaving Less

Goal:

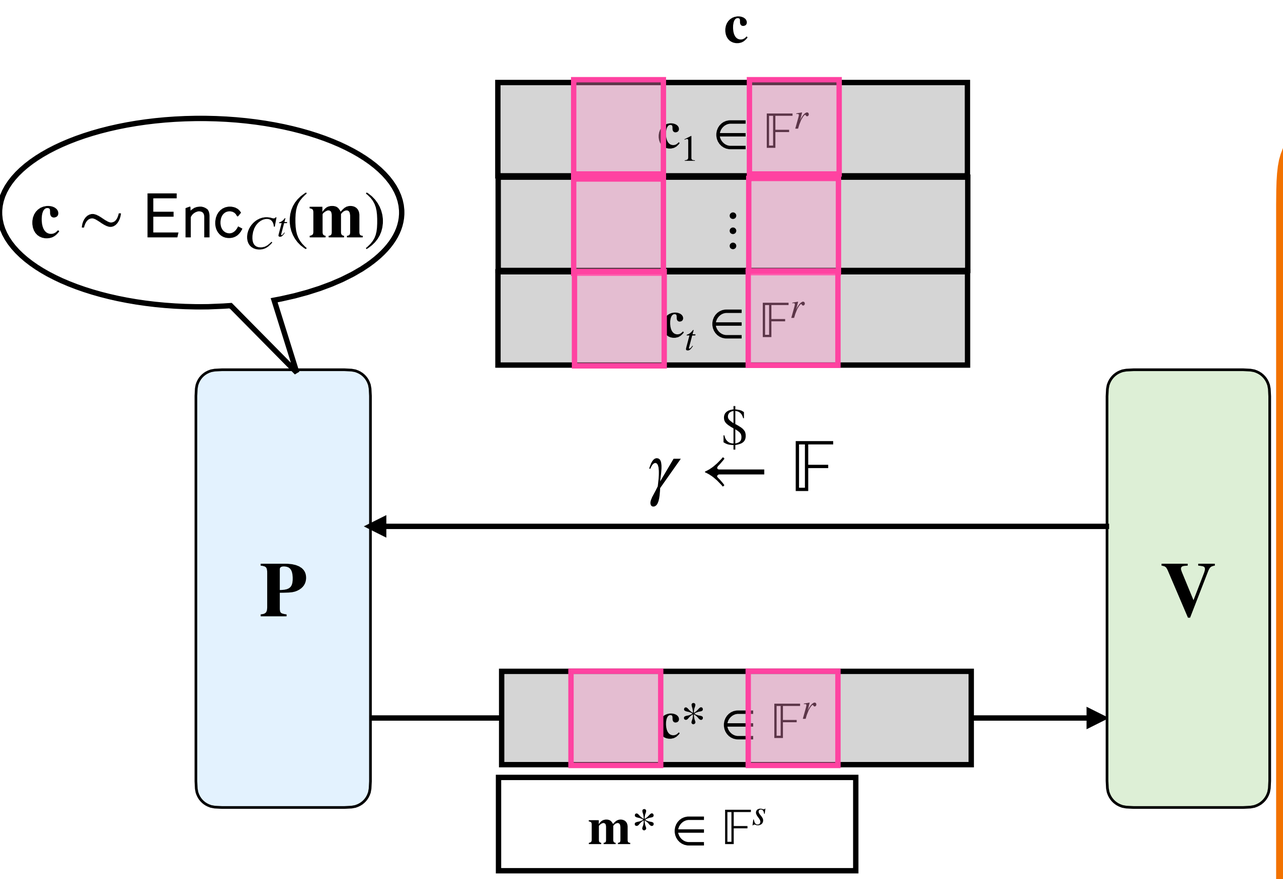
IOPP with $O(\lambda \log k)$ queries

Why is the IOPP Expensive?



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$ **spot checks:**
   1. $\mathbf{V}$ samples ν indices $j_1, \ldots, j_\nu$.
   2. For each index j:
      1. $\mathbf{V}$ queries $\mathbf{c}_1[j], \ldots, \mathbf{c}_t[j]$ and $\mathbf{c}^*[j]$
      2. $\mathbf{V}$ checks that $\mathbf{c}^*[j] = \sum_i \gamma^i \cdot \mathbf{c}_i[j]$
4. $\mathbf{V}$ checks that $\mathbf{c}^*$ is close to a codeword in C by checking that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$

Why is the IOPP Expensive?



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$ **spot checks:**
   1. $\mathbf{V}$ samples ν indices $j_1, \ldots, j_\nu$.
   2. For each index j:
      1. $\mathbf{V}$ queries $\mathbf{c}_1[j], \ldots, \mathbf{c}_t[j]$ and $\mathbf{c}^*[j]$
      2. $\mathbf{V}$ checks that $\mathbf{c}^*[j] = \sum_i \gamma^i \cdot \mathbf{c}_i[j]$
4. $\mathbf{V}$ checks that $\mathbf{c}^*$ is close to a codeword in C by checking that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$

Why is the IOPP Expensive?

1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$ **spot checks:**
    1. $\mathbf{V}$ samples ν indices $j_1, \ldots, j_\nu$.
    2. For each index j:
        1. $\mathbf{V}$ queries $\mathbf{c}_1[j], \ldots, \mathbf{c}_t[j]$ and $\mathbf{c}^*[j]$
        2. $\mathbf{V}$ checks that $\mathbf{c}^*[j] = \sum_i \gamma^i \cdot \mathbf{c}_i[j]$

Interactive "Reduction":
$\mathbf{c}$ is close to C^t
↓
$\mathbf{c}^*$ is close to C

queries: $O(\lambda t)$

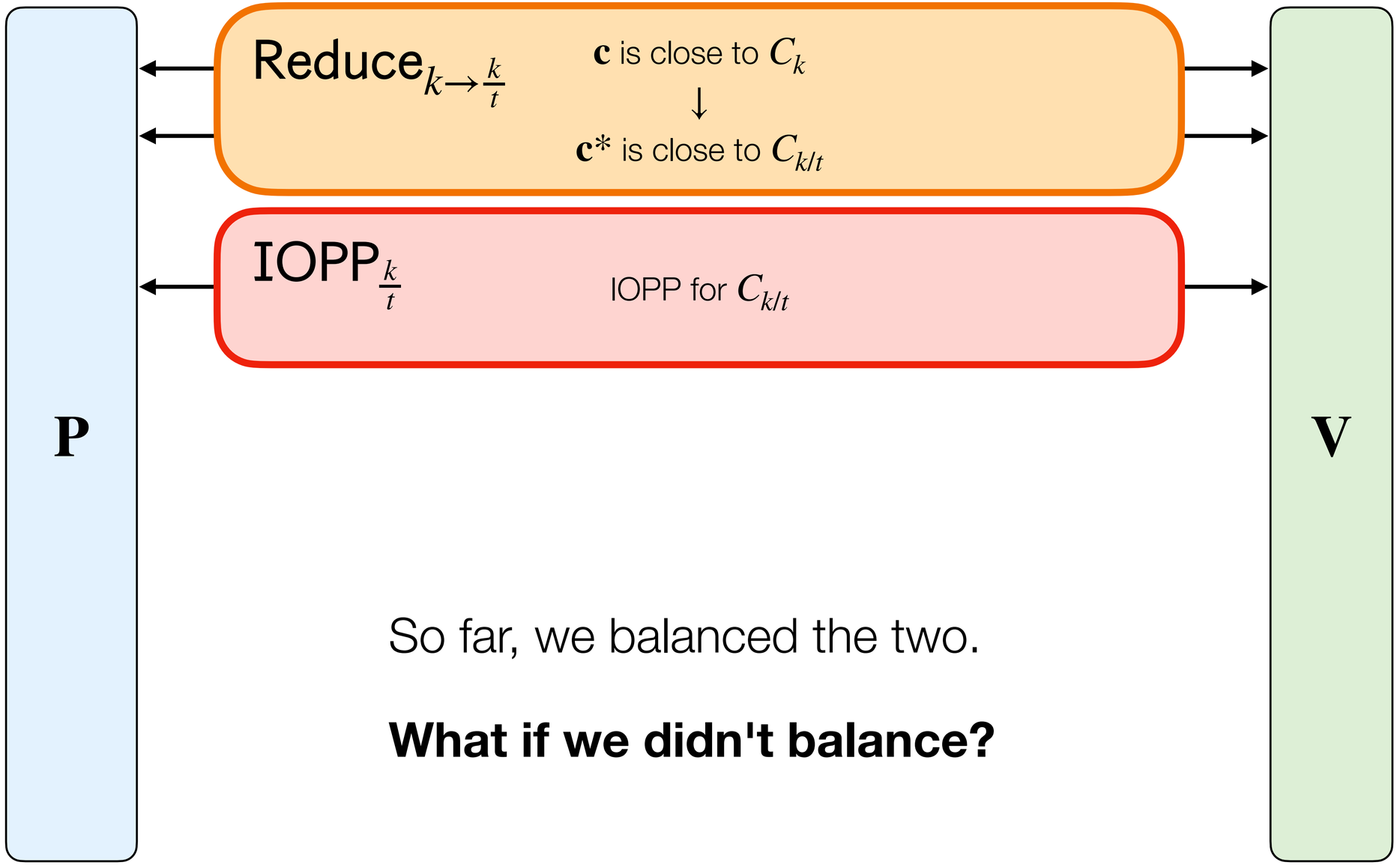
4. $\mathbf{V}$ checks that $\mathbf{c}^*$ is close to a codeword in C by checking that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$

Naive IOPP for C

queries: $O(s)$

$t = \sqrt{k/\lambda}$

Abstract Protocol

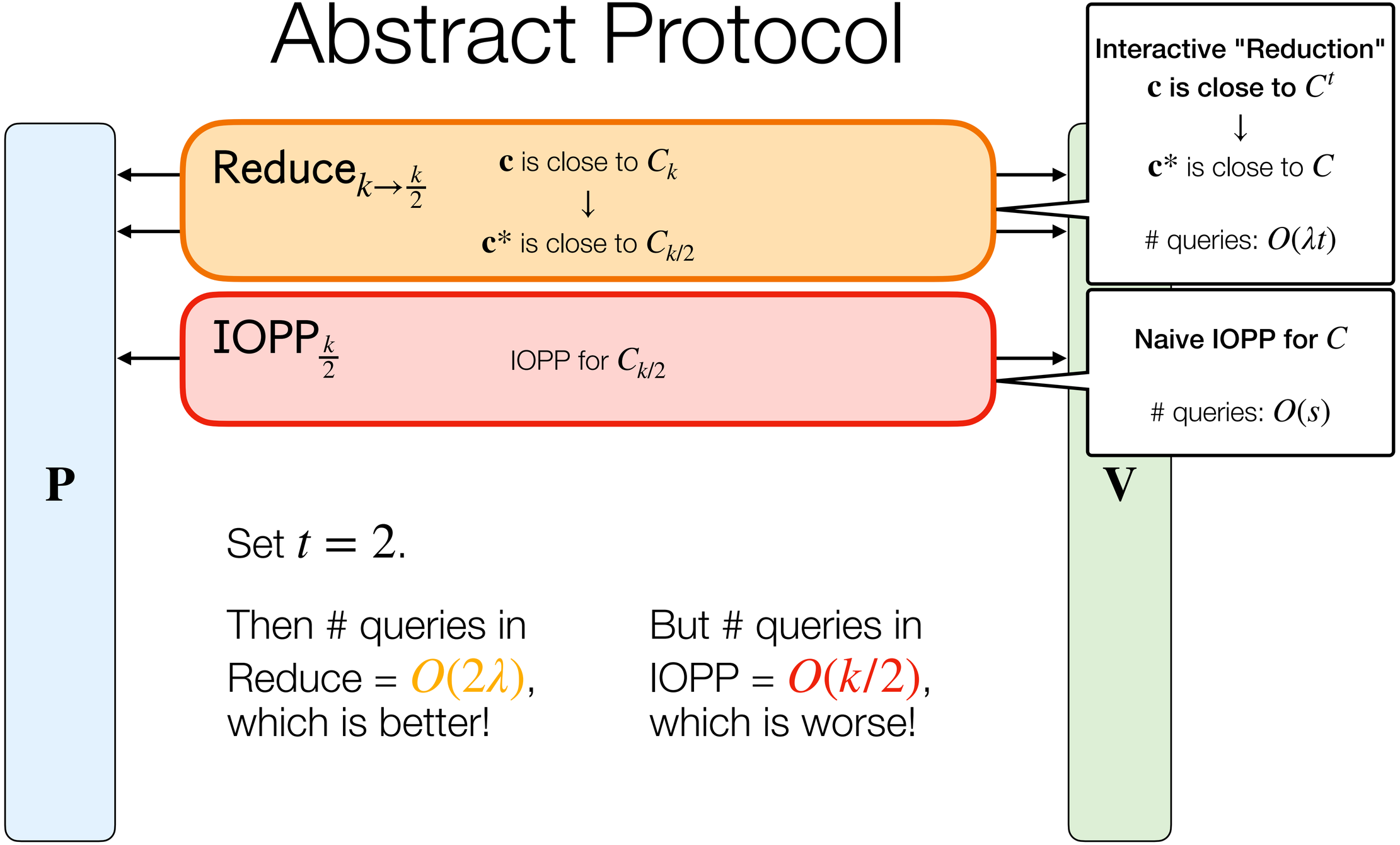


So far, we balanced the two.

What if we didn't balance?

$t = 2$

Abstract Protocol



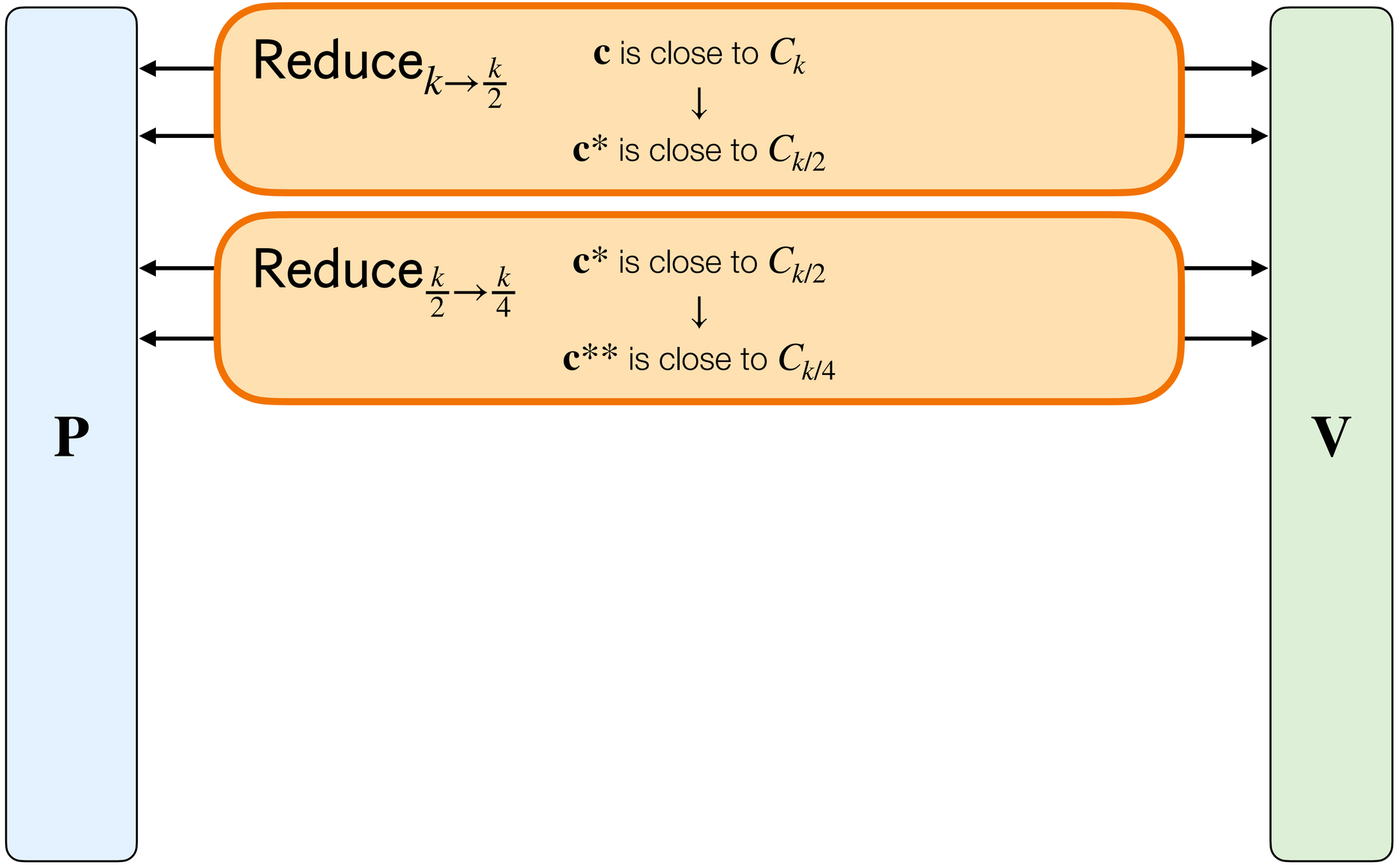
Set $t = 2$.

Then # queries in Reduce = $O(2\lambda)$, which is better!

But # queries in IOPP = $O(k/2)$, which is worse!

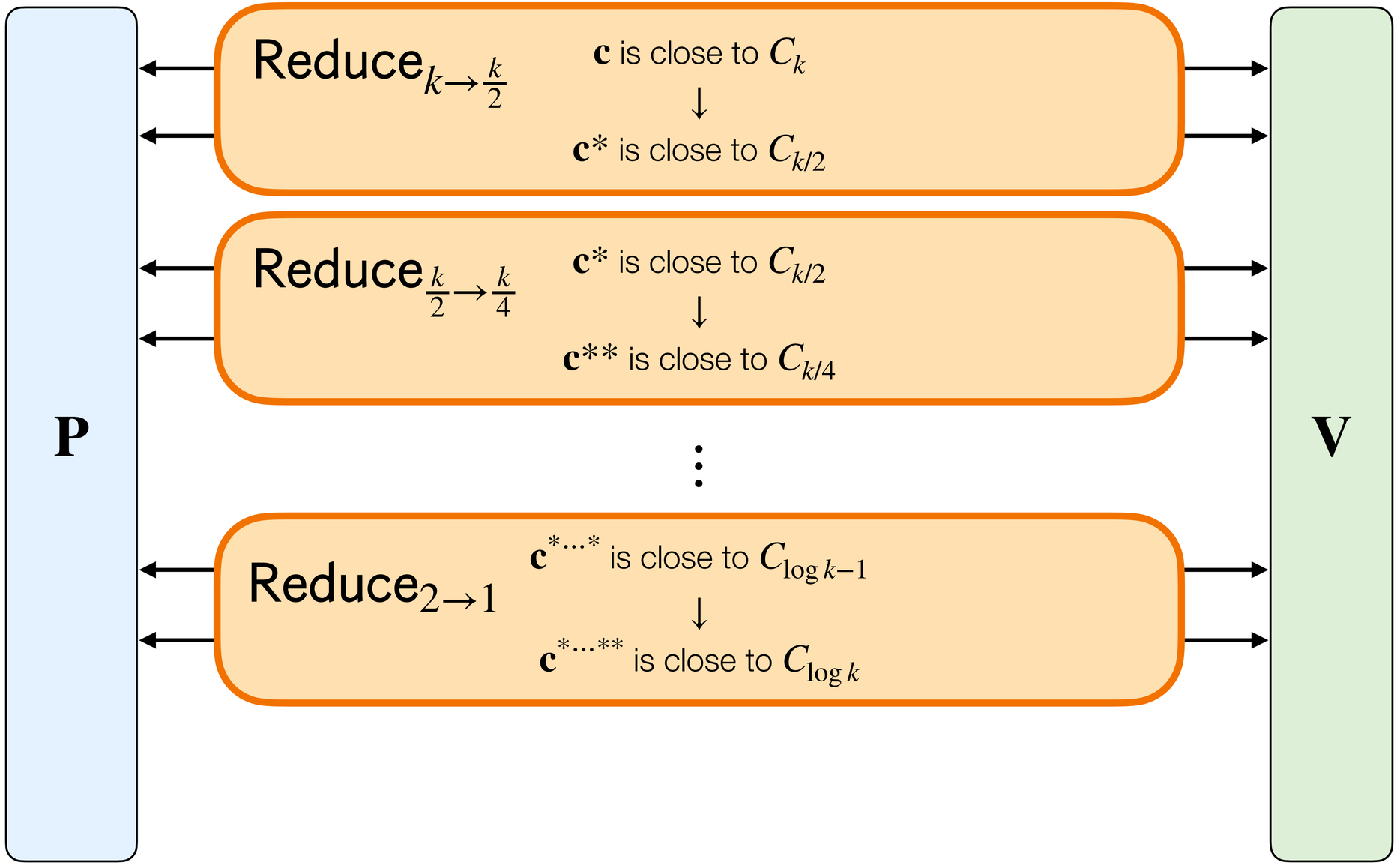
$t = 2$

Recurse!


P
Reduce$_{k \to \frac{k}{2}}$
$\mathbf{c}$ is close to C_k
↓
$\mathbf{c}^*$ is close to $C_{k/2}$
Reduce$_{\frac{k}{2} \to \frac{k}{4}}$
$\mathbf{c}^*$ is close to $C_{k/2}$
↓
$\mathbf{c}^{**}$ is close to $C_{k/4}$
V

$t = 2$

Recurse!


P
Reduce$_{k \to \frac{k}{2}}$
$\mathbf{c}$ is close to C_k
↓
$\mathbf{c}^*$ is close to $C_{k/2}$
Reduce$_{\frac{k}{2} \to \frac{k}{4}}$
$\mathbf{c}^*$ is close to $C_{k/2}$
↓
$\mathbf{c}^{**}$ is close to $C_{k/4}$
Reduce$_{2 \to 1}$
$\mathbf{c}^{*\cdots*}$ is close to $C_{\log k - 1}$
↓
$\mathbf{c}^{*\cdots**}$ is close to $C_{\log k}$
V

Codes and Reductions

Consider the following codes:

$B_{\frac{k}{2}}$ with msg. length $k/2$

$B_{\frac{k}{4}}$ with msg. length $k/4$

$\vdots$

B_1 with msg. length 1

Set $C_i = B^2_{i/2}$

[RR24, RR25, BCFRRZ25, NST25, BMMS26b]

$O(k/2^i)$ prover time and $O(\lambda)$ query IOR for codeswitchable B_i and arbitrary output code.

Reduce $\frac{k}{2^i} \to \frac{k}{2^{i+1}}$

reduce size

Fold

prox to $C_{\frac{k}{2^i}}$ → prox to $B_{\frac{k}{2^{i+1}}}$

+

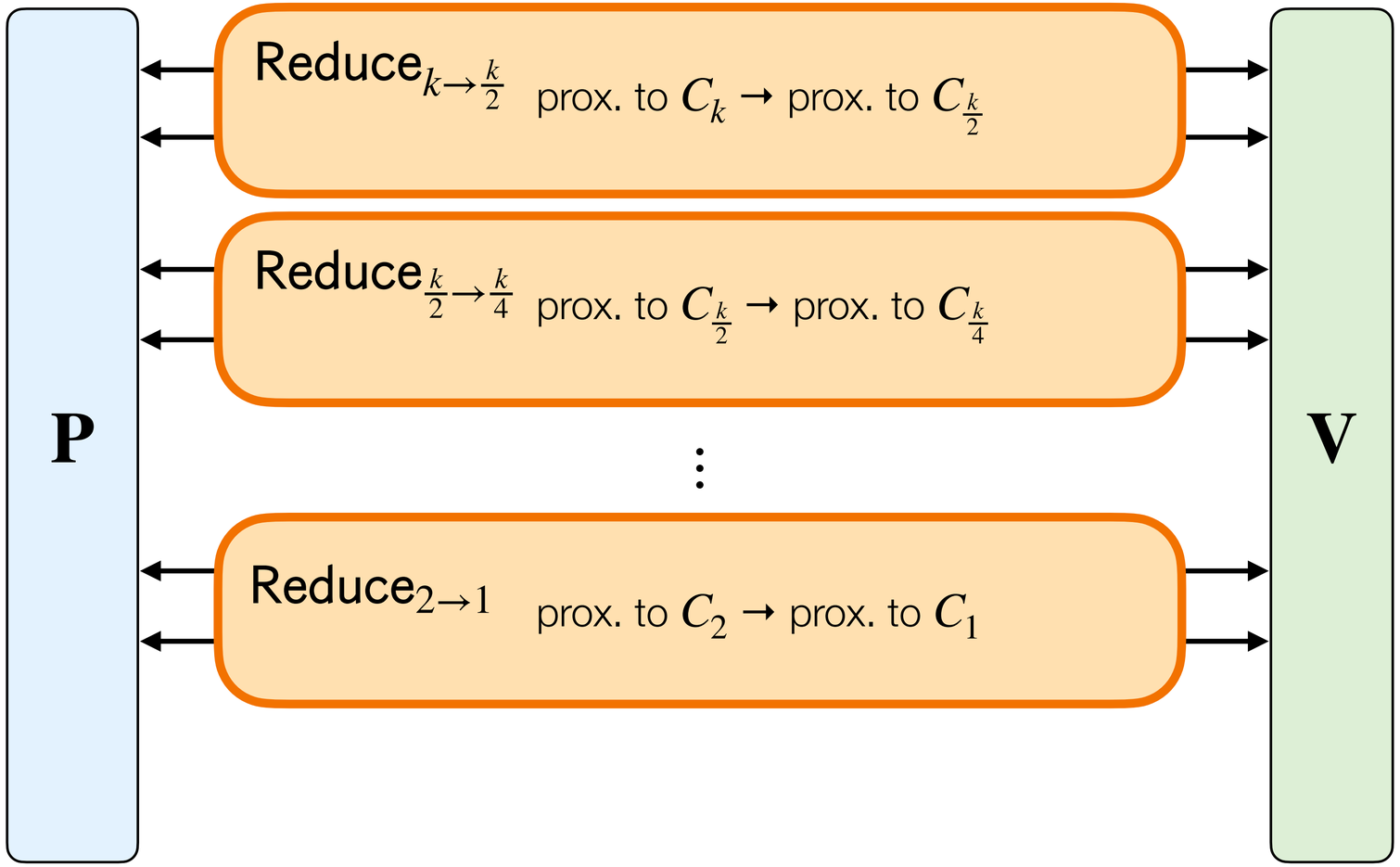
hop to foldable code

Codeswitch

prox to $B_{\frac{k}{2^{i+1}}}$ → prox to $C_{\frac{k}{2^{i+1}}}$

$t = 2$

IOPP Completeness and Soundness

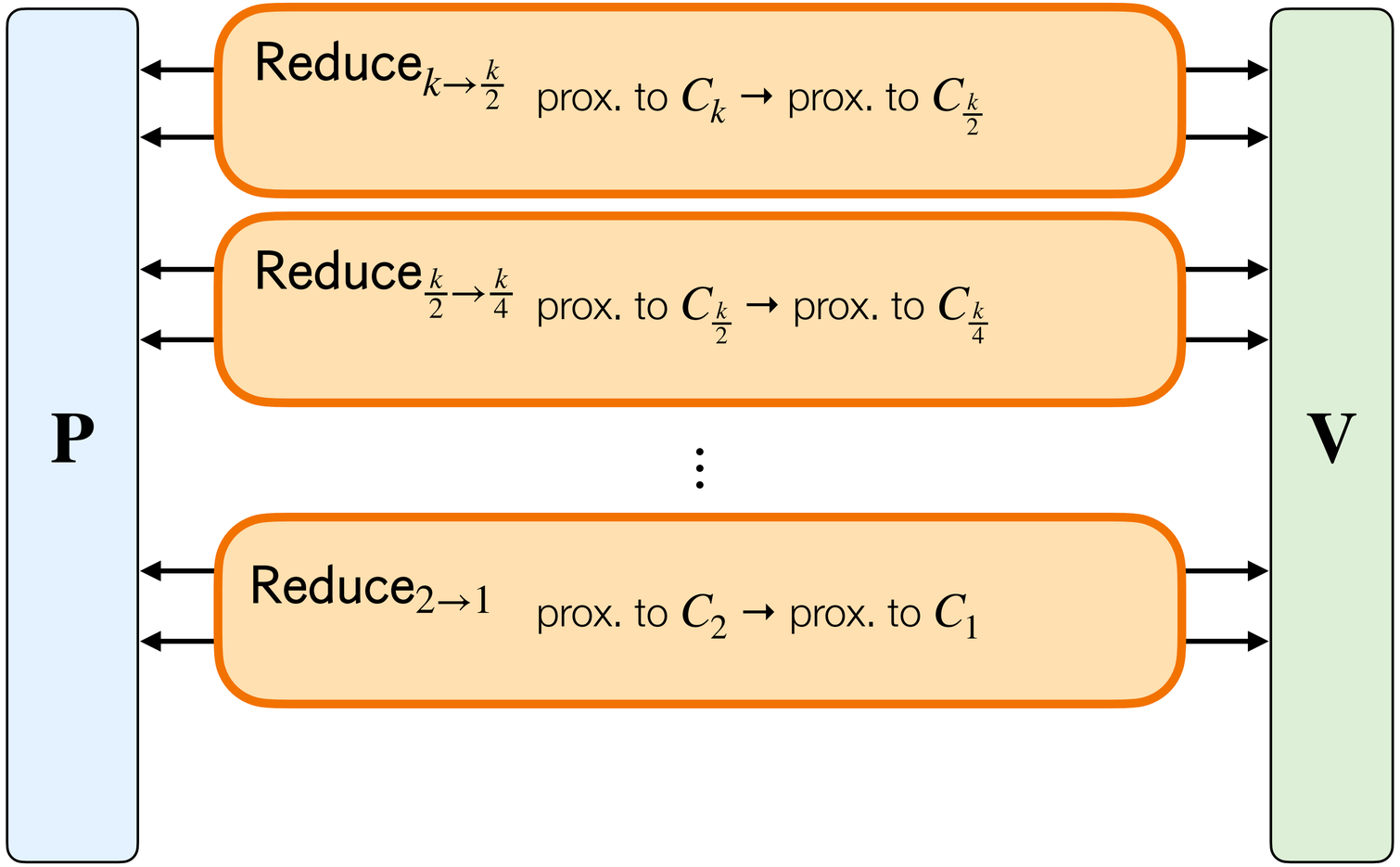


Follow from *composition* theorems for interactive oracle reductions

[B**M**NW25, B**M**MS26b]

$t = 2$

IOPP Efficiency



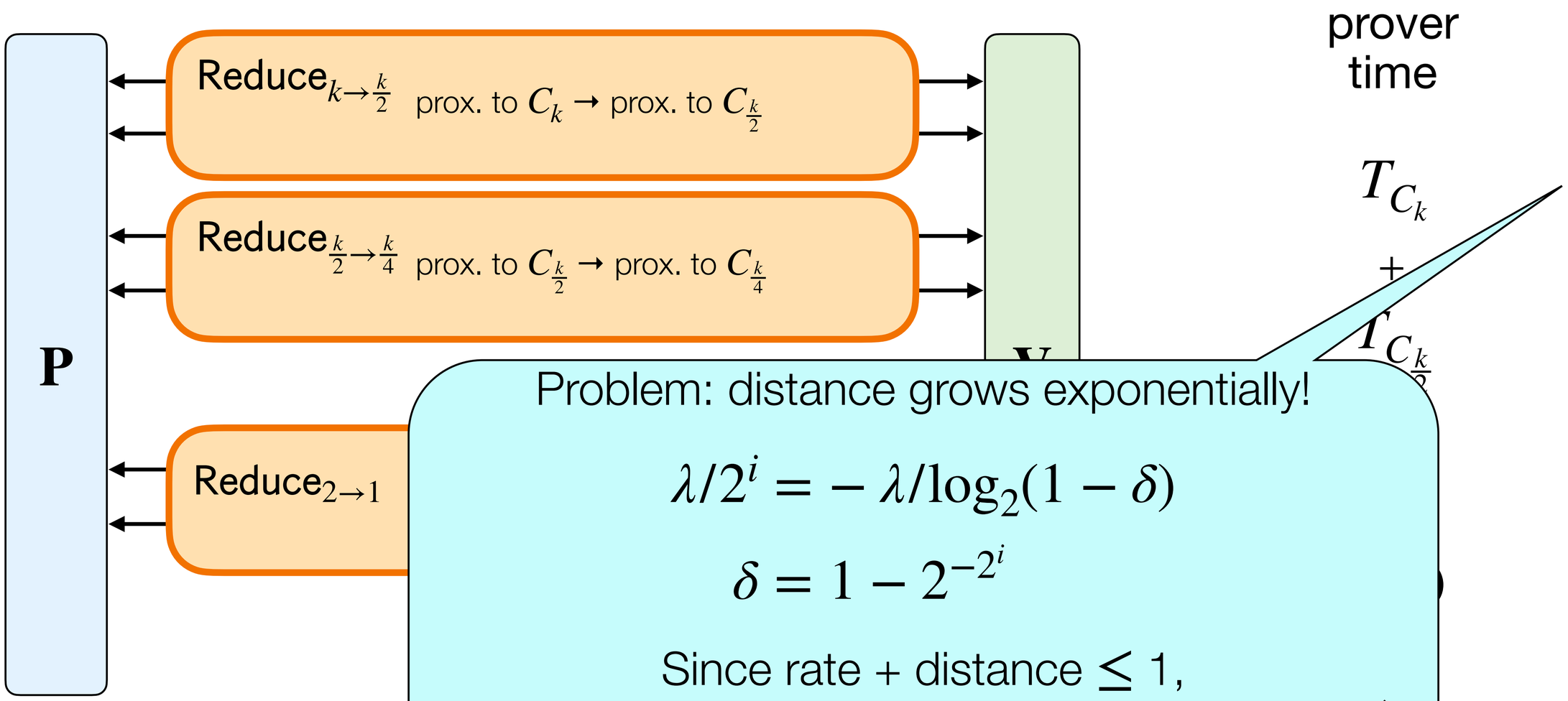
prover time	# queries
T_{C_k}	λ queries
+	+
$T_{C_{\frac{k}{2}}}$	λ queries
+	+
⋮	⋮
+	+
$O(1)$	λ queries
||	||
$O(T_{C_k})$	$O(\lambda \log k)$

Chapter 3
Query Reduction via Improving Distance

Goal:

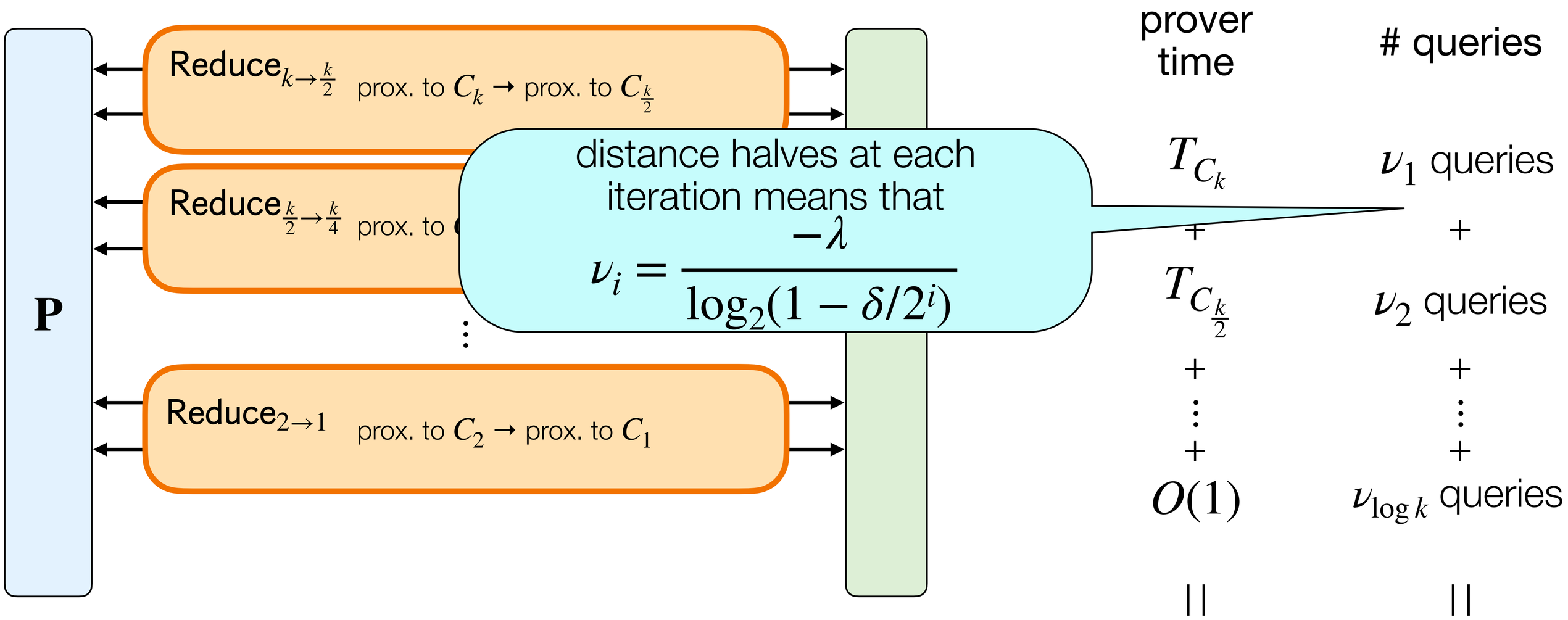
IOPP with $O(\lambda \log \log k)$ queries

Reduce # queries in each round

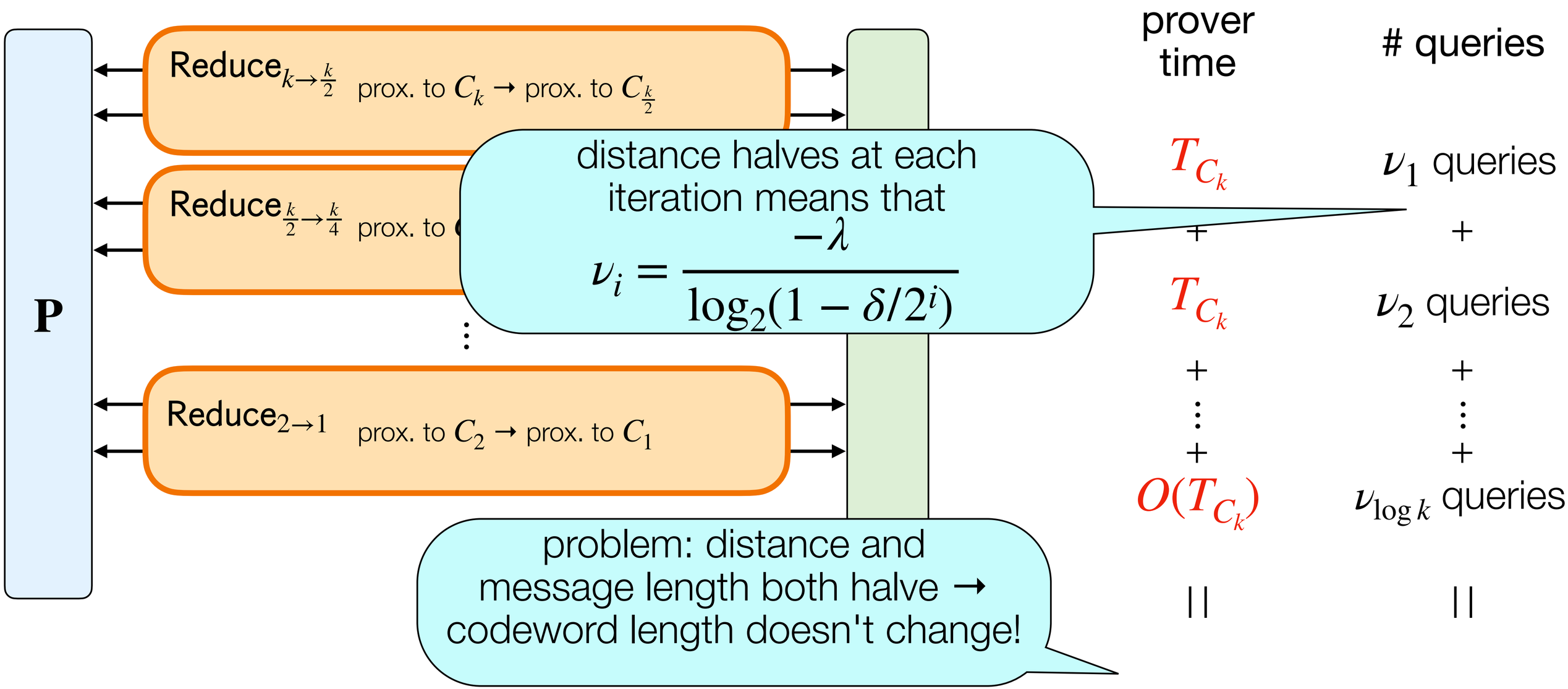


prover time	# queries
T_{C_k}	λ queries
+	+
$T_{C_{\frac{k}{2}}}$	$\frac{\lambda}{2}$ queries
+	+
⋮	⋮
+	+
$O(1)$	$O(1)$ queries
||	||
$O(T_{C_k})$	$O(\lambda)$

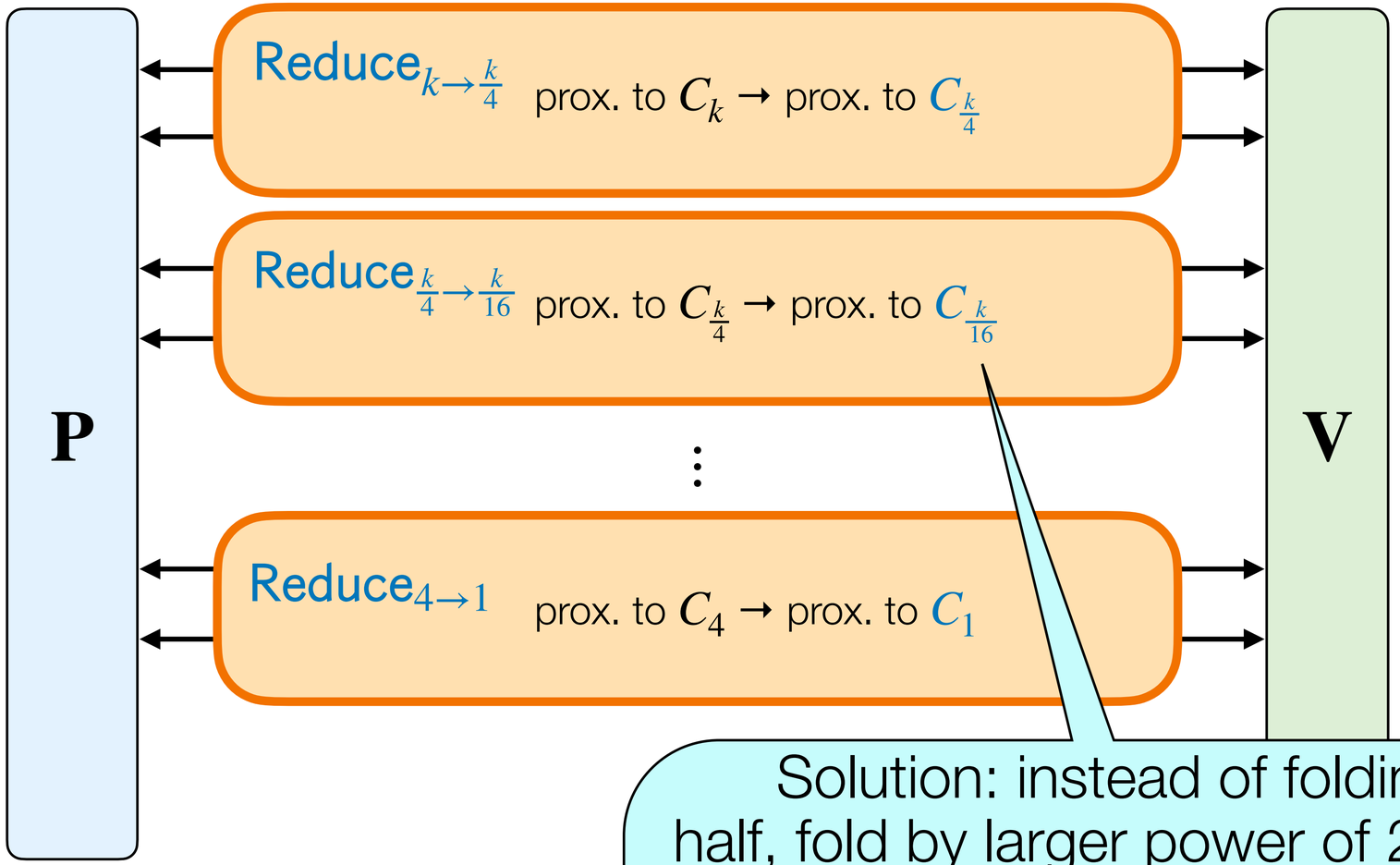
Reduce # queries by increasing distance



Problem: codewords don't shrink!



Solutions: fold more!



Solution: instead of folding in half, fold by larger power of 2, e.g. 4.

Codeword gets smaller faster than distance amplification makes it bigger

prover time	# queries
T_{C_k}	ν_1 queries
+	+
$T_{C_{k/2}}$	ν_2 queries
+	+
⋮	⋮
+	+
$O(T_{C_{k/2^i}})$	$\nu_{\log k}$ queries
||	||
$O(T_{C_k})$	$O(\lambda \log\log k)$

Chapter 4
Round *and* Query Reduction via Tensoring

Goal:

IOPP with $O(\lambda)$ queries

Reducing the Number of Reductions

Problem: Too many reductions, and can't reduce the queries per reduction enough!

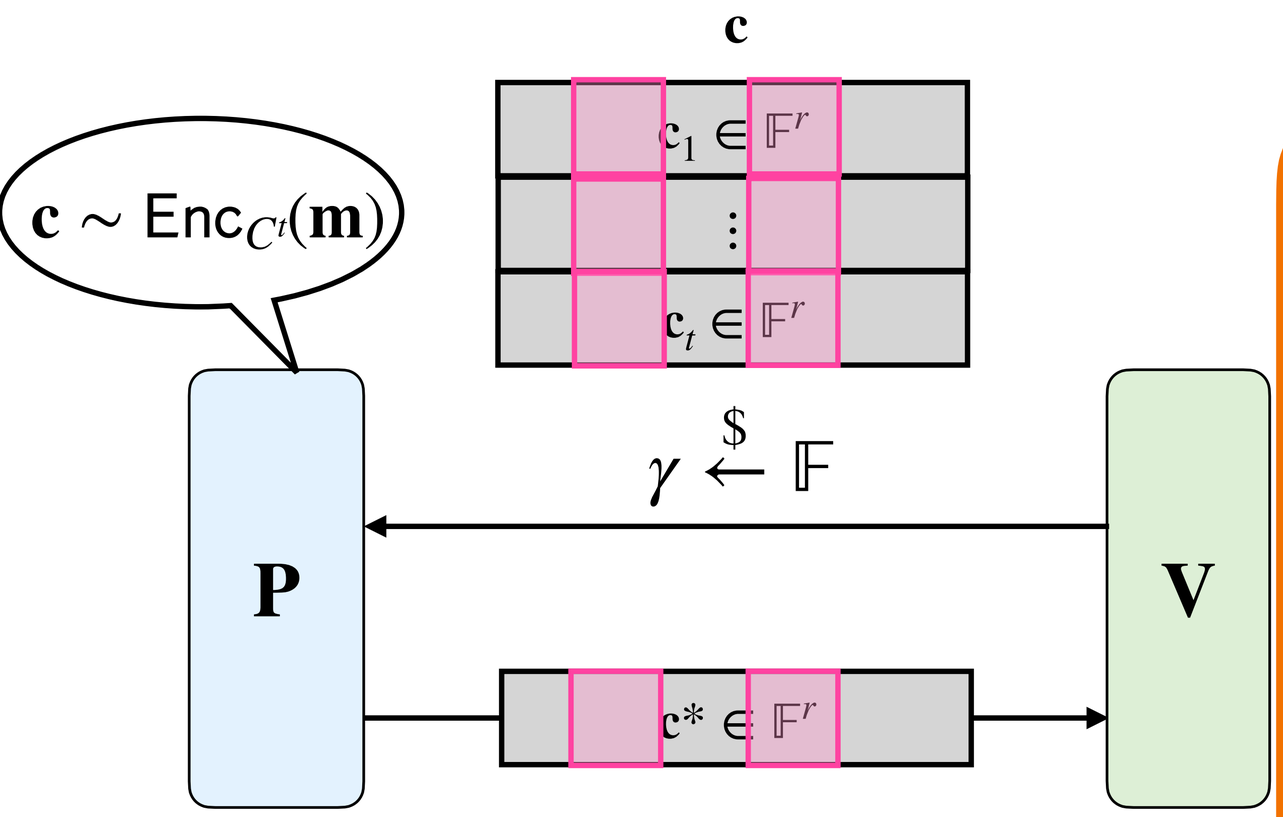
Need: $O(1)$ reductions each with $O(\lambda)$ queries

Good news: suffices to reduce message length to $O(\lambda k^{1/3})$

and codeswitch to RS code with rate $\frac{1}{k^{1/3}}$.

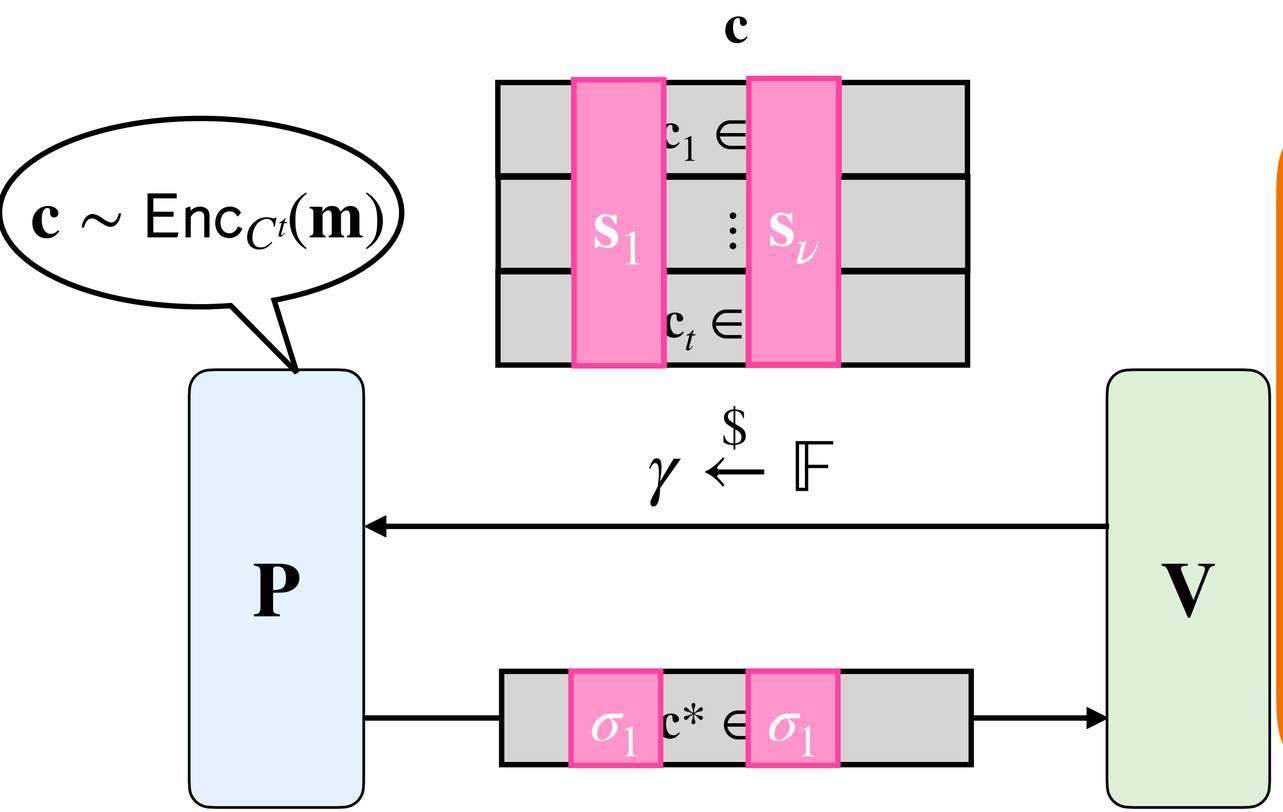
Goal: **Reduce**$_{k \to k^{2/3}}$

Idea 1: Interleave with $t = k^{1/3}$



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$ **spot checks:**
   1. $\mathbf{V}$ samples ν indices $j_1, \ldots, j_\nu$.
   2. For each index j:
      1. $\mathbf{V}$ queries $\mathbf{c}_1[j], \ldots, \mathbf{c}_t[j]$ and $\mathbf{c}^*[j]$
      2. $\mathbf{V}$ checks that $\mathbf{c}^*[j] = \sum_i \gamma^i \cdot \mathbf{c}_i[j]$

Idea 1: Interleave with $t = k^{1/3}$



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{V}$ checks $\mathbf{c}^* = \sum_i \gamma^i \mathbf{c}_i$ via $\nu = O(\lambda)$

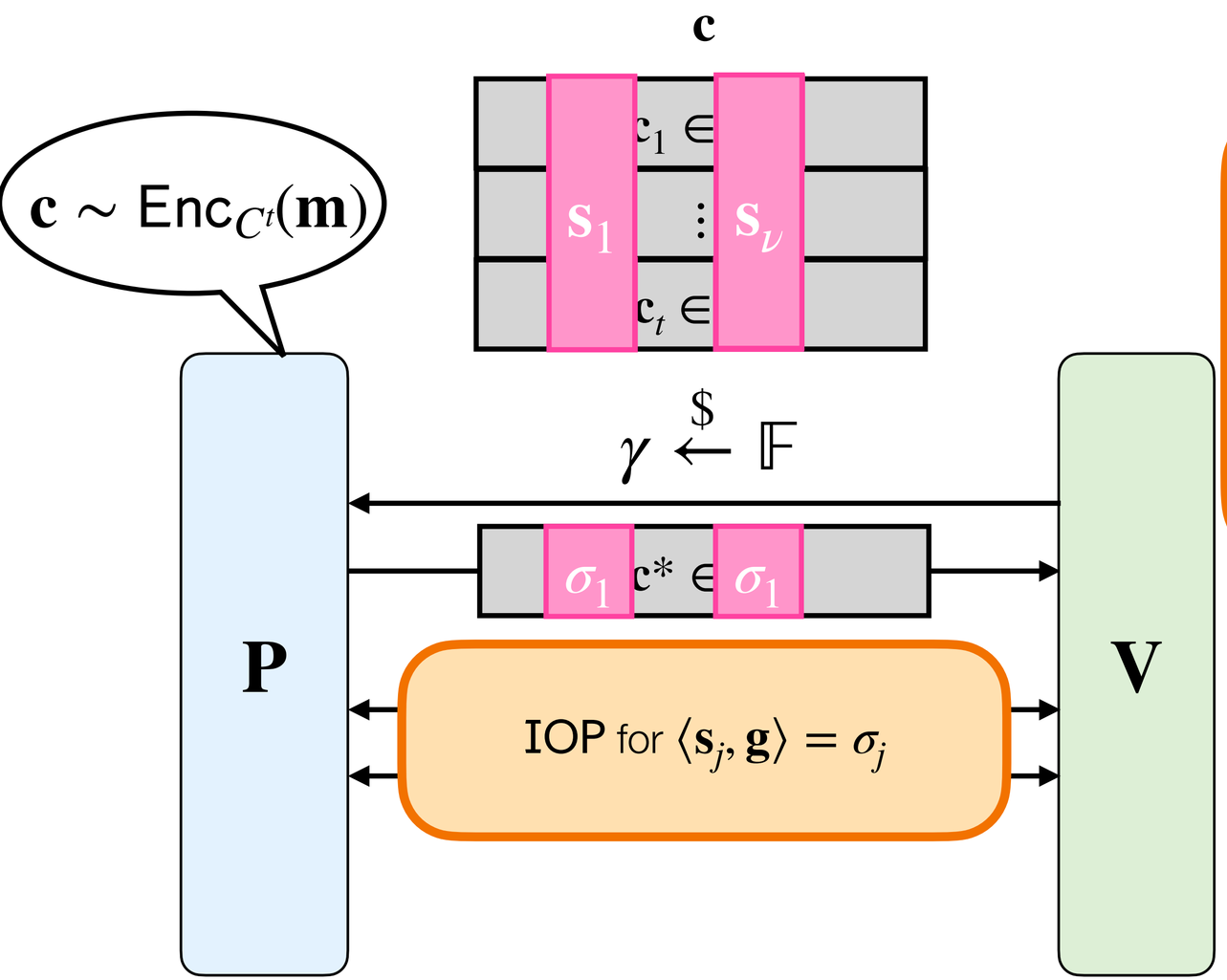
   spot checks:

   for each index j, $\mathbf{V}$ checks $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$,

   where $g = (1, \gamma, \gamma^2, \ldots, \gamma^{\nu-1})$.

Problem: $O(\lambda \cdot k^{1/3})$ queries!

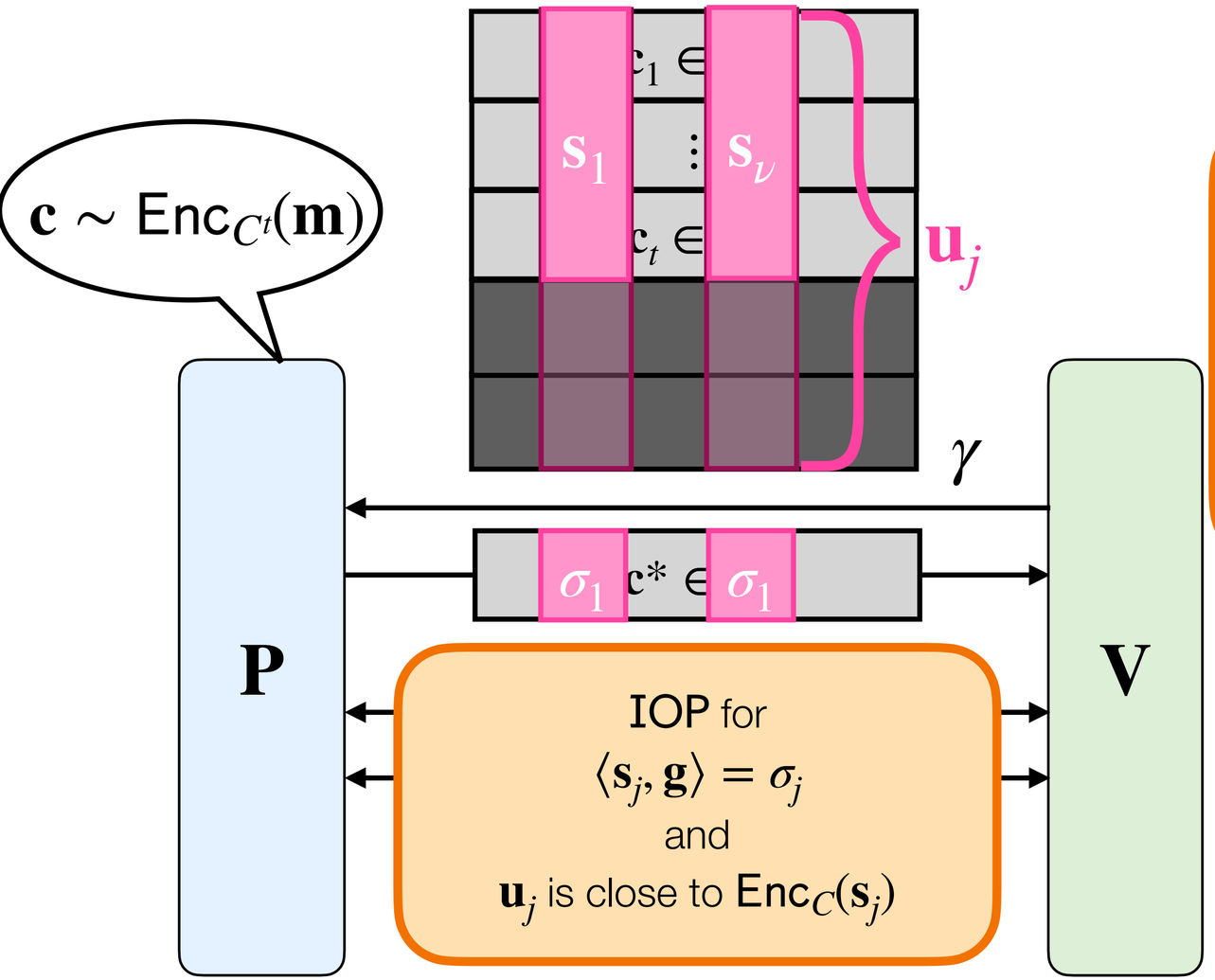
Idea 2: Delegate spot checks to prover



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{P}$ and $\mathbf{V}$ engage in IOP for proving $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$.

Problem: $\mathbf{V}$ must still read input $\mathbf{s}_j$'s

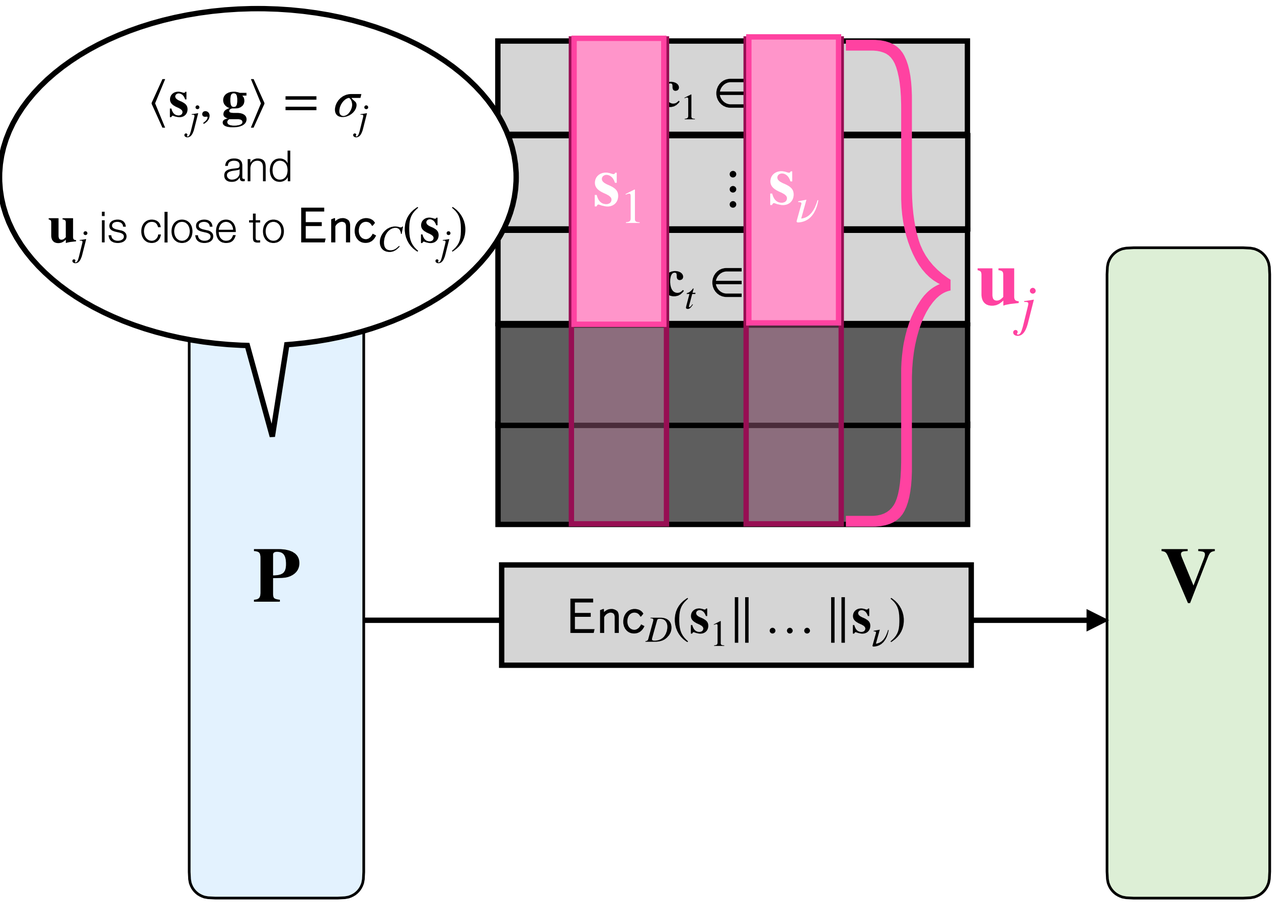
Idea 2.1: Encode Columns Too



1. $\mathbf{V}$ sends $\gamma \xleftarrow{\$} \mathbb{F}$
2. $\mathbf{P}$ sends $\mathbf{c}^*$ that is supposed to be $\sum_i \gamma^i \mathbf{c}_i$ and $\mathbf{m}^*$ such that $\mathbf{c}^* = \mathsf{Enc}_C(\mathbf{m}^*)$
3. $\mathbf{P}$ and $\mathbf{V}$ engage in IOP for proving $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$.

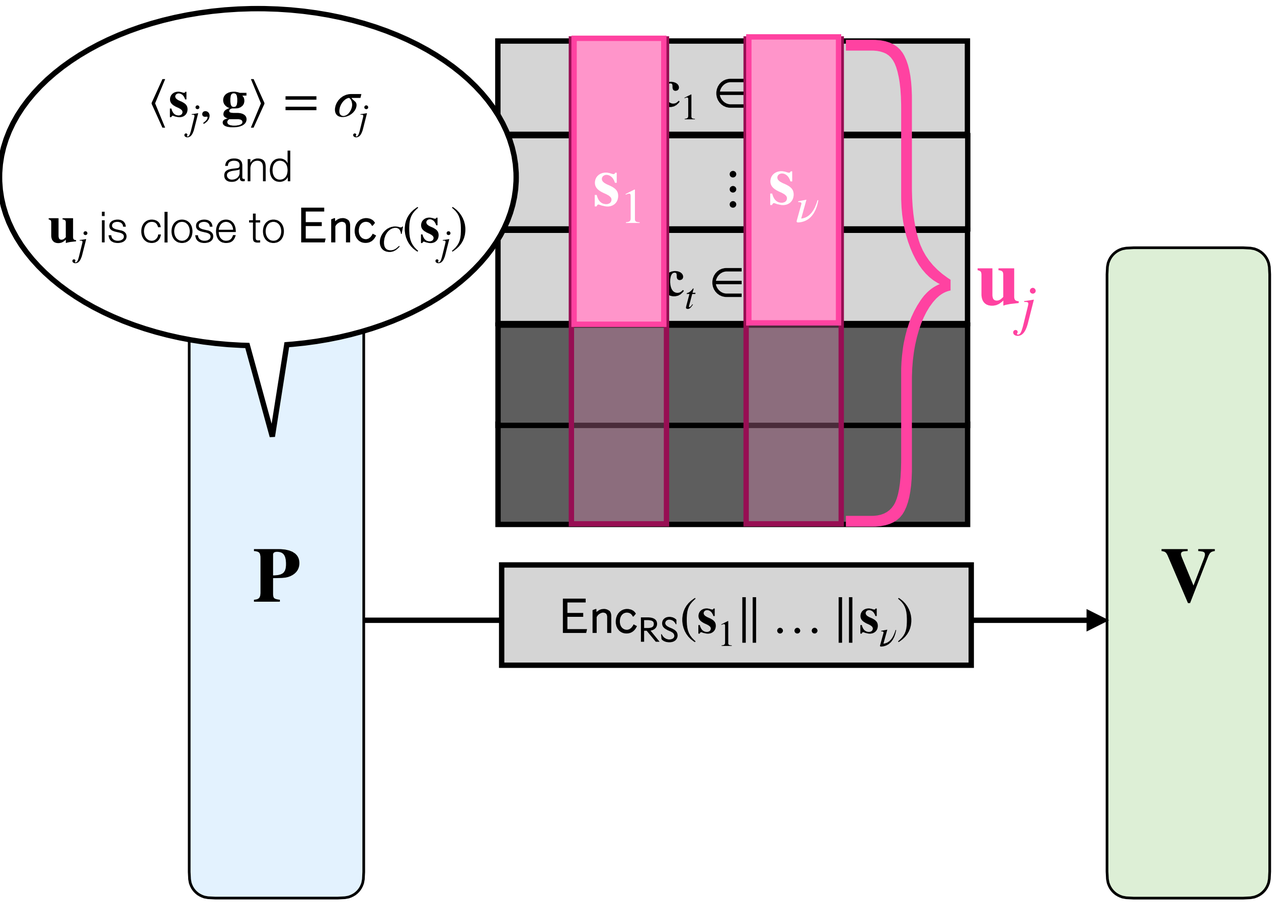
Problem: Naive IOP still incurs $O(\lambda^2)$ queries

IOP for Inner-Products and Proximity



1. $\mathbf{P}$ sends $\mathbf{u} = \mathsf{Enc}_D(\mathbf{s}_1 \| \ldots \| \mathbf{s}_\nu)$ under code D
2. $\mathbf{P}$ convinces $\mathbf{V}$ that:
    1. $\mathbf{u}$ is actually an encoding of $\mathbf{s}_1 \| \cdots \| \mathbf{s}_\nu$:
        1. $\mathbf{u}$ is close to D
        2. Message in $\mathbf{u}$ is consistent with $\mathbf{u}_j$'s
    2. For each j, $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$

IOP for Inner-Products and Proximity

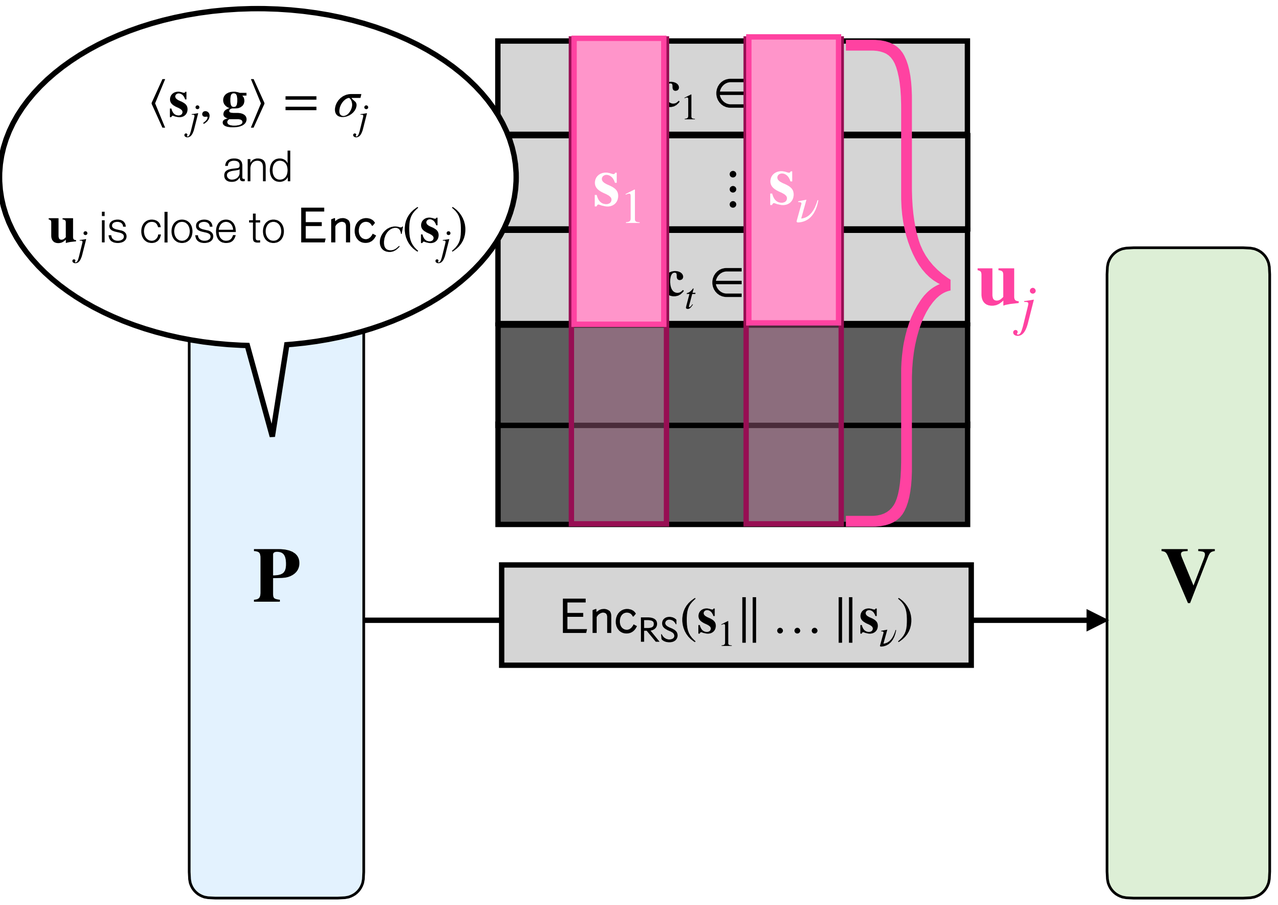


1. $\mathbf{P}$ sends $\mathbf{u} = \mathsf{Enc}_{\mathsf{RS}}(\mathbf{s}_1 \| \ldots \| \mathbf{s}_\nu)$ under RS code
2. $\mathbf{P}$ convinces $\mathbf{V}$ that:
    1. $\mathbf{u}$ is actually an encoding of $\mathbf{s}_1 \| \cdots \| \mathbf{s}_\nu$:
        1. $\mathbf{u}$ is close to D
        2. Message in $\mathbf{u}$ is consistent with $\mathbf{u}_j$'s
    2. For each j, $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$

RS code of msg length $\nu k^{1/3}$ & block length $k^{1/2}$ (this is small enough that encoding is still $\ll k$).

WHIR proves proximity and sumcheck claims; in this distance regime, WHIR requires $O(\lambda)$ queries.

IOP for Inner-Products and Proximity



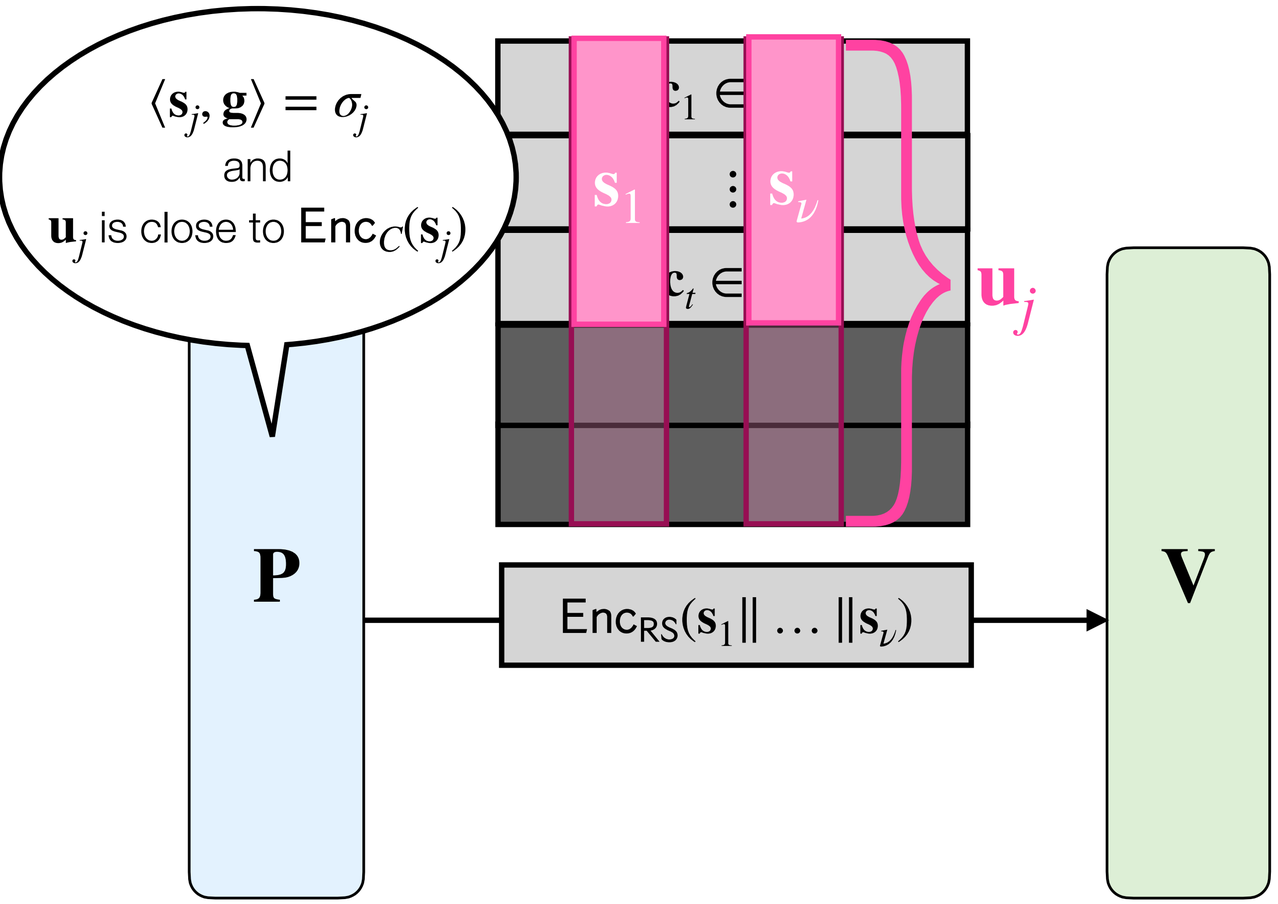
1. $\mathbf{P}$ sends $\mathbf{u} = \mathsf{Enc}_{\mathsf{RS}}(\mathbf{s}_1 \| \ldots \| \mathbf{s}_\nu)$ under RS code
2. $\mathbf{P}$ convinces $\mathbf{V}$ that:
    1. $\mathbf{u}$ is actually an encoding of $\mathbf{s}_1 \| \cdots \| \mathbf{s}_\nu$:
        1. $\mathbf{u}$ is close to D
        2. Message in $\mathbf{u}$ is consistent with $\mathbf{u}_j$'s
    2. For each j, $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$

Attempt 1:

1. $\mathbf{V}$ samples λ random indices per $\mathbf{u}_j$.
2. For each index i, $\mathbf{P}$ proves that $\langle \mathbf{G}_i, \mathbf{s}_j \rangle = \mathbf{u}_j[i]$

Problem: Still $O(\lambda^2)$ queries!

IOP for Inner-Products and Proximity



1. $\mathbf{P}$ sends $\mathbf{u} = \mathsf{Enc}_{\mathsf{RS}}(\mathbf{s}_1 \| \ldots \| \mathbf{s}_\nu)$ under RS code
2. $\mathbf{P}$ convinces $\mathbf{V}$ that:
    1. $\mathbf{u}$ is actually an encoding of $\mathbf{s}_1 \| \cdots \| \mathbf{s}_\nu$:
        1. $\mathbf{u}$ is close to D
        2. Message in $\mathbf{u}$ is consistent with $\mathbf{u}_j$'s
    2. For each j, $\langle \mathbf{s}_j, \mathbf{g} \rangle = \sigma_j$

Attempt 2:

1. $\mathbf{V}$ samples 4ν random columns $\mathbf{u}_j$.
2. Per column, she picks $O(1)$ random indices.
3. For each index i, $\mathbf{P}$ proves $\langle \mathbf{G}_i, \mathbf{s}_{j_i} \rangle = \mathbf{u}_{j_i}[i]$

Efficiency: Just $O(4\lambda)$ queries!

Soundness: ??

Soundness

Assume distance of column code is 1/2.

Claim: Out of 4ν sampled columns $\mathbf{u}_j$, number of 'bad' columns that are far from code is concentrated around average number of bad columns, i.e., 2ν.

Proof: Chernoff Bound asserts that probability that there exist 3ν good columns is negligible.

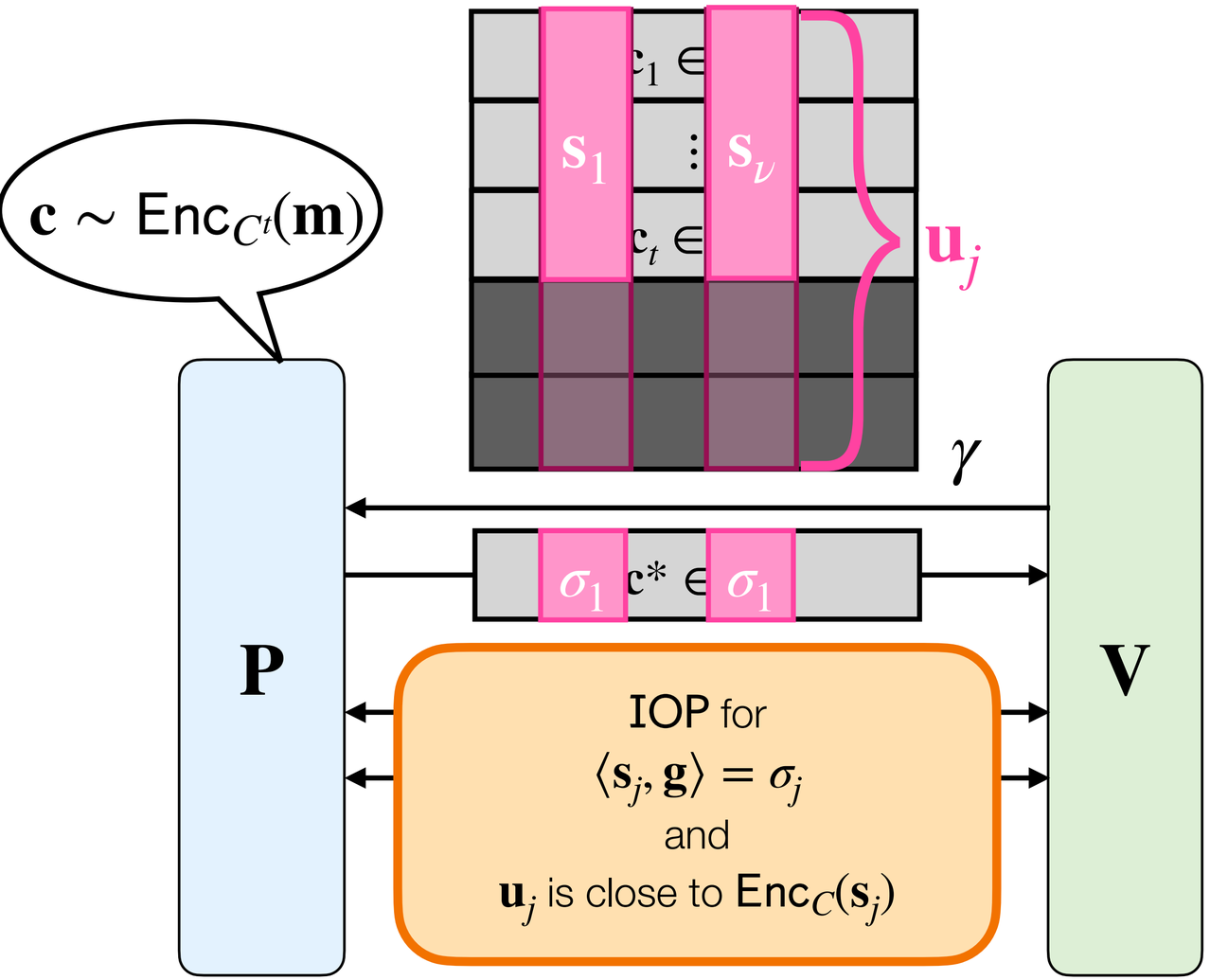
Corollary: With overwhelming probability, at least ν of the $\mathbf{u}_j$'s are bad.

Now, since $\mathbf{V}$ makes $O(1)$ queries per column, and each bad column is at least 1/2-distance away from code, $\mathbf{P}$ gets caught in each bad column with constant probability (say, p).

Combined with corollary, this means that $\mathbf{P}$ avoids detection in each bad column is $(1-p)^{\nu}$.

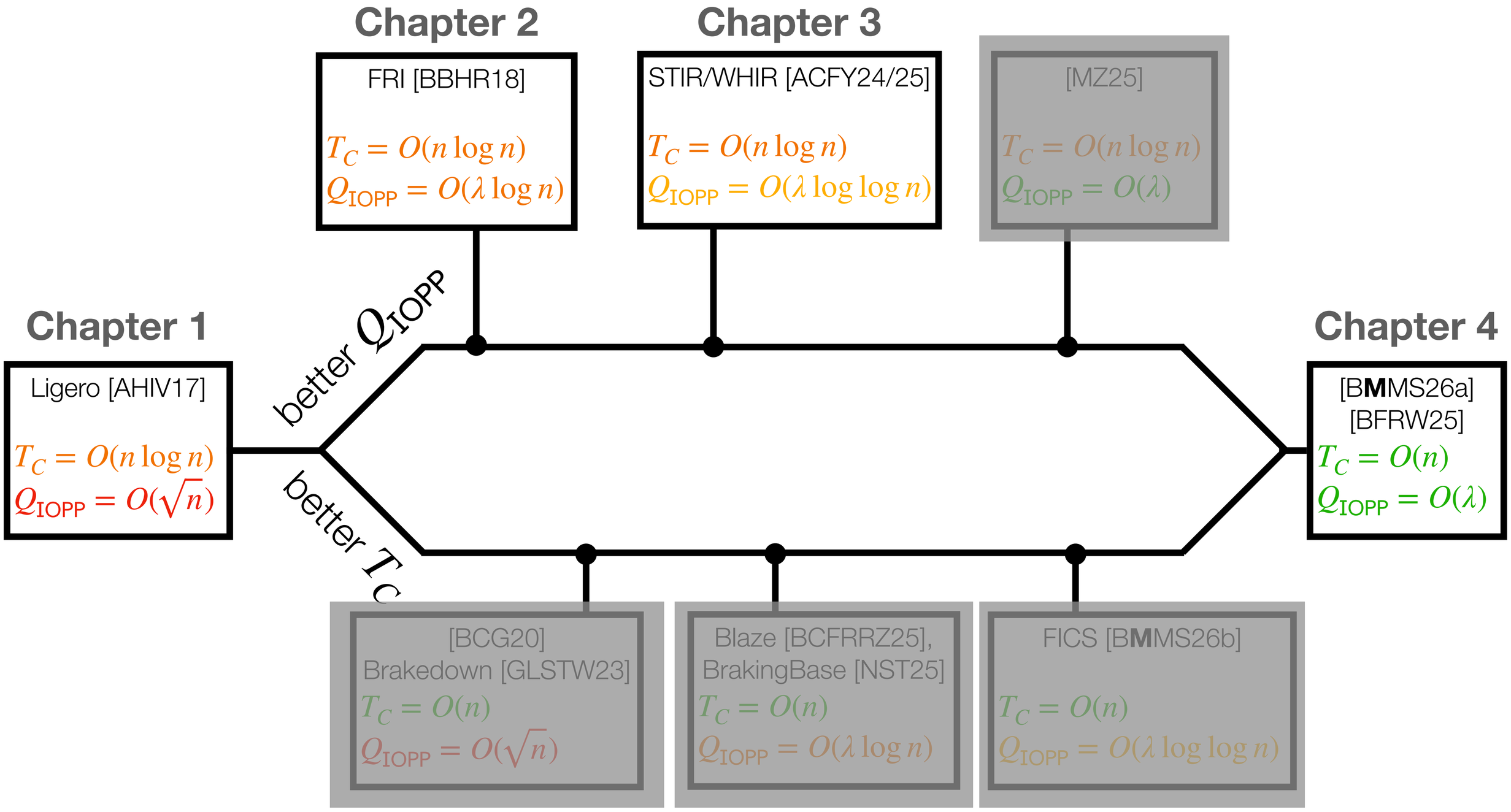
By setting ν appropriately, this is negligible.

Efficiency



Prover time: $T_C + T_{\mathsf{IOP}} = O(T_C)$.

Query complexity: $4\nu + O(\lambda) = O(\lambda)$


Chapter 2
Chapter 3
FRI [BBHR18]
$T_C = O(n \log n)$
$Q_{\text{IOPP}} = O(\lambda \log n)$
STIR/WHIR [ACFY24/25]
$T_C = O(n \log n)$
$Q_{\text{IOPP}} = O(\lambda \log \log n)$
[MZ25]
$T_C = O(n \log n)$
$Q_{\text{IOPP}} = O(\lambda)$
Chapter 1
Ligero [AHIV17]
$T_C = O(n \log n)$
$Q_{\text{IOPP}} = O(\sqrt{n})$
better Q_{IOPP}
better T_C
Chapter 4
[B**M**MS26a]
[BFRW25]
$T_C = O(n)$
$Q_{\text{IOPP}} = O(\lambda)$
[BCG20]
Brakedown [GLSTW23]
$T_C = O(n)$
$Q_{\text{IOPP}} = O(\sqrt{n})$
Blaze [BCFRRZ25],
BrakingBase [NST25]
$T_C = O(n)$
$Q_{\text{IOPP}} = O(\lambda \log n)$
FICS [B**M**MS26b]
$T_C = O(n)$
$Q_{\text{IOPP}} = O(\lambda \log \log n)$