Decoding Polynomials and How to Use Them


Aparna Gupte


Alexandra Henzinger


Ted Pyne


Seyoon Ragavan

BU Security Seminar

October 21, 2026

Roadmap

- What Is a Decoding Polynomial?

Roadmap

- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]

Roadmap

- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]
- New Application to Tree Evaluation [HP**R**26]

Roadmap

- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]
- New Application to Tree Evaluation [HP**R**26]
- New Construction [G**R**26]

Today’s Recipe

Matching vectors



Today’s Recipe

Matching vectors

Decoding polynomials



Today’s Recipe

Matching vectors

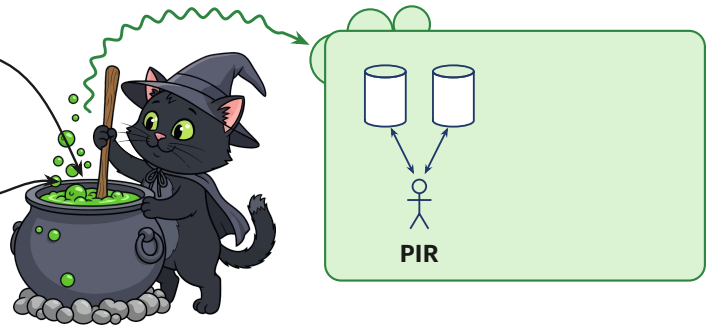
Decoding polynomials



Today's Recipe

Matching vectors

Decoding polynomials



Today's Recipe

Matching vectors

Decoding polynomials



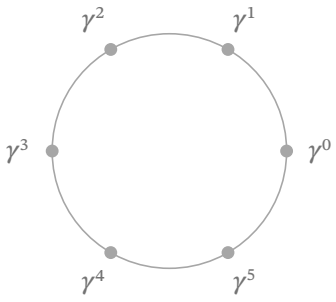
PIR

Tree evaluation

Roadmap

- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]
- New Application to Tree Evaluation [HP**R**26]
- New Construction [G**R**26]

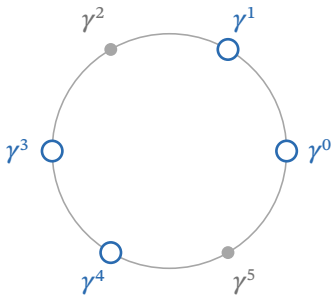
S-Decoding Polynomials [Efr09]



$m = 6$

- Let m be a modulus, and $\mathbb{F}$ a field with a primitive mth root of unity γ

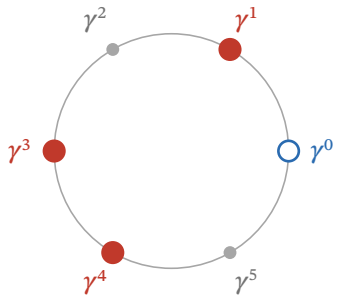
S-Decoding Polynomials [Efr09]



$m = 6$, $S = \{0, 1, 3, 4\}$

- Let m be a modulus, and $\mathbb{F}$ a field with a primitive mth root of unity γ
- $S \subseteq \mathbb{Z}_m$: any set containing 0

S-Decoding Polynomials [Efr09]



$m = 6$, $S = \{0, 1, 3, 4\}$

$P = 0$

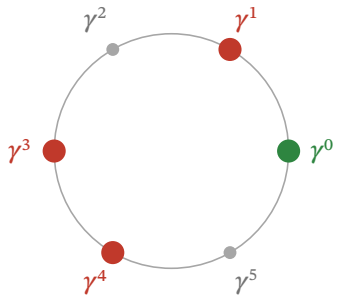
- Let m be a modulus, and $\mathbb{F}$ a field with a primitive mth root of unity γ
- $S \subseteq \mathbb{Z}_m$: any set containing 0

Definition (S-decoding polynomial)

A polynomial $P \in \mathbb{F}[X]$ is *S-decoding* if

- $P(\gamma^j) = 0$ for every $j \in S \setminus \{0\}$, and

S-Decoding Polynomials [Efr09]



- Let m be a modulus, and $\mathbb{F}$ a field with a primitive mth root of unity γ
- $S \subseteq \mathbb{Z}_m$: any set containing 0

Definition (S-decoding polynomial)

A polynomial $P \in \mathbb{F}[X]$ is *S-decoding* if

- $P(\gamma^j) = 0$ for every $j \in S \setminus \{0\}$, and
- $P(1) = 1$.

Main Cost Metric: Sparsity

- $\gamma^m = 1$, so only exponents mod m matter: think of $P \in \mathbb{F}[X]/(X^m - 1)$

Main Cost Metric: Sparsity

- $\gamma^m = 1$, so only exponents mod m matter: think of $P \in \mathbb{F}[X]/(X^m - 1)$
- Sparsity: the number of nonzero coefficients

$$P(X) = \underbrace{c_1 X^{b_1} + c_2 X^{b_2} + \cdots + c_s X^{b_s}}_{\text{sparsity } s}, \qquad b_1, \ldots, b_s \in \mathbb{Z}_m$$

Main Cost Metric: Sparsity

- $\gamma^m = 1$, so only exponents mod m matter: think of $P \in \mathbb{F}[X]/(X^m - 1)$
- Sparsity: the number of nonzero coefficients

$$P(X) = \underbrace{c_1 X^{b_1} + c_2 X^{b_2} + \cdots + c_s X^{b_s}}_{\text{sparsity } s}, \qquad b_1, \ldots, b_s \in \mathbb{Z}_m$$

- Lagrange interpolation $\Rightarrow$ always possible with degree $\leq |S| - 1 \Rightarrow$ sparsity $\leq |S|$

The Canonical Choice of S

- $m = p_1 p_2 \cdots p_k$ for distinct primes $p_1, \ldots, p_k$

The Canonical Choice of S

- $m = p_1 p_2 \cdots p_k$ for distinct primes $p_1, \ldots, p_k$
- S_m^*: residues that are 0 or 1 modulo every p_i

The Canonical Choice of S

- $m = p_1 p_2 \cdots p_k$ for distinct primes $p_1, \ldots, p_k$
- S_m^*: residues that are 0 or 1 modulo every p_i
- $|S_m^*| = 2^k$, by the Chinese remainder theorem

The Canonical Choice of S

- $m = p_1 p_2 \cdots p_k$ for distinct primes $p_1, \ldots, p_k$
- S_m^*: residues that are 0 or 1 modulo every p_i
- $|S_m^*| = 2^k$, by the Chinese remainder theorem
- Example: $S_6^* = \{0, 1, 3, 4\}$, as before

The Canonical Choice of S

- $m = p_1 p_2 \cdots p_k$ for distinct primes $p_1, \ldots, p_k$
- S_m^*: residues that are 0 or 1 modulo every p_i
- $|S_m^*| = 2^k$, by the Chinese remainder theorem
- Example: $S_6^* = \{0, 1, 3, 4\}$, as before
- Sparsity 2^k is always doable: can we do better?

x	x mod 2	x mod 3
0	0	0
1	1	1
3	1	0
4	0	1

S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$
3	3	4	3	4	0
1	4	4	5	5	0
3	3	5	4	2	1

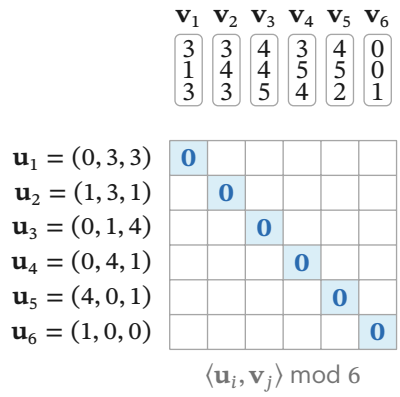
$\mathbf{u}_1 = (0, 3, 3)$
$\mathbf{u}_2 = (1, 3, 1)$
$\mathbf{u}_3 = (0, 1, 4)$
$\mathbf{u}_4 = (0, 4, 1)$
$\mathbf{u}_5 = (4, 0, 1)$
$\mathbf{u}_6 = (1, 0, 0)$

S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \dots, \mathbf{u}_N, \mathbf{v}_1, \dots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

- $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$ for every i, and



S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \dots, \mathbf{u}_N, \mathbf{v}_1, \dots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

- $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$ for every i, and
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S \setminus \{0\}$ for every $i \neq j$.

	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$
	3 1 3	3 4 3	4 4 5	3 5 4	4 5 2	0 0 1
$\mathbf{u}_1 = (0,3,3)$	**0**	3	3	3	3	3
$\mathbf{u}_2 = (1,3,1)$	3	**0**	3	4	3	1
$\mathbf{u}_3 = (0,1,4)$	1	4	**0**	3	1	4
$\mathbf{u}_4 = (0,4,1)$	1	1	3	**0**	4	1
$\mathbf{u}_5 = (4,0,1)$	3	3	3	4	**0**	1
$\mathbf{u}_6 = (1,0,0)$	3	3	4	3	4	**0**

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ mod 6, all in S_6^*

S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

- $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$ for every i, and
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S \setminus \{0\}$ for every $i \neq j$.

- The application sets N; the goal is d as small as possible

	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$
	3 1 3	3 4 3	4 4 5	3 5 4	4 5 2	0 0 1
$\mathbf{u}_1 = (0,3,3)$	**0**	3	3	3	3	3
$\mathbf{u}_2 = (1,3,1)$	3	**0**	3	4	3	1
$\mathbf{u}_3 = (0,1,4)$	1	4	**0**	3	1	4
$\mathbf{u}_4 = (0,4,1)$	1	1	3	**0**	4	1
$\mathbf{u}_5 = (4,0,1)$	3	3	3	4	**0**	1
$\mathbf{u}_6 = (1,0,0)$	3	3	4	3	4	**0**

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ mod 6, all in S_6^*

S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \dots, \mathbf{u}_N, \mathbf{v}_1, \dots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

- $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$ for every i, and
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S \setminus \{0\}$ for every $i \neq j$.

- The application sets N; the goal is d as small as possible
- S_m^*: $d = \exp(\tilde{O}((\log N)^{1/k}))$ [BBR94,Gro00]

	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$
	3 1 3	3 4 3	4 4 5	3 5 4	4 5 2	0 0 1
$\mathbf{u}_1 = (0,3,3)$	**0**	3	3	3	3	3
$\mathbf{u}_2 = (1,3,1)$	3	**0**	3	4	3	1
$\mathbf{u}_3 = (0,1,4)$	1	4	**0**	3	1	4
$\mathbf{u}_4 = (0,4,1)$	1	1	3	**0**	4	1
$\mathbf{u}_5 = (4,0,1)$	3	3	3	4	**0**	1
$\mathbf{u}_6 = (1,0,0)$	3	3	4	3	4	**0**

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ mod 6, all in S_6^*

S-Matching Vector Families [Efr09]

Definition (S-matching vector family)

Let $S \subseteq \mathbb{Z}_m$ contain 0. Vectors $\mathbf{u}_1, \dots, \mathbf{u}_N, \mathbf{v}_1, \dots, \mathbf{v}_N \in \mathbb{Z}_m^d$ form an *S-matching vector family* (S-MVF) if

- $\langle \mathbf{u}_i, \mathbf{v}_i \rangle = 0$ for every i, and
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S \setminus \{0\}$ for every $i \neq j$.

- The application sets N; the goal is d as small as possible
- S_m^*: $d = \exp(\tilde{O}((\log N)^{1/k}))$ [BBR94,Gro00]

	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$
	3 1 3	3 4 3	4 4 5	3 5 4	4 5 2	0 0 1
$\mathbf{u}_1 = (0,3,3)$	**0**	3	3	3	3	3
$\mathbf{u}_2 = (1,3,1)$	3	**0**	3	4	3	1
$\mathbf{u}_3 = (0,1,4)$	1	4	**0**	3	1	4
$\mathbf{u}_4 = (0,4,1)$	1	1	3	**0**	4	1
$\mathbf{u}_5 = (4,0,1)$	3	3	3	4	**0**	1
$\mathbf{u}_6 = (1,0,0)$	3	3	4	3	4	**0**

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ mod 6, all in S_6^*

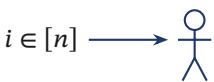
Takeaway: we focus on S_m^* because Grolmusz's construction is naturally suited to it

Application: Private Information Retrieval

Goal: privately read an entry from a remote database [CGKS95]

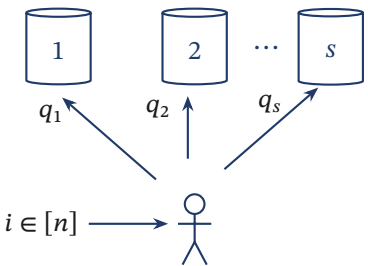
- s servers, each holding DB $\in \{0,1\}^n$





Application: Private Information Retrieval

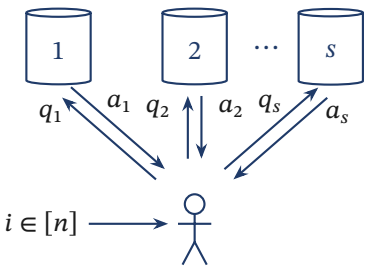
Goal: privately read an entry from a remote database [CGKS95]



- s servers, each holding DB $\in \{0,1\}^n$
- The user sends query q_j to server j

Application: Private Information Retrieval

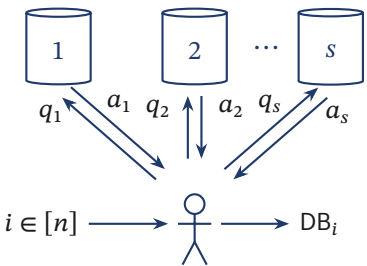
Goal: privately read an entry from a remote database [CGKS95]



- s servers, each holding DB $\in \{0,1\}^n$
- The user sends query q_j to server j
- Server j replies with a_j

Application: Private Information Retrieval

Goal: privately read an entry from a remote database [CGKS95]

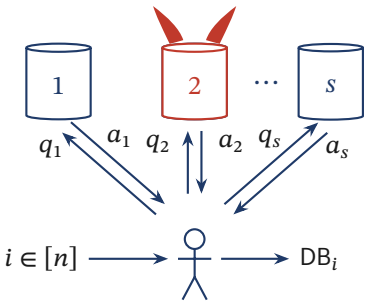


- s servers, each holding DB $\in \{0,1\}^n$
- The user sends query q_j to server j
- Server j replies with a_j

Correctness:
the user recovers DB_i from $a_1, \ldots, a_s$

Application: Private Information Retrieval

Goal: privately read an entry from a remote database [CGKS95]



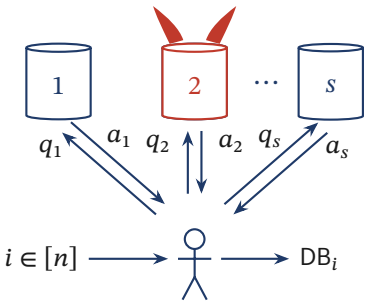
- s servers, each holding DB $\in \{0,1\}^n$
- The user sends query q_j to server j
- Server j replies with a_j

Correctness:
the user recovers DB_i from $a_1, \ldots, a_s$

Privacy:
each q_j alone reveals nothing about i

Application: Private Information Retrieval

Goal: privately read an entry from a remote database [CGKS95]



- s servers, each holding DB $\in \{0,1\}^n$
- The user sends query q_j to server j
- Server j replies with a_j

Correctness:
the user recovers DB_i from $a_1, \ldots, a_s$

Privacy:
each q_j alone reveals nothing about i

- Main cost: communication, trivially n

Matching-Vector PIR [Efr09]



Klim Efremenko

Matching vectors
S-MVF of size n in $\mathbb{Z}_m^d$



Matching-Vector PIR [Efr09]



Klim Efremenko

Matching vectors
S-MVF of size n in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



Matching-Vector PIR [Efr09]



Klim Efremenko

Matching vectors
S-MVF of size n in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



PIR

servers r

communication $\approx O(d)$

Matching-Vector PIR [Efr09]



Klim Efremenko

Matching vectors
S-MVF of size n in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



PIR
servers r
communication $\approx O(d)$

S_m^* with k primes: 2^k servers, communication $\exp(\tilde{O}((\log n)^{1/k}))$

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓
Efremenko [Efr09]	3	$\exp(\tilde{O}(\sqrt{\log n}))$	✓	✓	✓
	2^r	$\exp(\tilde{O}((\log n)^{1/r}))$			

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓
Efremenko [Efr09]	3 2^r	$\exp(\tilde{O}(\sqrt{\log n}))$ $\exp(\tilde{O}((\log n)^{1/r}))$	✓	✓	✓
Chee et al. [CFL+13] *	$3^{r/2}$	$\exp(\tilde{O}((\log n)^{1/r}))$		✓	

* Assuming number-theoretic conjectures

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓
Efremenko [Efr09]	3 2^r	$\exp(\tilde{O}(\sqrt{\log n}))$ $\exp(\tilde{O}((\log n)^{1/r}))$	✓	✓	✓
Chee et al. [CFL+13] *	$3^{r/2}$	$\exp(\tilde{O}((\log n)^{1/r}))$		✓	
Dvir–Gopi [DG16]	2^{r-1}	$\exp(\tilde{O}((\log n)^{1/r}))$			✓

* Assuming number-theoretic conjectures

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓
Efremenko [Efr09]	3 2^r	$\exp(\tilde{O}(\sqrt{\log n}))$ $\exp(\tilde{O}((\log n)^{1/r}))$	✓	✓	✓
Chee et al. [CFL+13] *	$3^{r/2}$	$\exp(\tilde{O}((\log n)^{1/r}))$		✓	
Dvir–Gopi [DG16]	2^{r-1}	$\exp(\tilde{O}((\log n)^{1/r}))$			✓
Ghasemi et al. [GKS25]	3	$\exp(\tilde{O}((\log n)^{1/3}))$			✓

* Assuming number-theoretic conjectures

Matching-Vector PIR Through the Years

Goal: PIR with $O(1)$ servers and as little communication as possible

Work	Servers	Communication	What improved		
			Matching vectors	Decoding polynomials	Use of both
Reed–Muller [CGKS95,BIKR02]	s	$n^{\tilde{O}(1/s)}$			
Yekhanin [Yek08]	3	$n^{O(1/\log\log n)}$	✓		✓
Efremenko [Efr09]	3 2^r	$\exp(\tilde{O}(\sqrt{\log n}))$ $\exp(\tilde{O}((\log n)^{1/r}))$	✓	✓	✓
Chee et al. [CFL+13] *	$3^{r/2}$	$\exp(\tilde{O}((\log n)^{1/r}))$		✓	
Dvir–Gopi [DG16]	2^{r-1}	$\exp(\tilde{O}((\log n)^{1/r}))$			✓
Ghasemi et al. [GKS25]	3	$\exp(\tilde{O}((\log n)^{1/3}))$			✓
Gupte–**R**–ChatGPT [G**R**26] *	r	$\exp(\tilde{O}((\log n)^{1/r}))$		✓	

* Assuming number-theoretic conjectures

Result 1: Improved Decoding Polynomials

Question:
when m is a product of k primes, what is the minimum sparsity of an S_m^*-decoding polynomial?

Result 1: Improved Decoding Polynomials

Question:
when m is a product of k primes, what is the minimum sparsity of an S_m^*-decoding polynomial?

- Trivial: 2^k

Result 1: Improved Decoding Polynomials

Question:
when m is a product of k primes, what is the minimum sparsity of an S_m^*-decoding polynomial?

- Trivial: 2^k
- Efremenko [Efr09], Chee et al. [CFL+13]: $\approx 3^{k/2}$ *

* Assuming number-theoretic conjectures

Result 1: Improved Decoding Polynomials

Question:
when m is a product of k primes, what is the minimum sparsity of an S_m^*-decoding polynomial?

- Trivial: 2^k
- Efremenko [Efr09], Chee et al. [CFL+13]: $\approx 3^{k/2}$ *
- Ghasemi–Kopparty [GK26]: lower bound of $k+1$

* Assuming number-theoretic conjectures

Result 1: Improved Decoding Polynomials

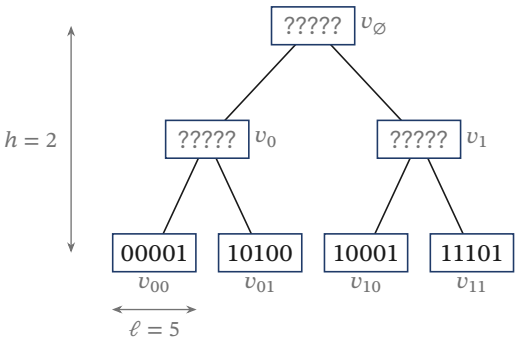
Question:
when m is a product of k primes, what is the minimum sparsity of an S_m^*-decoding polynomial?

- Trivial: 2^k
- Efremenko [Efr09], Chee et al. [CFL+13]: $\approx 3^{k/2}$ *
- Ghasemi–Kopparty [GK26]: lower bound of $k+1$
- Gupte–**R**–ChatGPT [G**R**26]: $k+1$ is attainable! *

* Assuming number-theoretic conjectures

Result 2: Application to Tree Evaluation

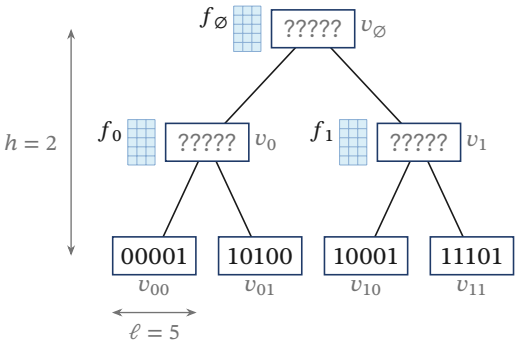
Goal: compute the root of a tree of functions in little space [CMW+12]



- Complete binary tree of height h, an ℓ-bit string at each leaf

Result 2: Application to Tree Evaluation

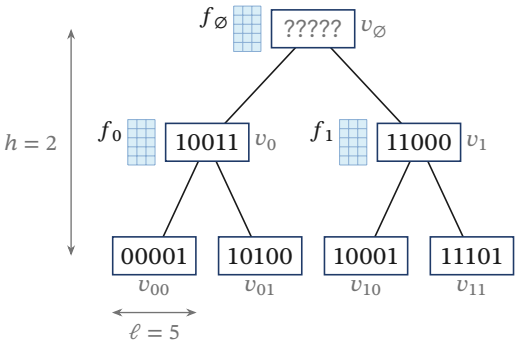
Goal: compute the root of a tree of functions in little space [CMW+12]



- Complete binary tree of height h, an ℓ-bit string at each leaf
- Internal node u: truth table of $f_u : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$

Result 2: Application to Tree Evaluation

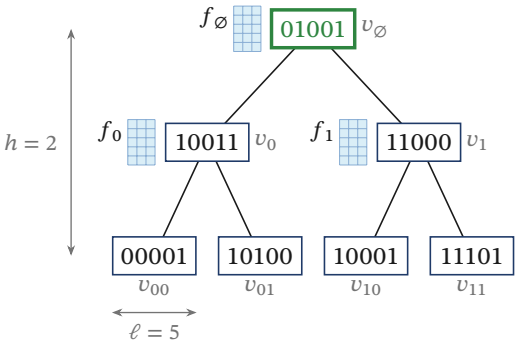
Goal: compute the root of a tree of functions in little space [CMW+12]



- Complete binary tree of height h, an ℓ-bit string at each leaf
- Internal node u: truth table of $f_u : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Propagate up: $v_u = f_u(v_{u0}, v_{u1})$

Result 2: Application to Tree Evaluation

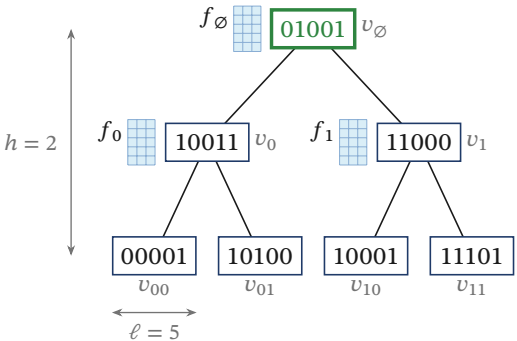
Goal: compute the root of a tree of functions in little space [CMW+12]



- Complete binary tree of height h, an ℓ-bit string at each leaf
- Internal node u: truth table of $f_u : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Propagate up: $v_u = f_u(v_{u0}, v_{u1})$
- Output: the root value $v_\varnothing$

Result 2: Application to Tree Evaluation

Goal: compute the root of a tree of functions in little space [CMW+12]



- Complete binary tree of height h, an ℓ-bit string at each leaf
- Internal node u: truth table of $f_u : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Propagate up: $v_u = f_u(v_{u0}, v_{u1})$
- Output: the root value $v_\varnothing$
- Input length $n = \text{poly}(2^{h+\ell})$; think of $h, \ell = \Theta(\log n)$

Tree Evaluation Through the Years

Defined; conjecture:
$O(\log^2 n)$ space
is optimal

2012
[CMW+12]

Tree Evaluation Through the Years

Defined; conjecture:
$O(\log^2 n)$ space
is optimal

2012
[CMW+12]

2012–19: evidence
in restricted models

Tree Evaluation Through the Years



James Cook

Ian Mertz

Defined; conjecture:
$O(\log^2 n)$ space
is optimal

Conjecture is false:
$O(\log^2 n / \log\log n)$

2012
[CMW+12]

2012–19: evidence
in restricted models

2020
[CM20]

Tree Evaluation Through the Years



James Cook

Ian Mertz

Defined; conjecture: $O(\log^2 n)$ space is optimal

2012 [CMW+12]

2012–19: evidence in restricted models

Conjecture is false: $O(\log^2 n / \log\log n)$

2020 [CM20]

Conjecture is REALLY false: $O(\log n \log\log n)$

2024 [CM24]

Tree Evaluation Through the Years



Ryan Williams

Defined; conjecture: $O(\log^2 n)$ space is optimal

2012 [CMW+12]

2012–19: evidence in restricted models

Conjecture is false: $O(\log^2 n / \log\log n)$

2020 [CM20]

Conjecture is REALLY false: $O(\log n \log\log n)$

2024 [CM24]

Time T in space $\tilde{O}(\sqrt{T})$ via tree evaluation

2025 [Wil25]

Tree Evaluation Through the Years



Ryan Williams

Defined; conjecture: $O(\log^2 n)$ space is optimal

2012
[CMW+12]

2012–19: evidence in restricted models

Conjecture is false: $O(\log^2 n / \log\log n)$

2020
[CM20]

Conjecture is REALLY false: $O(\log n \log\log n)$

2024
[CM24]

Time T in space $\tilde{O}(\sqrt{T})$ via tree evaluation

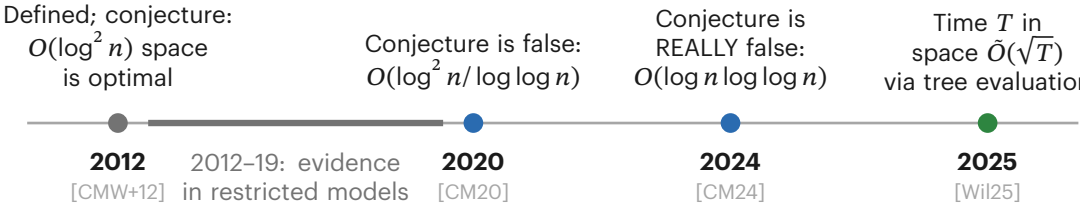
2025
[Wil25]

Question:
What powers the algorithms of [CM20,CM24]?

Tree Evaluation Through the Years



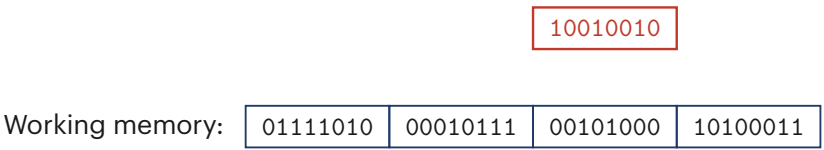
Ryan Williams



Question:
What powers the algorithms of [CM20,CM24]? **Catalytic computing!**

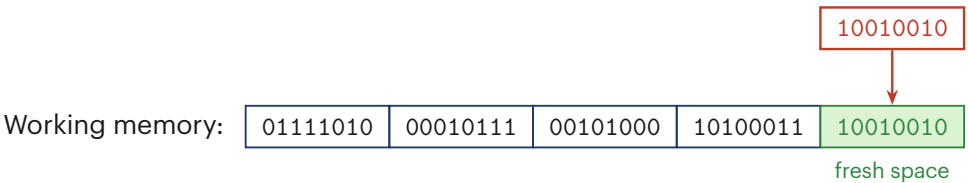
Catalytic Computing: The Idea [BCK+14]

- Setting: the algorithm has computed something new and must store it



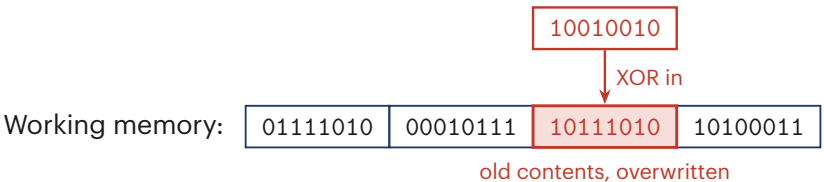
Catalytic Computing: The Idea [BCK+14]

- Setting: the algorithm has computed something new and must store it
- Most algorithms ("good" programming): allocate fresh space for it



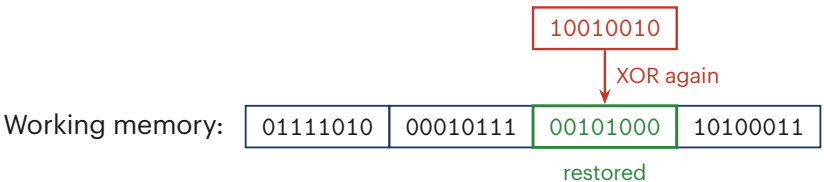
Catalytic Computing: The Idea [BCK+14]

- Setting: the algorithm has computed something new and must store it
- Most algorithms ("good" programming): allocate fresh space for it
- Catalytic computing ("bad" programming): write over occupied space!

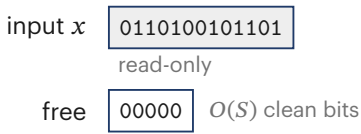


Catalytic Computing: The Idea [BCK+14]

- Setting: the algorithm has computed something new and must store it
- Most algorithms ("good" programming): allocate fresh space for it
- Catalytic computing ("bad" programming): write over occupied space!
  - ... reversibly, so the old contents can be restored at the end



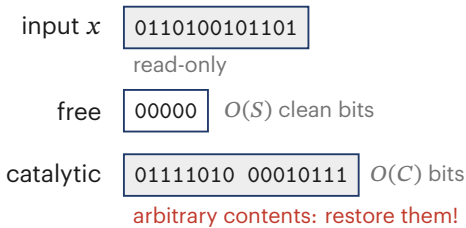
Catalytic Computing: Formal Definition



Definition (CatSpaceTime(C, S, T)**)**

An algorithm with input $x \in \{0,1\}^n$, time $O(T)$ and $O(S)$ bits of free space

Catalytic Computing: Formal Definition



Definition (CatSpaceTime(C, S, T)**)**

An algorithm with input $x \in \{0,1\}^n$, time $O(T)$ and $O(S)$ bits of free space, plus a catalytic tape holding an arbitrary $\tau \in \{0,1\}^{O(C)}$, which must hold τ again at the end.

Our Catalytic Tree Evaluation Algorithm

[HPR26]

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$



h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Our Catalytic Tree Evaluation Algorithm

[HPR26]

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Our Catalytic Tree Evaluation Algorithm

[HPR26]

Matching vectors
S-MVF of size 2^ℓ in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



Catalytic tree evaluation

catalytic	$O(d \log m)$
free	$O(h \log r + \ell)$
time	$\text{poly}(r^h \cdot 2^\ell)$

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Main Instantiation: k Primes

[HPR26]

Matching vectors
S_m^*, k primes
$d = 2^{\tilde{O}(\ell^{1/k})}$

Decoding polynomial
sparsity $r = 2^k$



h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Main Instantiation: k Primes

[HPR26]

Matching vectors
S_m^*, k primes
$d = 2^{\tilde{O}(\ell^{1/k})}$

Decoding polynomial
sparsity $r = 2^k$



Catalytic tree evaluation

catalytic	$2^{\tilde{O}(\log^{1/k} n)}$
free	$O(\log n)$
time	$\mathrm{poly}(n)$

h: tree height · ℓ: bits per value · $n = \mathrm{poly}(2^{h+\ell})$: input length

Main Instantiation: k Primes

[HPR26]

Matching vectors
S_m^*, k primes
$d = 2^{\tilde{O}(\ell^{1/k})}$

Decoding polynomial
sparsity $r = 2^k$



Catalytic tree evaluation

catalytic	$2^{\tilde{O}(\log^{1/k} n)}$
Cook–Mertz:	$O(\log n)$
free	$O(\log n)$
Cook–Mertz:	$O(\log n \log\log n)$
time	$\text{poly}(n)$
Cook–Mertz:	$n^{O(\log\log n)}$

New: free space $O(\log n)$, polynomial time, catalytic space $2^{\log^{\epsilon} n}$ for any $\epsilon > 0$

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Alternate Instantiation Capturing [CM24]

[HPR26]

Matching vectors
one prime $m = O(\ell)$, $d = O(\ell)$

Decoding polynomial
sparsity $r = O(\ell)$



h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Alternate Instantiation Capturing [CM24]

[HPR26]

Matching vectors
one prime $m = O(\ell)$, $d = O(\ell)$

Decoding polynomial
sparsity $r = O(\ell)$



Catalytic tree evaluation

catalytic	$O(\log n \log\log n)$
free	$O(\log n \log\log n)$
time	$n^{O(\log\log n)}$

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Alternate Instantiation Capturing [CM24]

[HPR26]

Matching vectors
one prime $m = O(\ell)$, $d = O(\ell)$

Decoding polynomial
sparsity $r = O(\ell)$



Catalytic tree evaluation

catalytic	$O(\log n \log\log n)$
free	$O(\log n \log\log n)$
time	$n^{O(\log\log n)}$

The same recipe also gives the Cook–Mertz bounds [CM24]

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

One Theorem, Two Algorithms

	Theorem (informal)	One prime m matches Cook–Mertz [CM24]	k primes new, with S^*_m
Matching vectors dimension d	d	$O(\ell)$	$2^{\tilde{O}(\ell^{1/k})}$
S-decoding polynomial sparsity r: r^2 calls per node	r	$O(\ell)$	2^k
[HPR26]			
Catalytic space	$O(d \log m)$	$O(\log n \log\log n)$	$2^{\tilde{O}(\log^{1/k} n)}$
Free space	$O(h \log r + \ell)$	$O(\log n \log\log n)$	$O(\log n)$
Time	$\text{poly}(r^h \cdot 2^\ell)$	$n^{O(\log\log n)}$	$\text{poly}(n)$

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

One Theorem, Two Algorithms

	Theorem (informal)	One prime m matches Cook–Mertz [CM24]	k primes new, with S^*_m
Matching vectors dimension d	d	$O(\ell)$	$2^{\tilde{O}(\ell^{1/k})}$
S-decoding polynomial sparsity r: r^2 calls per node	r	$O(\ell)$	2^k
[HPR26]			
Catalytic space	$O(d \log m)$	$O(\log n \log\log n)$	$2^{\tilde{O}(\log^{1/k} n)}$
Free space	$O(h \log r + \ell)$	$O(\log n \log\log n)$	$O(\log n)$
Time	$\text{poly}(r^h \cdot 2^\ell)$	$n^{O(\log\log n)}$	$\text{poly}(n)$

- Better ingredients plug straight in: matching vectors as short as the known lower bounds allow would give $O(\log n \log\log n)$ space in polynomial time

h: tree height · ℓ: bits per value · $n = \text{poly}(2^{h+\ell})$: input length

Roadmap

- What Is a Decoding Polynomial?
- ➜ Warm-Up Application to Private Information Retrieval [Efr09]
- New Application to Tree Evaluation [HP**R**26]
- New Construction [G**R**26]

Working Backwards from the Goal [Efr09]

- Database DB $\in \mathbb{F}^n$; the user wants DB_i

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathrm{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\mathrm{DB}_i = \sum_j \mathrm{DB}_j \cdot \mathbb{1}[i = j]$$

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\begin{aligned}\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] \\ &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0]\end{aligned}$$

S-MVF: $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0$ iff $i = j$

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\begin{aligned}\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] \\ &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] \\ &= \sum_j \mathsf{DB}_j \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle})\end{aligned}$$

S-MVF: $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0$ iff $i = j$

P is S-decoding

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\begin{aligned}
\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] \\
&= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t}
\end{aligned}$$

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$
\begin{aligned}
\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] && \\
&= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t} \\
&= \sum_t c_t \underbrace{\left(\sum_j \mathsf{DB}_j \gamma^{\langle b_t \mathbf{u}_i, \mathbf{v}_j \rangle} \right)}_{\text{server } t\text{'s contribution}} && \text{swap the sums}
\end{aligned}
$$

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$
\begin{aligned}
\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] && \\
&= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t} \\
&= \sum_t c_t \underbrace{\left(\sum_j \mathsf{DB}_j \gamma^{\langle b_t \mathbf{u}_i, \mathbf{v}_j \rangle} \right)}_{\text{server } t\text{'s contribution}} && \text{swap the sums}
\end{aligned}
$$

- Server t needs only $\mathbf{q}_t = b_t \mathbf{u}_i$: one server per term

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\begin{aligned}
\mathsf{DB}_i &= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[i = j] \\
&= \sum_j \mathsf{DB}_j \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t} \\
&= \sum_t c_t \underbrace{\left(\sum_j \mathsf{DB}_j \gamma^{\langle b_t \mathbf{u}_i, \mathbf{v}_j \rangle} \right)}_{\text{server } t\text{'s contribution}} && \text{swap the sums}
\end{aligned}$$

- Server t needs only $\mathbf{q}_t = b_t \mathbf{u}_i$: one server per term
- But $\mathbf{q}_t = b_t \mathbf{u}_i$ reveals i

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

$$\begin{aligned}
\gamma^{\langle \mathbf{z}, \mathbf{v}_i \rangle} \cdot \mathsf{DB}_i &= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot \mathbb{1}[i = j] \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t} \\
&= \sum_t c_t \underbrace{\left(\sum_j \mathsf{DB}_j \gamma^{\langle \mathbf{z} + b_t \mathbf{u}_i, \mathbf{v}_j \rangle} \right)}_{\text{server } t\text{'s contribution}} && \text{swap the sums}
\end{aligned}$$

- Server t needs only $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i$: one server per term
- $\mathbf{z}$ uniform, so each $\mathbf{q}_t$ alone reveals nothing about i

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Working Backwards from the Goal [Efr09]

- Database $\mathsf{DB} \in \mathbb{F}^n$; the user wants DB_i

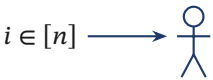
$$
\begin{aligned}
\gamma^{\langle \mathbf{z}, \mathbf{v}_i \rangle} \cdot \mathsf{DB}_i &= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot \mathbb{1}[i = j] && \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot \mathbb{1}[\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0] && S\text{-MVF: } \langle \mathbf{u}_i, \mathbf{v}_j \rangle = 0 \text{ iff } i = j \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \cdot P(\gamma^{\langle \mathbf{u}_i, \mathbf{v}_j \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_j \mathsf{DB}_j \, \gamma^{\langle \mathbf{z}, \mathbf{v}_j \rangle} \sum_t c_t \gamma^{b_t \langle \mathbf{u}_i, \mathbf{v}_j \rangle} && \text{expand } P(X) = \textstyle\sum_{t=1}^{r} c_t X^{b_t} \\
&= \sum_t c_t \underbrace{\left(\sum_j \mathsf{DB}_j \gamma^{\langle \mathbf{z} + b_t \mathbf{u}_i, \mathbf{v}_j \rangle} \right)}_{\text{server } t\text{'s contribution}} && \text{swap the sums}
\end{aligned}
$$

- Server t needs only $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i$: one server per term
- $\mathbf{z}$ uniform, so each $\mathbf{q}_t$ alone reveals nothing about i
- The user knows $\mathbf{z}$, so it divides out $\gamma^{\langle \mathbf{z}, \mathbf{v}_i \rangle}$

γ: primitive mth root of unity · $P(\gamma^s) = 0$ on $S \setminus \{0\}$, $P(1) = 1$

Making This a Protocol



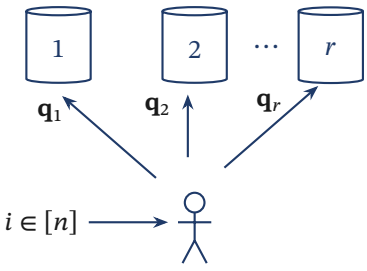


- One server per term of P: r servers, each holding DB

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S$, 0 iff $i = j$ · $\mathbf{z}$: uniform mask

$$P(X) = \sum_{t=1}^{r} c_t X^{b_t}$$

Making This a Protocol

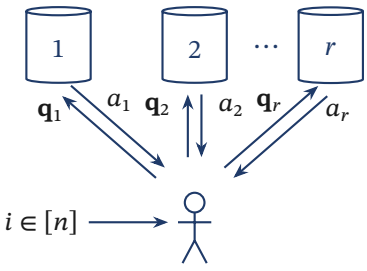


- One server per term of P: r servers, each holding DB
- Server t gets $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i \in \mathbb{Z}_m^d$

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S$, 0 iff $i = j$ · $\mathbf{z}$: uniform mask

$P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

Making This a Protocol

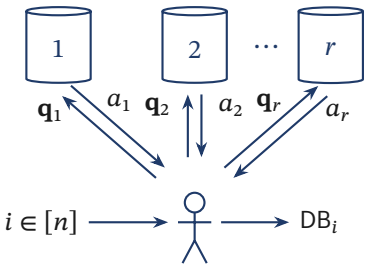


- One server per term of P: r servers, each holding DB
- Server t gets $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i \in \mathbb{Z}_m^d$
- It replies $a_t = \sum_j \mathrm{DB}_j \gamma^{\langle \mathbf{q}_t, \mathbf{v}_j \rangle}$

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S$, 0 iff $i = j$ · $\mathbf{z}$: uniform mask

$P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

Making This a Protocol

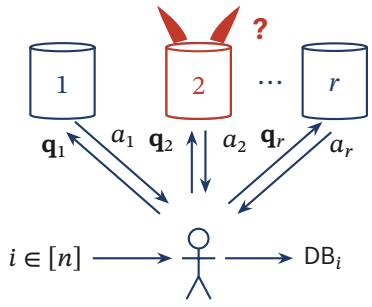


- One server per term of P: r servers, each holding DB
- Server t gets $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i \in \mathbb{Z}_m^d$
- It replies $a_t = \sum_j \mathrm{DB}_j \gamma^{\langle \mathbf{q}_t, \mathbf{v}_j \rangle}$
- The user outputs $\gamma^{-\langle \mathbf{z}, \mathbf{v}_i \rangle} \sum_t c_t a_t = \mathrm{DB}_i$

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S$, 0 iff $i = j$ · $\mathbf{z}$: uniform mask

$P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

Making This a Protocol



- One server per term of P: r servers, each holding DB
- Server t gets $\mathbf{q}_t = \mathbf{z} + b_t \mathbf{u}_i \in \mathbb{Z}_m^d$
- It replies $a_t = \sum_j \mathrm{DB}_j \gamma^{\langle \mathbf{q}_t, \mathbf{v}_j \rangle}$
- The user outputs $\gamma^{-\langle \mathbf{z}, \mathbf{v}_i \rangle} \sum_t c_t a_t = \mathrm{DB}_i$

Privacy:
each $\mathbf{q}_t$ alone is uniform, so even a malicious server learns nothing

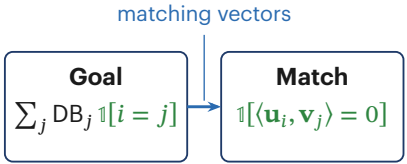
$\langle \mathbf{u}_i, \mathbf{v}_j \rangle \in S$, 0 iff $i = j$ · $\mathbf{z}$: uniform mask
$P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

Efremenko's PIR Framework: Summary

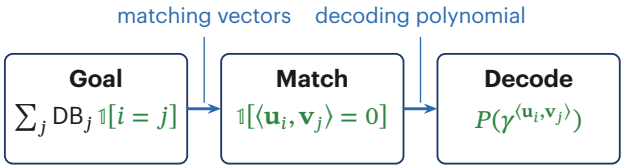
Goal

$\sum_j \mathrm{DB}_j \mathbb{1}[i = j]$

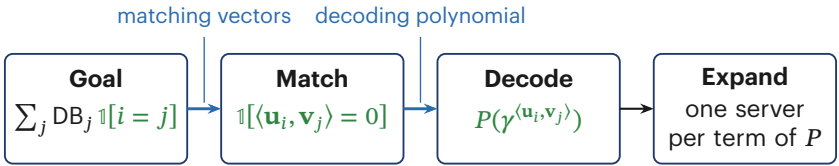
Efremenko's PIR Framework: Summary



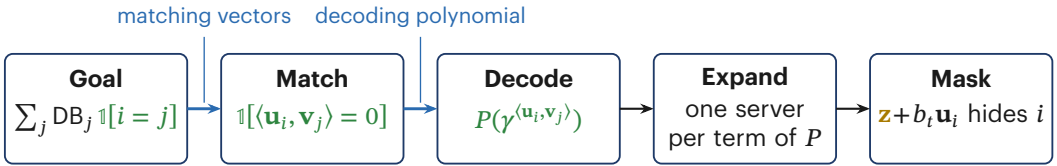
Efremenko's PIR Framework: Summary



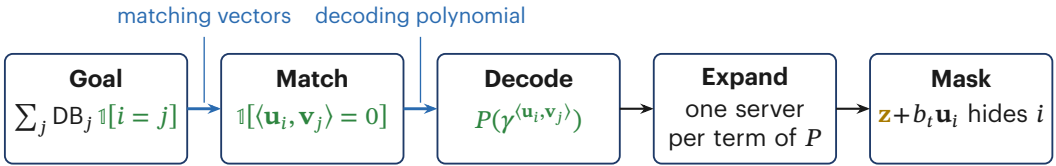
Efremenko's PIR Framework: Summary



Efremenko's PIR Framework: Summary

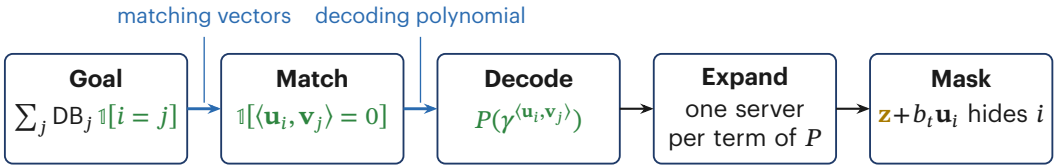


Efremenko's PIR Framework: Summary



- Servers: the sparsity r of P; communication: the dimension d
- That is Efremenko's theorem [Efr09]

Efremenko's PIR Framework: Summary



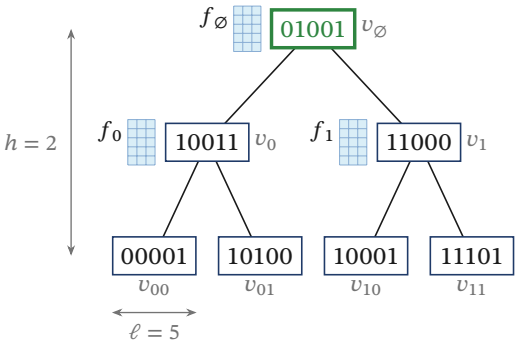
- Servers: the sparsity r of P; communication: the dimension d
- That is Efremenko's theorem [Efr09]
- Tree evaluation: the same procedure, the mask $\mathbf{z}$ is now whatever is on the catalytic tape

Roadmap

- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]
- ➜ New Application to Tree Evaluation [HP**R**26]
- New Construction [G**R**26]

Recall: Tree Evaluation

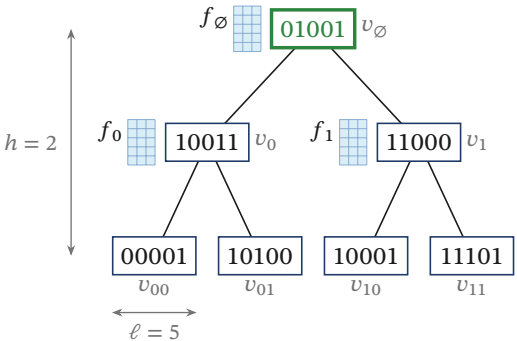
Goal: compute the root of a tree of functions in little space [CMW+12]



- Propagate up: $v_u = f_u(v_{u0}, v_{u1})$

Recall: Tree Evaluation

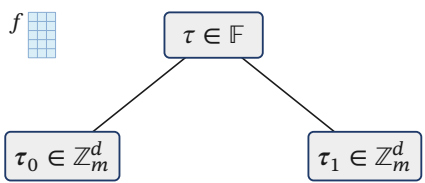
Goal: compute the root of a tree of functions in little space [CMW+12]



- Propagate up: $v_u = f_u(v_{u0}, v_{u1})$
- $h, \ell = O(\log n)$

Approach: The One-Level Gadget [CM24]

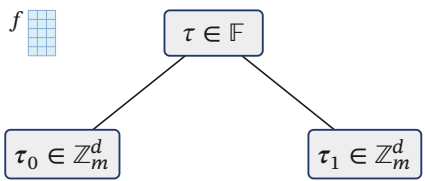
Goal: turn oracles for the children into an oracle for the parent



- Registers: $\tau_0, \tau_1 \in \mathbb{Z}_m^d$ for the children, $\tau \in \mathbb{F}$ for the parent

Approach: The One-Level Gadget [CM24]

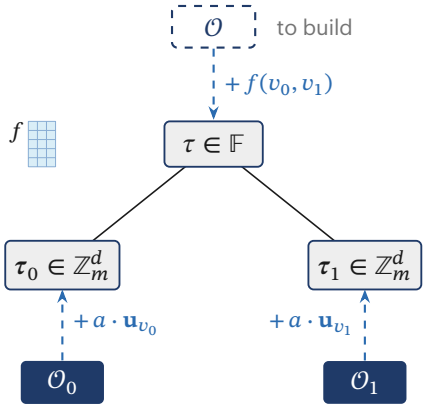
Goal: turn oracles for the children into an oracle for the parent



- Registers: $\boldsymbol{\tau}_0, \boldsymbol{\tau}_1 \in \mathbb{Z}_m^d$ for the children, $\tau \in \mathbb{F}$ for the parent
- Each $v \in \{0,1\}^\ell$ has an encoding $\mathbf{u}_v \in \mathbb{Z}_m^d$: think of a useful one

Approach: The One-Level Gadget [CM24]

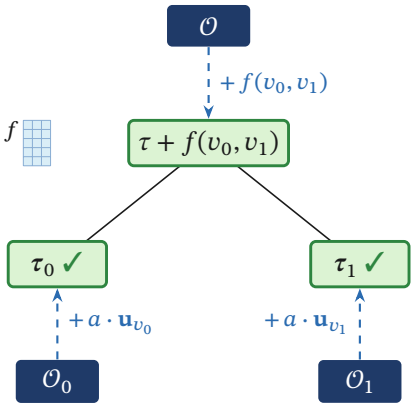
Goal: turn oracles for the children into an oracle for the parent



- Registers: $\tau_0, \tau_1 \in \mathbb{Z}_m^d$ for the children, $\tau \in \mathbb{F}$ for the parent
- Each $v \in \{0,1\}^\ell$ has an encoding $\mathbf{u}_v \in \mathbb{Z}_m^d$: think of a useful one
- Oracle for child b: $\mathcal{O}_b(a)$ sets $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ $(a \in \mathbb{Z}_m)$

Approach: The One-Level Gadget [CM24]

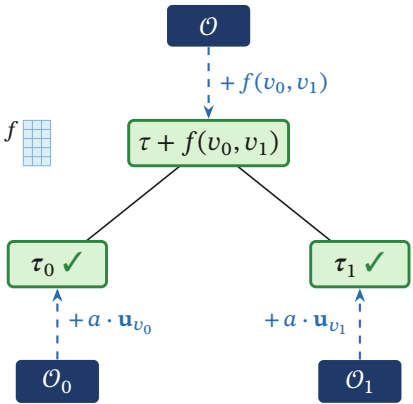
Goal: turn oracles for the children into an oracle for the parent



- Registers: $\tau_0, \tau_1 \in \mathbb{Z}_m^d$ for the children, $\tau \in \mathbb{F}$ for the parent
- Each $v \in \{0,1\}^\ell$ has an encoding $\mathbf{u}_v \in \mathbb{Z}_m^d$: think of a useful one
- Oracle for child b: $\mathcal{O}_b(a)$ sets $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ $(a \in \mathbb{Z}_m)$
- Goal: $\tau \leftarrow \tau + f(v_0, v_1)$, then restore τ_0, τ_1

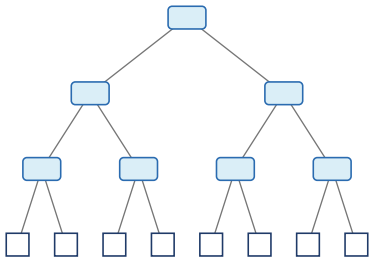
Approach: The One-Level Gadget [CM24]

Goal: turn oracles for the children into an oracle for the parent



- Registers: $\tau_0, \tau_1 \in \mathbb{Z}_m^d$ for the children, $\tau \in \mathbb{F}$ for the parent
- Each $v \in \{0,1\}^\ell$ has an encoding $\mathbf{u}_v \in \mathbb{Z}_m^d$: think of a useful one
- Oracle for child b: $\mathcal{O}_b(a)$ sets $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ $(a \in \mathbb{Z}_m)$
- Goal: $\tau \leftarrow \tau + f(v_0, v_1)$, then restore τ_0, τ_1
- Cost: q oracle calls, s free bits

From One Level to Many [CM24]

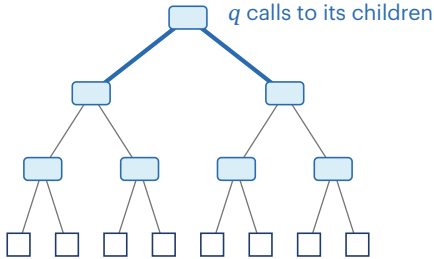


Informal theorem statement

A one-level gadget with q oracle calls, s bits of free space, time poly(2^{ℓ}) and registers in $\mathbb{Z}_m^d$ ($d \log m$ bits) gives tree evaluation in

h: tree height · ℓ: bits per value
$n = \text{poly}(2^{h+\ell})$: input length

From One Level to Many [CM24]



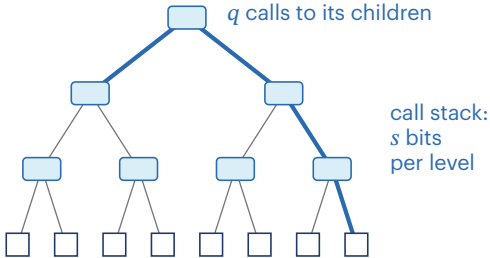
Informal theorem statement

A one-level gadget with q oracle calls, s bits of free space, time poly(2^{ℓ}) and registers in $\mathbb{Z}_m^d$ ($d \log m$ bits) gives tree evaluation in

- time poly$(q^h \cdot 2^{\ell})$,

h: tree height · ℓ: bits per value
$n = \text{poly}(2^{h+\ell})$: input length

From One Level to Many [CM24]



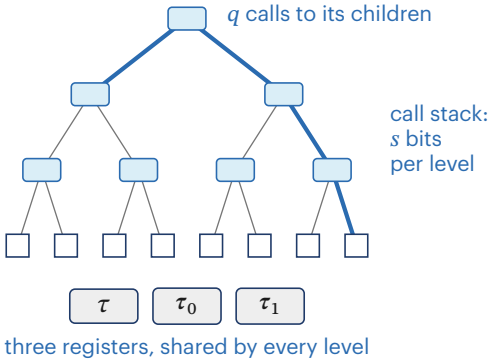
Informal theorem statement

A one-level gadget with q oracle calls, s bits of free space, time poly(2^ℓ) and registers in $\mathbb{Z}_m^d$ ($d \log m$ bits) gives tree evaluation in

- time poly$(q^h \cdot 2^\ell)$,
- free space $hs + O(\ell + \log(d \log m))$, and

h: tree height · ℓ: bits per value
$n = \text{poly}(2^{h+\ell})$: input length

From One Level to Many [CM24]



Informal theorem statement

A one-level gadget with q oracle calls, s bits of free space, time poly(2^ℓ) and registers in $\mathbb{Z}_m^d$ ($d \log m$ bits) gives tree evaluation in

- time poly$(q^h \cdot 2^\ell)$,
- free space $hs + O(\ell + \log(d \log m))$, and
- catalytic space $O(d \log m)$.

h: tree height · ℓ: bits per value
$n = \text{poly}(2^{h+\ell})$: input length

A Gadget Inspired by Efremenko's PIR

$$f(v_0, v_1) = \sum_{x,y} f(x, y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1]$$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$f(v_0, v_1) = \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1]$$
$$= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] \qquad S\text{-MVF}$$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$
\begin{aligned}
f(v_0, v_1) &= \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1] \\
&= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] && S\text{-MVF} \\
&= \sum_{x,y} f(x,y) \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) && P \text{ is } S\text{-decoding}
\end{aligned}
$$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$
\begin{aligned}
f(v_0, v_1) &= \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1] \\
&= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] && S\text{-MVF} \\
&= \sum_{x,y} f(x,y) \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_{x,y} f(x,y) \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} && \text{expand both}
\end{aligned}
$$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$
\begin{aligned}
f(v_0, v_1) &= \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1] \\
&= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] && S\text{-MVF} \\
&= \sum_{x,y} f(x,y) \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_{x,y} f(x,y) \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} && \text{expand both} \\
&= \sum_{t,t'} c_t c_{t'} \underbrace{\left(\sum_{x,y} f(x,y) \gamma^{\langle b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \right)}_{\text{contribution from recursive call } (t,t')} && \text{swap the sums}
\end{aligned}
$$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$
\begin{aligned}
f(v_0, v_1) &= \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1] \\
&= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] && S\text{-MVF} \\
&= \sum_{x,y} f(x,y) \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_{x,y} f(x,y) \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} && \text{expand both} \\
&= \sum_{t,t'} c_t c_{t'} \underbrace{\left(\sum_{x,y} f(x,y) \gamma^{\langle b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \right)}_{\text{contribution from recursive call } (t,t')} && \text{swap the sums}
\end{aligned}
$$

- For each of the r^2 pairs (t, t'): $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}, b_{t'} \mathbf{u}_{v_1}$; then sum over the truth table

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$
\begin{aligned}
f(v_0, v_1) &= \sum_{x,y} f(x,y) \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1] \\
&= \sum_{x,y} f(x,y) \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] && S\text{-MVF} \\
&= \sum_{x,y} f(x,y) \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) && P \text{ is } S\text{-decoding} \\
&= \sum_{x,y} f(x,y) \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} && \text{expand both} \\
&= \sum_{t,t'} c_t c_{t'} \underbrace{\left(\sum_{x,y} f(x,y) \gamma^{\langle b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \right)}_{\text{contribution from recursive call } (t,t')} && \text{swap the sums}
\end{aligned}
$$

- For each of the r^2 pairs (t, t'): $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}, b_{t'} \mathbf{u}_{v_1}$; then sum over the truth table
- But the registers start at τ_0, τ_1, not $\mathbf{0}$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$\gamma^{\langle \tau_0, \mathbf{v}_{v_0}\rangle + \langle \tau_1, \mathbf{v}_{v_1}\rangle} \cdot f(v_0, v_1) = \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x\rangle + \langle \tau_1, \mathbf{v}_y\rangle} \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1]$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x\rangle + \langle \tau_1, \mathbf{v}_y\rangle} \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x\rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y\rangle = 0] \quad S\text{-MVF}$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x\rangle + \langle \tau_1, \mathbf{v}_y\rangle} \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x\rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y\rangle}) \quad P \text{ is } S\text{-decoding}$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x\rangle + \langle \tau_1, \mathbf{v}_y\rangle} \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x\rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y\rangle} \quad \text{expand both}$$

$$= \sum_{t,t'} c_t c_{t'} \underbrace{\left(\sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x\rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y\rangle} \right)}_{\text{contribution from recursive call } (t,t')} \quad \text{swap the sums}$$

- For each of the r^2 pairs (t,t'): $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}, b_{t'} \mathbf{u}_{v_1}$; then sum over the truth table
- The registers' contents act as masks

$\langle \mathbf{u}_v, \mathbf{v}_x\rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

A Gadget Inspired by Efremenko's PIR

$$\gamma^{\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle} \cdot f(v_0, v_1) = \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x \rangle + \langle \tau_1, \mathbf{v}_y \rangle} \cdot \mathbb{1}[x = v_0] \cdot \mathbb{1}[y = v_1]$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x \rangle + \langle \tau_1, \mathbf{v}_y \rangle} \cdot \mathbb{1}[\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle = \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle = 0] \quad S\text{-MVF}$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x \rangle + \langle \tau_1, \mathbf{v}_y \rangle} \cdot P(\gamma^{\langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle}) \cdot P(\gamma^{\langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle}) \quad P \text{ is } S\text{-decoding}$$

$$= \sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0, \mathbf{v}_x \rangle + \langle \tau_1, \mathbf{v}_y \rangle} \sum_{t,t'} c_t c_{t'} \gamma^{b_t \langle \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + b_{t'} \langle \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \quad \text{expand both}$$

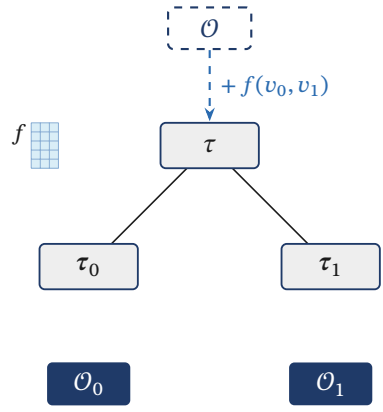
$$= \sum_{t,t'} c_t c_{t'} \underbrace{\left(\sum_{x,y} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \right)}_{\text{contribution from recursive call } (t,t')} \quad \text{swap the sums}$$

- For each of the r^2 pairs (t,t'): $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}, b_{t'} \mathbf{u}_{v_1}$; then sum over the truth table
- The registers' contents act as masks
- Divide out $\gamma^{\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle}$: a few more calls find its exponent, if the oracles can also add $\mathbf{v}_{v_b}$

$\langle \mathbf{u}_v, \mathbf{v}_x \rangle \in S$, 0 iff $v = x \cdot P(X) = \sum_{t=1}^{r} c_t X^{b_t}$
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)

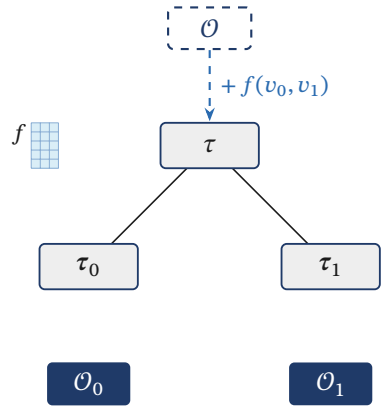


$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):

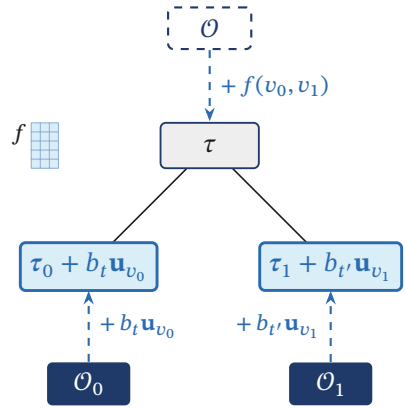


$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):
  - $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}$, $b_{t'} \mathbf{u}_{v_1}$



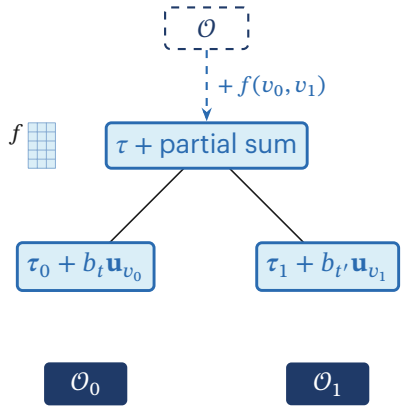
$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):
  - $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}$, $b_{t'} \mathbf{u}_{v_1}$
  - For each (x, y), read the registers and add to τ:

$$c_t c_{t'} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \cdot \gamma^{-\langle \tau_0, \mathbf{v}_{v_0} \rangle - \langle \tau_1, \mathbf{v}_{v_1} \rangle}$$



$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

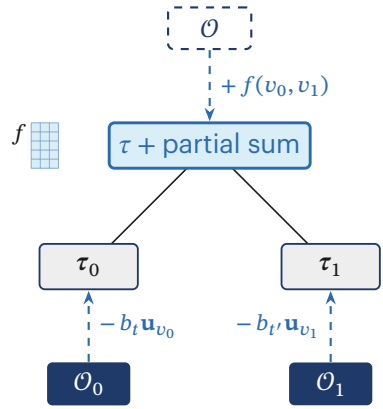
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):
  - $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}$, $b_{t'} \mathbf{u}_{v_1}$
  - For each (x, y), read the registers and add to τ:

$$c_t c_{t'} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \cdot \gamma^{-\langle \tau_0, \mathbf{v}_{v_0} \rangle - \langle \tau_1, \mathbf{v}_{v_1} \rangle}$$

  - $\mathcal{O}_0(-b_t)$, $\mathcal{O}_1(-b_{t'})$ undo the additions



$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

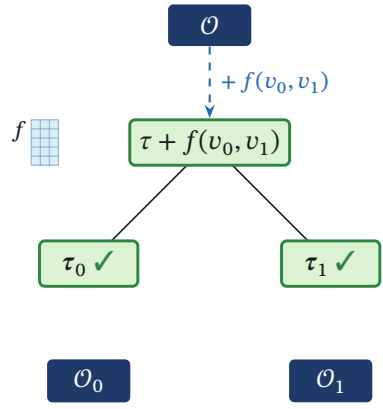
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):
  - $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}$, $b_{t'} \mathbf{u}_{v_1}$
  - For each (x, y), read the registers and add to τ:

$$c_t c_{t'} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \cdot \gamma^{-\langle \tau_0, \mathbf{v}_{v_0} \rangle - \langle \tau_1, \mathbf{v}_{v_1} \rangle}$$

  - $\mathcal{O}_0(-b_t)$, $\mathcal{O}_1(-b_{t'})$ undo the additions
- By the identity, τ gains exactly $f(v_0, v_1)$



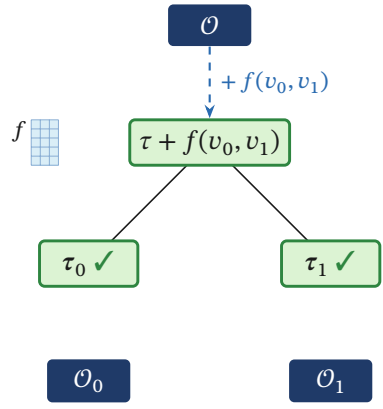
$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children
$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

The Gadget, Step by Step

- Setup: find $\langle \tau_0, \mathbf{v}_{v_0} \rangle + \langle \tau_1, \mathbf{v}_{v_1} \rangle$ (a few calls)
- For each of the r^2 pairs (t, t'):
  - $\mathcal{O}_0(b_t)$, $\mathcal{O}_1(b_{t'})$ add $b_t \mathbf{u}_{v_0}$, $b_{t'} \mathbf{u}_{v_1}$
  - For each (x, y), read the registers and add to τ:

$$c_t c_{t'} f(x,y)\, \gamma^{\langle \tau_0 + b_t \mathbf{u}_{v_0}, \mathbf{v}_x \rangle + \langle \tau_1 + b_{t'} \mathbf{u}_{v_1}, \mathbf{v}_y \rangle} \cdot \gamma^{-\langle \tau_0, \mathbf{v}_{v_0} \rangle - \langle \tau_1, \mathbf{v}_{v_1} \rangle}$$

  - $\mathcal{O}_0(-b_t)$, $\mathcal{O}_1(-b_{t'})$ undo the additions
- By the identity, τ gains exactly $f(v_0, v_1)$
- Cost: $q = O(r^2)$, $s = O(\log r)$



$\tau \in \mathbb{F}$: parent · $\tau_0, \tau_1 \in \mathbb{Z}_m^d$: children

$\mathcal{O}_b(a)$: $\tau_b \leftarrow \tau_b + a \cdot \mathbf{u}_{v_b}$ · $P(X) = \sum_{t=1}^{r} c_t X^{b_t}$

Putting It Together

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Putting It Together

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



One-level gadget
$q = O(r^2)$ calls
$s = O(\log r)$ bits

h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Putting It Together

Matching vectors
S-MVF of size 2^ℓ in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



One-level gadget
$q = O(r^2)$ calls
$s = O(\log r)$ bits

Cook–Mertz compiler [CM24]

Tree evaluation

catalytic space	$O(d \log m)$
free space	$O(h \log r + \ell)$
time	$\text{poly}(r^h \cdot 2^\ell)$

h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Main Instantiation: k Primes

- Matching vectors, as a black box [BBR94,Gro00]: for $m = p_1 \cdots p_k$, an S_m^*-MVF of size 2^ℓ in $\mathbb{Z}_m^d$, with $d = 2^{\tilde{O}(\ell^{1/k})}$ and entries computable in small space

h: tree height · ℓ: bits per value

Main Instantiation: k Primes

- Matching vectors, as a black box [BBR94,Gro00]: for $m = p_1 \cdots p_k$, an S_m^*-MVF of size 2^ℓ in $\mathbb{Z}_m^d$, with $d = 2^{\tilde{O}(\ell^{1/k})}$ and entries computable in small space
- Decoding polynomial for S_m^*: the generic one, with $r = 2^k$ terms

h: tree height · ℓ: bits per value

Main Instantiation: k Primes

- Matching vectors, as a black box [BBR94,Gro00]: for $m = p_1 \cdots p_k$, an S_m^*-MVF of size 2^ℓ in $\mathbb{Z}_m^d$, with $d = 2^{\tilde{O}(\ell^{1/k})}$ and entries computable in small space
- Decoding polynomial for S_m^*: the generic one, with $r = 2^k$ terms

	Theorem	$m, r = O(1)$, $d = 2^{\tilde{O}(\ell^{1/k})}$, $h, \ell = O(\log n)$
Catalytic space	$O(d \log m)$	$2^{\tilde{O}(\log^{1/k} n)}$
Free space	$O(h \log r + \ell)$	$O(\log n)$
Time	$\text{poly}(r^h \cdot 2^\ell)$	$\text{poly}(n)$

h: tree height · ℓ: bits per value

Alternate Instantiation Capturing [CM24]

- One prime $m = p$, and distinct Boolean vectors $\mathbf{a}_i \in \{0,1\}^M$ of weight $p-1$: $p, M = O(\ell)$ suffice

h: tree height · ℓ: bits per value

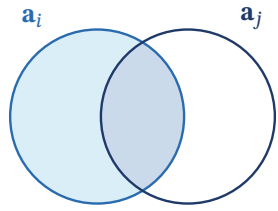
Alternate Instantiation Capturing [CM24]

- One prime $m = p$, and distinct Boolean vectors $\mathbf{a}_i \in \{0,1\}^M$ of weight $p-1$: $p, M = O(\ell)$ suffice
- $\mathbf{u}_i = \mathbf{v}_i = (1, \mathbf{a}_i) \in \mathbb{Z}_p^{M+1}$

h: tree height · ℓ: bits per value

Alternate Instantiation Capturing [CM24]

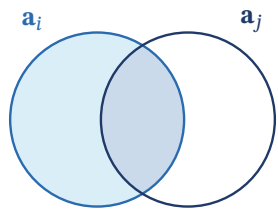
- One prime $m = p$, and distinct Boolean vectors $\mathbf{a}_i \in \{0,1\}^M$ of weight $p-1$: $p, M = O(\ell)$ suffice
- $\mathbf{u}_i = \mathbf{v}_i = (1, \mathbf{a}_i) \in \mathbb{Z}_p^{M+1}$
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 1 + \langle \mathbf{a}_i, \mathbf{a}_j \rangle$: this is $p \equiv 0$ iff $i = j$



h: tree height · ℓ: bits per value

Alternate Instantiation Capturing [CM24]

- One prime $m = p$, and distinct Boolean vectors $\mathbf{a}_i \in \{0,1\}^M$ of weight $p-1$: $p, M = O(\ell)$ suffice
- $\mathbf{u}_i = \mathbf{v}_i = (1, \mathbf{a}_i) \in \mathbb{Z}_p^{M+1}$
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 1 + \langle \mathbf{a}_i, \mathbf{a}_j \rangle$: this is $p \equiv 0$ iff $i = j$
- $P(X) = \frac{1}{p}(1 + X + X^2 + \cdots + X^{p-1})$ is $\mathbb{Z}_p$-decoding, with p terms



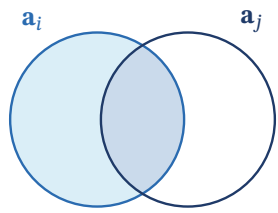
supports, each of size $p-1$

$$\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 1 + \langle \mathbf{a}_i, \mathbf{a}_j \rangle$$

h: tree height · ℓ: bits per value

Alternate Instantiation Capturing [CM24]

- One prime $m = p$, and distinct Boolean vectors $\mathbf{a}_i \in \{0,1\}^M$ of weight $p-1$: $p, M = O(\ell)$ suffice
- $\mathbf{u}_i = \mathbf{v}_i = (1, \mathbf{a}_i) \in \mathbb{Z}_p^{M+1}$
- $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 1 + \langle \mathbf{a}_i, \mathbf{a}_j \rangle$: this is $p \equiv 0$ iff $i = j$
- $P(X) = \frac{1}{p}(1 + X + X^2 + \cdots + X^{p-1})$ is $\mathbb{Z}_p$-decoding, with p terms
- Plug in $d, m, r = O(\ell)$: $O(\log n \log\log n)$ space, $n^{O(\log\log n)}$ time



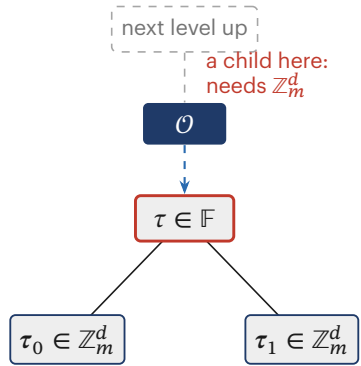
supports, each of size $p-1$

$\langle \mathbf{u}_i, \mathbf{v}_j \rangle = 1 + \langle \mathbf{a}_i, \mathbf{a}_j \rangle$

h: tree height · ℓ: bits per value

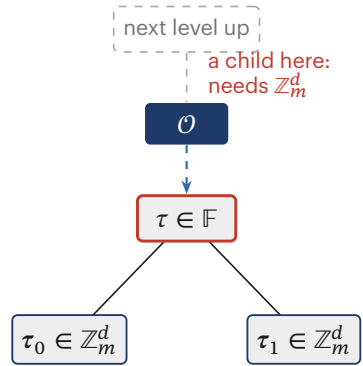
Technical Fine Print

- To recurse, the parent and child types need to match



Technical Fine Print

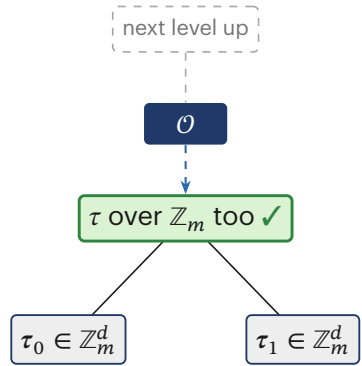
- To recurse, the parent and child types need to match
- This is a serious issue: if $m = 0$ and $\gamma^m = 1$, no S_m^*-decoding polynomial exists



Technical Fine Print

- To recurse, the parent and child types need to match
- This is a serious issue: if $m = 0$ and $\gamma^m = 1$, no S_m^*-decoding polynomial exists
- Fix: an S**-decoder** over $\mathbb{Z}_m$: $h_t : \mathbb{Z}_m \to \mathbb{Z}_m$ and slopes j_t with

$$\sum_t h_t(x + j_t s) = \begin{cases} 1 & \text{if } x = s = 0, \\ 0 & \text{if } s \in S \setminus \{0\}, \\ \text{arbitrary} & \text{otherwise} \end{cases}$$

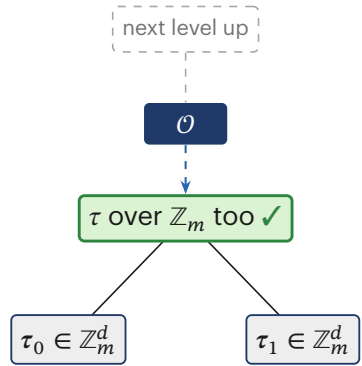


Technical Fine Print

- To recurse, the parent and child types need to match
- This is a serious issue: if $m = 0$ and $\gamma^m = 1$, no S_m^*-decoding polynomial exists
- Fix: an S**-decoder** over $\mathbb{Z}_m$: $h_t : \mathbb{Z}_m \to \mathbb{Z}_m$ and slopes j_t with

$$\sum_t h_t(x + j_t s) = \begin{cases} 1 & \text{if } x = s = 0, \\ 0 & \text{if } s \in S \setminus \{0\}, \\ \text{arbitrary} & \text{otherwise} \end{cases}$$

- Decoding polynomials implement this but with $h_t : \mathbb{Z}_m \to \mathbb{F}$



Tree Evaluation: Summary

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Tree Evaluation: Summary

Matching vectors
S-MVF of size 2^{ℓ} in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



One-level gadget
$q = O(r^2)$ calls
$s = O(\log r)$ bits

h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Tree Evaluation: Summary

Matching vectors
S-MVF of size 2^ℓ in $\mathbb{Z}_m^d$

Decoding polynomial
S-decoding, sparsity r



One-level gadget
$q = O(r^2)$ calls
$s = O(\log r)$ bits

Cook–Mertz compiler [CM24]

Tree evaluation

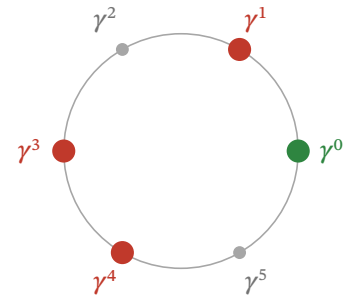
catalytic space	$O(d \log m)$
free space	$O(h \log r + \ell)$
time	$\text{poly}(r^h \cdot 2^\ell)$

h: tree height · ℓ: bits per value
r: sparsity · d: dimension of the vectors

Roadmap

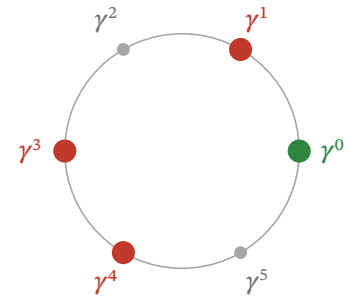
- What Is a Decoding Polynomial?
- Warm-Up Application to Private Information Retrieval [Efr09]
- New Application to Tree Evaluation [HP**R**26]
- ➜ New Construction [G**R**26]

Back to Decoding Polynomials



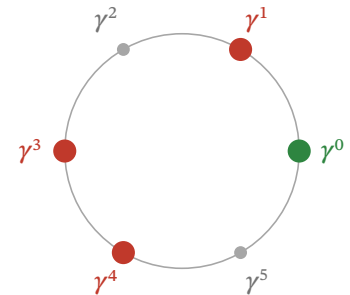
- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$

Back to Decoding Polynomials



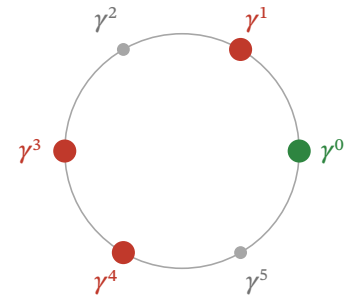
- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms ⇒ fewer servers in PIR and fewer recursive calls in tree evaluation

Back to Decoding Polynomials



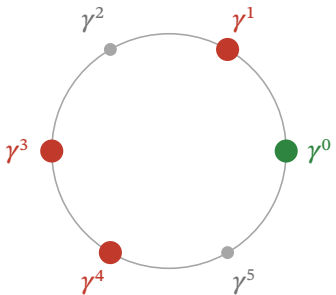
- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms $\Rightarrow$ fewer servers in PIR and fewer recursive calls in tree evaluation
- With k primes:

Back to Decoding Polynomials



- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms $\Rightarrow$ fewer servers in PIR and fewer recursive calls in tree evaluation
- With k primes:
  - 2^k terms is trivial

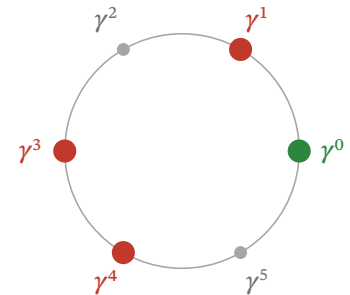
Back to Decoding Polynomials



- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms ⇒ fewer servers in PIR and fewer recursive calls in tree evaluation
- With k primes:
  - 2^k terms is trivial
  - $\approx 3^{k/2}$ is known [CFL+13] *

* Assuming number-theoretic conjectures

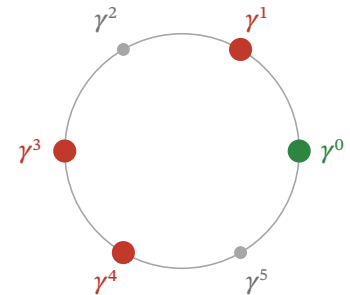
Back to Decoding Polynomials



- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms ⇒ fewer servers in PIR and fewer recursive calls in tree evaluation
- With k primes:
  - 2^k terms is trivial
  - $\approx 3^{k/2}$ is known [CFL+13] *
  - $k + 1$ is a lower bound [GK26]

* Assuming number-theoretic conjectures

Back to Decoding Polynomials



- Recall: $\gamma \in \mathbb{F}$ a primitive mth root of unity, and $P : \mathbb{F} \to \mathbb{F}$ with $P(1) = 1$ and $P(\gamma^s) = 0$ for every $s \in S_m^* \setminus \{0\}$
- Goal: fewer terms ⇒ fewer servers in PIR and fewer recursive calls in tree evaluation
- With k primes:
  - 2^k terms is trivial
  - $\approx 3^{k/2}$ is known [CFL+13] *
  - $k + 1$ is a lower bound [GK26]
- This part: a construction attaining the lower bound! *

* Assuming number-theoretic conjectures

The CRT Basis

- $m = p_1 \cdots p_k$: the CRT map $\varphi(x) = (x \bmod p_1, \ldots, x \bmod p_k)$ is a bijection $\mathbb{Z}_m \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_k}$

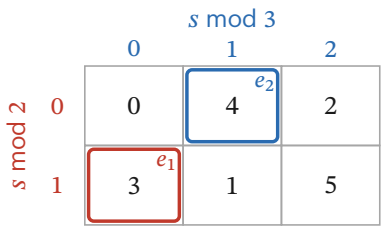
Example: $m = 6 = 2 \cdot 3$

$s \bmod 2$ \ $s \bmod 3$	0	1	2
0	0	4	2
1	3	1	5

The CRT Basis

- $m = p_1 \cdots p_k$: the CRT map $\varphi(x) = (x \bmod p_1, \ldots, x \bmod p_k)$ is a bijection $\mathbb{Z}_m \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_k}$
- CRT basis: $e_1 = \varphi^{-1}(1, 0, \ldots, 0)$, ..., $e_k = \varphi^{-1}(0, \ldots, 0, 1)$, so $x \equiv \sum_i (x \bmod p_i)\, e_i$

Example: $m = 6 = 2 \cdot 3$



The CRT Basis

- $m = p_1 \cdots p_k$: the CRT map $\varphi(x) = (x \bmod p_1, \ldots, x \bmod p_k)$ is a bijection $\mathbb{Z}_m \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_k}$
- CRT basis: $e_1 = \varphi^{-1}(1, 0, \ldots, 0)$, ..., $e_k = \varphi^{-1}(0, \ldots, 0, 1)$, so $x \equiv \sum_i (x \bmod p_i)\, e_i$
- $\gamma_i = \gamma^{e_i}$ has order p_i, and $\gamma^x = \prod_i \gamma_i^{x \bmod p_i}$

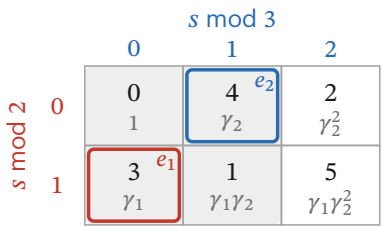
Example: $m = 6 = 2 \cdot 3$

$s \bmod 2$ \ $s \bmod 3$	0	1	2
0	0 1	4 e_2 γ_2	2 γ_2^2
1	3 e_1 γ_1	1 $\gamma_1\gamma_2$	5 $\gamma_1\gamma_2^2$

The CRT Basis

- $m = p_1 \cdots p_k$: the CRT map $\varphi(x) = (x \bmod p_1, \dots, x \bmod p_k)$ is a bijection $\mathbb{Z}_m \to \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_k}$
- CRT basis: $e_1 = \varphi^{-1}(1, 0, \dots, 0)$, ..., $e_k = \varphi^{-1}(0, \dots, 0, 1)$, so $x \equiv \sum_i (x \bmod p_i)\, e_i$
- $\gamma_i = \gamma^{e_i}$ has order p_i, and $\gamma^x = \prod_i \gamma_i^{x \bmod p_i}$
- On S_m^*, $\sigma_i = s \bmod p_i \in \{0, 1\}$: evaluating $\gamma^s = \prod_{\sigma_i = 1} \gamma_i$ is a **subset product**

Example: $m = 6 = 2 \cdot 3$



$S_6^* = \{0, 1, 3, 4\}$: the 0/1 corner

Recasting the Goal

Goal: an S_m^*-decoding polynomial with $k+1$ terms

- Want $P(X) = \sum_{j=1}^{r} c_j X^{b_j}$ with $P(\gamma^s) = \mathbb{1}[s = 0]$ on S_m^*

$m = p_1 \cdots p_k \;\cdot\; \sigma_i = s \bmod p_i \;\cdot\; \gamma^s = \prod_{\sigma_i = 1} \gamma_i$

Recasting the Goal

Goal: an S_m^*-decoding polynomial with $k+1$ terms

- Want $P(X) = \sum_{j=1}^{r} c_j X^{b_j}$ with $P(\gamma^s) = \mathbb{1}[s=0]$ on S_m^*
- $P(\gamma^s) = \sum_j c_j \prod_{\sigma_i=1} \gamma_i^{b_j}$

$m = p_1 \cdots p_k \cdot \sigma_i = s \bmod p_i \cdot \gamma^s = \prod_{\sigma_i=1} \gamma_i$

Recasting the Goal

Goal: an S_m^*-decoding polynomial with $k+1$ terms

- Want $P(X) = \sum_{j=1}^{r} c_j X^{b_j}$ with $P(\gamma^s) = \mathbb{1}[s=0]$ on S_m^*
- $P(\gamma^s) = \sum_j c_j \prod_{\sigma_i=1} \gamma_i^{b_j}$
- $\zeta_{ij} = \gamma_i^{b_j}$ only sees b_j mod p_i: by CRT, a free choice in μ_{p_i}, the p_ith roots of unity

$m = p_1 \cdots p_k \cdot \sigma_i = s \bmod p_i \cdot \gamma^s = \prod_{\sigma_i=1} \gamma_i$

Recasting the Goal

Goal: an S_m^*-decoding polynomial with $k+1$ terms

- Want $P(X) = \sum_{j=1}^{r} c_j X^{b_j}$ with $P(\gamma^s) = \mathbb{1}[s=0]$ on S_m^*
- $P(\gamma^s) = \sum_j c_j \prod_{\sigma_i=1} \gamma_i^{b_j}$
- $\zeta_{ij} = \gamma_i^{b_j}$ only sees $b_j \bmod p_i$: by CRT, a free choice in μ_{p_i}, the p_ith roots of unity

TLDR:
an S_m^*-decoding polynomial with r terms is a choice of $c_j \in \mathbb{F}$ and $\zeta_{ij} \in \mu_{p_i}$ with $\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$

$m = p_1 \cdots p_k \cdot \sigma_i = s \bmod p_i \cdot \gamma^s = \prod_{\sigma_i=1} \gamma_i$

New Object: Root-of-Unity Grids

Definition ($(\mathbb{F}, m)$-root-of-unity grid)

Nonzero $a_i \in \mathbb{F}$ and distinct nonzero $t_j \in \mathbb{F}$ $(i, j \leq k)$ with every $1 + a_i t_j \in \mu_{p_i}$.

μ_{p_i}: the p_ith roots of unity in $\mathbb{F}$

New Object: Root-of-Unity Grids

Definition ($(\mathbb{F}, m)$-root-of-unity grid)

Nonzero $a_i \in \mathbb{F}$ and distinct nonzero $t_j \in \mathbb{F}$ $(i, j \leq k)$ with every $1 + a_i t_j \in \mu_{p_i}$.

$$\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \end{array} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \quad \Big\} \text{ in } \mathbb{F}$$

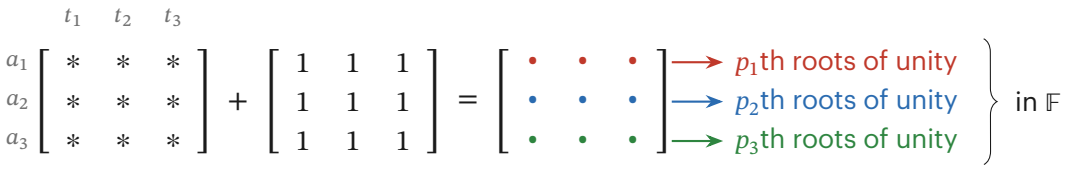
rank 1: $a_i t_j$ | all ones | root-of-unity grid

μ_{p_i}: the p_ith roots of unity in $\mathbb{F}$

New Object: Root-of-Unity Grids

Definition ($(\mathbb{F}, m)$-root-of-unity grid)

Nonzero $a_i \in \mathbb{F}$ and distinct nonzero $t_j \in \mathbb{F}$ $(i, j \leq k)$ with every $1 + a_i t_j \in \mu_{p_i}$.

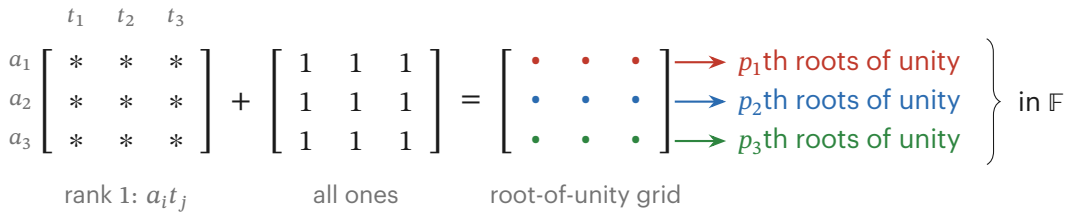


μ_{p_i}: the p_ith roots of unity in $\mathbb{F}$

New Object: Root-of-Unity Grids

Definition ($(\mathbb{F}, m)$-root-of-unity grid)

Nonzero $a_i \in \mathbb{F}$ and distinct nonzero $t_j \in \mathbb{F}$ $(i, j \leq k)$ with every $1 + a_i t_j \in \mu_{p_i}$.



Theorem [GR26]

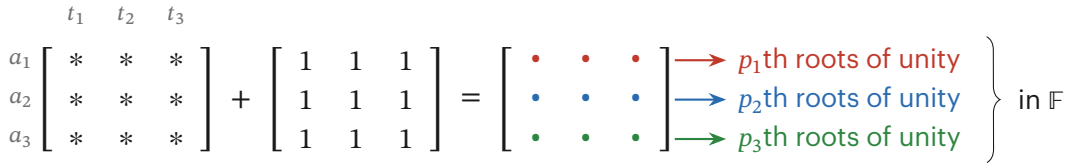
An S_m^*-decoding polynomial with $k+1$ terms exists over $\mathbb{F}$ if and only if an $(\mathbb{F}, m)$-root-of-unity grid does.

μ_{p_i}: the p_ith roots of unity in $\mathbb{F}$

New Object: Root-of-Unity Grids

Definition ($(\mathbb{F}, m)$-root-of-unity grid)

Nonzero $a_i \in \mathbb{F}$ and distinct nonzero $t_j \in \mathbb{F}$ $(i, j \le k)$ with every $1 + a_i t_j \in \mu_{p_i}$.



Theorem [GR26]

An S_m^*-decoding polynomial with $k+1$ terms exists over $\mathbb{F}$ if and only if an $(\mathbb{F}, m)$-root-of-unity grid does.

Proof: necessity ($\Rightarrow$) and sufficiency ($\Leftarrow$); then, do grids exist?

μ_{p_i}: the p_ith roots of unity in $\mathbb{F}$

Necessity of Root-of-Unity Grids

Warm-up: k terms are impossible [GK26]

- Recall: $\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$

	term 1	term 2	term 3
X_1	ζ_{11}	ζ_{12}	ζ_{13}
X_2	ζ_{21}	ζ_{22}	ζ_{23}
X_3	ζ_{31}	ζ_{32}	ζ_{33}

$P(X) = \sum_{j=1}^{r} c_j X^{b_j} \cdot \sigma_i = s \bmod p_i$

$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}$: p_ith roots of unity

Necessity of Root-of-Unity Grids

Warm-up: k terms are impossible [GK26]

- Recall: $\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$
- Introduce formal variables $X_1, \ldots, X_k$:

$$\sum_{j=1}^{r} c_j \prod_{i=1}^{k} (1 + \zeta_{ij} X_i) = 1$$

	term 1	term 2	term 3
X_1	ζ_{11}	ζ_{12}	ζ_{13}
X_2	ζ_{21}	ζ_{22}	ζ_{23}
X_3	ζ_{31}	ζ_{32}	ζ_{33}

$P(X) = \sum_{j=1}^{r} c_j X^{b_j} \cdot \sigma_i = s \bmod p_i$

$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}$: p_ith roots of unity

Necessity of Root-of-Unity Grids

Warm-up: k terms are impossible [GK26]

- Recall: $\boxed{\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]}$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$
- Introduce formal variables $X_1, \ldots, X_k$:

$$\sum_{j=1}^{r} c_j \prod_{i=1}^{k} (1 + \zeta_{ij} X_i) = 1$$

- Proof: expand and compare coefficients; that of $\prod_{\sigma_i=1} X_i$ gives the equation for $\boldsymbol{\sigma}$

	term 1	term 2	term 3
X_1	ζ_{11}	ζ_{12}	ζ_{13}
X_2	ζ_{21}	ζ_{22}	ζ_{23}
X_3	ζ_{31}	ζ_{32}	ζ_{33}

$$P(X) = \sum_{j=1}^{r} c_j X^{b_j} \cdot \sigma_i = s \bmod p_i$$

$$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}\text{: } p_i\text{th roots of unity}$$

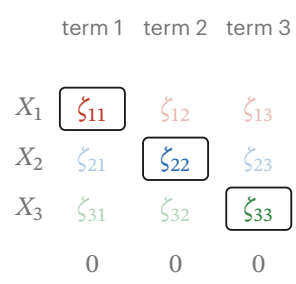
Necessity of Root-of-Unity Grids

Warm-up: k terms are impossible [GK26]

- Recall: $\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$
- Introduce formal variables $X_1, \ldots, X_k$:

$$\sum_{j=1}^{r} c_j \prod_{i=1}^{k} (1 + \zeta_{ij} X_i) = 1$$

- Proof: expand and compare coefficients; that of $\prod_{\sigma_i=1} X_i$ gives the equation for $\boldsymbol{\sigma}$
- If $r \le k$, get a contradiction via a diagonalisation trick: set $X_i = -\zeta_{ii}^{-1}$



every term is 0, but the sum is 1

$P(X) = \sum_{j=1}^{r} c_j X^{b_j} \cdot \sigma_i = s \bmod p_i$

$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}$: p_ith roots of unity

Necessity of Root-of-Unity Grids

$k+1$ terms force a root-of-unity grid

- With $k+1$ terms: the same diagonalisation trick kills k terms and leaves one

$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}$ · μ_p: pth roots of unity

Necessity of Root-of-Unity Grids

$k+1$ terms force a root-of-unity grid

- With $k+1$ terms: the same diagonalisation trick kills k terms and leaves one
- After some work, these force $\zeta_{i,k+1}/\zeta_{ij} = 1 + a_i t_j$, a p_ith root of unity: a root-of-unity grid!

$$\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \end{array} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \begin{array}{l} \longrightarrow p_1\text{th roots of unity} \\ \longrightarrow p_2\text{th roots of unity} \\ \longrightarrow p_3\text{th roots of unity} \end{array} \Big\} \text{ in } \mathbb{F}$$

rank 1: $a_i t_j$ — all ones — root-of-unity grid

$\zeta_{ij} = \gamma_i^{b_j} \in \mu_{p_i}$ · μ_p: pth roots of unity

Sufficiency of Root-of-Unity Grids

$$a_1 \cdots a_k \cdot \mathbb{1}[s = 0] = \mathrm{Coeff}_{X^k} \prod_{i=1}^{k} (1 + a_i X)^{1-\sigma_i}$$

degree k only when $\sigma = \mathbf{0}$

grid: $1 + a_i t_j \in \mu_{p_i}$, plus $t_0 = 0 \cdot \lambda_j = 1/\prod_{l \neq j}(t_j - t_l)$

Sufficiency of Root-of-Unity Grids

$$a_1 \cdots a_k \cdot \mathbb{1}[s = 0] = \mathrm{Coeff}_{X^k} \prod_{i=1}^{k} (1 + a_i X)^{1-\sigma_i}$$

degree k only when $\sigma = \mathbf{0}$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j)^{1-\sigma_i}$$

Lagrange at $t_0, t_1, \ldots, t_k$

grid: $1 + a_i t_j \in \mu_{p_i}$, plus $t_0 = 0 \cdot \lambda_j = 1/\prod_{l \neq j}(t_j - t_l)$

Sufficiency of Root-of-Unity Grids

$$a_1 \cdots a_k \cdot \mathbb{1}[s = 0] = \mathrm{Coeff}_{X^k} \prod_{i=1}^{k} (1 + a_i X)^{1-\sigma_i}$$ degree k only when $\sigma = \mathbf{0}$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j)^{1-\sigma_i}$$ Lagrange at $t_0, t_1, \ldots, t_k$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j) \cdot \prod_i \zeta_{ij}^{\sigma_i}$$ $\zeta_{ij} = (1 + a_i t_j)^{-1} \in \mu_{p_i}$

grid: $1 + a_i t_j \in \mu_{p_i}$, plus $t_0 = 0 \;\cdot\; \lambda_j = 1/\prod_{l \neq j}(t_j - t_l)$

Sufficiency of Root-of-Unity Grids

$$a_1 \cdots a_k \cdot \mathbb{1}[s = 0] = \mathrm{Coeff}_{X^k} \prod_{i=1}^{k} (1 + a_i X)^{1-\sigma_i}$$

degree k only when $\sigma = \mathbf{0}$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j)^{1-\sigma_i}$$

Lagrange at $t_0, t_1, \ldots, t_k$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j) \cdot \prod_i \zeta_{ij}^{\sigma_i}$$

$\zeta_{ij} = (1 + a_i t_j)^{-1} \in \mu_{p_i}$

$$= \sum_{j=0}^{k} c_j \, (\gamma^s)^{b_j}$$

$\gamma_i^{b_j} = \zeta_{ij}$ for all i, by CRT

grid: $1 + a_i t_j \in \mu_{p_i}$, plus $t_0 = 0 \;\cdot\; \lambda_j = 1/\prod_{l \neq j}(t_j - t_l)$

Sufficiency of Root-of-Unity Grids

$$a_1 \cdots a_k \cdot \mathbb{1}[s=0] = \mathrm{Coeff}_{X^k} \prod_{i=1}^{k} (1 + a_i X)^{1-\sigma_i}$$ degree k only when $\sigma = \mathbf{0}$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j)^{1-\sigma_i}$$ Lagrange at $t_0, t_1, \ldots, t_k$

$$= \sum_{j=0}^{k} \lambda_j \prod_i (1 + a_i t_j) \cdot \prod_i \zeta_{ij}^{\sigma_i}$$ $\zeta_{ij} = (1 + a_i t_j)^{-1} \in \mu_{p_i}$

$$= \sum_{j=0}^{k} c_j \, (\gamma^s)^{b_j}$$ $\gamma_i^{b_j} = \zeta_{ij}$ for all i, by CRT

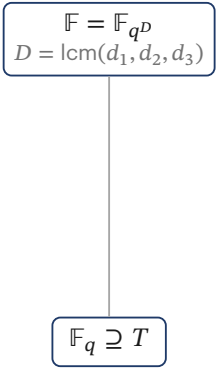
Recall:
an S_m^*-decoding polynomial with r terms is a choice of $c_j \in \mathbb{F}$ and $\zeta_{ij} \in \mu_{p_i}$ with $\sum_j c_j \prod_{\sigma_i=1} \zeta_{ij} = \mathbb{1}[\boldsymbol{\sigma} = \mathbf{0}]$ for every $\boldsymbol{\sigma} \in \{0,1\}^k$

grid: $1 + a_i t_j \in \mu_{p_i}$, plus $t_0 = 0 \cdot \lambda_j = 1/\prod_{l \neq j}(t_j - t_l)$

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \ldots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$

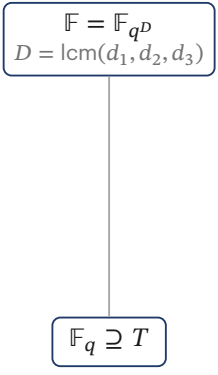


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \dots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$

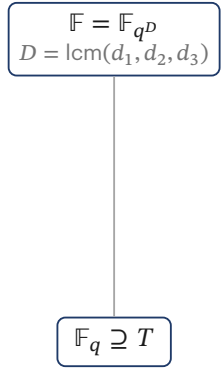


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \ldots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$
- $a_i \leftarrow \mathbb{F}$: success w.p. $\approx (p_i/|\mathbb{F}|)^k$, far too small

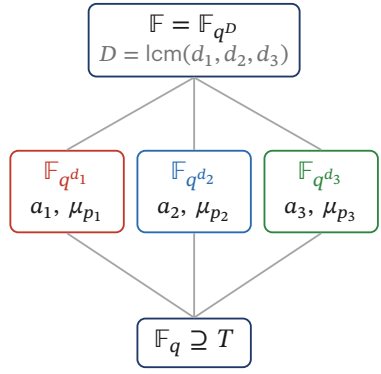


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \ldots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$
- $a_i \leftarrow \mathbb{F}$: success w.p. $\approx (p_i/|\mathbb{F}|)^k$, far too small

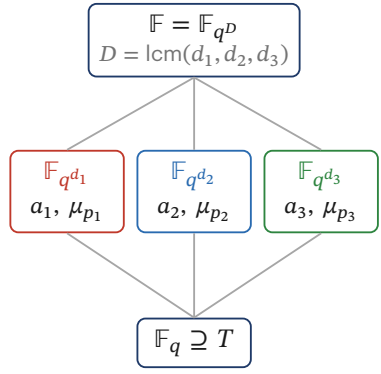


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \dots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$
- $a_i \leftarrow \mathbb{F}$: success w.p. $\approx (p_i/|\mathbb{F}|)^k$, far too small
- $a_i \leftarrow \mathbb{F}_{q^{d_i}} \supseteq \mu_{p_i}$: w.p. $\approx (p_i/q^{d_i})^k$

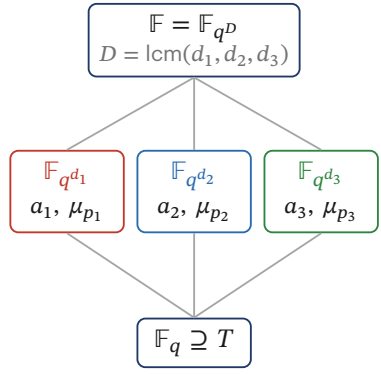


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \dots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$
- $a_i \leftarrow \mathbb{F}$: success w.p. $\approx (p_i/|\mathbb{F}|)^k$, far too small
- $a_i \leftarrow \mathbb{F}_{q^{d_i}} \supseteq \mu_{p_i}$: w.p. $\approx (p_i/q^{d_i})^k$
- If $p_i = (q^{d_i} - 1)/(q - 1)$ is prime: $\approx q^{-k} \gg q^{-d_i}$ once $d_i > k$ ✓

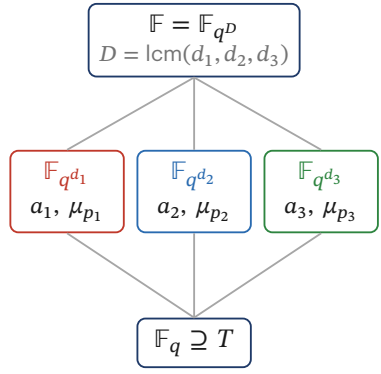


$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

Existence of Root-of-Unity Grids

Informal proof sketch via the probabilistic method

- Fix the t_j: $T = \{1, 2, \ldots, k\} \subseteq \mathbb{F}_q$, for a prime $q > k$
- The rows decouple: for each i, find a_i with $1 + a_i T \subseteq \mu_{p_i}$
- $a_i \leftarrow \mathbb{F}$: success w.p. $\approx (p_i/|\mathbb{F}|)^k$, far too small
- $a_i \leftarrow \mathbb{F}_{q^{d_i}} \supseteq \mu_{p_i}$: w.p. $\approx (p_i/q^{d_i})^k$
- If $p_i = (q^{d_i} - 1)/(q - 1)$ is prime: $\approx q^{-k} \gg q^{-d_i}$ once $d_i > k$ ✓
- Rigorous via Weil bounds



$m = p_1 \cdots p_k$ · μ_p: pth roots of unity

The Number-Theoretic Conjecture for Existence

Conjecture

For every prime q, there are infinitely many d such that $\frac{q^d-1}{q-1}$ is prime.

The Number-Theoretic Conjecture for Existence

Conjecture

For every prime q, there are infinitely many d such that $\frac{q^d-1}{q-1}$ is prime.

- Repunits: $\frac{q^d-1}{q-1} = 11\cdots1$ in base q; for example, $1093 = 1111111_3$ is prime

The Number-Theoretic Conjecture for Existence

Conjecture

For every prime q, there are infinitely many d such that $\frac{q^d-1}{q-1}$ is prime.

- Repunits: $\frac{q^d-1}{q-1} = 11\cdots1$ in base q; for example, $1093 = 1111111_3$ is prime

Theorem [GR26]

Assuming this conjecture, for every k there exist $m = p_1 \cdots p_k$ and $\mathbb{F}$ for which there is an $(\mathbb{F}, m)$-root-of-unity grid.

The Number-Theoretic Conjecture for Existence

Conjecture

For every prime q, there are infinitely many d such that $\frac{q^d-1}{q-1}$ is prime.

- Repunits: $\frac{q^d-1}{q-1} = 11\cdots1$ in base q; for example, $1093 = 1111111_3$ is prime

Theorem [GR26]

Assuming this conjecture, for every k there exist $m = p_1 \cdots p_k$ and $\mathbb{F}$ for which there is an $(\mathbb{F}, m)$-root-of-unity grid.

Corollary. For every constant $s \geq 2$, there is an s-server PIR with communication $\exp(\tilde{O}((\log n)^{1/s}))$.

PIR with Superconstant Servers

Goal: let the number of servers s grow with n

- So far, $s = O(1)$ servers: communication $\exp(\tilde{O}((\log n)^{1/s}))$

PIR with Superconstant Servers

Goal: let the number of servers s grow with n

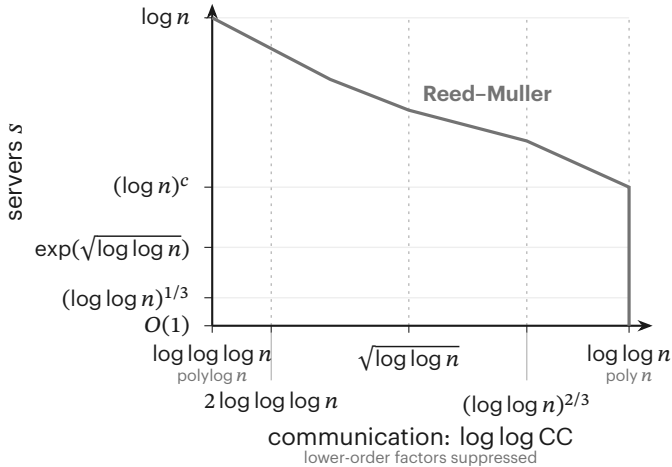
- So far, $s = O(1)$ servers: communication $\exp(\tilde{O}((\log n)^{1/s}))$
- The $\tilde{O}$ hides a constant factor that grows if the number of servers is allowed to grow

PIR with Superconstant Servers

Goal: let the number of servers s grow with n

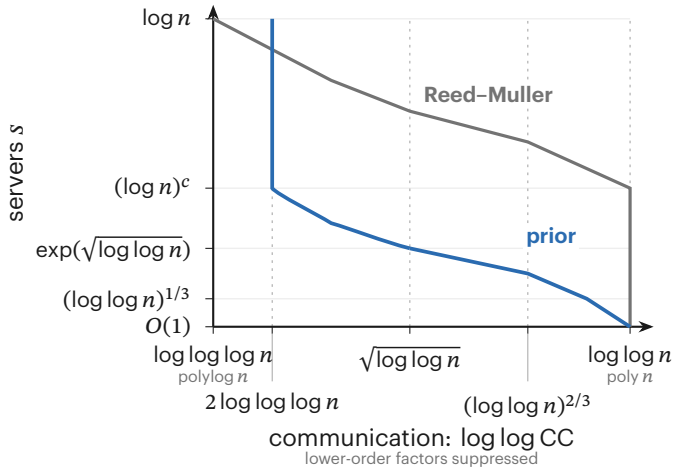
- So far, $s = O(1)$ servers: communication $\exp(\tilde{O}((\log n)^{1/s}))$
- The $\tilde{O}$ hides a constant factor that grows if the number of servers is allowed to grow
- So our exponential reduction in server count need not persist in the superconstant-server regime

Superconstant Servers: Phase Diagram



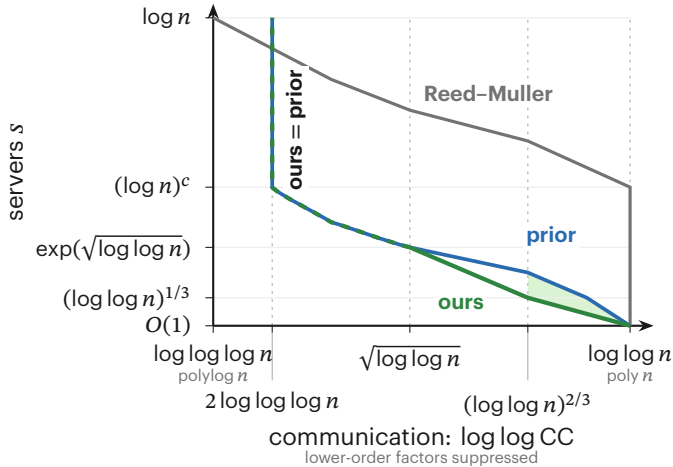
- Reed–Muller: polylog(n) communication with $\log n$ servers

Superconstant Servers: Phase Diagram



- Reed–Muller: polylog(n) communication with log n servers
- Matching vectors: better with few servers, but always log log CC $\gtrsim 2 \log \log \log n$

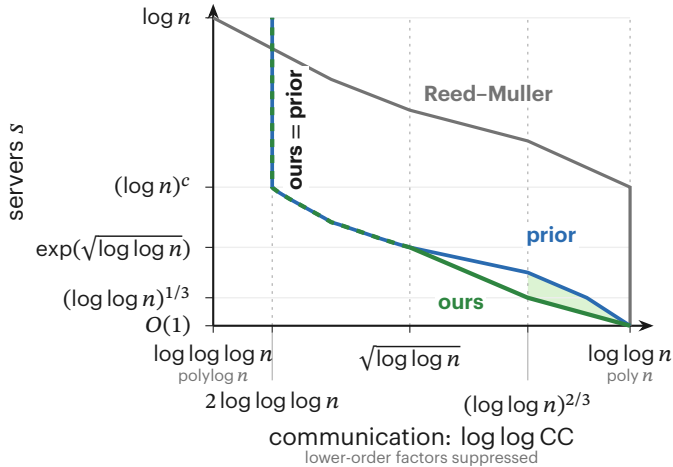
Superconstant Servers: Phase Diagram



- Reed–Muller: polylog(n) communication with $\log n$ servers
- Matching vectors: better with few servers, but always $\log\log \mathrm{CC} \gtrsim 2\log\log\log n$
- Ours, few servers: exponentially fewer servers

"Ours" [GR26] assumes a stronger conjecture on repunits

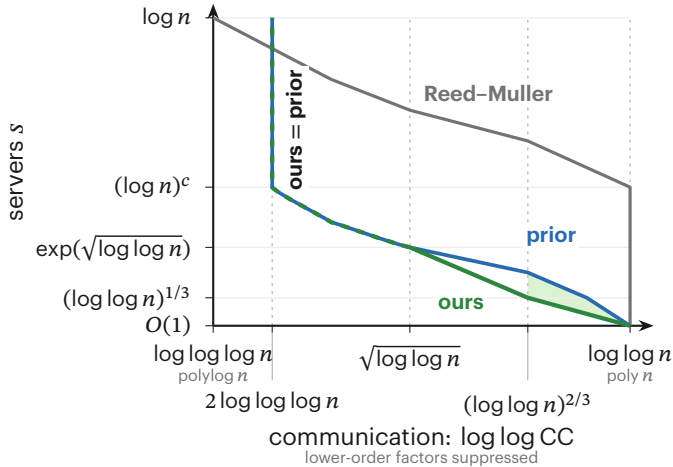
Superconstant Servers: Phase Diagram



- Reed–Muller: polylog(n) communication with $\log n$ servers
- Matching vectors: better with few servers, but always $\log\log \mathrm{CC} \gtrsim 2\log\log\log n$
- Ours, few servers: exponentially fewer servers
- Ours, up to $\approx \exp(\sqrt{\log\log n})$ servers: superpolynomially less communication

"Ours" [GR26] assumes a stronger conjecture on repunits

Superconstant Servers: Phase Diagram



- Reed–Muller: polylog(n) communication with $\log n$ servers
- Matching vectors: better with few servers, but always $\log\log \text{CC} \gtrsim 2\log\log\log n$
- Ours, few servers: exponentially fewer servers
- Ours, up to $\approx \exp(\sqrt{\log\log n})$ servers: superpolynomially less communication
- Even more servers: no gain over classic MV PIR

"Ours" [GR26] assumes a stronger conjecture on repunits

New Construction: Summary

Recast
by CRT, γ^s is
a subset product

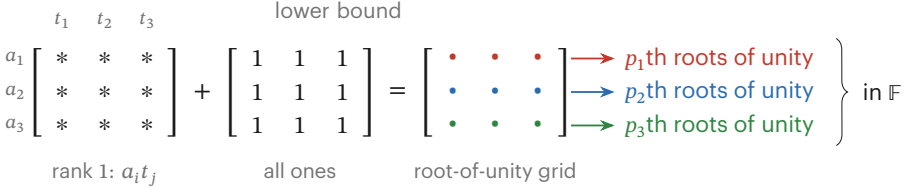
New Construction: Summary

Recast
by CRT, γ^s is a subset product

→

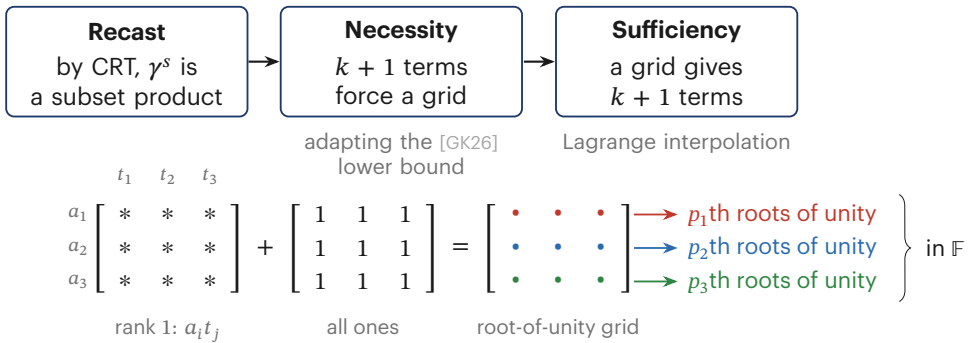
Necessity
$k+1$ terms force a grid

adapting the [GK26] lower bound

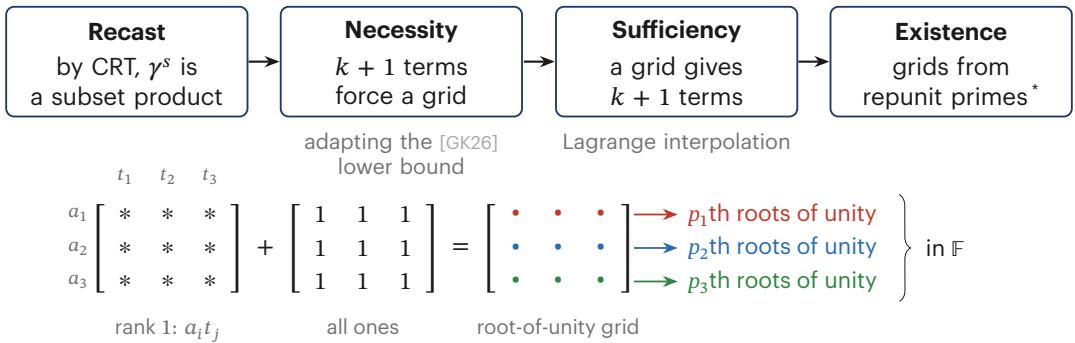


rank 1: $a_i t_j$ | all ones | root-of-unity grid

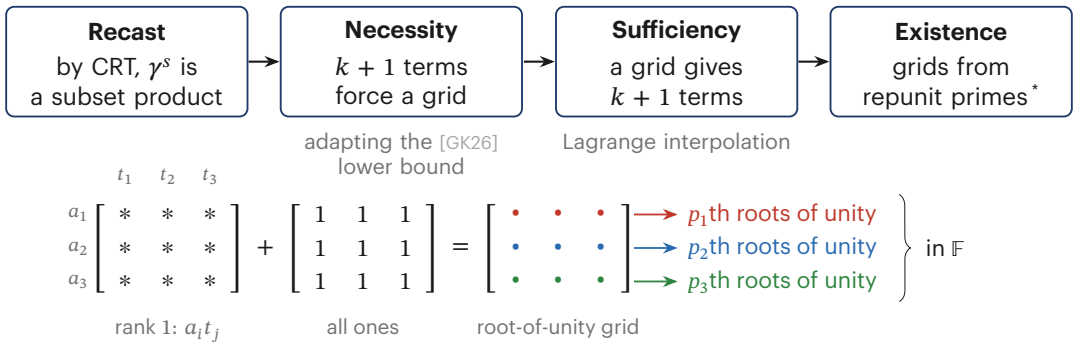
New Construction: Summary



New Construction: Summary



New Construction: Summary



Takeaway: sparsity $k+1$ is optimal, and attainable![*]

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$ *, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$*, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$
- Tree evaluation [HPR26]: $O(\log n)$ free space, polynomial time, catalytic $2^{\tilde{O}(\log^{1/k} n)}$

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$*, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$
- Tree evaluation [HPR26]: $O(\log n)$ free space, polynomial time, catalytic $2^{\tilde{O}(\log^{1/k} n)}$
- We believe $(k+1)$-term polynomials also improve tree evaluation (not yet checked)

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$*, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$
- Tree evaluation [HPR26]: $O(\log n)$ free space, polynomial time, catalytic $2^{\tilde{O}(\log^{1/k} n)}$
- We believe $(k+1)$-term polynomials also improve tree evaluation (not yet checked)

Open questions:

- $(k+1)$-term decoding polynomials without a conjecture?

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$*, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$
- Tree evaluation [HPR26]: $O(\log n)$ free space, polynomial time, catalytic $2^{\tilde{O}(\log^{1/k} n)}$
- We believe $(k+1)$-term polynomials also improve tree evaluation (not yet checked)

Open questions:

- $(k+1)$-term decoding polynomials without a conjecture?
- Other sources of root-of-unity grids, to help with many servers?

* Assuming a conjecture on repunits

Summary and Open Questions

- Decoding polynomials: sparsity is the cost in both applications
- PIR [GR26]: optimal sparsity $k+1$*, so s servers with $\exp(\tilde{O}((\log n)^{1/s}))$
- Tree evaluation [HPR26]: $O(\log n)$ free space, polynomial time, catalytic $2^{\tilde{O}(\log^{1/k} n)}$
- We believe $(k+1)$-term polynomials also improve tree evaluation (not yet checked)

Open questions:

- $(k+1)$-term decoding polynomials without a conjecture?
- Other sources of root-of-unity grids, to help with many servers?
- Tree evaluation in $O(\log n)$ total space?

* Assuming a conjecture on repunits

Thank You! Questions?

Matching vectors

Decoding polynomials



PIR

Tree evaluation

GR26 HPR26

Backup: Time-Space Tradeoffs [Wil25]

- Any time-T computation reduces to a tree evaluation instance with $h, \ell = O(\sqrt{T})$, so $n = 2^{O(\sqrt{T})}$

Backup: Time-Space Tradeoffs [Wil25]

- Any time-T computation reduces to a tree evaluation instance with $h, \ell = O(\sqrt{T})$, so $n = 2^{O(\sqrt{T})}$
- World 1: a cleverer tree evaluation algorithm uses $O(\log n)$ space
  - Then any time-T computation runs in space $O(\sqrt{T})$

Backup: Time-Space Tradeoffs [Wil25]

- Any time-T computation reduces to a tree evaluation instance with $h, \ell = O(\sqrt{T})$, so $n = 2^{O(\sqrt{T})}$
- World 1: a cleverer tree evaluation algorithm uses $O(\log n)$ space
  - Then any time-T computation runs in space $O(\sqrt{T})$
- World 2: no such algorithm exists
  - Then tree evaluation is in P but not L, so L $\subsetneq$ P
  - P: polynomial time; L: space $O(\log n)$